EN.601.461/661

Computer Vision

Assignment #4

Due date: October 8th 11:59PM

5 marks (10% per day late submission)

Instructions: Submit your Python files on Gradescope. Please follow the instructions for each file. Test images can be found on Piazza (Ressources). Templates files are not provided but the name of the files, functions, inputs and outputs are clearly defined. We will test your function with the expected names, input and output. Grades will be deducted for non-compliance. Most of the functions should be short (around 5 lines of code each) and use OpenCV. You will need to use OpenCV to for the followings: Canny, Hough, SIFT (detection and matching), find homography and warping. Refrain from displaying images in your functions (except for the two functions who are supposed to display them). Finally, there is no "main" function, you should be able to call your functions yourself (and display intermediate results if needed).

1) Use Canny edge detection
2) Use Hough transform and display results
3) Find vanishing point
4) Detect and match SIFT features
5) Calculate a homography and rectify matched image

Canny.py (0.5pt)

def canny():
return: An image (numpy ndarray) with Canny's edges

This function has the following purpose:

1) Read the image bt.000.png
2) Use Canny edge detection and return the resulting image.

This function is straightforward with OpenCV. The biggest challenge will be to tune the thresholds. Don't display an image within this function.

Hough.py (1pt)

def hough(img):
return: An array of N lines (numpy ndrarray)

This function only calculates and returns the lines from edges detected from Canny's detection. OpenCV has a weird way of packing these lines (it returns an Nx1x2 array) and you should return this array as is (don't reshape the array). As for Canny, the biggest challenge will be to tune the parameters to get "good" lines. Don't display an image within this function.

DrawLines.py (1pt)

def drawlines(img, lines):
return: Nothing

This function draws the lines calculated by Hough on the original image. Tune the values in Canny.py and Hough.py to get something “decent” as illustrated below (10-15 lines is fine). The biggest challenge is to figure out how to map the lines to the image.

SIFT.py (0.5pt)

def sift(filename):
return: keypoints, descriptors

This function receives a file name (these will be box.pgm and scene.pgm). The function reads the image, extracts and returns SIFT keypoints and descriptors. This is straightforward with OpenCV. Do not display images in this function.

Matching.py (0.5pt)

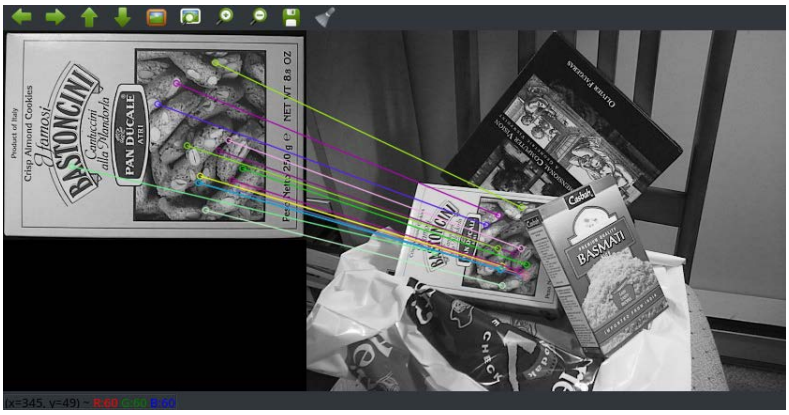
def matching(descriptor1, descriptor2, N):
return: A list of the N strongest matches

This function matches two sets of descriptors. OpenCV has several ways of doing this (use whichever you want). Only return the N strongest matches as a list of “DMatch” (DMatch is the structure used by OpenCV for storing matching data). Do not display images in this function.

DrawMatches.py (0.5pt)

def drawmatches(img1, img2, keypoints1, keypoints2, matches):
return: Nothing

This function draws keypoint matches. OpenCV has a function for this purpose. Tune N in Matching to obtain more than 10 strong matches. You should get something similar to this (no outliers!). Use the box.pgm and scene.pgm images.



FindHomography.py (0.5pt)

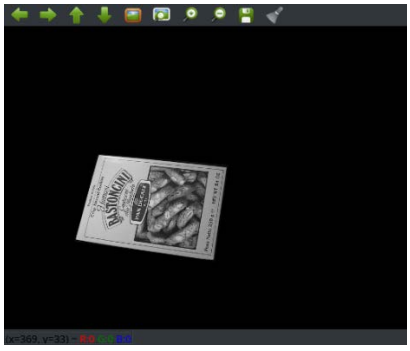
def findhomography(keypoints1, keypoints2, matches):
return: Homography

From the SIFT keypoints and the matches (no outliers), calculate and return the homography (use the same function as assignment #2). The biggest challenge of this function will be to pack the points from the matches.

WarpImage.py (0.5pt)

def warpimage(img, H):
return: Warped image (numpy ndarray)

The homography calculated above maps point between the box (a plane) and the scene. Here, you need to apply the homography to distort (warp) the image of the box into the scene (OpenCV has a function for doing this). You can hardcode the resulting image size to be the same size as the scene 512x384. For example, you should obtain something like this. Do not display images in this function.



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”.