Università della Svizzera italiana

Faculty of Informatics

Image and Video Processing **2023**

Students: Filippo Casari, Alessandro De Grandi

Report for Assignment 2

1. Spatial Filtering

1.1.

The minimim is obtained with $min = -1 \cdot 63 - 2 \cdot 63 - 2 \cdot 63 = -315$
The maximim is obtained with $max = 1 \cdot 63 + 3 \cdot 63 + 1 \cdot 63 = 441$

1.2.

To map values in the range $[a, b]$ to the range $[c, d]$:

$$f(x) = \frac{(x - \min(a,b))}{\max(a,b) - \min(a,b)} \cdot (d - c) + c$$

So to map values in the range $[-315, 441]$ to the range $[0, 63]$:

$$f(x) = \frac{(x - (-315))}{441 - (-315)} \cdot (63 - 0) + 0 = \frac{x + 315}{756} \cdot 63$$

1.3. BONUS

The filter is separable if the rank is one:

$$K = \begin{bmatrix} -1 & -2 & 0 \\ -2 & 0 & 3 \\ 0 & 3 & 1 \end{bmatrix}, rank(K) = 3$$

To obtain a separable approximation we chose the row and column of the filter that we think would best conserve the properties of the filter, for example the second row and second column, then multiplied them together:

$$K_a = c_2 * r_2 = \begin{bmatrix} 4 & 0 & -6 \\ 0 & 0 & 0 \\ -6 & 0 & 9 \end{bmatrix}, rank(K_a) = 1$$

To separate this kernel we used the matlab function svd:

$$[U, S, V] = \mathrm{svd}(K_a), \tag{1}$$

$$k_1 = U_{:,1}\sqrt{S_{1,1}}, \tag{2}$$

$$k_2 = V_{1,:}\sqrt{S_{1,1}}. \tag{3}$$

$$k_1 = \begin{bmatrix} -2 \\ 0 \\ 3 \end{bmatrix} k_2 = \begin{bmatrix} -2 & 3 & 0 \end{bmatrix} \tag{4}$$

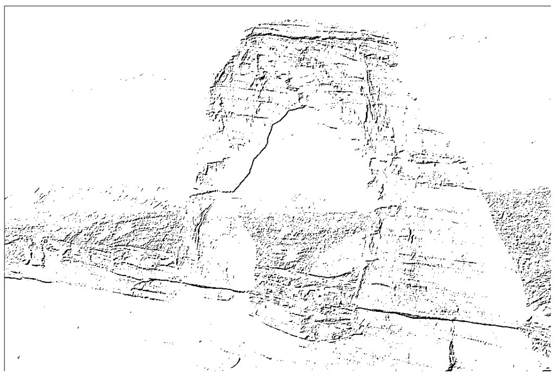


Figure 1: $Conv(image, K)$

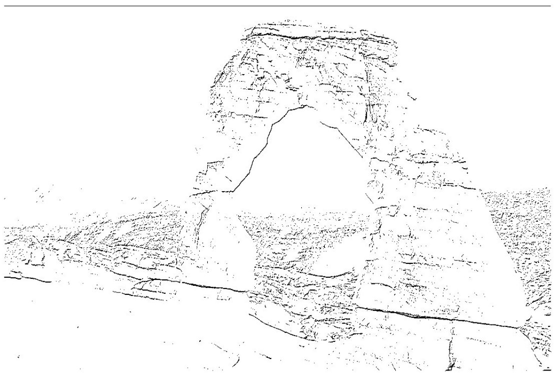


Figure 2: $Conv(image, K_1) + Conv(image, K_2))$

2. Combining linear operations

To combine into one single kernel the unsharpening procedure described by the following formula: $g_{sharp} = f + \gamma(f - h_{blur} * f)$
We first proceeded intuitively to combine a gaussian kernel H with a kernel S_2 that would sum the image to itself (with all zeros and a 2 in the middle). Noticing that using S_2-H is equivalent to summing the image with itself minus the blurred version, obtaining the equivalent of g_{sharp} when $\gamma = 1$.
γ is multiplying the details of the image so to obtain that effect in our combined kernel we used $S_1 + (\gamma S_1) - (\gamma H)$, where S_1 is a kernel with a 1 in the middle, but in the implementation this effectively means using the kernel -H and adding 1+γ at the center of the kernel.

