Virtual Memory Based Radar Display and Control System

Zengshan Tian, Mingxiao Wang(✉), Mu Zhou, and Feng Qiu

Chongqing Key Lab of Mobile Communications Technology,
Chongqing University of Posts and Telecommunications,
Chongqing 400065, People's Republic of China
{tianzs, zhoumu}@cqupt.edu.cn, wangmxl99lll@163.com,
qiufeng245@outlook.com

Abstract. Since the graphics processor of common X86 platform does not support the display function of multi-layer graphics, this paper proposes to combine the Qt Graphics-view framework with OpenGL pixels operation function and utilizes the graphics memory under the Linux system to achieve the layered and efficient display based on the radar information. The proposed approach can effectively realize the main functions of radar display and control system, including the radar Plan Position Indicator (PPI) display, target glint in warning area, and Automatic Identification System (AIS) target management and display. The experimental results prove that the proposed approach has the advantages of high efficiency, smooth image update, and low hardware requirements. In addition, the proposed approach has been successfully applied to a typical shipborne radar navigation system.

Keywords: Radar · PPI display · Qt · OpenGL · AIS

1 Introduction

Since the radar system has the advantages of low environmental requirement and target detection with real-time capacity, high positioning accuracy, and long ranging property, it is indispensable for ship navigation. The main features of modern shipborne navigation radar display systems are the digital information processing, high-performance information display, and friendly interactive experience [1]. Most of the implementation platform is based on the existing System on Chip (SOC) platform or embedded platform by selecting the Advanced RISC Machines (ARM) as the core. As one of the most important parts for human-computer interaction in the radar system, the radar display and control terminal undertakes the fundamental tasks of original radar image display, radar control, and tracking target display. In addition, the modern radar can integrate the radar images with the Global Position System (GPS), Automatic Identification System (AIS), compass, and log to guarantee the well navigation performance.

Since the radar Plan Position Indicator (PPI) display involves a large amount of image data with high refresh rate, the radar display terminal requires high-grade processor and graphics card. Many research institutes mainly focus on the advanced embedded processors combined with specialized embedded graphics processor, which generally has multiple images display layers and 2D/3D graphics acceleration. The

X.-L. Huang (Ed.): MLICOM 2016, LNICST 183, pp. 383–392, 2017.
DOI: 10.1007/978-3-319-52730-7_39

three typical types of software development approaches used for radar display terminal are summarized as follows. (1) Based on the embedded Linux system, the advanced GUI development environment, like the Qt and GTK +, is adopted to develop the display and control interface and render the secondary radar information, and meanwhile the Framebuffer technology is used to draw the radar original image through the direct operation of video memory [2]. (2) The third-party plug-in which supports hardware acceleration, like the OpenGL and DirectFB, is combined with the GUI development environment, like the echo display method based on OpenGL texture mapping [3] and DirectFB library, to write driver for the sake of achieving the image layering and hardware acceleration [4]. And (3) On the X86 platform, the DirectDraw technology based on the Windows system is adopted to achieve the multi-layer image display and acceleration, by using the Qt or VC++ to develop the radar interface. For example, the direct video technology with Visual C++ and DirectX programming method is used to realize the radar image display [5].

Another common approach to develop the radar display and control system is to select the Field Programmable Gate Array (FPGA) as the main controller, with an external Video Graphics Array (VGA) conversion chip and Digital Signal Processor (DSP) chip or ARM processor to process the radar data, and then rely on the FPGA processing of image data fusion to display the radar images [6].

Aiming at the deficiencies of the approaches mentioned before, this paper proposes to combine the Qt Graphics-view framework with OpenGL hardware acceleration to achieve the layered and efficient display for radar information. This approach is based on the X86 platform with low configuration, and does not require the graphics processor to support the multi-layer image display function on hardware. Since the graphics memory, Qt multi-thread, and OpenGL pixel operation interface are used to develop the cross-platform radar display and control software, this approach can effectively compensate the radar display and control terminal development vacancies on the low configure X86 platform based on the Linux system.

2 Rendering of Radar Images

2.1 Qt Graphics-View Framework

The Qt Graphics-view framework in Fig. 1 can manage and display a large number of custom 2D graphic items and interact with each other by using the view component to visualize the graph and support the scaling and rotation. It includes event propagation architecture, with the ability of accurately doubling the precision interaction with the items in scene. The UI controls can be easily embedded in the QGraphicsScene and use the Qt's QSS to separate the interface beautification and software code which can be used to customize many GUI elements.

