International Journal of Computational Intelligence Systems
Vol. **14(1)**, 2021, *pp.* 238–247
DOI: https://doi.org/10.2991/ijcis.d.201126.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

A Simulation Method of Specific Fish-Eye Imaging System Based on Image Postprocessing

Wenhui Li, You Qu*, Ying Wang, Jialun Liu

College of Computer Science and Technology, Jilin University, Changchun, 130012, China

ARTICLE INFO

Article History
Received 07 July 2020
Accepted 15 Nov 2020

Keywords
Sensor simulation
Intelligent vehicle system
Fish-eye camera
Camera calibration
Stereographic projection model

ABSTRACT

In order to enable the implementation of the computer vision-based perception techniques in the physical-based simulation environment, visual sensors need to be simulated physically. Among others, fish-eye cameras are commonly used visual sensors to provide an omni-directional field of view. The existing methods cannot simulate the output of a specific real fish-eye imaging system. In this paper, we present a postprocessing fish-eye imaging system simulation method. According to the stereographic projection model, a mapping is established between the ideal fish-eye image and a cube-map rendered by the graphic engine. The distortion of the real camera is measured and added to the ideal image, in order to simulate the output of a specific camera. Experimental result shows that our simulation method can give output close enough to the image captured by a specific fish-eye camera. The practicality of our method is validated in an actual application of vehicle omniview system.

1. INTRODUCTION

Over the years, physical-based simulation tools are widely used in many industrial fields, such as intelligent vehicle [1] and robotics [2]. In the early stages of development of intelligent vehicle systems, making prototype experiments in a virtual environment can help reducing the experiment cost, protecting the safety of persons and devices, and avoiding the legal and moral risk. There are several intelligent vehicle-dedicated simulation platforms on the market. Pro-SiVIC of CIVITEC is a sensor simulator. It can simulate various complex scenarios like roundabouts, multiple vehicles and pedestrians, and different weather conditions. Some applications [3] use Pro-SiVIC simulator to test cooperative algorithms on intelligent vehicles. CarSim [4–6] of the Mechanical Simulation Company, providing dynamics vehicle models, is designed to validate driving assistance algorithms. PreScan [7] of Tass International is also a simulator that is specialized on sensor modeling. It has a full-spectrum camera simulation for reliable virtual development and validation of automated driving applications. In the industrial production, off-line programming of the industrial robots in simulation software can reduce the frequency of stoppage of the production line, so as to reduce the impact on production efficiency. While almost every robot manufacturer offers a brand-specific simulation tool, there are also many third-party simulators like CoppeliaSim [8], Robotmaster [9], Delfoi [10], etc. In the abovementioned intelligent or automatic systems, various sensors are mounted to provide information about the surrounding situations. Among others, visual sensors are the most commonly used ones [11]. The functions of these sensors need to be reproduced physically in the simulation tools.

Nowadays, in many applications such as intelligent surveillance [12,13] and autonomous navigation [14,15], an omni-directional field of view (FOV) is often required in order to see more in one shot. Compared with other omni-directional imaging systems, such as multi-camera devices [16,17] and catadioptric cameras [18,19], fish-eye cameras, which combine fish-eye lenses and conventional cameras, are cheaper, smaller, and easier to set up, and thus are becoming increasingly popular in the computer vision community.

There are few simulation methods dedicated to the fish-eye cameras. The Pro-SiVIC platform chooses two techniques to simulate omni-directional cameras. The first one generates fish-eye images by merging 6 images rendered by the OpenGL library, which takes no account of the imaging geometry of real cameras. The second one uses a mesh to form the desired shapes and reflect the environment on it. It also cannot be used to generate the output of a specific fish-eye camera. Ref. [20] presented a three-dimensional simulation method for fish-eye lens distortion in a vehicle rear-view camera. This method can only generate the image of straight lines, and needs the field number of the target fish-eye lens, which is not always available. In summary, for the following reasons, it is necessary to propose a new simulation method for fish-eye cameras:

1. Physical-based simulation technology is widely used in the field of intelligent system;
2. Fish-eye camera is widely used in computer vision technology, the latter is one of the main technologies in intelligent systems;

* *Corresponding author. Email: quyou12@mails.jlu.edu.cn*

3. So, it is necessary to provide a method to simulate a specific fish-eye camera in the physical-based simulation environment;
4. However, there are few existing methods to simulate the output of a specific fish-eye camera.

In this paper, we proposed a postprocessing method to simulate the output of a specific fish-eye imaging system. The graphic engine is used to generate a cube-map centered on the principal point of the fish-eye camera, and then the relationship between the pixels on the simulated fish-eye image and the cube-map is established according to the fish-eye imaging model, i.e., the stereographic projection model. Then the simulated fish-eye image is generated. The essence of this method is to transform the image pre-generated by the engine, which is called "postprocessing method." To make the simulated image closer to the real output of a specific device, the deviation from the ideal projection model to the imaging geometry of a real camera is estimated quantificationally and applied to the fish-eye image. Experimental result shows that our simulation method can give output close enough to the image captured by a specific fish-eye camera. The practicality of our method is validated in an actual application of vehicle omniview system.

This paper is organized as follows: Section 2 introduces the projection models used to describe a fish-eye imaging system. The proposed simulation method is detailed in Section 3. After providing experimental results and validations in Section 4, Section 5 concludes the paper.

2. PRELIMINARIES

2.1. Projection Models

A projection model is a mapping from the 3D object point $P(X, Y, Z)$ to its corresponding 2D image point $p(x, y)$. A pinhole camera is always described by the perspective projection model. If we denote the focal length of the camera as f, the perspective projection could be formulated as

$$\begin{bmatrix} x \\ y \end{bmatrix} = -\frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix} \tag{1}$$

In addition to the projection model, imaging system design usually involves many other considerations, e.g., size, cost, and uniform illumination. Assembly defects are also introduced during camera manufacture. These will make the actual output of cameras deviates from the ideal projection model to varying degrees. This deviation is often treated as distortion. In classical camera calibration methods, the most commonly used distortion model is the Brown–Conrady polynomial model [21]:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x_u \\ y_u \end{bmatrix}\left(1 + k_1 r^2 + k_2 r^4 + \cdots + k_n r^{2n}\right). \tag{2}$$

Here, $[x_d, y_d]^T$ is the actual position of the image point, $[x_u, y_u]^T$is the position of the corresponding point without distortion, and r denotes the distance from $[x_u, y_u]^T$ to the distortion center $c = [x_c, y_c]^T$, which is also the principle point here. Further, k_n is the nth radial distortion coefficient. When the distortion is small, usually two coefficients k_1 and k_2 are sufficient. We can say that the distortion of a camera is a mapping from $[x_u, y_u]^T$ to $[x_d, y_d]^T$ and can be characterized by a set of parameters $\lambda = \{x_c, y_c, k_1, k_2\}$.

