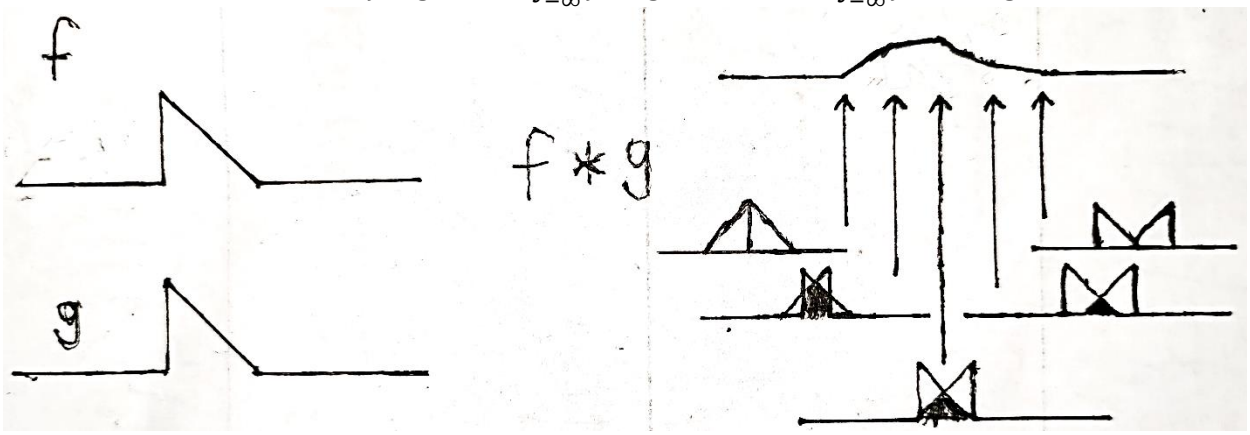
1.

The basic meaning of “convolution”: Shows how the shape of two functions are modified by each other. In other word, the result that the two functions “fused” in shape.

Math equation definition: $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$




2.

The basic meaning of “Fourier Transform”: Applying Fourier Transform on a signal f in time domain gives a function in frequency domain that describe the proportion of all the frequency components in f.

Math equation definition: $\hat{g}(f) = \int_{-\infty}^{\infty} g(t)e^{-2\pi itf}dt$

3.

How: Take Fourier Transform of the two signals, multiply the two Fourier Transform results, then take the inverse Fourier Transform of the product as the convolution of the two signals.

Why: This method is much faster than the original method that follows the original definition.

4.

The basic meaning of “cross-correlation”: Shows how the two functions “synchronize” (being similar and aligned) when they are shifting relatively in spatial domain.

Math equation definition: $(f \otimes g)(t) = \int_{-\infty}^{\infty} \overline{f(\tau)}g(\tau+t)d\tau = \int_{-\infty}^{\infty} \overline{f(\tau-t)}g(\tau)d\tau$

5.

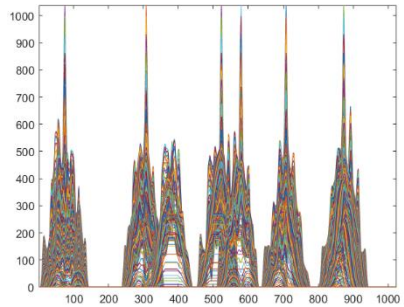
How: Take Fourier Transform of the two signals, perform conjugation on one of the two Fourier Transform results (depending on which correlation result you want of possible two, they are reverse of each other in spatial domain), then multiply them and take the inverse Fourier Transform of the product as the correlation of the two signals.

Why: Same with FFT convolution. Both FFT convolution and FFT correlation are for performance reasons.

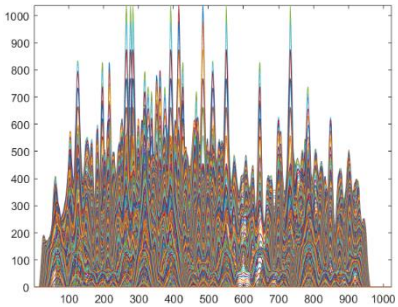
6.

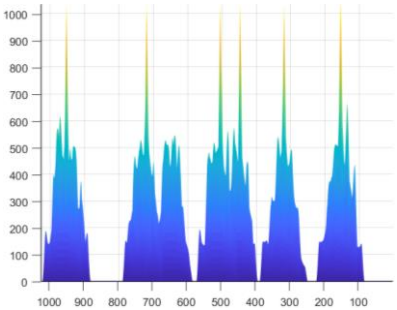
(a) About 5～10 seconds, depending on how “busy” my computer currently is.

