OCR Software Architecture for Embedded Device

Seho Kim' , Jaehwa Park

Computer Science, Chung-Ang University, Seoul, Korea
seho@hil.cau.ac.kr, jaehwa@cau.ac.kr

Abstract. Character recognition is the field in which studies have been conducted for a few decades. It is applied to commercialized products in the market to be used in recognition of generalized mark-up languages in print letters. However, it is used in limited areas such as electronic document recognition and mail recognition in cursive script. Considering supplies of digital camera-embedded device being currently popularized in a rapid speed, social demands on the character recognition of camera-embedded device are expected to increase.

The embedded device generally uses a central processing unit with relatively lower calculation capability and smaller volume of memory places compared with the personal computer. Nonetheless, problems in the performance speed can easily occur. The reusability of software can also be unexpected if character recognizer for embedded device is realized in a similar software method for an existing personal computer. The embedded software is difficult to be repaired and reused because of too many connection loops or due to different supporting methods applied for each platform resource. Therefore, it is necessary to design systematic software architectures that enhance reusability and portability [1] [2].

Currently, certain types of architectures being widely used are represented by the pipe and filter, data abstract pictures, object-oriented organization styles, event-based implicit invocation style, and hierarchical types. Among

those, the hierarchical types can minimize complexity by dividing matters into gradational steps and enable interactions in each hierarchy and information transmitted in both directions. Moreover, the hierarchical types provide high quality of abstract pictures, maintain independence from system, and allow possible reuse [3].

This paper now designs software architecture for optical character recognizer to reuse on different mobile telecommunication platforms based on the type of hierarchical architecture type. This paper aims to enable developers to apply and reuse easily without depending on the platforms. This paper divides into four hierarchies in the architecture for the optical character recognizer in the embedded platform environment: the platform-based support, interface, engine support, and engine hierarchies. The engine support hierarchy separates stored data from the recognition system to cope with the different memory storage method (Endian) in each platform while using the data required for the optical character recognizer. This hierarchy adopts plug-in data structures in its design for the system to renew effectively regardless of data changes. The real type data are altered to the integer type to and used as a method to improve the performance of the recognizer, and the linear approximation method is used to remove the library dependence.

Large weights should be distributed in designing architecture to develop software in which maintenance and repairs are important. That is because the costs for the reuse and maintenance and repairs of the developed system vary depending on the architectural design. Specifically, the embedded platform generally uses the central processing unit with relatively lower calculation capability and smaller volume of memory places. As compared with the personal computer, problems in the performance speed can easily occur and reusability of software may not be anticipated if the character recognizer for the embedded device is realized using a similar software method with the existing personal computer. Reusing the software with the embedded software is difficult to be repaired in most cases because of too much dependence between modules, or it becomes impossible due to the different methods used for each platform resource.

Software architectures proposed in this paper which aims to improve the aspect of reusability in the development of optical character recognition system of the mobile telecommunication device environment.

This architecture is designed based on the style of hierarchical architecture. The architecture is split into four units of hierarchies: the platform-based support, interface, engine support, and engine hierarchies from the lower level.

The platform-based support facilitates expansion by maintaining independence from the character recognizer for memory retention and clearance, image loading and clearance, and diversity of data producing dependent segments at each platform whenever it changes. The interface completely isolates the user from issues on how the objects can be summoned, how the data can be stored, and how these aspects are realized to prevent misuse and damages. This hierarchy supplies the four functions required to realize encapsulation for the improvement of the independence and portability. The engine support hierarchy solves the issues accrued by each platform from different memory storage methods (Endian) while reading data to the memory in the case the character recognition system can store and use data for recognition. The hierarchy removes the dependence of each platform by adaptively reading data according to the memory storage method of the platform, as it facilitates updates and expansions. Lastly, the engine hierarchy, where the character recognition is being conducted, separates the common segments and removes the dependent segments on the platform. This hierarchy facilitates the expansions in the modification of new functions and in the removal of additions by modularizing as each independent unit.

The recognizer materialized based on the neural network. Performance tests were conducted based on the camera images comparing with the personal computer character recognizer developed by ETRI, Korea. The recognizer was transplanted in the mobile telecommunication device environments such as Korea's standard platform WIPI, PDA GUI Qt, MS PocketPC, Windows Phone 7, Android, and iPhone to test the facilitation of reuse. The WIPI, PocketPC, Windows Phone 7, Android and iPhone used the *Little Endian* method. The Qt board used the *Big Endian.*

The same recognition results were obtained in the experimental results with six platforms like those of the existing personal computer environment developed by ETRI, Korea and the recognition performance was improved by converting the real type data into the integer type. Just modifying the dependent segments of the optical character recognizer enabled more than 98.55% of the entire codes reused in the environments. These are the embedded device platform environments different from each other and supporting other memory storage methods.

In conclusion, this paper designed the software architecture to reuse the character recognizer at the embedded device platforms. The proposed software architecture was designed as a hierarchal architecture that consists of four hierarchies: the platform-based support, the interface, the engine support, and the engine hierarchies. The engine support hierarchy was designed with the plug-in data structures to respond adaptively to other memory storage methods by each platform in using the data in the character recognizer. The character recognizer was applied with the linear approximation method to remove the dependence of library.

The experiment converted the existing character recognizer developed by

ETRI, Korea based on the proposed architecture in this paper and used the neural network as a sorter for the recognition. The converted character recognizer was experimented in various embedded device The experimental result confirmed that more than 98% of the codes for character recognition software were reusable while all six platforms obtained the same recognition results of enhanced performances.

Keywords: OCR, Software Architecture, Embedded Device

References

[I] R. Kazman, G. Abown, L. Bass and P. Clements, "Scenario-Based Analysis of Software Architecture," IEEE Software, vol.13, no.6, pp.45-55, November 1996.

[2] M. Shaw, "Architectural Issues in Software Reuse : It's Not Just the Functionality, It's the Packaging," IEEE Symposium on Software Reuse, New York, USA, pp.3-6, 1995.

[3] D. Garlan and M. Shaw, An Introduction to Software Architecture, World Scientific Publishing, Singarpore, 1993.