The FOV of a pinhole camera is restricted by the size of the sensor. To take the entire scene of a 180° FOV into a limited sensor, fish-eye lenses are designed to severely bend the incident ray coming from an ultra-wide angle and therefore cannot be described properly by the linear model given by (1). A two-stage procedure using a sphere (say, a viewing sphere), whose center coincides with the optical center of the camera and whose radius is equal to the camera's focal length, is introduced to describe a nonlinear projection model [22]. The procedure is demonstrated in Figure 1.

First, a 3D point P is projected perspectively through the center O onto point P_s on the sphere. Then P_s is projected to a 2D point p on the image plane π from another point O' on the optical axis ON. The position of O' along ON defines a series of projection models. Here O' coincides with N in Figure 1. A projection model can be clearly represented as a mapping $r = \rho(\alpha)$, where α is the incident angle of ray emitted from P, and r is the radial distance as mentioned. The representative models below have been widely discussed:

$$\begin{aligned} \rho(\alpha) &= \tan\alpha \quad (\textit{perspective projection}), \\ \rho(\alpha) &= 2f\tan\frac{\alpha}{2} \quad (\textit{stereographic projection}), \\ \rho(\alpha) &= f\alpha \quad (\textit{equidistance projection}), \\ \rho(\alpha) &= f\sin\alpha \quad (\textit{sine-law projection}), \\ \rho(\alpha) &= f\sin\frac{\alpha}{2} \quad (\textit{perspective projection}). \end{aligned} \tag{3}$$

Note that the perspective projection can also be described by this procedure. In Ref. [23], the author prefers to stereographic projection rather than other alternative ones because it can represent a wider FOV while preserving a suitable range of shape properties of the projected objects, including the circularity, the local symmetry, the shape of small regions, and the angle at which two curves intersect. These properties are very useful in applications such as intelligent surveillance and automatic navigation. Thus, in this paper, the stereographic projection model is chosen for our estimation method. It is exactly the model shown in Figure 1.

2.2. Geometric Invariants Under Stereographic Projection

Simple geometric targets, such as straight lines or spheres, are often utilized in the distortion estimation methods because the shape of

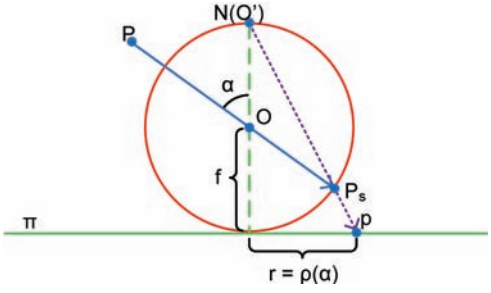

Figure 1 Two-stage procedure for describing a projection model. A 3D point is first projected onto the viewing sphere, followed by another perspective projection from some point on the optical axis onto the image plane.

the images of these targets is known under a certain projection model. For stereographic projection, this is shown in Figure 2.

When projected onto the viewing sphere from its center O, the image of a space line L is a great circle determined by the sphere surface and the plane OL. On the other hand, the image of a sphere S on the surface of the viewing sphere is a small circle. Let (n_x, n_y, n_z, d_0) be the plane containing the small circle. Here, (n_x, n_y, n_z) is the normal vector of the plane, and d_0 is the distance from O to the plane. Obviously, the great circle is actually a special case of the small circle in which d_0 is zero. When projected from N onto the image plane π, the function of a point on the image of S can be derived as

$$x^2 + y^2 - \frac{4f^2}{fn_z - d_0}(n_x x + n_y y + fn_z + d_0) = 0. \tag{4}$$

We set $d_0 = 0$ to obtain the formula of the image of L:

$$x^2 + y^2 - \frac{4f^2}{fn_z}(n_x x + n_y y + fn_z) = 0. \tag{5}$$

Notice that both (4) and (5) are the general equations of a circle. So the conclusion is that the image of a space line section under the stereographic projection is a circular arc (a portion of a circle), and the image of a sphere is a circle. These are the main geometric invariants used in our estimation method.

3. SIMULATION METHOD DETAILS

In this section, we propose a fish-eye image generation method based on the engine-rendered image postprocessing. First, an ideal fish-eye image is generated. To get the value of a particular pixel in the output image, the corresponding incident ray is back-traced into the virtual environment according to the projection model. To simplify the implementation, a cube-map centering at the center of the viewing sphere is rendered with the help of a 3D graphic engine. Then, the distortion parameters of the target camera are estimated and used to distort the ideal image in order to connect the simulation to the specific imaging system.

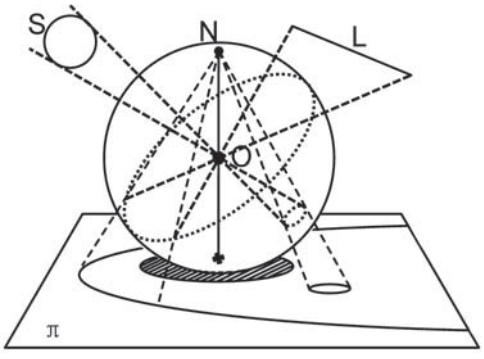

Figure 2 | Images of a line and a sphere under the stereographic projection. The projection of a space line on the viewing sphere, as well as the projection of a sphere (dotted line), is a circle. Circles on the viewing sphere maintain their shape when projected onto the image plane from the northern pole of the viewing sphere (solid line on the image plane).

3.1. Generate the Ideal Fish-Eye Image

The common method to generate images of a virtual scene is backward ray-tracing. For fish-eye camera simulation, it is the inverse procedure of the projection model described in last section. For a given 2D point p on π, a direction vector starting from O is found. A ray in this direction is traced until it hits a surface in the scene at a point P. The value of p can be computed according to the properties of P (material, color, orientation, lighting, etc.). Let the polar coordinate of point p is denoted by (r, ϕ), where r is the radial distance, and ϕ is determined by

$$\phi = \arctan\frac{y_p}{x_p}. \tag{6}$$

From the stereographic projection function given in (3), the incident angle α of the light ray corresponding to p is

$$\alpha = 2 * \arctan\frac{r}{2f} \tag{7}$$

where f is the focal length of the fish-eye camera. Then we have the direction vector we want:

$$\overrightarrow{OP} = (sin(\alpha) * cos(\phi), sin(\alpha) * sin(\phi), cos(\alpha)). \tag{8}$$

Actually, in order to achieve a more realistic rendering effect, the tracing may not stop at P. It will go between objects in the scene until it reaches a light source. This is a tedious work, and difficult to implement. So we let the 3D engine to do this job. This is a so called postprocessing method, because it is based on the prerendered images.

