CSC 262 Lab: Gaussian Pyramids

Table of Contents

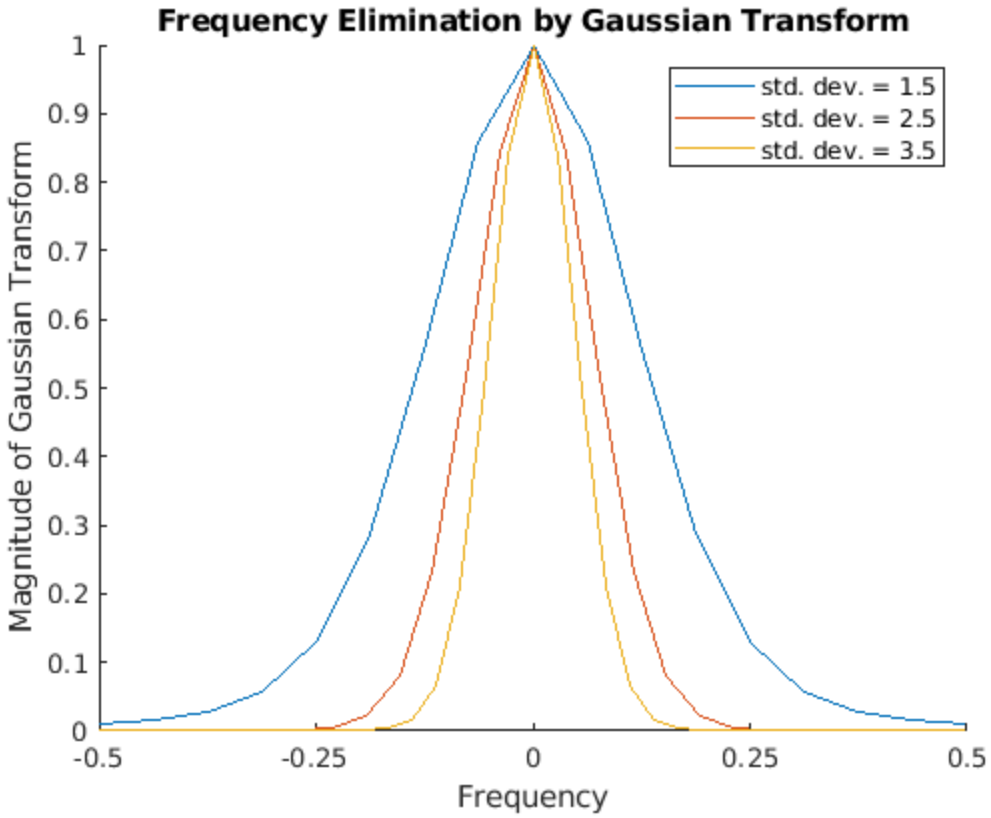
Overview

In this lab, we examine a method of multi-scale search using a Gaussian pyramid--which is a collection of images of increasing coarse scale. We write a function to generate our own pyramid that minimizes the effect of aliasing by convolving with the Gaussian before down-sampling. From an image of a sunflower field, we exercise this function to create a particular pyramid, on which we perform a search (for individual sunflowers) using the Laplacian of the Gaussian (LoG). Then, we consider how effective this detection process is as scale varies. We hope to develop familiarity with creating Gaussian Pyramids and learn a new approach to detecting regions of interest in a picture.

A. Gaussian Pyramids

First, we implement a function, gausspyr, which constructs a Gaussian pyramid based on a given image and with a specified number of levels. Each level of the pyramid (besides the first which contains the original image) results from convolving the previous with a Gaussian kernel and then downsampling. We convolve to avoid aliasing when we create smaller images.

