Calibration and Mapping for the Omnidirectional Image

Zihua Xu

INRIA
March 30, 2020

Abstract. This report mainly discuss the calibration and mapping for the omnidirectional images . We first build the projection model of omnidirectional images. And then we do the calibration to estimate the parameters of the projection model. Finally we can build the mapping function between the omnidirectional images and the equirectangular images, which a presentation way for the latter processing method of the omnidirectional image.

1 Omnidirectional Camera Calibration

1.1 Introduction

Unified Camera Model Calibration To correctly build the relationship of the 3D points in world and the pixel points in the omnidirectional images, we need to build the projection model for that. One of the most adopted model is the unified camera model (UCM) [Geyer and Daniilidis, 2000]. The structure of this model is shown in Figure 1. This model allows the projection from 3D points to pixels ($p \in \mathbb{P}^2$) on the omnidirectional image following four steps:

- Projecting the 3D points onto a unit sphere, is done by a spherical normalization as:
$$P_s = \frac{P}{\|P\|} = (x_s, y_s, z_s)$$
.
- The change of the camera frame with translation ϵ along Z. The choice of ϵ is depend on the shape of the mirror (between 0 to 1);
$$P_\epsilon = (x_s, y_s, z_s + \epsilon).$$
- The point P_ϵ is then projected onto the normalized image plane with a perspective transformation (coordinates divide by $z_s + \epsilon$), distant 1 along the positive direction of Z axis from C_ϵ:
$$m = (\frac{x_s}{z_s + \epsilon}, \frac{y_s}{z_s + \epsilon}, 1) = (x_m, y_m, 1).$$

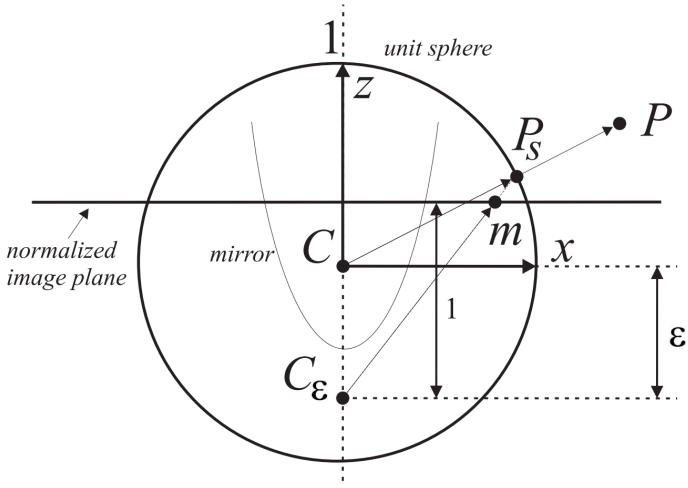


Fig. 1. The unified camera projection model (UCM) [Scaramuzza, 2014].

– Finally, the pixel point $p = (u, v, 1)$ is given by the intrinsic-parameter matrix K:

$$p = Km, K = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

So there are totally 5 intrinsic parameters $(\epsilon, \alpha_u, \alpha_v, u_0, v_0)$ we need to estimated. Generally, these parameters can not be directly estimated in images, but by using the calibration pattern. Also we need to estimate the extrinsic parameters (transform from the world frame to the camera frame) and the distortion parameters (caused by imperfect physical system). After combine all the projection models we can denote the total projection model as G. And then we can finally turn the calibration problem into the optimization problem as [Mei and Rives, 2007]:

$$\boldsymbol{V} = argmin \frac{1}{2} \sum_{i=1}^{m} [G(\boldsymbol{V}, \boldsymbol{g}_i) - \boldsymbol{e}_i]^2 \tag{1}$$

where $\boldsymbol{V}$ is the total parameters, $\boldsymbol{e}_i$ is the pixel coordinate and $\boldsymbol{g}_i$ is the 3D coordinate. This formula aim at find a set of parameters that can minimize the distance between the pixel points in camera frame and the points project from 3D world frame. To solve optimization problem there are two steps, initialization and find a direction to improve the result.

– The optimization of calibration is not convex, which means it need to set a proper initial value, otherwise it can not find the global minimum. So we can

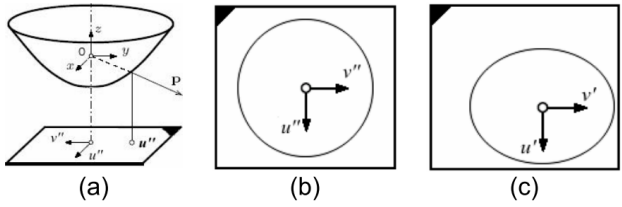


Fig. 2. (a) Coordinate system in the catadioptric case. (b) The perfect projection plane. (c) The actual camera image plane, expressed in pixel coordinates. (b) and (c) are related by an affine transformation.[Scaramuzza et al., 2006a]

set the initial value of the parameters based on different rules (experience, theory, assumption).

