Virtual Modeling of Vision Sensor on Smart Cars

Ning-ning ZHANG¹,a, Xiang YANG², Xu YANG¹, Xine-YAN¹, Ya-Ou WANG¹, Li-Gao¹, Xing-Ren WANG³

¹Xi’an Traffic Engineering Institute, Xi’an, Shaanxi, China
²Yunnan Science Research Institute of Communication Co., Ltd. Kunming, Yunnan, China
³email: zoujixue@eurasia.edu

Abstract: In order to build a visual sensor in a virtual environment and simulate a smart car obtaining information from external environment, the OpenGL development environment is established based on the Microsoft Visual C++6.0 language program; then the image acquisition unit and processing unit model is built, and finally the vision sensor model is installed on the car model to realize the virtual modeling of the vision sensor on smart cars, so that the smart car can obtain information from external environment in a virtual manner.

1. Introduction

Before the mass production of smart cars, it is imminent to establish a R&D evaluation test field to ensure the safe driving of smart cars on the road and the development and evaluation of smart cars. The research and development methods based on simulation technology and driving simulator have become the mainstream in automobile intelligent technology and product research and development worldwide [1-2]. OpenGL is an independent transparent network operating system, which helps programmers to develop processing software of high-performance, highly powerful visual expressive graphics on hardware devices such as PC, workstation and supercomputers[3]. Therefore, it is a good choice to test the intelligent level of smart cars by building OpenGL development environment and realize the virtual simulation of visual sensors on smart cars with Microsoft Visual C++6.0 language program. Measurable photographs or auxiliary calibration camera imaging can be simulated with OpenGL, which means it will greatly improve the efficiency of photogrammetry as an important technical means[4]. The study[4] studied the consistency of the photogrammetric collinear equation and the perspective imaging mechanism of OpenGL. By analyzing the quantitative relationship between the photographic parameters and the model matrix, projection matrix and affine matrix in OpenGL[5], together with the operation on target object, imaging results similar to real images can be obtained. However, the above method is only limited to the operation of the model, so the parameter calculation is more complicated, and the photography process cannot be reproduced intuitively. The virtual visual sensor model established in this paper is realized under the OpenGL development environment built with the Microsoft Visual C++6.0 language program. It includes the vision sensor image acquisition unit (the parameters include the longitudinal angle of the vision sensor, the depth of the vision sensor, the resolution of the vision sensor, the installation position and the angle of the vision sensor) and the image processing unit (the parameters include the image pincushion deformation and the frame rate of the vision sensor). By corresponding the functions of the above units to the real vision sensors, users configure different parameters in the “Visual Sensor Parameter Setting” dialog box as needed. After the setting, the program will calculate the corresponding virtual vision sensor parameters; then the virtual vision sensor parameters are passed to the scene rendering unit, and the scene manager will re-render the 3D scene of the vehicle position and direction according to the dynamic data of the host.
vehicle (including position, speed, etc.), and recalculate the position and direction of the virtual vision sensor based on these data, so that the color image and depth image of the virtual vision sensor can be finally rendered; in order to simulate the image effect taken by the real vision sensor, the "image pincushion" parameter will perform certain distortion processing on the ideal image according to the parameters set by users[4].

2. Modeling of image acquisition unit

2.1 Establishment of part of the internal parameter model of visual sensors

Purpose: simulate the focal length of the vision sensor: adjust the size and shape of the photographed area.

2.1.1 The realization of focal length

The realization of focal length: gluPerspective(m_fFocusDis, m_aspect_ratio, 0.1f, 1000.0f).

Wherein m_fFocusDis is the focal length of the vision sensor. The first parameter fovy in the gluPerspective function in OpenGL represents the perspective. Here, the relationship between the perspective and the focal length is used to simulate the focal length. m_aspect_ratio is the aspect ratio of the vision sensor shooting area, in which the parameter m_aspect_ratio should be the same size as the window aspect ratio, so that the object will not be distorted in the window display. Specify the distance from the vision sensor to the closest clip plane of the shooting area (positive number): 0.1f, and specify the distance from the vision sensor to the farthest clip plane of the shooting area (positive number): 1000.0f.

2.1.2 Realization of longitudinal angle

First draw the vision sensor shooting area, and then establish an association with the parameter dialog: read the angle of view parameters from the configuration file; when the "vision sensor longitudinal angle" parameter is modified from the interface, the data is updated accordingly. The initial value of the angle of the two vision sensors are both 60°.

2.1.3 Realization of depth

Define the depth of the vision sensor in the area (0.1f, 1000.0f), and then establish an association with the parameter dialog box: read the depth parameter from the configuration file. The initial values of the angle of the two vision sensors are both 32°. Figure 1 shows a comparison of the scenes taken by the vision sensor when the viewing angle is 60° (left viewport) and 120° (right viewport). It can be seen from the figure that the range between the near and far planes of the viewing angle of 120° is greater than the viewing angle of 60°.