For this reason, it is important to determine the correct standard deviation for the Gaussian kernel--one that accords with Nyquist's Theorem. Nyquist's Theorem states that accurate reconstruction of an image is possible when the sampling rate is twice the highest frequency in the image. The highest possible frequency is 0.5 cycles per pixel (in the case when pixel brightness alternates between fully bright and fully dark every pixel). This is also the sampling rate we use to generate a new level in the pyramid (we pick every other pixel row and column). To avoid aliasing, then, we need the highest frequency in the image after convolution to be half of the sampling rate: 0.25 cycles per pixel. How to find the standard deviation to eliminate all frequencies larger than 0.25 cycles/pixel? We visualize the magnitude of the Fourier Transformed Gaussian at a variety of standard deviations (see figure below). We observe what frequencies are "zero-ed" out by this particular Gaussian; that is, we looked for the highest frequency which corresponds to a non-zero magnitude. We found that a standard deviation of 2.5 best fits our needs. It creates a low pass Gaussian filter which eliminates all frequencies higher than 0.25 cycles/pixel.

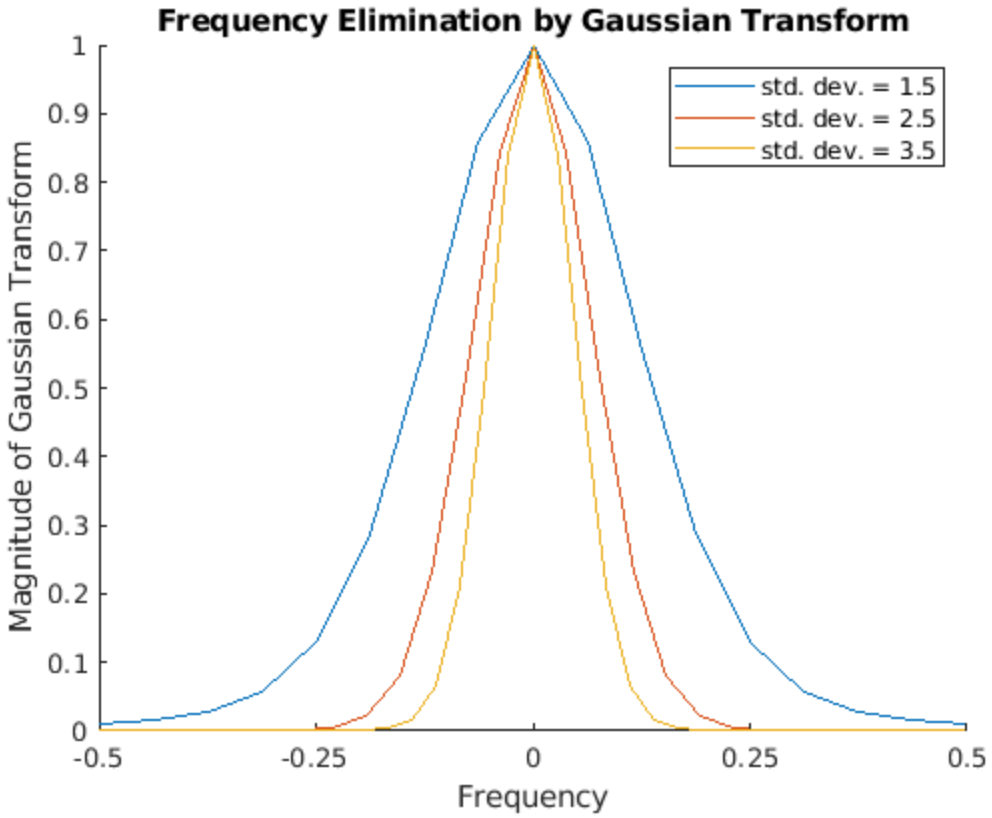


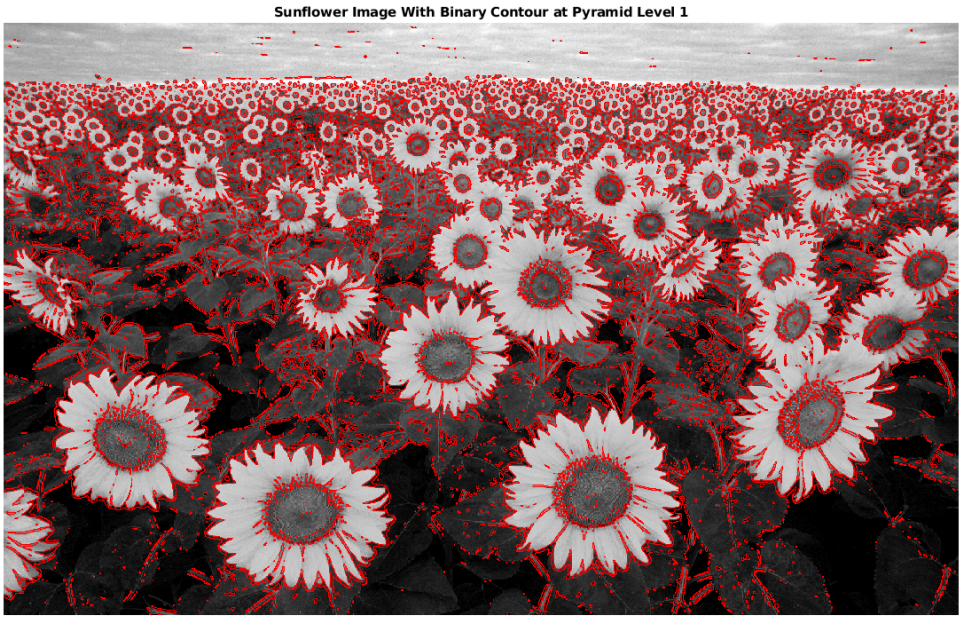
Armed with this standard deviation (2.5), we complete the remainder of our Gaussian pyramid constructing function in accordance with Algorithm 8.1 from the Forsyth and Ponce textbook.

Multi-Scale Search

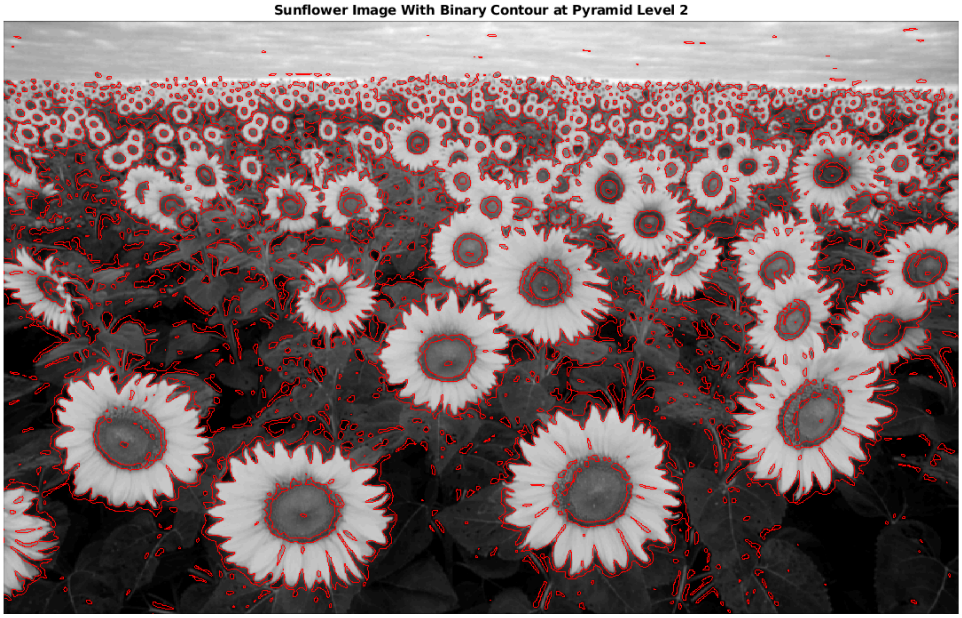
Previously, we were concerned with the initial task of Gaussian Pyramid construction. Now, we are ready to leverage the function we created as well as a LoG kernel to search for sunflowers in the target image at multiple scales. The LoG operator is useful here because it acts as a "blob detector"--meaning that it can find the flat face of a sunflower--and it is separable--meaning it can be used efficiently.

