360 Degree Panoramic Image Stitching

Yu Shi
Tandon School of Engineering
New York University
netID: ys3049
studentID:N12671812

Yiang Zeng
Tandon School of Engineering
New York University
netID: yz4420
studentID:N15860799

***Abstract*—This paper mainly concerns the problem of 360 spherical panoramic stitching using images taken by handy cellphone camera, while most previous approaches use professional 360 degree camera or has already known the angle of each rotation. In this work, SIFT feature detection is used to find homography matrix between two adjacent images vertically and along the equator. After that, rotation matrix between them can be computed based on homography matrix decomposition, where camera calibration are needed. Since we can pick an anchor image artificially, all the points in images can be rotated to corresponding spherical coordinates and converted into certain sub-part of panorama based on spherical transformation. Finally, we use gain compensation and blending to reduce the seam between pictures to form the entire panorama.**

I. Overview of the project

Camera calibration: Taking some pictures of chess board and implement zhang's method to get the camera intrinsic matrix.

Image feature extract, feature matching, and computing homography matrix: Implement SIFT algorithm to extract feature points of images. Implement HoG and k-nearest neighbor algorithm to find matches between two pictures. Calculate homography matrix by solving homogeneous linear equation with 8 parameters, and implement RANSAC algorithm to find the best solution for all matched pairs.

Inverse spherical projection and panoramic stitching: Implementing inverse spherical projection and computing the relative rotation matrix of adjacent matrix based on recursive method to get the panorama. Since this method using affine to stitch the every two sub-part panorama, which is proved to be not a good approach. Only result will be provided as contrast.

Alignment and Spherical Projection: Using camera matrix and homography matrix to get the rotation matrix of two pictures. Using matrix transfer to transfer all images' pixel correspond to anchor image coordinate. With the same coordinate, project all these pixel to a sphere expressed by angle theta and phi.

Compensation and Blending: Using central weight mask to give every pixel of the image a "weight". Stitch all pictures by there weight. For seam appear at the edge of overlapping area, implement pyramid blending to eliminate the seam.

II. Project accomplishment

A. Capturing images

In our approach, certain points in the spherical coordinate can be transformed into that in panorama, the relationship of which is illustrated in Figure 1. In Figure 2, where a camera is mounted on a tripod is the optimal condition. Since we are using handy cellphone, it's unable to bring a tripod with us. However, in order to get better results, we should simulate as similar as possible that we are a tripod and rotate the cellphone camera evenly.

The strategy to take images are is following: Firstly, we vertically rotate the cellphone for one certain αangles, and take images. Secondly, we horizontally rotate the cellphone for different βangles, and take another column of images. Repeat these two steps until all the direction are covered.

There are two things we must ensure. One is that since we are using SIFT feature detection, images with flag texture will be hard to find feature points, so we need to avoid the clear sky and similar floor. Another thing is that we must ensure that the adjacent images are overlapped partly.

