Image Processing and Pattern Recognition—Homework 1

Lukas Glaszner—01430043

November 12, 2020

1 Dehazing

In this part of the exercise we implement the dehazing algorithm explained on the assignment sheet. The so-called dark channel is defined as

$$J_i^{dark} = \min_{c \in \{r,g,b\}} \left\{ \min_{k \in \omega_i} J_k^c \right\}, \tag{1}$$

we implement this using the function `minimum_filter()` from `scipy.ndimage`. Subsequently, we compute the transmission using the formula

$$t_i = 1 - u \min_{c \in \{r,g,b\}} \left\{ \min_{k \in \omega_i} \frac{I_k^c}{A^c} \right\}, \tag{2}$$

where $u \in (0, 1]$ is a free parameter and A is an estimate of the ambient light. To calculate A we select the brightest $0.1\,\%$ of pixels in the dark image and among these calculate the mean of the brightest $1\,\%$ of pixels in the input image I. At this point we use a guided image filter to refine the transmission map—i.e. the input is the estimated transmission per Eq. (2) and the guidance image consists of the mean of the three color channels of the input image. Finally, the output is calculated by

$$J_i = \frac{I_i - A}{\max(t_i, t_0)} + A, \tag{3}$$

where $t_0 = 0.1$.

Figure 1 shows the input image I; the patch size is set to $w = 15$ and the guided image filter uses $r = 25$ and $\epsilon = 10^{-9}$. The output using both the estimated and the refined transmission maps are shown in Figure 2. Haze is removed well in most parts of both images, in fact they are very similar

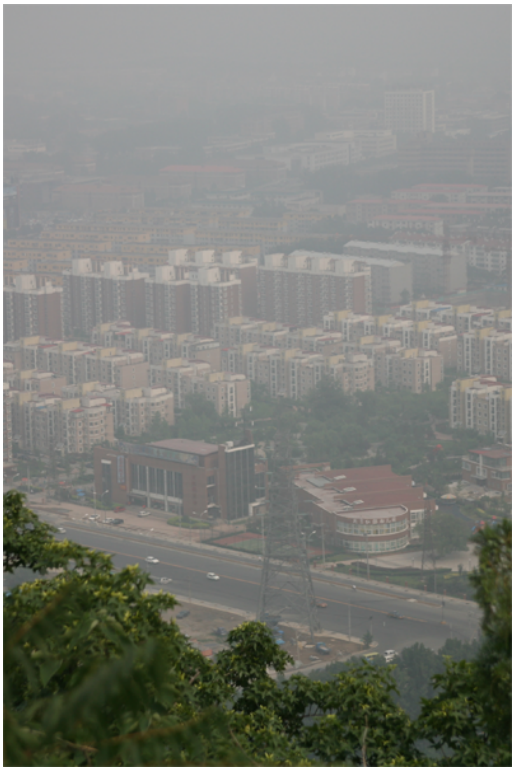


Figure 1: The input image

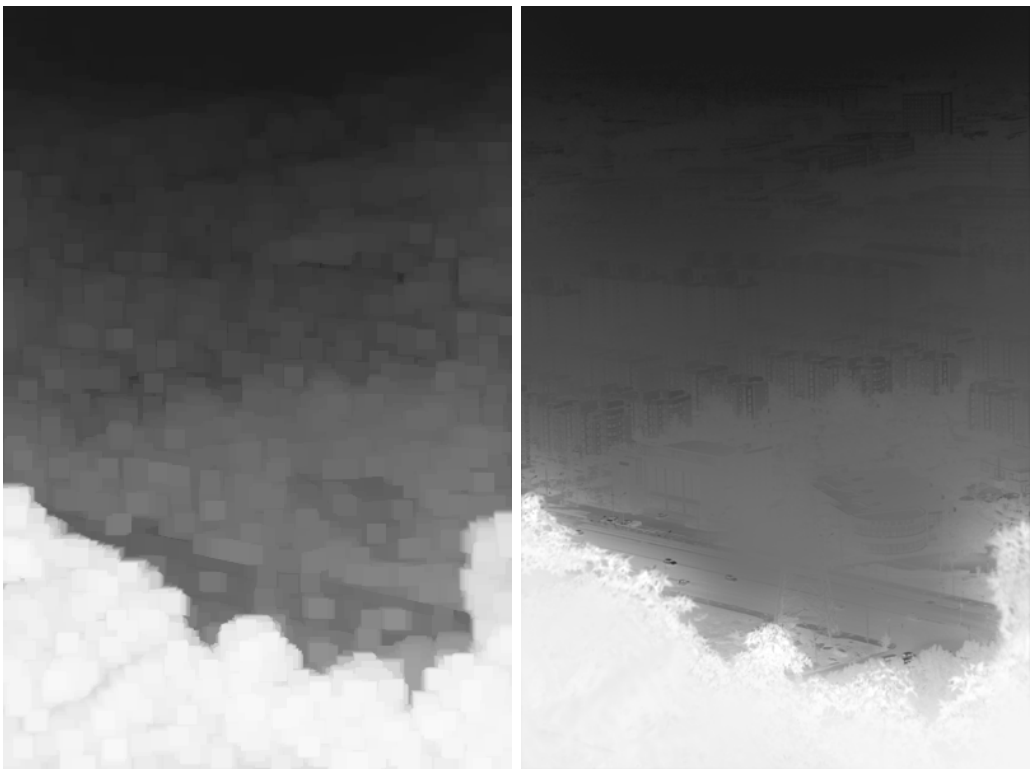
except in the border region of the trees. There we can see very well why the refinement of the transmission map is necessary.

