무선 이동 통신 기기용 휴먼인터페이스 소프트웨어

(Human Interface Software for Wireless and Mobile Devices)

김 세 호 † 이 찬 근 ††
(Seho Kim) (Chan-gun Lee)

요 약 최근 카메라가 장착된 이동통신기기 환경에서 사용자로부터 정보의 입력을 위한 문자 인식 기술이 많이 요구되고 있다. 일반적으로 임베디드 환경에서 개발된 광학 문자 인식기(Optical Character Recognizer) 모듈은 특정 플랫폼에 대한 의존성 때문에 재사용하기 어렵다. 본 논문은 다양한 이동통신기기 환경에 쉽게 적용될 수 있는 광학 문자 인식기를 위한 소프트웨어 아키텍처를 제안한다. 제안된 아키텍처는 플랫폼 의존 지원계층, 인터페이스 계층, 엔진 지원계층 그리고 엔진 계층으로 나뉜다. 엔진 지원계층에서는 다양한 하드웨어 엔디안 정책에 대응하기 위해 플러그인 데이터 구조를 지원한다. 제안된 아키텍처의 실제적인 적용을 통해 제안된 방법의 실효성을 보인다.

키워드 : 이동통신기기, 문자 인식, 재사용, 소프트웨어 아키텍처

Abstract Recently, the character recognization technique is strongly needed to enable the mobile communication devices with cameras to gather input information from the users. In general, it is not easy to reuse a CBOCR(Camera Based Optical Character Recognizer) module because of its dependency on a specific platform. In this paper, we propose a software architecture for CBOCR module providing the easy adaptability to various mobile communication platforms. The proposed architecture is composed of the platform dependency support layer, the interface layer, the engine support layer, and the engine layer. The engine layer adopts a plug-in data structure to support various hardware endian policies. We show the effectiveness of the proposed method by applying the architecture to a practical product.

Key words : mobile communication devices, character recognition, reuse, software architecture

1. 서 론

문자 인식은 지난 수십 년간 많이 연구되어온 분야로, 현재에는 인쇄체 문자의 경우 범용 문서의 인식에 사용되는 상용 제품이 출시되어 사용되고 있으며, 필기체 문자의 경우 전장표 인식, 우편물 인식 등 제한적인 인식 분야에서 문자 인식 기술이 사용되고 있다. 더불어 현재 빠른 속도로 대중화되고 있는 디지털 카메라를 장착한 이동통신기기의 보급에 미루어 볼 때 카메라 기반의 이동통신 기기 상에서 문자 인식에 대한 사회적 요구는 증가될 것으로 예측 된다.

하지만, 이동통신기기의 경우, 일반적으로 개인용 컴퓨터에 비해 상대적으로 연산 능력이 떨어지는 중앙처리장치와 적은 양의 기억장소를 사용하기 때문에 이동통신기기용 문자 인식기를 기존의 개인용 컴퓨터와 비슷한 소프트웨어 방법으로 구현할 경우 수행 속도면에서 문제점이 생기기 쉽고, 소프트웨어 재사용성을 기대하기 어렵다. 임베디드 소프트웨어들의 경우 재사용 하려 할 때 연관 고리가 너무 많아 고쳐 쓰기 어렵거나, 각 플랫폼의 자원을 이용하기위한 지원 방법이 달라 재사용이 불가능 한 경우가 많다. 따라서 다양한 플랫폼에

· 이 논문은 2007년도 중앙대학교 학술연구비(일반연구비) 지원에 의한 것임을 밝힙니다.

† 학생회원 : 중앙대학교 컴퓨터공학부
seho@hil.cau.ac.kr
†† 정 회 원 : 중앙대학교 컴퓨터공학부 교수
cglee@cau.ac.kr
(Corresponding author임)

논문접수 : 2009년 12월 2일
심사완료 : 2009년 12월 8일

정보과학회논문지: 정보통신 제37권 제1호(2010.2)

서의 재사용과 이식성을 높이기 위해서는 체계적 소프트웨어 아키텍처를 설계할 필요가 있다[1,2].

현재, 일반적으로 넓게 쓰이고 있는 대표적인 아키텍처 유형으로 파이프와 필터, 데이터 추상화와 객체 지향 조직 스타일, 이벤트 기반 암시적 호출 스타일, 계층 형 유형등이 있다. 여기서 계층 형 유형의 경우, 복잡한 문제를 점차적인 스텝으로 나누어 복잡성을 최소화 시킬 수 있는 장점 있으며, 각 계층 사이에 훨씬 다양한 상호작용을 가능하게 하고 정보의 양방향 전송을 가능하게 한다. 또한 높은 추상화 제공과 시스템과의 독립성을 유지시키고, 재사용이 가능하게 해 준다[3].