![Figure 1 Comparison of shooting scenes of vision sensors from different perspectives](image)

2.2 Establishment of external parameter model of vision sensor

Purpose: Observe the road information in the virtual scene by moving the visual sensor (position, direction).

After setting the size and shape of the shooting area, start to shoot the road scene by setting the position and direction of the vision sensor in the world coordinate system. Use functions in OpenGL: gluLookAt(eyeX, eyeY, eyeZ, lookAtX, lookAtY, lookAtZ, 0.0f, 1.0f, 0.0f); the position of the vision
sensor in the world coordinate system is \((\text{eyeX}, \text{eyeY}, \text{eyeZ})\); the orientation of the vision sensor is: \((\text{lookAtX}, \text{lookAtY}, \text{lookAtZ})\); the direction of the orientation of vision sensor: \((0.0f, 1.0f, 0.0f)\), and the top of the vision sensor is upward along the \(y\) axis. Figure 2 shows the comparison of scenes taken by the vision sensor with the vertex facing the negative \(x\)-axis direction (left viewport) and the vertex facing the positive \(y\)-axis direction (right viewport).

![Figure 2 Comparison of scenes taken by vision sensors with different vertex orientations](image)

**2.3 Establishment of viewing zone**

Purpose: to map the captured image to the pixel viewport.

\[
glViewport(0, 0, \text{rect. Width()}, \text{rect. Height()})
\]

By default, the viewport is set to occupy the entire pixel rectangle of the open window. The window size is the same as the set viewport size. But in order to select a smaller drawing area, this transformation can be achieved with the `glViewport` function. Defined a pixel rectangle in the window, and finally the captured image is mapped into this rectangle. This article divides the window area and displays two screens in the same window to display the shooting views of the two vision sensors. Among them \((0,0)\) is in pixels, and the position of the lower left corner of the viewport (with \((0,0)\) as the origin) is specified; \((\text{rect. Width()}, \text{rect. Height()})\) represents the rectangle of the viewport \(\text{Width}\) and \(\text{height}\).

![Figure 3 Schematic diagram of the viewing area of the vision sensor](image)

**2.3.1 Realization of resolution**

First, draw the vision sensor viewport: two deformed viewports of \(1/3\) window size

\[
glViewport (0, 0, \text{rect. Width ()}, \text{rect. Height ()});
\]

\[
\text{float width} = \text{rect. Width ()}/3;
\]

\[
\text{float height} = \text{rect. Height ()}/3;
\]

The position coordinates of the lower left corner on the screen are \((i, 0)\) \((i+400, 0)\)

Then, read the pixels: from the lower left corner of the image \((i, 0)\), \((i+400, 0)\) points, on a rectangular image with width and height of \(w\) and \(h\) respectively, read the content as GL_RGBA in OpenGL, and the fetched data is saved in the address of \&m_camera0Data.front() in GLubyte format.

Finally, establish an association with the parameter dialog: read the resolution width and height parameters from the configuration file, and assign the product of the two to the resolution. When the "width
and height" parameters are modified from the interface, the data will be updated accordingly. The initial values of the resolution width and height of the two vision sensors are 800*600.

3. Modeling of image processing unit

The image processing unit includes a processor and a memory. The function of the real vision sensor processor is to process slightly deformed or noisy images so that the pictures taken can be close to the view of human eyes. In order to achieve the shooting effect of the real vision sensor, this section adopts pincushion distortion for the images taken by the virtual vision sensor. In addition, the deformed image is stored.

3.1.1 The realization of pincushion deformity

First draw the pincushion distortion polygon rendering of the image, as shown in Figure 4:

![Figure 4 The effect picture of pincushion distortion polygon of the image](image)

Then, two rows of deformed pixel data are generated from the upper and bottom edges of the picture respectively, wherein the upper and lower x-axis directions are divided into i equal parts. Use temp. u=float (i)/(width-1) to represent the array subscript, starting from 0; use trigonometric functions to achieve smooth curve deformation: calculate distortion parameters: the upper edge data of the picture: temp.y=m_fboThreshold*sin((float)(i)/(width)*π); The bottom edge data of the picture: temp.y=height-m_fboThreshold*sin((float)(i)/(width)*π);

Create and bind an FBO, and then render the buffer object for off-screen rendering. Wherein there are two types of objects linked to the FBO. One is a texture object, and the other is a rendering buffer object (Renderbuffer object). Then link the rendering buffer to the FBO, and finally output the rendering result to the FBO. The texture is mapped to the deformed picture to realize the distortion of the picture and be output to the display interface.

Establish an association with the dialog:

Read from configuration file

sscanf (szBuf,"%f",&m_fboThreshold);

After modifying the "pincushion variable" parameter from the interface, update the data

CString strFBO;

GetDlgItem(IDC_EDIT_IMAGE_ANGLE)->GetWindowText(strFBO);

m_view->m_fboThreshold = atof(strFBO);

The image pincushion variables of the vision sensor can be adjusted in different parameters in the parameter dialog box according to the user's needs. The pincushion variables of the two vision sensors are both: 20.