– In many situations if we initialize properly, gradient descent is good enough to do the optimization. And gradient descent is simple to apply, that's why we choose it. The final result also prove this method worked.

Extended UCM Calibration In [Scaramuzza, 2014], it treats the imaging system as a unique compact system. The relationship between the 3D points and pixel points is shown in Figure 2.

Let us note the $\boldsymbol{x}$ the scene point, and the $\boldsymbol{u''}$ be the perfect project point on image (assume that the physical system is perfect). So we will have the follow relationship:

$$\lambda * g(\boldsymbol{u''}) = P\boldsymbol{x}, \tag{2}$$

where λ is the depth scale, function $P \in \mathbf{R}^{3\times4}$ is the transfer matrix that project $\boldsymbol{x}$ from world frame to camera frame and $g(\boldsymbol{u''}) = (u'', v'', f(\rho))^T$. Here $f(\rho) = a_0 + a_1\rho + a_2\rho^2 + \cdots + a_N\rho^N, \rho = \sqrt{u''^2 + v''^2}$.

So if the physical system is perfect, then we just need to calibrate the coefficients of function f and the extrinsic parameters of P. However, the physical system is not perfect, so the real point $\boldsymbol{u'}$ has the relationship as $\boldsymbol{u''} = \boldsymbol{A}\boldsymbol{u'} + \boldsymbol{t}$. $\boldsymbol{A}$ is a 2 by 2 affine matrix for distortion, and $\boldsymbol{t}$ is the translation between image center and the omnidirectional image center. So also need to calibrate the $\boldsymbol{A}$ and $\boldsymbol{t}$.

As the degree of the polynomials, most situation the higher one means the better accuracy, but not for the polynomial from camera to world. So we need to choose it depend on the mapping result. From the world to camera we can get the point direction directly from the x, y coordinates, but we also need to know the distance from the center to the 2D points. So we can also use a polynomial to reconstruct the distance. This inverse function is $f_{inv}(\theta) = \rho = a'_0 + a'_1\theta + a'_2\theta^2 + \cdots + a'_N\theta^N, \theta = atan(\frac{z}{\rho})$. And this polynomial will have a better performance with higher degree. So we just increase the degree until the error is small enough.

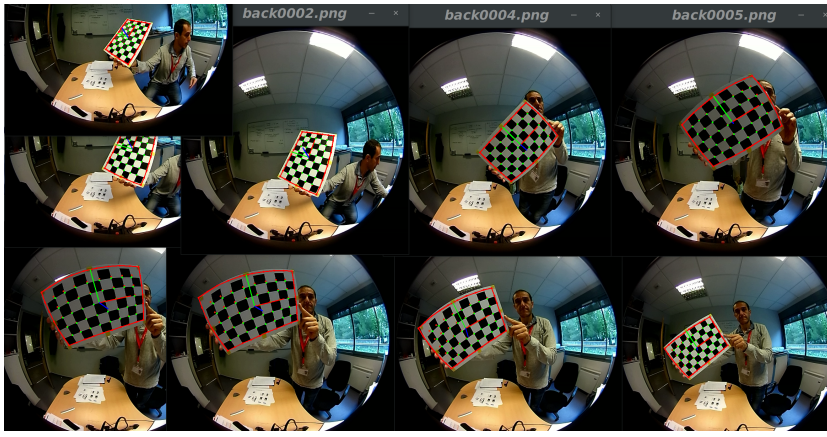


Fig. 3. The visualize results of the calibration of UCM. Green points mean the detected corners, the red points are the 3D points project on the image based on the parameters from calibration, the red boundary is the project plane.

1.2 Results of the Calibration

Results of the UCM Calibration The calibration method is simply based on the gradient descent, which derivative the total projection model G for each parameters to find a direction of changing parameters so that decrease the re-project error. The compute method of the Jacobian matrix and more implement details can be found in the [Mei and Rives, 2007]. The 30 images for calibration test comes from the ricoh camera's back camera. This calibration program develop on C++ and mainly based on the OpenCV and ViSP library. The final visualize results are shown in the Figure 3. The value of the parameters are $f_x = 190.252, f_y = 189.149, u_0 = 318.108, v_0 = 318.069, \alpha = 0.626533$. The distortion parameters are $k_1 = 0, k_2 = 23.715, k_3 = 0.009412, k_4 = 0, k_5 = 1.343577$.