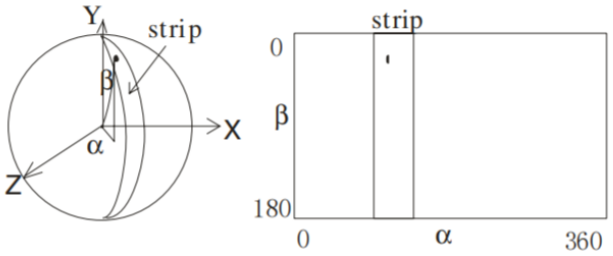

Figure 1. Representation of a spherical projection

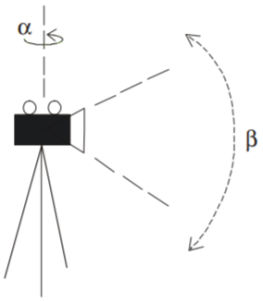

Figure 2. The way to capture the images

B. Camera Calibration

The relationship between real world points and camera coordinate points are based on extrinsic matrix and intrinsic matrix. While extrinsic matrix is due to camera rotation, translation and other camera movements, Intrinsic matrix is specific to a camera. It includes the focal length $_{f_x,f_y}$ and optical centers $_{c_x,c_y}$ of the pinhole modeled camera. It is also known as camera matrix. It depends on the camera lens, CCD or CMOS sensor and sort of hardware. Since once it is produced, it is fixed (so called intrinsic matrix). We only need to compute it once and store it for future computation. We need it to decompose the homography matrix to get rotation matrix, as will shown in the following sections. It is described as a 3*3 matrix:

$$K_{camera} = \begin{bmatrix} f_x & 0 & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix}$$

We got our camera calibrated by calling the code produced by opencv on this website.[**8**]Also, what we need to pay attention is that all the values acquired in the matrix are in pixel instead of millimeter.

C. Homography matrix decomposition

In order to get the homography matrix of two adjacent images, SIFT feature detection are used, which including feature-based registration,key points detecting, feature matching, RANSAC. You can turn to the reference of Prof. Wang for more information.

From above section we know that the projection of a object in space can translated to a image plane. The formula is:

$$w[x, y, 1] = K[RT][X, Y, Z, 1] \tag{1}$$

Now we unify all images which has homography matrix with anchor image into anchor image's camera coordinate. For those pictures which do not have same feature point to the anchor, using images which already got rotation matrix is a good idea. That is, if we knew the rotation and translation matrix from plane A to plane B as [R1 T1], and the rotation and translation matrix from plane C to plane A as [R2 T2]. The rotation and translation matrix from plane C to plane A can be represent as [R1R2 (R1T2)+T1]. Using this method we can get all rotation and translation matrix from other images to anchor images

From last section we know the meaning of extrinsic matrix and intrinsic matrix. From extrinsic matrix, we transfer global coordinates (X,Y,Z) into camera coordinates $_{(X_c,Y_c,Z_c)}$. From intrinsic matrix, we transfer camera coordinates $_{(X_c,Y_c,Z_c)}$ into image coordinates (x,y). In our algorithm, we set an anchor image as the beginning, which is tangency to the sphere that radius is focal length f. This means the anchor image has same $_{Z_c}$ coordinate f. And the anchor coordinate is anchor image's camera coordinate.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_0 \\ R_{21} & R_{22} & R_{23} & T_1 \\ R_{31} & R_{32} & R_{33} & T_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_c \\ 0 & f_y & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{3}$$

In order to project space coordinates onto a sphere, we need to unify all images coordinates into one coordinate system. In our case is anchor image's camera coordinate. For all images, we transfer global coordinates into their image images. The homography matrix is the transform matrix from one image coordinate to another image, which is

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{4}$$

Using intrinsic matrix we have:

$$K_{camera} \begin{bmatrix} X_{c2} \\ Y_{c2} \\ Z_{c2} \end{bmatrix} = HK_{camera} \begin{bmatrix} X_{c1} \\ Y_{c1} \\ Z_{c1} \end{bmatrix} \tag{5}$$

To unify all images coordinates into anchor image coordinate. We use rotation matrix and translation matrix. As shown in equation , The transformation of two camera coordinate can represent by rotation matrix R and translation matrix T, which is:

$$K_{camera} \begin{bmatrix} X_{c2} \\ Y_{c2} \\ Z_{c2} \end{bmatrix} = R \begin{bmatrix} X_{c1} \\ Y_{c1} \\ Z_{c1} \end{bmatrix} + T \tag{6}$$

We assume all images camera coordinate never change their focal length and the principle point never shift, which means translation matrix equals to zero and all intrinsic matrix are the same.Finally, we have:

$$\begin{bmatrix} K^{-1}HK & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{7}$$

OpenCV already have a function to calculate Rotation and translation matrix from homography matrix and intrinsic matrix.

