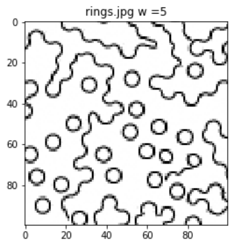
Problem 1

Let the source image size be N, target image size be S and window size be w. Then for each of the S^2 pixels in the output, we are considering $(N - W)^2$ candidate pixels. Each candidate is scored by computing the squared difference of two w size matrices, so the total runtime is $O(S^2(N - W)^2 w^2) = O(S^2 N^2 w^2)$ since $N >> w$, i.e. quadratic in the window size.

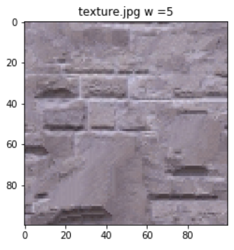
Therefore $w = 7$ will be about twice as slow as $w = 5$, and $w = 13$ will be 6 to 7 times as slow as $w = 5$.

While decreasing computational performance, increasing the window size improves the quality of the resulting texture. This is because we are considering a bigger window, and therefore the algorithm increases the probability that this pixel value ”makes sense” in that part of the image. We can see from the outputs below that for small windows, the algorithm does okay-ish for some parts of the image, but the results deteriorate at a certain point.

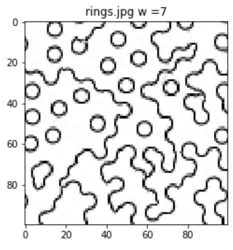
We will not get the same picture if we start with the same image seed, as we are choosing a random pixel from the candidates returned by Find_matches(). If we chose this pixel deterministically, let's say by always picking the best match, we should get the same image every time. This holds for all window sizes.



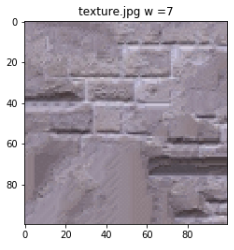
(a) w = 5



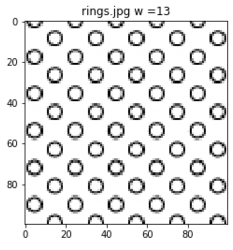
(b) w = 5



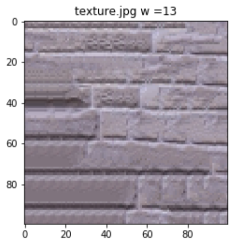
(c) w = 7



(d) w = 7



(e) w = 13



(f) w = 13