Assignment 4

Task: Stereo disparity C/C++ implementation

Sequential Implementation of Stereo Disparity Algorithm (in C/C++) and profiling

Task description

- The algorithm to be implemented is "Depth estimation based on Zero-mean Normalized Cross Correlation (**ZNCC**)".

- ZNCC is a function that expresses the correlation between two grayscale images (or patches). The result of the function is an integer that becomes larger the more that these two patches are correlated. Estimating depth with ZNCC can be done evaluating the following expression.

$$\mathbf{ZNCC}(x,y,d) \triangleq \frac{\sum_{j=-\frac{1}{2}(B-1)}^{\frac{1}{2}(B-1)} \sum_{i=-\frac{1}{2}(B-1)}^{\frac{1}{2}(B-1)} [I_L(x+i,y+j) - \overline{I_L})] * [I_R(x+i-d,y+j) - \overline{I_R}]}{\sqrt{\sum_{j=-\frac{1}{2}(B-1)}^{\frac{1}{2}(B-1)} \sum_{i=-\frac{1}{2}(B-1)}^{\frac{1}{2}(B-1)} [(I_L(x+i,y+j) - \overline{I_L}]]^2} * \sqrt{\sum_{j=-\frac{1}{2}(B-1)}^{\frac{1}{2}(B-1)} \sum_{i=-\frac{1}{2}(B-1)}^{\frac{1}{2}(B-1)} [I_R(x+i-d,y+j) - \overline{I_R}]^2}}$$

B = window size, I_L, I_R= left and right images, $\overline{I_L}$, $\overline{I_R}$ window average, d = disparity value (0 to d)

- The final disparity value will be chosen based on the Winner-takes-it-all-approach, where the value of d with largest ZNCC(x,y,d) value is chosen for the pixel value (x,y) of the final disparity map.

Note that for our simple case, the disparity can be considered one-dimensional (horizontal only) and there is no disparity in y-axis. This is due to the fact that the sample images that we use for our computation are already *rectified*, as it is usually the case from the input of stereo cameras.

- Once the ZNCC equation is evaluated and a first estimation of the depth map is obtained, it should be post processed for refinement.
- The post processing stage consists of two phases: cross-checking and occlusion filling.

Cross checking: (Two input images, one output image)

Cross checking is a process where you compare two depth maps. The left disparity map is obtained by mapping the image on the left against the one on the right. The right disparity map is obtained analogously by mapping the image on the right against the one on the left.

To obtain a consolidated map, the process consists in checking that the corresponding pixels in the left and right disparity images are consistent. This can be done by comparing their absolute difference to a threshold value. For our case, it is recommended to start

with a threshold value of 8, while the best threshold value for your implementation can be obtained by experimentation. If the absolute difference is larger than the threshold, then replace the pixel value with zero. Otherwise the pixel value remains unchanged. This process helps in removing the probable lack of consistency between the depth maps due to occlusions, noise, or algorithmic limitation.

Occlusion filling: (One input image, one output image)

Occlusion filling is the process of eliminating the pixels that have been assigned to zero by the previously calculated cross-checking. In the simplest form it can be done by replacing each pixel with zero value with the nearest non-zero pixel value. However, in this exercise it is recommended that you experiment with other more complex post-processing approaches that obtain the pixel value in different forms.