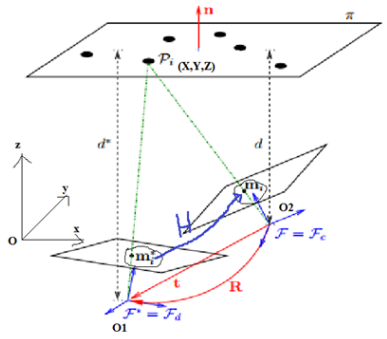

Figure 3. The transformation from current frame and next frame

To get the position relationship between two pictures. Past essay had talked a lot about the decompensation of homography(Ezio Malis and Manuel Vargas,2007, Deeper understanding of the homography decomposition for vision-based control).In a easier understand way. We build two coordinates for two camera position which can then get the point in the space coordinate $_{[X,Y,Z]^T}$ to the projection to two image plane:

$$[x_1, y_1, 1]^T = w^{(-1)} K_1 [R_1 T_1][X, Y, Z]^T \tag{8}$$

$$[x_2, y_2, 1]^T = w^{(-1)} K_1 [R_2 T_2][X, Y, Z]^T \tag{9}$$

Because we use the same camera, the scale factor w are the same. Scale factor can be easily calculated by the size of image. The Intrinsic matrix is known above. So the relationship between two image can then represent as:

$$K_1^{(-1)}[R_1 T_1]^{(-1)}[x_1, y_1, 1]^T = K_2^{(-1)}[R_2 T_2]^{(-1)}[x_2, y_2, 1]^{T_1} \tag{10}$$

And from above section we know that homography is the translation matrix which can represent the relationship of a point between two images. Using the coordinate we just talked about we have

$$[x_1, y_1, 1] = H[x_2, y_2, 1]^{T_2}. \tag{11}$$

From formula above,we have H = K' [R' T']. So we can get the rotation matrix and translation matrix of two corresponded images using homography and camera matrix we got above. OpenCV provide a decomposition function called retval, rotations, translations, normals = decomposeHomographyMat(H,K) which can help us get the rotation matrix. Because we use rotation matrix between target image and anchor image. we can get homography for nearby picture which has overlap area to get the same feature points. For those pictures which do not have same feature point to the anchor, using images which already got rotation matrix is a good idea. That is, if we knew the rotation and translation matrix from plane A to plane B as $_{[R_1,T_1]}$, and the rotation and translation matrix from plane C to plane A as $_{[R_2T_2]}$. The rotation and translation matrix from plane C to plane A can be represent as $_{[R_1R_2(R_1T_2)+T_1]}$. Using this method we can get all rotation and translation matrix from other images to anchor images.

D. Projection Transformation

Suppose the origin of camera based coordinates is equivalent to that of the real world coordinates, the relationship between a real world point **p** and its homogeneous coordinates **q** of image is described as:

$$\mathbf{q} = K_{camera} * R * \mathbf{p} \tag{12}$$

where

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi_i & -sin\phi_i \\ 0 & sin\phi_i & cos\phi_i \end{bmatrix} \begin{bmatrix} cos\theta_i & 0 & sin\theta_i \\ 0 & 1 & 0 \\ -sin\theta_i & 0 & cos\theta_i \end{bmatrix}$$

$_{K_{camera}}$ is camera matrix acquired by camera calibration, $_{R_i}$ is rotation matrix, $_{\theta_i,\phi_i}$ are the rotated angles in $_{\theta,\phi}$ directions. While, in our approach the rotation matrix is acquired directly by decomposing the homography matrix instead of computing the rotation angles.

The camera coordinate, the origin of which is the optical center of the lens, is illustrated in the Figure 3. Based on the principle of the camera, the images taken by the same camera have the same optical center and focal length. We can assume that all the image originally located in the Z=f plane and optical center $_{(c_x,c_y)}$ is the on the Z axis. X and Y are the pixel indices of the image coordinate starting from the top-left corner. In addition, the focal length f is also in pixel value instead of millimeter.