Here is how we approached multi-scale search. First, we use our gausspyr function to create a Gaussian pyramid of the sunflower image. Note that we created a pyramid with 7 levels because we thought such a pyramid would contain enough levels to render (at the coarsest scale) the larger sunflowers sufficiently small to be detected by the LoG operator. Next, we apply the LoG operator on each level of the pyramid, storing the result in a new structure (also a pyramid). The LoG image will show detected images but we only want to detect sunflowers so we impose a threshold to create a binary image (where, hopefully, white regions will cover all the sunflowers and nothing more). To check the accuracy of our method, we draw the contours of this result (the binary image derived from applying LoG) on top of the original image at each scale in the pyramid. The sequence of images below shows the output of this process (after thresholds were manually adjusted to 0.002).

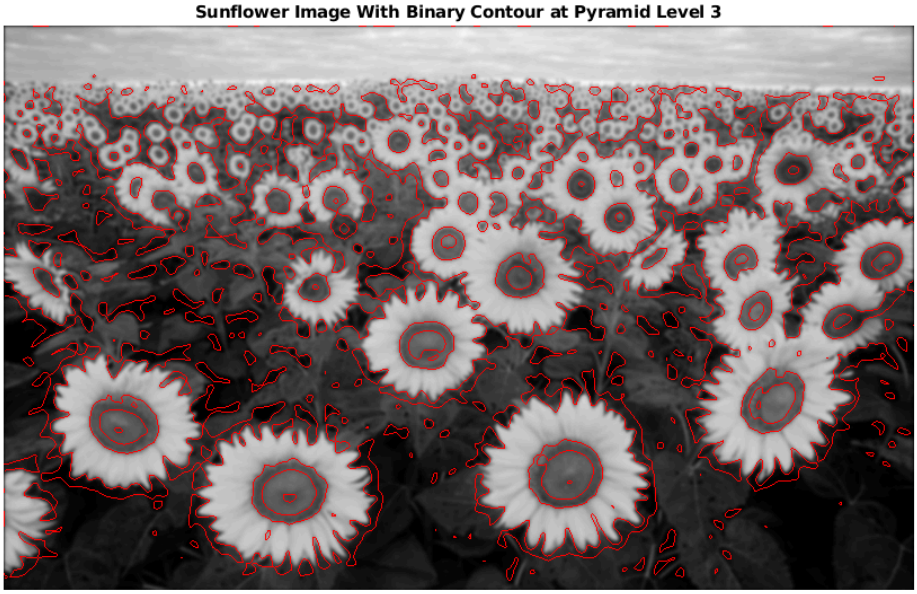

Frequency Elimination by Gaussian Transform
std. dev. = 1.5
std. dev. = 2.5
std. dev. = 3.5
Magnitude of Gaussian Transform
Frequency
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.5
-0.25
0
0.25
0.5


Sunflower Image With Binary Contour at Pyramid Level 1

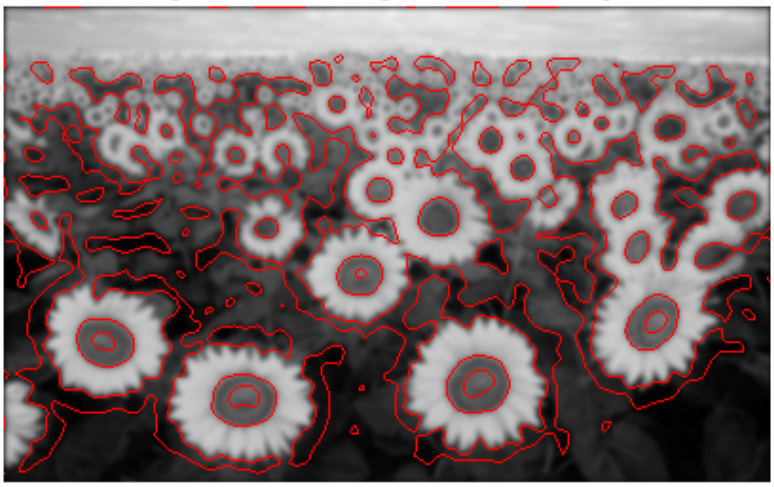
Sunflower Image With Binary Contour at Pyramid Level 2



