Skybox

Ruoqi He & Chia-Man Hung

February 26, 2016

1 Introduction

In this project, we present a method to construct a skybox from a series of photos we took ourselves. It is a graphical procedure of creating an entire spherical background of a scene. The main idea consists of minimizing the error of the rotation matrices of the camera.

To represent a 360 degrees panoramic view covering all the directions, the simplest way is that of a skybox. A skybox consists of six images: one per face of the cube and the camera is located at the center of the cube. Our goal is to project the photos correctly onto the cube to construct a skybox. The first difficulty we encountered is to obtain appropriate photos to build the panorama. Our problem is then to determine the rotation and the scale parameter associated to every photo.

2 Evolution

2.1 Taking photos

We would like to make a skybox of the Grand Hall. We used the camera Canon EOS 70D with the lens Canon EF 28-135mm f/3.5-5.6 USM IS. We took several rings of pictures by varying the angle of elevation. Each photo ring consists of around twenty individual photos with a quasi-uniform angle between two consecutive photos. The pictures are taken in a fixed position using a tripod with the same manual configuration, including focal length, aperture, shutter speed, ISO, and white balance (non-auto). The photos are then processed by the Adobe Lightroom software to remove the distortion using the corresponding lens profile. Photos are scaled to 1280x960 pixels.

2.2 Finding homography between two photos

Before matching every photo to construct a skybox, we have to know how to match two photos, i.e. find homography between two photos having one part in common. This is done in our main code *findCentricPhotoHomography* by using the OpenCV library.

First, we extract some characteristic points named AKAZE. Then, we match these points. After the matching, we compute the homography by the RANSAC algorithm (*findHomography*). By applying the homography, we can project the points of the first photo onto the second one and thus combining the two photos.

2.3 If only the world was perfect

In an ideal world, everything is perfect. All we have to do is to compute the homography between pairs of photos and project them one after another. However, when applying this method to the central ring of photos, we observed that the product of all the homographies is not close to the identity - there was actually a big accumulated error. It is difficult at first to correct the error - just multiplying all homographies by a power of the error matrix has no physical meaning. It is also difficult to project a photo onto the 3d cube only knowing his homography with the reference photo. We will now seek to know the rotation of the camera for each photo using homographies.

2.4 Estimating the rotation

We then think of estimating the rotation of the camera for every photo from the homography of two consecutive photos and computing the projection onto the skybox. There is a direct relationship between a homography and a rotation, which will be explained in details in Section 2.4.1. We try to minimize the error corresponding to the rotation of the camera by using a non-linear solver, which will be explained in Section 2.4.2.

2.4.1 Homography-rotation relationship

Given two photos having a part in common, numbered 1 and 2 respectively, we have the following relationships.

(1) $H_{1/2}m_1 = m_2$
(2) $R_1M_{|1} = R_2M_{|2} = M$
(3) $R_{2/1}M_{|2} = M_{|1}$

Figure 1: First attempt – the last photo does not match the first one well enough

(4) $K_i[R_i^{-1}|0]M = m_i$, $i \in \{1,2\}$ où $K_i = \begin{pmatrix} \alpha_i & & 0 \\ & \alpha_i & 0 \\ & & 1 \end{pmatrix}$

since we consider the center of the photo as the origin of the pixel coordinates (m_i)

The above relationships are in homogeneous coordinates.

Notations:

m_i, $i \in \{1,2\}$ is a point of the photo i in the pixel coordinates of the photo i.

$H_{1/2}$ is the homography which transforms the pixel coordinates of a point in the photo 1 to the pixel coordinates of the same point in the photo 2.

$M_{|1}$, $M_{|2}$ et M are the coordinates of the same point in the frame of reference of the camera 1 and 2 and in the universal frame of reference respectively.

R_i, $i \in \{1,2\}$ is the rotation of the camera i in the universal frame of reference.

$R_{2/1}$ is the rotation of the camera 2 in the frame of reference of the camera 1.

K_i, $i \in \{1,2\}$ characterize the camera i.

α can be characterized by the distance in pixels between the optic center and the virtual retina. When increasing α, it reduces the field of view.

