INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 5 - Feature Descriptors and Geometric Transformations

Submission Date: 11.1.2023

Introduction

In this assignment we will learn about feature descriptors and geometric transformations on the image domain.
Before you start, please read the submission instructions at the end of the assignment and follow them during your work. For any question regarding this assignment please refer to the course forum on the moodle website, for personal questions **only** please email inbalri@post.bgu.ac.il

Note: In this assignment you may use any MATLAB functions but a special emphasis will be made to your written explanations.

1 Mona Surfaliza

In this assignment we will use SURF features in order to align the image and detect objects in it. You may use all available MATLAB functions but a special emphasis will be made to your written explanations.

1.1 Inroduction to SURF

1. Read about the MATLAB function *detectSURFFeatures(I,Name,Value)*. Briefly explain what are the optional parameters of Name-Value pair arguments ? What are the default values assigned to these parameters?

2. Read about the MATLAB function *extractFeatures(I,points)*. Briefly explain what is the purpose of the function? Where do the '*points*' values come from?

3. Read the images '*mona_org.jpg*' and convert it to grayscale normalized image.

4. Find all the features in the image using *detectSURFFeatures(I)* function.Time the feature extraction using *tic; toc;* commands as in assignment 4 (Hough Transform). What is the runtime of this option? Look at the features extracted and present them along with the image. How many features did you find?

5. Find features in the image using *detectSURFFeatures(I,Name,Value)* function and the pair *'ROI', [59, 5, 128, 120]*. Present the selected 'ROI' in the image- what do you see? What is the runtime of this option in compare to default parmeters run (section 1.1.4)? Look at the features extracted and present them along with the image. How many features did you find? What is the tradeoff?

6. Find features in the image using *detectSURFFeatures(I,Name,Value)* function and 2 differents *'NumOctaves'*. Compare the runtime of the two options with the default parmeters run. Look at the features extracted and present them along with the image. How many features did you find? What is the tradeoff?

7. Find features in the image using *detectSURFFeatures(I,Name,Value)* function and 2 differents *'NumScaleLevels'*. Compare the runtime of the two options with the default parmeters run. Look at the features extracted and present them along with the image. How many features did you find? What is the tradeoff?

1.2 Make Mona Straight Again

1. Read the images '*straight_mona.PNG*' and *'crooked_mona.jpg*'.

2. Extract the SURF feature points of each of the images and display the ten strongest features for each of the images correspondingly.

3. Use the extracted feature points to straighten the *'crooked_mona.jpg*' image so that Mona's face will be straight. Display the images. Explain your algorithm using a block diagram and elaborate on each of the steps.

1.3 Fake or Real

1. Read the image '*mona_org.jpg*' and use the SURF features to automatically detect images in which the real Mona Liza face exists.

   (a) The 12 test images are in the zip file named 'Monas'.

   (b) The last letter in the image name 'Y' or 'N' suggests if we consider that Mona's face is in the image or not.

   (c) The code should print out the names of the images you detected Mona's face inside.

   (d) The algorithm should be totally automatic and run on all the images in the same manner.

(e) Try taking under consideration the trade-off between finding a lot of correct images and falsely detecting wrong images.

(f) Try to do your (and your algorithms) best, if you can't detect an image, just try your best and explain why it was impossible.

2. Display the images you found.

3. Explain your algorithm using a block diagram and elaborate on each of the steps.

4. For each image show the matching features. See example in Figure 1.

Figure 1: Example for matching features image

5. Analyze the results. Why did you succeed in the image you did? Why did you get it wrong in the images you did? Where did the SURF features have good performance and where did they not?

2 QR Code Reader

In this assignment we will learn to read a simplified version of the QR Code. This is the main part of the assignment.

1. Insert your 9 digit ID to the attached function ID2QR(id) where ID is a string. Print the QR code on paper. The QR code is a matrix of 6x6 binary values.

2. Take three photos of the QR code from three different angles: easy, intermediate, hard.

The “hard” photo should be one that is on the limit where you can no longer properly interpret the QR code. (see example in figure 2).

