Due on 14 February 2024

Computer vision - Assignment 3

1. This is the exercise pending from Assignment 1 because the image wasn;t available. I'm repeating it here for your benefit. If you have already submitted this assignment with another image from the collection, please ignore this problem and proceed to the next one.

   Aligning the 3 channels of the Prokudin-Gorskii Photo Collection:

   For this assignment we have chosen one of the images in the above mentioned collection - windmills.jpg. Download this image, split the image into three parts to extract the channels. The first channel is the blue channel, followed by the green channel, and the red channel. Keep the blue channel fixed and align the green and red channels to the blue channel in the following way:

   (a) To compute the best alignment between two channels, "slide" one image over the other i.e. search over a window of possible displacements say $[-10, 10]$ pixels, score each one using an image matching metric such as SSD, and take the displacement with the best score.

   (b) What is your best SSD value and displacement vector for each channel?

   (c) Once you find the best alignment between the channels, merge the 3 channels to get a coloured image.

   (d) Display the coloured image.

2. For this assignment, you may work in groups of 2 or 3. However, each student should submit their copy on Moodle. On each copy, please state the percent effort each individual member of your team has contributed to this assignment (numbers should add up to 100%).

   For calibration, OpenCV implements Zhang's algorithm. This algorithm inputs correspondences between 2D image points and 3D scene points over a number of images and outputs the intrinsic camera matrix as well as the radial distortion coefficients $k1$ and $k2$. It also computes the rotation and translation of the scene towards the camera for every image, but this information is not needed for calibration. The scene points of the correspondence have to be in the same plane and to obtain good results, this plane should be different for every calibration image. In practice, an easy way to achieve this is to print a chessboard pattern onto a piece of paper, attach this paper to a rigid surface and take the corners between the chessboard squares as the correspondence points.

   - Print and use this checkerboard calibration pattern.
   - Acquire and save at least 10 images for the calibration.
   - You may use the following tutorial to guide you through this exercise: `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html`
   - If you use any other online resource, please mention it in your references.

   (a) Create the set of correspondence points for each of your images. For the 3D scene coordinates, measure in centimeters and use $(0, 0, 0)$ as coordinates for the top left corner. In the image, measure in pixels using $(0, 0)$ as the top left corner. Add these correspondence points to the solution document.

   (b) Manually selecting the corner pixels in the image is tedious and we want to automate it. Use cv.findChessBoardC and cv.cornerSubPix to find and refine these positions automatically.

   (c) Use the function cv.calibrateCamera to calibrate your camera using your correspondences. Add the intrinsic camera matrix and the lens distortion parameters to your solution document. Format floats using printf("%.4f").

   (d) Calculate the reprojection error.