$$\begin{gathered}\text{With:}\\ \sigma = 1\\ kernel\_size = 4\sigma + 1 = 5\\ \gamma = 2\end{gathered}$$

$$K = \begin{bmatrix} -0.0059 & -0.0266 & -0.0439 & -0.0266 & -0.0059 \\ -0.0266 & -0.1193 & -0.1966 & -0.1193 & -0.0266 \\ -0.0439 & -0.1966 & 2.6758 & -0.1966 & -0.0439 \\ -0.0266 & -0.1193 & -0.1966 & -0.1193 & -0.0266 \\ -0.0059 & -0.0266 & -0.0439 & -0.0266 & -0.0059 \end{bmatrix}$$

(a) $f + \gamma(f - h_{blur} * f)$

(b) $(S_1 + \gamma S_1 - \gamma H) * f$

3. Morphological Operations A

In order to find out which structuring element could lead to the correct output, we used a for loop to try both erosion and dilation techniques (one after the other) on theta image. Theta image was created by scratch. It turns out that either structuring "c" or "d" is correct for our goal. This is due to the fact that the line in the middle is removed with these structures since the only parts preserved are those matching the structuring element. Indeed, both c and d set to 1 only pixels which have neighbors (up, or up and down) equal to 1. This is definitely not the case of the middle line.