본 논문에서는 계층 형 아키텍처 유형을 기반으로 다양한 이동통신 플랫폼 상에서 재사용을 위한 광학 문자 인식기를 위한 소프트웨어 아키텍처를 설계함으로써, 개발자들이 쉽게 적용하여 더 이상 플랫폼에 의존적이지 않고, 재사용이 가능하도록 한다. 본 논문에서는 이동통신 플랫폼 환경에서 광학 문자 인식기를 위한 아키텍처를 각각 플랫폼 의존 지원계층, 인터페이스 계층, 엔진 지원 계층 그리고 엔진 계층으로 크게 4 계층으로 나누었다. 또한, 엔진 지원 계층에서는 광학 문자 인식기에서 필요한 데이터 사용 시 플랫폼마다 다른 메모리 저장 방식(Endian)에 대응하도록 저장된 데이터를 인식 시스템과 분리시켜, 데이터들의 변화에 관계없이 효율적으로 시스템 갱신이 일어나도록 플러그-인(Plug-In) 데이터 구조를 설계하였다. 그리고 인식기의 성능을 향상시키기 위한 방안으로 실수형을 정수형으로 변환하여 사용하였으며, 라이브러리의 의존성을 없애기 위한 선형 근사 방법을 사용하였다.

실험은 기존의 개인용 컴퓨터용으로 ETRI에서 개발된 카메라기반 문자 인식기를 제안된 소프트웨어 아키텍처에 기반 하여 재작성하고 이것을 바탕으로 이동통신 한국 표준 플랫폼 WIPI(Wireless Internet For Interoperability)과 노르웨이 트롤텍 사에서 개발한 PDA GUI Qt, 그리고 PDA용 MS PocketPC에 이식하여 재사용 여부를 실험 하였다.

논문은 총 5장으로 구성되어 있다. 2장에서는 소프트웨어 아키텍처와 광학 문자 인식기의 기본 개념을, 3장에서는 제안한 소프트웨어 아키텍처를, 4장에서는 이식되어진 WIPI와 Qt 그리고 PocketPC 플랫폼에, 카메라 영상을 바탕으로 실험 결과를 살펴본다. 마지막으로 실험 결과를 바탕으로 5장에서 분석하고 결론을 도출한다.

2. 배 경

2.1 소프트웨어 아키텍처

소프트웨어 아키텍처는 새로운 응용 개발 비용을 감소시키고, 서로 밀접하게 관련된 다양한 소프트웨어 제품군 간의 일반성을 향상시킬 수 있는 방법을 제공한다. 다시 말해, 소프트웨어 아키텍처의 목표는 재사용을 목적으로 하는 컴포넌트와 연결자들을 나타내는 높은 수준의 구조를 제공하는데 있다[3-6].

소프트웨어 아키텍처가 해결하고자 하는 문제는 크게 2가지로 나눌 수 있는데, 그것은 소프트웨어 복잡도 해결의 문제와 소프트웨어의 품질 속성 획득의 문제이다. 따라서 소프트웨어 개발에서 정형화된 소프트웨어 아키텍처를 사용하게 되면 시스템 설계에서 고수준의 추상화가 가능하여 복잡한 시스템에 대한 이해를 돕고 다중 레벨 재사용이 가능하다. 그리고 시스템의 진화 예측이 가능하고, 소프트웨어 유지 보수자로 하여금 변화를 보다 쉽게 이해할 수 있도록 함으로써 수정에 대한 비용을 정확하게 측정할 수 있게 한다. 또한 시스템 구현 시 소프트웨어 시스템의 기본적인 요구사항과 기대되는 발전 방향에 대한 시뮬레이션을 통한 요구사항과 발전 범위의 축적을 통해 소프트웨어 개발에 있어서 위험성을 감소시킨다.

하지만, 아직 소프트웨어 아키텍처에 대해서는 보편화된 정의가 없으며 다양한 정의가 존재한다. 표 1은 소프트웨어 아키텍처에 관한 다양한 정의를 보여주고 있다. 그리고 소프트웨어 아키텍처는 각 목적에 따라 다양한 유형이 존재하는데, 그 중, 대표적으로 쓰이고 있는 소프트웨어 아키텍처 유형을 표 2에 나타냈다. 표 2에서 볼 수 있듯이, 각 소프트웨어 아키텍처는 각 유형에 따라 트레이드-오프(trade-off)가 존재한다[3].