To evaluate the accuracy of these parameters, we can compute the mean reproject error (MRE,here we define the error as euclidean distance between the detect points and the reproject points) as:

$$MRE = \frac{\sum \sqrt{(x - x_d)^2 + (y - y_d)^2}}{N}, \tag{3}$$

where x, y are the project points coordinates, x_d, y_d are the detect points and N is the number of points. The final error is 1.315 pixels, and the standard deviation is 0.5794 pixels. This error is small enough since the we need to compare with the magnitude of pixels number in the general image. From the observation of the reprojection result in the Figure 3, we almost cannot find dislocation by human eye. Both of them prove that under the UCM to do the calibration, we can build the correct relationship of the pixel points and the 3D points.

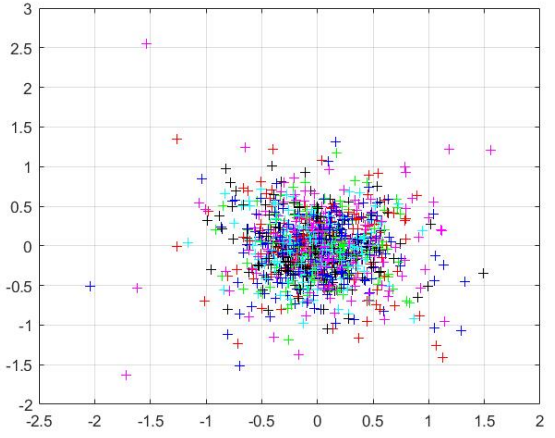


Fig. 4. The reproject error of each point for all the checkerboards. Colors refer to the different images of the checkerboard. The unit of horizontal and vertical axis are pixel. The MRE is 0.473463 pixels and the standard deviation is 0.317803 pixels.

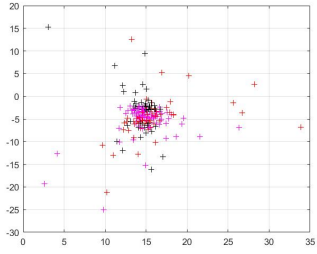
Results of the Extended UCM Calibration The author of this model already provide a nice tool box [Scaramuzza et al., 2006b] on matlab, more calibration details can also be found in this paper. In order to compare the calibration result, we use the same images set as the above model. We finally choose the degree as 5 (the reason will be present in next section), and the coefficients are (-1.909881e+02, 0.000000e+00, 8.439589e-04, 1.525027e-05, -7.854981e-08, 1.511523e-10). And the correspond degree of inverse polynomial is 15 and the coefficients are (286.782225, 147.224129, -45.521772, 18.912762, 38.518309, -35.825602, -20.323798, 55.149712, 11.036111, -43.679341, -7.316963, 19.550107, 4.932180, -3.601535, -1.265170).

To know the project accuracy of this model, we show the reproject error in Figure 4. We can see that most error is less than 1 pixel. Compare with the UCM, this one also show the excellent performance on the projection. Although this model has a more complex calibration procedures, it have a more concise projection model than the UCM. And in the follow image processing we will reuse the projection model many times while the calibration only once. This model will reduce many computation resource, so we will choose this projection model for the follow image processing.

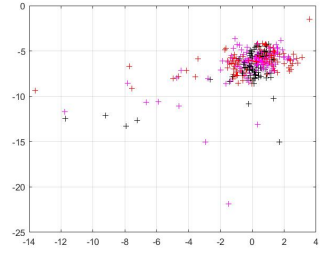
1.3 Affecting Factors for Calibration

Since we will choose the extended projection model of [Scaramuzza, 2014] and the more accuracy parameters are always better, so we need to analyze the important affecting factors of the calibration. In the calibration of this model, actually we can only control the number of calibration images and the degree of

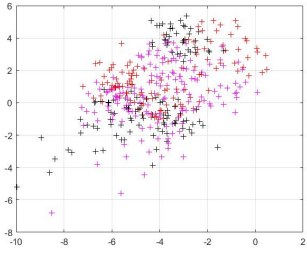
the polynomial. To test the influence of number of images, we randomly choose $1, 5, 10, 15, 20, 25$ images from the same camera to do the calibration. Recording the omnidirectional camera parameters of them. And then randomly choose 10 images to do the calibration again, but replace the omnidirectional camera parameters with the above ones. Finally we can compute the reproject error and show in Figure 5.



