INTRODUCTION TO DIGITAL IMAGE PROCESSING

361.1.4751

EXERCISE 5 - Geometric Transformations

Submission Date: 07.01.19

Introduction

In this assignment we will learn geometric transformations on the image domain. **Before you start,** please read the submission instructions at the end of the exercise and follow them during your work. For any question regarding this assignment please refer to the course forum on the moodle web site, for personal questions **only** please email hgazit@post.bgu.ac.il, davidit@post.bgu.ac.il, royshau@post.bgu.ac.il or arbellea@post.bgu.ac.il.

Note: In this exercise you may use the any MATLAB functions.

1 QR Code Reader

In this exercise 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 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 1).

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.

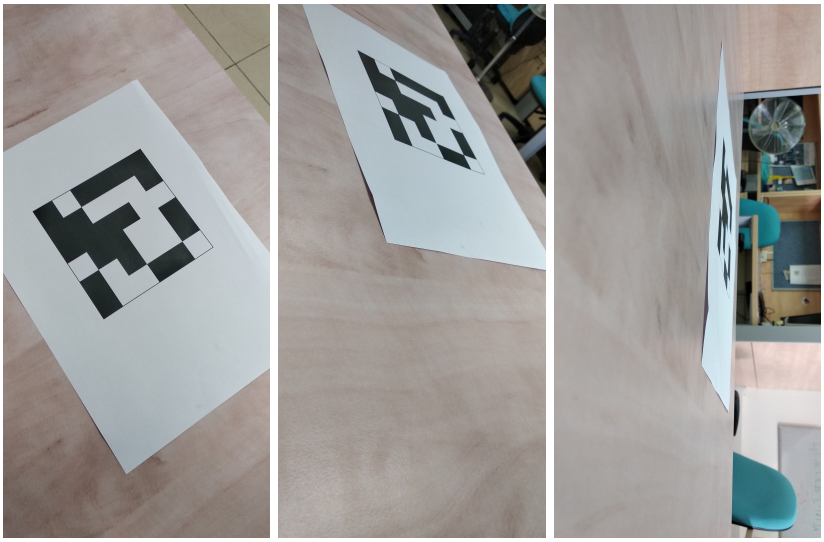

Figure 1: Example of possible angles: easy, intermediate and hard

2 Automatic Corner Detector

In this exercise 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 exercise.
Note: 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 you don't have to explain their algorithm. You do need to explain why you use them.

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 exercise. Show and analyze the results on the images you took.

Note: 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.

Submission Instructions

The assignment is to be done in MATLAB and submitted to the course moodle page in the form of a *.zip (**not RAR**) containing *.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. `'HG-1234567_ AA-7654321.pdf/zip'`.
Submit only one assignment per pair!

Document 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.

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

Code 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.

- 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.
- A **main** function should call all the section functions in the correct order and should be named *Ex5.m*.
- The first line of *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.
- The code should be clearly written with correct indentations (you could use automatic indentation Ctrl+A followed by Ctrl+I).
- 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!**