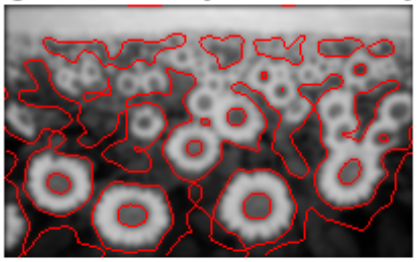
Sunflower Image With Binary Contour at Pyramid Level 3



Sunflower Image With Binary Contour at Pyramid Level 4



Sunflower Image With Binary Contour at Pyramid Level 5



wer Image With Binary Contour at Pyramid L

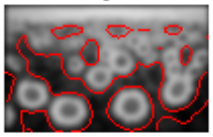
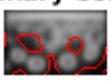


Image With Binary Contour at Pyran



At coarser scales, our method detects prominent sunflowers in the foreground with a respectable degree of accuracy. That is, it detects the faces of the flowers hoistically as opposed to images at finer scales which detect many details (unnecessary to our purposes) of the stems and small intricacies of the flower petals. The image at the coarsest scale is an exception because it appears that the lines do not capture any individual flower in particular. At fine scales, the method is very reliable in finding individual sunflowers; it detects flowers both in the foreground and the background. However, the detection plots at fine scales are riddled with false positives. Notice in the image with finest scale (that is, the first one in the sequence), red contours are painted all over the leaves, stems, and other foliage which surrounds the sunflowers. Even some regions in the sky are "detected". As scale becomes coarser, however, the false positive rate decreases. This poses a tradeoff where detection possess a lower false positive rate as scale becomes more coarse, but at the same time fewer sunflowers (especially less prominent ones such as those in the background) are detected.

