Homework 1

Computer Graphics

Francesco Palandra 1849712

October 15, 2021

Contents

1 Sepia
(easy)

For this filter we followed this algorithm [1] to get the sepia effect[1]. The algorithm applies for each pixel the adjustment for each channel to get the sepia's tonality. We added some check for the output's values, if any of these output values was greater than 255, we simply setted it to 255. These specific values are the values for sepia tone that are recommended by Microsoft.[Sepia Filter]

Algorithm 1 Sepia's algorithm

```
Input: image $I$
Output: image $I'$
1: for each pixel do
2:     $outputRed \leftarrow (inputRed \cdot 0.393) + (inputGreen \cdot 0.769) + (inputBlue \cdot 0.189)$
3:     $outputGreen \leftarrow (inputRed \cdot 0.349) + (inputGreen \cdot 0.686) + (inputBlue \cdot 0.168)$
4:     $outputBlue \leftarrow (inputRed \cdot 0.272) + (inputGreen \cdot 0.534) + (inputBlue \cdot 0.131)$
5: end for
```

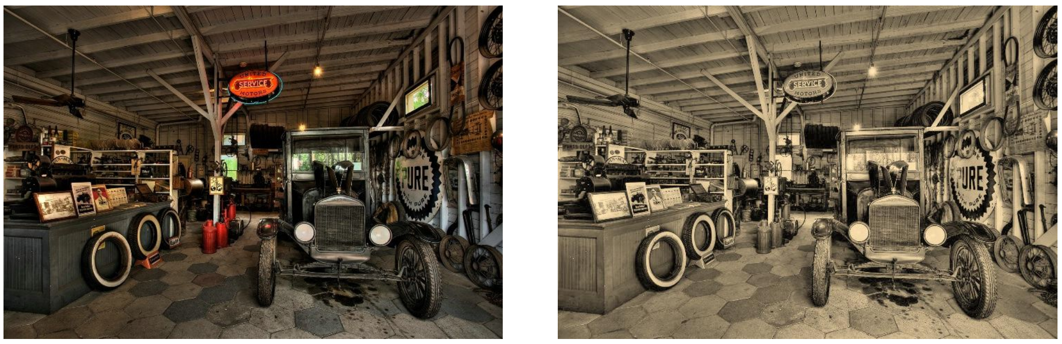

(a) Input Image (b) Sepia with contrast

Figure 1: Application of Sepia's Algorithm [1]

2 Negative (easy)

In this case we simply iterate for each pixel of the input image and take the opposite color. That means we subtract 1 and take the module, or we take the pixel and subtract to 1.

Algorithm 2 Negative's algorithm

```
Input: image I
Output: image I'
1: for each pixel px do
2:     $px \leftarrow 1 - px$
3: end for
```