2.2 OpenGL Pixels Operation Interface

The OpenGL is the interface of developing high-quality image in SGI's graphics workstation and has become a representative 3D graphics interface [7]. The OpenGL

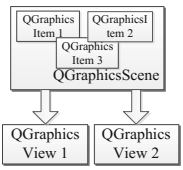

Fig. 1. Qt graphics-view framework.

provides three basic functions of processing image data, namely the frame buffer pixel read function glReadPixels, frame buffer pixel write function glDrawPixels and frame buffer pixel duplicate function glCopyPixels. The glDrawPixels reads a rectangular array of pixels from the processor memory and writing the data to frame buffer at the raster position specified by glRastePos* [8]. During the process of pixel transfer from the memory to frame buffer, the glPiexlZoom is used to set the image scaling and rotation and glPixelStore is used to set the display range of image.

2.3 Radar Image Display

Since most of the Intel integrated graphics on the X86 platform do not support the image layered display, the directly operating frame buffer to realize the radar image display will not benefit the drawing of radar secondary information, like the AIS targets, ARPA targets, and navigation routes. The proposed approach is based on the characteristics of Qt Graphics-view framework which can coexist with the OpenGL by using the image layering method and creating the virtual graphics memory, to realize the raw radar data drawing. The implementation steps are described as follows.

Step 1: Create the QGLWidget in the main window and use the member function makeCurrent to set its RenderContext to be the current value.

Step 2: Create the custom view and scene object which are inherited from the QGraphicsView and QGraphicsScene respectively, and then set the QGLWidget as the window of the custom view object. The background RenderContext is set to be the OpenGLContext, with the purpose of avoiding the CPU rendering and improving the efficiency of drawing.

Step 3: Create the virtual memory in the custom view object by a two-dimensional array, namely radar video buffer in processor memory, which size is $1024 \times 1024 \times 4$ bytes. We need to initialize it with the background color and modify the array to implement the real-time change of radar image. The subsequent drawing of radar image only requires changing the pixel values in virtual memory area, which is similar to the process of the direct operation of memory.

Step 4: In the function drawBackgroud of custom view object, use the QPainter member function beginNativePainting and endNativePainting to make the pure OpenGL rendering, and then use the function InitGL, ResizeGL and PaintGL to execute the OpenGL operations.

Step 5: In the function PaintGL, use the OpenGL function glDrawPixels to draw the virtual memory data into the screen as a background layer.

Step 6: Select the azimuth circle, heading line, AIS targets and tracking targets as QGraphicsItem added to the custom scene object, and then use the function setZValue to set the items' display order.

Our system uses the GL_RGBA format when calling the function glDrawPixels and the data format are GL_UNSIGNED_BYTE because using the same image format and data format with frame buffer can reduce the workload of OpenGL implementation. When zooming in the radar image, we use the function glPixelStore* to set the corresponding parameters in order to capture the image display area, and meanwhile set the scale factor and display position on the screen by the function glPixelZoom and glRasterPos*.

3 Design of Radar Display and Control System

Our system relies on the Linux system to receive the raw radar data which are processed by the FPGA, and then parse and display the data. In addition, it is also responsible for the deployment and control of the entire radar system, secondary information display, recording and managing the AIS targets, and providing a graphical interactive interface.

3.1 Interface Design

Our system combines the GPS, AIS, compass, and log to enhance the function and enrich the display content. The display and control system interface includes the radar display area, information monitoring area, display control menu, function control menu, and install menu. Among them, the information monitoring area overlaps the menu area and is switched by the control button for the sake of ensuring that the interface is concise. The interface is designed in Fig. 2.

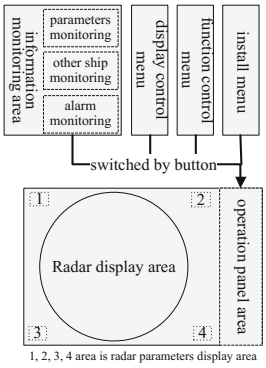

Fig. 2. Radar display and control interface.