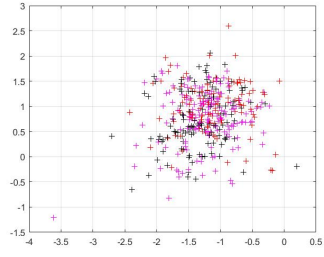
(a) Calibrate images: 1



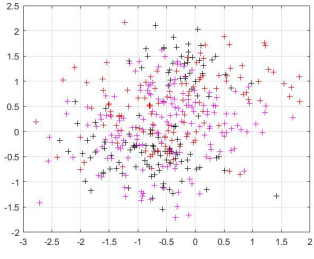
(b) Calibrate images: 5



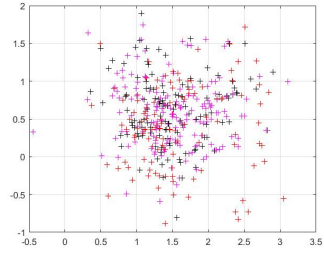
(c) Calibrate images: 10



(d) Calibrate images: 15



(e) Calibrate images: 20



(f) Calibrate images: 25

Fig. 5. The reproject error with different number of calibrate images. All the models' degree of polynomial are 5. From (a) to (f) are the reproject error which calibrate with 1,5,10,15,20,25 images. When number of images equal to 20, calibration has the best performance.

We can see that in the (a) and (b) of Figure 5 which contain less points than the others. Because some of the reproject points already outside of the image, so computing error for these points become meaningless. And for the other situations we compute the MRE and the standard deviation which shown in Table 1.

Table 1. Error analyze for calibration with different number of images. The data is shown in the format as MRE±standard deviation

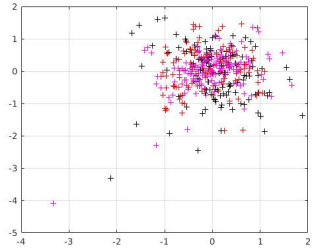
Number of images	Mean reproject error (MRE)
10	4.930966 ± 1.668428
15	1.582495 ± 0.443723
20	1.163253 ± 0.539307
25	1.697083 ± 0.520237

From the figure and table we can find that when number of images increase the errors obviously decreased. Although when images become 25 increased error, it is because the extrinsic parameters calibrate under 10 images, which maybe not accuracy enough. Also the errors decreased speed will become slower as the number of images become big enough. This results shows that we need calibrate with no less than a certain amount of images. However, we also should not calibrate with too much images which not much improve the results.

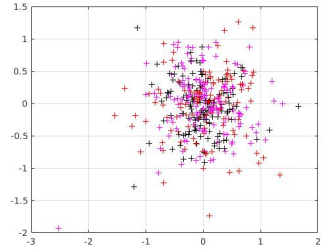
In order to know the importance of the degree of the polynomial, we randomly choose 10 images calibrate with degree of polynomial from 2 to 7. The reproject error is shown in Figure 6. Also we compute the MRE and standard deviation which are shown in Table 2. From the figure we can see that as the degree increase from 2 to 5 the error decreased. But when degree increase from 5 to 7, the error not decrease and even increase a little. From these results, we can know that the degree of polynomial not the higher the better. We need to test with different degree of polynomial to choose the best one.

Table 2. Error analyze for calibration with different degree of polynomial. The data is shown in the format as MRE±standard deviation

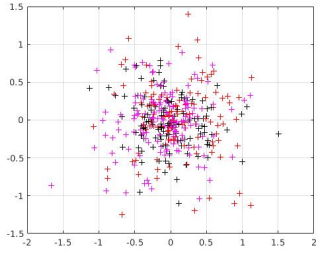
Number of images	Mean reproject error (MRE)
2	0.772387 ± 0.536455
3	0.569220 ± 0.363534
4	0.491975 ± 0.322422
5	0.491978 ± 0.305084
6	0.488928 ± 0.303081
7	0.490742 ± 0.307151



