CS 485/685 Project 2

Image Filtering

Due 3/22/2024 11:59pm

Logistics

Implementation

You must implement everything stated in this project description that is marked with an **implement** tag. You cannot import any packages unless specifically noted by the instructor. In this project, you may import the numpy, math, cv2, PIL, matplotlib, and skimage packages. Unless otherwise specifically noted, you must implement all project elements from scratch. You are welcome to ask about particular packages as they come up. The idea is to implement everything from scratch.

Deliverables

Each student should submit a single ZIP file, containing their project code (a single project2.py file) and your writeup (README.txt). You should create a folder called Project2 (exactly like this with the capital P) and put all your code and your writeup in this folder. Then zip this folder up. That way when I extract your folder, I can run my tests from outside the directory without having to change anything. Your zip file should be named lastname_firstname_project2.zip. Your code should run without errors in a Linux environment. If your code does not run for a particular problem, you will lose 50% on that problem. You should submit only one py file, named project2.py. If your file does not match the filename provided exactly, you will lose 50%. Do not worry about what Webcampus does to your file. Each time you submit your project, it will add a -1, -2, -3 onto it. When I download it and unzip it, there is no issue!

Grading

I have provided you with a test script (test_script.py) you can use to test out your functions. I will be using a similar test script, so if your code works with this script it should work with mine! I will run the test script from directly outside of the Project2 directory. Each student must work independently. You are to submit your own original work. You can work with whatever images you'd like, just use grayscale.

1 Load and Display Images (Copy from Project 1)

You will **implement** two functions:

load_img(file_name)

and

display_img(image)

You are welcome to use whatever packages you would like to do this (e.g. cv2, PIL, skimage, matplotlib). Try out the different packages to see what kind of differences they have! Choose your favorite and explain why you used this package in the README.txt.

The first function should return an image, and the second function should not return anything.

Assume for the entire project that you are working with grayscale images.

2 Moravec Detector (20 points)

You will **implement** a function to find interest points in an image:

moravec_detector(image)

Assume you're working with a 3x3 window. This function should return a list of keypoints ((x,y) coordinates) in the image.

Detail your implementation in your README.txt.

3 Plot Keypoints (10 points)

Implement a function to plot keypoints in an image.

`plot_keypoints(image, keypoints)`

function should return a new image, where where the keypoints are plotted on `image` in red.

4 LBP Features (20 points)

Implement a function to extract LBP features from a keypoint.

`extract_LBP(image, keypoint)`

Assume you are working with one keypoint. This function should return a single feature vector.
Detail your implementation in your README.txt.

5 HOG Features (20 points)

Implement a function to extract HOG features from a keypoint.

`extract_HOG(image, keypoint)`

Assume you are working with one keypoint. This function should return a single feature vector.
Detail your implementation in your README.txt.

6 Feature Matching (20 points)

Implement a function to match features across two images.

`feature_matching(image1, image2, detector, extractor)`

`detector` should either be "Moravec" or "Harris". Anything other than these two options should result in an error. If you do not implement the extra credit (see below) then anything other than "Moravec" should result in an error.
`extractor` should either be "LBP" or "HOG". Any other value should result in an error.
Detail your implementation in your README.txt. This function should return two lists of matching keypoints. For example, list 1 would have its first element (x1,y1) and list 2 would have its first element (x2,y2) and this would mean that (x1,y1) in image 1 matches with (x2,y2) in image 2. This function should call your `moravec_detector` or `harris_detector` and your `extract_LBP` or `extract_HOG` functions.

7 Plot Matches (10 points)

Implement a function to plot corresponding matches between two images.

`plot_matches(image1, image2, matches)`

`matches` should be the result of the above `feature_matching` function, that is, two lists. You can pass this however you want, as you're the one who will use it in the function. Just make sure it is only one parameter so that my test script works. This function should return a new image, where image1 is on the left and image 2 is on the right, and there are red points marking keypoints in image1 and image2 and red lines connecting matching keypoints.

EXTRA CREDIT: Harris Detector (+20points)

You will **implement** a function to find interest points in an image:

`harris_detector(image)`

Make any assumptions you'd like about the gaussian window inside the function. This function should return a list of keypoints ((x,y) coordinates) in the image.
Detail your implementation in your README.txt.
THIS FUNCTION CAN ONLY BE IMPLEMENTED IF EVERYTHING ELSE IS WORKING.