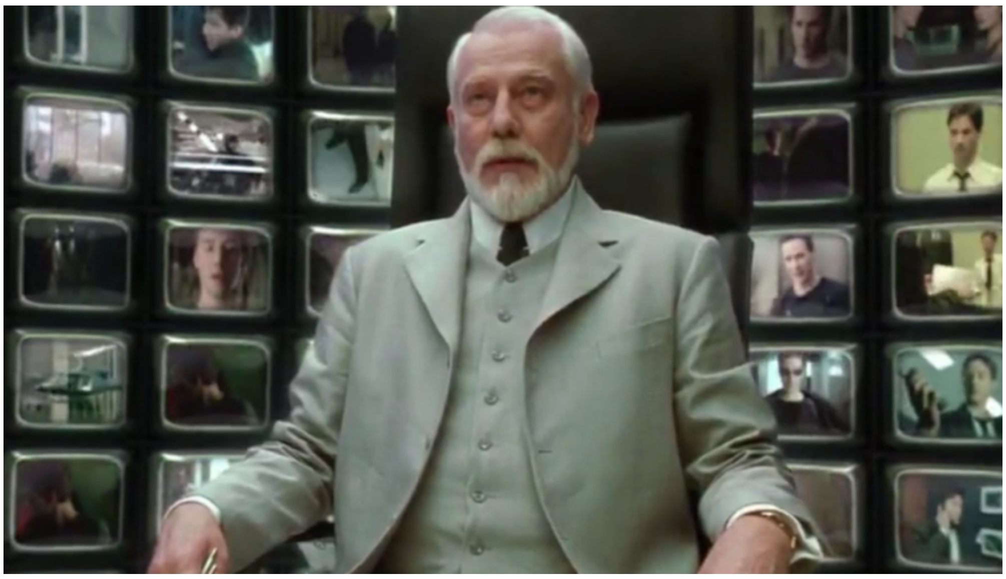


Figure 2: Input Image

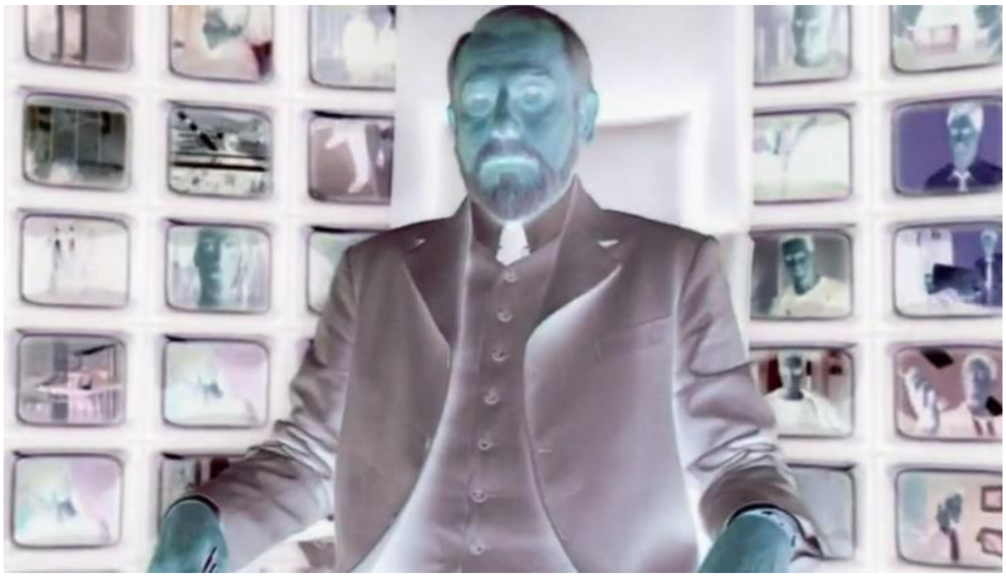


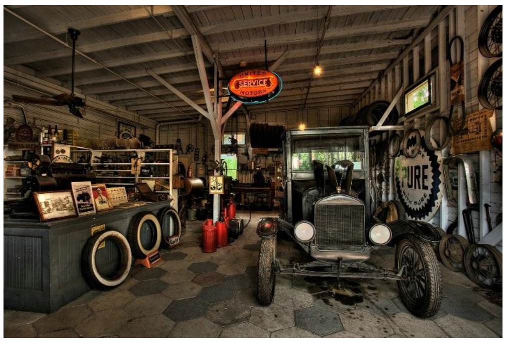
Figure 3: Negative

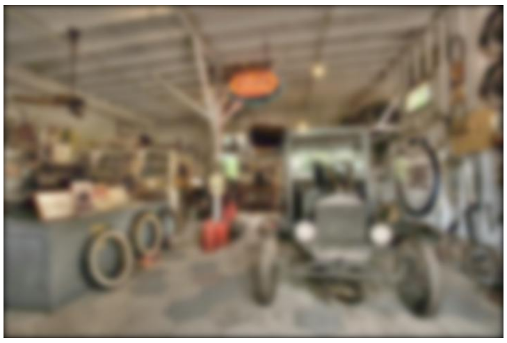
3 Gaussian Blur
(easy)

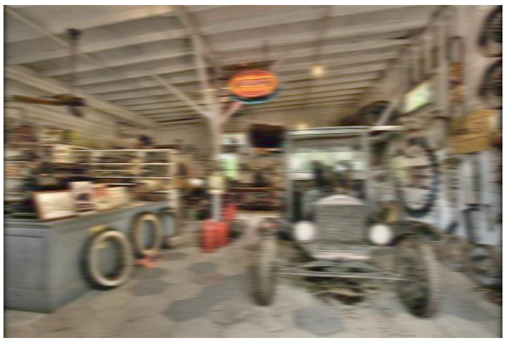
For the Gaussian blur we first generate the kernel, a matrix with values between 0 and 1 following the gaussian distribution on the center of the matrix. We then iterate for each pixel and we calculate the value of the new pixel by summing the weight of the pixel around based on the value of the kernel. More far a pixel is from the pixel which are we calculating the blurred value less it will influence it. To simply the algorithm we first compute the calculus of the horizontal blur and then we compute for the vertical blur. Implementation Gaussian Blur