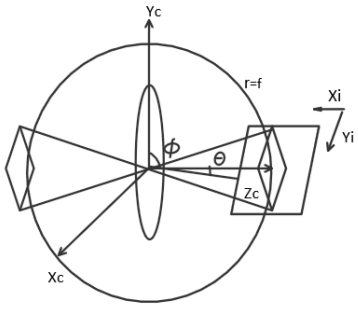

Figure 4. Camera coordinate of anchor image

After rotation, corresponding coordinate can be used to compute the $_{\theta,\phi}$ of certain points.

$$\theta = arctan(X_c/Z_c) \tag{13}$$

$$\phi = arctan(Y_c/\sqrt{X_c^2 + Z_c^2}) \tag{14}$$

where

$$\mathbf{q} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

computed by equation (12).

So that, we can get corresponding coordinates of the real world ones on the spherical panorama located in the domain θ: [-π, π], φ: [-π,π]

In order to form the panorama, setting the angle difference$_{\Delta_x,\Delta_y}$ between to adjacent pixels can determine the

resolution of panorama. Based on this, the coordinates can be convert to pixel indices in the final panorama image.

$$i = round(\theta/\Delta_x) \tag{15}$$

$$j = round(\phi/\Delta_y) \tag{16}$$

where using round() is due to the indices of image have to be integer.

Here is the anchor image after spherical projection transformation.

Figure 5. Spherical transform result of anchor image

This is the rotated image after spherical projection transformation.

Figure 6. Spherical transform result of rotated image

E. Image compensation and blending

In previous sections, we described how to determine picture's pixel position on final pictures, which represent theta and phi. In this section, we show how to solve for a photometric parameter, namely the overall gain between images. And how to deal with intensity difference between images. If all of the images are in perfect registration and identically exposed, compensation and blending is an easy problem (any pixel or combination will do). However, for real images, visible seams (due to exposure differences), blurring (due to misregistration), or ghosting (due to moving objects) can occur. We tried averaging, feathering, and central weighting to deal with overlapping area.

$$I_{i,j} = \frac{\sum_n [I_{n(i,j)} W_{n(i,j)}]}{\sum_n W_{m(i,j)}} \tag{17}$$

In upper formula, Ii,j represent the final pixel value in overlapping area. In means pictures which have overlapping area. W is the weight function of each image. For pixel weighting, our approach is to averaging is to weight pixels near the center of the image more heavily and to down-weight pixels near the edges. When an image has some cutout regions, down-weighting pixels near the edges of both cutouts and edges is preferable. This can be done by applying a gaussian function, computing a distance map or grassfire transform. Compensation and weighting function can significantly reduce the seam between images. However, when rotation angle is small or the exposure change between two images is too large. You will also find seam between images. A pyramid blending can fix this seam. Our approach is to down sample the images and their weight masks. We initialize blending mask for each image by finding the set of points for which image n is most responsible. That is image n projected to sphere (θ, φ) has maximum weight, we set its value to 1, and 0 where some other image has a higher weight. These are the results acquired by inverse spherical projection.

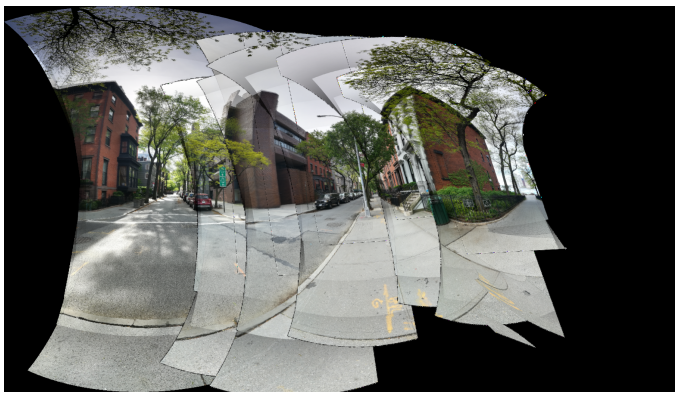

Figure 7. Inverse spherical projection without blending

Figure 8. Inverse spherical projection with blending

Here are the panoramas produced by spherical projection.

