HACETTEPE UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

BBM415 FUNDAMENTALS OF IMAGE PROCESSING LABORATORY – 2022 FALL

PROGRAMMING ASSIGNMENT 2

November 13, 2022

Student Name	Student Number
Yasin Şimşek	**21827847**

INTRODUCTION

The main goal of this assignment is to apply color transfer. We take one picture as the source, another picture as the reference, and try to make the source image looks like to target image colors. We use Reinhard formula to change colors. The method of Reinhard:

$$\begin{aligned} l^* &= l - \mu_l \\ \alpha^* &= l - \mu_\alpha \\ \beta^* &= l - \mu_\beta \end{aligned} \quad \longrightarrow \quad \begin{aligned} l' &= l^* \frac{\sigma_s^l}{\sigma_t^l} \\ \alpha' &= \alpha^* \frac{\sigma_s^\alpha}{\sigma_t^\alpha} \\ \beta' &= \beta^* \frac{\sigma_s^\beta}{\sigma_t^\beta} \end{aligned}$$

The mean and the standard deviation are calculated for each channel and the pixel values of the source image is changed by referenced these values. You can see details of the method of Reinhard via. I modified the formula given in assignment pdf. Instead of the formula above, I divided the standard deviation of the target images to the standard deviation of the source image. It gave me better solutions.