We define 6 identical pinhole cameras in the virtual scene, all positioned at the center of the viewing sphere. The cameras are arranged orthogonally. If the orientation of the fish-eye camera is (0, 0, 1), then the orientations of the 6 pinhole cameras are (0, 0, 1), (0, 0, −1), (0, 1, 0), (0, −1, 0), (1, 0, 0), (−1, 0, 0), respectively. The cameras all have an FOV of 90 degrees in both horizontal and vertical directions. The deployment of the cameras is shown in Figure 3.

In this way, the images rendered through these six cameras form a cube-map. In our simulation, only five cameras are sufficient. The camera looking backward is omitted because we just need an FOV of 180 degrees in front of our fish-eye camera. $\overrightarrow{OP}$ is transformed into the coordinate frames of the five cameras to see if it hits one of the image planes at a point $m(x_m, y_m)$, as shown in Figure 4.

This creates a mapping between the simulated fish-eye camera and the five cameras. This mapping doesn't change from frame to frame in real-time rendering, thus can be stored in a mapping table. Each camera is assigned an ID, and for a pixel in the output image, its corresponding entry in the lookup table is like (ID, x_m, y_m). The pixel value can then be obtained from the position in the pointed image. The rendered outputs of the five cameras and the simulated ideal fish-eye image is given in Figure 5.

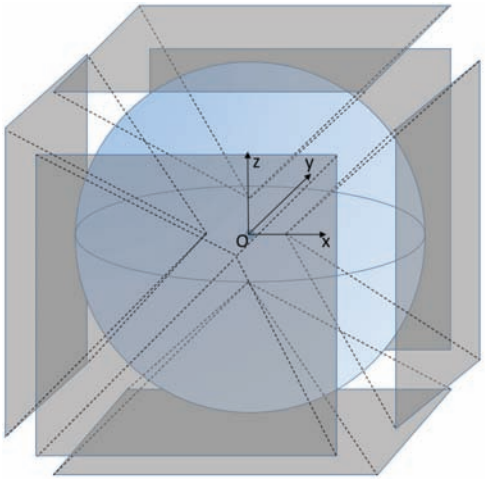

Figure 3 The viewing sphere and the deployment of the six pinhole cameras. The cameras are moved from their original position (i.e., the center of the viewing sphere) along the axis to make the figure clearer.

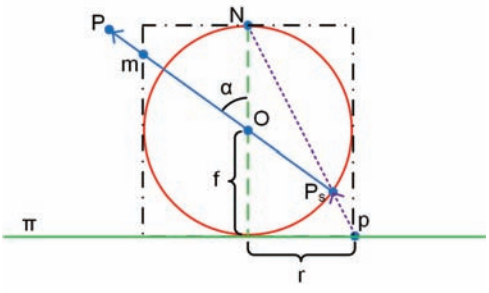

Figure 4 Mapping from the fish-eye image plane to the five orthogonal cameras. The incident light ray is back-traced and hits one of the faces of the cube-map at a point .

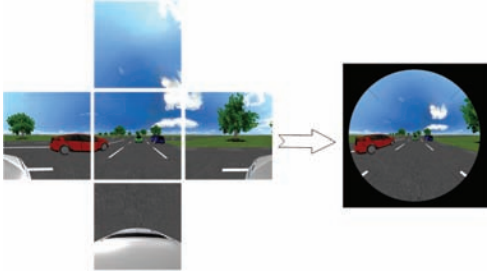

Figure 5 The rendered outputs of the five cameras and the simulated ideal fish-eye image.

3.2. Image Distortion

Figure 6(a) shows the ideal image of several spheres generated by the method described in last section. According to the property of the stereographic projection, the image of a sphere is a circle. However, it is not the case in a real fish-eye image as shown in Figure 6(b).

In this section, we detail our distortion estimation method. The estimation of the distortion center and the distortion coefficients

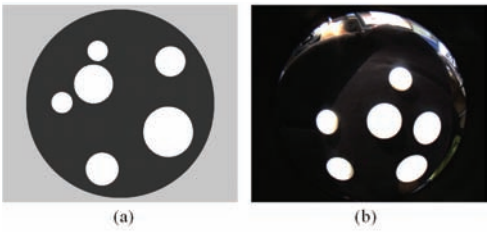

Figure 6 (a) The simulated ideal fish-eye image of several spheres. (b) Image of spheres captured by a real fish-eye camera.

are performed in a hierarchically iterative manner because it has been shown that including the distortion center in the nonlinear optimization may make the system unstable with the presence of noise [21]. In the outer loop, feature points extracted from images of parallel lines are undistorted with the latest estimated distortion parameters and then used to update the distortion center. If the new center is close enough to the current one, the algorithm terminates. Otherwise, in the inner loop, with the latest distortion center, a genetic algorithm (GA) module is adopted to optimize the distortion coefficients based on the geometric invariants discussed in Section 2.2.

3.2.1. Estimation of the distortion center

We estimate the distortion center with the help of the vanishing points. A vanishing point is the intersection of a group of parallel straight lines under the stereographic projection model. A straight line in space can be given as a point direction form equation as

$$L(t) = qt + b, \tag{9}$$

where $q = [q_X, q_Y, q_Z]^T$ is the direction vector, and $b = [b_X, b_Y, b_Z]^T$ is a fixed point on the line. Further, t is a parameter indicating any given point on the line. By combining (1) and (9), we have the projection of a straight line on the perspective image plane:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = -\frac{f}{q_Z t + b_Z} \begin{bmatrix} q_X t + b_X \\ q_Y t + b_Y \end{bmatrix}. \tag{10}$$

Let $r(t)$ denotes the radial distance from $[x(t), y(t)]^T$ to the distortion center. If we assume that the distortion center coincides with the origin $[0, 0]^T$, then we have $r(t) = \sqrt{x^2(t) + y^2(t)}$.

