Panoramic Images Computer Vision Project Assignment

Joemah Magenya
Physics of Data Masters Student
University of Padua

2020

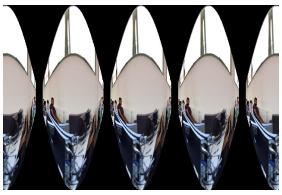
1 Report

This report describes c++ code to compute a Pano image from a given set of images with the help of opencv library.

Create a vector of strings to store the names of images in a folder. Using the opencv glob function, propagate to the folder containing the images you want to use, read all files using
cv::glob("foldername/*.imagetype", fname, false);
Further, create a vector of Mat objects to store images within the folder.
std::vector<cv::Mat> imgvector;
Iterate over the size of the string contain names of images, for each iteration, append each element to the vector of images.

In the PanoramicUtils class, Images from a given dataset are projected to cylindrical coordinates using the projectToClyldrical function which takes a vector of input images and returns a vector of cylindrical images 1. The function also has a variable scaling factor f which helps to set a better look for the projected images as well as a better alignment of images when computing the panorama. For example, 1a and 1b shows output images for different scaling factors. This shows that the bigger the scale factor the better the look of the projected image. The projectToClyldrical function serves to better align images together when merged to compute the Pano. After the images have been made cylindrical, they contain black borders. These borders makes the alignment of images difficult, since they create a black space for pairs of images which share similar features.

In order to get rid of these black rows and columns, the images are then feed to the clearBlackBorders which take an image and compare rows and columns and determine if there is a pixel in the column with color other than that indicated by keycolor. In this case it returns false. After the image colors have been compared, the function returns an image without borders surrounding it 2a and 2b. This also helps to remove the bridge between merged images 2.

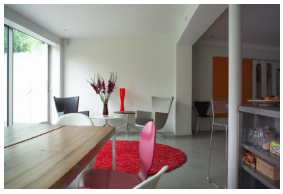


(a) Projected Image for f = 50

(b) Projected image for f = 500

Figure 1: Projected Image

(a) Projected Image with black rows



(b) Projected image without black rows

Figure 2: Projected Image

Furthermore, for each image, keypoints are detected and descriptors are extracted by applying the DetectAndCompute function which uses SIFT (Scale Invariance Feature Transform) a powerfull feature extractor. Keypoints are stored in a vector of a vector using
std::vector<std::vector<cv::KeyPoint» keypoints and the descriptors are stored in a vector using std::vector<cv::Mat> descriptors.

In addition, compute features for each pair of images found using match-pair function. Here, the Brute-force matcher (cv::BFMatcher(cv::NORM_L2)) is used to detect the matches which are then sorted in order to find the best matches. Filter out all good features found for each couple of images by comparing the distances between keypoints, if the distance is greater than the threshold reject the keypoints, otherwise consider them thus get the overlapping points. Store them in a vector containing a vector of matches
(std::vector<std::vector<cv::DMatch > > matches);.

Moreover, after all good matches are selected, using the findhomographs function, for every couple of images, select keypoints from all good matches found. The matches found are then visualised using the cv::drawMatches function 4. The keypoints and good matches are further compared and stored into a vector of floating points. The floating points are then used to compute the homograph for each pair of images. The homographs are stored in a vector of unsigned characters. Iterating over all matches for each pair of images, the position of x is computed for all good matches found corresponding to each pair of images.

After all keypoints, descriptors, homogragraphs and good matches have been computed, the images are then feed to the panorama function which compute

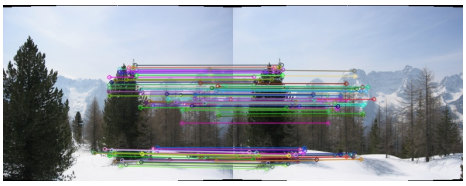


Figure 3: Good matches between a pair of images.

Figure 4: Good matches between a pair of images.

the final panorama for a given total number of images. Basically, the length of the Pano is given by the final x position dist_x for each pair of images computed in the findhomography function plus the width of the first image. Create an empty canvas with with rows and type similar to the first an element of the vector of images and of width given by the total length for all merged images. Iterate over the size of the vector of images and copy each image to a temporary panorama at the same time, increment the x coordinate of the image by appending all the added positions for every image to compute the final length of the Pano. For a better alignment and view of the final Pano computed **??**, the clearBlackBorders function indeed plays a crucial role to removing all the black rows separating the merged keypoints 12.

Sample results for different dataset::

Figure 5: Kitchen Dataset.

Figure 6: Kitchen Dataset

Figure 7: dolomites dataset.

Figure 8: dolomites dataset.

Figure 9: Lab dataset.

Figure 10: Lab dataset

Figure 11: Padova dataset by my phone

Figure 12: Padova dataset by my phone