IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 13, 2019, accepted August 27, 2019, date of publication September 2, 2019, date of current version September 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938975

Embedded GPU 3D Panoramic Viewing System Based on Virtual Camera Roaming 3D Environment

HAO MENG, FEI YUAN, YILI XU, AND TIANHAO YAN
College of Automation, Harbin Engineering University, Harbin 150001, China

Corresponding author: Fei Yuan (bohelion@hrbeu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61203255 and Grant 61175089, and in part by the Fundamental Research Funds for the Central Universities under Grant HEUCF180405.

ABSTRACT The traditional 2D panoramic view has the problem of small imaging range, low utilization of original image and serious stretching of the foreground. This paper designs a 3D panoramic viewing system. The real-time performance of the panoramic system requires a large number of parallel image transformations. The general-purpose processor CPU is incapable. This paper designs an embedded GPU solution. The core computing module is based on NXP's i.MX6Q application processor and builds an embedded Linux system. Using V4L2 to capture images to reduce memory and CPU usage, and use OpenGL ES shaders to process image transformations in parallel at high speed. Because it is in a 3D environment, it is impossible to cover all images through a still viewpoint like a 2D look-around. In order to roam the 3D scene, this paper proposes to design a virtual camera system. Using the virtual camera to roam the 3D environment, realize the virtual viewpoint in the 3D scene, traverse each scene in the grid, eliminate the dead angle of the field of view, and finally realize the 3D panoramic view, and verified on the embedded hardware platform, which can clearly roam every image on the surface. And recorded the video of 3-D looping. Experiments show that the 3D panoramic viewing system proposed in this paper has excellent performance and good effects. It has a good real-time performance on the image display, and the response speed is increased by 5%. The main operation is undertaken by the GPU and only takes a small number of CPU resources. The proposed virtual camera system can better realize the 3D panoramic view.

INDEX TERMS Embedded GPU, Linux, OpenGL ES, virtual camera, 3D panoramic view.

I. INTRODUCTION

The increase in the number of cars has brought about frequent traffic accidents, and the safety of automobiles has received more and more attention. From the initial ultrasound-based reversing radar [1] to the current camera-based reversing imaging system [2], all of them protect the safety of vehicles. However, these technologies can only monitor a limited area at the rear of the vehicle, and the blind areas on the left and right sides of the vehicle head increase the safety hazard of the vehicle. Therefore, research on the panoramic view system of automobiles has important practical significance for eliminating car blind spots and improving driving safety.

The automobile panoramic viewing system is to install several cameras around the automobile body to collect the image information around the automobile body, transform the collected image into a top view through image processing method, and finally stitch the top view image into a panoramic top view to display on the display, displaying the driving environment image to the driver in real-time. With the help of four or six cameras mounted around the car body, the third-person perspective of the car body is provided, so that the driver can observe every corner around the car body, eliminate blind areas and avoid the collision. This can provide more intuitive driving environment information for the driver, and reduce traffic accidents caused by blind areas of the driver's vision, especially. Provide the driver with reliable blind area display assistant function in the bad driving environment [3].

The original research of the panoramic view system mainly focused on the 2D look-around solution, which can seamlessly connect the captured images through image correction

The associate editor coordinating the review of this manuscript and approving it for publication was Yong Yang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

by multiple cameras. At the same time, the brightness and color of different cameras are adjusted to make the synthesized panoramic view look more coordinated However, there are some defects in 2D panoramic looping, such as small imaging range, low utilization of the original image, and serious stretching of the vision, which will make the object in the looping image distort and confuse people [4].In the current 2D viewing, only the portion of the fisheye image that is closer to the carrier is used, that is, the final viewing image can only display the situation near the carrier, and the original image information cannot be fully utilized.

At present, the panoramic viewing system has been fully developed in the direction of 3D. TI's 3D panoramic viewing solution enables images to be processed in real-time [5], using a lookup table generated by geometric correction parameters to simultaneously map input images to output images using DSP and DMA techniques. Finally, the 3D look-around is generated by the GPU. When the input resolution is 720p (1280' 720) and the output resolution is 880' 1080, the frame rate can reach 30fps. Since it is in a 3D environment, it is impossible to cover all images through a still viewpoint like 2D look-around. There is a dead angle and it is impossible to roam every image on the surface. It is increasingly important to use a virtual camera to roam a 3D environment. X Sun presented a novel paradigm for participatory design of 3D virtual scenes on mobile devices [6]. Z Duan uses Shanghai Technician School as the object of the 3D virtual roaming system and builds a completely immersive campus roaming system with 3Ds Max and Quest 3D platforms as development tools [7]. The 360-degree intelligent panoramic parking guidance system developed by Delphi China Research and Development Center can provide almost omni-directional three-dimensional images, has touch function, and can be operated by finger touch, regardless of the angle the driver wants to see, he can freely choose [8]. However, the price of the system is too high to be popularized and can only be applied to high-end vehicles.

This paper designs the hardware and software platform of the panoramic vision system by analyzing the demand of the panoramic viewing system of the car and combining the image processing simulation research. To cover the scene around the carrier with the minimum number of cameras, four fisheye cameras are selected to be placed in the front, rear, left and right positions of the carrier to take pictures. The image processing part of the real-time display is carried out on GPU, and the programming interface of GPU is OpenGL ES (OpenGL for Embedded Systems) [9]. The real-time performance of the panoramic system requires a large number of parallel image transformations. The general-purpose processor CPU is incapable. This paper designs an embedded GPU solution. The core computing module is based on NXP's i.MX6Q application processor and builds an embedded Linux system. In the process of image transformation, in order to realize the real-time display of a panoramic view, this paper uses V4L2 (Video for Linux 2) [10] to interact with the underlying layer of the camera. Linux uses V4L2 as the driving framework for video and audio capture devices and unifies the interface functions of the driver and user layers [11] to reduce memory and CPU usage. And uses OpenGL ES shaders to process image transformations in parallel at high speed. A 3D bowl mesh model is created by 3ds Max [12], [13], and the 2D image is restored to a 3D scene and imported into OpenGL for pseudo 3D reconstruction. In the 3D environment, it is impossible to cover all images through a still viewpoint like a 2D look-around. In order to roam the 3D scene, this paper proposes to introduce a virtual camera system. The virtual camera is used to roam the 3D environment to realize the virtual viewpoint in the three-dimensional scene, traverse each scene in the grid, eliminate the dead angle of the field of view, and finally realize the 3D panoramic view, which overcomes the defect of 2D look-around. And verified on the embedded hardware platform, able to clearly roam every image on the surface. Experiments show that the embedded GPU 3D panoramic viewing system proposed in this paper has excellent performance and good effects. It has a good real-time performance on the image display. The main operation is undertaken by the GPU and only takes up a small number of CPU resources. The paper is organized as follows. Section II designs the 3D panoramic view system, including the design of the panoramic system frame, the 3D look-around image transformation, and modeling loading, and the part of the virtual camera designed to roam the 3D environment. Section III is the experimental part. The corresponding experiments and results have specific descriptions in the second and third parts. Firstly, the fisheye camera is corrected, and then the 3D surround system proposed in this paper is implemented on the embedded device. Finally, we conclude the paper in the section IV.

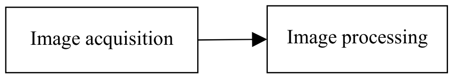


FIGURE 1. The panoramic abstract view.

II. DESIGN OF 3D PANORAMIC SCANNING SYSTEM

A. FRAMEWORK DESIGN OF THE PANORAMIC SYSTEM

The composition of the panoramic view system can be highly abstracted into two parts: image acquisition and image processing. As shown in Figure 1, the hardware and software design of this chapter will focus on these two parts.

