Intro to Image Understanding (CSC420)

Assignment 3

Posted Oct. 25, 2016; Submission Deadline: Nov. 4th 11.59pm 2016

Instructions for submission: Please write a document and submit a **PDF** with your solutions (include pictures where needed). Include your code inside the document, and submit through **MarkUS**.

For full marks you must show your work, not just your final answer.

Max points: 13, max extra credit points: 3

1. **[2 points]** A robber left his/her shoe behind. Police took a picture of it, see **shoe.jpg**. Estimate the width and length (in centimeters) of the shoe from the picture as accurately as possible! Show your work.

2. For this question you do not need to write code, you do need to write the equations and show your work (i.e. how you derived them). If you need to calculate the location of any points or lines in the image plane, state those as givens; but use the minimal set. You can assume that the ground plane is orthogonal to the image plane. *Hint: Draw the scenes on paper from the side.*

   (a) **[2 points]** Examine image **tracks.jpg**. Assume you are given K, R, t. Are you able to estimate the distance between adjacent railway ties in world co-ordinates? If so, write the necessary equations as a function of the pixel locations, camera intrinsic and/or extrinsic matrices.

   (b) **[2 points]** Examine image **man.jpg**. The camera centre is 95 centimetres off the ground. You can assume the ground is planar. Are you able to estimate the height of the man in centimetres without K? If so, derive and write the necessary equations as a function of the pixel locations in the image, and estimate the height.

3. You've joined a CSI unit, our suspect **mugShot.jpg** has been implicated in a grisly crime. Raiding his apartment we recovered a photograph of him sitting with his accomplice, but he knew we were on to him and shredded the photograph! We need you to implement RANSAC and re-assemble it. You may also use a downloaded implementation of SIFT and any code you wrote for Assignment 2. Python users check opencv-contrib.

   (a) **[2 points]** Create a controlled test case between two affine transformed images, and develop your RANSAC algorithm to calculate the affine transformation via least-squares. Visualize the best transformation for the best matching image just

like you did for Assignment 2, exercise 2(d). You can use any tutorial code to help create your test-case image, or draw one, or take some pictures.

(b) **[2 points] Shredded** contains the shredded picture pieces. Using the mugShot, try reassembling the image in random permutations and keep the one that best fits your model. Rank your models by the mean residual SSD. You could alternatively try a greedy or dynamic programming approach. Show the re-constructed image. Display your final, best reassembled image.*Hint: For speed use down-sampling.*

4. For this question you will build your own panorama stitching program. You can use your RANSAC code from above, in conjunction with your matching code from assignment 2 to establish correspondence. However, you now need to estimate a homography via least squares instead of an affine transformation (the math is in Lecture 9). You may also use SIFT and any code you wrote for Assignment 2. If you were unable to get RANSAC working, you can manually set your valid correspondences (or just use the top k), but state it clearly. You can look at the example here to see the workflow, but you must write your own code to compute the homographies and blend the panorama. Do not use **matchfeatures.m, estimateGeometricTransform.m, step.m**. Note, matlab users imwarp uses $x\mathbf{A}$ for its transform, and we've written things for $\mathbf{A}x$

   (a) **[1 point]** Compute the homography between pairs of adjacent images via your matches. Write your own solution via eigendecomposition; don't use **maketform.m** or equivalents. Visualize your result by transforming your keypoints from one image onto the next, and displaying them as a different colour. Use the image set in **hotel**.

   (b) **[1 point]** Perform the projective transformation to place the images points into a common co-ordinate system (at the central image). Remember, this must be performed sequentially and you'll need to accumulate your transforms from/to the centre image. Instead of the images, transform the image co-ordinates (via meshgrid, or 4-point rectangles) and display the resulting plot. Your result should look something like **panoMap.png** but with axes to show your co-ordinate system.

   (c) **[1 point]** Create your panorama. First create a large enough area to display it; look at your co-ordinate system from (b). You can use imwarp to transform your images. You do not need to implement any fancy blending, just sum or overlap your results.

5. **Extra: [3 points] total** (this is an optional exercise) Augment your panorama stitching program to include image blending with centre re-weighting and Laplacian pyramids. See Chapters 3.5.3 and 9.3 in the course textbook.

   (a) **[1 point]** Use centre re-weighting to simply average the image pairs.

   (b) **[1 point]** Use centre re-weighting to weight the Laplacian pyramids.

   (c) **[1 point]** Create your own fun examples showing the better performance in comparison to 4d.

1 Sources

1. man.jpg Chris Ford, https://www.flickr.com/photos/chrisschoenbohm/4817576839/
2. tracks.jpg Ryan Voetsch, https://www.flickr.com/photos/voetshy/