Visual C + + and MFC Application in Safety Monitoring System of Airplane Depot

Wenhua Qiu, Yongliang Zhang and Liang Wang
[1]College of Information Engineering Guangdong Jidian Polytechnic Guangzhou, China
[2]Guangdong post Guangzhou, China

Abstract. In order to avoid airplanes parking in the depot from crashing other moving objects, this paper proposed a method of designing an intelligent security surveillance system for airplane depot. The function of the system is divided into two relatively independent modules. The first part is the monitoring module, is mainly used to monitor the activities of airplane depot and store video; The second part is anti-collision and warning module, mainly used to achieve aircraft anti-collision alarm. In terms of software implementation, we use C/C++ language to implement the algorithm. Making full use of the advantages of multi-threading technology, and optimizing the storage management to make the monitoring system run stably.

Keywords: Visual C++, MFC, multi-threading technology, storage management, monitoring system.

1. Introduction

In recent years, digital technology-based security monitoring systems have been widely used in production and life. In modern intelligent security monitoring systems, moving target tracking is a key technology. Target tracking combines image processing, automatic control, and information science to form a technology that automatically tracks moving targets from image signals in real time. Since the image is projected on the 2-D plane objectively in 3-D, it is a morbid problem in itself, plus the randomness and complexity of the target motion in the actual environment, such as target size, deformation, motion speed, motion track, the degree of similarity between the target color and the background color, the stability of the background, etc., all bring difficulties to the target tracking. Its theoretical and applied technology research is extremely challenging, and many problems and difficulties remain unresolved. Therefore, the study of target tracking in complex contexts not only has important theoretical significance, but also has wide and high practical value.

This paper designs a set of hangar safety monitoring system with certain intelligent features. It is based on the video recording and real-time network preview and control functions of the existing monitoring system. It uses digital image processing technology such as image recognition and moving object detection to realize the function of aircraft anti-collision tracking alarm. Firstly, the color image segmentation method is used to realize the positioning of the first aircraft that has been parked in the hangar; Then, the four main aspects (moving target extraction and target region segmentation, target feature extraction, target description and target tracking process) involved in visual target tracking are studied and discussed to realize the detection and tracking of the second sports aircraft.

In terms of software implementation, this paper uses Visual C++ 6.0 development tools on Microsoft Windows XP operating system platform, and implements the discussed algorithm in C/C++ language. Make full use of the advantages of multi-threading technology, and optimize the storage management to make the monitoring system run stably.

2. Advantages of Hangar Security Monitoring System With Visual C++ and MFC

Visual C++ (hereinafter referred to as VC++) is a new C++ compiler developed by Microsoft Corporation based on Microsoft C/C++. It is a continuation of Microsoft Windows program

development tools. MFC (Microsoft Foundation Classes) encapsulates most of the Windows API functions. Using Visual C++ to implement hangar security system has many advantages:

(1) VC++ supports the ANSI C++ standard, so there are a lot of algorithms written in C/C++ that can be used in the process of writing programs.

(2) Programs written in C/C++ have faster execution speeds than other high-level languages. Algorithms for image processing generally require a large amount of computation, so it is reasonable to choose VC++ as the development platform.

(3) VC++ has a friendly user interface and powerful source management functions. When the program is more complicated and there are many source files, convenient source access is a must.

(4) MFC encapsulates most of the Windows API functions, and the encapsulated functions are much simpler than direct calls.

(5) With the help of MFC's Application Wizard, it is easy to generate an application framework, so that developers do not have to write basic code every time, which greatly improves the efficiency of writing programs [1].

(6) MSDN provides powerful online help. Almost all of the required library functions are detailed in MSDN, and the rich sample code makes it easier for developers to learn how to use various library functions.

(7) VC++ technology is very mature. It is rare for a compiler to fail due to a compiler. Developers can then focus on finding errors in the program code. A powerful debugger also makes it easier to find errors in the program.

3. Multi-threading Technology

Multi-threading technology allows programs to perform different tasks at the same time, making fuller use of CPU resources. Our system needs to respond to operations while performing a large number of image processing operations. This requirement can be easily met by using multi-threading technology [2].

