EN.601.461/661

Computer Vision

Assignment #2

Due date: September 24 11:59PM

5 marks (10% per day late submission)

Instructions: Submit your Python files on Gradescope. Please follow the instructions for each file. Templates for the file and image can be found on Piazza (Ressources). A template is provided for each file with the function's signature. Do ***not*** change the signature of the functions and ensure that the arguments and returned types are of the same as those specified below. We will test your function with the expected names, input and output. Grades will be deducted for non-compliance. You will need to import OpenCV and numpy to manipulate images and arrays.

In this assignment, you will implement Question #2 from assignment #1. Specifically, you will

1) Pick (with the mouse) 4 points in an image
2) Calculate the distortion
3) Display the distorted square
4) Use your calculations to rectify the square and display the result
5) Use OpenCV to calculate the homography and display the result

PointPicker.py (1pt)

This function has the following purpose:

1) Read the square image from a file
2) Display the image in a window
3) Enable the selection of points in the image with left mouse clicks
4) Close the window (can be done manually with the keyboard)
5) Return the normalized homogeneous coordinate of the selected coordinates (last row of 1s)

The function PointPicker has no input but ***must*** return a *numpy.ndarray* of size *3xN* (an array of homogeneous coordinates) where N is the number of selected points (we will use N=4). You can read, display and use mouse event with OpenCV. Documentation on OpenCV GUI can be found here.

Be very careful with coordinate frames. OpenCV, and most image software, use (row, column) coordinates instead of (x,y) and do not use the bottom left image as the origin.

You must divide the selected coordinates by 300 to transform pixels to units square (300 is may not exact but it is good enough). Otherwise your distortion will be wacky.

ProjectiveDistortion.py (0.5pt)

The function ProjectiveDistortion does the following:

1) It receives homogeneous coordinate of the points $\boldsymbol{p}_i$ as argument. These should be of the same type as the points returned by PointPicker.
2) It creates the same 3x3 homography H as in assignment #1 (you can hardcode the values)

3) It maps the points $\boldsymbol{p}_i' = H\boldsymbol{p}_i$
4) Returns the 3xN ***normalized*** coordinates of $\boldsymbol{p}_i'$ (last row of 1s) (type numpy.ndarray)

DrawPolygon.py (1pt)

The function DrawPolygon does the following:

1) Receives 3xN normalized coordinates (i.e. from PointPicker or ProjectiveDistortion)
2) Create a white image large enough to draw all the coordinates
3) Draw black lines of the polygon defined by the points
4) Display the image in a window
5) Close the image manually (i.e. with the keyboard)

You can refer to the OpenCV documentation listed in PointPicker on how to draw lines in an image.

ProjectiveRectification.py (1pt)

The function ProjectiveRectification does the following:

1) Receives 3xN distorted points (from ProjectiveDistortion)
2) Calculate the lines, intersections, points at infinity and line at infinity (using cross products)
3) Calculate the rectified coordinates
4) Display the rectified coordinate (using DrawPolygon)
5) Return the homography (3x3 numpy.ndarray)

OpenCVRectification.py (1pt)

The function OpenCVRectification does the following:

1) Receives 3xN undistorted points (form PointPicker)
2) Receives 3xN distorted points (from Projective Distortion)
3) Use OpenCV to calculate the homography (not just the projective distortion) from the points correspondence
4) Return the homography (3x3 numpy.ndarray)

Documentation of OpenCV 3D reconstruction module can be found here. This is C++ documentation. I couldn't find the equivalent documentation for Python although small tutorials can be found here (but the functions in these tutorials will not help with your implementation).

main.py (0.5pt)

This is the main file. It mainly calls the aforementioned functions:

1) Pick the points in the images
2) Distort the points
3) Display the distorted shape
4) Calculate and display the rectified shape
5) Call OpenCVRectification to estimate the homography

Reminder: By submitting your assignment you acknowledge that you have read the JHU Academic Misconduct Policy and that you are the author of 100% of the code submitted.

Furthermore, distributing or sharing your assignments (whereas online, with classmates or students that will take this course in future semesters) is not authorized and will be sanctioned with a course grade “F”.