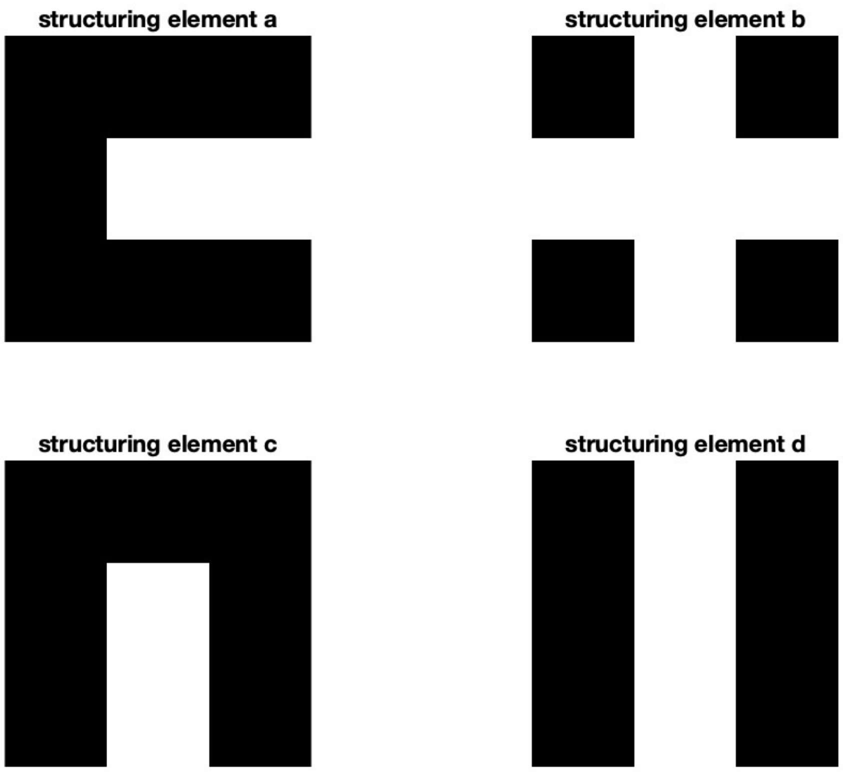


Figure 4: structuring elements

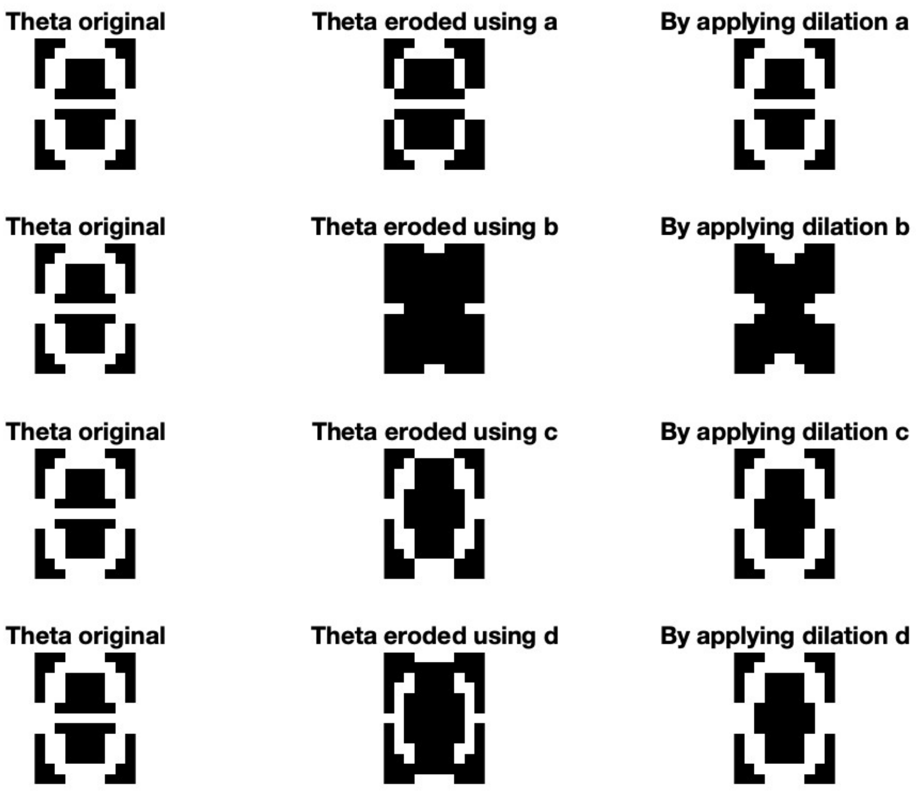


Figure 5: Results

4. Morphological Operations B

We found that by applying erosion to this image, it is shifted depending on the position of the foreground pixel. For instance, taking structuring element "8" which has the foreground pixel at the center bottom, the resulting image is shifted up. However, when performing erosion with structuring element 5 (foreground pixel in the center) the image is unchanged.