In order to further investigate the effect of the guided image filter on the transmission maps, we visualize them in Figure 2. The appearance of the estimated transmission map—i.e. the squares—can be explained by the fact that it is estimated using patches of size $w = 15$. The guided image filter preforms soft matting on the estimate, leading to a much mote detailed result.

(a) The output using the estimated transmission map.

(b) The output using the refined transmission map.

Figure 2: When using the estimated transmission we can see that there is a clearly visible border near the trees, where we can still see haze. The refined transmission map reduces haze visible around the trees and leads to a smoother transition.



(a) The estimated transmission map (b) The refined transmission map

Figure 3: The estimated transmission map consists of many squares and is very coarse, the refined transmission map does not show these squares and is a lot more detailed.

Figure 4: Dark channel.

2 Flash/No-Flash Denoising

We can also use the guided image filter to implement flash/no-flash denoising, where we use an image taken with flash as a guide for denoising a picture taken without flash. In order to filter a color image, we apply the guided image filter to each color channel separately and subsequently recombine the result.

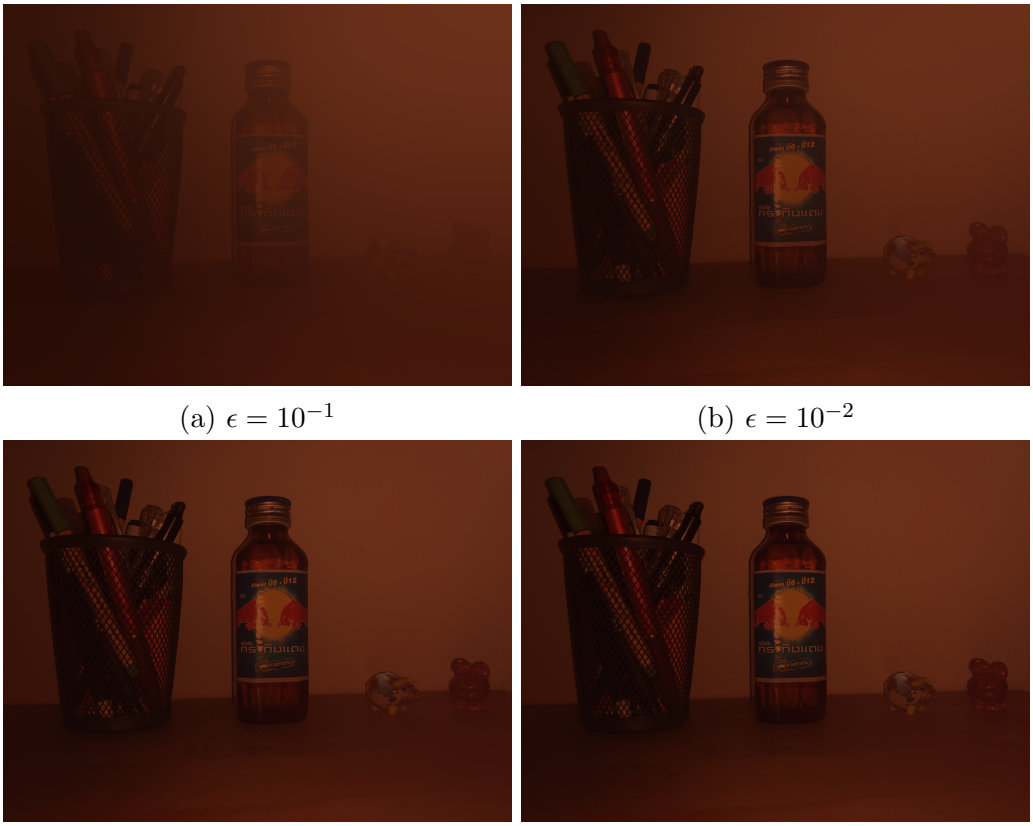
(a) No-flash input (b) Flash input (c) Output

Figure 5: The estimated transmission map consists of many squares and is very coarse, the refined transmission map does not show these squares and is a lot more detailed.