Solution of the equations:

$$H_{1/2} = \begin{pmatrix} 1 & & \\ & 1 & \\ & & \alpha_2^{-1} \end{pmatrix} R_{2/1}^{-1} \begin{pmatrix} 1 & & \\ & 1 & \\ & & \alpha_1 \end{pmatrix}$$

In our implementation, we consider the frame of reference of the camera corresponding to the first photo of the central ring as the universal frame of reference.

Estimation of α:

Assume that α is the same for all photos. One can express the rotation R depending on α by the homography-rotation relationship by knowing the homography. To make R a rotation, R should verify $RR^T = I_3$, which can be translated to an equation of α. This equation is used to calculate our α. After doing so, we see that the α calculated varies considerably (about 10 %) of a homography to another. Different α are finally tested to ensure that the last photo of the central ring transformed by the corresponding homography match the first one well enough.

2.4.2 Solving with ampl

The problem we face is actually a quadratic optimization problem. We consider α as a constant and the rotation matrices of the camera as variables. There are 9 variables for a photo. As input, we have the 9 initial points and their transformation after applying the corresponding homography calculated as in Section 2.2 for each pair of photos. We try to minimize sum of the error for every pair of photos defined by the distance between the 9 transformed points from the input and the 9 points obtained by applying the homography computed from the rotation of the camera to the initial points. We have constraints to ensure that rotation matrices are really rotations $RR^T = I_3$ and that the first rotation matrix is the identity. Some auxiliary variables are added to make the objective function a linear function and to deal with the homogeneity.

The whole thing is written in Ampl and tested with two different non-linear solvers: BARON and Convex Over and Under ENvelopes for Nonlinear Estimation (COUENNE). BARON seemed to be completely blocked and COUENNE never converged (after six hours of computation).

3 Final solution

3.1 Optimizing with Ceres solver

As mentioned on its website, Ceres Solver is an open source C++ library for modeling and solving large, complicated optimization problems. It is a feature rich, mature and performant library which has been used in production at Google since 2010. Ceres Solver can solve two kinds of problems: Non-linear Least Squares problems with bounds constraints and General unconstrained optimization problems. We were convinced to be able to solve our problem with Ceres.

The main structure of the modeling the problem is almost the same but this time, the rotation is represented by a 3d vector whose direction is the rotation axis and whose magnitude is the angle of the rotation (angle axis representation). The α is also set as a variable for each photo. To sum up, there are four variables for each photo and we do not need any constraints. We set simple initial values to the variables by placing the photo evenly in order not to be stuck at an unwanted local minimum.

3.2 Interlude: Let's give ampl another chance

After using the Ceres solver, we realized that it is better to consider this problem as a local minimization problem. We therefore tried three other local minimum solvers with initial values for the rotation: knitro, minos and snopt. We again tested with the central horizontal ring with the intial directions of the photos evenly distributed on a circle. Knitro and minos all converged on an infeasible point. Snopt said Nonlinear infeasibilities minimized. None of the results satisfies the orthonormality of the rotation matrices.

The reason why we did not use the angle axis representation previously when solving with ampl is that we thought we would have more chance to solve our problem with quadratic constraints. Now, we are actually having too many constraints and the solvers cannot satisfy all of them. So we tried to write in ampl exactly the same inputs, variables, and constraints as used in Ceres solver. This time the objective function and the intermediate constraints are no longer quadratic, but every intermediate constraint can be directly computed from the previous ones. We first tried with the central ring of the photos. Minos returned exactly the initial values claiming that the local minimum had been reached, which is obviously wrong. Snopt almost immediately finished (Nonlinear infeasibilities minimized) but the values were still almost the same as the initial values. Knitro did not converge after 10000 iterations (default number of iterations) and returned a result with

higher infeasibility after 20000 iterations. The result after 20000 iterations is displayed and shown in Figure 2. Since it is already incorrect, we did not try to apply the solver to all the photos.