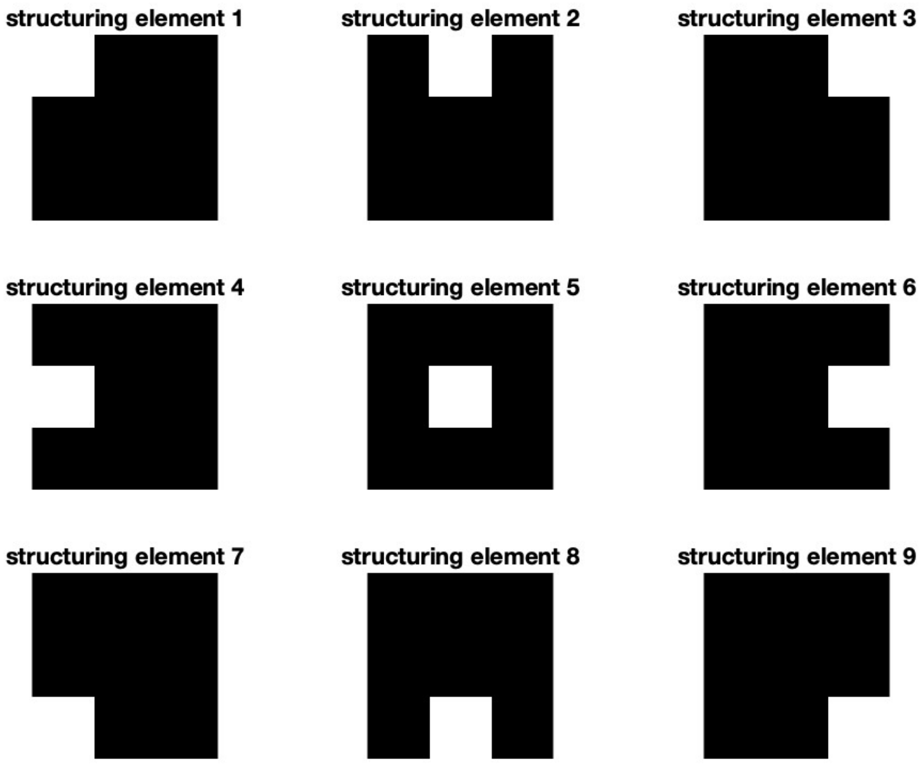


Figure 6: structuring elements

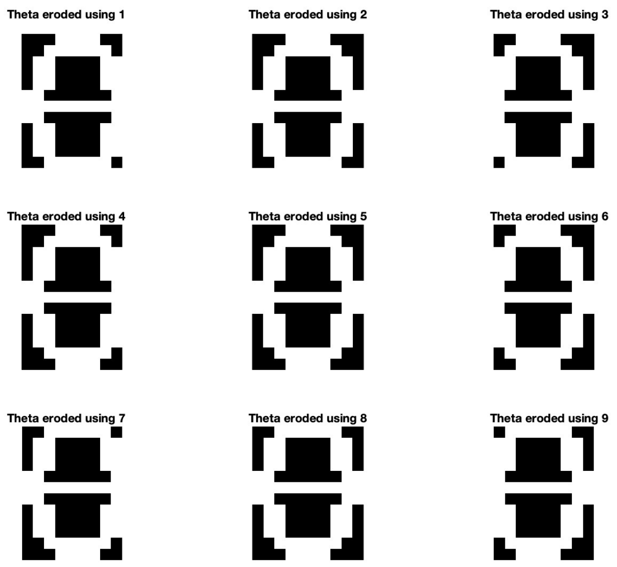


Figure 7: Results

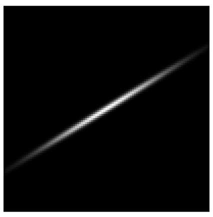
5. Linear Motion Blur Filter

5.1.

We implemented the function using the given formulas, and applied it to the given image

5.2.

Kernel size: $= 101$
$\sigma_X = 25$
$\sigma_Y = 1$
$\theta = 45°$



(a) Applied filter

(b) Anisotropic Gaussian result

Bonus

We found here that

$$\frac{1}{\sigma\sqrt{2\pi}}e^{-x^2/2\sigma^2} > \epsilon \tag{5}$$

$$\tag{6}$$

$$x < \sqrt{-2\sigma^2 ln(\epsilon\sigma\sqrt{2\pi})} \tag{7}$$

where ϵ refers to the weight given to the image's pixels that are farthest from the center pixel. If we set it to 0.00001 we get:

$$x < \sqrt{-2 * 25^2 ln(0.00001 * 25\sqrt{2\pi})} \tag{8}$$

$$\tag{9}$$

$$x < \sqrt{9219} \tag{10}$$

$$\tag{11}$$

$$x < 96 \tag{12}$$

where x is the radius of the gaussian kernel. Indeed, x must be at least almost 4 times sigma. Same approach would be for the y dimension.

6. Iterative filtering

As mentioned in the assignment the result by applying 10 times a smaller gaussian (sigma=2) filter 10 times, or only one bigger Gaussian filter is the same (fig. 9). the reason why it works in this way will be explained later in the bonus.

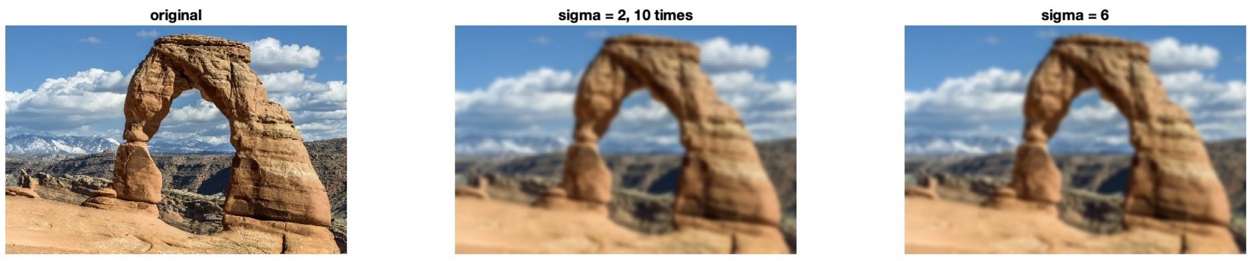


Figure 9: Results

The parameters for bilateral filter are based on the cropped image (the sky, fig. 10). Indeed, in order to smooth and get the sky more uniform we chose DegreeOfSmoothing according with the following formula:

$$DoS = 2 * std2(patch)^2$$

The output image of this non linear filtering is visible in the second image (fig. 11). As expected, the ground is smoothed as well.
Regarding the difference between applying a big bilateral filter, and applying multiple times a smaller bilateral filter is that the latter acts like a Gaussian blurring. Consequently, applying n times a smaller bilateral filter or applying 1 time a bigger filter is not the same.