3. Locate the points on the QR code in each image manually. You may use MATLAB’s *ginput(4)*.

4. Transform the image such that the QR code is straightened. For each image use all the transformations learned in class: Rigid (rotation, translation and scale transformation), Affine (shearing added), Perspective.

5. Explain the result of each transformation and why does it work/fail for the given image.

6. Extract the binary values from the straightened QR. The matrix is row major meaning that the correct order is (1,1), (2,1), ..., (6,1), (1,2), (2,2), ..., (6,6).

7. Convert every 4 bits to an integer: e.g. 0110 –> 6, 1001–>9.

8. Make sure you read the correct ID.

9. The zip file should contain all three images and automatically transform the image. You should save/load the corners coordinates in a *.mat file and include it in the zip.

Figure 2: Example of possible angles: easy, intermediate and hard

3 Automatic Corner Detector

Notes:

1. In this task you may use any MATLAB function. Look up MATLAB's morphological operations on images, it might assist you.
   If you use builtin functions, briefly explain their algorithm and why you use them.

2. if you're having a trouble finding the corners, you may add assumptions to ease the task. It is preferable to come up with a working algorithm with assumptions than to show no results.

In this assignment you will use previously learned subjects to design an automatic corner detection algorithm for the QR code reader. This detector will be used to replace the manual input of the previous assignment.

1. Think of an algorithm to detect the QR code corners. You may search online to find an algorithm or ideas.

2. Implement the algorithm. You should be able to find the corners of an easy image. Finding the corners of more difficult images is encouraged.

3. Thoroughly explain the different algorithm steps and why they are done. If any assumptions are made - describe them (e.g. assuming that the QR code is printed on a paper with white margins).
   If a step can be visualized using an image, show it!

4. Use the corners you found as an input to the first assignment. Show and analyze the results on the images you took.

4 Bonus

Take two images and use the extracted features methods that you learned in class and in this assignment to create a new trashed image. **Be creative! One of the images will be chosen by the course staff and it's authors will receive one bonus point to the final grade**. The staff will judge by the visual result, originality and the code.
In order to implement your algorithm, you can use any code you find online.

Submission Instructions

The following instructions are mandatory and will be checked and graded by the course staff. Failing to follow these instructions **will** reduce points from you grade.

The assignment **must be done in pairs.**
Only one of the team members should submit the file !

The assignment is to be done in MATLAB and submitted to the course moodle page in the form of a *.zip (**not RAR**) containing ***Ex5.mlx*** or ***Ex5.m*** -the main MATLAB file, other *.m files and images along with a report in the form of a PDF file (NOT .doc). **Both the PDF and ZIP file names should be the initials and ID of both of the team members ex. 'SC-1234567_RS-7654321.pdf' and 'SC-1234567_RS-7654321.zip', respectively.**

Academic integrity: the originality of the submitted exercises **will be checked.**

Document Instructions

- The report should be written in Hebrew or English.
- Each section should have the relevant title as is in this document.
- Every image should be accompanied with the relevant explanation.
- The displayed images should be large enough for us to see them.
- The document should be organized and readable.

Code Instructions

- Use MATLAB version 2014b or later. If you don't have one on your computer, you can work from the computer laboratories in building 33 using VPN.
- A **main** function should call all the section functions in the correct order and should be named ***Ex5.mlx*** or ***Ex5.m***.
- The first line of ***Ex5.m*** / ***Ex5.m*** should print the full names and IDs of all team members. Use MATLAB's *disp()* function.
- Write modular functions for the subsections and reuse those functions throughout your code whenever possible.
- Every **.m* file should start with a comment containing the full names and IDs of all team members.
- Use meaningful names for all functions and variables.
- Try to avoid overriding variables.
- Write comments for every line of code that is not completely self explanatory.

- For every image displayed give a meaningful title using MATLAB's title() function.
- Use subplots whenever possible.
- All paths to files should be relative paths. If you are using subfolders use MATLAB's *fullfile()* function to construct the path to the file. Do not hard code '/' or '\' in the paths.
- The code should run completely without errors. A project with errors **will not be checked!**