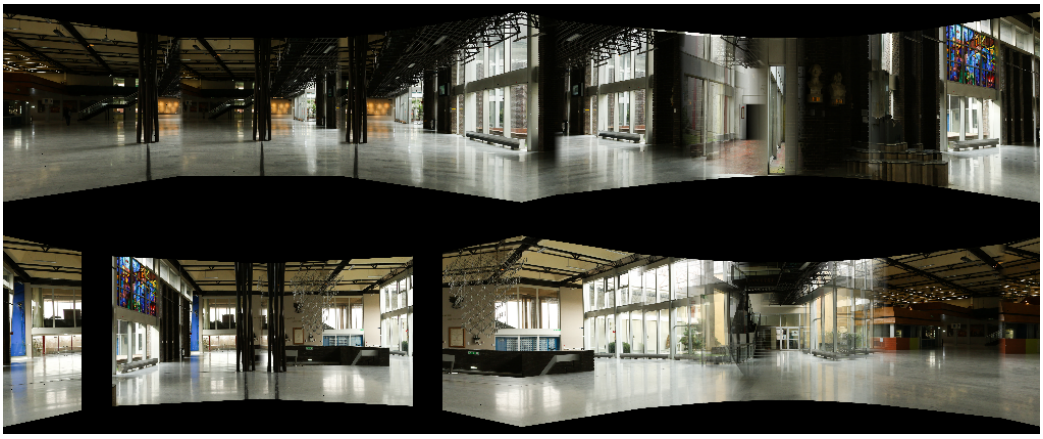

Figure 2: The result after 20000 iterations of knitro

3.3 Matching the photos

We obtain the rotations and the α of the camera corresponding to each photo as the output of the Ceres Solver. These values allow us to compute the homography between a photo and a reference photo (see Section 2.4.1) and thus to construct our skybox.

Figure 3: Choice of the color – left : maximal value / right : Z-Buffer + interpolation

Several photos can be superposed at the same place, so we have to decide how to choose the color we would like to use for our skybox. We first chose to use the maximal value of the colors for every pixel. The skybox generated this way is quite satisfying and it helps us to detect the eventual problem of the placement of every photo. To make it even better, we chose at last to use a sort of Z-Buffer. The idea consists of placing the virtual retinas of the photos

around the camera, and we see the "closest" image as illustrated in Figure 4. To achieve this, we create a depth map in addition (Z-Buffer) for every face of the skybox. During the projection, the pixel with the smallest depth is chosen. We can still improve the result by implementing an interpolation zone. If the new pixel has a depth close to the one of the existing pixel, we interpolate these colors according to the difference of the depth. See Figure 3 for the comparison and Figure 4 for the principle of Z-Buffer and of interpolation.

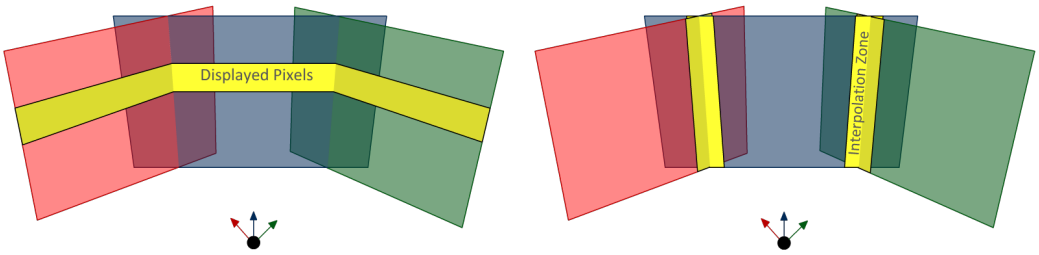

Figure 4: Matching of the photos by a Z-Buffer and interpolation

However, it is possible that the homography computed previously and set as input is obviously wrong, due to the lack of characteristic points. This causes some problems when solving the minimization problem. We decide to use a Huber loss function, which reduces the impact of incorrect homography. If by any chance there are still some misplaced photos, they can be identified by the abnormal α value and we can choose not to display them. The result can be still satisfying since photos are in general largely overlapped. In the end, we actually successfully configured the solver such that all photos are correctly calculated and displayed, even if many of the homographies are completely wrong.

4 Results

See Figure 5.
In total, we used 204 photos.