The overall hardware and software framework of this paper is shown in Figure 2. The hardware layer of image acquisition is the TW6865[14] acquisition chip, which supports up to 4 channels of image acquisition. The operating system layer is the Linux system built by yocto [15] and the TW6865 driver in the kernel. The software interface layer mainly contains the Qt graphics interface library and V4L2 image. Collection library, OpenCV [16] image processing library, and OpenGL ES.

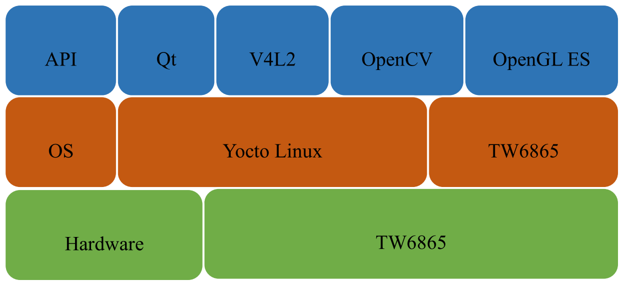


FIGURE 2. The system framework.

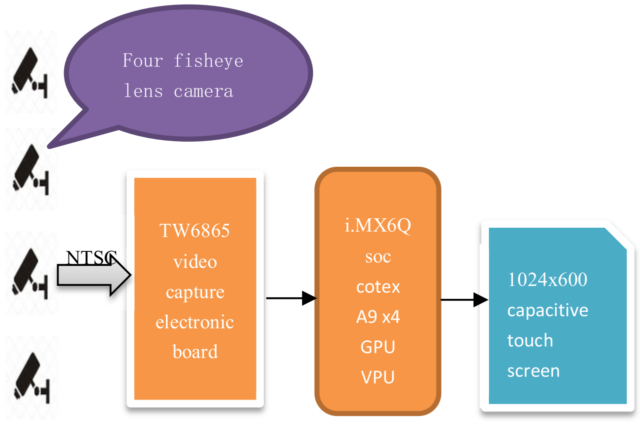

Four fisheye lens camera
NTSC
TW6865 video capture electronic board
i.MX6Q soc cotex A9 x4 GPU VPU
1024x600 capacitive touch screen

FIGURE 3. The hardware framework.

The structure diagram of the whole hardware system is shown in Figure 3. Firstly, the fisheye image of the 4-way NTSC system is transmitted to the video acquisition board, and the analog signal is decoded and encoded into digital signals, which constitute the image acquisition part, and then it is transmitted to the core computing module through the PCIE interface, which is responsible for the image processing part, and finally the image is processed and displayed on the screen.

The real-time performance of the panoramic view system requires a large number of parallel image transformations, and the general-purpose processor CPU is incapable. The embedded GPU scheme is designed in this paper. The core computing module is based on i.MX6Q application processor of the NXP company.

The program flow of the view system is shown in Figure 4. Image acquisition and processing can be divided into two parts: calibration and real-time display. The purpose of calibration is to obtain the parameters for transformation, and real-time performance is not required, so image acquisition is done by OpenCV. After the transformation parameters are obtained, they are processed by the real-time display part. The image acquisition here is accomplished by V4L2, and the GPU programming interface is OpenGL ES, which is used for image transformation operation of the panoramic viewing system. The embedded Linux systems run by this software are constructed by Yocto and the graphic interactive interface in Qt.

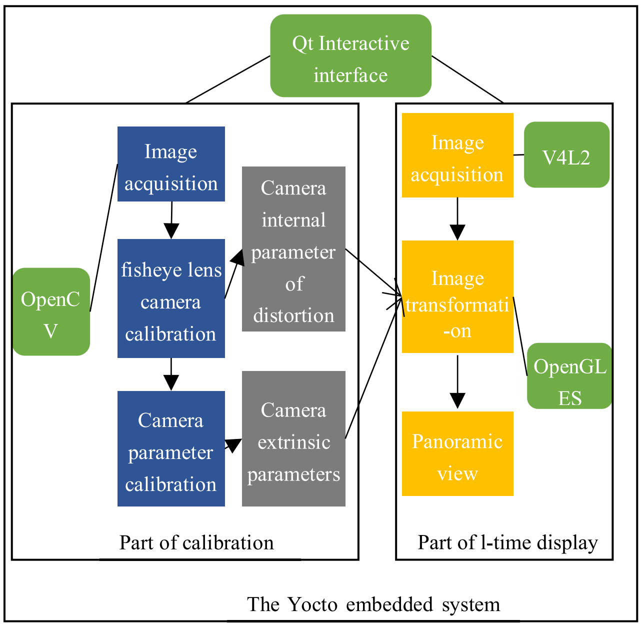


FIGURE 4. The software system flow.

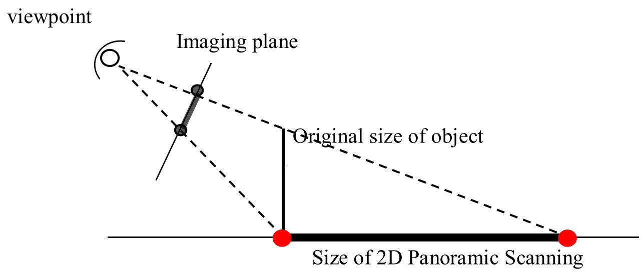


FIGURE 5. 2D looking around stretch.

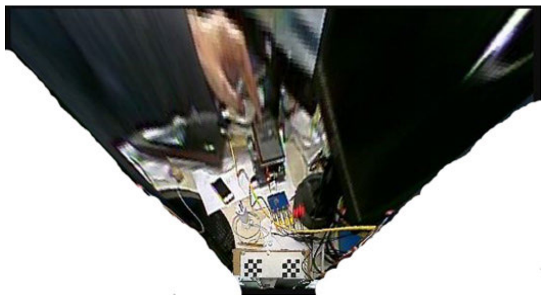


FIGURE 6. 2D looking around stretch.

B. 3D LOOK-AROUND IMAGE TRANSFORMATION AND MODELING

There are some defects in the 2D panoramic view. The higher the object is from the ground and the farther away from the camera, the more serious the stretching of the mosaic image is compared with the image on the imaging plane, the process is shown in Figure 5. This will deform the object in the loop image. The actual effect of this paper is shown in Figure 6.

To solve this problem, the pseudo 3D reconstruction was used in this article, which artificially assumes that the surrounding environment of the carrier is a bowl-shaped

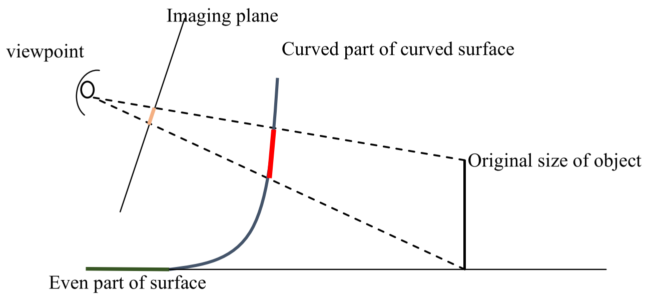


FIGURE 7. 3D panoramic scene.

structure, and projects the fisheye image onto the 3D bowl-shaped surface [17], which can be implemented on embedded devices with limited performance.

The virtual scene side view of 3D panoramic view is shown in Figure 7. The bowl-shaped surface mainly consists of two parts. The nearest surface to the carrier is the plane and the farther away is the surface [18]. The close object is projected onto the plane, and the distant object is projected onto the surface, which effectively eliminates the stretching problem in 2D looping.

