Uniwersytet Wrocławski | Instytute Informatyki

LAB 3: DUE 10 NOVEMBER 2016

This Lab is an adaptation of the Assignment 6 for Fredo Durand's Computational Photography course at MIT. You can find a longer description of his assignment in FredoDurand.pdf, which might help you. Note that his notation changes the coordinate system to y,x,w instead of x,y,w. You are free to use his notation if you want, just be consistent!

You don't need to use all the data sets. Do at least, one with 2 images.

Task 1: Apply Homography (20 pts)

Write a function to apply a homography to an image into another. You will iterate over the output image, and take image2 when possible, and image1 when the coordinates are outside image2.
Test your function on the provided green.png and poster.png images using the homography: H=array([[0.8025, 0.0116, -78.2148],[-0.0058, 0.8346, -141.3292], [-0.0006, -0.0002, 1.]])

Remember you are using homogeneous coordinates.

image 1

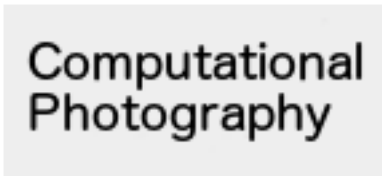


image2

Task 2: Bilinear Interpolation (20)

When sampling, the coordinates will be "between pixels". Use bilinear interpolation to get a better image. https://en.wikipedia.org/wiki/Bilinear_interpolation. Compare with nearest neighbour sampling.

Task 3: Compute Homography from Points (40 pts)

Taking points correspondences from two images, compute the homography. You can do this with 4 points, fixing i = 0. (See Lecture notes), and it's worth trying to do so, but the assignment is to use the Least Squares method, using more points and computing the solution with SVD. You have extra notes and the FredoDurand.pdf documents to help you. Check tomographies.pdf and FredoDurand.pdf for more information.

Test your solution with with the previous example:
pointListPoster=[array([0, 0, 1]), array([w, 0, 1]), array([w, h, 1]), array([0, h, 1])]

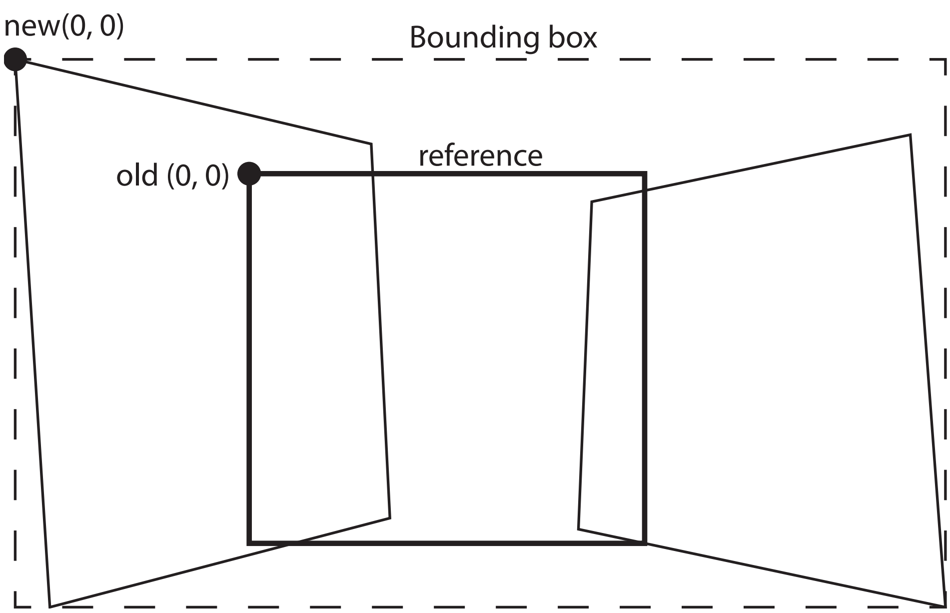
pointListTrain=[array([95, 170, 1]), array([238,171, 1]), array([235, 233, 1]), array([94, 239, 1])]

Try as well with pano/stata-1.png
im1=imread('pano/stata-1.png')
im2=imread('pano/stata-2.png')
pointList1=[array([218, 209, 1]), array([300,425, 1]), array([337, 209, 1]), array([336, 396, 1])]
pointList2=[array([4, 232, 1]), array([62, 465, 1]), array([125, 247, 1]), array([102, 433, 1])]

To ease finding the correspondence, you can use the simple UI in Pano-UI/click.html . You can change the image in the code. **You have to click in the same order in both images. Coordinates X and Y are changed!!**

Task 4: Bounding Box (20)

In the previous examples, you used the boundaries of the first image. For panorama stitching, we want to create a bigger image, as shown in the lecture examples. For this, we



need to estimate the size of the output. You will need to compute the inverse of your Homography, and compute the coordinates for the corners. Then use an offset to translate between “outside of the image” coordinates and the new bigger image coordinates.

Extra: Blending

You can try and compare smooth and two bands blending. Compute the weights as the distance to the boundary in the image, the center of the image should have value 1, and 0 at the boundaries. Use this weights to mix both images, keeping track of the weights, so the sum of both images has a value of one.

For the two band blending, compute a gaussian blurred image and divide the original by this, getting an image with the high frequencies only. Blend the blurred version with the same process as before, and blend the high frequency by simply taking the pixel with the biggest weight. Then, multiply both together to get the final image.