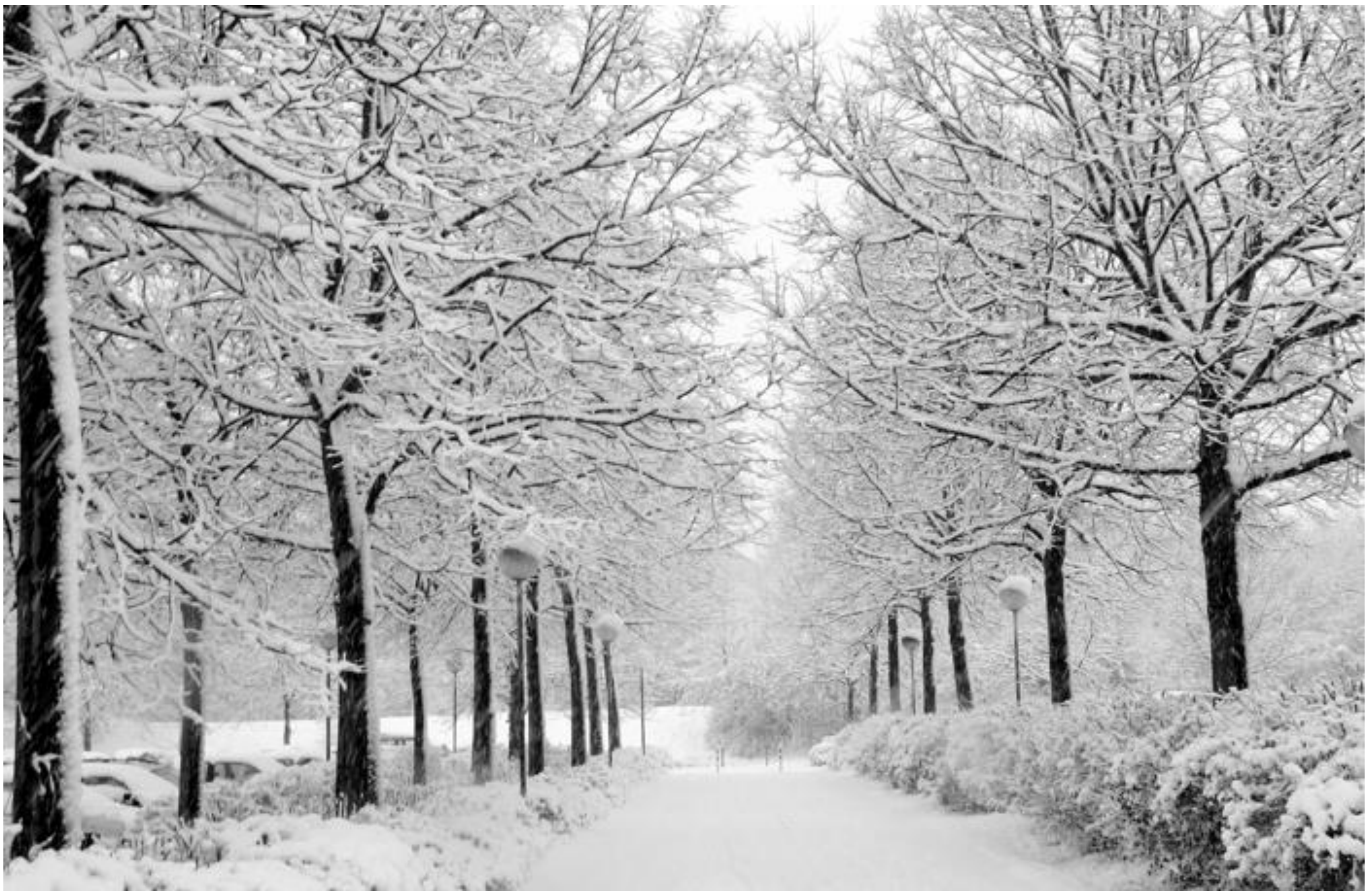
EXPERIMENT

PART 1

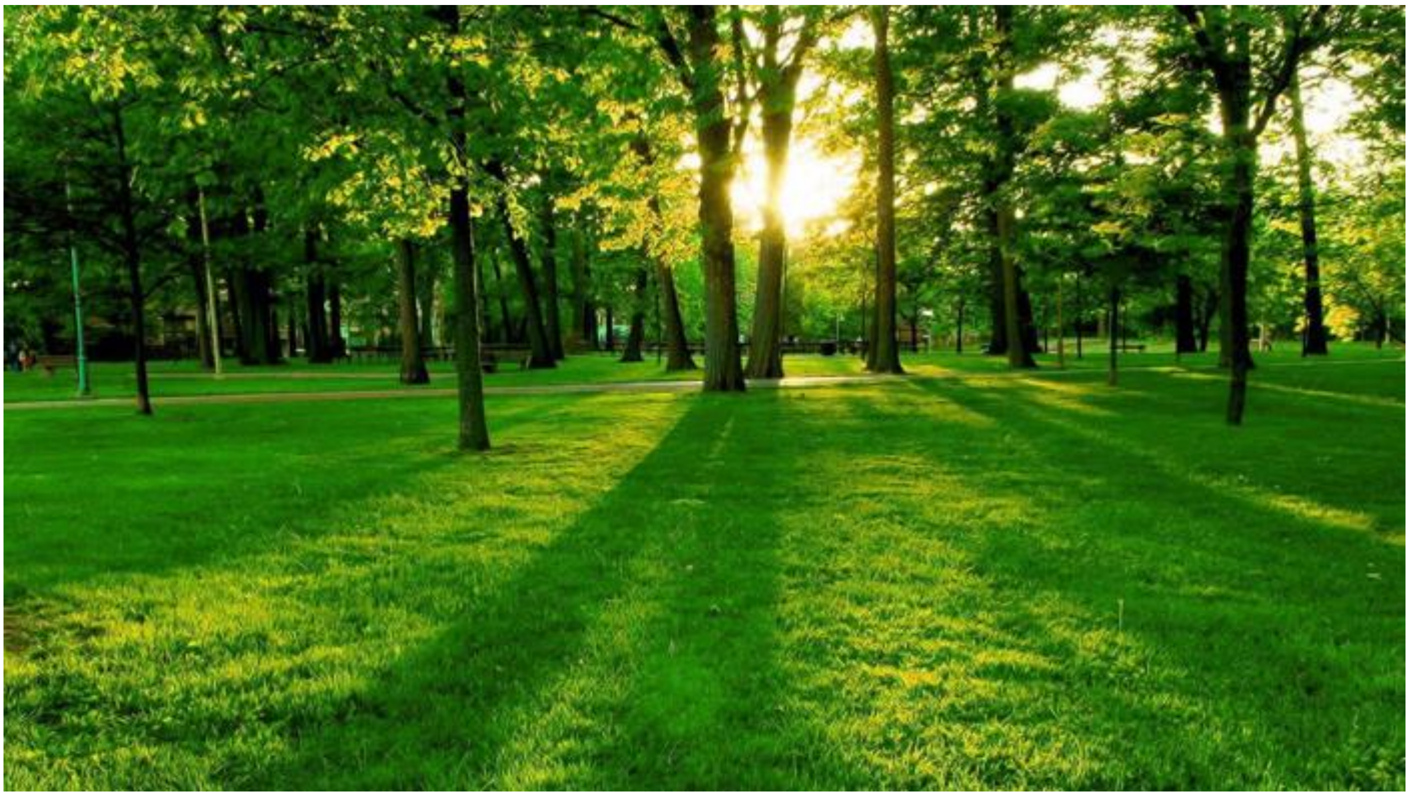
In this part, I applied only Rehinard method without any image matching. Firstly, I calculated the mean and the standard deviation of each panels of the source image and the target image. After that, I calculated new pixel value by using the mean and the standard deviation values. Before record it, I checked the new value if there is any oversizing and corrected it if there is. Finally, I changed the old value with new value. Thus, the result image was created by referencing the color distribution of the target image. The code of this algorithm:

```
def color_transfer(source_image_panel, x_start, x_finish, y_start, y_finish,
                   source_mean, target_mean, source_std, target_std):
    for x in range(x_start, x_finish):
        for y in range(y_start, y_finish):
            old_pixel = source_image_panel[x][y]
            new_pixel = (old_pixel - source_mean) * (target_std / source_std)
            new_pixel += target_mean
            new_pixel = round(new_pixel)

            if new_pixel > 255:
                new_pixel = 255
            elif new_pixel < 0:
                new_pixel = 0

            source_image_panel[x][y] = new_pixel

    return source_image_panel

s_mean = get_mean(0, sources[j].shape[0], 0, sources[j].shape[1], sources[j])
t_mean = get_mean(0, targets[j].shape[0], 0, targets[j].shape[1], targets[j])
s_std = get_std(0, sources[j].shape[0], 0, sources[j].shape[1], sources[j])
t_std = get_std(0, targets[j].shape[0], 0, targets[j].shape[1], targets[j])
new_sources = []
for p in range(0, 3):
    new_sources.append(color_transfer(sources[j][:, :, p], 0, sources[j].shape[0], 0,
sources[j].shape[1], s_mean[p], t_mean[p], s_std[p], t_std[p]))
new_sources = np.dstack((new_sources[0], new_sources[1], new_sources[2]))
Image.fromarray(new_sources).save("result1_" + str(j + 1) + ".png")
```