While multithreading brings us convenience, it also has some undesirable consequences. For example, multiple threads may have to read and write to the same data at the same time. When one thread has not finished writing data, another thread may need to modify or read the data, which will cause confusion in data access.

The way to solve this confusion is to use some synchronization objects. MFC encapsulates synchronization classes such as CCirticalSection, CEvent, CMutex, CSenaphore, CSingleLock, and CMultiLock. These synchronization classes can be used to easily synchronize between threads

The best way to create a thread in MFC is to use the CWinThread object encapsulated inside MFC or you can implicitly create a CWinThread object directly using the AfxBeginThread function. After the thread object is created, you can immediately enter our predefined thread control function, or call the CWinThread ResumeThread member function when we need it, and manually start the thread. MFC allows us to create two types of threads, user interface threads and auxiliary thread. In general, we use the main thread automatically generated by the MFC framework to meet the requirements without having to create user interface threads again. Therefore, we only need to create a secondary thread in the alarm program

To end a thread, you can return it through the return statement in the thread control function, or you can use the AfxEndThread function to end the current thread. The return value of the thread can be obtained using the GetExitCodeThread function.

Communication between threads is common. We can do this through global variables and messages. The method of global variables is likely to need to use a variety of synchronization objects to keep threads synchronized. A large number of global variables are used in the software implementation of the alarm module, and a class is created to manage these global variables. It is also necessary to use the method of message communication.

4. Use Messages to Achieve Communication Between Threads

Windows programs are based on event-driven, messaging mechanisms. It is therefore convenient to use messages to communicate between threads. First we need to define a user-defined message and then send the message to the thread or form that needs to be notified.

The object of the form class already has a message loop in itself. So we just need to map the message to a message handler in the form. The secondary thread does not have its own message loop, and the user needs to establish a response mechanism for the message.