Taking point $\pi_f(x, y, z)$ of the forward-looking part of the surface as an example to explain how to find the points in the corresponding source graph by reverse mapping [19].Known internal reference M_f, distortion coefficient K_f, homography matrix. Because now is the transformation from three-dimensional point to two-dimensional point, it is necessary to re-calculate homography matrix, so that the homography matrix is H_{3f}, and the corresponding point in the final source map is p_d, the calculation process is as shown in equation (1).

$$\begin{aligned}
\tilde{p}_u &= \begin{bmatrix} x_u \\ y_u \\ 1 \end{bmatrix}, \quad \tilde{\pi}_f = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \; H_{3f} = M_f \cdot \begin{bmatrix} R_f & \vec{T}_f \end{bmatrix} \\
r_u &= \sqrt{x_u^2 + y_u^2} \\
\theta &= \arctan r_u \\
r_d &= K_f \cdot \begin{bmatrix} \theta & \theta^3 & \theta^5 & \theta^7 & \theta^9 \end{bmatrix}^{\mathrm{T}} \\
p_d &= \frac{r_d}{r_u} \cdot p_u \\
K_f &= \begin{bmatrix} k_0 & k_1 & k_2 & k_3 \; k_4 \end{bmatrix}
\end{aligned} \tag{1}$$

Then interpolate p_d. The position of coefficients on the mixing table corresponds to the (x, y) coordinates of π_f. It can be regarded as an orthophoto projection. Finally, the pixel value of π_f can be obtained.

The bowl-shaped surface is composed of plane and bending parts. But in practice, to use OpenGL, it is necessary to transform continuous surface into discrete points and to string these points orderly by index. This paper uses professional 3D modeling software 3ds Max to model the surface.

The brief steps are as follows:

1) Draw a two-segment polyline in the left view of 3DS Max, with a total of three endpoints, and activate the vertex option.

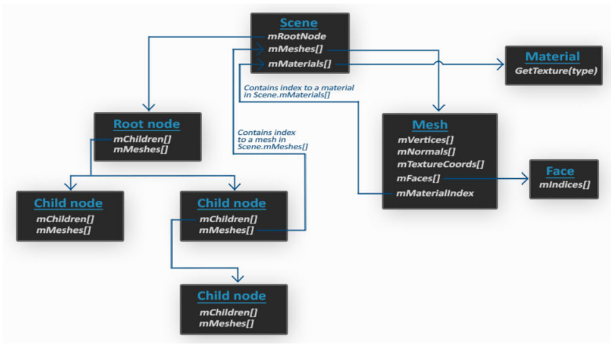


FIGURE 8. The data structure of the model.

2) Select the second endpoint, Bessel, and then convert it to the Bessel angle. Drag the next vertex vertically down to align horizontally with the vertex at the bottom of the broken line.

3) Select the bottom end of the broken line, Bessel, and then convert to Bessel angle, dragging horizontally to the right to a suitable position.

4) By adding a lathe object to the adjustment list, a bowl-shaped surface can be formed, and the Segments variable can control the number of vertices.

With the model, it also needs to be imported into OpenGL. The model loading library selected in this paper is Assimp (Open Asset Import Library), which can import dozens of different formats of model files. Once the model file is loaded by Assimp, the model data needed in this paper can be obtained from the Assimp object. It can transform the model files from different software into a unified data structure, which unifies the access mode of the program [20], [21]. When importing a model file, that is, when Assimp loads a whole file containing all model and scene data into a scene object, Assimp generates a corresponding data structure for all scene nodes and model nodes in the file, and the elements in the scene correspond to the model data. A simple model data structure generated by Assimp is shown in Figure 8.

C. VIRTUAL CAMERA ROAMING 3D ENVIRONMENT

Now in the 3D environment, it is impossible to cover all images through a static point of view like 2D looping, so this paper designs a virtual camera, which can roam every image on the surface. Before defining a camera, it needs its position in the world space, the direction of observation, a vector pointing to its right side and a vector pointing to its upper side. This actually creates three mutually perpendicular coordinate systems with the camera's position as the origin, which is called the camera's observation space [22]–[24].

The position of the camera is well understood. Simply speaking, the coordinates of the camera in the world coordinate system, as shown in Figure 9a), mark it P.

The next thing needs to be defined is the camera's direction vector, where the camera is pointing. Now let the camera point to the origin of the world coordinate system, and subtract the camera position from the original coordinates to

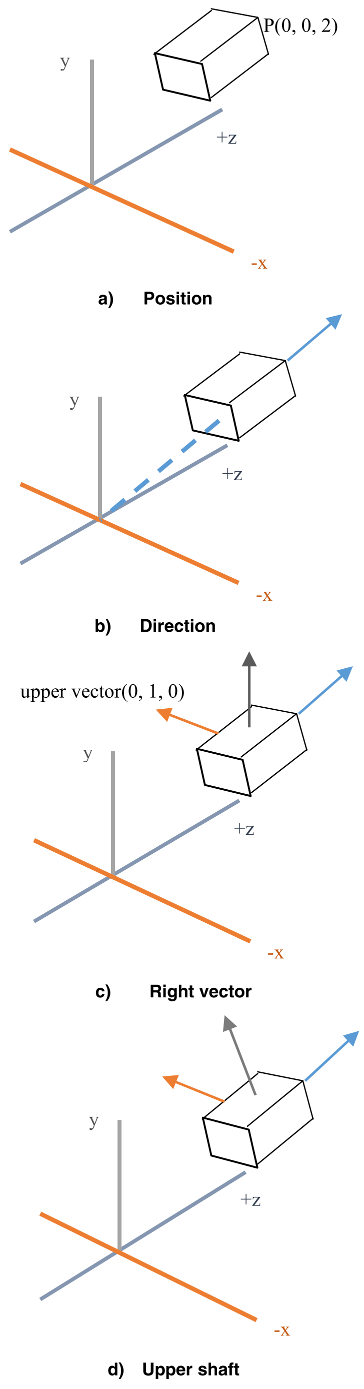


FIGURE 9. **System of an aerial camera.**

be the camera's pointing vector. In this paper, the direction vector is defined opposite to the direction vector. The purpose is to describe the projection of an external object onto the camera plane, and define this vector as the z-axis positive direction of the camera, as shown in Figure 9b), define it as $\vec{D}$.

Then a right vector is needed, which represents the positive x-axis of the camera space. Define the upper vector in a world coordinate system firstly, then cross-multiply the upper vector and the direction vector, and the result will be

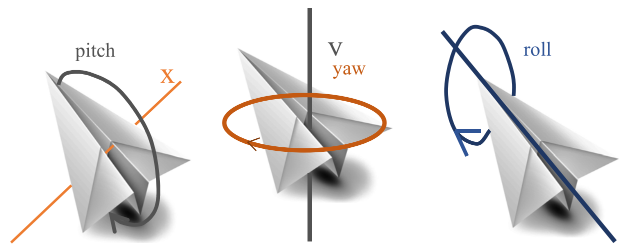


FIGURE 10. **Euler angle.**

perpendicular to the two vectors at the same time, this is the right vector, as shown in Figure 9c), which is defined as $\vec{R}$.