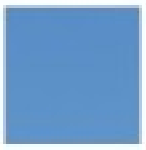


Figure 10: sky portion

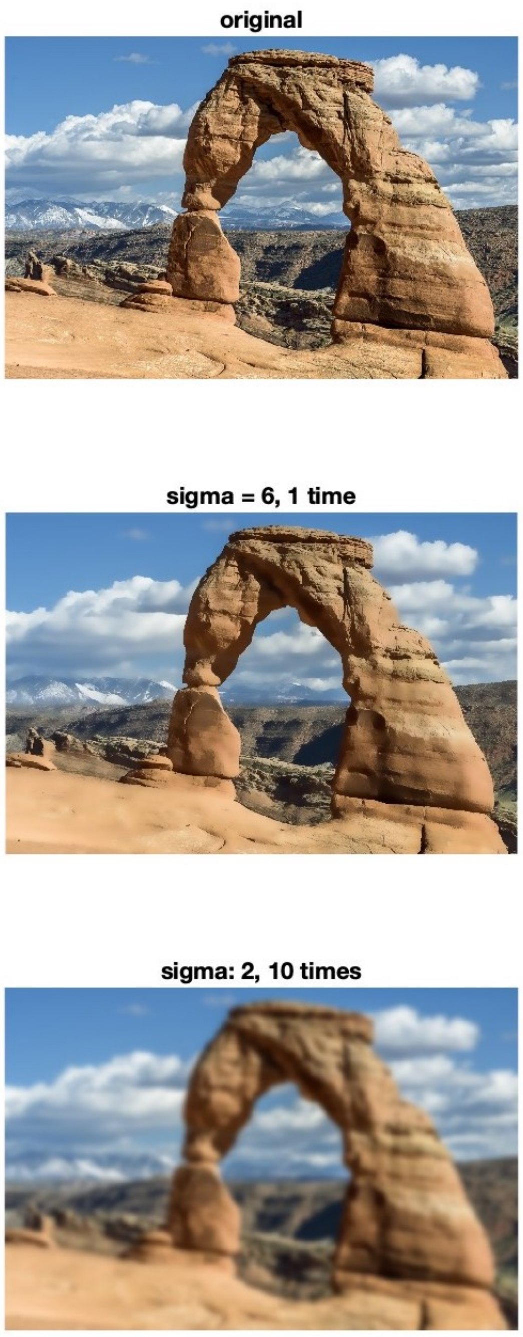


Figure 11: Bilateral filter results

Bonus

We can split f(x,y) in f(x) and f(y) in order to work for simplicity separately. Same splitting will be applied for Gaussian filters.
The 2 convolutions are applied according with these equations:

$$convImage1_x = f(x) * g_1(x, \gamma)$$

$$convImage2_x = convImage1 * g_2(x, \tau)$$

Substituting *convImage1* in the second equation we get:

$$convImage2_x = f(x) * g_1(x, \gamma) * g_2(x, \tau)$$

where "*" is the convolution operation, f(x) is the image, and g denotes the Gaussian filter. Since convolution operation is associative we can rewrite the equation as:

$$convImage2_x = f(x) * (g_1(x, \gamma) * g_2(x, \tau))$$

Since it is hard to compute the convolution between 2 Gaussian functions (g_1 and g_2), we switch to Fourier domain. According with this reference, we get the corresponding fourier function of the gaussian:

$$e^{-x^2/2\sigma^2} \quad \underset{Fourier}{\xrightarrow{\hspace{3em}}} \quad \sigma\sqrt{2\pi}e^{-\sigma^2\omega^2/2}$$

Moreover, multiplying by a constant in the space domain lead to a multiplication in the fourier domain as well (called Linearity property). Consequently, we divide both f(x) and F(x) above by the normalization factor $\sigma\sqrt{2\pi}$. Indeed,

$$\frac{1}{\sigma\sqrt{2\pi}}e^{-x^2/2\sigma^2} \quad \underset{Fourier}{\xrightarrow{\hspace{3em}}} \quad e^{-\sigma^2\omega^2/2}$$

Since convolution between g(x, γ) and g(x, τ) is only multiplication in the Fourier domain. We just multiply these 2 functions:

$$e^{-\gamma^2\omega^2/2} \cdot e^{-\tau^2\omega^2/2} = e^{-\omega^2\cdot(\gamma^2+\tau^2)/2}$$

Now, we have a form very simple to bring back to the spatial domain f(x).

$$e^{-\omega^2\cdot(\gamma^2+\tau^2)/2} \quad \underset{InverseFourier}{\xrightarrow{\hspace{5em}}} \quad \frac{1}{\sigma_{bigger}\sqrt{2\pi}}e^{-x^2/2\sigma_{bigger}^2}$$