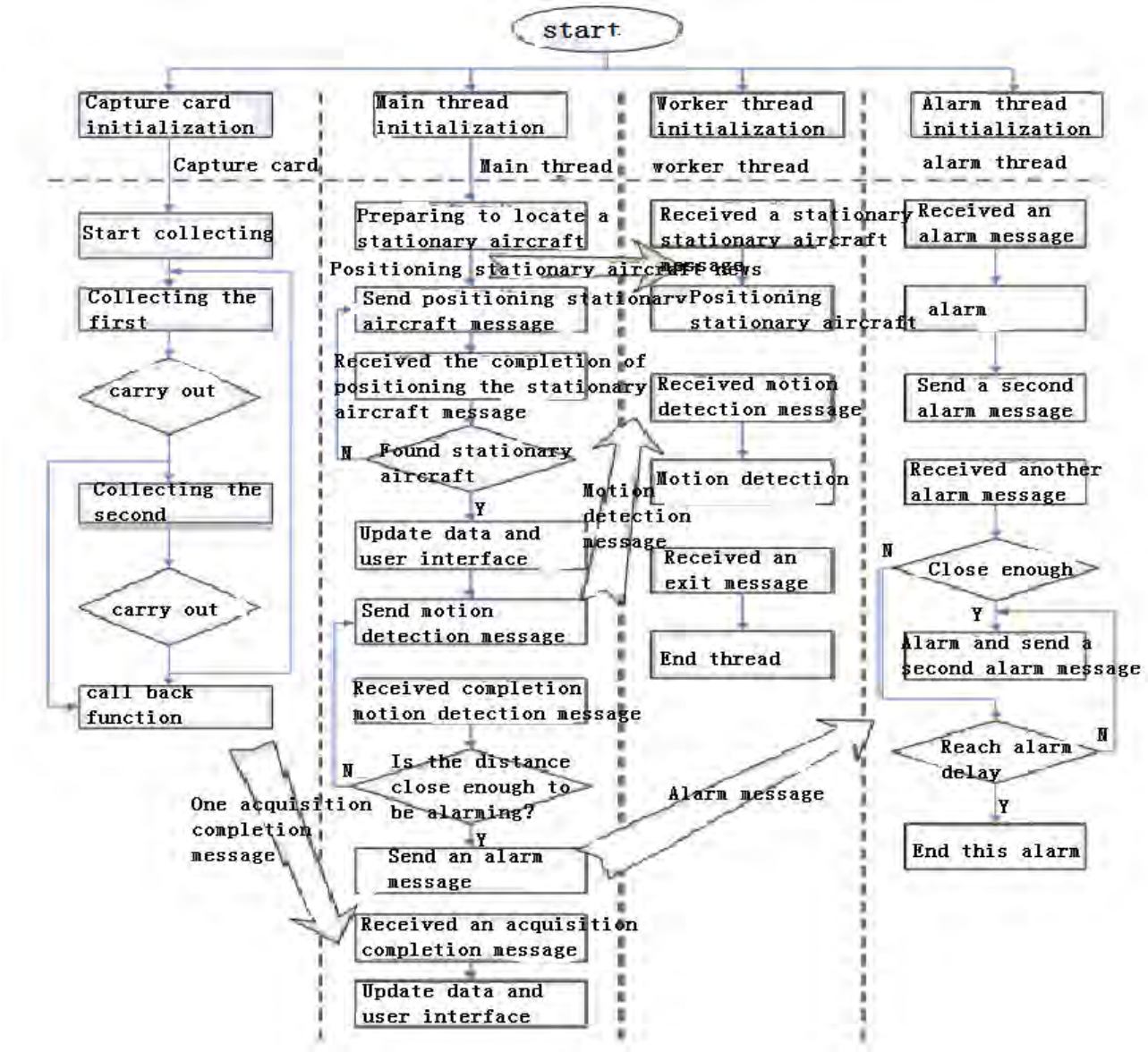
In the alarm module program we create a message loop for each thread that needs to run for a long time. There are some additional benefits to this approach. One of them is that both the creation thread and the end thread need to consume a lot of CPU resources. After the message loop is established, only when the thread receives a request message from another thread that performs some kind of operation, the thread starts executing, otherwise the thread is in a wait state.

This saves considerable CPU resources for image processing operations [3].

Common functions for thread communication are:

(1) PostMessage
(2) SendMessage
(3) GetMessage
(4) PeekMessage
(5) PostThreadMessage

PostMessage does not wait for the sent message to be processed and returns. In contrast, SendMessage waits until the sent message is processed before executing the following code. GetMessage accepts the message and removes the message from the message queue, and waits for the message queue to be empty. PeekMessage does not remove messages by default, so it is often used to detect messages. PostThreadMessage is used to send a message to a thread.



5. Storage Management Optimization

Real-time image processing requires constant updating of live images. The amount of temporary data generated during processing is also very large. Constantly allocating and releasing memory not only requires a lot of CPU processing time, but also easily leads to memory leaks due to inadvertence in code writing. The solution adopted here is to use a fixed buffer. That is to say, for each data block that needs to appear frequently and must occupy the storage space independently, an appropriate size memory space is allocated at the beginning of the program creation. At the same time, such access is more conducive to synchronization between threads.

6. Program Thread Settings

Each program will have a thread at the beginning of its creation. In fact, the application object itself in MFC is derived from the CWinThread object. We call this thread the main thread. In response to user actions, user interface updates and some computationally intensive processing can be handled by the main thread. Since our program needs to respond to the user's operation at any time, if a large number of operations are put into the main thread, it will inevitably cause the user's operation to be unable to get a response for a long time, and the program will appear white screen or unstable. It is

therefore necessary to create a secondary thread to handle these complex tasks. We call this thread a worker thread.