From the perspective and stereographic projection functions in (3), the radial distance of the corresponding point $[x'(t), y'(t)]^T$ under the stereographic projection is

$$r'(t) = 2f \tan \frac{\arctan \frac{r(t)}{f}}{2}. \tag{11}$$

Rearranging gives

$$r'(t) = 2f \frac{\sqrt{f^2 + r^2(t)} - f}{r(t)}. \tag{12}$$

The projection of a given 3D point under different projection models always moves along the radial direction. Therefore, the projected 2D points are related by their radial distance:

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{r'(t)}{r(t)} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}. \tag{13}$$

Inserting (10) and (12) into (13) gives

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{1-D}{(q_X t + b_X)^2 + (q_Y t + b_Y)^2} \cdot 2f \cdot \begin{bmatrix} q_X t + b_X \\ q_Y t + b_Y \end{bmatrix}. \tag{14}$$

where $D = \sqrt{(q_X t + b_X)^2 + (q_Y t + b_Y)^2 + (q_Z t + b_Z)^2}$.

The limit of (14) as $|t| \to \infty$ then gives two vanishing points, v_1 and v_2, of a group of parallel lines projected onto the stereographic image plane, i.e.,

$$\begin{bmatrix} x_v \\ y_v \end{bmatrix} = \lim_{t \to \pm\infty} \begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{1 \mp \sqrt{q_X^2 + q_Y^2 + q_Z^2}}{q_X^2 + q_Y^2} \cdot 2f \cdot \begin{bmatrix} q_X \\ q_Y \end{bmatrix}, \tag{15}$$

If we consider the two points as two vectors from the distortion center, examining the angle ϕ between these two vectors gives

$$\phi = \arccos \frac{v_1 \cdot v_2}{|v_1||v_2|} = \pi, \tag{16}$$

which means that the two vanishing points lie in opposite directions from the distortion center. Thus, the line joining v_1 and v_2 must pass through the distortion center. The position of the distortion center is determined as follows:

i. **Extract points.** An edge detection algorithm such as Sobel or Canny is used to extract the points on the image of lines. As shown in Figure 7(a), the extracted points are grouped into several sets:

$$P_L = \left\{ \left\{ \bar{l}_j^i = (x_j^i, y_j^i) \right\}_{j=1}^{m_i} \right\}_{i=1}^{n},$$

where for each $i = 1, \ldots, n$ the set $\{\bar{l}_j^i\}_{j=1}^{m_i}$ is a collection of extracted points belonging to the image of the ith space line.

ii. **Undistort the points and do circle fitting.** P_L is undistorted according to (2) with the currently available distortion parameters, if they exist. Circles are fitted to each set of the resulting points $\bar{P}_L$ in a least square manner. Let the circle fitted to the point set $\{\bar{l}_j^i\}_{j=1}^{m_i}$ be C^i. Ideally, all $\{C^i\}_{i=1}^n$ will converge exactly at a pair of vanishing points. However, this is not the case because of the error introduced by the distortion and measurement noise, as shown in Figure 7(b). Thus, the estimation of the vanishing points needs a nonlinear optimization.

iii. **Initialize the vanishing points.** An initial guess is required for the optimization. The intersections of every two circles in $\{C^i\}_{i=1}^n$ are calculated to be the candidates of initial vanishing points. All these candidates are examined to pick the best one. To do this, we do circle fitting again to $\bar{P}_L$. For a pair of candidate vanishing points $(\hat{v}_1, \hat{v}_2)$ that is being examined, the circle $\hat{C}^i$ fitted to the point set $\{\bar{l}_j^i\}_{j=1}^{m_i}$ is forced to pass through both of these two points. $\hat{C}^i$ is then adjusted to minimize the error function:

$$\epsilon_i = \sum_{j=1}^{m_i} \left(\left(\bar{x}_j^i - x_0^j \right)^2 + \left(\bar{y}_j^i - y_0^j \right)^2 - R^{i2} \right), \tag{17}$$

where $[x_i^0, y_i^0]^T$ and R^i are the center and radius of $\hat{C}^i$ respectively. Summing the minimized fitting errors calculated from every point set in $\bar{P}_L$ gives

$$E = \frac{1}{\sum_{i=1}^n \epsilon_i}. \tag{18}$$

E is taken as the score of the pair $(\hat{v}_1, \hat{v}_2)$. The pair with the highest score is chosen as the initial position of the vanishing points.

iv. **Optimize the vanishing points.** The initial positions of v_1 and v_2 are then refined with the Levenberg–Marquardt algorithm. Eq. (18) is used again as the objective function. To ensure the stable convergence of the optimization, the two vanishing points are refined alternately. When v_1 is being refined, the position of v_2 is fixed. Once v_1 converges, v_1 is fixed and v_2 is refined. This repeats until both v_1 and v_2 converge. Figure 7(c) shows the final position of v_1 and v_2.

v. **Estimate the distortion center.** Several other pairs of vanishing points are determined from other pictures of the parallel lines in the same way. The lines joining each pair of vanishing points intersect with each other, and the centroid of the intersections is located as the distortion center, as shown in Figure 7(d).

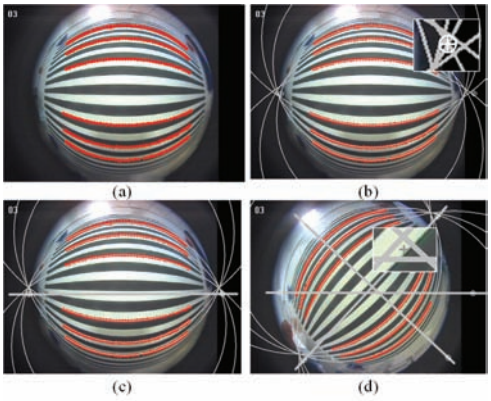

Figure 7 Estimation of the distortion center: (a) Points on the images of lines are extracted and grouped into several sets. (b) The fitting circles have many pairs of intersections, and the one with the lowest score is chosen as the initial position of the vanishing points. (c) The initial points are optimized and then linked. (d) At least three pairs of vanishing points need to be estimated to determine the distortion center.

3.2.2. Estimation of the distortion coefficients

In this subsection, the geometric invariants discussed in Section 2.2 are used to estimate the distortion coefficients of the fish-eye cameras. We need to define an object function by which, when minimized, the distortion parameters are optimized to put the points extracted from the image of the geometric targets back onto given shapes. Our object function is defined via the images of spheres as well as straight lines. As we will show later in Section 4, spheres are more appropriate to serve as the geometric target for the distortion estimation of fish-eye cameras, especially with the presence of measurement noise.

