

Image processing course – homework #2

imageprocessinghaifau@gmail.com

Lost Wormhole

In this homework you will be writing python functions which perform Geometric operations on an image (left) to create a **lost wormhole** poster (right):



To do this, we'll need three functions only:

(Use helping functions: `np.linalg.pinv`, `np.linalg.inv`, `np.matmul`, `round`)

1) **find_transform(pointset1, pointset2)**

This function calculates the global transformation T that transform pointset1 to pointset2.

Input: pointset1, pointset2 – arrays of size $N \times 2$

N is the number of points, 2 is the x and y .

Output: T – transformation matrix of size 3×3 .

Method: T fulfills the equation: $Pointset2 = T * pointset1$

To find T , you'll need to build the matrices X, X' in slides 58-62 in lecture5.

2) transform_image(image, T)

This function generates an image that is the result of applying T on image.

Input: image – 2D matrix with gray levels [0..255]
T – transformation matrix of size 3x3.

Output: new_image - 2D matrix with gray levels [0..255]

Method:

1. Define a new_image full of zeros the size of the given image.
2. Iterate over coordinates $[x', y']$ of the pixels of the new image
3. Answer the question: which pixel $[x, y]$ in image is transformed to $[x', y']$ in new image
4. Use nearest neighbor interpolation: just use round on the result of 3.
5. Insert gray level inside $[x', y']$

3) create_wormhole(image, T, iter)

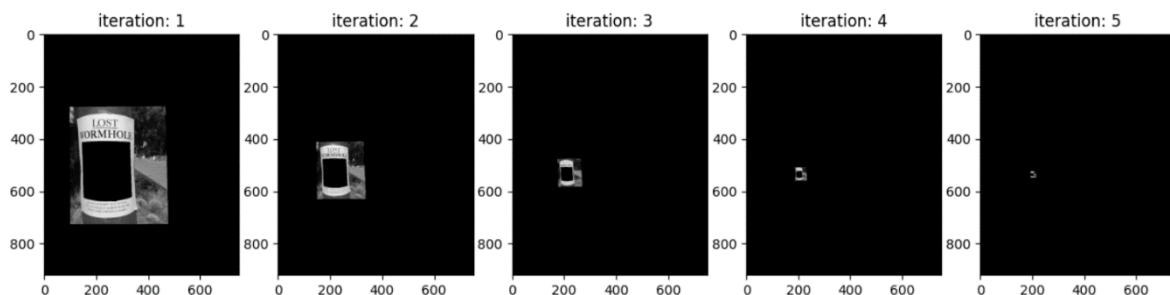
This function builds the final image.

Input: image – 2D matrix with gray levels [0..255]
T – transformation matrix of size 3x3.
Iter – an integer number

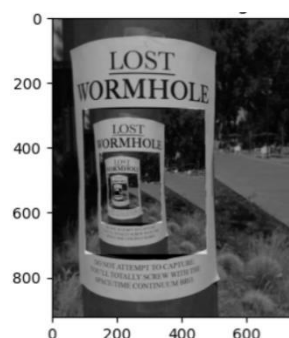
Output: new_image - 2D matrix with gray levels [0..255]

Method: this function iterates to create the image inside image (wormhole) multiple times (number of times is indicated with the variable iter).

Example: iter=5. The function creates 5 images.



Summing the original image with all 5 images will generate the wormhole:



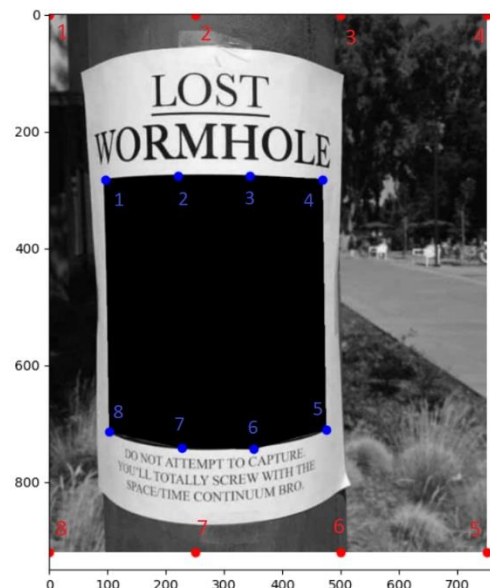
You only need to return the final image (the full wormhole).

Note: this function is not as hard as it looks. It is very easy if you implemented the previous functions correctly. Iteration1 is just applying T to the original image. How do you get iterations 2-5?

Hint: if T was scaling by 0.5. then iteration2 is scaling by 0.25, iteration3 is scaling by 0.125 and so on. (in this homework T is more than just scaling).

You are provided with two files **hw2_123456789.py**, in it you'll find blank functions and two arrays `src_points` (the red points below) and `dst_points` (the blue ones). The points are in matching order.

Important note: the points (1-8) are given in $[x,y]$, where x is the horizontal axis and y is the vertical. When finding the transform use them.



Be careful in function `transform_image` to the x,y .

As I said in class, the x,y are flipped when it comes to image indices.

Example: `image[x,y]` - x is the rows of the matrix (vertical axis) and y is the columns (horizontal axis).

Submission

Please submit one .py file with the functions implemented

Name the file **hw2_123456789.py** (or in case of pair: **hw2_123456789_987654321.py**)

(Replace 123456789, 987654321 with your ids)

Good luck!