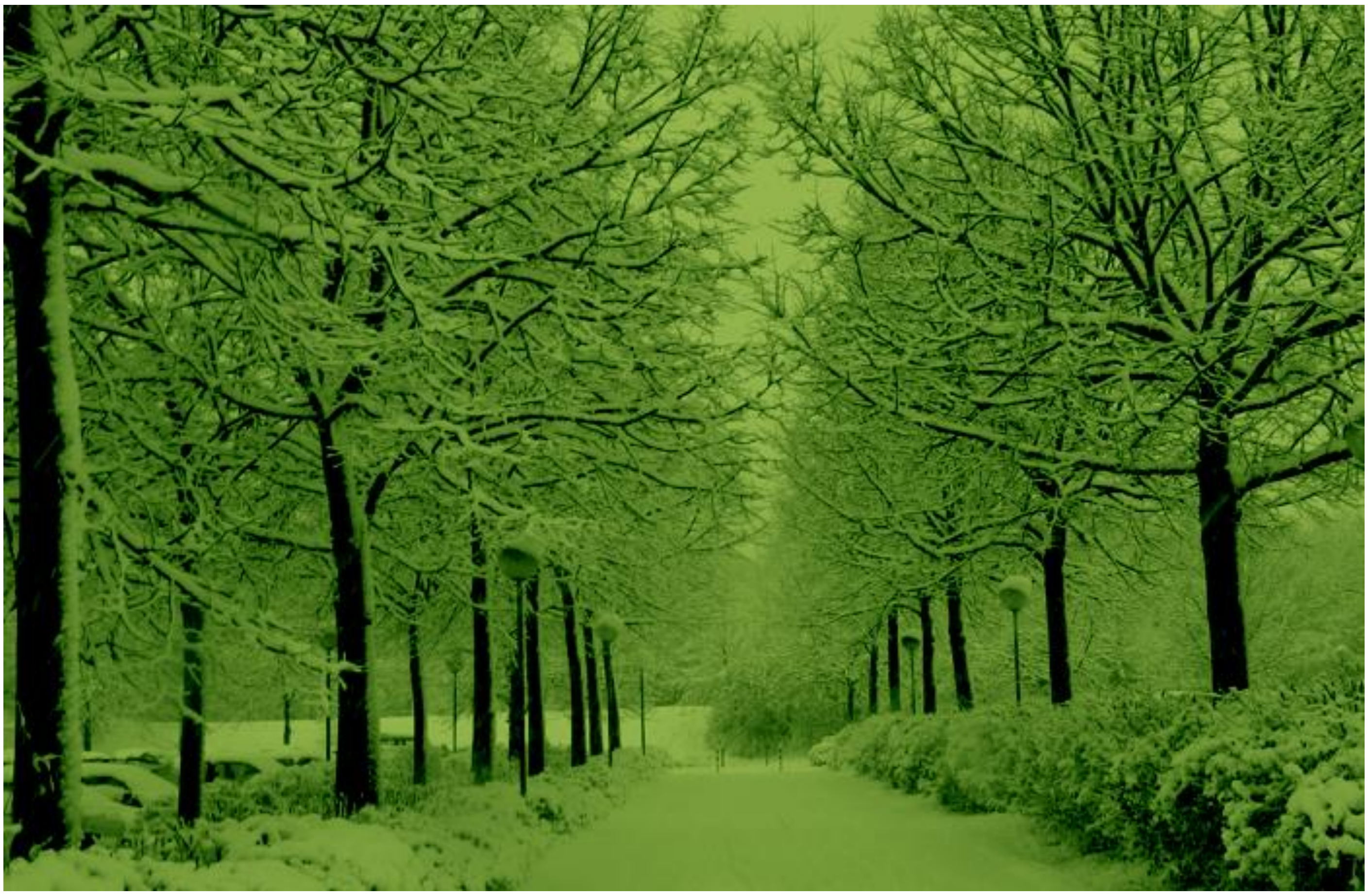
THE SOURCE IMAGE – 1



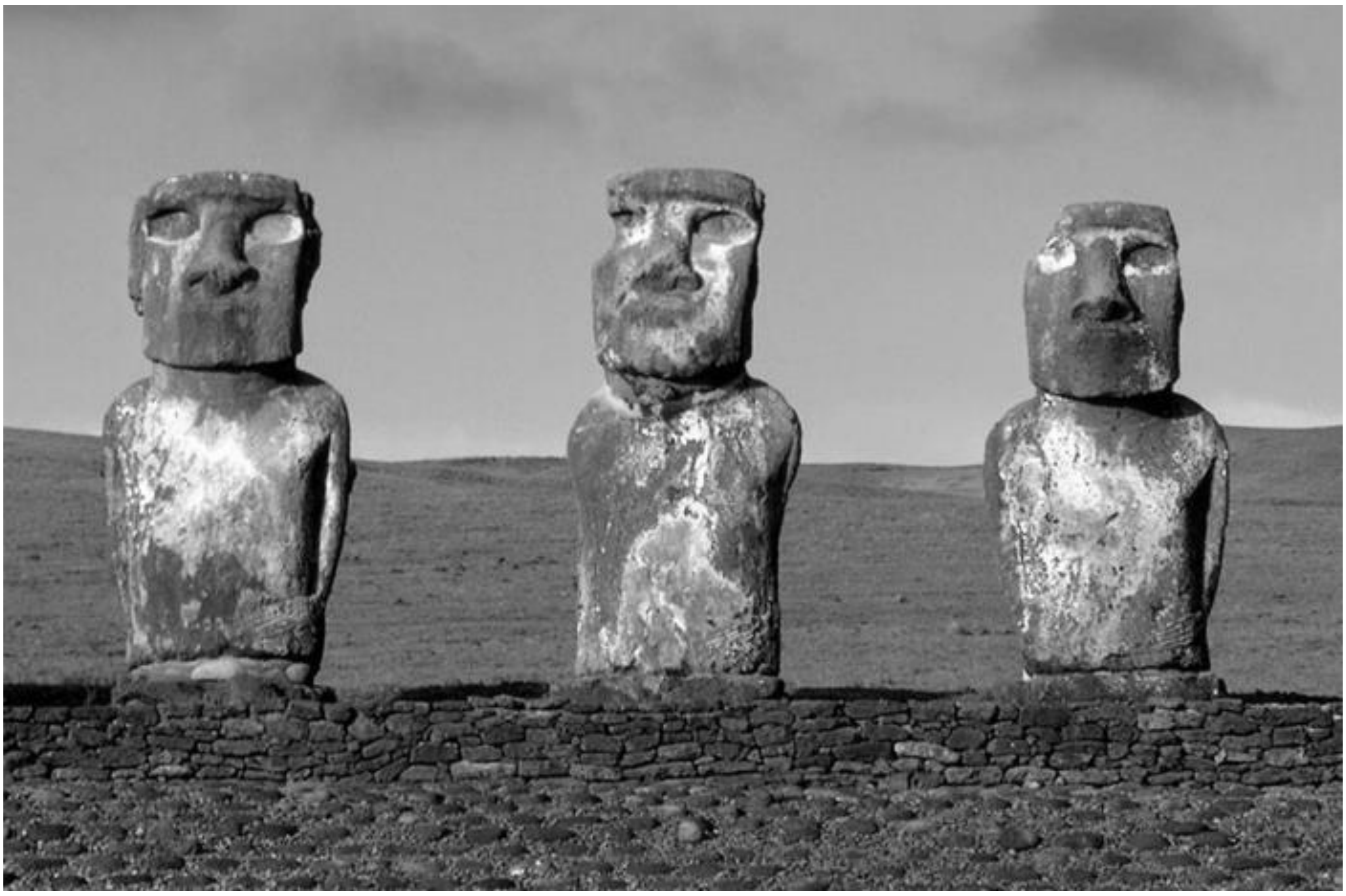
THE TARGET IMAGE - 1



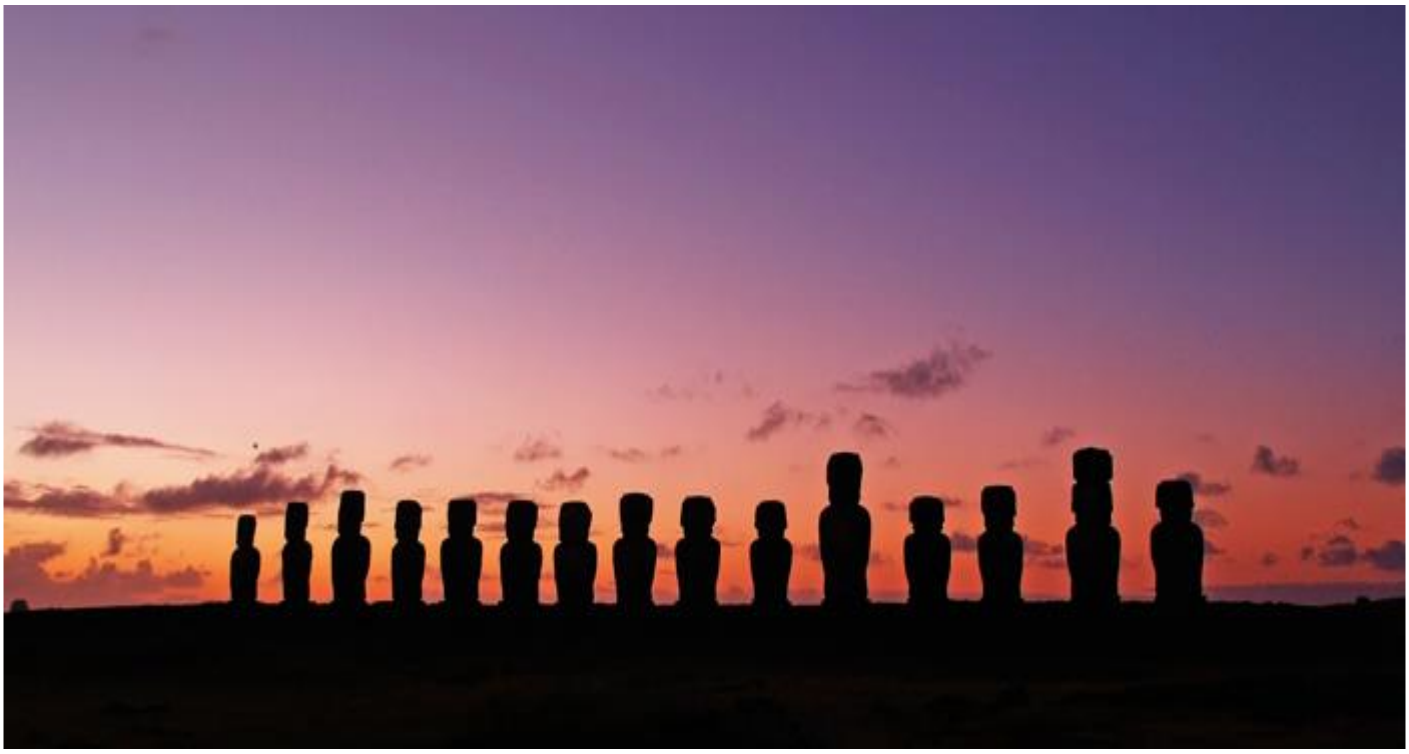
THE RESULT IMAGE – 1



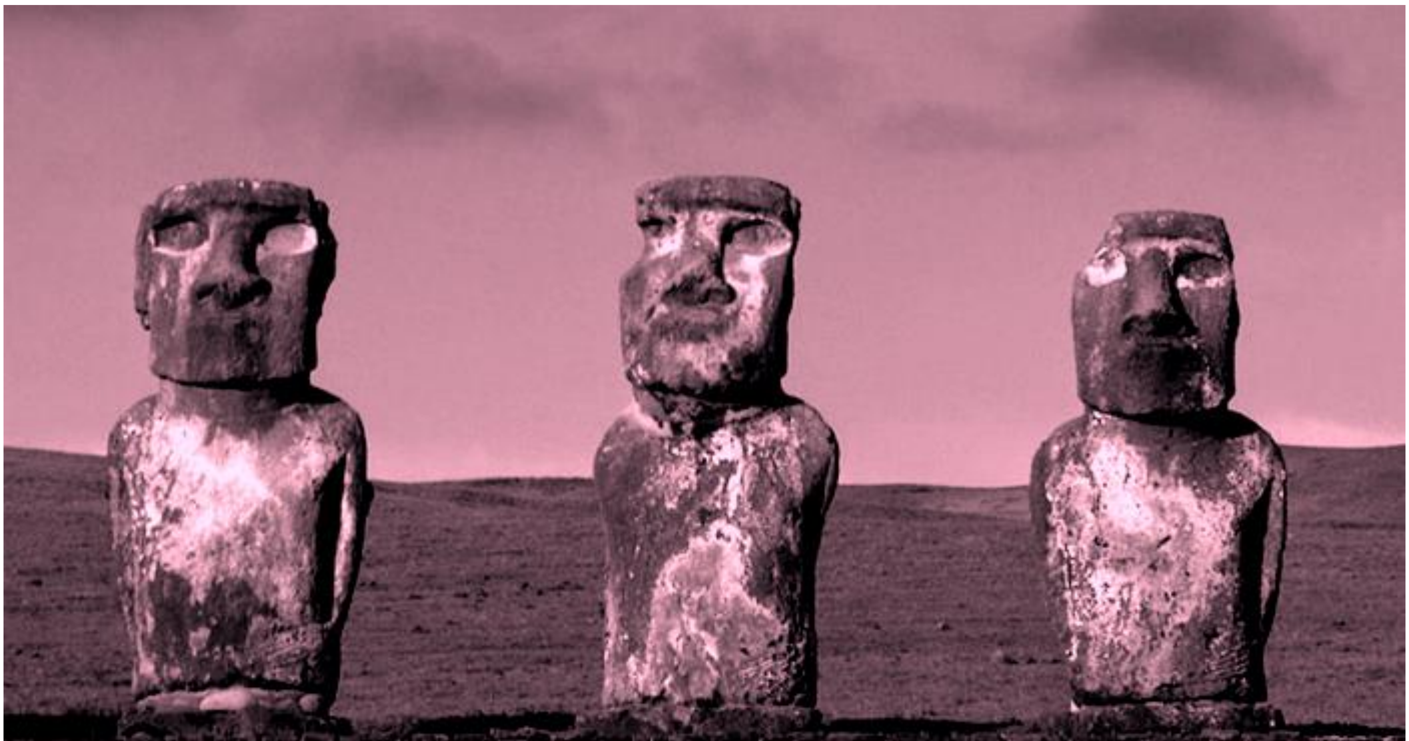
THE SOURCE IMAGE – 2



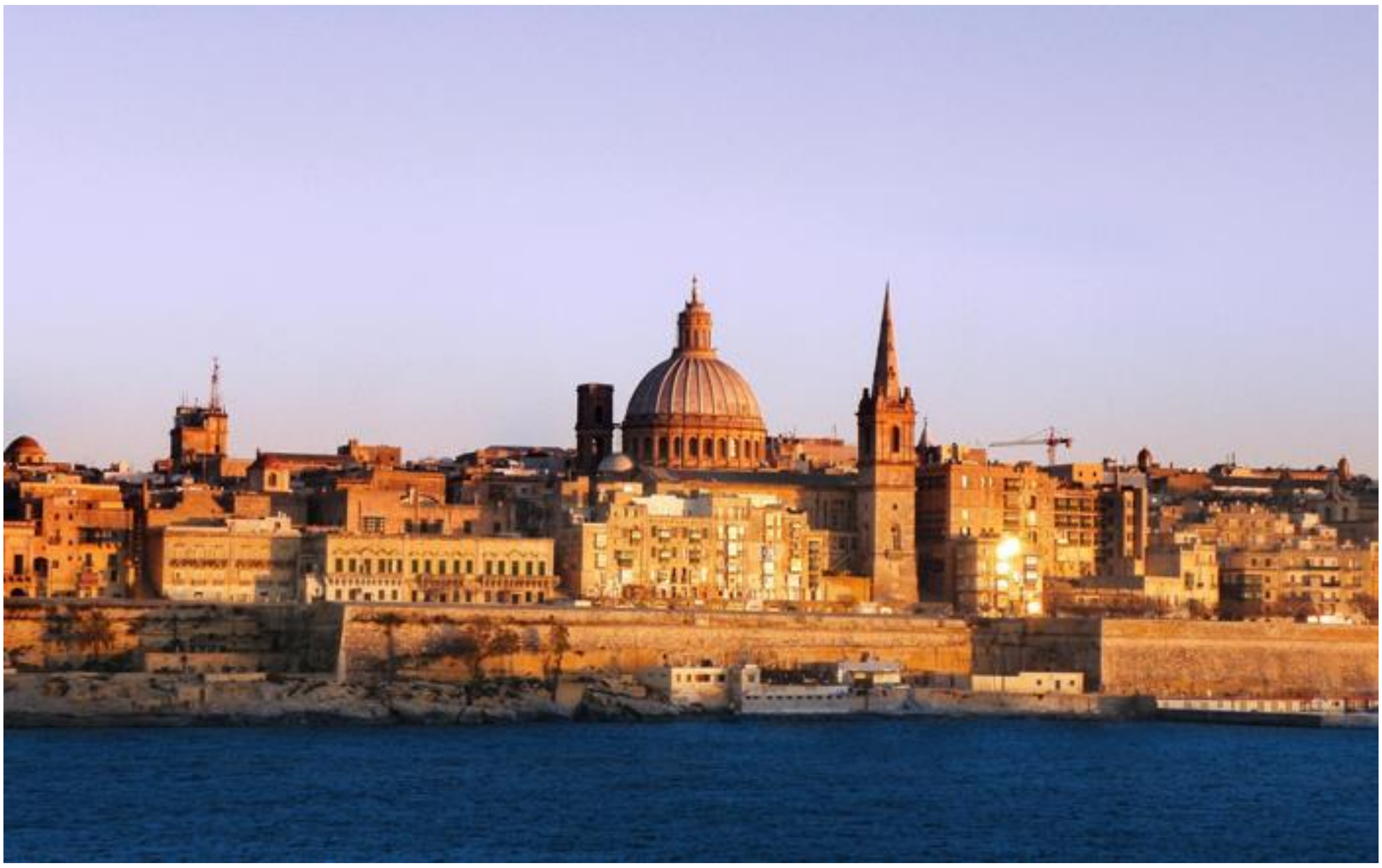
THE TARGET IMAGE – 2