The first two steps are similar to the ones described in last subsection. Pictures of the target objects (lines in space or spheres) are captured, and edge points P_{tar} are extracted. P_{tar} is then undistorted with a set of distortion parameters $\hat{\lambda} = \{\hat{x}_c, \hat{y}_c, \hat{k}_1, \hat{k}_2\}$, and circles are fitted to the resulting points $\overline{P}_{tar}$.

Measurement noise in the extracted points will be amplified nonlinearly due to the distortion if the objective function is defined in the space of the undistorted points [24]. Thus, in this paper, we define the object function in the space of distorted image points to ensure the robustness of the estimation method with the presence of noise. As shown in Figure 8, p_j^i is one of the points in P_{tar}, $\overline{p}_j^i$ is the corresponding point in $\overline{P}_{tar}$, and C^i is the circle fitted to $\{\overline{p}_j^i\}_{j=1}^{m_i}$. We search for a point $\hat{p}_j^i$ near p_j^i which, after being undistorted with $\hat{\lambda}$, will lie exactly on C^i at $\hat{p}_j^i$. Then the object function is defined as follows:

$$e = \sum_{i=1}^{n}\sum_{j=1}^{m_i} \| p_j^i - \hat{p}_j^i \|^2 . \tag{19}$$

The GA is used here to minimize (19). A simple GA using single-point crossover and single-point mutation is sufficient for our problem. Any other nonlinear optimization algorithm could also be used here. Once the distortion parameters are determined, the distorted position of the pixels in the ideal image can be found, and the mapping relationships between the output image and the cube-map recorded in the mapping table need to be adjusted accordingly.

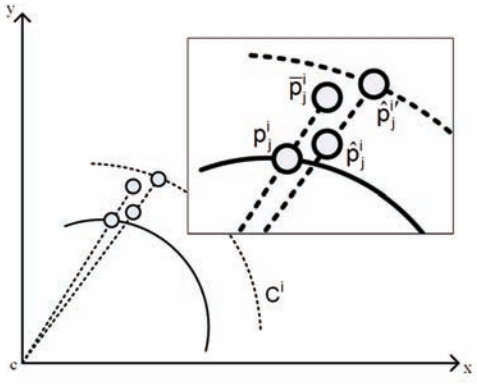

Figure 8 Setup for determining objective function..

4. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, three sets of experimental results are presented to show the robustness of our estimation method. In the first experiment, all image points are artificially generated, thus the distortion-free position of the image points is known. In the second experiment, a 3D engine is employed to construct a virtual scene containing the geometric targets. The method is applied to the images rendered by the 3D engine. The procedures that may introduce noise, including image acquisition and point extraction, are involved. This makes the experiment closer to the real situation. We then apply our method to the real images, and give a quantitative evaluation of our simulation method. Finally, an actual application of vehicle omniview system is implemented to show the practicality of our method.

The GA module searches the distortion coefficients in the intervals $k_1 \in [-1E^{-5}, 1E^{-5}]$ and $k_2 \in [-1E^{-12}, 1E^{-12}]$. This is large enough to cover the distortion of fish-eye cameras with respect to the stereographic projection model. The precisions of k_1 and k_2 are approximately $3E^{-10}$ and $3E^{-17}$, respectively. The size of the population is 50, and the initial population is generated randomly. The roulette-wheel selection strategy is adopted. The incidence probabilities of crossover and mutation are 0.75 and 0.06, respectively.

4.1. Experiment on Fully Synthetic Points

In this experiment, we show how the number of the target objects affects the estimation result with the presence of noise of different levels. The effectiveness of lines and spheres as the target is also compared. A fish-eye camera using an 800*800 CCD sensor is considered. With a focal length of 160 pixels, the whole FOV of 180° can be fitted within the sensor under the stereographic projection. The ideal distrotion-free image of parallel space lines and spheres are shown in Figure 9(a). We assume that all the lines in space are parallel to the image plane, which will not affect the objectiveness of the result, so the vanishing points are at the angular position of 90°, i.e. on the edge of a FOV of 180°. The images of spheres is relatively simple, only circles with random positions and radii. Points evenly sampled on these arcs, denoted as P_{LG} and P_{SG}, are grouped like P_L.

Next, P_{LG} and P_{SG} are distorted with known parameters to obtain P_{LD} and P_{SD}, as shown in Figure 9(b). Here, for simplicity, we let the distortion center coincide with the center of the image, i.e. $[x_c, y_c]^T = [0, 0]^T$. Figure 9(c) shows that an independent, zero mean, zero-correlated, two-dimensional Gaussian noise is added to P_{LD} and P_{SD}. The points with noise are denoted as P'_{LD} and P'_{SD}.

In a single run of the estimation, a group of three P'_{LD}s is generated to serve as P_L which is used to estimate the distortion center. A P'_{SD} or another P'_{LD} is generated to serve as P_{tar}. In order to make the experiment more rigorous, we generate another P_{SD}(notice, not P'_{SD} here) other than that used in the estimation to evaluate the estimation result. This P_{SD} is undistorted by the estimated distortion parameters to get $\overline{P}_{SD}$. The average distance d between $\overline{P}_{SD}$ and its corresponding ground truth P_{SG} can intuitively reflect the accuracy of our method. So a test set used here is $\{P'_{LD} * 3, P'_{SD}(orP'_{LD}), P_{SD}\}$.

We vary the noise level w in increment of 1 pixel starting from 1 pixel, and vary the number of the targets n under different noise levels. For each (w, n) pair, the proposed method is performed on 100 random test sets. Figure 10 reports the inter-quartile range (IQR) of d when P'_{SD} is used as P_{tar}.

Although becoming larger with the increasing noise level, d is controlled no more than 3.5 pixels, even when w reaches up to 5 pixels. When w is lower than 2 pixels, our method achieves sub-pixel accuracy. We can see that when more spheres are involved, the performance of the method is enhanced. Both the accuracy and the stability of the algorithm are increased when the number of the spheres reaches 6. The distortion is a polynomial of the radial distance r. The estimation of the distortion parameters is essentially a process of polynomial fitting. More targets will provide more image points which can cover a larger range of r. This improves the stability of the polynomial fitting. These results demonstrate that our method is robust to noise when the spheres are employed.