Since $(\gamma^2 + \tau^2)$ would be the square of the goal σ_{bigger}, we just apply the square root to find the resulting sigma.

$$\sigma_{bigger} = \sqrt{(\gamma^2 + \tau^2)}$$

Finally, by substituting the new found sigma in the initial equation:

$$convImage2_x = f(x) * (g_1(x, \gamma) * g_2(x, \tau)) = f(x) * g_3(x, \sigma_{bigger})$$

To conclude, we have just proved that applying 2 Gaussian convolutions to an image is like applying a bigger Gaussian filter with a bigger σ. Same approach would be for f(y) and the corresponding convolutions with g(y, γ) and g(y, τ) extending it to 2 dimensions.
This demonstrates why applying in the previous exercise one single Gaussian filter with simga=6 and by applying 10 times a Gaussian filter with sigma=2 lead almost to the same result output image. In fact:

$$\sigma_2 = \sqrt{10\sigma_1^2} = about\ 6$$

where $\sigma_1 = 2$

6.1. Image Stylization

To perform image stylization, we converted the image to grayscale using the provided code, and applied a bilateral filtering for smoothing. This allows to perform edge detection on the smoothed image, to do this we tried different methods and kernels, and the best result was obtained using the matlab function 'edge' with the canny method. We then binarized the edges image to obtain a mask with the edges in black, and used dilation with the imdilate matlab function with a disk structuring element to make edges thicker, then erosion with a smaller disk. Finally we can apply the mask to the smoothed image reconverted to rgb.

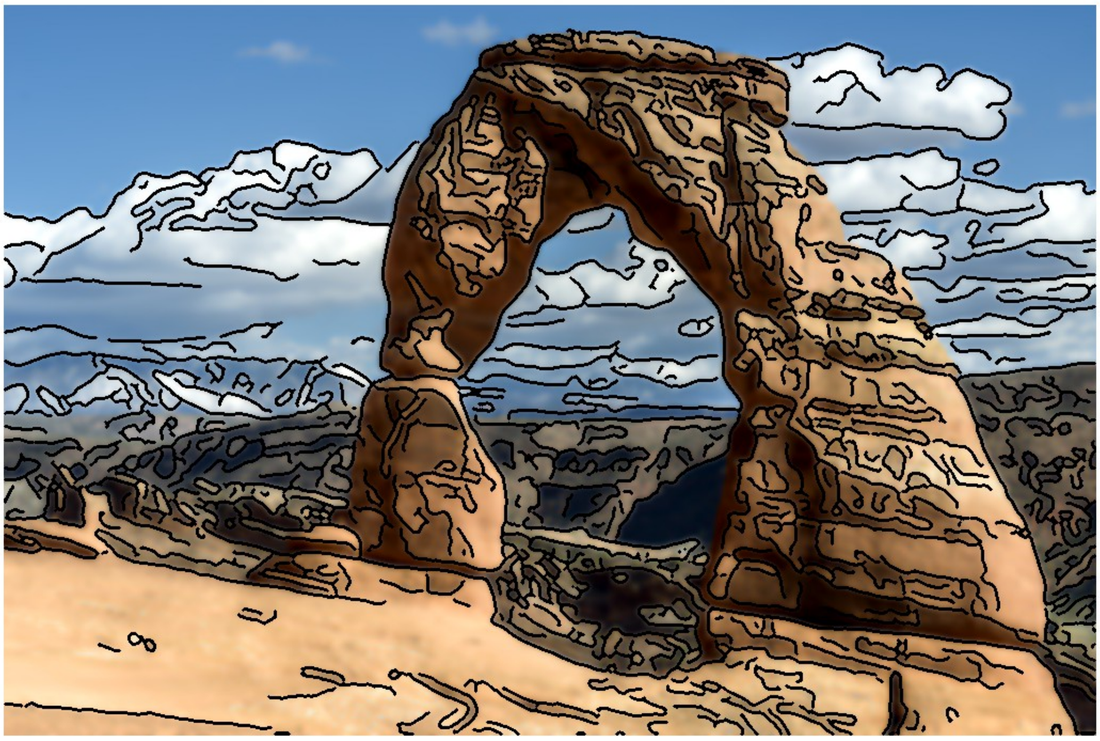


Figure 12: Stylization results