Assignment 1 — Image Abstraction using the Bilateral Gaussian Filter

Image Processing and Pattern Recognition

Deadline: November 12, 2024

1 Goal

In this exercise you will become familiar with and implement the bilateral Gaussian filter and other tools such as Difference-of-Gaussians (DoG) edge detection. In particular, you will implement an image-cartoonification pipeline based on the bilateral Gaussian filter.

2 Bilateral Gaussian Filter

In this assignment, we view images as maps from a multi-index, defined on a subset of $\mathbb{N}^2$, to some feature space. Specifically, $F : \Omega \to \mathcal{F}$ denotes an image from the image domain $\Omega = \{1, 2, \dots, M\} \times \{1, 2, \dots, N\}$ to the feature space $\mathcal{F}$. For instance, $\mathcal{F}$ may represent a three-tuple of RGB values in the interval $[0, 1]$, i.e. $\mathcal{F} = [0, 1]^3$. If the image is defined by LAB features, $\mathcal{F}$ is some other subset of $\mathbb{R}^3$. All norms $\|\cdot\|$ are 2-norms unless specified otherwise. In Fig. 1 we illustrate the notation graphically.

2.1 Gaussian Filtering

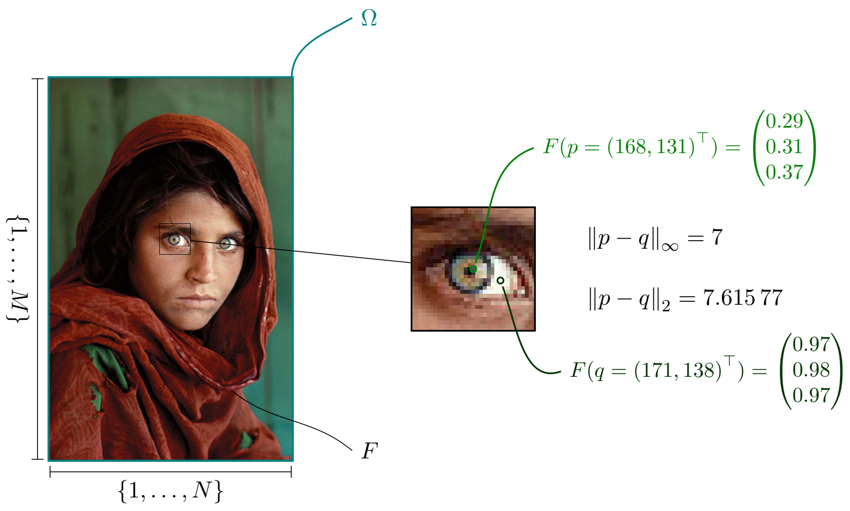
Consider some pixel location $p = (p_1, p_2)^\top \in \Omega$. Let $U : \Omega \to \mathcal{F}$ denote the result of filtering F with a Gaussian filter $g_\sigma : \mathbb{R}_+ \to \mathbb{R}$ of radius r. We can write

$$U(p) = (g_\sigma * F)(p) = \frac{\sum_{q \in \omega_p} g_\sigma(\|p - q\|) F(q)}{\sum_{q \in \omega_p} g_\sigma(\|p - q\|)}, \tag{1}$$

where the window $\omega_p = \{q : \|p - q\|_\infty \leq r\}$ describes all multi-indices q with ∞-distance at most r from p. If an element $q \in \omega_p$ is outside of the image domain Ω, we clamp it to the boundary of Ω (i.e. we use constant boundary conditions). The Gaussian weighting function with variance σ^2 is

$$g_\sigma(\gamma) = \exp\left(-\frac{\gamma^2}{2\sigma^2}\right), \tag{2}$$

where σ denotes the standard deviation of the filter (i.e. σ^2 is the variance). In practice, the filter radius r is typically chosen such that ω_p covers at least two standard deviations. We show an example (F, g_σ, U)-tuple in Fig. 2.



Ω
$F(p = (168, 131)^\top) = \begin{pmatrix}0.29\\0.31\\0.37\end{pmatrix}$
$\|p - q\|_\infty = 7$
$\|p - q\|_2 = 7.615\,77$
$F(q = (171, 138)^\top) = \begin{pmatrix}0.97\\0.98\\0.97\end{pmatrix}$
F
$\{1, \dots, M\}$
$\{1, \dots, N\}$

Figure 1: Graphical illustration of the notation: The image $F\colon \Omega \to \mathcal{F}$ is defined on the image domain $\Omega = \{1, \dots, 400\} \times \{1, \dots, 254\} \subset \mathbb{N}^n$. The image is represented as floating-point RGB tuples with maximum magnitude 1, thus the feature space $\mathcal{F}$ is $[0, 1]^3$. At the image location $q = (171, 138)^\top \in \Omega$, F attains the value $F(q) = (0.97, 0.98, 0.97)^\top$, which corresponds to the white sclera of the girl's eye.