The radar display area shows raw radar data, tracking targets, and AIS targets. The parameters monitoring area shows the ship information, like the latitude, longitude, course, and speed, while other ship monitoring area shows the tracking and AIS targets information, like the name or label, Maritime Mobile Communication Service Identifier (MMSI), speed, heading, Time to Closest Point of Approach (TCPA), and Distance to Closest Point of Approach (DCPA). The alarm monitoring area shows all the types of alarm information in a real-time manner. The display control menu contains the color choices, trail display, vector line length, brightness adjustment and adjustment buttons. The function control menu contains the warning area selection, route points, TCPA, DCPA and function setting buttons. The install menu is used for the radar installation and requires the password for entering.

3.2 Radar Video Display

The conventional radar PPI display depends on the afterglow effect of fluorescent material to display the echo signal [9], whereas the modern radar is based on the raster scanner. Since the echo signal is described by polar coordinates, it is required to perform the coordinates-transformation before drawing. Our system combines the coordinates index look-up table method and uniform motion model [1] to exhibit the relations between the raw radar data and drawing positions in the virtual memory [10]. The polar coordinates represent the target position with the distance and azimuth (ρ, θ). The conversion from the polar coordinates into rectangular coordinates (x, y) is shown in (1).

$$\begin{cases} x = \rho \sin\theta \\ y = \rho \cos\theta \end{cases} \tag{1}$$

Where θ is the angle between the target and ship's heading.

The purpose of coordinate index look-up table method is to calculate each point's corresponding quantization angle and display radius based on its two-dimensional index in the virtual memory, and then create the two-dimensional coordinate conversion table. Based on (2), we first create a two-dimensional data list Index$[r][\theta]$ $(0 \le r < MaxRng,\ 0 \le \theta < MaxAzi)$, where *MaxRng* is the maximum display radius and *MaxAzi* is the maximum quantization angle, and then use (3) to calculate each point's index idx and store it into the list Index. When drawing the radar data, we need to look up the table according to its polar coordinates to obtain the corresponding index in the virtual memory, and assign a color value.

$$\begin{cases} r = \sqrt{\left((x_1 - R)^2 + (y_1 - R)^2\right)} \\ \theta = \text{atan}(x_1 - R, y_1 - R) * MaxAzi\ /2\pi \end{cases} \tag{2}$$

where R is the half of the height of screen.

$$idx = y_1 2R + x_1 \tag{3}$$

Since the uniform motion model selects the true motion of radar operation as the off-center relative motion, it is not necessary to calculate the two-dimensional coordinate transformation table when the observing mode is changed. The size of the two-dimensional coordinate index table should be doubled from the size of actual screen by the reason that the radar display range is doubled under the off-center condition. Thus, when the radar is off-center in anywhere, we can obtain the actual screen location of each echo point from the look up table with simple coordinates-transformation.

The radar display area of our system is a square area with the size of 1024×1024 pixels, and the actual PPI display area is a circle with the radius is equaling to 500. After the raw radar data are parsed, we store the range, angle, index, and video data into the structure temEchoData as defined below.

```
Typedef struct temEchoData{
  quint16 range;
  quint16 angle;
  quint16 packetNum;
  quint8 echo[1000];
}EchoData;
```

Since the angle of two adjacent frames echo data are not fixed in FPGA sampling, we use the angle difference of two frames echo data to draw a sector. Our system selects the previous frame echo data as the actual drawing data, and selects the current frame data's angle as the end angle of the sector drawing. In addition, the refresh rate of the image is determined by the division angle means the radar image span of each drawing, and our system choose 6°. When the division angle is large, both the refresh rate and hardware consumption will be low, but the image will not be smooth. When the division angle is little, the image will be smooth, but the hardware consumption will be high.

3.3 Radar Warning Area Alarm

When using the shipborne navigation radar, the warning area alarm is a convenient and practical function, which achieves the automatic detection and alarm in the designated area and reduces the workload of radar operator. We aim to detect whether there are targets in warning area, and make the warning area targets glint when the radar image is in real-time refresh.

Since the radar image display system uses the virtual memory combined with angle dividing, the blinking effect of targets can be achieved by changing target content in the virtual memory between two adjacent refresh of radar images. However, the significant attention should be paid when the target content is changed since the logical relationship between the target content and actual echo data is used to modify the virtual memory.

3.4 AIS Targets Management

The modern common radar is limited by the hardware constrain, and meanwhile the number of automatic tracking target based on the raw radar data is generally less than 100 and error tracking occurs due to the existence of noise. As a new type of digital navigation system, the AIS broadcasts to the nearby ships and coast stations via Very High Frequency(VHF) and enables the nearby ships and coast stations to master the dynamic and static information of ships which greatly increased the accuracy of the target identification and tracking.