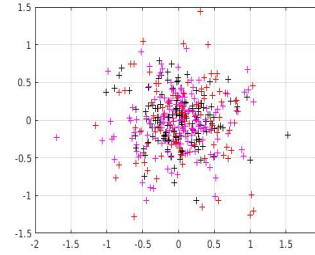
(a) Degree of polynomial: 2



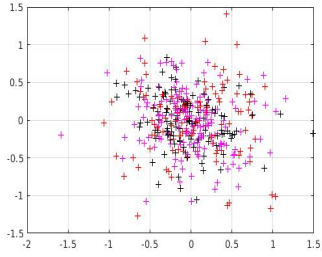
(b) Degree of polynomial: 3



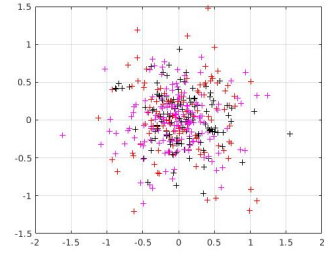
(c) Degree of polynomial: 4



(d) Degree of polynomial: 5



(e) Degree of polynomial: 6



(f) Degree of polynomial: 7

Fig. 6. The reproject error with differnet degree of polynomial. From (a) to (f) the degree of polynomial are: 2,3,4,5,6,7. When degree is 5, the error is minimize.

2 Mapping between Omnidirectional Image and Equirectangular Image

2.1 Introduction

As all the following processing method will require the relationship between the omnidirectional image and unit sphere surface, so we need to build the mapping function between the two presentation ways. And a useful representation of unit sphere surface is the equirectangular plane (in θ, ϕ coordinate). The relationship between these two presentation will be a little different depend on the

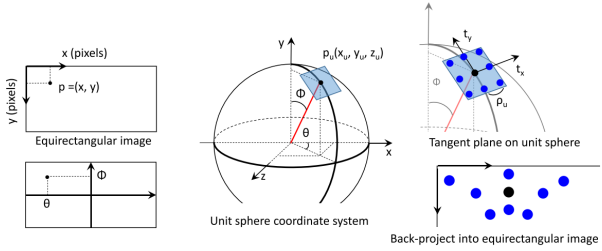


Fig. 7. The coordinate frame relationship [Tateno et al., 2018]. We should be care that the actual θ, ϕ is the complementary angle of the angle show in the image.

definition of the coordinates frame. Generally we will use the follow relationship: $p_u = (cos(\phi)sin(\theta), sin(\phi), cos(\phi)cos(\theta)), \theta \in [0, 2\pi], \phi \in [0, \pi]$. The relationship between the frame is shown in Figure 7.

Actually there are forward wrapping and backward wrapping, as the forward one will lead to some holes in the result, so we choose the backward wrapping.

Mapping function from the omnidirectional image to (θ, ϕ) plane:

- First we do the discretization of (θ, ϕ) to get like $\theta_k = \frac{2k\pi}{N}, \phi_k = \frac{k\pi}{N}$.
- And then we project the (θ_k, ϕ_k) to the unit sphere surface by $p_u = (cos(\phi_k)sin(\theta_k), sin(\phi_k)sin(\theta_k), cos(\theta_k))$.
- Finally we can project the spherical points to omnidirectional image by: $p = K * p_u$, where K is the intrinsic matrix which get from calibration. Or we can use Scaramuzza's method, and the degree of this inverse polynomial is 15.

We can get the correspond coordinates from the above mapping, and then just put the correspond color value to the (θ, ϕ) plane.

Mapping function from (θ, ϕ) plane to the omnidirectional image:

- We project the pixel coordinate (u, v) to the unit sphere surface by Scaramuzza's method with the polynomial of degree 5.
- And then we project the (x, y, z) to the (θ, ϕ) plane by $\phi = \arccos(y), \theta = \arctan(x, z)$.
- We do the discretization to get the pixel coordinates: $\theta_k = (0.5 + \theta/(2\pi)) * width, \phi_k = (0.5 + \phi/\pi) * height$

2.2 Results of Mapping

In order to test our mapping function, we apply it on different omnidirectional image datasets. One is the same fisheye images as the previous section, while the other two catadioptric images comes from [Scaramuzza et al., 2006b] and [Zhang et al., 2016]. The mapping result show in Figure 8. From the image we can see that limited by the field of view, actually when the omnidirectional image can