Once we obtain the six faces of the cube, we visualize the generated skybox using Unity. See the demos.

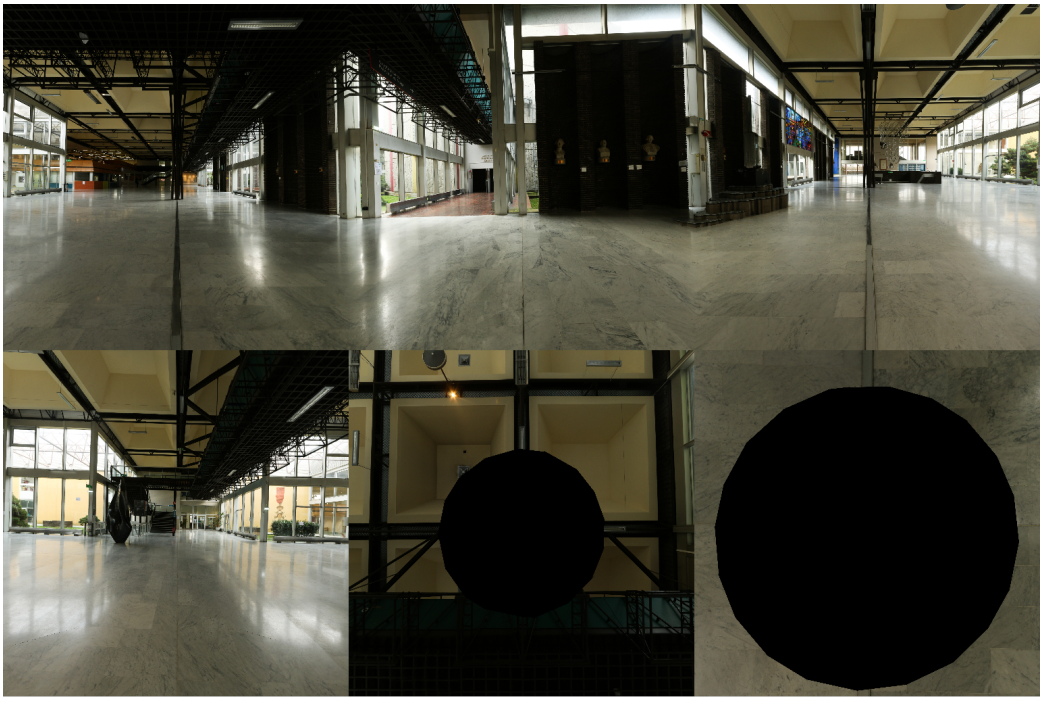

Figure 5: The six faces of the skybox of the Grand Hall

5 Remaining problems

There are still some problems due to our method or associated to our material.

- Comparison between two photos without any characteristic point (Akaze in our case)

  For the ceiling, the wall, the floor, the sky, and there are often very few characteristic points, even after increasing the contrast of the photos. The Ransac algorithm therefore can not be applied to find the homography between two adjacent photos. Thus our method, based on the homography, does not work.

- Optic center of the camera not fixed

  See Figure 6.

  When we turn the camera around on the tripod to take rings of photos, the optical center of the camera is not a fixed point as we cannot take into account the unknown shift in our calculation. More specifically, the optical center is in the middle of the lens, which itself rotates around of the rotation center of the tripod.

Figure 6: Optic center of the camera not fixed

- Distorsion

  Even after the correction of distortion by Adobe Lightroom software, the distortion is less serious but still exists.

6 Conclusion

We managed to determine the rotation of our photos, which is the heart of our problem, with the homography-rotation relationship, the calculated homography between two adjacent photos by the Ransac algorithm, and the minimization of self-defined error in the Ceres Solver. We projected the photos onto the six faces of the cube. The matching of the photos is improved by a Z-Buffer and interpolation. The results we obtained is quite satisfying. The Unity demos are very realistic. Nevertheless, the top face and the bottom are not completed because of the limit of the material and some small flaws can be observed. With a special panoramic patella, the problem of optic center could be solved and we might also be able to fill up the whole skybox.