The final y-axis direction of the camera is obtained by cross-multiplying the right and direction vectors, as shown in Figure 9d). The above derivation is shown in equation (2).

$$\begin{aligned} \vec{D} &= \frac{0-P}{|P|} \\ \vec{R} &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \times \vec{D} \\ \vec{U} &= \vec{D} \times \vec{R} \end{aligned} \tag{2}$$

An observation matrix needs to be defined next, multiply it by any vector, and transform it into the coordinate space of the camera. As shown in equation (3), it is created by three axes plus a translation vector. Note that the position is negative, because this article expects the world to move in the opposite direction to its own. This observation matrix can effectively transform the world coordinates into the observation space just defined.

$$\begin{aligned} &LookAt \\ &= \begin{bmatrix} R_x & R_y & R_z & 0 \\ U_x & U_y & U_z & 0 \\ D_x & D_y & D_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3}$$

The movement of the camera can be achieved by making an offset along each axis of the camera The movement of the camera can be achieved by making an offset along each axis of the camera position, so that the offset unit is b, as shown in equation (4).

$$\begin{aligned} P &= P \pm b \cdot \vec{D} \\ P &= P \pm b \cdot \vec{R} \end{aligned} \tag{4}$$

Now it is necessary to make the virtual camera move the angle of view, which is to change the direction vector of the camera according to the output. This involves the Euler angle which represents arbitrary rotation in 3D space [25], [26]. It consists of three values, Pitch, Yaw and Roll, as shown in Figure 10.

Pitch angle is the angle describing how to look up and down, yaw angle is the degree of looking left and right, roll angle is how to roll the camera. Combining these three angles, any rotation vector in 3D space can be calculated. In this virtual camera system, only the pitch angle and yaw angle are concerned in this paper, and the roll angle is not involved.

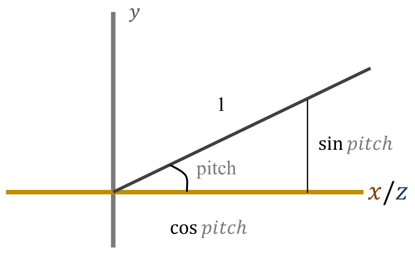


FIGURE 11. Angle of pitch.

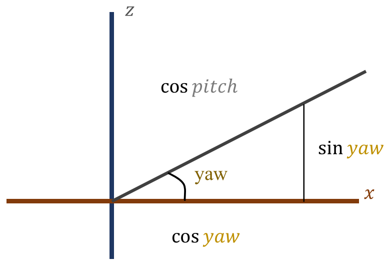


FIGURE 12. Yaw angle.

Given a pitch and yaw angle, they can be converted into a 3D vector representing the new direction vector.

Assuming that the angle of view looks to the y-axis on the *xz* plane, the components in each direction can be calculated based on a triangle, as shown in Figure 11, the components in the *y* direction are $\sin pitch$ and the components in the *xz* direction are $\cos pitch$.

The calculation of the yaw angle is the same as that of pitch angle. The component of direction *x* is $\cos yaw$, and the component of direction *z* is $\sin yaw$. In summary, the final result is summed up as equation (5), as shown in Figure 12.

$$\begin{aligned} x_d &= \cos pitch \cdot \cos yaw \\ y_d &= \sin pitch \\ z_d &= \cos pitch \cdot \sin yaw \end{aligned} \tag{5}$$

With these formulas, this paper can use these two angles to change the orientation of the camera. Define the offset of the elevation angle is *m* and the offset of the yaw angle is *n*. At the same time, to limit the elevation angle of the camera, it must be changed in the range of -90 and 90 degrees, so as to ensure that the virtual camera can only see the sky or the foot. The direction vector of the camera is shown in equation (6).

$$\begin{aligned} yaw &= yaw + n \\ pitch &= pitch + m, pitch \in (-90^\circ, 90^\circ) \\ \vec{D} \cdot \left|\vec{D}\right| &= \begin{bmatrix} \cos pitch \cdot \cos yaw \\ \sin pitch \\ \cos pitch \cdot \sin yaw \end{bmatrix} \end{aligned} \tag{6}$$

The next step is to give the virtual camera an additional function to achieve zooming, which is changed by changing the visual angle of the virtual camera. As shown in Figure 13, the size of the imaging plane remains unchanged. When the visual angle is reduced, a smaller area of the scene is projected onto the imaging plane, which results in the effect of zooming out, whereas when the visual angle is increased, the image is reduced. The comfortable visual angle of human eyes is about 60 degrees, and the virtual visual angle range defined in this paper is 45 degrees to 145 degrees.

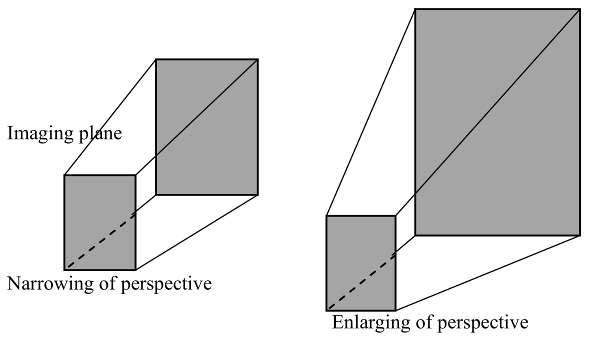


FIGURE 13. Scaling of virtual camera.

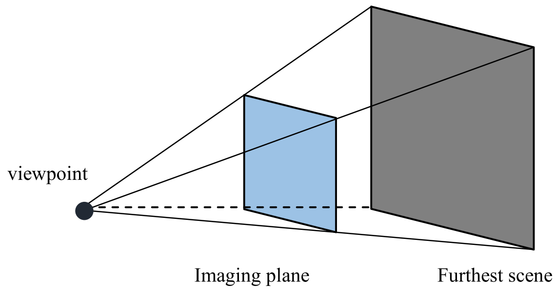


FIGURE 14. The structure of virtual camera.

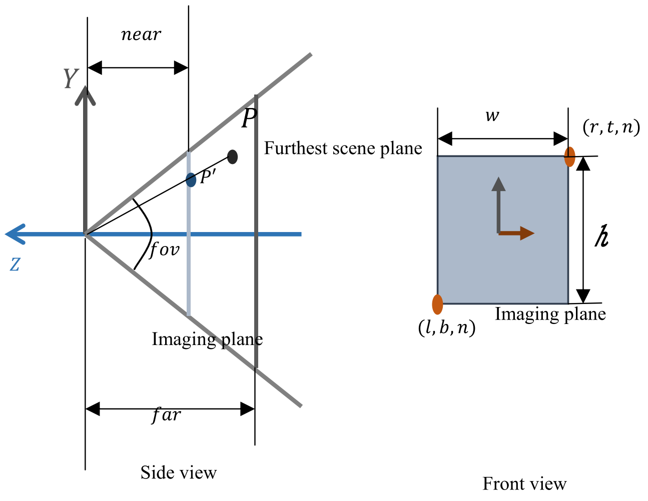


FIGURE 15. Side view of imaging structure.

Now it is possible to define the projection matrix of the virtual camera and project the point in the camera coordinate system onto the image plane. The imaging structure of the virtual camera is shown in Figure 14. The projection matrix is to project the points between the imaging plane and the farthest perspective onto the image plane.

The lateral view and emmetropic view of the imaging structure are shown in Figure 15.

Define the distance from the imaging plane to the lens as *n*, the distance from the perspective plane to the lens as *f*,

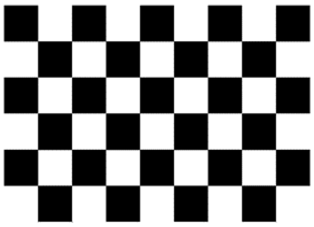


FIGURE 16. 8 × 5 chessboard.

the angle of view as fov, the coordinates of the upper right corner of the imaging plane as (r, t, n), the lower left corner as (l, b, n), the width as w, the height as h, the point between the imaging plane and the perspective plane is P, the imaging point is P'. The relationship between P and P', and the range of P is shown in equation (7).

$$\begin{aligned} P' &= \frac{n}{z}P \\ n &\leq z \leq f \\ -z\tan\frac{fov}{2} &\leq y \leq z\tan\frac{fov}{2} \\ -\frac{zw}{h}\tan\frac{fov}{2} &\leq x \leq \frac{zw}{h}\tan\frac{fov}{2} \\ P' &= \begin{bmatrix} x' & y' & n \end{bmatrix}^T,\ P = \begin{bmatrix} x & y & z \end{bmatrix}^T \end{aligned} \tag{7}$$

In this paper, a virtual camera class is encapsulated with c++. The membership attributes and basic algorithms of the virtual camera refer to bowl modeling. The scope of formula (7) needs to be regulated to $-1\sim1$, which is realized by the perspective function in QMatrix 4 × 4 class. In this case, the virtual camera is required only to be able to turn and zoom, and the offset value here is obtained by the gesture input value of the touch screen.