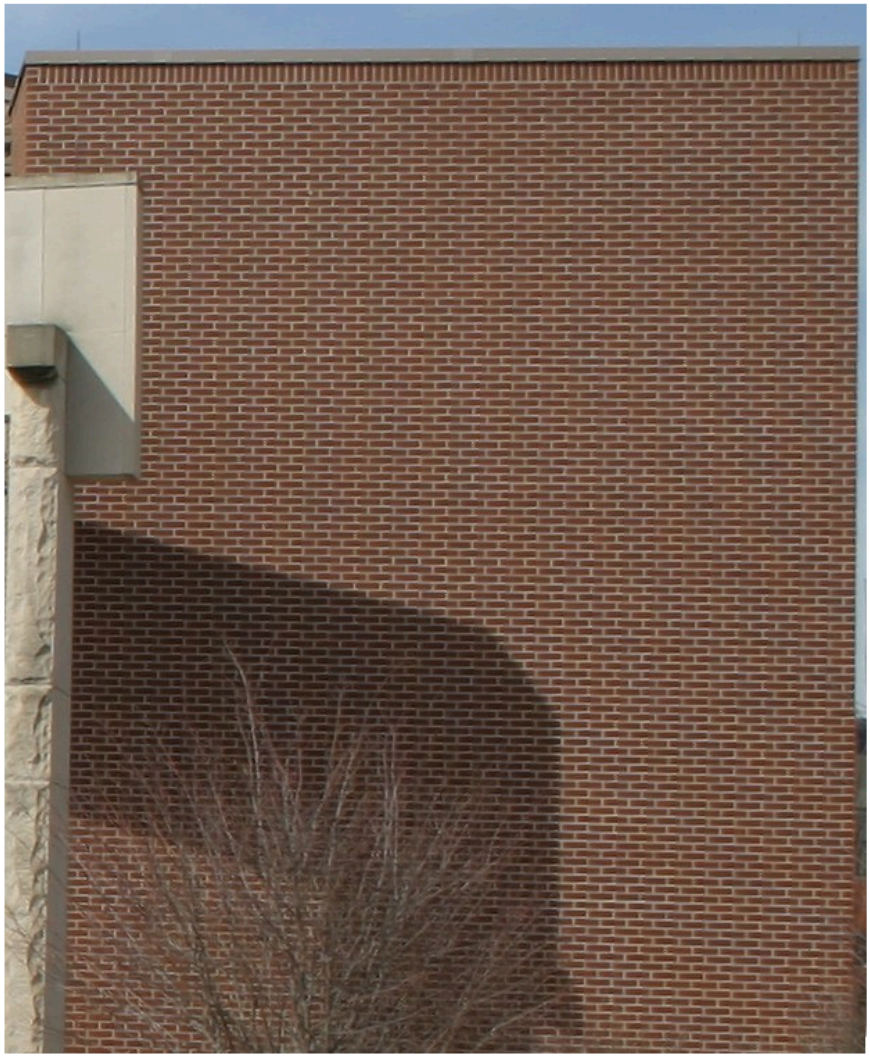
Conclusion

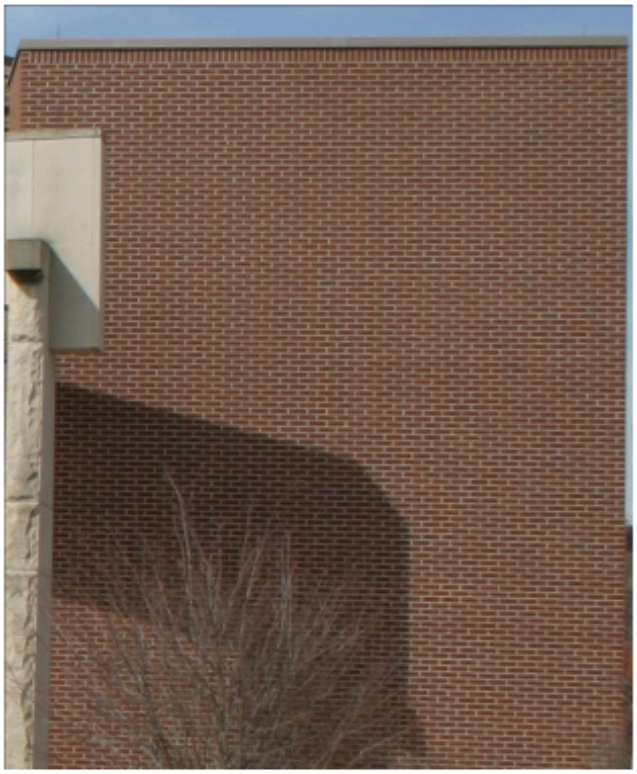
Over the course of these exercises, we implemented a function to construct a Gaussian Pyramid of specified size based on a given image. We learned how functions are written and used in matlab and we gained first hand experience for creating and handling image pyramids. We then used this knowledge to investigate multi-scale image search. In particular, we used a Laplacian of the Gaussian (LoG) operator to detect sunflowers in an image of a agricultural landscape. We discovered that our method detects sunflowers with pretty good reliability. At coarse scales, the method detects prominent sunflowers (especially in the foreground) with a low false positive rate. Also, the flower in general is encircled rather than specific details on the flower. However, many flowers (especially in the background) were not detected. At fine scales, the opposite seemed to be true: many flowers in both the foreground and background were detected but with a high false positive rate and with the detection of several unnecessary features (like small changes in petal outline).

Blurring, Subsampling, and Aliasing (Extra Credit)

As an additional exercise, we apply several separable decimation or blurring kernels from Table 3.4 of the Szeliski textbook. The results of convolving and subsampling with these kernels on the bricks image are found in the sequence of images below. In order, we display the result from use of linear, binomial, cubic (a = -1), cubic (a = -0.5), Windowed sinc, QMF-9, JPEG 2000 (the filters are numbered according to the order of their appearance in this list).