The AIS target management mainly includes the AIS target information extraction, target dynamic refresh, alarm detection, display position locating, and quantity management. Considering the characteristics, like the large amount of calculation about the AIS target data, frequent calculation of display position, and stability of refresh time, our system defines two different coordinate system when calculate the AIS target screen coordinates. The first one is the rectangular coordinate system between the AIS target and the corresponding ship, and while the second one is the screen coordinate system between the screen and the corresponding ship.

The AIS target information is processed with fixed time, to create a QHash table to store the parsed data of AIS information with MMSI number as index. During the processing of AIS target information, we refresh the AIS target state, calculate the alarm information relevant to the corresponding ship, update the historical points, and analyze whether the AIS target is lost.

Since the number of AIS targets in the harbor or busy waterway is large, handling all the AIS targets and displaying them on the screen will cause difficulties and waste hardware resources. To solve this problem, we limit the number of AIS targets by removing the long-range AIS targets based on the distance between the AIS targets and the corresponding ships. The steps of AIS target screening is as follows.

Step 1: By assuming that the screened number of AIS targets is k, we create two arrays, namely Rng[k] and Mmsi[k], as the sorting containers.

Step 2: We select k targets from AIS targets storage list and store them into array Rng in ascending order of distance, and then store the MMSI number of the corresponding objects into Mmsi.

Step 3: Based on the AIS targets storage list, if the distance is less than the first element in Rng, we ignore it, and otherwise insert it into Rng with the corresponding position, remove the minimum, and insert the target's MMSI number into Mmsi with the corresponding position. Finally, the MMSI number saved in Mmsi is the AIS targets index to be removed.

4 Test Results

In our testing, the hardware platform is the Celeron 2 GHz CPU, 1 GHz memory, intel integrated graphics and flash 8G hard disk.

The control and display system is developed in Ubuntu12.04 and the test system is TimeSys Linux. The Ethernet cable is selected as the transmission channel between the baseband process board and display and control terminal. The terminal and baseband

process board are separated for the sake of facilitating the miniaturization of radar display terminal.

4.1 Real-Time Performance Testing

The system operation interface is shown in Fig. 3. The yellow and gray images represent the actual target and clutter in the circular display area respectively. The antenna rotation period is 2.5 s and image refresh angle is 6°. The image refreshes 24 times per second with smooth running effect. The number of echo data to be processed reaches the millions level. The system is able to perform coordinate conversion, drawing display, and target detection and trail control, as well as respond quickly to a variety of radar operations.

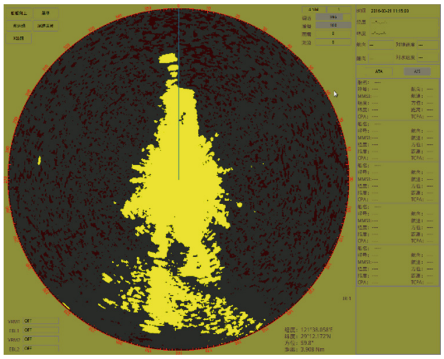

Fig. 3. Operation interface of radar display and control system.

In the Qt environment, the time consumption of rendering the 1024 × 1024 pixels is shown in Table 1. From this table, we can find that the drawing efficiency of OpenGL is higher than other methods and it's fully meet the real-time requirements.

Table 1. Time consumption by different drawing approaches

Approaches	QPainter	QGraphicsItem	OpenGL
Time consumption(ms)	134	625	15

4.2 Alarm Function Testing

The radar warning area function is to detect the targets and glint alarm automatically based on the original echo image. Our system detects the values of virtual memory, and when the warning area appeared targets than glint alarm immediately, and there is no false alarm or leak detection. Figure 4 shows the normal display image after delimiting the warning area surrounded by the white line. Figure 5 shows the image under the condition that the warning area targets are hided. Our system achieves the glint effect of targets by rapidly alternating the image refresh.

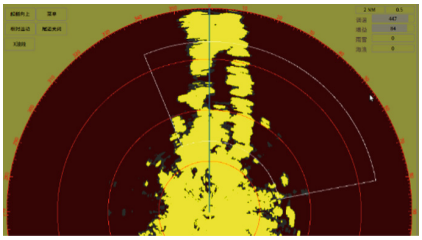

Fig. 4. Normal display images of warning area.