여기서 계층형 아키텍처 스타일의 경우, 복잡한 문제를 점차적인 스텝으로 나누어 복잡성을 최소화시킬 수 있는 장점 있다. 그리고 각 계층 사이에 다양한 상호작용을 가능하게 하고 정보의 양방향 전송을 가능하게 하며, 높은 추상화 제공과 시스템과의 독립성을 유지시키

표 1 소프트웨어 아키텍처의 다양한 정의

Perry, Wolf [7]	요소, 형태, 이론적 근거로 결정되는 특정한 형태를 가지는 아키텍처 요소의 집합으로 정의
Garlan, Perry [8]	프로그램/시스템을 구성하는 컴포넌트의 구조, 상호관계, 그리고 설계와 변경을 결정하는 원칙과 지침으로 정의
Bass, Clements, Kazman [9]	소프트웨어 컴포넌트로 구성되는 시스템의 구조, 이러한 컴포넌트의 외부적으로 볼 수 있는 속성과 컴포넌트 사이의 관계로 정의
IEEE Std1471-2000 [10]	컴포넌트로 구체화되는 시스템의 기본적인 구성, 컴포넌트 서로간의 관계와 설계, 변경을 통제하는 원칙이라고 정의

표 2 소프트웨어 아키텍처 스타일 [3]

아키텍처 스타일	설명	장점	단점
파이프와 필터	각 컴포넌트는 입력 집합과 출력 집합을 갖는다.	- 시스템 행동을 쉽게 이해 - 재사용이 쉽다. - 유지 보수가 쉽다.	- 대화식 프로그램을 다루기 어렵다. - 연관이 있는 흐름을 유지하기 어렵다.
데이터 추상화와 객체 지향 조직	데이터와 오퍼레이션이 캡슐화되어 있다.	- 변경 시 클라이언트에 영향을 미치지 않는다.	- 객체 서로에 대해 알고 있어야 한다.
이벤트 기반, 암시적 호출	컴포넌트가 하나 혹은 그 이상의 이벤트를 전할 수 있다.	- 컴포넌트는 다른 컴포넌트에 영향을 미치지 않고 진화될 수 있다.	- 이벤트 시스템이 상호작용을 위해 공유자원에 접근할 경우 전체 성능 또는 리소스 관리 문제 발생
계층형 시스템	각 계층은 하위와 상위 계층에 서비스를 제공하는 수직 구조	- 높은 추상화 - 높은 성능 향상 - 높은 재사용성	- 커플링이 나타날 수 있다. - 추상 레벨을 제대로 잡기가 힘들다.

고, 재사용이 가능하게 해 준다.

2.2 광학 문자 인식기

일반적인 광학 문자 인식기는 그림 1과 같이 스캐너, 카메라 등 광학적 디지타이저에 의해 이미지 파일로 저장되어진 영상을 입력받아 전처리 과정, 영역 분할 과정, 특징 추출 과정, 인식 과정, 후처리 과정을 거쳐 인식되어진 문자들의 집합을 결과로 얻게 되는 구조를 가진다.

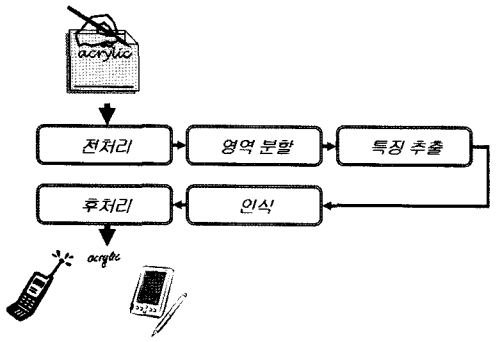

그림 1 일반적인 광학 문자 인식 과정

전처리 과정은 입력 장치의 특성으로 인하여 발생하는 잡음의 분석, 입력 영상 처리 시에 발생하는 정보의 손실 그리고 변형 형태에 따라 저역 통과 필터나 침식, 팽창 연산 등을 이용하여 일차적으로 각종 잡음 성분을 제거 및 위치, 기울어짐에 따른 보정을 하고 각 문서 영상을 인식 하고자 하는 문자 정보에 대한 부분과 그림 영역 등의 배경 부분으로 구분하여 이진 영상으로 변환하는 과정이다[11-15].

