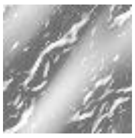
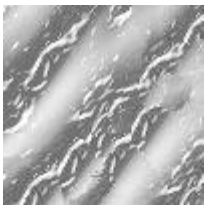
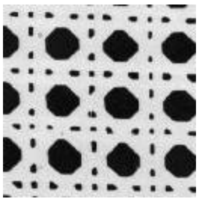
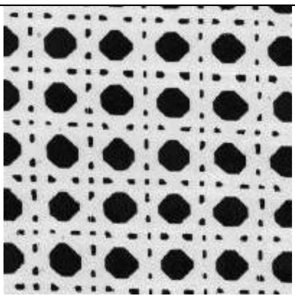
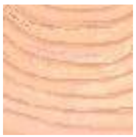
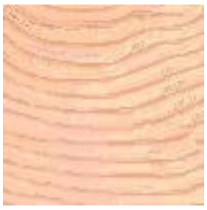
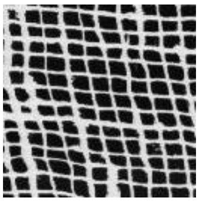
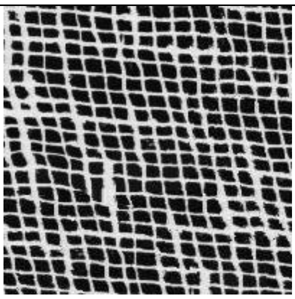
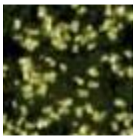
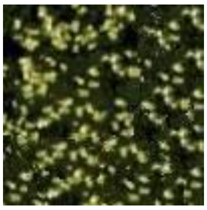
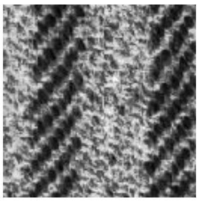
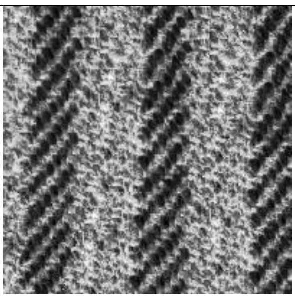
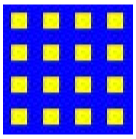
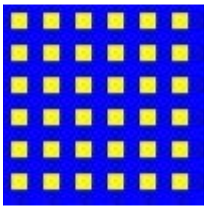
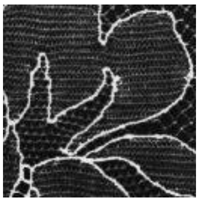
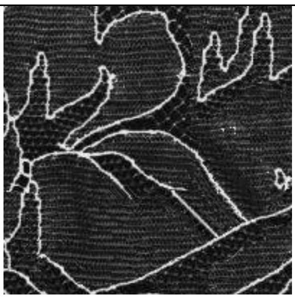
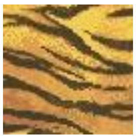
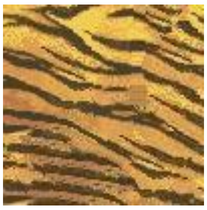
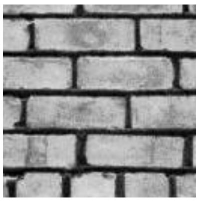
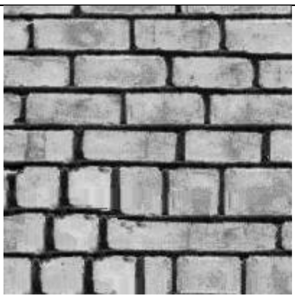
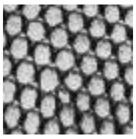
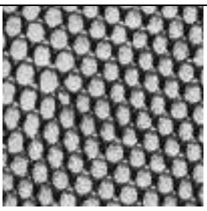
EE4212 – Part 2 Assignment 1 [Non-parametric Texture Synthesis]

Tay Kai Yang, A0136114U

Synthesized Results

Synthesized textures with window size = 31 x 31 pixels

Name	Initial Texture	Synthesized Texture	Name	Initial Texture	Synthesized Texture
texture1	 64 x 64	 100 x 100	texture7	 128 x 128	 200 x 200
texture2	 65 x 65	 100 x 100	texture8	 128 x 128	 200 x 200
texture3	 65 x 65	 100 x 100	texture9	 128 x 128	 200 x 200
texture4	 65 x 65	 100 x 100	texture10	 128 x 128	 200 x 200
texture5	 65 x 65	 100 x 100	texture11	 128 x 128	 200 x 200

texture6	 65 x 65	 100 x 100

Discussion

How neighbourhood size affects synthesis results and speed?

A. **Synthesis results**
    1. Neighbourhood size should minimally cover the basic pattern element of the texture, otherwise the synthesized texture will be vastly different from expected.
    2. The larger the neighbourhood size, the better the synthesis results.

B. **Speed**
    1. Larger neighbourhood size requires larger processing time.

What kind of data works best/worst?

A. **Works best**
    - Textures that are organized and have sub-elements at regular intervals.
    - Textures that have no distortion.

B. **Doesn’t work as well**
    - Irregular textures that do not follow an organized pattern.

How implemented algorithm can be improved in terms of result quantity and runtime efficiency.

A. **Improving result quantity**
    1. For irregular textures, we can use a larger initial texture image, which may produce better matches.
    2. Add a threshold to ignore matches that do not have high similarity. Proceed to fill other pixels first.

B. **Improving runtime efficiency**
    1. Implement a priority queue for choosing the next pixel with largest number of known neighbours. (Implemented using array).
    2. Represent all the windows from the initial texture in a feature space. For each next picture, obtain the neighbourhood window and find the closest match in the feature space.
        a. Can use approximate nearest neighbour algorithm rather than doing a linear search across the whole texture.
    3. Vectorize the calculation of Gaussian weighted SSD.
        a. Given a texture and neighbourhood size, extract all possible windows and stack them in a matrix. (number of pixels X height X width X channels)
        b. For a given neighbourhood in the output image, use broadcast operations to do comparisons across all neighbourhoods.
    4. Synthesize patches instead of pixels at a time.