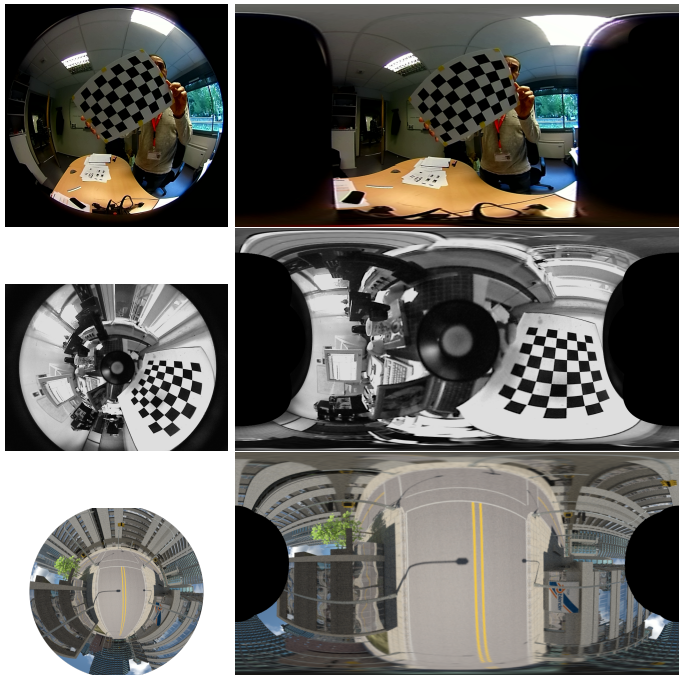


Fig. 8. The (θ, ϕ) plane image project from omnidirectional image. The first col images are the origin omnidirectional images (the first one from fisheye camera, and other two from catadioptric camera). The second col images are represent in equirectangular image.

not entirely cover all the surface of the unit sphere. And we can clear see that catadioptric image have a larger field of view than the fisheye image. So we will use two or even more fisheye images project into equirectangular image and then blend them together to generate a real omnidirectional image.

The back mapping result is shown in Figure 9. From the human eye, the back project images almost the same as the origin images. However because the limitation of human eye, we still need some metrics to compare the the results. Here we will choose several similarity metrics to compute the similarity of the back project images and the original images. In order to know the sensitive of the different metrics, we also use the incorrect parameters to generate some images, which shown in Figure 10.

The most common metrics are mean square error (MSE), structural similarity index metric (SSIM) and histogram. MSE is an index value used to calculate the similarity of two matrices. The smaller the value is, the more similar the two matrices are. It require the two matrices have the same shape. Its calculation

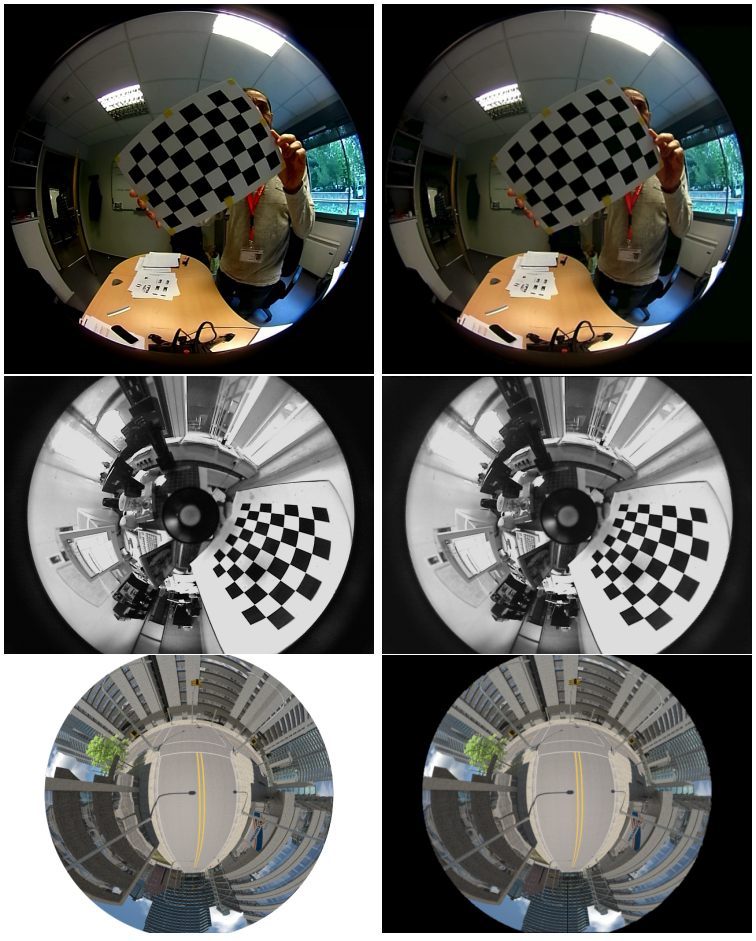


Fig. 9. The first col images are the origin images. The second col images are the back mapping image from the generated equirectangular image.

formula is (for the images version):

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)]^2, \tag{4}$$

where m, n are the rows and cols of the image, and I, K are the two images.