Algorithm 3 Gaussian Blur's algorithm

```
Input: $image\ I,\ \sigma_x,\ \sigma_y$
Output: $image\ I'$
1: $Kernel \leftarrow generateKernel(\sigma_x, \sigma_y)$
2: for $(each\ pixel; p_0)$ do
3:     $blackPixel \leftarrow (0,\ 0,\ 0,\ 0)$
4:     for $(e_0\ in\ kernel)$ do
5:         $Blurredpixel+ = p_0 \cdot e_0$
6:     end for
7:     $p_0 = Blurredpixel$
8: end for
```


(a) Input Image


(b) output Image


(a) Orizzontal blur

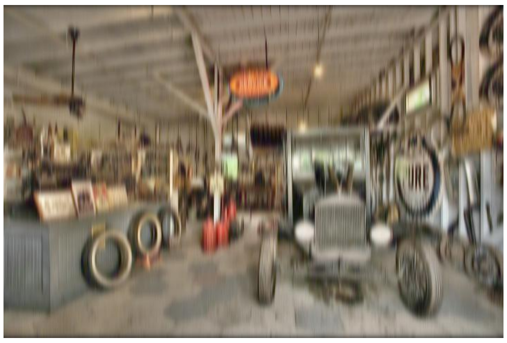

(b) Vertical blur

Figure 5: Application of Gaussian blur's Algorithm [3]

4 Local Laplacian Filters

(hard)

The Local Laplacian Filter [Paper] is an improvement of the Laplacian Filter that allow the user to manipulate the levels of details without create the halo effect on the edges. This can be possible thanks to the distinction that the algorithm does, handling them with two different ways.

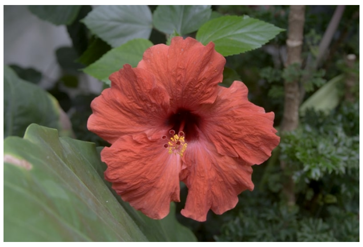


Figure 6: Input Image

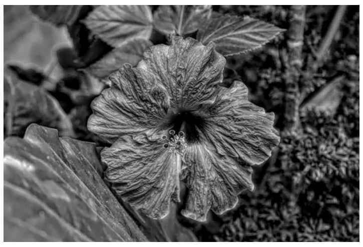


Figure 7: Grayscale

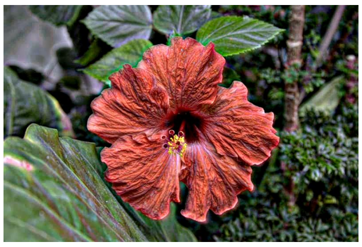


Figure 8: $\alpha = 0.25,\ \sigma = 0.3$

4.1 Implementation

For the implementation we took this code [github] as reference. Given an image I as input we firstly generate his Gaussian pyramid $G[I]$ that simply half the image each level until a certain minimum. Then for each pixel of each level $g_0 \leftarrow (l_0, y_0, x_0)$ we generate a subregion R_0 around the pixel and we remap that we the given parameters α, β and σ_r. The result $\hat{R}_0$ will increase or decrease the clarity of the subregion, to which we will create a new Gaussian pyramid $G_{l_0}[\hat{R}_0]$ with $\ell + 1$ levels and the respective Laplacian pyramid $L_{\ell_0}[\hat{R}_0]$. To generate this last one we use the Gaussian pyramid of the level ℓ and we subtract the expanded image of the level $\ell + 1$ so the resulted image will be a black image with only the details and the edges highlighted. Once we have the final Laplacian pyramid with all the levels we collapse them in one single image I' starting from the last level (the smaller one) of the pyramid, the one with the colors, we expand that and we sum it with the image of the level above up to the first level. For the test with this image srgb level has to be false.

Algorithm 4 Local Laplacian Filter's algorithm

```
Input: $image\ I,\ parameter\ \sigma_r,\ remappingfunction\ r$
Output: $image\ I'$
1: $G[I] \leftarrow createGaussianPyramid[I]$
2: for all $(\ell_0, y_0, x_0)$ do
3:     $g_0 \leftarrow G_{\ell_0}(x_0, y_0)$
4:     $R_0 \leftarrow subregion(\ell_0, x_0, y_0)$
5:     $\hat{R}_0 \leftarrow r_{g_0,\sigma_r}(R_0)$
6:     $L_{\ell_0}[\hat{R}_0] \leftarrow createLaplacianPyramid(\hat{R}_0)$
7:     $L_{\ell_0}[I'](x_0, y_0) \leftarrow L_{\ell_0}[\hat{R}_0](x_0, y_0)$
8: end for
9: $I' \leftarrow collapse(L_\ell[I'])$
```