III. EXPERIMENTS AND RESULTS

A. FISHEYE CAMERA CORRECTION

The picture of the camera needs to be captured firstly. The Linux kernel provides driver support for TW6865 chips, and abstracts four cameras into character devices, which are presented in the "/dev" directory under the name of "video0"~ "video3". In OpenCV, only the VideoCapture class is called to open the camera object to read the picture, and the resolution of the picture is set to 720 × 480 in advance. Then set the parameters of the checkerboard. The checkerboard used in this paper is shown in Figure 16. Locate the corners in the checkerboard as shown in Figure 17. The number of corners is 8_5 and the size of the square is 30 mm. To improve the accuracy of calibration, 20 checkerboard pictures are needed. The flow chart of the internal parameter calibration program in this paper is shown in Figure 18.

The checkerboard corner recognition algorithm is realized by using the findChessboard Corners function encapsulated in OpenCV. If the recognition is successful, the position of the corner in the picture will be marked. Then take a picture of the checkerboard from different angles and collect 20 pieces. Next, the fisheye:: calibrate function is used to extract the internal parameters, external parameters and distortion parameters of the fisheye camera. If successful, the fisheye:: initUndistortRectifyMap function will be called first to generate a lookup table using the above parameters, and then the remap function will be called to correct the image using the table and show it out. The remap function comes with bilinear interpolation. If the error is appropriate, the camera parameters can be saved. The interface of the calibration program designed in this paper is shown in Figure 19.

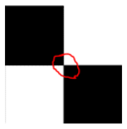
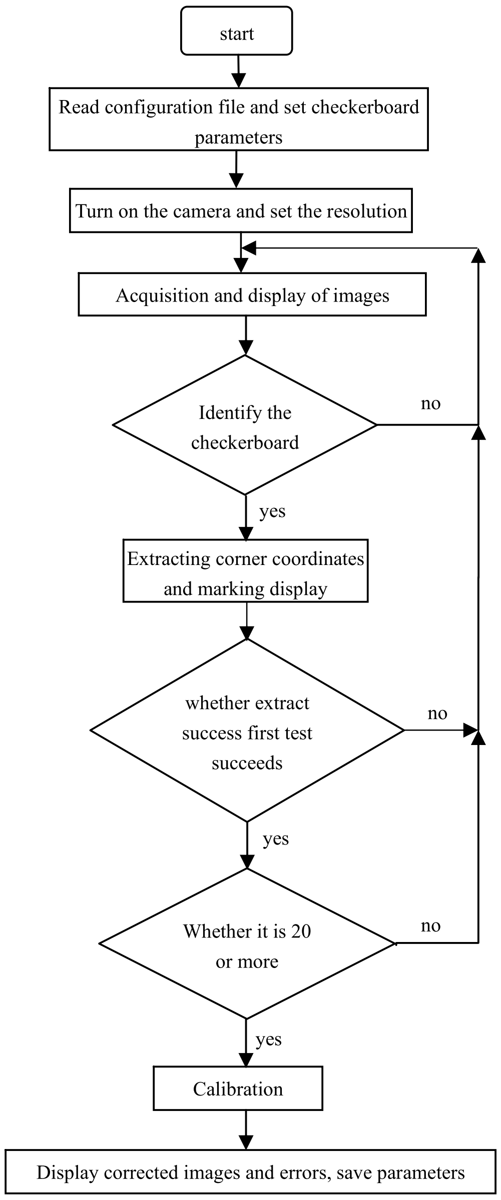


FIGURE 17. The corner in the red circle.


start
Read configuration file and set checkerboard parameters
Turn on the camera and set the resolution
Acquisition and display of images
Identify the checkerboard
no
yes
Extracting corner coordinates and marking display
whether extract success first test succeeds
no
yes
Whether it is 20 or more
no
yes
Calibration
Display corrected images and errors, save parameters
FIGURE 18. The flowchart of camera internal reference calibrated.

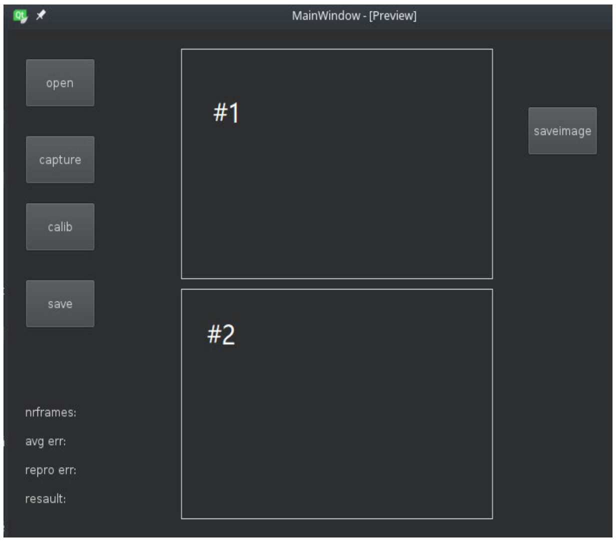


FIGURE 19. Camera correction interface.

TABLE 1. Calibration results of fisheye camera.

$$M_f = \begin{bmatrix} 252.228048 & 0 & 343.175729 \\ 0 & 216.266756 & 244.104780 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K_f = [-0.032284 \quad -0.003159 \quad 0.004117 \quad -0.002268]$$

$$E_f = 0.21$$

$$M_l = \begin{bmatrix} 250.390839 & 0 & 354.727356 \\ 0 & 215.471664 & 251.633865 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K_l = [-0.042230 \quad -0.032477 \quad 0.078630 \quad -0.043406]$$

$$E_l = 0.34$$

$$M_b = \begin{bmatrix} 235.988690 & 0 & 361.665318 \\ 0 & 206.590978 & 259.955237 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K_b = [-0.006227 \quad 0.013107 \quad -0.017921 \quad 0.004272]$$

$$E_b = 0.28$$

$$M_r = \begin{bmatrix} 256.347095 & 0 & 357.178065 \\ 0 & 221.247667 & 243.854762 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K_r = [-0.085608 \quad 0.060236 \quad -0.033393 \quad 0.003373]$$

$$E_r = 0.41$$

The internal parameters, distortion parameters and errors of the four fisheye cameras calibrated in this paper are shown in Table 1. M_f, K_f and E_f are internal parameters, distortion coefficients and errors of forward-looking cameras, M_l, K_l and E_l are left-looking camera parameters, M_b, K_b and E_b are rear-looking camera parameters, M_r, K_r and E_r are right-looking camera parameters.

The image before and after the pair of corrections is shown in Figure 20. The reduced image after correction is partly due to the backward mapping used in image transformation, and the 720 × 480 resolution of the corrected image will shrink after distortion.

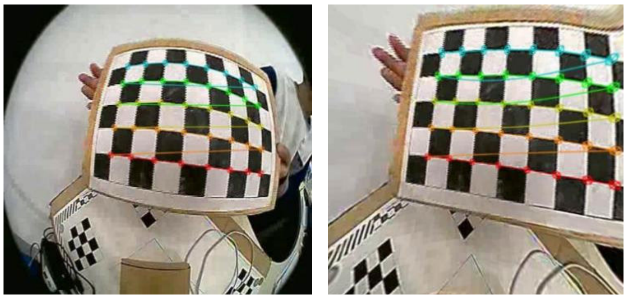


FIGURE 20. left) Before correction right) After correction.