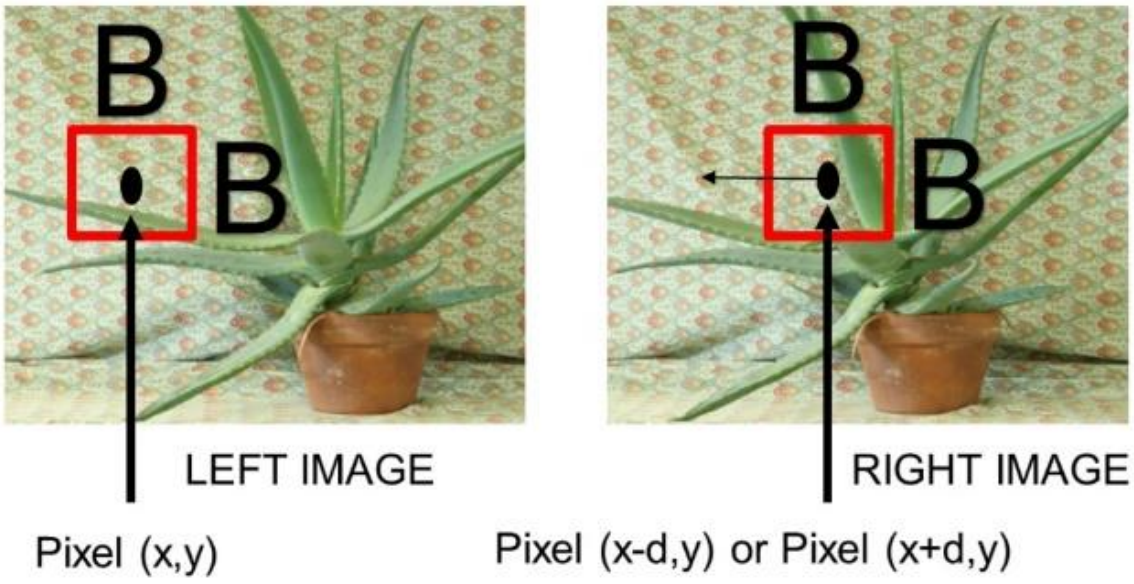
Task Implementation:

Two input images required for the task will be provided along with the assignment documentation. Having understood the description of stereo disparity task from previous section, the following custom functions have to be implemented.

- **ReadImage** – Use lodepng.cpp/lodepng.c, lodepng.h libraries to read the input images in RGBA format
- **ResizeImage** – Write your custom C/C++ function to downscale the read input RGBA images by 4
- **GrayScaleImage** – Write your custom C/C++ function to convert images to gray scale. **Save** the Gray Scale Images as well and submit them along with the report
- **CalcZNCC** – Write your custom C/C++ functions to compute the ZNCC value for pixels as described in the previous section. **Save** the output image after applying ZNCC and submit it along with the report
- **CrossCheck** – Write your custom C/C++ function to applying this post processing functionality. The description regarding this is mentioned under previous section. **Save** the output image after applying CrossCheck post processing and submit it along with the report
- **OcclusionFill** – Write your custom C/C++ function to apply the second post processing functionality. The description regarding this is provided under previous section as well. **Save** the final output image after applying OcclusionFill post processing and submit it along with the report
- **WriteImage** – Use lodepng.cpp/lodepng.c libraries to save the output images at different times of the implementation. **Remember** to not comment this functionalty while measuring the timing/profiling information of other functions present under this task
- **ProfilingInfo** – Provide profiling information on each of these functionalities. You are free to choose to implement this part. Simplest profiling information is the timing information of each of these functionalities. Do observe the time, CPU consumes for each one of them. You can **include** any analysis on this in your report.

Pseudo Code:

A pseudo code is also provided here for reference



ZNCC VALUE = UPPER SUM / (SQRT (LOWER SUM_0) * SQRT(LOWER SUM_1))

UPPER SUM += left_pixel_val_diff_from_avg * right_pixel_val_diff_from_avg

LOWER SUM_0 += left_pixel_ val_diff_from_avg * left_pixel_ val_diff_from_avg

PSEUDO CODE:

```
FOR J = 0 to image height
    FOR I = 0 to image width
        FOR d = 0 to MAX_DISP
            FOR WIN_Y = 0 to WIN_SIZE
                FOR WIN_X = 0 to WIN_SIZE
                    CALCULATE THE MEAN VALUE FOR EACH WINDOW
                END FOR
            END FOR

            FOR WIN_Y = 0 to WIN_SIZE
                FOR WIN_X = 0 to WIN_SIZE
                    CALCULATE THE ZNCC VALUE FOR EACH WINDOW
                END FOR
            END FOR

            IF ZNCC VALUE > CURRENT MAXIMUM SUM THEN
                UPDATE CURRENT MAXIMUM SUM
                UPDATE BEST DISPARITY VALUE
            END IF
        END FOR

    DISPARITY_IMAGE_PIXEL = BEST DISPARITY VALUE
    END_FOR
END FOR
```

Note: The previously depicted description of the task, algorithmic implementation steps and the enclosed pseudocode are just meant to help you get started with the algorithm implementation. However, notice that this is not the only (or best) solution. You are free to experiment with other approaches.

Expected result:

A working version of the implementation, a brief report (max1-2 pages), saved output images all together in the form of a compressed folder (.zip file)

The report should contain about the task solved, brief description of your implement and the final screenshot of your outputs asked to be displayed under the assignment.