영역 분할 과정은 전처리 과정의 출력 영상을 바탕으로 문서를 구성하는 단위, 즉 문단, 줄, 단어, 문자 영역으로 분리 하여 인식 시스템이 처리할 수 있는 인식 단위의 영상 정보로 나누어, 다음 단에 연결되는 인식 시스템과 연결 될 수 있도록 정확한 인식 단위 영역을 찾아내는 과정이다. 즉, 이진화된 영상 정보를 이용하여 문단, 선, 단어, 문자 영역을 인식기와 연결 될 수 있도록 정확한 인식 단위 영역을 찾아내는 과정이다[16-18].

특징 추출 단계에서는 영역 분할 과정을 거친 문자 영상으로부터 정규화된 영상에서 특징 추출 알고리즘을 통해 문자를 나타내는 특징 점들을 추출한다. 즉 입력 영상의 구조적인 특징 및 화소 특징 등을 추출하여 수치화한다[19].

인식 과정은 분할된 문자 혹은 단어 단위로 추출 되어진 특징들을 바탕으로 가장 비슷한 결과를 다양한 인식 시스템을 통하여 찾아 문자 코드 또는 코드열의 형태로 변환 한다. 문자 인식에 주로 이용되어 지고 있는 인식 시스템은 은닉 마르코프 모델(Hidden Marcov Model), 신경망(Neural Network) 등이 있다[20]. 후처리 과정은 오인식되어진 정보를 원 영상의 정보와 일치시킴으로 인해서 인식 오류를 바로 잡기 위한 방법이다[21].

3. 소프트웨어 아키텍처

유지보수가 중요한 소프트웨어 개발 시에 아키텍처 설계에 큰 비중을 두고 해야 한다. 그 이유는 어떻게 아키텍처를 설계하느냐에 따라 개발된 시스템의 재사용성 및 유지보수에 들어가는 비용이 달라지기 때문이다. 특히, 이동통신 플랫폼의 경우, 일반적으로 개인용 컴퓨터에 비해 상대적으로 연산 능력이 떨어지는 중앙처리장치와 적은 양의 기억장소를 사용하기 때문에 이동통신 플랫폼용 문자 인식기를 기존의 개인용 컴퓨터와 비슷한 소프트웨어 구현 방법으로 구현할 경우 실시간 처리에 문제가 생기게 되며, 소프트웨어 재사용성은 기대하기 어렵다. 대개 임베디드 소프트웨어의 경우 재사용 하려고 하면, 모듈간의 의존성이 너무 많아 고쳐 쓰기 어렵거나, 각 플랫폼의 자원을 이용하기 위한 방법이 달라 재사용이 어려운 경우가 많다.

본 논문은 이러한 이동 통신기기 환경에서 광학 문자 인식 시스템 개발 시 재사용성을 높일 수 있는 그림 2에 제시된 소프트웨어 아키텍처를 제안한다.

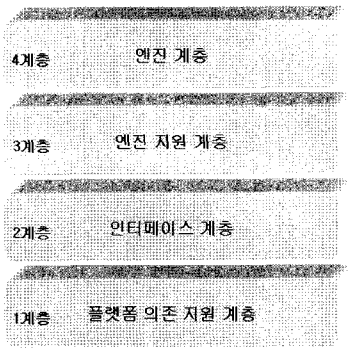

그림 2 CBOCR 소프트웨어를 위한 소프트웨어 아키텍처

이 아키텍처는 계층형 아키텍처 유형을 기반으로 설계 되었다. 그리고 설계한 아키텍처는 4개의 계층으로 나누었으며 하위부터 각각 플랫폼 의존 지원 계층, 인터페이스 계층, 엔진 지원 계층 그리고 엔진 계층으로 나누었다.