Subsampled Image with Filter #1



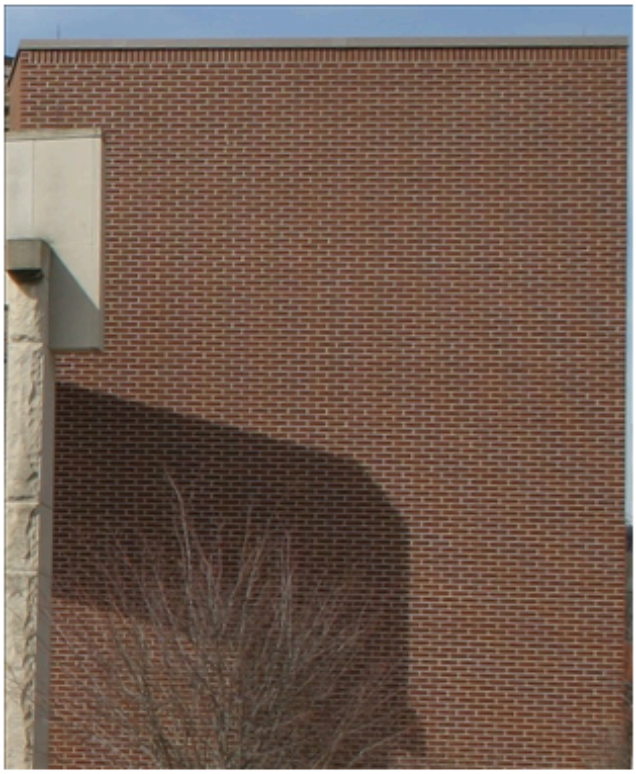


sub-image (Filter #1)

Subsampled Image with Filter #2

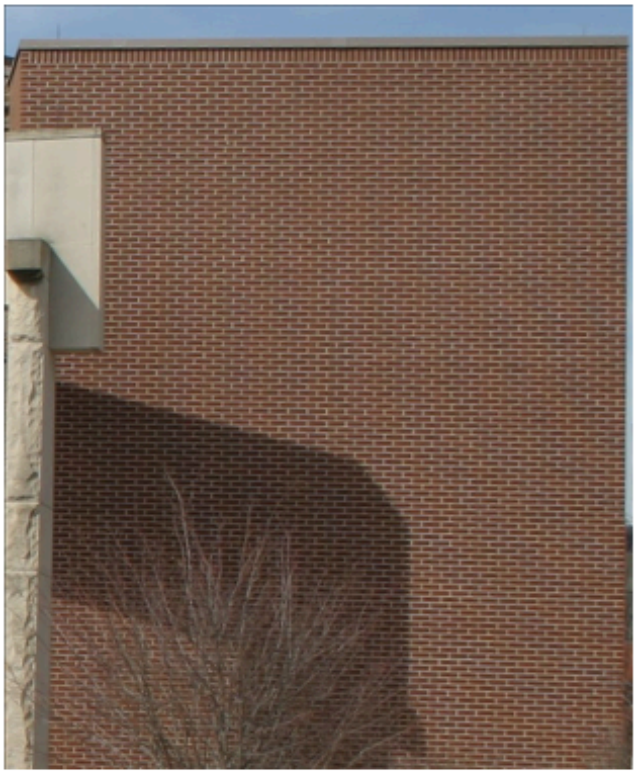
sub-image (Filter #2)