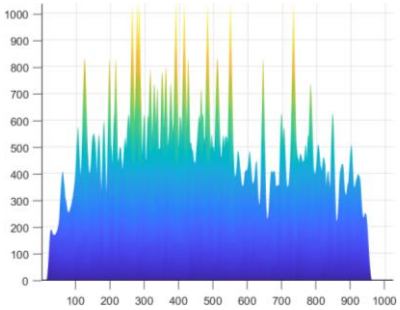
(b) 2D from left



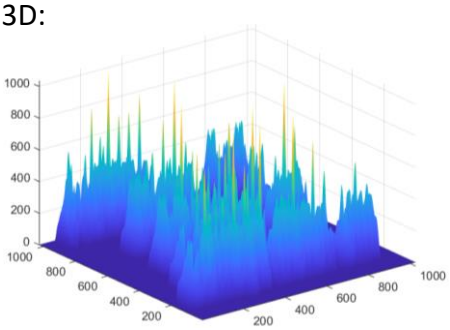
2D from front



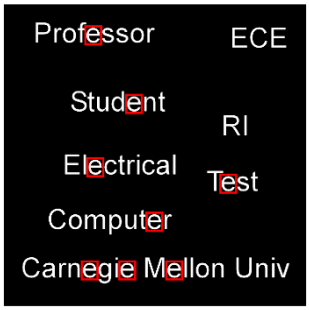




3D:

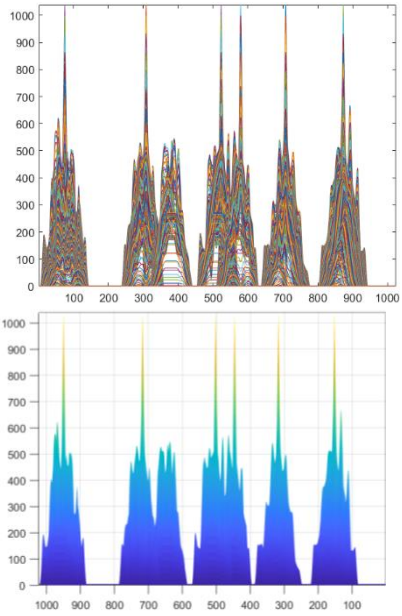


(c) Threshold: 1000

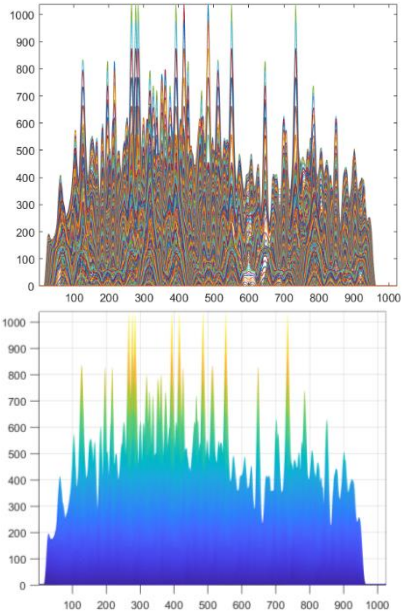



(d) Time: about 0.05 seconds

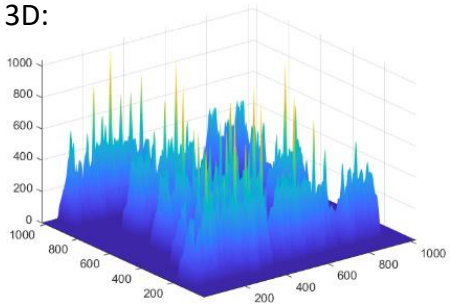
2D from left



2D from front



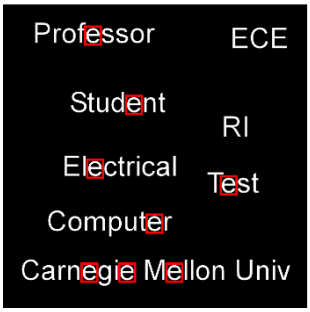
3D:







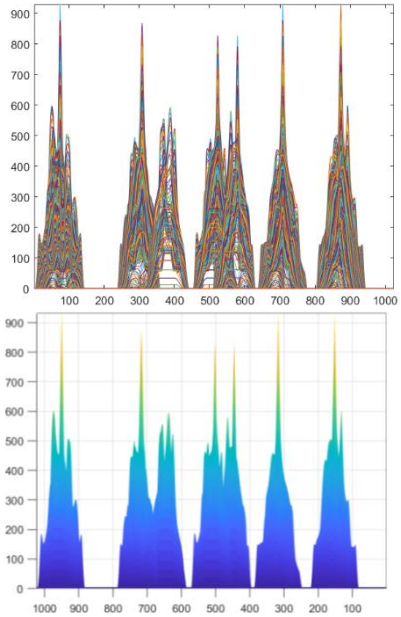
Threshold: 1000



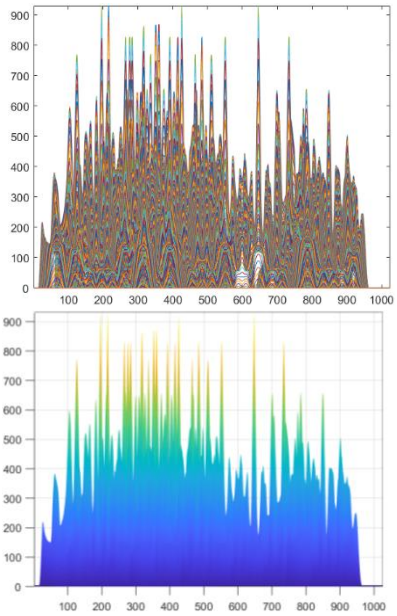

As shown, the results of the two method are almost identical, with the only difference being the speed.

(e) Time: about 0.05 seconds

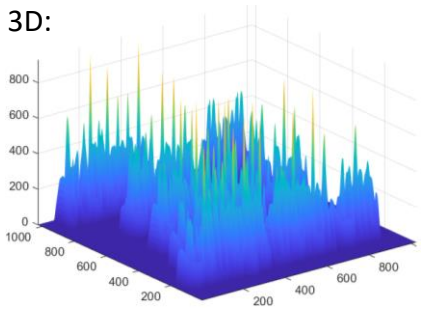
2D from left



2D from front



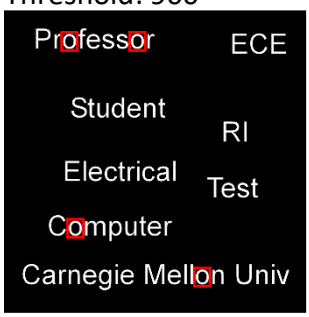
3D:







Threshold: 900



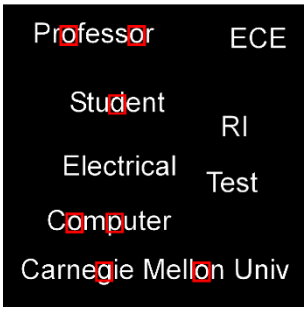

The main difference being the threshold we choose. At the beginning I used the same threshold as for (d) which is 1000, finding that nothing was detected. A quick observation tells me that the highest peak being about 900, not above 1000 as in (d).

Why? Note that letter ‘e’ has a “bar” at the middle compared to letter ‘o’, with only a bit notch which letter ‘o’ doesn’t have. As we know that white is ‘1’ and black is ‘0’, we may say that ‘e’ is more “weight” than ‘o’, and this is what leads to a difference in the product result for which the target letter aligned on the scene.

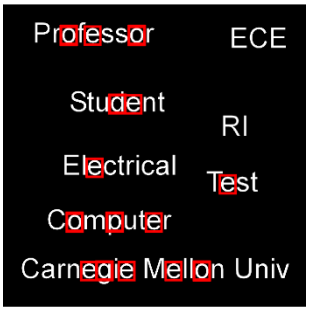
This comes with an idea about “relative peak”: When we are trying to find peaks for correlations, we should focus on how the peak being relative with other locations, rather than its actual value.