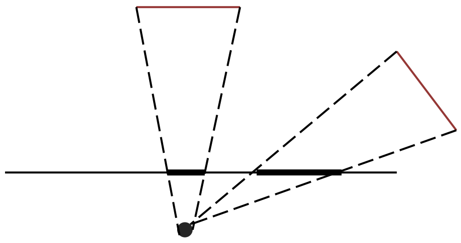


FIGURE 21. Large viewing angle leads to edge stretching.

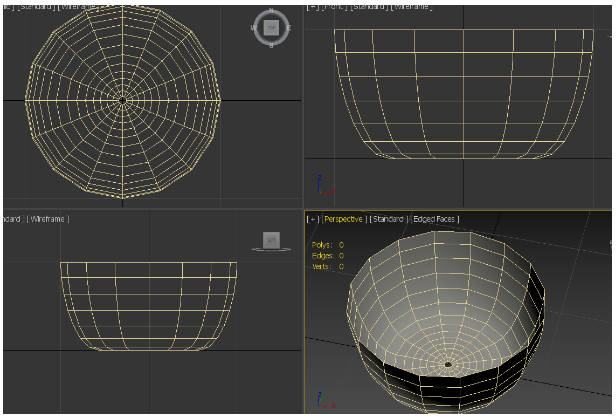


FIGURE 22. Bowl grid.

At the same time, it can be observed that the edge of the corrected image will be stretched, which is caused by the large angle of view. The process is shown in Figure 21, which is a normal phenomenon.

B. IMPLEMENTATION OF 3D CIRCUMVISION BASED ON OPENGL

In this paper, V4L2 is used to take charge of image acquisition in image transformation, and it is encapsulated into a class by C++.

The basic functions are as follows:

1) Open the device file. int fd = open(”/dev/video0”, O_RDWR)

2) Query the format supported by the device and complete it with the VIDIOC_QUERYCAP command.

3) Set the video frame format, pixel format (RGB565\YUYV\UYVY, etc.), resolution, scan order.

4) Request a frame buffer from the driver, command VIDIOC_REQBUFS.

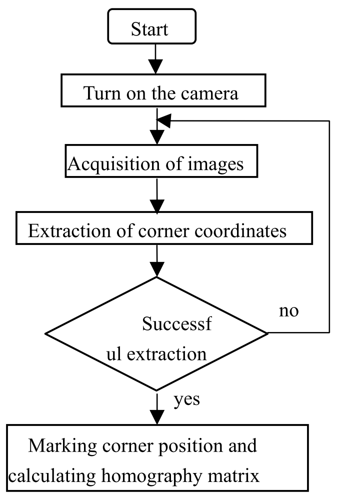

Start
Turn on the camera
Acquisition of images
Extraction of corner coordinates
Successf ul extraction
no
yes
Marking corner position and calculating homography matrix

FIGURE 23. **Homography matrix extraction flow chart.**

5) The frame buffer area is mapped to user space instead of replication to reduce memory and CPU usage.

6) Enter all frame buffers to store the data collected later.

7) Start video acquisition, command VIDIOC_ STREAMON.

8) Buffer the captured video frames out of the queue, command VIDIOC_DQBUF.

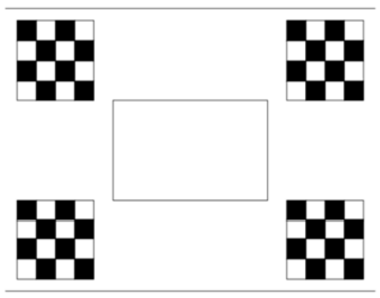


FIGURE 24. **Calibration map.**

9) After processing the frame buffer, re-entry and cyclic acquisition are performed, command VIDIOC_QBUF.

10) Stop video capture, command VIDIOC_STREAMOFF.

11) Close the video device, function close.

The modeling effect of the bowl surface using 3dmax is shown in Figure 22. The model is also divided into four sections, corresponding to the front, rear, left and right cameras.

In this paper, the bowl model is exported to obj format file by 3ds Max, and then imported into the program by Assimp, which is used as VAO object for OpenGL.

The homography matrix in 3D looping is different from that in 2D looping. The homography matrix in 2D looping

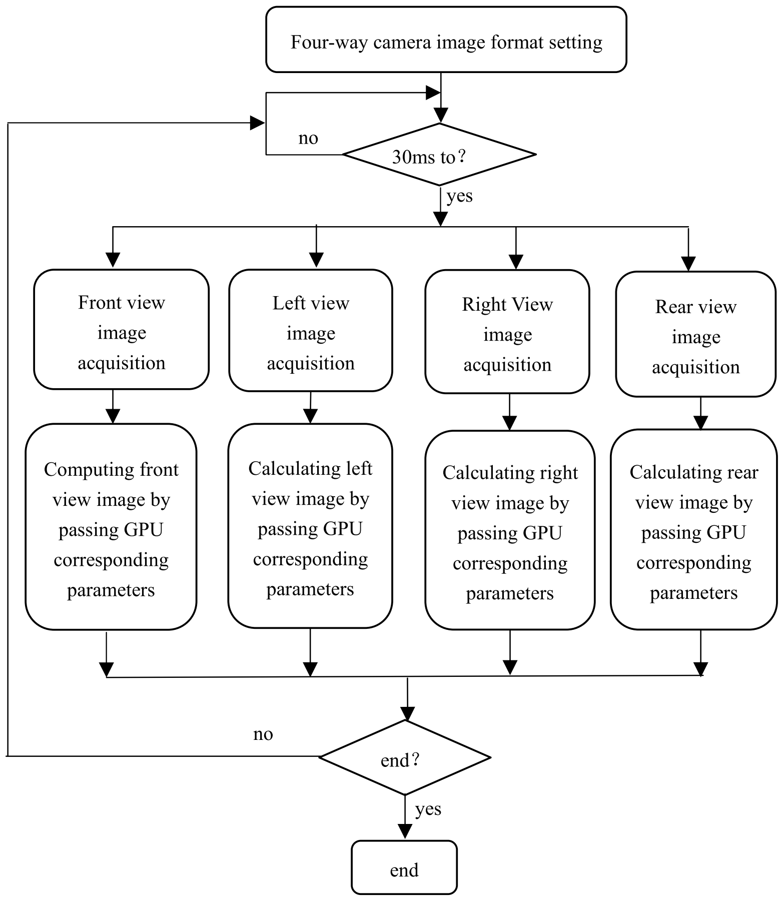

Four-way camera image format setting
no
30ms to?
yes
Front view image acquisition
Left view image acquisition
Right View image acquisition
Rear view image acquisition
Computing front view image by passing GPU corresponding parameters
Calculating left view image by passing GPU corresponding parameters
Calculating right view image by passing GPU corresponding parameters
Calculating rear view image by passing GPU corresponding parameters
no
end?
yes
end

