Digital Image Processing HW2

Yuval Goshen 205810179

Tuval Gelvan 312419971

Question 1

We can use the definitions of convolution:

$$h[n] = \big(f(x) * p_H(x)\big)(n) = \int f(t)p_H(x-t)dt|_{x=n} = \int f(t)p_H(n-t)dt$$

$$l[n] = \ \big(f(x) * p_L(x)\big)(n) = \int f(t)p_L(x-t)dt|_{x=n} = \int f(t)p_L(n-t)dt$$

Question 2

Again, we can use the definition of convolution and subsampling on the given formula:

$$l[n] = \downarrow_\alpha (h * k)[n] = (h * k)[\alpha n] = \sum_m h[m]k[\alpha n - m]$$

Question 3

We can substitute $l[n]$ and $h[m]$ in the equation we got in question 2 with their expression in question 1:

$$l_{\frac{1}{\alpha}\mathbb{Z}^2}[n] = \int f(t)p_L(n-t)dt, \qquad l_{\mathbb{Z}^2}[n] = \sum_m \int f(t)p_H(m-t)dt\, k[\alpha n - m]$$

But we want them both to be on the same lattice $\frac{1}{\alpha}\mathbb{Z}^2$, therefore we must move the second expression to this lattice:

$$l_{\mathbb{Z}^2}[n] = \downarrow_\alpha \sum_m \int f(t)p_H\left(\frac{m}{\alpha} - t\right)dt\, k\left[n - \frac{m}{\alpha}\right]$$

$$l_{\frac{1}{\alpha}\mathbb{Z}^2}[n] = \sum_m \int f(t)p_H\left(\frac{m}{\alpha} - t\right)dt\, k\left[n - \frac{m}{\alpha}\right]$$

Now we can get:

$$\int f(t)p_L(n-t)dt = \sum_m \int f(t)p_H\left(\frac{m}{\alpha} - t\right)dt\, k\left[n - \frac{m}{\alpha}\right]$$

$$= \sum_m (f * p_H)\left[\frac{m}{\alpha}\right]k\left[n - \frac{m}{\alpha}\right] = \sum_{m' = \alpha m} (f * p_H)[m']k[n - m']$$

$$(f * p_L)[n] = \big((f * p_H) * k\big)[n] = \big(f * (p_H * k\,)\big)[n]$$

We use the assumption that it holds for any $f(t)$ and get:

$$p_L[n] = (p_H * k)[n] = \sum_m p_H[m]k[n-m]$$

Question 4

We know that:

$$p_L(x) = (p_H * k_c)(x)$$

We can apply the Fourier transform so the convolution becomes pointwise multiplication:

$$P_L(\xi) = P_H(\xi) K_c(\xi)$$

Assuming $P_H(\xi) \neq 0$ we can divide by it and get:

$$K_c(\xi) = \frac{P_L(\xi)}{P_H(\xi)}$$

We can use $p_H(x) = \alpha p_L(\alpha x)$ because they are sampled on a lattice scaled by α. Using the Fourier transform properties we can say that $P_H(\xi) = \alpha \frac{1}{\alpha} P_L\left(\frac{\xi}{\alpha}\right) = P_L\left(\frac{\xi}{\alpha}\right)$. Finally we got:

$$K_c(\xi) = \frac{P_L(\xi)}{P_L\left(\frac{\xi}{\alpha}\right)}$$

Question 5

When P_L is a $sinc$, it's Fourier transform is the box function, so we get:

$$K_c(\xi) = \frac{box(\xi)}{box\left(\frac{\xi}{\alpha}\right)} = box(\xi) = P_L(\xi)$$

(we used the fact that the support of the box in the denominator is larger then the support of the numerator, and their value at the support is one)

Therefore the assumption holds for a $sinc$.

In the gaussian case we get:

$$p_L(x) = \frac{1}{2\pi|\Sigma|^{0.5}} \exp\left(-\frac{1}{2} x^T \Sigma^{-1} x\right)$$

In the Fourier domain, we get gaussian as well, using this formula from the lecture:

$$e^{-\mathbf{x}^{\top}\mathbf{C}^{-1}\mathbf{x}} \overset{\mathcal{F}}{\longleftrightarrow} \frac{1}{\sqrt{\det \mathbf{C}}} e^{-\pi \xi^{\top} \mathbf{C} \xi}$$

$$P_L(\xi) = \frac{1}{2\pi|\Sigma|^{0.5}} \mathcal{F}\left\{\exp\left(-x^T \frac{1}{2} \Sigma^{-1} x\right)\right\} = \frac{1}{2\pi|\Sigma|^{0.5}} \frac{1}{\sqrt{2|\Sigma|}} \exp(-2\pi \xi^T \Sigma \xi)$$

$$K_c(\xi) = \frac{\frac{1}{2\pi|\Sigma|^{0.5}}\frac{1}{\sqrt{2|\Sigma|}}\exp(-2\pi\xi^T\Sigma\xi)}{\frac{1}{2\pi|\Sigma|^{0.5}}\frac{1}{\sqrt{2|\Sigma|}}\exp\left(-\frac{2\pi\xi^T\Sigma\xi}{\alpha^2}\right)} = \exp(-2\pi\xi^T\Sigma\xi(1-1/\alpha^2)) =$$