2.2 Bilateral Filtering

As illustrated in Fig. 2, the Gaussian filter smooths the image irrespective of its content. To steer the filtering to be edge-aware, we use a bilateral approach as in [1]. In addition to weighting based on spatial distances, we additionally weight the pixels by their distance in feature space. In detail, at pixel location p we have

$$U(p) = \frac{\sum_{q \in \omega_p} b_{\sigma_s, \sigma_r}(\|p - q\|, \|F(p) - F(q)\|_{\mathcal{F}}) F(q)}{\sum_{q \in \omega_p} b_{\sigma_s, \sigma_r}(\|p - q\|, \|F(p) - F(q)\|_{\mathcal{F}})}, \tag{3}$$

with the filter weights $b_{\sigma_s, \sigma_r}\colon \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ given by

$$b_{\sigma_s, \sigma_r}(\gamma, \delta) = \exp\left(-\frac{\gamma^2}{2\sigma_s^2}\right) \exp\left(-\frac{\delta^2}{2\sigma_r^2}\right). \tag{4}$$

The norm $\|\cdot\|_{\mathcal{F}}$ is a natural norm in $\mathcal{F}$. For instance, if $\mathcal{F} \subset \mathbb{R}^3$, it is just the standard 2-norm (this is the case in this assignment). Observe that in Eq. (4), the standard Gaussian filter is additionally weighted by an exponential function that acts on the similarity of F at the pixel locations p, q. We show an example on an input image and the corresponding filter b_{σ_s, σ_r} in Fig. 4.

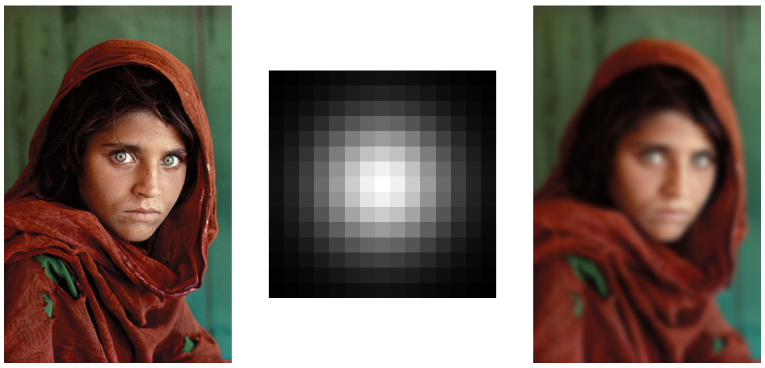


Figure 2: An example (input, filter, output)-tuple of Gaussian filtering with $\sigma = 3$. To account for at least two standard deviations in the filter, we used $r = 7$.

3 Image Abstraction

In this assignment, we closely follow [2] for implementing an image-abstraction pipeline. In detail, the image-abstraction pipeline consists of the following steps:

1. Image abstraction by recursive application of the bilateral filter Eq. (3),
2. edge extraction using smooth DoG filtering,
3. luminance quantization for emphasized cartoon-like features, and
4. the combination of the above for the final image.

All of these steps happen in the LAB color-space, i.e. $\mathcal{F}$ reflects a 3-tuple consisting of the real-valued luminance (typically between 0 and 100) and two real-valued entries describing the color (typically ranging between -128 to 127). Bilateral filtering was already discussed in the previous section, and we now focus the remaining parts.

3.1 DoG Edge Detection

The zero-crossings of the second derivative of any function are a good indicator of edges. As an example, consider the sigmoid function $x \mapsto (1 + \exp(-x))^{-1}$, which is shown along with its second derivative in Fig. 3. The Laplace filter approximates the sum of the second derivatives on a two-dimensional discrete grid. Thus, to find edges in images, a first idea would be to convolve the luminance channel with a Laplace filter and look for zero-crossings. However, we would find that the result is dominated by noise. To remedy this, it is advantageous to pre-convolve the image with a Gaussian filter.

A drawback of this approach is that we need to convolve the image twice. However, since convolutions are associative, we can first convolve the Gaussian filter with the Laplace filter. This effectively computed the second derivative of the Gaussian, which is known as the Mexican Hat (MH) function. Unfortunately, the MH filter is not separable and consequently does not enjoy the speedup of separable filters. Luckily, the MH function can be well approximated by the separable DoG filter (see Fig. 3, right).