FIGURE 25. **3D the flow chart of Surround view.**

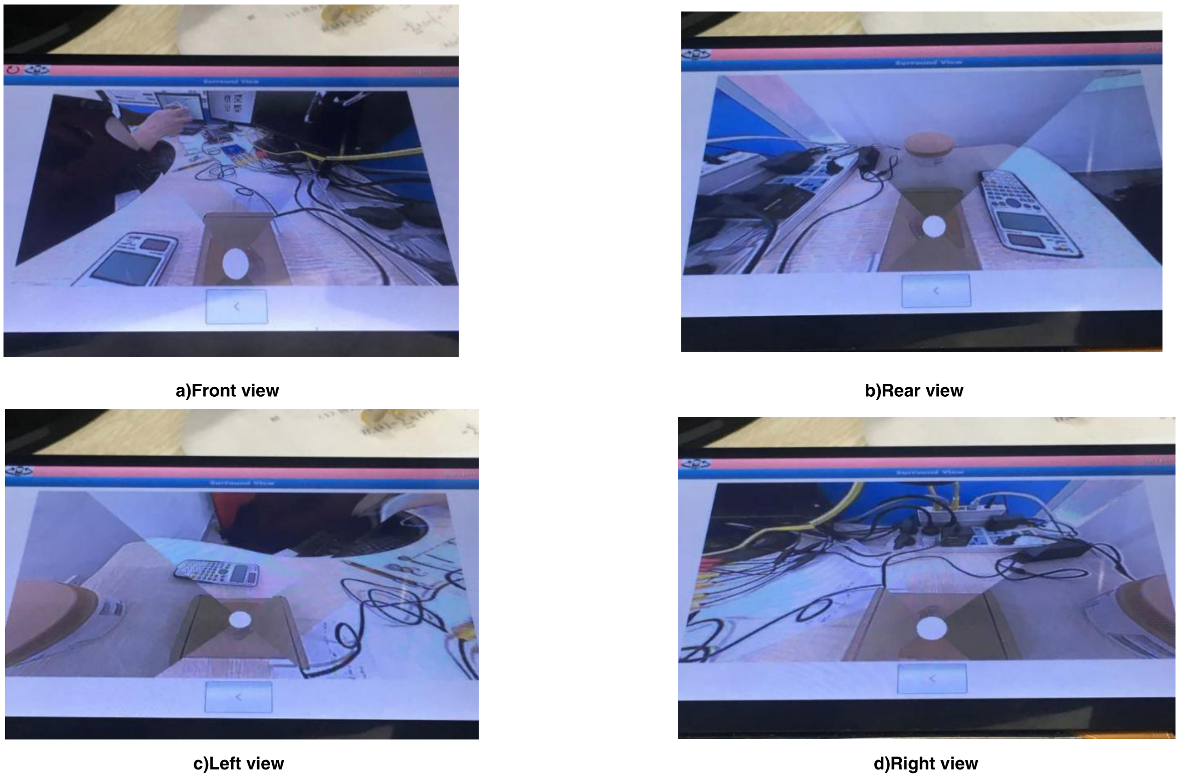


FIGURE 26. 3D surround view effect.

TABLE 2. Homography matrix for 3D circumferential view.

$H_{3f} = \begin{bmatrix} 0.998688 & -0.044495 & -0.025354 & 0.046824 \\ -0.048127 & -0.646208 & -0.761642 & 0.616983 \\ 0.017506 & 0.761863 & -0.647501 & 0.200464 \end{bmatrix}$
$H_{3l} = \begin{bmatrix} -0.061939 & 0.998078 & -0.002096 & -0.007226 \\ 0.998018 & 0.061911 & -0.011292 & 0.226193 \\ -0.011141 & -0.002791 & -0.999934 & 0.618810 \end{bmatrix}$
$H_{3b} = \begin{bmatrix} -0.999113 & -0.037253 & 0.019656 & -0.029931 \\ -0.041007 & 0.753657 & -0.655987 & 0.551392 \\ 0.009624 & -0.656211 & -0.754516 & 0.216988 \end{bmatrix}$
$H_{3r} = \begin{bmatrix} 0.074644 & -0.997150 & -0.010940 & -0.025741 \\ -0.997002 & -0.074400 & -0.021263 & 0.206539 \\ 0.020389 & 0.012494 & -0.999714 & 0.632498 \end{bmatrix}$

deals with the transformation between two-dimensional planes, while the homography matrix in 3D looping deals with the transformation from three-dimensional points to two-dimensional planes. The calibration flow of the matrix is shown in Figure 23. In this paper, based on a special calibration plate, the coordinates of the feature points of the calibration plate are changed from two-dimensional to three-dimensional, and the positional proportion of the feature points on the calibration plate is known, as shown in Figure 24. Assume that the plane z-axis of the calibration plate in Figure 24 is 0, combined with the two-dimensional coordinates in the image, and finally, the homography matrix is obtained by the solvePnP function in OpenCV [27], front view, left view, and rear view. And the right-view homography matrix results are shown in Table 2.

Then the V4L2 images and image fusion images are imported into GPU memory as textures. In order to reduce memory and CPU usage, the OpenGL extension function glTexDirectVIV is used to map the image memory area directly into GPU memory without any memory copy. And then the camera internal parameters, distortion parameters, bowl model and 3×4 homography matrix are imported into the fragment shader as uniform variables. Formula (1) is also compiled in the fragment shader to realize image transformation. The virtual camera system is introduced, which essentially brings a 4×4-projection matrix to transform objects in the world coordinate system into the standardized space of the virtual camera. The process is computed by the vertex shader.

Because there are four cameras, the program will create four threads responsible for each 30ms of image acquisition and transformation, in order to achieve a parallel effect. The final process of the program is shown in Figure 25, and the substantiation effect is shown in Figure 26. And recorded the video of 3-D looping. Experiments show that the frame rate of 3D looping can reach 28.5fps when the four input resolution is 720 × 480, the frame rate is 30fps and the output resolution is 720 × 480, which meets the requirements of real-time acquisition and display.

IV. CONCLUSION

The panoramic surround view is an important technology in advanced driver assistance systems. It is mainly composed of multi-channel wide-angle camera, image acquisition module and image processing module. It performs real-time processing on multiple images acquired at the same time to form a 360° view around a carrier, eliminating blind spots in the field of view, allowing the controller to observe the situation around the carrier in real time, and significantly improving security. In order to overcome the problems of small imaging range, low utilization of original image and serious stretching of vision, a 3D panoramic imaging system is designed in this paper, which can provide image information without dead angle around the carrier through a fisheye camera. Aiming at the requirement of embedded system and real-time image display, the hardware and software platform of panoramic view system is designed to provide the basis for algorithm research. On the hardware side, in order to cover the scene around the carrier with the minimum number of cameras, four fisheye cameras are selected to be placed in the front and back of the carrier for shooting, and the image processing part of real-time display is carried out on GPU. In terms of software, the calibration algorithm of camera parameters is mainly implemented by OpenCV, and the programming interface of GPU is OpenGL ES. In order to reduce the memory and CPU occupancy, this paper proposes using V4L2 to collect images, and using OpenGL ES shader to process image transformation in high speed and parallel, in order to achieve real-time display effect. A three-dimensional bowl-shaped mesh model is built by 3ds Max. The two-dimensional image is restored to a three-dimensional scene and imported into OpenGL for pseudo-3D reconstruction. In order to roam the three-dimensional scene, a virtual camera system is proposed in this paper. It uses the virtual camera to roam the three-dimensional environment, realizing the virtual viewpoint in the three-dimensional scene, traversing every scene in the grid and eliminating the dead angle of the field of view. Finally, it completes 3D panoramic looping without dead angle, overcomes the shortcomings of 2D looping, and realizes 3D panoramic looping on embedded devices based on V4L2, Qt and OpenGL ES. And recorded the video of 3-D looping. Although the image display has good real-time performance, there are still some shortcomings, mainly including: 1) the camera parameter calibration needs to determine the feature points manually, which is not convenient and fast enough. 2) During the whole process from camera calibration to circle vision, a lot of manual intervention is needed, which will cause a lot of trouble to a certain extent. The next step is to synthesize these steps.