Figure 9. One result

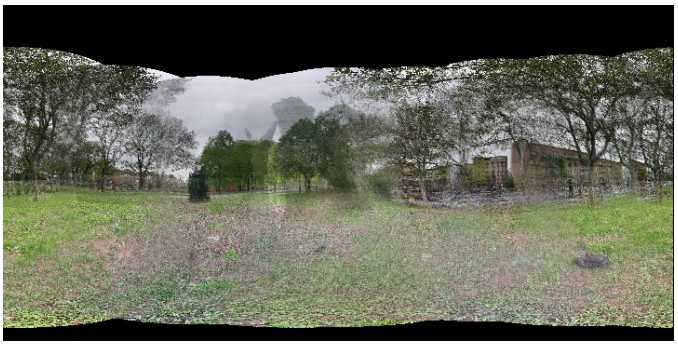

Figure 10. Another result

III. Summary

Our approach can produce a 360 spherical panoramic image using camera pictures. The key idea is to find out all pixel's space coordinate using the same Coordinate System which original point is the center of the sphere. For far objects panorama, we can just use rotation matrix to got a good panorama, but for objects which has different distance to camera, translation matrix needed to be added. And because of the cumulative error there will be some mismatch for those pictures far away from anchor image. This may cause blur and ghosting. Local alignment is a good way to reduce this problem. Our future work may focus on local alignment and more complex compensation method to optimize our final result

Acknowledgment

Thanks for the help of Prof. Wang Yao and TA Yixiang Mao.

References

[1] Brown Matthew, Lowe David. *Automatic Panoramic Image Stitching using Invariant Features.*, International Journal of Computer Vision. Aug2007, Vol. 74 Issue 1, p59-73. ⟨http://eds.a.ebscohost.com.proxy.library.nyu.edu/eds/detail/detail?vid=1&sid=1de7c5b4-fff2-4c92-8d3c-c5065b12fa6a%40sdc-v-sessmgr03&bdata=JnNpdGU9ZWRzLWxpdmU%3d#AN=24940147&db=a9h⟩

[2] Shum H.-Y, Szeliski R. *Systems and Experiment Paper: Construction of Panoramic Image Mosaics with Global and Local Alignment.*, NTERNATIONAL JOURNAL OF COMPUTER VISION. 36(2):101-130. ⟨http://eds.a.ebscohost.com.proxy.library.nyu.edu/eds/detail/detail?vid=1&sid=044d2672-e276-481f-93fa-efa8b5a013e2%40sessionmgr4009&bdata=JnNpdGU9ZWRzLWxpdmU%3d#db=edsbl&AN=RN077019819⟩

[3] R. Szeliski, *Image alignment and stitching: A tutorial*, Journal of Foundations and Trends in Computer Graphics and Vision, Volume 2 Issue 1, January 2006, Pages 1 – 104 ⟨http://research.microsoft.com/pubs/75695/Szeliski-FnT06.pdf⟩

[4] Fischler, M. A. and Bolles, R. C. (1981). *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.*, Communications of the ACM , 24(6):381–395. ⟨https://dl-acm-org.proxy.library.nyu.edu/citation.cfm?doid=358669.358692⟩

[5] Richard Szeliski, *Computer Vision: Algorithms and Applications.*, 2012.Sec. 2.1, 4.1, 4.2, 6.1, 9.

[6] Yao Wang. Feature Detection and Feature Descriptors. Image and Video Processing lecture 7.

[7] Yao Wang. Image Alignment through Feature Correspondence. Image and Video Processing lecture 8.

[8] OpenCV.https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html