Subsampled Image with Filter #3



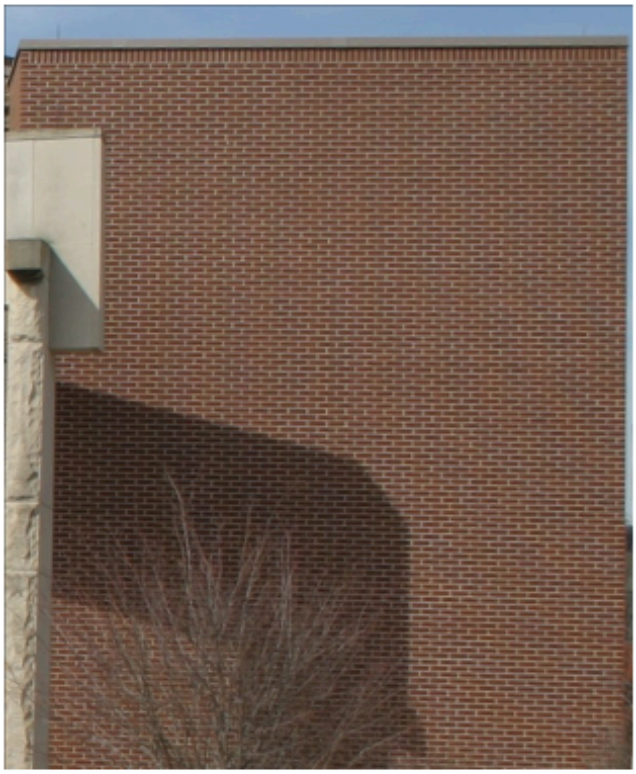
sub-image (Filter #3)

Subsampled Image with Filter #4



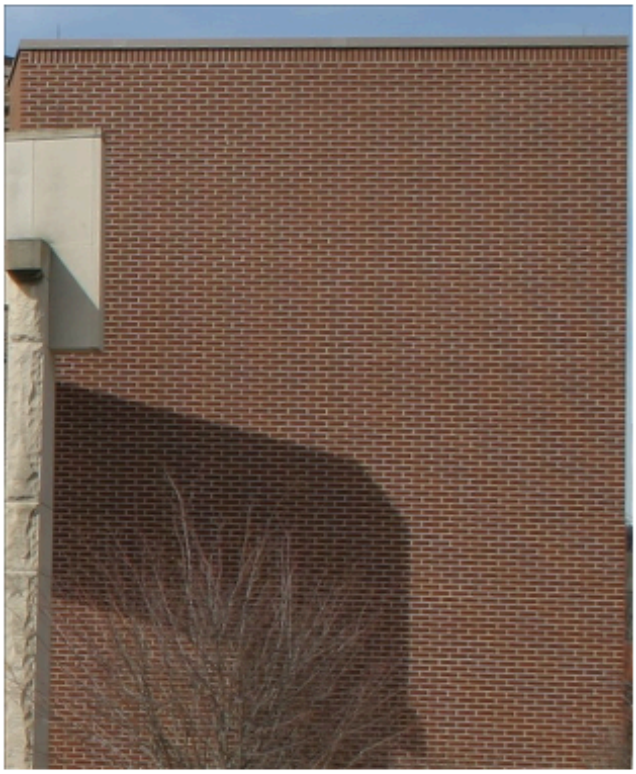
sub-image (Filter #4)

Subsampled Image with Filter #5



sub-image (Filter #5)

Subsampled Image with Filter #6



sub-image (Filter #6)

By our eyes, differences between the results of the various filters are hard to notice.

Acknowledgement

We informed our answers based on the code and text of the Image Pyramids lab for CSC-262 provided by Professor Jerod Weinman. We also benefited from class Q&A as well as the class Piazza discussion board. The sunflowers image is by Bruce Fritz, taken on behalf of the USDA Agricultural Research Service, and is therefore in the public domain. The brick image is by Colin M.L. Burnett and is released under the GNU Free Documentation License. Finally, we used excerpts from the Szeliski and Forsyth & Ponce textbooks. Szeliski Textbook: Sections 3.5-3.5.4, Richard Szeliski, Computer Vision: Algorithms and Applications, Electronic Draft, 2010. Forsyth & Ponce textbook: Sections 8.5-8.5.2, David Forsyth and Jean Ponce, Computer Vision A Modern Approach, Prentice Hall, 2003.

Published with MATLAB® R2018b