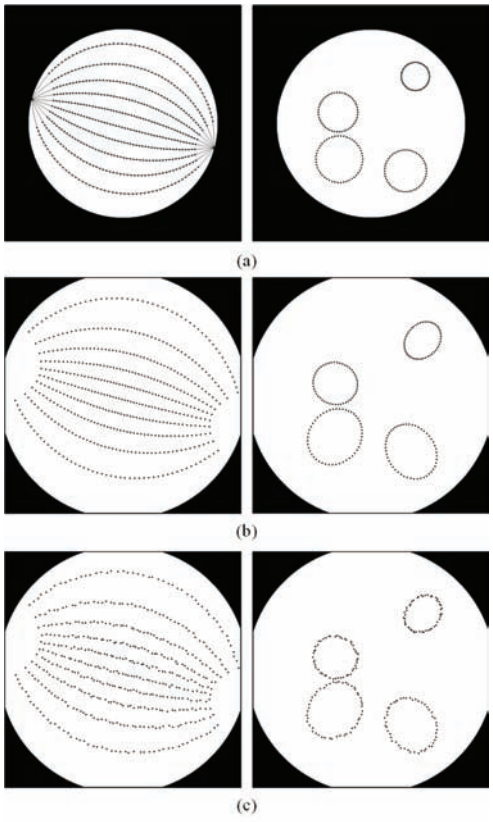

Figure 9 | Illustration of the experiment using fully synthetic data. (a) Points on the ideal image. (b) Points in (a) are distorted. (c) Distorted points are contaminated by Gaussian random noise.

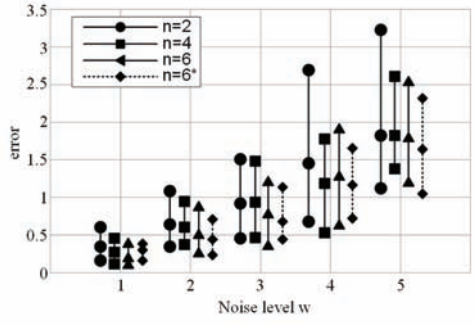

Figure 10 | The inter-quartile range (IQR) of is fixed at 0.

On the other hand, when a group of lines is used as the target, the result is not so good, as show in Figure 11. Even if the number of the lines increases, the improvement of the result is limited. The main reason for this is that the image of a space line under stereographic projection is only a portion of a circle, and circle fitting with points lying on a portion of a circle is an ill-conditioned problem. When a circle is fitted to these points, owing to the stronger fitting capacity of quadratics and the lack of geometric constraints, the objective function (19) has a great chance of reaching the minimum even if the undistorted points are very far from the ground truth. This will cause the iteration to converge to the wrong parameters. However, the image of a sphere is itself a complete circle; thus, the problem described above will not exist when spheres are used.

4.2. Experiment on Simulated Images

The complete process of the method is carried out in this experiment, including the extraction of the image points. The experimental setup is the same as that described in Section 4.1 except that the ground truth of the points doesn't exist. The parameters of the distortion are still known. Therefore, in this section, we directly examine the estimated parameters. To obtain the simulated images, two virtual scenes are created by the OGRE engine, as shown in Figure 12(a) and 12(b).

Scene 1 contains a plane target textured by some stripes to provide parallel lines. Scene 2 contains several identical spheres placed at random positions. We don't care about the interval between the lines and the size of the spheres. We have modified the engine to render blurry images. A Gaussian Blur with a radius of 2 pixels is applied here, and the coefficients k_1 and k_2 are set to $3E^{-6}$ and $6E^{-13}$, respectively. Three images of scene 1 and one image of scene 2 comprise a test set. The proposed method is performed 100 times on a single test set to investigate its stability. The estimated distortion center is $c = [-0.047 \pm 0.89, -0.036 \pm 0.88]^T$, which is relatively stable. Figure 13 shows the results of the distortion coefficients.

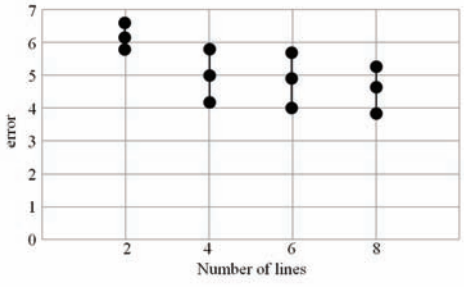

Figure 11 | The inter-quartile range (IQR) of is set to 1.

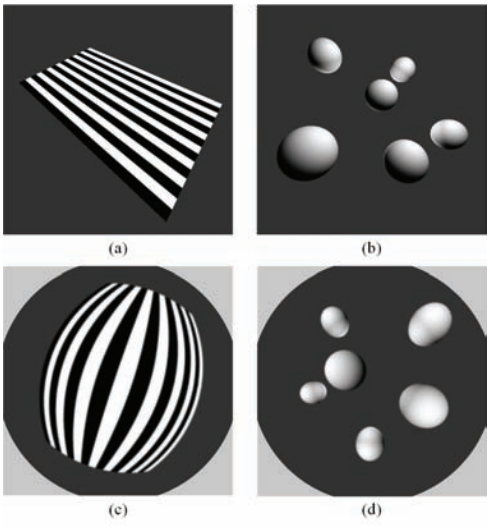

Figure 12 (a) and (b) Scenes containing a plane textured by black-and-white stripes and some spheres, respectively. (c) and (d) Distorted fish-eye images of (a) and (b), rendered by the modified OGRE engine.

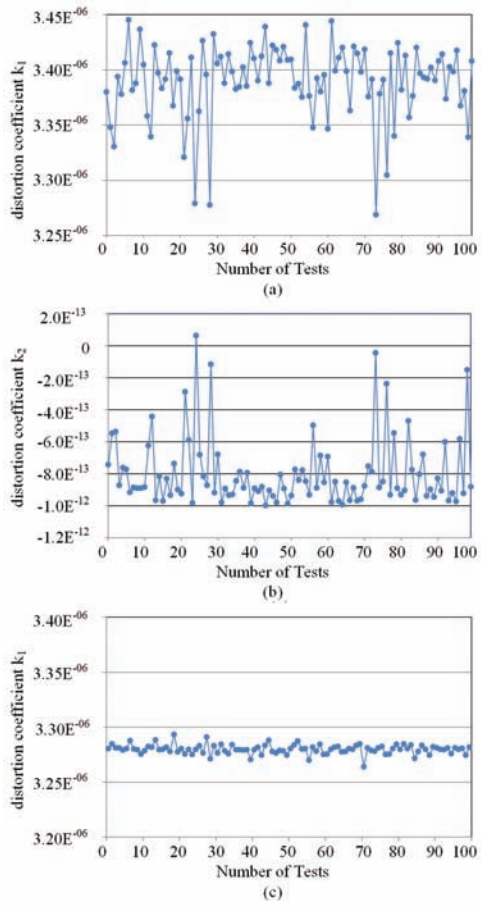

Figure 13 (a) and (b) Results of 100 runs of our method with the same test set. Both is fixed at 0.

