COMPUTER VISION - LAB 5 HINTS

Part 1: Keypoints, Descriptors and Matching

1. Load into a std::vector< cv::String > the filenames of the input images. You may use the OpenCV function `cv::utils::fs::glob()` with pattern a filter pattern based on '*'/'?' symbols (e.g., img*.bmp).
2. Load all the images into a `std::vector< cv::String >`.
3. Project all images by using the provided `PanoramicUtils::cylindricalProj()` function.
4. Compute keypoints and descriptors of each projected image (you can use either ORB or SIFT features, by creating the corresponding object with `ORB::create()` or `SIFT::create()`, respectively).
5. Compute the matches between consecutive projected images as described in the lab5.pdf document.
6. For each pair of consecutive images, estimate the x, y translation in pixels. This can be done by calculating the average translation dx, dy between the matched keypoints. To be robust against outliers, don't use all the matches: call the `findHomography()` function, with `CV_RANSAC` as third parameter. `findHomography()` will provide a rigid body transformation between the two images but also a mask that highlights the inlier actually points used to estimate the transformation. To compute the average translation, just use the points marked as inlier.
7. Prepare a large output image in which to draw the global landscape. For example, to compute the width of such image, consider the projected images widths, and the translations along x.
8. Draw each projected image into a submat of the output image, considering the computed translations. To select a submat of an image, you may use the operator `cv::Mat operator() (cv::Range rowRange, cv::Range colRange).`
9. To improve the final result, you could equalize the projected images with the function `cv::equalizeHist()` before copying them to the output image.