Since it is isotropic gaussian, Σ is diagonal, therefore we can switch the order of multiplication and get:

$$= \exp\left(-2\pi\Sigma\left|\left|\xi\right|\right|_2^2\left(1-\frac{1}{\alpha^2}\right)\right)$$

We can see that although K_c and P_L are both gaussian, they are different because they have different scales and variances, therefore the assumption doesn't hold.

In real life a gaussian is more likely for p_L since it decays without fluctuations, which is more likely how the camera sensor works. In addition, in real life we can't create a perfect window filter on the frequency, and it is more likely to look more like a gaussian.

Question 6

$$P(q_i|k, j_i) = P\left(\downarrow_\alpha\left(p_{j_i} * k\right) + n_i\right)$$

Since j_i is uniformly distributed over the indexes 1-N:

$$P(q_i|k) = \frac{1}{N}\sum_{j=1}^{N} P\left(\downarrow_\alpha\left(p_j * k\right) + n_i|k\right)$$

Now we would like to look at $l = \{q_1 \dots q_M\}$, due to independency, we can multiply the probabilities:

$$P(l|k) = \prod_{i=1}^{M}\frac{1}{N}\sum_{j=1}^{N} P\left(\downarrow_\alpha\left(p_j * k\right) + n_i|k\right)$$

We would like to find k that maximizes this distribution:

$$k^* = \underset{k}{\operatorname{argmax}}\, P(l|k) = \underset{k}{\operatorname{argmax}}\log\left(P(l|k)\right) = \underset{k}{\operatorname{argmax}}\ \sum_{i=1}^{M}\log\left(\sum_{j=1}^{N} P\left(\downarrow_\alpha\left(p_j * k\right) + n_i|k\right)\right)$$

Now we know the distribution of the noise, there fore we can say:

$$\sum_j P\left(q_i = \downarrow_\alpha\left(p_j * k\right) + n_i|k\right) = \sum_j P\left(n_i = \downarrow_\alpha\left(p_j * k\right) - q_i|k\right)$$
$$= \frac{1}{constant}\sum_j \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha\left(p_j * k\right) - q_i\right)^2\right)$$

Finally, we can plug that into k^*:

$$k^* = \underset{k}{\operatorname{argmax}} \sum_{i=1}^{M} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right)$$

Question 7

We want Dk to have normal distribution. Therefore:

$$P(Dk = x) = \frac{1}{\sqrt{2\pi\sigma_D^2}} \exp\left(-\frac{1}{2\sigma_D^2}(x)^2\right)$$

$$P(k = D^{-1}x) = \frac{1}{\sqrt{2\pi\sigma_D^2}} \frac{1}{constant} \exp\left(-\frac{1}{2\sigma_D^2}(D^{-1}x)^2\right)$$
$$= \frac{1}{\sqrt{2\pi\sigma_D^2}} \frac{1}{constant} \exp\left(-\frac{1}{2\sigma_D^2} D^{-1}x^2 D^{-1^T}\right)$$

We can now derive the MAP

$$k^* = \underset{k}{\operatorname{argmax}}\, P(k|l) =$$
$$\underset{k}{\operatorname{argmax}} \frac{P(l|k)P(k)}{P(l)} =$$
$$\underset{k}{\operatorname{argmax}}\, P(l|k)P(k) =$$
$$\underset{k}{\operatorname{argmax}}\ \log\big(P(l|k)\big) + \log\big(P(k = D^{-1}x)\big) =$$
$$\underset{k}{\operatorname{argmax}} \sum_{i=1}^{M} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right) + \log\left(\frac{1}{\sqrt{2\pi\sigma_D^2}} exp\left(-\frac{1}{2\sigma_D^2}(D^{-1}x)^2\right)\right) =$$
$$\underset{k}{\operatorname{argmax}} \sum_{i=1}^{M} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right) - (D^{-1}x)^2 =$$
$$\underset{k}{\operatorname{argmax}} \sum_{i=1}^{M} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right) - k^2$$

Question 8

Deriving the first term:

$$\frac{\partial}{\partial k} \sum_{i=1}^{M} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right) =$$
$$\sum_{i=1}^{M} \frac{\partial}{\partial k} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right) =$$

$$\sum_{i=1}^{M} \frac{\partial}{\partial k} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right) =$$

$$\sum_{i=1}^{M} \frac{\sum_{j=1}^{N} \frac{\partial}{\partial k} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)}{\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right)} = (*)$$