The result shows that when both of the coefficients are estimated, the value of k_2 fluctuates, and so does k_1. On the other hand, when k_2 is fixed to 0, the stability of the estimation is enhanced. This explains why the stability of the algorithm is improved when only k_1 is estimate, especially when the noise level increases, as shown in Figure 10. Thus, it is recommended that k_2 should be omitted when an actual estimation is performed.

4.3. Experiment on Real Images

In this subsection, we give a quantitative evaluation of our simulation method. The error between the relative distortion of the real and the simulated image is measured.

The device used in the experiment is an inexpensive fish-eye camera. The size of its sensor is 704*576 in pixels. The distortion parameters are estimated from the pictures of the targets, as shown in Figure 14.

The estimation result is $c = [25.82, -6.83]^T$ and $(k_1, k_2) = (-1.61E^{-6}, 2.5E^{-13})$.

A chessboard pattern is captured with the target device, and the corresponding simulated image is given, as shown in Figure 15.

Corners are extracted from both of the images. The ratio between the radial distance of the corners in the observed image and their distortion-free positions is a relative measure of distortion. The similarity between the real and the simulated image can then be determined by comparing this ratio, as shown in Figure 16.

For points near the image center (radial distance smaller than 60%), the relative error is very small. The error increases with the radial distance, which could have been expected due to the limited number of terms used in the distortion polynomial. At the edge of the images, the error between the real and the simulated images is about 1.2%. This is a good enough result.

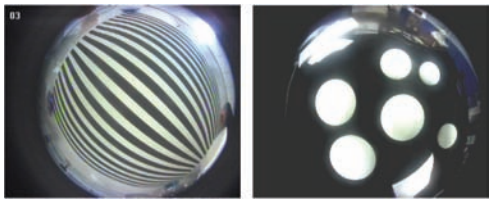

Figure 14 Images captured by a real fish-eye camera used to estimate the distortion.

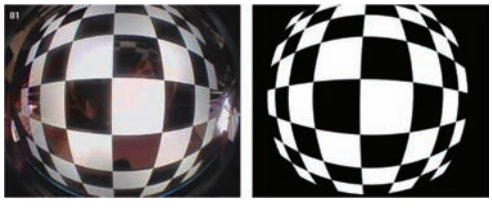

Figure 15 An image of a chessboard pattern captured by the real fish-eye camera and the corresponding simulated image.

4.4. Validation of Practicality

In this subsection, the practicality of our method is validated through an actual application of vehicle omniview system. In a common vehicle omniview system, there are often four fish-eye cameras mounted: one in the front of the vehicle, one in the back of the vehicle, one in the left outside mirror, and one in the right outside mirror. These four simulated fish-eye images are shown in Figure 17(a). The imaging system simulated above is used here. The ground is textured with chessboard pattern to clarify the result. The inverse perspective transformation is adopted to transform the image parts containing the ground into top views according to the parameters of the cameras. The top views are then fused into one single omniview image, as shown in Figure 17(b).

As we can see, compared with the sides of the vehicle, the top views of the ground in front and rear of the vehicle are more blurred and mismatched. The main reason of this is the mounting angle of the cameras. Both of the side cameras look straight downward, while the cameras on both ends of the vehicle raise their line of sight in order to avoid being occlude by the bumpers. The image parts of ground in these images are closer to the edge, where the distortion is more severe. The quality of the top view transformed from these images is poor. To solve this, we can modify the algorithm, change the camera mounting position, or even consider other cameras. In this way, problems can be found before the assembling of the actual system and the cost is saved.

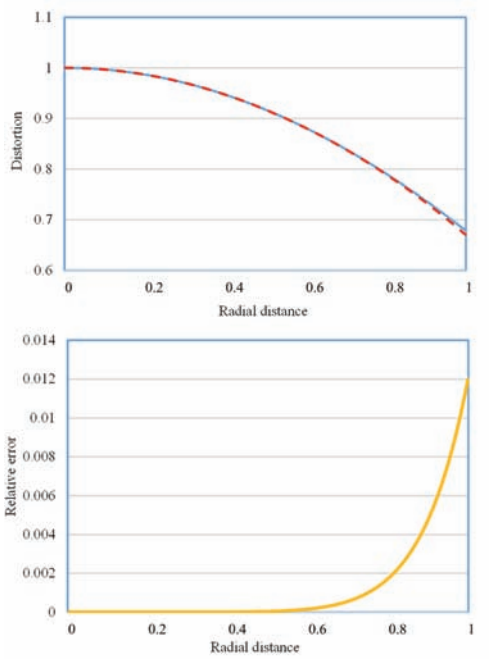

Figure 16 The ratio between the ideal and distorted position of the corners in the real and simulated image and the relative error calculated between the two images.

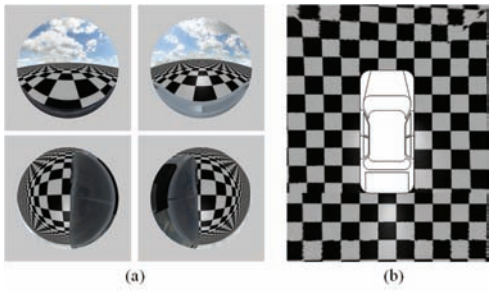

Figure 17 (a) The simulated images from the four fish-eye cameras mounted on the vehicle. (b) The omniview image generated from the images in (a).

5. CONCLUSION

In this paper, we presented a postprocessing method to simulate the output image of fish-eye cameras. The proposed method establishes a pixel-wise mapping between the simulated image and the cube-map rendered by a graphic engine, according to the stereographic projection model. For a given pixel position in the simulated image, a direction vector starting from the origin is determined. Its pixel value can be obtained from the pixel pointed by this vector in one of the prerendered images. The output of this step is an ideal fish-eye image. It has nothing to do with any specific camera. After the mapping is established, the distortion parameters of the camera is taken into account. We proposed a robust estimation method of the distortion parameters of the fish-eye imaging system. It outputs the position of the distortion center on the image plane of the camera, and the coefficients of a polynomial describing the radial distortion in the fish-eye image. The method was tested on synthetic, simulated, and real images. The experimental results showed that our method can yield accurate parameters even in the presence of noise. The estimated distortion parameters are then used to adjust the mapping relationship. A major contribution of this work is the camera-specific simulation that optimizes the efficiency of the algorithm development and the cost of the selection and preinstallation of the equipment in intelligent perception systems using fish-eye cameras. Also, the proposed simulation method is easy to implement and less expensive than existing commercial simulators.