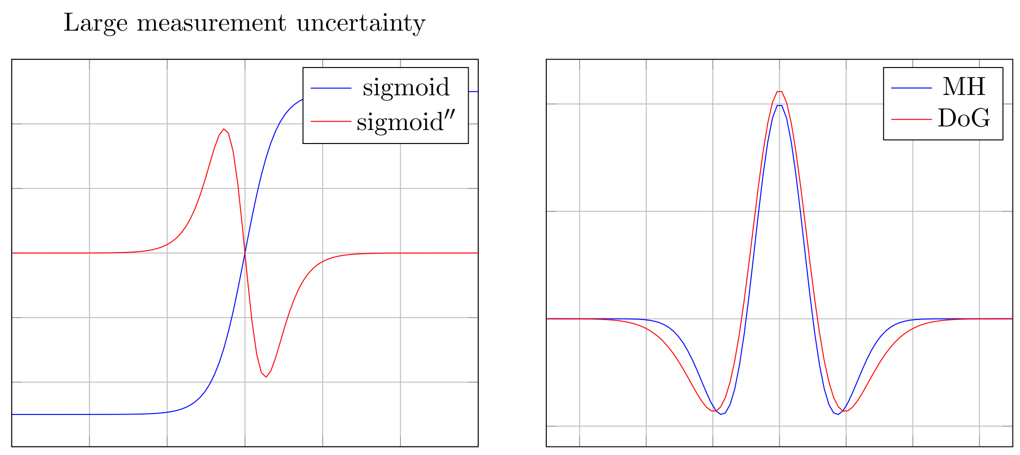


Figure 3: Left: The zero-crossing at $x = 0$ of the second derivative (red) of the sigmoid function (blue) indicate an "edge". Right: The MH function, i.e. the negative second derivative of a Gaussian function, is well approximated by a difference of two Gaussians with different standard deviation (DoG, blue).

To extract smooth edges from an image F we proceed as follows. Let F_l denote the luminance channel of F, i.e. $F_l : \Omega \to \mathbb{R}$. We compute two Gaussian-filtered images $S_e = g_{\sigma_e} * F_l$, $S_f = g_{\sigma_f} * F_l$, where σ_e selects the scale for edge detection and we fix $\sigma_f = \sqrt{1.6}\sigma_e$. Given S_e, and S_f, a straight-forward edge detector could threshold the difference image $(S_e - \tau S_f)$, where $\tau = 0.98$ relates to the sensitivity. To detect edges in a smooth manner, we introduce the smooth step-function $D_{\phi_e} : \mathbb{R} \to \mathbb{R}$:

$$D_{\phi_e}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 1 + \tanh(\phi_e x) & \text{else}, \end{cases} \tag{5}$$

where $\phi_e \in \mathbb{R}$ determines the sharpness of the smooth step. The edge image E is given by applying D to the DoG image $(S_e - \tau S_f)$ pixel-wise, i.e.

$$(\forall\, p \in \Omega) \quad E(p) = \big(D_{\phi_e} \circ (S_e - \tau S_f)\big)(p). \tag{6}$$

3.2 Luminance Quantization

To further cartoonify the images, we quantize the luminance channel. Specifically, we define evenly-spaced quantization steps $\mathcal{Q} = \{0, \Delta_l, \ldots, l_{\max} - \Delta_l, l_{\max}\}$, where $l_{\max} = 100$ denotes the maximum luminance and $\Delta_l = \frac{l_{\max}}{N_{\text{bins}}}$ is the bin width of the N_{bins} evenly spaced bins. To make transitions between the quantization steps smooth, we define the smooth step function $SS_{\phi_{\mathcal{Q}}} : \mathbb{R} \to \mathbb{R}$:

$$SS_{\phi_{\mathcal{Q}}}(x) = \mathcal{Q}_{\hat{\imath}(x)} + \frac{\Delta_l}{2} \tanh\big(\phi_{\mathcal{Q}}(x - \mathcal{Q}_{\hat{\imath}(x)})\big), \tag{7}$$

where $\hat{\imath}(x) = \arg\min_i |x - \mathcal{Q}_i|$ and $\phi_Q \in \mathbb{R}$ defines the sharpness of the step. Then, we compute

$$(\forall\, p \in \Omega) \quad Q(p) = \big(SS_{\phi_Q} \circ F_l\big)(p), \tag{8}$$

where $Q : \Omega \to \mathbb{R}$ is the quantized luminance image.

3.3 The Algorithm

Having defined the building blocks, we are now able to state the final algorithm for image abstraction. We denote with F the input image, and with $F^{(k)}$ the k-fold iterative application of the bilateral Gaussian Eq. (3) onto F. To minimize noise in the edge image, we extract edges with Eq. (6) after N_e iterations, but continue to abstract the image for a total of N_b iterations (in this assignment, $N_e = 2$ and $N_b = 4$). To get the quantized luminance, we apply Eq. (8) to $F^{(N_b)}$. Denoting the edge image and luminance-quantized image E and Q respectively (keep in mind that these only have one feature, namely the luminance), we define the final abstracted image $A : \Omega \to \mathbb{R}^3$ by

$$(\forall\, p \in \Omega) \quad A(p) = \begin{pmatrix} E(p)Q(p) \\ \big(F^{(N_f)}(p)\big)_2 \\ \big(F^{(N_f)}(p)\big)_3 \end{pmatrix}. \tag{9}$$

This is illustrated in Fig. 4.