4.3 AIS Targets Screening Testing

In this section, we simulate 10 AIS targets to test the AIS targets management and screening function. The display interfaces before and after screening are shown in Fig. 6. In the figure, the green triangle is the AIS target, while the triangle surrounded by a box is the selected AIS target. When setting the number of AIS display targets as 4, we can find our system reserves four AIS targets which are closest to the corresponding ships. The distances from AIS targets to the corresponding ship before and after scanning are shown in Table 2.

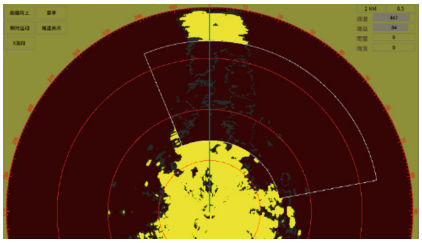

Fig. 5. Hidden images of warning area.

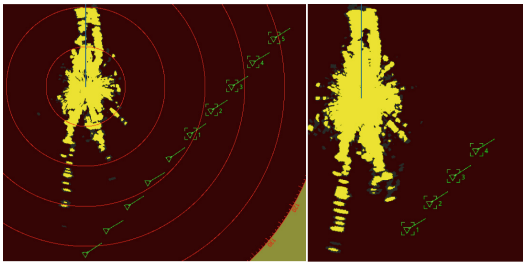

Fig. 6. AIS targets display before and after scanning.

Table 2. Distance from AIS targets and ship before and after scanning

Target IDs	1	2	3	4
Before(Nm)	3.226	3.698	4.267	4.902
After(Nm)	3.176	2.873	2.774	2.901

5 Conclusion

Since the hardware graphics processor on the common X86 platform does not support the multi-layer image display, this paper proposes a new radar image drawing method that selects the graphic memory combined with the Qt Graphics-view framework and OpenGL hardware acceleration function to achieve the real-time radar image display, glint alarm in warning area, and AIS targets management and integrated display. This method does not require the graphics processor to support the multi-layer image display, and meanwhile it has the advantages of cross-platform, low hardware requirements and well display performance. Based on the extensive testing result, this method is proved to be able to work well under the condition that the refresh rate of radar image equals to 36 per second.

Acknowledgments. This work was supported by the Program for Changjiang Scholars and Innovative Research Team in University (IRT1299), National Natural Science Foundation of China (61301126), Special Fund of Chongqing Key Laboratory (CSTC), and Fundamental and Frontier Research Project of Chongqing (cstc2013jcyjA40041, cstc2015jcyjBX0065).

References

1. Wei, B., Guo, Y., Mo, H.: The realization of echo displaying for marine navigation radar under high-rotating speed and multiple operating modes. Sci. Technol. Eng. **13**(17), 4962–4967 (2013)
2. Ren, Q., Yang, J.: Implementation and application of FrameBuffer in radar display. Electron. Sci. Technol. **22**(6), 61–63 (2009)
3. Zhang, P.: A method of radar echo display based on OpenGL. Shipboard Electron. Countermeasure **34**(3), 39–42 (2011)
4. Liang, W.: Research on the DirectFB graphics engine transplant based on BCM7241 platform. Comput. Telecommun. **4**, 53–55 (2015)
5. Wu, W.: Design and implementation of radar control system software on Direct3D under windows. Sci. Technol. Inf. **1**, 88–89 (2014)
6. Ying, S., Zhang, X.: A radar display terminal based on a partial-screen-updating method. In: IEEE International Conference on Embedded Software and Systems Symposia, pp. 28–31 (2008)
7. He, Y., Zhang, G., Zhang, Q.: Design and simulation of radar terminal display based on OpenGL. Informatization Res. **38**(2), 15–18 (2012)
8. OpenGL Architecture Review Board, Dave, S., Mason, W., Jackie, N., Tom, D.: OpenGL Programming Guide(Version 4). Posts & Telecom Press, Bei Jing (2005)
9. Fan, W.: An efficient algorithm for radar PPI display. Mod. Radar **37**(2), 41–45 (2015)
10. Li, B., Liu., D.: Research and realization of coordinate conversion in radar video display. In: Ninth International Conference on Computational Intelligence and Security, pp. 277–279 (2013)