Figure 5 presents both the no-flash and flash input (taken on an iPhone X), as well as the output image, where the guided image filter parameters are $r = 1000$ and $\epsilon = 10^{-3}$. In order to be able to take a picture without flash under low lighting conditions, the camera sensor has to have a very high sensitivity—i.e. a high ISO value. However, this leads to a very noisy image, as is apparent in Figure 5a. Taking an image using flash reduces the amount of noise drastically, however, the colors of the image become unnatural—compare Figure 5b. Applying the guided image filter greatly reduces the noise in the no-flash image, while preserving the true colors, because it uses the flash image merely as the guide.

Next we would like to investigate the influence of the filter parameters ϵ and r; to do so we first compute the output image for various parameters ϵ—shown in Figure 6. In the case of edge preserving filtering, ϵ serves as a threshold, where if $\sigma_p^2 \gg \epsilon$ the pixels are considered to be on the same side of an edge and hence the corresponding filter kernel value is large; if $\sigma_p^2 \ll \epsilon$ they are considered to be on opposing sides and the corresponding value is very low. In this case $F \neq I$, however, we can still use this insight to explain the behavior of our algorithm. If ϵ is large, the variance needs to be very large to preserve an edge; if the value decreases, pixels with lower variance between them are also considered to be on opposing sides of an edge.

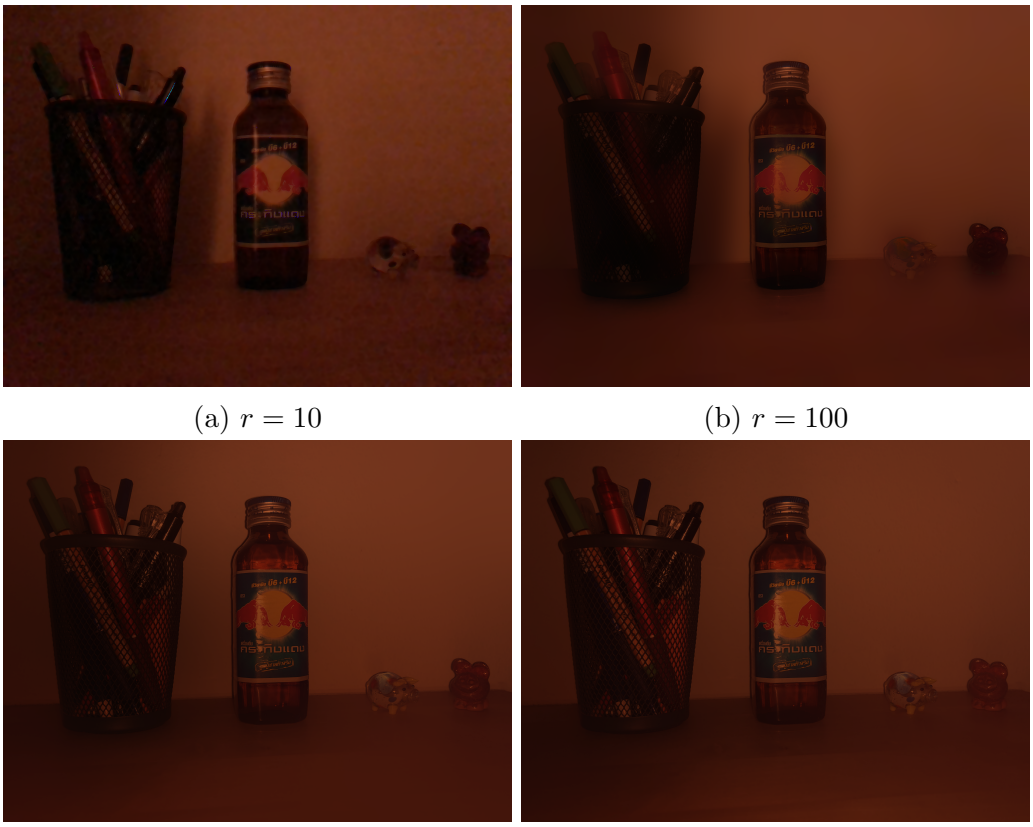
Figure 7 corresponds to the size of the window the filter operates on, it thus needs to be chosen considering the input image size, which is 3024×4032 in this example. TODO: EXPLAIN



(a) $\epsilon = 10^{-1}$ (b) $\epsilon = 10^{-2}$

(c) $\epsilon = 10^{-3}$ (d) $\epsilon = 10^{-9}$

Figure 6: If ϵ is very large, the output image is hazy; contrast is very low. This problem vanishes as the value decreases; we can also see that for any value below 10^{-3} the result is more or less the same.



(a) $r = 10$ (b) $r = 100$

(c) $r = 500$ (d) $r = 1000$

Figure 7: If r is very low, the denoising algorithm performs very poorly; as the value increases noise is reduced further, however, edges are blurred initially. Edges are preserved very well with even larger values of r; the difference between $r = 500$ and $r = 1000$ seems to be mostly in the image colors.