SSIM can better reflect the similarity of two pictures. The range of this indicator is $[-1, 1]$. When $SSIM = -1$, it means that the two pictures are completely dissimilar. When $SSIM = 1$, it means two pictures. very similar. That is the value the closer to 1, the more similar the two pictures are. The

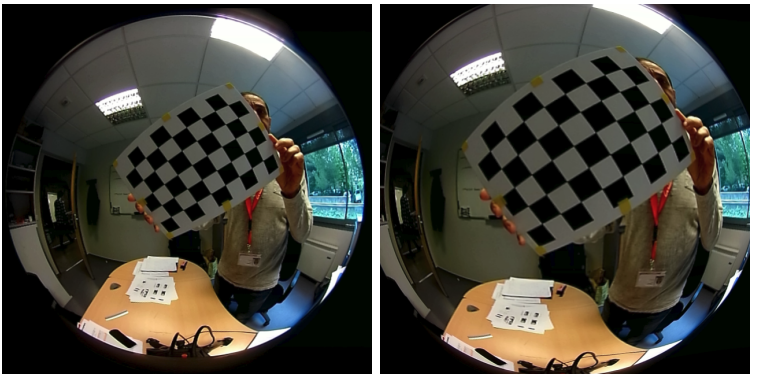


Fig. 10. The back project image with wrong parameters. The left one is slightly changed and right one is huge changed.

calculation formula for this indicator is as follows:

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{({\mu_x}^2 + {\mu_y}^2 + c_1)({\sigma_x}^2 + {\sigma_y}^2) + c_2}, \tag{5}$$

where x, y are the two compare matrices, μ_x, μ_y are the average x, y, ${\sigma_x}^2, {\sigma_y}^2$ are the covariance of x, y, c_1, c_2 are two variables to stabilize the division with weak denominator.

The main idea of image Hash is that each image is reduced down to a small hash code or 'fingerprint' by identifying salient features in the original image file and hashing a compact representation of those features (rather than hashing the image data directly). Finally, count the number of bit positions that are different (Hamming distance). The value closer to 1 means more similarity. There are 3 types of the Hash. Average Hash (aHash), for each of the pixels output 1 if the pixel is bigger or equal to the average and 0 otherwise. Perceptive Hash (pHash), does the same as aHash, but first it does a Discrete Cosine Transformation and works in the frequency domain. Derivative Hash (dHash), calculate the difference for each of the pixel and compares the difference with the average differences. As speed goes, dHash is as fast as aHash and very few false positives. Here we will choose the dHash to test.

The results of computing the similarity with different metrics are shown in Table 3. From the results of that we can find that MSE is very sensitive of the change of image. Even a slightly change will make it increase a lot. As for the SSIM and dHash can still recognize the similar image with slightly changed, which means these two metrics can perform better on the similarity of the whole structure.

Histogram which counted the number of pixels with different value can present the distribution of the images. The result is shown in Figure 11. Although the histogram are not completely the same, but they have the same trend. The histogram similarities from 0 to 1, the bigger means the more similar. But the

Table 3. Different similarity metrics apply on the images in Figure 9

Back project images	MSE	SSIM	dHash
Fisheye	333.35	0.79	0.921875
Catadioptric 1	503.1	0.79	0.875
Catadioptric 2	144.47	0.82	0.90234375
Slightly incorrect fisheye	1426.46	0.63	0.859375
Huge incorrect fisheye	5109.6	0.40	0.6484375

