Comp. 4102: Assignment #3
Due: Thursday March 31, 2017 at 4:00 PM

1) The goal of the first questions is to implement some code that performs calibration using the method described in the book; by first computing a projection matrix, and then decomposing that matrix to find the extrinsic and intrinsic camera parameters. Use the approach described in the slides. I have given you a program, written in C++ that uses OpenCV, called projection-template.cpp. This program takes ten given 3d points and projects them into a 2d image using the given supplied camera calibration matrix, rotation matrix and translation vector. Your goal is to write the two routines that are missing, which are `computeprojectionmatrix` and `decomposeprojectionmatrix`. The first routine computes the projection matrix using the method described in Section 6.3.1 of the book, and the second uses the method in Section 6.3.2 to decompose the projection matrix into a camera calibration matrix, rotation matrix and translation vector. It should be the case that the computed camera matrix, rotation matrix and translation vector are the same (or very similar) to the original versions that were used to create the projected points. This shows that your two routines are working properly. You hand in your program source and the resulting output file assign3-out created by running this modified program.
5 marks

2) The goal of this question is to create a program that take as input two images that are related by a rotation homography; a left (keble_a_half), middle (keble_b_long) and creates a single panoramic image (same size as keble_b_long) as output. This is done by warping the left "into" the middle image. I have made the middle image big enough to hold both the warped left and the original middle image. I have given you a program called akaze-match-template.cpp which takes these two images and computes a set of features that you can use to compute the homography between them. To actually compute the homography you use the routine findhomography(, , RANSAC) and then you use warperspective routine with the computed homography to warp the left image into an image of the same size as the middle image. In other words you warp img1 into img3, and after that you paste (essentially an OR operation) img3 into img2. You should output two images; warped, which is the warped version of img1, and merged which is the warped version of img1 (img3) combined with img2. I have included two images called warped and merged which show you how they should look like. Notice that the final merged image has some anomalies because of the OR operation. In real mosaicking programs you do not see these anomalies. Write down a short (one paragraph) description of how you would get rid of these visible anomalies. The answer is simple.
5 marks