

EECS 504 Foundations of Computer Vision: HW2

Term: Fall 2018

Instructor: Jason J. Corso, EECS, University of Michigan

Due Date: 10/22 23:59 Eastern Time

Constraints: This assignment may be discussed with other students in the course but must be written independently. Programming assignments should be written in Python using the open-source library ETA. Over-the-shoulder Python coding is strictly prohibited. **Web/Google-searching for background material is permitted. However, everything you need to solve these problems is presented in the course notes and background materials, which have been provided already.**

Goals: Test mathematical basics and deepen the understanding of images as functions.

Data: You need to download `hw2.zip` to complete this assignment. All paths in the assignment assume the data is off the local directory.

Problem 1 (30): Local Image Features and Image Stitching

We motivated and derived an initial local image feature point detector based on an eigen decomposition of the structure tensor. We also demonstrated an application example of image stitching. In this question, you will explore these topics further. You can use `pipelines/image_stitching.json` for this question

- (a) (10) Implement the local structure tensor construction and eigen decomposition as described in class. This method is called Harris detector, but implement the simplified corner response measure based on the minimum eigenvalue of the structure tensor, as discussed in class. You should fill in your code in `_get_harris_corner()` of `modules/harris.py`. Read comments in code for further instructions on arguments and return values.

You should run the pipeline on `checkerboard.png`. This will execute your Harris function and will run a non-maximal suppression routine and extract corner locations from response image.

Include both the images:- (a) checkerboard response image (directly from Harris) and (b) actual corner detections, and the code verbatim in the pdf report.

You can run the the pipeline using the following command:

```
eta build -r requests/only_harris.request.json --run-now
```

- (b) (5) Run the pipeline using `concentric_circles.png` as your input. Include the corner response image in the write-up. Also please discuss the response image explaining why it looks like the way it does. Specifically describe why there are so many responses, when there are no "corners" in the image, and why, even though, there are many responses, no full ring has a high corner response over the entire ring.

- (c) (12) **Image Stitching.** This question will show you an end-to-end use of these computer vision tools by stitching together the two images below `red1.png` and `red2.png`. The code for doing this has three basic pieces: reduction (extraction of cor-



Figure 1: red1.png



Figure 2: red2.png

ners and representation with HOG features), matching (finding the best K correspondences across the images), and estimation (computation of the homography that aligns the two images).

To make this possible, the code for extracting HOG features and matching correspondences is provided to you. You have to fill in the functions `_get_homography()` (to compute the homography matrix) and `_overlap()` in file `modules/image_stitching.py`. Perform the stitching from the perspective of `red1.png`.

You can run the the pipeline using the following command:

```
eta build -r requests/image_stitching_request.json --run-now
```

- (d) (3) Repeat the process with different number of correspondences (4,10 and 20)
Explain (1) why 4 correspondences is worse than 10; (2) why 20 correspondences, which we may expect to be better, completely fails. Include the output images and your explanations in the report.

Problem 2 (15): In the last assignment, you have implemented kernels for x and y direction for finding the derivative and gradient. Think of a situation when you have to do the same in an arbitrary direction. Will it be feasible to construct specific kernels for all the possible directions?

The **steerable filters** are a class of filters in which a filter of arbitrary orientation is synthesized as a linear combination of a set of "basis filters."

- (a) (5) You are given an image `lines.png`. You have implemented the Sobel horizontal and vertical kernels in HW1. Use these kernels to detect the lines in the image that do not lie along 'x' or 'y' axis and attach the output image in your writeup. Your output should consist of **only** the lines that are not horizontal or vertical. You can use `pipelines/steerable_filter.json` and `modules/steerable_filter.py` to do this task

You can run the the pipeline using the following command:

```
eta build -r requests/steerable_filter_request.json --run-now
```

- (b) (7) Is the first-order derivative of 2-D Gaussian filter steerable? Why? Prove your answer. If yes, what are its basis functions?
(c) (3) Is the second-order derivative of 2-D Gaussian filter steerable? If yes, give the dimension of its basis. Can this be extended to n^{th} order derivative?

Problem 3 (20): Mapping Kernels, Cost and Separability

Kernel-based spatial range maps allow for operator generality but can be computationally expensive.

- (a) (5) For an image of size $n \times n$ and a kernel of size $2k + 1 \times 2k + 1$ what is the total number of operations in terms of multiplies and adds to map the kernel over the image (to convolve the image by the kernel) only apply it in *valid* locations where the kernel fits fully over the image lattice Λ .
(b) (5) For certain convenient forms of a 2D kernel, such as one that samples a Gaussian weighting function of a given variance σ^2 , this cost can be reduced by sequential operations of a 1D kernel. Show that a convolution by a 2D Gaussian kernel is equivalent to two sequential operations (one vertical and one horizontal) of the appropriate 1D Gaussian kernel. Specify the relationship between the 2D and 1D Gaussian kernel. (Hint: observe the linearity of convolution.)
(c) (5) For this separable kernel of size $2k + 1 \times 1$ and an image of size $n \times n$, how many operations (multiplies and adds) are needed to achieve the equivalent operation as if we just applied the 2D kernel (i.e., nonseparable).
(d) (5) Make an argument for or against whether or not we can leverage this separability when looking for step edges in images as zero crossings of the Laplacian operator output.

Submission Process: Submit a single pdf using **Latex** with your answers to these problems, including all plots and discussion. Submit the pdf to Gradescope.

For coding assignments, include your code verbatim in your writeup. Pack the original program files into one zip file and upload it to Canvas. **Code should be well-commented for grading**

Grading and Evaluation: The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given for both paper and python questions. For python questions, if the code does not run, then limited or no credit will be given.