Also, since I hit the perfect threshold straightly, I’m still curious about what’s the “fewest other letters” mentioned in the problems document, here are my experiments:

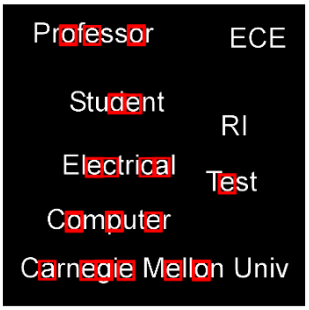
Threshold: 850




Threshold: 800




Threshold: 750

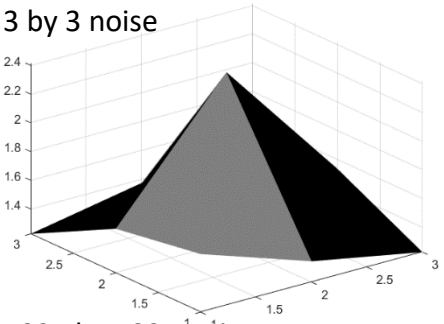



As the threshold chosen lower, some letters that’s not ‘o’ are detected. These are all letters that have “onion shapes” which is similar to ‘o’, so it makes sense.

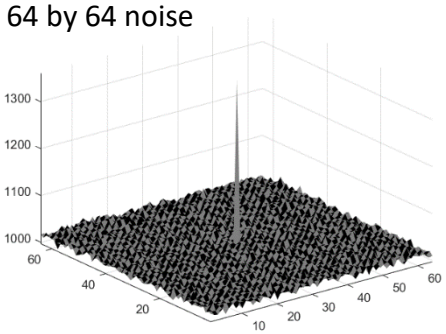
Also, I noticed that the drawing line of some detection boxes are becoming “thicker”, especially those indicating letter ‘o’. Since the values around the peak locations are supposed to also be high, over-sensitive threshold will not only lead to false-positive detections, but also results in “multi-detections” for peaks that are detected with higher threshold.

7.

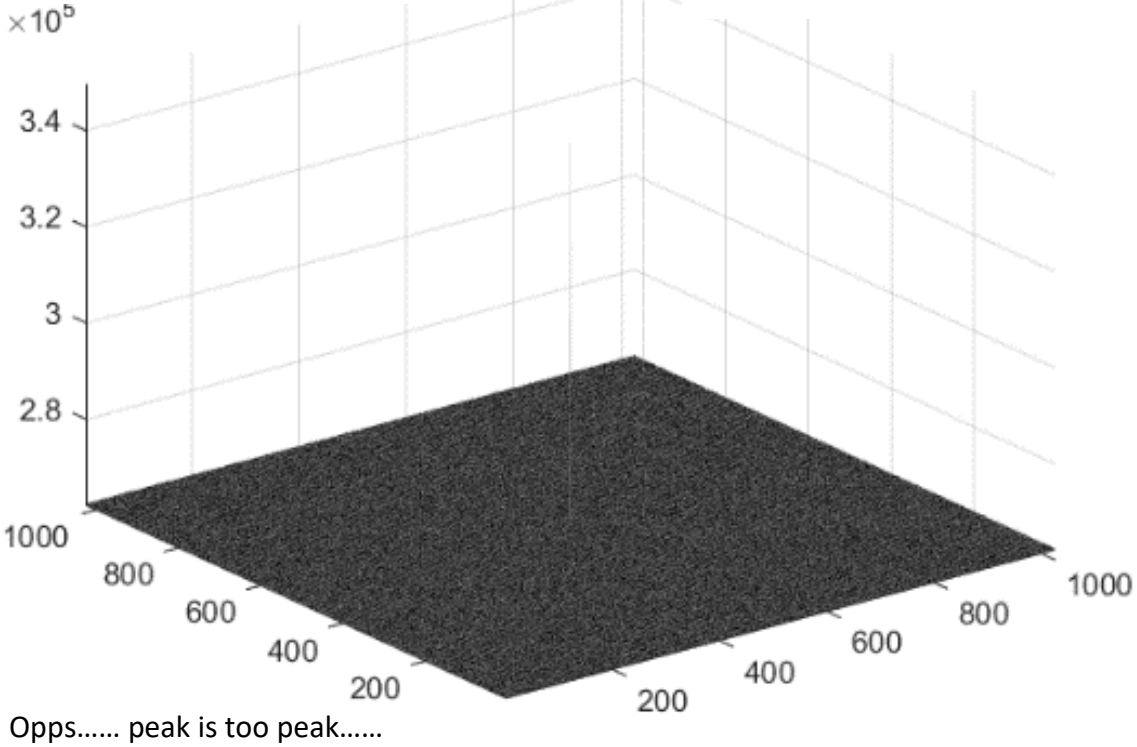
3 by 3 noise




64 by 64 noise




1024 by 1024 noise




Opps…… peak is too peak……

P.S. I did some animation works, run the “ccNoise.m” script if you are interested.