HVCA - Exercise 3

Please submit the exercise in a single pdf file including all the code you wrote and your answers to all questions. You are welcome to use any programming language you want, but the helper files we supply will be in Matlab so you will need to convert them. Please be prepared to discuss your answers in the live sessions. For automatic checking, you will also need to provide the numerical output of your algorithm for different inputs in the Moodle.

Theoretical preamble:

The two dimensional convolution between two matrices X and Y is defined as follows:

$$Z[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} X[i,j]Y[m-i,n-j]$$

And we designate: $Z = X * Y$.

We use the following conventions:

- The central row of the matrix is defined as row 0. If the number of rows is even then the lower of the two central rows is row 0.
- The central column of the matrix is defined as column 0. If the number of columns is even then the leftmost of the two central columns is column 0.
- If a matrix X is not defined on the indices i, j then $X[i,j] = 0$.
- A matrix X is defined on those indices i, j such that there exists at least one term $X[k,l] \neq 0$, where $|k| \geq |i|$ and $|l| \geq |j|$, and where k, i have the same sign and l, j have the same sign.

1. Convolution is associative. i.e. $(X * Y) * Z = X * (Y * Z)$

2. Convolution is linear. i.e. $X * (aY + bZ) = aX * Y + bX * Z$ where a,b are scalars.

3. Given the derivative kernels K_x, K_y, we define the Laplacian of an image:

$$\nabla^2 I = (I * K_x) * K_x + (I * K_y) * K_y$$

If $K_x = \frac{1}{2}\,(1 - 1)$, $K_y = \frac{1}{2}\begin{pmatrix} -1 \\ 1 \end{pmatrix}$, then $\nabla^2 I$ is equivalent to $I * K_l$ where

$$K_l = \frac{1}{4}\begin{pmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{pmatrix}$$

4. If there exists a kernel K_α such that $K_l * K_\alpha = \bar{\delta}$. Where

$\bar{\delta}[i,j] = \begin{pmatrix} 1 & i,j = (0,0) \\ 0 & else \end{pmatrix}$. Then$\left(\nabla^2 I\right) * K_\alpha = I$.

5. **Theorem:** Given two images u, v, the image x that minimizes

$$J(x) = \|K_x * x - u\|^2 + \|K_y * x - v\|^2$$

satisfies

$$\nabla^2 x = u * K_x + v * K_y$$

6. **Corollary:** If $u = K_x * x$ and $v = K_y * x$ then solving:

$$\nabla^2 x = u * K_x + v * K_y$$

will recover x up to a constant.

Programming:

Guidelines:

- For the rest of the exercise assume that $K_x = \frac{1}{2}\,(1 - 1)$, $K_y = \frac{1}{2}\begin{pmatrix} -1 \\ 1 \end{pmatrix}$.
- To load an image onto Matlab use `im=double(imread('imagename'));`
- Unless noted otherwise, to view an image in matlab use `show(im,[0 255])` . To view a Laplacian or binary image use `show(im)`.

1. Write a function named `ImageDerivatives.m` that receives an image I and returns the derivatives I_x and I_y . Each derivative is the convolution of the image with the appropriate kernel. Use Matlab's `conv2` function but do not use the `'same'` option (use the default parameter which is `'full'`). From the resulting matrix discard the last column (I_x) or the last row (I_y). Also, set all the borders to zero (that is, set the first and last column and row to 0, using `end` should help).

2. Write a function named `Deriv2Laplace.m` that receives as input the derivatives of an image and returns the Laplacian of the image. Here you may use `conv2` with the `'same'` option.

3. look at the image `simul_cont_squares.tif` . Which square appears lighter to you? We explore the possibility that this phenomenon can be explained by the brain using a Laplacian of the image. Compute the Laplacian of the image, and show that there exists a threshold T where there are pixels bigger than T (in absolute value) only on the left side of the Laplacian image (there may be one or two "errant" pixels on the right side, you may disregard them). To show those pixels in a binary image, use `show(binary_im)`.

4. look at the image `cross.tif` . Do you see the cross formed by the diagonals of the squares as brighter? Compute the Laplacian of the image. Show that there exists a threshold T where there are pixels bigger than T (in absolute value) only on the cross in the Laplacian image.

5. look at the image `kofka_ring.tif` . Compute the Laplacian image. Can you explain the illusion using a threshold T as in the previous two cases?

6. Can the three previous illusions be explained just by the brain using the image Laplacian? Explain your answer. Provide an alternate hypothesis.

7. Write a function `do_retinex` that performs the Retinex algorithm. The function should receive an inputs an image I and a threshold T and return the reflectance image R and the illumination image L. The algorithm steps are:

   (a) Take the logarithm of the input image.

   (b) Calculate the derivatives of the log image. use `ImageDerivatives`.

   (c) Calculate the norm of the derivative at each point. (the square-root of the sum of the squares of the derivatives at each point).

   (d) Set all derivatives whose norm is less than T to zero.

   (e) Reconstruct the log reflectance from these new derivatives:

      i. Use the function `Deriv2Laplace` to compute the Laplacian.

      ii. Compute the inverse Laplacian kernel $K_\alpha$using the provided `invDel2` function. The argument to this function should be $size(I)$.

iii. Convolve the Laplacian with K_α (use the 'same' option) to obtain the log reflectance.

(f) Exponentiate the log reflectance to obtain a reflectance image. The illumination image is then calculated by dividing the input image by the reflectance.

As mentioned in class, the actual retinex algorithm is more complicated than this.

8. Consider the synthetic image constructed by the function `twoSquares(1)`. Use `show(im,[0 2])` to view the image. Show the output of your algorithm with $T = 0.07$. Does the algorithm return the same reflectance for the two squares ? To verify this, plot the diagonal elements of the reflectance $R[x, x]$ as a one dimensional function. Repeat the experiment with with the image constructed by the function `twoSquares(2)`. Show that the algorithm does not return the same reflectance in this case (again plot the diagonal elements). Which of the Retinex assumptions are violated in this case? Would changing the threshold help ? If not, why?

9. Load the Matlab file `checkershadow.mat`. It contains an image `im1` and two coordinates `x1,y1,x2,y2` . Use `show(im1,[0 1])` to view the image. The coordinates define two points inside the two checks marked A and B. Verify that indeed the two checks have exactly the same intensity by printing `[im1(y1,x1) im1(y2,x2)]`. Run your algorithm on this image with $T = 0.07$. What is the calculated reflectance of the two checks (print `[R(y1,x1) R(y2,x2)]`) ? Does this correlate with the illusion ? Discuss the various regions in the image where the algorithm fails in recovering the expected reflectance. Would changing the threshold help ? If so, how would you change it? If not, why?

10. Load the Matlab file `runner.mat`. It contains an image `im1` of a runner. Use `show(im1)` to view the image. Show the output of your algorithm for thresholds going from 0.05 to 0.15. Does the algorithm correctly remove the shadows? What additional cues might help the algorithm in this case? Discuss.

11. Load the Matlab file `couch.mat`. It contains an image `im1` of a couch. Use `show(im1)` to view the image. Show the output of your algorithm for thresholds going from 0.01 to 0.04. Does the algorithm correctly remove the shadows ? If there is a qualitative difference between the performance here and in the previous question explain it.