REFERENCES

[1] M. V. Paulet, A. Salceanu, and O. M. Neacsu, "Ultrasonic radar," in *Proc. Int. Conf. Expo. Elect. Power Eng. (EPE)*, Oct. 2016, pp. 551–554.

[2] Y.-C. Lin, C.-T. Lin, W.-C. Liu, and L.-T. Chen, "A vision-based obstacle detection system for parking assistance," in *Proc. IEEE 8th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2013, pp. 1627–1630.

[3] Z. Zhu, G. Xu, B. Yang, D. Shi, and X. Lin, "VISATRAM: A real-time vision system for automatic traffic monitoring," *Image Vis. Comput.*, vol. 18, no. 10, pp. 781–794, Jul. 2000.

[4] M. W. Park, K. H. Jang, and S. K. Jung, "Panoramic vision system to eliminate driver's blind spots using a laser sensor and cameras," *Int. J. Intell. Transp. Syst. Res.*, vol. 10, no. 3, pp. 101–114, Sep. 2012.

[5] B. Zhang, I. Pekkucuksen, V. Appia, and A. U. Batur, "Method, apparatus and system for processing a display from a surround view camera solution," U.S. Patent 9 533 618, Jan. 3, 2017.

[6] X. Sun, Y. Wang, G. De Melo, W. Gai, Y. Shi, L. Zhao, Y. Bian, J. Liu, C. Yang, and X. Meng, "Enabling participatory design of 3D virtual scenes on mobile devices," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 473–482.

[7] Z. Duan, "The practice and exploration of virtual roaming based on 3Ds max," in *Proc. 3rd Int. Conf. Mechatronics Eng. Inf. Technol. (ICMEIT)*, Apr. 2019, pp. 1–7.

[8] M. Yu and G. Ma, "A visual parking guidance for surround view monitoring system," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 53–58. doi: 10.1109/IVS.2015.7225662.

[9] J. Han, *Introduction to Computer Graphics With OpenGL ES*. Boca Raton, FL, USA: CRC Press, 2018.

[10] Z. W. Chen, G. P. Zhang, G. Y. Lin, P. P. Cai, and X. L. Gao, "Design and implementation of embedded DVR system based on V4L2," *Electron. Des. Eng.*, vol. 25, no. 1, pp. 176–179, 2017.

[11] K. Mamindla, V. Padmaja, and C. H. NagaDeepa, "Embedded real time video monitoring system using ARM," *IOSR J. Eng.*, vol. 3, no. 7, pp. 14–18, Jul. 2013.

[12] T.-C. Huang and C.-Y. Lin, "From 3D modeling to 3D printing: Development of a differentiated spatial ability teaching model," *Telematics Inform.*, vol. 34, no. 2, pp. 604–613, May 2017.

[13] H. Chen, "Design of digital cell plane based on 3DMAX," in *Proc. Int. Conf. Virtual Reality Intell. Syst. (ICVRIS)*, Aug. 2018, pp. 279–283.

[14] H. Choi, J. G. Seo, D. H. Park, and E. S. Kang, "Precision distortion correction technique based on FOV model for wide-angle cameras in automotive sector," in *Emerging Trends in Image Processing, Computer Vision and Pattern Recognition*. 2015, pp. 533–549. doi: 10.1016/B978-0-12-802045-6.00035-1.

[15] O. Salvador and D. Angolini, *Embedded Linux Development Using Yocto Projects: Learn to Leverage the Power of Yocto Project to Build Efficient Linux-based Products*. Birmingham, U.K.: Packt Publishing, 2017.

[16] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. Newton, MA, USA: O'Reilly Media, 2016.

[17] T. Zhang, Y. Du, T. Huang, and X. Li, "GPU-accelerated 3D reconstruction of porous media using multiple-point statistics," *Comput. Geosci.*, vol. 19, no. 1, pp. 79–98, Feb. 2015.

[18] B. J. Fernández-Palacios, D. Morabito, and F. Remondino, "Access to complex reality-based 3D models using virtual reality solutions," *J. Cultural Heritage*, vol. 23, pp. 40–48, Jan./Feb. 2017.

[19] E. R. Dougherty, *Random Processes for Image and Signal Processing*. Bellingham, WA, USA: SPIE, 1999.

[20] G. Madges, I. Miles, and E. F. Anderson, "AnimDiff: Comparing 3D animations for revision control," in *Proc. Eurographics*, Apr. 2017, pp. 1–4. doi: 10.2312/egsh.20171007.

[21] T. Schulze, A. Gessler, K. Kulling, D. Nadlinger, J. Klein, M. Sibly, and M. Gubisch. (Jan. 2012). Open Asset Import Library (ASSIMP). Computer Software. [Online]. Available: https://github.com/assimp/assimp

[22] Y. Abdel-Aziz, H. Karara, and M. Hauck, "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry," *Photogram. Eng. Remote Sens.*, vol. 81, no. 2, pp. 103–107, Feb. 2015.

[23] C.-S. Lee, S. Chun, and S. W. Park, "Tracking hand rotation and various grasping gestures from an IR camera using extended cylindrical manifold embedding," *Comput. Vis. Image Understand.*, vol. 117, no. 12, pp. 1711–1723, Dec. 2013.

[24] N. O'Dwyer, N. Johnson, E. Bates, R. Pagés, J. Ondřej, K. Amplianitis, and A. Smolić, "Virtual play in free-viewpoint video: Reinterpreting samuel beckett for virtual reality," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR-Adjunct)*, Oct. 2017, pp. 262–267.

[25] P. D. Groves, "Principles of GNSS, inertial, and multisensor integrated navigation systems," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 30, no. 2, pp. 26–27, Mar. 2015. doi: 10.1109/MAES.2014.14110.

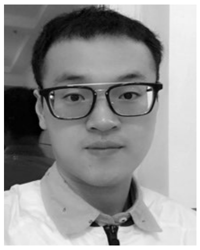
[26] B. Wang, J. Yu, C. Liu, M. Li, and B. Zhu, "Data snooping algorithm for universal 3D similarity transformation based on generalized EIV model," *Measurement*, vol. 119, pp. 56–62, Apr. 2018.

[27] Z. Qiu, D. Fu, M. Jiao, and H. Cao, "Three-dimensional model reconstruction based on non-measured photo," *Geomatics Sci. Technol.*, vol. 6, no. 1, pp. 1–7, 2018.

HAO MENG was a Visiting Scholar with the Robotics Laboratory, Dalhousie University, Canada, from 2006 to 2007. He is currently a Professor with the School of Automation, Institute of Robotics and Intelligent Control, Harbin Engineering University. He has participated in many scientific research projects, such as the International Cooperation Project of the Ministry of Science and Technology, the National Defense Science Foundation. At present, as a Project Leader, he has been presiding over two military horizontal projects and one municipal science and technology innovation talent special fund project. His research interests include advanced control theory and applications, robotics and intelligent control, machine vision inspection technology, and underwater high-speed vehicle motion control technology. He also participated in several international academic conferences.

FEI YUAN is currently pursuing the Ph.D. degree in control science and engineering with the School of Automation, Institute of Robotics and Intelligent Control, Harbin Engineering University, with a focus on computer vision. She helped her teacher in several projects, mainly using deep neural networks. Her main research interests include panoramic view, deep neural networks, image processing, and expression recognition.

YILI XU is currently pursuing the degree in control engineering with the College of Automation, Harbin Engineering University, with a focus on machine vision. His research interest includes panoramic vision. He is good at hardware and has a good hands-on ability.



TIANHAO YAN is currently pursuing the Ph.D. degree in control science and engineering with the School of Automation, Institute of Robotics and Intelligent Control, Harbin Engineering University. His research interests include spoken signal processing, pattern recognition, machine learning, and affective computing.