Skybox Creation by Photo Merging

Chia-Man Hung, Ruoqi He

CDT Autonomous Intelligent Machines & Systems, Department of Engineering Science, University of Oxford

Introduction

To reduce 3D scene complexity, video games often employ skyboxes to present distant sceneries. A skybox is a graphical procedure to represent a 360 degrees spherical panoramic view covering all directions. It consists of six square images forming a cube, such that when viewed from the centre they appear as the entire view of the environment.

Skyboxes can be easily generated from virtual 3D scenes, but reconstructing real-world scenes from photos is a more challenging subject. Unlike techniques of generating panoramic images based on digital rotating cameras or a cluster of cameras like the ones used by Google Street View and Cyclomedia, which are often pricey and thus less accessible to the public, we aim for a low-cost method using minimal equipment but of high quality. Our problem consists of taking photos, combining and projecting them onto a skybox. After taking photos from approximately the same position covering all directions, our main challenge is to determine the rotation and the scale parameter of the camera associated to every photo, and to merge the photos in a seamless way.

skybox

Methods

Stitching two photos taken from the same position together is a well-studied problem. A classical solution consists of first detecting feature points in both photos, matching them and then applying the Random Sample Consensus (**RANSAC**) algorithm to determine an adequate homography between two images.

homography

We derived a **relation between the rotation and the homography** found from the previous approach.

(1) $H_{1/2}m_1 = m_2$

(2) $R_1M_{|1} = R_2M_{|2} = M$

(3) $R_{2/1}M_{|2} = M_{|1}$

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

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

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.

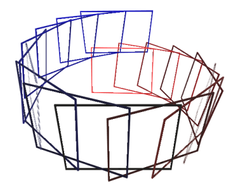
$R_i,\ i \in \{1,2\}$ is the rotation of the camera i in the universal frame of reference.

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

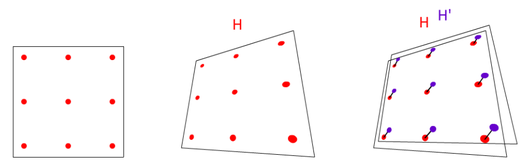
$K_i,\ i \in \{1,2\}$ characterize the camera i.

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.

Simply computing the homography-induced rotations to place our photos results in significant **accumulated errors**. A photo does not match itself after a round of pairwise matching. We need a mechanism to correct the rotation estimation.



The problem of rotation estimation can be reformulated as a **quadratic optimisation** problem. We consider α (1 value) and the rotation of each photo in angle axis representation (3 values) as variables. As input, we have 9 initial points and their transformation after applying the corresponding homography calculated by the previous method for each pair of photos. We minimise the sum of the error for every pair of photos defined as the distance between the 9 transformed points from the input and the 9 points obtained by applying the rotation-induced homography to the initial points. We do not have any constraints.

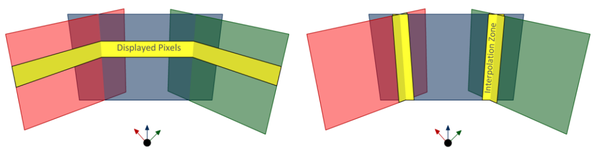


Definition of distance between homographies

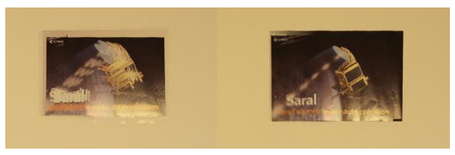
This optimisation problem is then solved by **Ceres with dense Schur solver**. Ceres is an open source C++ library for modelling and solving large, complicated optimisation problems. It can solve two kinds of problems: Non-linear Least Squares problems with bounds constraints and **General unconstrained optimisation problems**.

Some auxiliary variables are added to make the objective function a linear function and to deal with the homogeneity. We set simple initial values to the variables by placing the photo evenly in order not to be stuck at an unwanted local minimum.

By using **blending** on boundaries of Voronoi regions generated by the centres of the photos, we achieved further visual improvements.

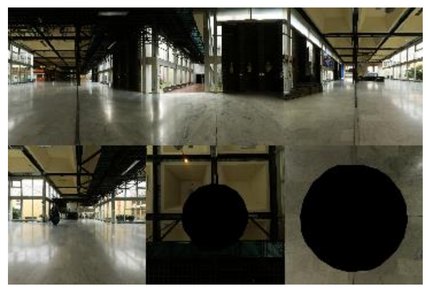


Mapping of the photos by a depth-buffer and interpolation



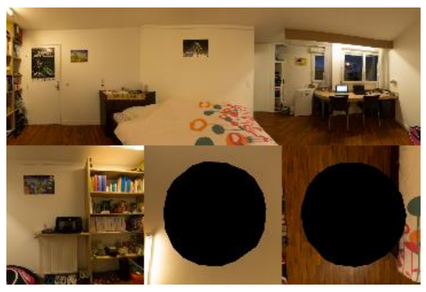
Choice of color
Left: max / Right: depth-buffer + interpolation

Results



The six faces of the skybox of Ecole Polytechnique's Grand Hall.

(Merged 204 photos.)



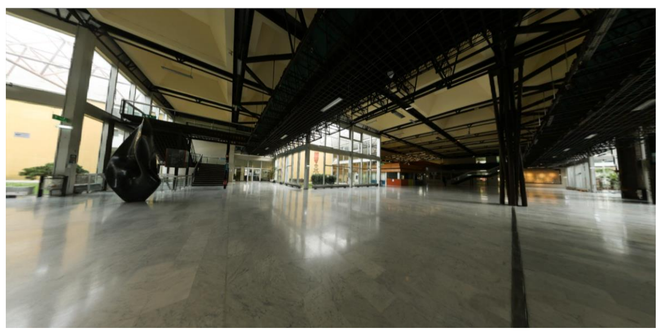
The six faces of the skybox of my previous room.

(Merged 59 photos.)

Once we obtain the six faces of the cube, we visualise the generated skybox through Unity.

Conclusion

We determined the rotation of our photos, which is the heart of our problem, from the homography-rotation relationship, the calculation of homographies between adjacent photos by the Ransac algorithm, and the minimisation of a 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 depth-buffer and interpolation. The results we obtained are 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 centre could be solved and we might also be able to fill up the whole skybox.



A wide-angle shot of the skybox in Unity

References

[1] Fischler, M. A., & Bolles, R. C. (1987). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision* (pp. 726-740).

[2] Shoemake, K., & Duff, T. (1992, May). Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface* (Vol. 92, pp. 258-264).

[3] Van Waveren, J. M. P. (2005). From quaternion to matrix and back. *Id Software, Inc.*

[4] Agarwal, S., & Mierle, K. (2012). Ceres solver: Tutorial & reference. *Google Inc, 2*, 72.

For more information:

chiaman@robots.ox.ac.uk

https://ascane.github.io/

Unity demo