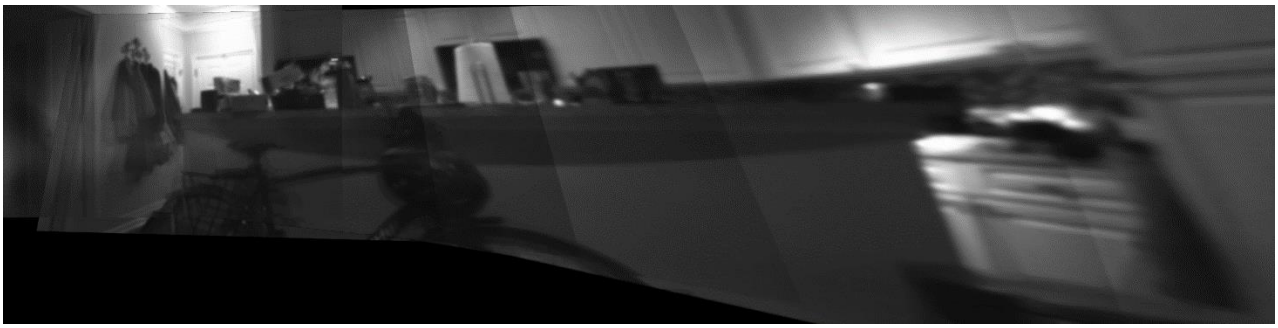
LAB 3: MOSAIC AND VISUAL ODOMETRY

Due: Thursday, March 10th

3pm class demo, 6pm code/video submission

PatrolCorp liked your previous planning algorithm for their robots, so they have decided to hire your team for a follow-up project on surveillance. Recently, PatrolCorp has been working on deploying security cameras that rotate to survey the scene and produce mosaic images of their surroundings. Unfortunately, their lead developer has recently left to join a rival company, Centillion, and you have been hired to pick up the pieces and complete their code.

For this project, you will generate a mosaic from a series of camera measurements as the robot rotates in place. This process is very similar to the image stitching feature available on cameras and phones. PatrolCorp still will not give you access to their top secret facilities for testing, but they generously provided a dataset with an example mosaic, as well as a calibration matrix for their cameras. At the completion of your project, you will have to demonstrate your system in a live demo in a real visual scene.



Download mosaic.py starter code from Canvas. As part of this lab, you will need to:

1. Implement *propose_pairs()*: find quality feature correspondences between consecutive image frames using the keypoints and their corresponding descriptors generated in the starter code.

2. Implement *homog_ransac()*: RANSAC to fit homography using good feature point matches between frames.

3. Implement *homog_dlt()*: called inside each RANSAC iteration to fit model using point correspondences.

Given the above, you should now be able to create mosaic images. In itself, the mosaic provides a useful visualization of the environment, but the homography matrix can also be used to derive the angle of rotation of the camera (robot) between each image. This information can be useful for odometry or calibration.

4. Write a short program to have your robot turn 5°, 10°, 15°, 20°, 25°, 30°, 35°, 40° and 45° in succession, taking and saving a picture between each rotation.

5. Implement *rot_from_homog()*: extract the rotation between frames from the homography, H, given the camera calibration matrix K. You may use the following parameters for the intrinsic matrix: $f_x = 1211.2959, f_y = 1206.00512, s = 0.0, u = 657.15924, v = 403.17667$. Alternately you can estimate the intrinsic parameters for your specific robot using the camera calibration toolbox at http://www.vision.caltech.edu/bouguetj/calib_doc/.

6. Implement *extract_y_angle()*: extract yaw angle from rotation matrix R

7. Run your turning program 5 times in different scenes. Produce a single plot that shows the difference between the wheel and visual odometry of the robot for all five runs. The x-axis of the plot should be the number of degrees the robot was instructed to turn, the y-axis should represent the error, calculated as $\frac{wheelTurnAngle}{visualTurnAngle}$.

Evaluation:

Submit the following files on Canvas by 6pm on March 10th.

a. A zip file containing your code. (*Lastname1Firstname1_Lastname2Firstname2_code.zip*)
b. An image of the mosaic generated with your code based on images 0-4 in ImageSet2. (ImageSet2Mosaic.jpg)
c. An image file showing the odometry offset graph over five trial runs.
d. Each partner should complete the peer evaluation form (please complete this even if you a working without a partner).

Grading Rubric

Implement *propose_pairs*	20 points
Implement *homog_ransacI*	20 points
Implement *homog_dlt*	20 points
Implement *rot_form_homog*	5 points
Implement *extract_angle_y*	5 points
Plot of 5 wheel vs visual odometry runs	10
Mosaic in-lab demo	15 points
Peer evaluation form	5 points

Notes:

- You will have to install OpenCV 2.4 and numpy for this lab. Note that later versions of OpenCV are not easily compatible with the starter code being provided.
- A few helper functions already remain in the code
- When capturing your own images, make sure to only take pictures when the robot is stationary to reduce motion blur.
- Do not modify the return statements in the functions you fill in.