3.1.2 Image storage

Storage format:

FreeImage is a free, open source, cross-platform (Windows, Linux and Mac OSX) image processing library that supports more than 20 image types (such as BMP, JPEG, GIF, PNG, TIFF, etc.). The biggest advantage of it is that it adopts a plug-in-driven architecture, which is fast, flexible, and easy to use, and has been widely used. In order to display the image (bitmap in any format) data stored in the memory array in the dialog box. This article adopts FreeImage library to achieve it with the storage format as .PNG.

Frame rate realization and image storage:
First create a timer in C++6.0: Define the parameters including: the number of frames recorded per second per frame, the file name generated each time, the serial numbers of the files, the current status after the start of the recording, recording or non-recording, temporary recording saving, then call the SetTimer function. SetTimer(2, 1000/m_captureNum, 0) means: call once every 1000/m_captureNum.

Finally, establish association with the image storage and the parameter dialog box: read the frame rate parameter from the configuration file, and when the “frame rate” parameter is modified from the interface, the data is updated accordingly. The initial values of the frame rate of the two vision sensors are: 20, which means 20 images are output (stored) in 1 second, and they are stored in a new folder in the form of bitmap mapping images.

4. Modeling of virtual smart car

In the road information collected by the vision sensor, the installation position of the vision sensor and that of the smart car are relatively static, and the front of the vehicle is the main area for obtaining environmental information. In this paper it is determined to install two vision sensors outside the body of the virtual test smart car model.

4.1 Determination of the coordinates of the virtual smart car

Virtual test smart car model: a 3D cuboid model with a body length of L meters, a width of W meters, and a height of H meters. Defines the position coordinates of the car body relative to the world coordinate system: the geometric center point of the car is the origin of the local coordinate system O (0.0, 0.0, 0.0) of the car, and the X axis is along the transverse center axis of the car body and points to the right side of the direction car going forward. The Y axis is perpendicular to the tangent plane of the car body upwards, and the Z axis is along the longitudinal center axis of the car body, pointing to the direction of the car going backward, as shown in Figure 5:

![Figure 5 Schematic diagram of the position coordinates of the virtual smart car from different perspectives](image)

4.2 Installation of virtual vision sensor

The virtual vision sensor model studied in this paper is fixed on the body of the test smart car model. Two virtual vision sensors are installed on the car body. Their positions and directions are as follows: vision sensor 1 is installed at the upper left of the center point of the model car with the height of H/2 meters, the relative position of S1 (-W/2, H/2, 0.0) and the direction: offset θ along the negative direction of the Z-axis toward the negative direction of the Y-axis; the vision sensor 2 is installed above the center point of the model car with the height of H/2 meters, the position relative coordinate of S2 (0.0, H/2, 0.0) and the direction: offset θ along the negative direction of the Z axis toward the negative direction of the Y axis. As shown in Figure 6:
6. Conclusion
The image acquisition unit and processing unit of the vision sensor are modeled in this paper, which includes vision sensor parameters such as resolution, longitudinal angle, image pincushion, frame rate, and depth. The image pincushion distortion variable is used to simulate the distorted picture taken by the real vision sensor and make the picture taken by the virtual vision sensor closer to the real. The image distortion is realized through the FBO in OpenGL, and the distortion variable can be adjusted according to different requirements.

In order to realize the photography of the vehicle vision sensor, after completing the modeling of the main parameters of the virtual vision sensor, the virtual vision sensor is installed on the smart car. In this paper, the upper end of the center position of the smart car and the left end parallel to the center position of the car are installed 2 vision sensors respectively. A good foundation has been laid for the application of on-board cameras in virtual scenes by installing on-board vision sensors in virtual scenes.

Acknowledgments
This article is one of the phased results of the Xi'an Traffic Engineering Institute's Young and Middle-aged Fund Project "Research on the Reachable Set-based Autonomous Vehicle Control Decision Method", serial number:20KY-14.

References
[1] PanoSim Technologies.(2017)Intelligent Automotive Technology and product development tools [EB/OL].https://wenku.baidu.com/view/8af8a656a55177232f60ddcda38376bae1fe042.html
[2] CIVITEC. Pro-SiVIC Documentation-Version 2.2. (2013). Montigny-le-Bretonneux.M.
[3] Eternal Yang,Y. OpenGL [EB/OL].(2017)https://baike.so.com/doc/4785957-5001950.html.
[4] Liu,X.C.,Ding,Y.H.,(2006) Application of 3D Laser Scanning Technology in Ancient Building Protection J. Surveying Engineering. 15(1):48-49.
[5] Han,F.(2014) Research on Vehicle Lane-changing Assistant System Based on Visual Sensing in Virtual Environment.D.