4 Tasks

You are provided with the skeleton file `abstraction.py` in which you find the parts you should implement. In essence, you have to implement Eq. (3), Eq. (6) and Eq. (8).

4.1 Tasks

1. Implement the required equations.
2. Describe the influence of the parameters σ_r, σ_d, σ_e, N_{bins}, and ϕ_Q, and show some representative examples of cartoonified images. The values of all parameters in the skeleton file are good starting points.
3. Try the algorithm on your own images and describe the results. Did you have to tweak any parameters, and if yes, which and how?

4.2 Notes and Hints

1. Do not change the `import` statements.
2. We provide a reference output for the parameters that ship with the script. Your output should be the same up to numerical precision.
3. For implementing the bilateral Gaussian filter efficiently, have a look at `numpy.lib.stride_tricks.sliding_window_view`.

4. Use `skimage.filters.gaussian` to implement edge detection.
5. `numpy.ndarray.min` or `numpy.ndarray.argmin` might be useful for implementing the luminance quantization.

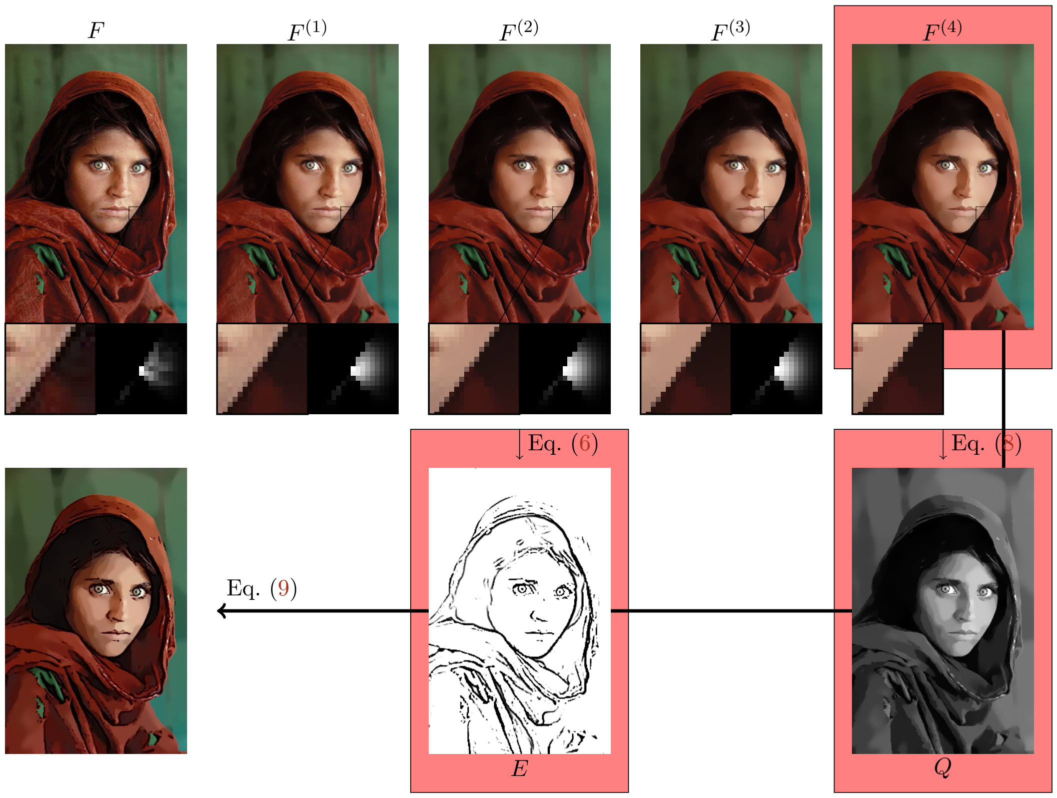


Figure 4: The image abstraction pipeline. The insets show the filter kernels (and the corresponding image region) used for bilateral filtering.

Acronyms

DoG Difference-of-Gaussians 1, 3, 4

MH Mexican Hat 3, 4

References

[1] Carlo Tomasi and Roberto Manduchi. "Bilateral filtering for gray and color images". In: *Sixth International Conference on Computer Vision* (1998), pp. 839–846.

[2] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. "Real-time Video Abstraction". In: *ACM Transactions on Graphics* 25.3 (July 2006), pp. 1221–1226.