DATA AVAILABILITY STATEMENT

All data used to support the findings of this study are included within the article.

CONFLICTS OF INTEREST

The authors declare they have no conflicts of interest.

AUTHORS' CONTRIBUTIONS

All authors contributed to the design and implementation of the research, to the analysis of the results, and to the writing of the manuscript. All authors read and approved the final manuscript.

Funding Statement

The funding sources have made a financial contribution to the research and the preparation of the article. They had no involvement in the work beyond providing a financial contribution.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 51805203, and by Jilin Scientific and Technological Development Program under Grant No. 20190201023JC, and by the Development and Reform Commission of Jilin Province under Grant No. 2019C054-2.

REFERENCES

[1] L. Sun, L. Yao, J. Rong, J. Lu, B. Liu, S. Wang, Simulation analysis on driving behavior during traffic sign recognition, Int. J. Comput. Intell. Syst. 4 (2011), 353–360.
[2] Q. Shao, J. Hu, W. Wang, Y. Fang, M. Han, J. Qi, J. Ma, Composable instructions and prospection guided visuomotor control for robotic manipulation, Int. J. Comput. Intell. Syst. 12 (2019), 1221–1231.
[3] F. Bounini, D. Gingras, V. Lapointe, D. Gruyer, Real-time simulator of collaborative autonomous vehicles, in Proceeding of 2014 International Conference on Advances in Computing, Communications and Informatics, New Delhi, India, 2014, pp. 723–729.
[4] M.A. Abbas, R. Milman, J.M. Eklund, Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles, Can. J. Electr. Comp. Eng. 40 (2017), 12–22.
[5] P. Gáspár, J. Bokor, A. Mihály, Z. Szabó, T. Fülep, F. Szauter, Robust reconfigurable control for in-wheel motor vehicles, in Proceeding of 2014 IEEE International Electric Vehicle Conference (IEVC), IEEE, Florence, Italy, 2014, pp. 1–6.
[6] M. Hafner, T. Pilutti, Dynamic trajectory planning for trailer backup, in Proceeding of 2014 International Conference on Systems, Man, and Cybernetics (SMC), IEEE, San Diego, CA, USA, 2014, pp. 2501–2506.
[7] R. Molenaar, A. van Bilsen, R. van der Made, R. de Vries, Full spectrum camera simulation for reliable virtual development and validation of ADAS and automated driving applications, in Proceeding of 2015 IEEE Intelligent Vehicles Symposium (IV), IEEE, Seoul, South Korea, 2015, pp. 47–52.
[8] Introduction to the robot simulator CoppeliaSim on the official website of CoppeliaSim, 2020, https://www.coppeliarobotics.com
[9] Introduction to Robotmaster on the official products page, 2020, http://www.robotmaster.com/en/products
[10] Introduction to Delfoi Robotics on the official website, 2020, http://www.delfoi.com/references/robotics/
[11] W. Shi, M.B. Alawieh, X. Li, H. Yu, Algorithm and hardware implementation for visual perception system in autonomous vehicle: asurvey, Integr. VLSI J. 59 (2017), 148–156.
[12] X. Hu, H. Zheng, Y. Chen, L. Chen, Dense crowd counting based on perspective weight model using a fisheye camera, Optik. 126 (2015), 123–130.
[13] L. Meinel, M. Findeisen, M. Heß, A. Apitzsch, G. Hirtz, Automated real-time surveillance for ambient assisted living using an omnidirectional camera, in Proceeding of 2014 International Conference on Consumer Electronics (ICCE), IEEE, Las Vegas, NV, USA, 2014, pp. 396–399.
[14] H.M. Becerra, C. Sagüés, Y. Mezouar, J.B. Hayet, Visual navigation of wheeled mobile robots using direct feedback of a geometric constraint, Auton. Robot. 37 (2014), 137–156.
[15] I.F. Mondragón, P. Campoy, C. Martinez, M. Olivares, Omnidirectional vision applied to Unmanned Aerial Vehicles (UAVs) attitude and heading estimation, Robot. Auton. Syst. 58 (2010), 809–819.
[16] H. Afshari, V. Popovic, T. Tasci, A. Schmid, Y. Leblebici, A spherical multi-camera system with real-time omnidirectional video acquisition capability, IEEE Trans. Consum. Electron. 58 (2012), 1110–1118.
[17] Y. Wang, X. Gong, Y. Lin, J. Liu, Stereo calibration and rectification for omnidirectional multi-camera systems, Int. J. Adv. Robot. Syst. 9 (2012), 467–485.
[18] F. Benamar, S. Elfkihi, C. Demonceaux, E. Mouaddib, D. Aboutajdine, Visual contact with catadioptric cameras, Robot. Auton. Syst. 64 (2015), 100–119.
[19] S. Yu, M. Lhuillier, Surface reconstruction of scenes using a catadioptric camera, in: A. Gagalowicz, W. Philips (Eds.), Proceeding of 5th International Conference on Computer Vision/Computer Graphics Collaboration Techniques, Springer, Berlin, Germany, 2011, pp. 145–156.
[20] D. Kim, J. Paik, Three-dimensional simulation method of fish-eye lens distortion for a vehicle backup rear-view camera, J. Opt. Soc. Am. A 32 (2015), 1337–1343.
[21] D.C. Brown, Close-range camera calibration, Photogramm. Eng. 37 (1971), 855–866.
[22] C. Geyer, K. Daniilidis, A unifying theory for central panoramic systems and practical implications, in: D. Vernon (Ed.), 6th European Conference on Computer Vision, Lecture Notes in Computer Science, vol 1843, Springer, Berlin, Heidelberg, Germany, 2000, pp. 445–461.
[23] M. Fleck, Perspective Projection: The Wrong Imaging Model, Technical Report, Department of Computer Science, University of Iowa, 1995.
[24] R. Swarninathan, S.K. Nayar, Non-metric calibration of wide-angle lenses and polycameras, in Proceeding of 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, Fort Collins, CO, USA, 1999, pp. 413–419.