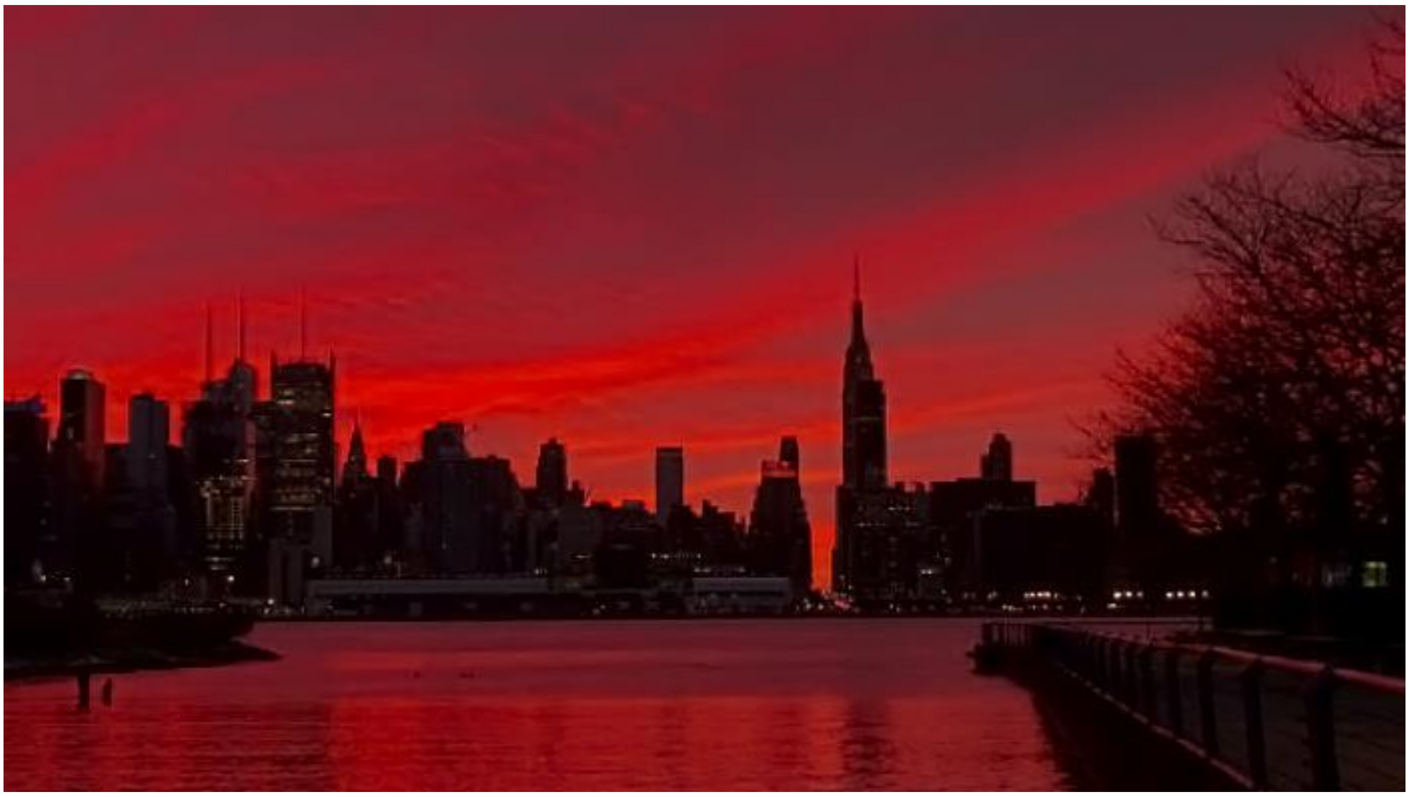
THE RESULT IMAGE – 2



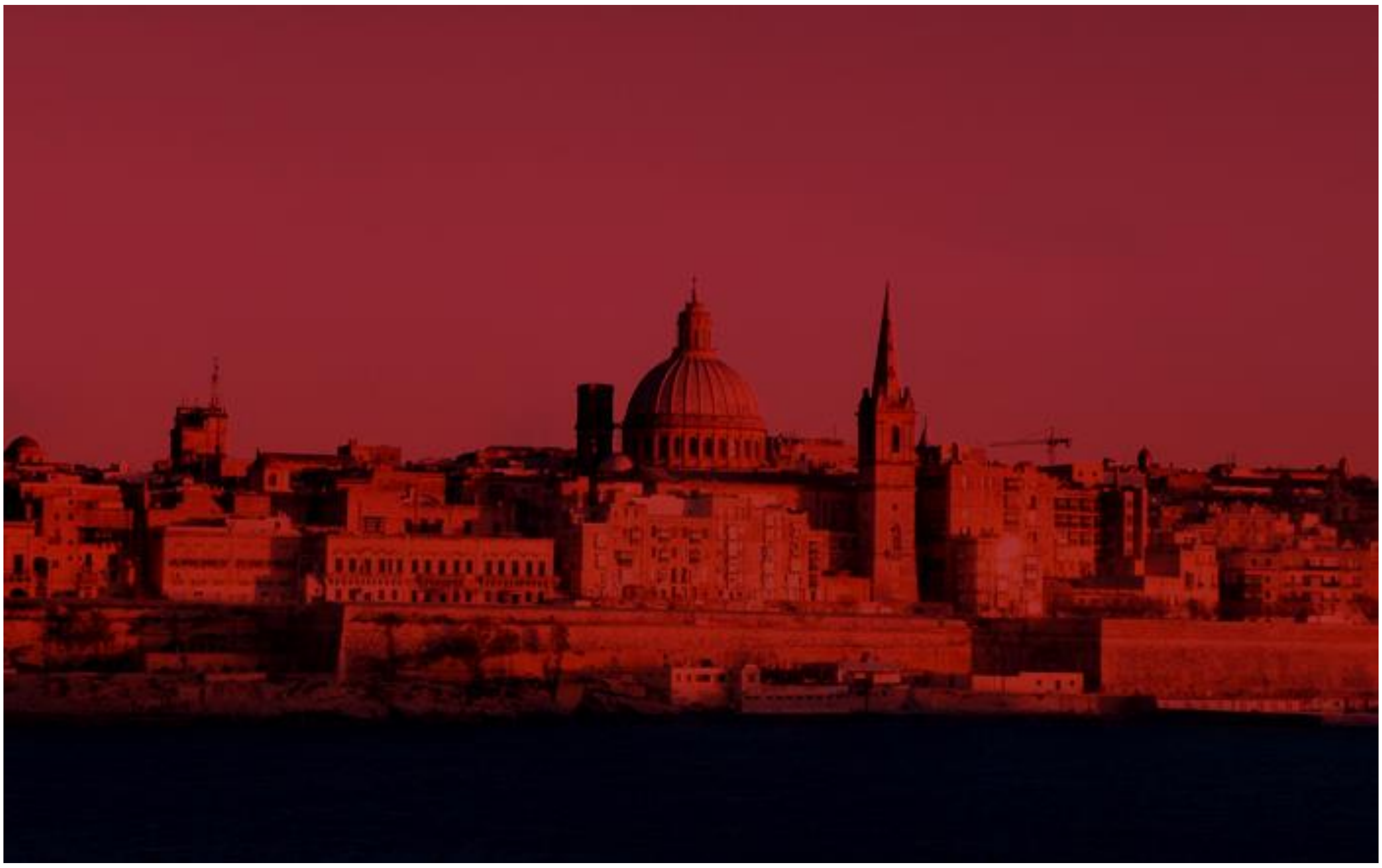
THE SOURCE IMAGE – 3



THE TARGET IMAGE – 3



THE RESULT IMAGE – 3



PART 2

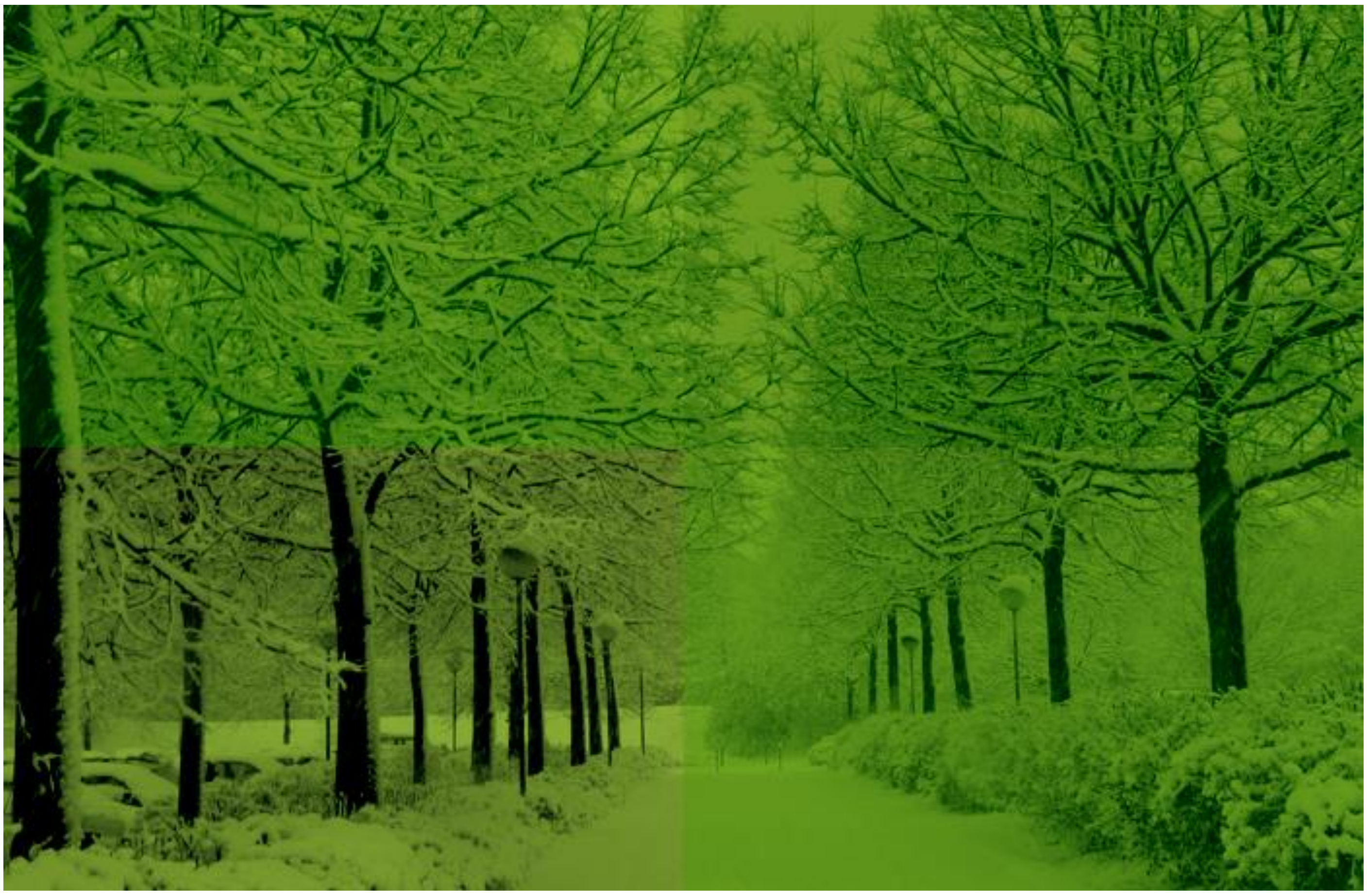
In this part, I applied SSD before color transfer. Its purpose is to find similar regions of the source images and the target images. I divided the images to four part. Before decided to divide into four region, I tried to several possibilites like dividing into 9 region or 16 region, as a result, dividing into four region was the best solution for this project.

After dividing images, I compared all regions each other and kept the regions with the best match score (This means the SSD is smallest). These regions were used as reference to find the mean and the standard deviation. Finally, using these values, I transferred color separately for each matching region. The code for this algorithm:

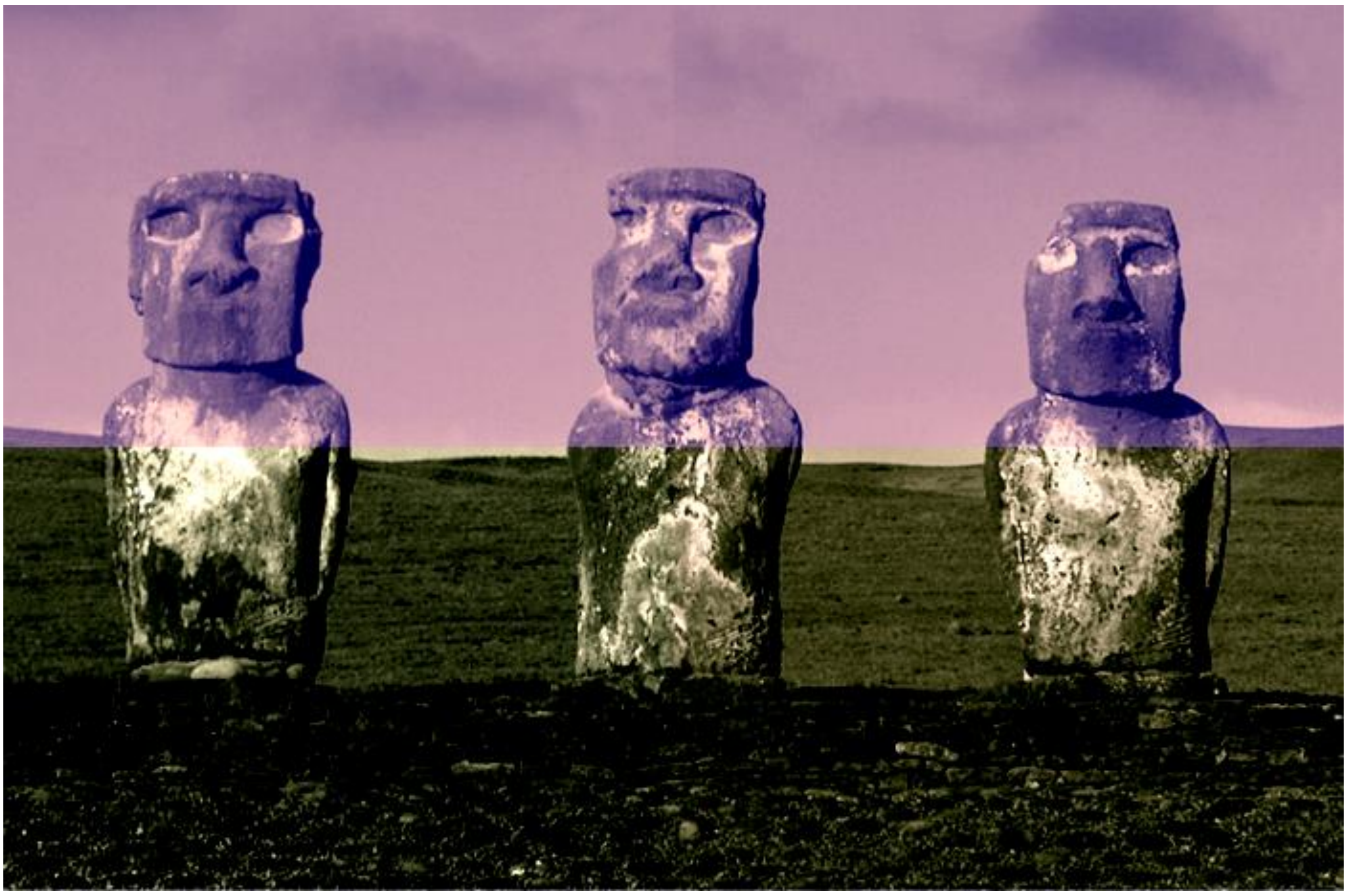
```
def color_transfer_according_to_ssd(source_image_array, target_image_array):
    y = target_image_array.shape[1] // 2
    if target_image_array.shape[0] > source_image_array.shape[0]:
        x = source_image_array.shape[0]
    else:
        x = target_image_array.shape[0]

    x = x // 2

    source_image_array = [source_image_array[m:m + x, n:n + y] for m in range(0,
target_image_array.shape[0], x)
                          for n in range(0, y * 2, y)][:4]

    target_image_array = [target_image_array[m:m + x, n:n + y] for m in range(0,
target_image_array.shape[0], x)
                          for n in range(0, y * 2, y)][:4]

    limits_of_regions = []

    for m in range(0, 4):
        best_ssd = float("inf")
        matching = (0, 0)
        for n in range(0, 4):
            current_ssd = get_ssd(source_image_array[m], target_image_array[n])
            if current_ssd < best_ssd:
                best_ssd = current_ssd
                matching = (m, n)
        l_source = matching[0] // 2 * x, matching[0] // 2 * x + x, (matching[0] % 2) *
y, (matching[0] % 2) * y + y
        l_target = matching[1] // 2 * x, matching[1] // 2 * x + x, (matching[1] % 2) *
y, (matching[1] % 2) * y + y
        limits_of_regions.append((l_source, l_target))

    return limits_of_regions
```

The result obtained from the input photos shown in Part 1 as a result of the operations in Part 2:

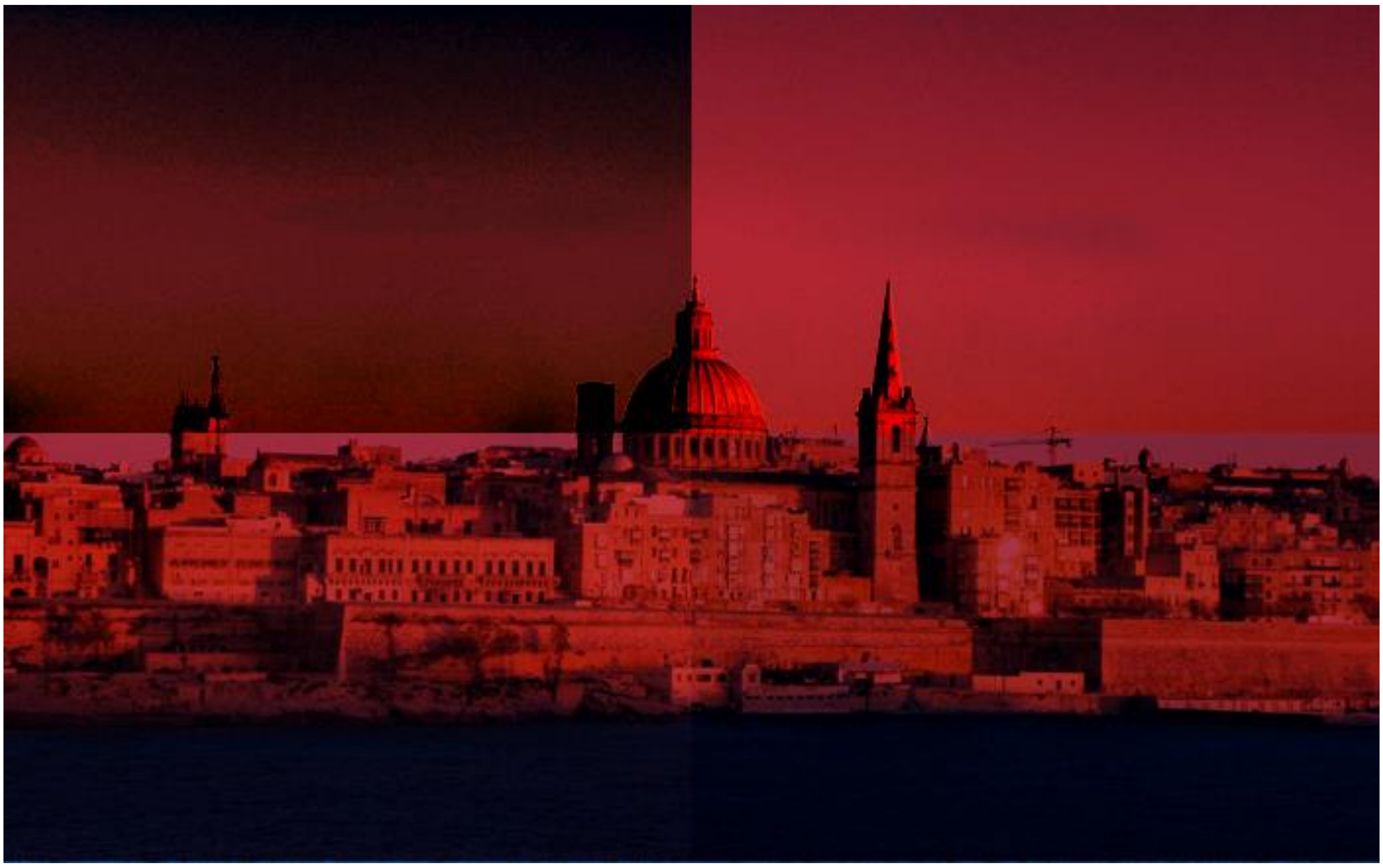
THE RESULT IMAGE – 4



THE RESULT IMAGE – 5



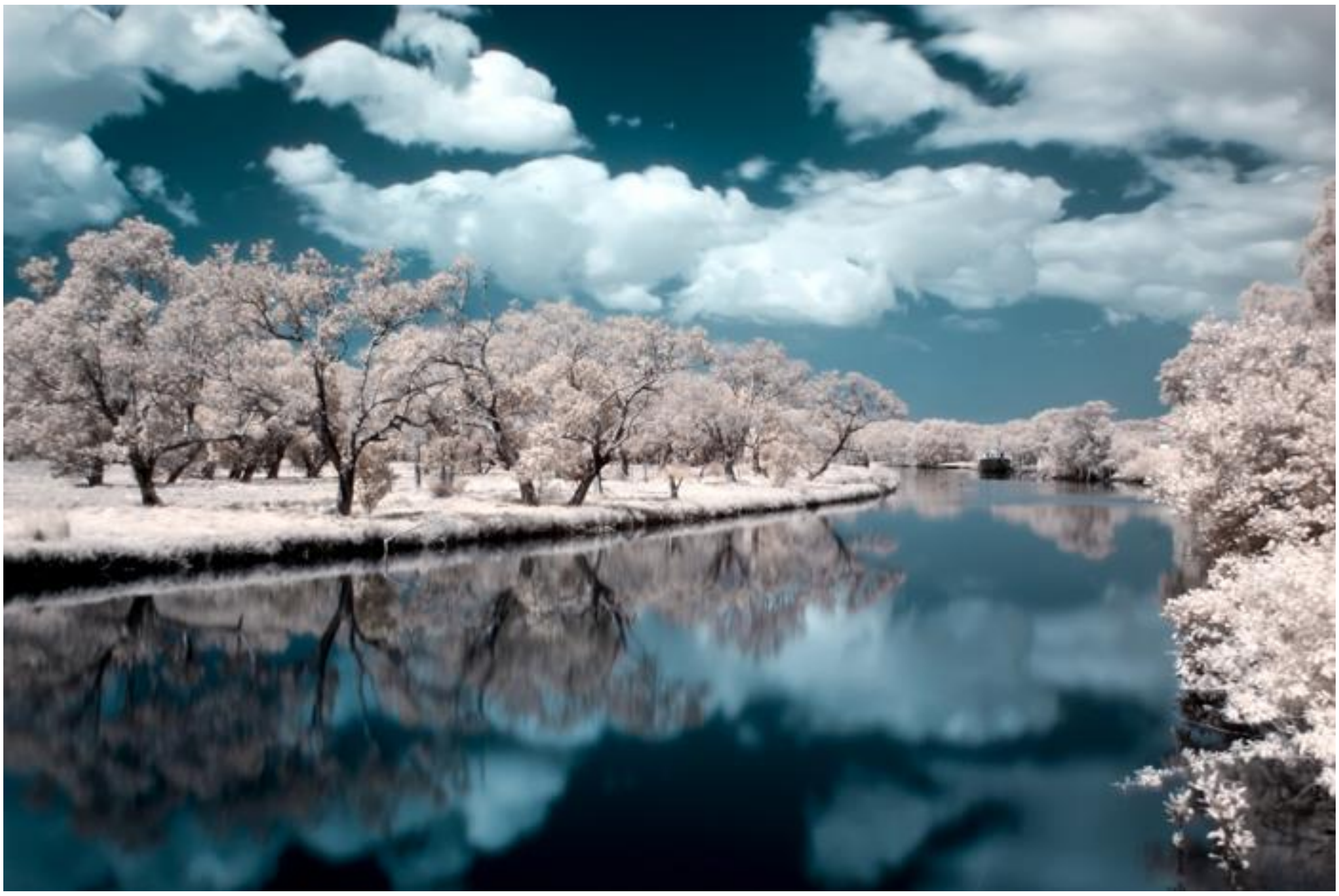
THE RESULT IMAGE – 6



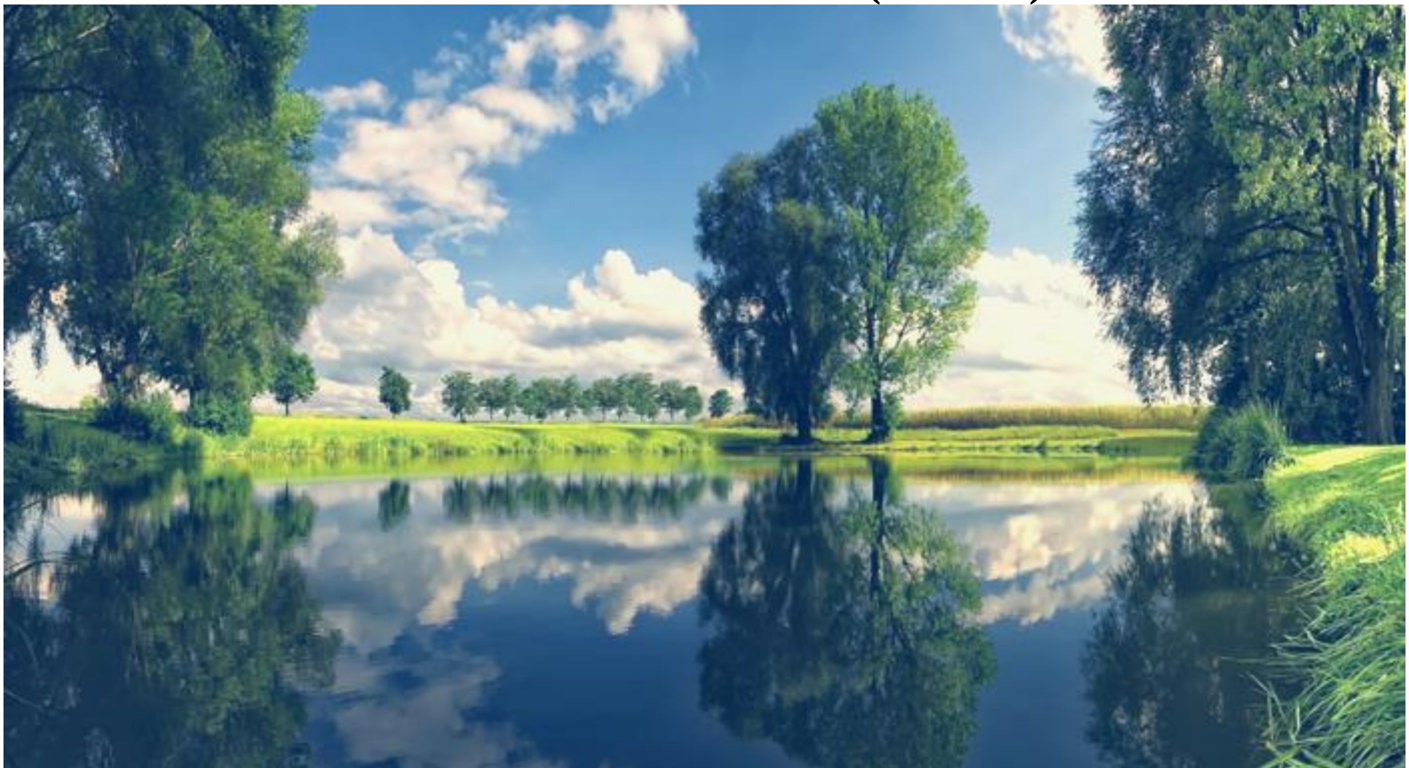
I could not produce the completely right results from some images. Also I could not apply color transfer in Part 2, the images look like as combined from some piece of different images. In my opinion, the pure Reinhard method is not enough, I have should applied more complex changes. The another reason I think, I could not apply the SSD successfully. Because of these reasons, I could not produce completely correct solution for this assignment. An example of a failed result:

THE SOURCE IMAGE - 7

THE TARGET IMAGE - 7



THE RESULT IMAGE – 7 (PART 1)



THE RESULT IMAGE – 7 (PART 2)