By positioning the stationary aircraft, the task of motion detection becomes the task that the worker thread needs to accomplish. After the program is initialized, the main thread sends a message to locate the stationary aircraft to the worker thread. The worker thread receives the message and begins executing instructions to locate the stationary aircraft. After the positioning is completed, the result is fed back to the main thread. Based on the results of the positioning, the main thread asks the worker thread to continue to find a stationary aircraft or start motion detection to find large objects (including the second aircraft) that are close to the stationary aircraft. The main thread receives the result of the motion detection feedback (also the message of the worker thread). If it is determined that a large object is close to the stationary aircraft and the distance between the two reaches a certain range, an alarm request is issued. At this time, the worker thread must continue to perform motion detection to monitor the motion of the moving object. The task of the alarm is to play a sound file to draw the attention of the staff. In order to ensure that the staff can hear the sound of the alarm, the playing time must be a certain length. At this time, the worker thread will continue to monitor the situation of the moving object, and it is impossible to do the work of playing the sound. The main thread must respond to the user's operation, and need to judge the distance between the moving object and the stationary aircraft based on the result of the motion detection, and can not perform the operation of the alarm. Therefore, we also need to create another alarm thread, which is specially used for alarm work.

The image data in the system is acquired by the capture card to capture the data captured by the camera. These data need to be constantly updated (25 frames per second) to display the collected data to the user interface through the monitor in time, while also providing the latest live image for image processing. The acquisition card does not need to occupy the CPU resources of the system host during the collection process. Obtaining the data collected by the acquisition card can be obtained through the interface function provided by the acquisition card supplier. In order to improve the collection efficiency of the acquisition card, the system adopts a circular buffer to control the acquisition data collected by the acquisition card. Each time a picture is captured, the capture card enters the callback function. The callback function sends a message to collect a message completed by the picture to the main thread, and starts the acquisition of the next picture. Therefore, the process of collecting data and updating data by the capture card can be regarded as a thread of the program.

The threads constantly cooperate and communicate with each other to form a coordinated mechanism throughout the program. Of course, it is somewhat inadequate to pass data between messages only through threads. Such as large amounts of image data and many public data that needs to be accessed in multiple threads may need to control access by certain threads. So we created a data management class to manage almost all global variables. Figure 1 shows the main work and part of the message communication lines of each thread inside the program. It should be noted that this flowchart does not include all the work of each thread. The message sent to the main thread by the work is contained inside each function and does not appear explicitly in the diagram.

7. Test of Program Operation

In the state where the laboratory adjusts the parameters to fit the aircraft model, the program can be used to sound an alarm sound by using an aircraft model close to another stationary aircraft model. The program runs stably during the test.

8. Conclusions and Prospects

By reading a large number of domestic and foreign literatures and materials, according to the functional requirements of the hangar security system alarm module, the identification algorithm of the stationary aircraft and the motion detection algorithm of large objects are designed, and the program is written by Visual C++. The function of aircraft anti-collision alarm in the hangar is realized on the software.

In the process of algorithm design, the author tried different methods many times. After summarizing the advantages and disadvantages of various methods and doing a lot of analysis on the data generated by the algorithm, some optimizations were made to the existing algorithms, and a solution suitable for this system is proposed.

In the process of programming, I read a lot of programming technical data that I need to cover, and strive to leave more CPU resources for image processing work, in order to obtain higher real-time performance. At the same time, in terms of storage management, this paper also considers static cache space as much as possible, so that the program can get faster speed and higher stability.

However, some algorithms in the program still have some shortcomings. For example, the detection and positioning of a stationary aircraft still takes a long time, which affects the real-time performance of the system to some extent. Further research is needed in these areas.

Acknowledgements

This article is one of the phased results of 2016 Guangdong Province Higher Vocational Education Professional Leadership Project, 2017 provincial key platform and major scientific research project in Guangdong Province - the characteristic innovation project (education and scientific research) "The research and practice of the artisan spirit leading the new era application of electronic technology professionals training model".（2017GGXJK006）

References

[1]. Hou Junjie. Exploring MFC [M]. www.jjhou.com

[2]. He Bin. Visual C++ Digital Image Processing. Beijing: People's University of Posts and Telecommunications Press, 2002

[3]. Zhang Honglin. Visual C++ digital image pattern recognition technology and engineering practice. Beijing: People's Posts and Telecommunications Press, 2003.