Homework 6

Assigned on June 1, 2020
Due on June 19, 2020

Learning Outcomes:

After this homework, you should be able to:

(a) Read images in MATLAB and apply linear filters on them.

(b) Construct a computer vision pipeline for determining different region properties, after segmentation.

Tasks

Problem 1
7 points

(a) Linear image filters are implemented through 2D convolutions. The convolution operation was defined in class as:

$$\mathbf{O}[u,v] = \sum_{(i,j)\in\mathcal{W}} \mathbf{I}[u-i, v-j]\,\mathbf{K}[i,j], \quad \forall (u,v) \in \mathbf{I},$$

where $\mathbf{K} \in \mathbb{R}^{w\times w}$ is the convolution kernel. The center of the window and kernel is considered to be coordinate $(0,0)$ and $i,j \in [-h,h]$.

Work out (using pen and paper) the 2D convolution for the input image and kernel shown in Figure 1. The output image should be of the same size as the input image. You can choose to find output at the boundary by assuming any of the stated techniques in the slides (zero padding, loop around, etc.)

The convolution formula stated above can also be viewed as a graphical procedure, whose steps are outlined below:

1. Flip the kernel in both the horizontal and vertical directions about the center pixel.
2. Align the now flipped kernel over the input image, such that center pixel of the kernel is aligned with pixel location for which you want to compute output value.

1	2	3
4	5	6
7	8	9

n \ m	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

Figure 1: **Left:** Input image; **Right:** Kernel

3. Multiply the kernel values with overlapping input values respectively.
4. Sum up all the obtained products to find the output value.

(b) The kernel of a Gaussian filter is:

$$\mathbf{G}[i,j] = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}},$$

where σ^2 is the variance. As we know the Gaussian function has an infinite support, so we are considering a finite version of the kernel here obtained by truncation, e.g. if we want a 3×3 sized Gaussian kernel, then $i, j \in \{-1, 0, 1\}$. Using the definition from the previous part, an image can be filtered using a Gaussian kernel as:

$$\mathbf{O} = \mathbf{I} * \mathbf{G}.$$

An interesting property of the Gaussian kernel is that 2D convolution of an image with the Gaussian kernel can be written as two 1D sequential convolutions, i.e.

$$\mathbf{O}[x,y] = \mathbf{I}[x,y] * \mathbf{G}[x,y] = \mathbf{I}[x,y] * G[y] * G^T[x],$$

where $G[y]$ is a vertical 1D Gaussian kernel and $G^T(x)$ is a horizontal 1D Gaussian kernel $\left(G[y] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}\right)$. **Prove this property, using the definition of 2D convolution and 1D convolution.**

Note that the order of convolution does not matter because its commutative, i.e. image could be convolved with the horizontal vector first and then with the vertical vector.

2D kernels that allow decomposition into 1D kernels are called separable, and kernel matrix in such cases can be written as $\mathbf{K} = \mathbf{v}\mathbf{h}^T$, where $\mathbf{v}$ is a column vector and $\mathbf{h}^T$ is a row vector.

Work out using pen and paper the convolution of the input image from Figure 1 with kernel corresponding to

$$G[y] = G[x] = \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix}.$$

(c) **Write a MATLAB function** `convolve(image,kx,ky)` **that accepts an input image and two 1D kernels as arguments, and generates a filtered image as an output**, assuming that 1D kernels form a separable 2D kernel. You cannot use any MATLAB functions that trivialize this task, e.g. `conv2/conv`. You may use any method to find the convolution output at the boundary pixels. The MATLAB functions `imread,imshow` or functions `iread,idisp` from Corke's library may be of help here.

(d) **Choose a grayscale image and apply a Gaussian filter of size** 5×5 **to it. Comment on the output using at max three sentences**. Recall that Gaussian kernel is separable, and so you can use an 5×1 Gaussian kernel using the expression $G[y] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}$ where $y \in \{-2, -1, 0, 1, 2\}$. Choose $\sigma = 0.83$

(e) **Sharpen the same image using the Gaussian kernel.**

(f) An image $I(x, y)$ is a 2D function $I : \mathbb{R}^2 \to \mathbf{R}$. The derivatives of this function can be computed and approximated as:

$$I_x(x,y) = \frac{\partial I}{\partial x}(x,y) \approx \frac{1}{2}\left[I(x+1,y) - I(x-1,y)\right]$$
$$I_y(x,y) = \frac{\partial I}{\partial y}(x,y) \approx \frac{1}{2}\left[I(x,y+1) - I(x,y-1)\right].$$

Find the kernels (i)$k_x \in \mathbb{R}^{1\times 3}$ **(ii)** $k_y \in \mathbb{R}^{3\times 1}$**, such that** $I_x = I * k_x$ **and** $I_y = I * k_y$**.**

(g) **Write a MATLAB function that accepts an image as an argument and computes** I_x**,** I_y**, and** $\sqrt{I_x^2 + I_y^2}$ **for this image. The last magnitude is returned as the output image.**

(h) **Use an original gray-scale image and Gaussian filtered image as inputs to the function in the previous part, and comment on the output for the original image and differences between the two outputs.**

Problem 2
8 points

In this problem, you're going to establish an image segmentation pipeline based exactly on the methods discussed in the class. Your goal in this problem is to have the computer find the answers to the following questions about the included image `objects.png`:

1. How many objects are in the image?
2. Provide a descriptor for the location of each object in the image.
3. Which object has the largest area?
4. Can question 1 be answered for `ducks.png`? You can threshold experimentally.
5. If you knew apriori that the image contained geometrical shapes, then can you think of some strategy for determining the shape? (You don't have to write code)

You are to provide your own MATLAB code for the entire pipeline. You cannot use MATLAB functions that trivialize these tasks. Examples of disallowed MATLAB functions include `bwconncomp,bwlabel,bwpropfilt,imbinarize,regionprops,visboundaries`. Your submission should include code for the following steps:

(a) A `threshold_image` MATLAB function that thresholds an image. It should accept an input image and a threshold value as an argument, and return a binary image of same size as output. You don't have to implement automatic thresholding, and can determine threshold experimentally. The MATLAB functions `rgb2gray`, `imtool`, and `imhist` may be of help here.

(b) A `label_image` MATLAB function that accepts the binary image from the previous step as argument and returns a labeled image. Each blob in the image should be assigned a different label, based on 4-connectivity. The MATLAB function `label2rgb` can help you visualize your labels.

(c) A `calculate_centroids` MATLAB function that accepts labeled image from the previous step as an argument and returns a matrix with locations of all centroids in (x, y) format.

(d) Necessary additions to the previous code to answer the questions outlined above.