Now, in order to derive the down sampling and the convolution operators, we will define them as a matrix T as shown in the FAQ:

$$\downarrow_\alpha (p_j * k) = T_j k$$

We can now procced:

$$(*) = \sum_{i=1}^{M} \frac{\sum_{j=1}^{N} \frac{\partial}{\partial k} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)}{\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)\right)} =$$

$$\sum_{i=1}^{M} \frac{\sum_{j=1}^{N} -\frac{1}{\sigma_N^2} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)(T_j k - q_i) T_j}{\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)\right)}$$

Now lets look at the prior:

$$\frac{\partial}{\partial k} k^2 = 2k$$

So we got:

$$\frac{\partial}{\partial k}\left(\sum_{i=1}^{M} \log\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(\downarrow_\alpha (p_j * k) - q_i\right)^2\right)\right) - k^2\right)$$

$$= \sum_{i=1}^{M} \frac{\sum_{j=1}^{N} -\frac{1}{\sigma_N^2} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)(T_j k - q_i) T_j}{\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)\right)} - 2k$$

To get the optimal k we set the derivative to zero and get:

$$k = \frac{1}{2} \sum_{i=1}^{M} \frac{\sum_{j=1}^{N} -\frac{1}{\sigma_N^2} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)(T_j k - q_i) T_j}{\left(\sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_N^2}\left(T_j k - q_i\right)^2\right)\right)}$$

An iterative algorithm:

1. Initialize some k
2. Until convergence :

a. Calculate $T_j k$ for each j

b. Update $k = \frac{1}{2}\sum_{i=1}^{M} \frac{\sum_{j=1}^{N} -\frac{1}{\sigma_N^2}\exp\left(-\frac{1}{2\sigma_N^2}(T_j k - q_i)^2\right)(T_j k - q_i)T_j}{\left(\sum_{j=1}^{N}\exp\left(-\frac{1}{2\sigma_N^2}(T_j k - q_i)^2\right)\right)}$

Where D is Laplacian operator as circulant matrix

Question 9

Since the sampling is done on a zoomed in image by α, which is exactly the scale between the lattices $\mathbb{Z}^2\ and \frac{1}{\alpha}\mathbb{Z}^2$ we can look at it as sampling using p_H from the zoomed in image.

$$z[n] = \int_x \frac{1}{\alpha} f\left(n - \frac{x}{\alpha}\right) p_H\left(\frac{x}{\alpha}\right) dx = \int_x f(n-x) p_H(x) dx = (f * p_H)[n]$$

Note that here f is not the original image but the zoomed in image.

Question 10

We have shown in question 2 that generally:

$$l[n] = \downarrow_\alpha (h * k)[n]$$

We can use the same result on the zoomed in sampled image and use the fact that $h[n] = (f * p_H)[n]$ and get:

$$l[n] = \downarrow_\alpha (z * k)[n]$$

Question 11

To get high resolution patch, we need to estimate the kernel K that transforms high resolution image to a low-resolution image. In order to do that, we can use the fact that small pattern appear in different scales, apply an initial guess for the kernel to the low-resolution image, which will create a lower resolution image, so we have l , the original low-res image and l^α the even lower one. Now we can address l as high-res image related to l^α, and sample patches from it. Then, for each patch in l we find KNN patches in l^α (we know there should be some similar patches) and use the algorithm from Q8 to estimate a better k iteratively.
Finally, when we have the optimal k, we can use it on patches from l to generate patches from h and get $\{p_1, p_2 \dots p_n\}$:

$$p_i = \underset{p_i}{\operatorname{argmin}} \|\downarrow_\alpha (p_i * k) - q_i\|_2^2 + \lambda P(p_i) = \underset{p_i}{\operatorname{argmin}} \|Dp_i - q_i\|_2^2 + \lambda P(p_i)$$

- Here we added regularization term which depend on a prior $P(h)$ we can choose
- We used D as the matrix that performs down sampling and convolution. In order to do that on a patch, it can be reshaped as a vector then multiplied by the matrix

Question 12

We can use what we did in question 11, but instead for a small patch, we can use the kernel to estimate the entire image:

$$h = \underset{h}{\operatorname{argmin}} \|Dh - l\|_2^2 + \lambda P(h)$$

This is a least square problem with regularization.

Practical Part Brief Explanation

In the practical part we implemented what we explained here and what we read on the paper.

Files:

- conv_as_matrix.py – Implements functions to convert image patches into convolution and down-sampling matrix (with a kernel stacked to 1d array)
- wet2_utils.py - Implements so utility functions such as extracting patches from an image
- wet2.py – Implements the algorithms and runs it. It saves the result images

The high-res images we used is the original image (aka, sampled with ratio of 1), and the low-res image is down-sampled from the high-res image with $\alpha = 3$.
In the algorithm, both small and large patches from the (low-res) image were sampled from a grid which made the image bank quite large. As a result, the algorithm converged fast (We ran 5 iterations) but each iteration took a few seconds.

PSNR values are in the files name. They are mostly around 17-20.