COMP-ENGN-6528-Computer Vision Assignment

Q1. The following is a separable filter. What does it mean to be a separable filter?(0.5 mark) Write down the separate components of the following filter. (1 marks)

4	4	6
4	4	6
6	6	9

Figure 1: Separable Filter

Q2. Calculate the value of the blue patch in Fig. 2 using bilateral filtering. Assume the Domain kernel is of size 5×5, the standard deviation $\sigma_d = 2$, provided as $\begin{pmatrix} 0.0232 & 0.0338 & 0.0383 & 0.0338 & 0.0232 \\ 0.0338 & 0.0492 & 0.0558 & 0.0492 & 0.0338 \\ 0.0383 & 0.0558 & 0.0632 & 0.0558 & 0.0383 \\ 0.0338 & 0.0492 & 0.0558 & 0.0492 & 0.0338 \\ 0.0232 & 0.0338 & 0.0383 & 0.0338 & 0.0232 \end{pmatrix}$ the Range kernel is of size 5×5 and the standard deviation $\sigma_r = 50$. Please 1) provide the range filter associated to the pixel high-lighted in the Fig. 2 (2 marks), and 2) show the filtered value for the high-lighted pixel (2 marks).

128	128	100	100	103	50
150	100	120	30	53	54
150	112	127	40	35	20
132	125	112	43	20	10
133	130	100	30	10	20
140	130	120	20	10	20

Figure 2: Image Patch

Q3. Contour Detection [10 marks+5 marks(extra)]

Acknowledgement: Lab material with code are adapted from the one by Professor Saurabh Gupta from UIUC, copyright by UIUC.

In this problem we will build a basic contour detector. We will work with some images from the BSDS dataset (see [1]), and benchmark the performance of our contour detector against human annotations. You can review the basic concepts from lecture on edge detection (Week 3). We will generate a per-pixel boundary score. We will start from a very simple edge detector that simply uses the gradient magnitude of each pixel as the boundary score. We will add non-maximum suppression, image smoothing, and optionally additional bells and whistles. We have provided some starter code, images from the BSDS dataset and the evaluation code. Note that we are using a faster approximate version of the evaluation code, so metrics here won't be directly comparable to ones reported in papers.

Preliminaries. Download the starter code, images and evaluation code from wattle (Assignment.zip)(see contour–data, contour_demo.py/contour_demo.m). The code has implemented a contour detector that uses the magnitude of the local image gradient as the boundary score. This gives us overall max F-score, average max F-score and AP (average precision) of 0.51, 0.56, 0.41 respectively. Reproduce these results by running contour_demo.py/contour_demo.m. Confirm your setup by matching these results. Note that the matlab version may be with 0.01difference from the python code due to the difference in inbuilt functions.

When you run contour_demo.py/contour_demo.m, it saves the output contours in the folder outputdemo, prints out the 3 metrics, and produces a precision–recall plots at contour–output/demo_pr.pdf. Overall max F–score is the most important metric, but we will look at all three.

- Warm-up. As you visualize the produced edges, you will notice artifacts at image boundaries. Modify how the convolution is being done to minimize these artifacts. (1 mark)
- Smoothing. Next, notice that we are using $[-1,\ 0,\ 1]$ filters for computing the gradients, and they are susceptible to noise. Use derivative of Gaussian filters to obtain more robust estimates of the gradient. Experiment with different sigma for this Gaussian filtering and pick the one that works the best. (3 marks)
- Non-maximum Suppression. The current code does not produce thin edges. Implement non- maximum suppression, where we look at the gradient magnitude at the two neighbours in the direction perpendicular to the edge. We suppress the output at the current pixel if the output at the current pixel is not more than at the neighbors. You will have to compute the orientation of the contour (using the X and Y gradients), and implement interpolation to lookup values at the neighbouring pixels. (6 marks) In the code, you may need to define your own edge detector with non-maximum suppression. Note that all the functions are called in 'detect_edges()'.
- Extra Credit. You should implement other modifications to get this contour detector to work even better. Here are some suggestions: compute edges at multiple different scales, use color information, propagate strength along a contiguous contour, etc. You are welcome to read and implement ideas from papers on this topic. (upto 5 marks)

For each of the modifications above, your report should include:

- key implementation details focusing on the non-trivial aspects, hyper-parameters that worked the best,
- contour quality performance metrics before and after the modification.
- impact of modification on run-time,
- visual examples that show the effects of the modification on two to three images.

————-This is the end of the assignment. ————