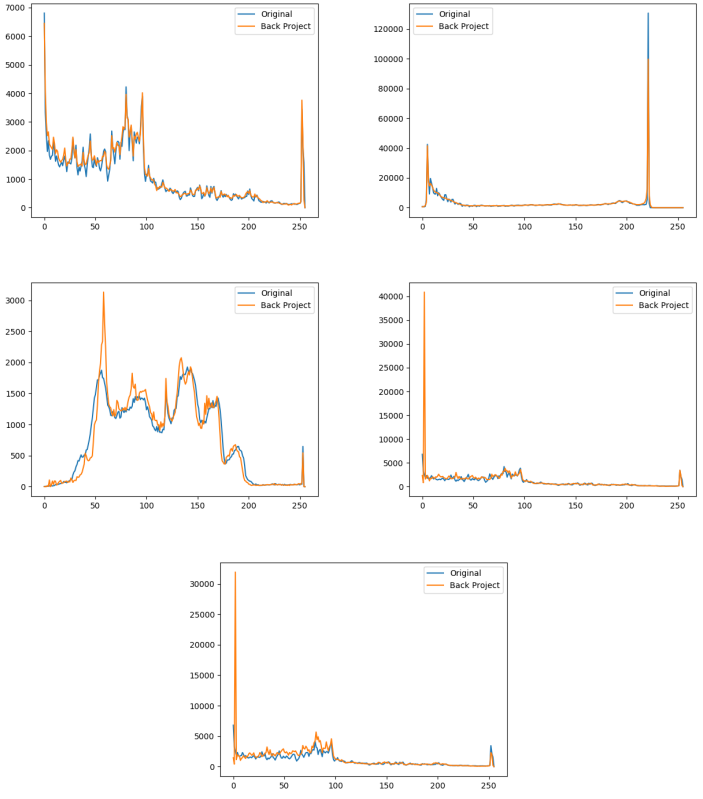


Fig. 11. The histogram of original image and the back project one. From the top to the bottom correspond to the same order images in the Figure 9 and Figure 10. And the similarities of the histogram from top to bottom and from left to right are 0.9849671467753949, 0.9874105979154342 0.9571264770654119, 0.3923028916709083 and 0.5030881661408604.

results also shows that histogram cannot well recognize the change of image

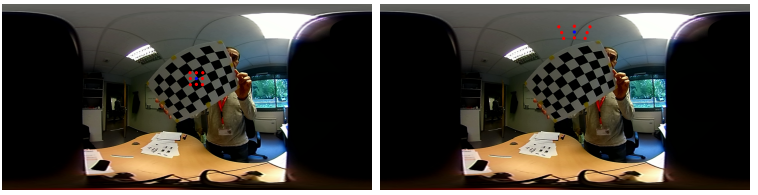


Fig. 12. The new neighbors (red point) of the point (blue point) in the equitangular plane.

as the larger incorrect image with higher similarity in histogram. So we should combine with other metric to compute the similarity.

Combine the results of MSE, SSIM, dHash and histogram we can see that, although the back project image seems the same as the original one from the human eye, actually the information will be changed or even lost after a complex transform. However, these results also show that the back project images still keep the main information of the original images. So we can say that we successfully build the mapping function for processing the omnidirectional image.

2.3 Redefining Pixel Neighboring

Because the spatial distribution of omnidirectional image is variance. Which means the pixel will have different physical influence on its neighbor depend on the position. However, the traditional perspective image processing method did not consider this situation. In order to develop the adaptive processing method for omnidirectional image, we want to redefining pixel neighboring to find the same physical influence neighbours. One method is that use the neighbors on the unit sphere surface which considered as spatial invariance, and then project it back to the equirectangular image. Base on the method from [Tateno et al., 2018], we try to implement the new neighbors of pixels. The mapping procedure is mentioned above. Here we just show how to choose the neighbor in the unit sphere surface with a tangent plane:

- Generate the basic vectors of the tangent plane:

$$t_x = |v \times p_u|, t_y = |p_u \times t_x|,$$

  where p_u is the center spherical coordinates, and $v = (0, 1, 0)$.
- And then just to get the neighbours' coordinates by the linear combination the basic vectors.

Finally we can use the back mapping function describe in the previous section to project the neighbours into the equirectangular plane. The neighbours project in the equitangular plane is shown in the Figure 12. Limited by the space, we can only show several examples of the new neighbours. We can see that the neighbours' position will change as the center changed. When it close to the

$\phi = 0, \pi$, the neighbours will become more dispersion, while close the center the neighbours become compact. Because $\phi = 0, \pi$ are the poles of the sphere, where contain less information, and the neighbours influence is also weaker than the middle one.

Bibliography

Geyer, C. and Daniilidis, K. (2000). A unifying theory for central panoramic systems and practical implications. In *European conference on computer vision*, pages 445–461. Springer.

Mei, C. and Rives, P. (2007). Single view point omnidirectional camera calibration from planar grids. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3945–3950. IEEE.

Scaramuzza, D. (2014). Omnidirectional camera. *Computer Vision: A Reference Guide*, pages 552–560.

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006a). A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 45–45. IEEE.

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006b). A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE.

Tateno, K., Navab, N., and Tombari, F. (2018). Distortion-aware convolutional filters for dense prediction in panoramic images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 707–722.

Zhang, Z., Rebecq, H., Forster, C., and Scaramuzza, D. (2016). Benefit of large field-of-view cameras for visual odometry. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 801–808. IEEE.