첫 번째 계층은 플랫폼이 변경될 때마다 각 플랫폼에 의존적인 부분이 생기게 되는, 메모리 확보 및 해제, 이미지 로딩 및 해제, 데이터 타입의 다양성을 문자 인식기와 독립성을 유지시켜 확장이 용이 하도록 한 계층이다. 두 번째 계층인 인터페이스 계층은 호출하는 사용자에게 객체가 어떻게 호출되고, 데이터가 어떻게 저장되며, 어떻게 구현되어 있는가 하는 문제로부터 완전히 격리시켜 정보의 오용과 손상을 방지하고, 독립성 및 이식성을 증가시키기 위한 캡슐화 구현을 위한 계층으로 필요한 4개의 함수만을 제공한다. 세 번째 계층은 엔진 지원 계층으로, 문자 인식 시스템 에서는 인식을 위해 데이터를 저장하여 사용하는 경우, 데이터를 메모리상으로 읽어올 때 플랫폼 마다 다른 메모리 저장 방식(Endian)으로 발생하는 문제점을 해결하기 위한 계층이다. 즉, 플랫폼의 메모리 저장 방식에 따라 적응적으로 데이터를 읽어 들여 각 플랫폼에 따른 의존성을 없애고, 업데이트와 확장이 가능하도록 하였다. 마지막으로 문자 인식 엔진 계층은 실제적인 문자 인식이 이루어지는 계층으로, 공통부분을 분리하고 플랫폼에 의존적인 부분을 제거 하였으며, 각 독립적인 단위로 모듈화 하여 새로운 기능의 수정, 추가, 삭제 시 확장이 가능하도록 구성된 계층이다.

3.1 플랫폼 의존 지원 계층

플랫폼 의존 지원 계층은 최하위 계층으로서, 문자 인식기를 다양한 플랫폼으로 부터 독립성 및 확장성을 유지시켜 주기 위한 계층이다. 이는 문자 인식기에서 각 플랫폼에 의존적인 메모리 확보 및 해제, 이미지 로딩 및 해제, 데이터 타입의 다양성에 대한 부분을 따로 분리하여 정의한 계층이다.

- 메모리 관리 : 일반적으로 메모리 할당 및 해제 방법은 각 플랫폼마다 다르게 제공된다. 예를 들어, WIPI의 경우 MC_knlAlloc, MC_knlCalloc과 MC_knlFree 함수를 통하여 메모리를 할당하고 해제하지만, Qt의 경우 new와 delete를 사용하여 할당하고 해제한다. 따라서 기존의 광학 문자 인식기의 경우, 플랫폼이 바뀔 때마다 각 플랫폼 방식에 맞춰서 메모리 관리 관련 코드를 모두 찾아 수정해야했다. 물론, 파일의 수가 적은 경우 적은 비용으로 수정이 가능하지만, 파일의 수가 많아지고, 규모가 커질수록 전체를 일관성 있게 수정하는 것이 쉽지 않다. 하지만, 메모리 관리 부분을 따로 분리하여 플랫폼 변경 시 전체 시스템을 수정 하지 않고, 따로 분리한 메모리 정의파일 일부분만을 변경하여 전체 시스템이 수정되는 효과를 얻을 수 있다.
- 이미지 관리 : 이동통신 플랫폼에서의 광학 문자 인식기는 대부분 카메라를 통해 얻어진 문서 영상을 바탕으로 처리된다. 이때, 카메라를 통해 얻어진 문서 영상을 로딩하고 해제 하는 방식은 플랫폼 마다 다르게 제공된다. 예를 들어, WIPI의 경우 MC_grpDrawImage 함수를 이용하여 이미지를 로딩시키지만, Qt의 경우 QImage클래스의 load 함수를 통하여 이미지를 로딩 시킨다. 따라서 이미지 관리 부분을 따로 분리하여, 표 3과 같이 문자 인식 시스템에서 필요로 하는 이미지 정보만을 제공하고, 문자 인식 시스템으로부터 출력된 인식 결과만을 입력으로 해야 한다. 즉, 플랫폼에 의존적인 이미지 관리 부분을 시스템과 분리하여 플랫폼 변경 시 적응적으로 대응할 수 있도록 의존성을 없애야 한다. 그림 3에서는 문자 인식기와 이미지 관리 모듈을 분리한 그림이다. 이미지 관리 모듈은 현재 로딩한 이미지 정보만을 출력 값으로 하고, 문자 인식 시스템의 출력 값인 인식 결과를 입력으로 하고 있다.

표 3 입력 이미지 정보

이미지 정보
픽셀 값
가로 축 크기
세로 축 크기

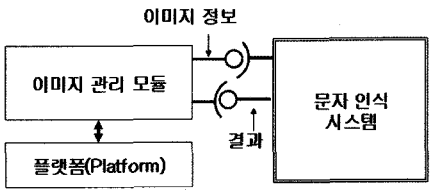

그림 3 이미지 관리

• 독립적 데이터 타입 정의 : 동일 데이터 타입에 대해서 플랫폼 마다 실제 소비하는 데이타의 크기가 다를 수 있다. 또한 유사한 데이터 타입이 플랫폼에 따라 다른 이름으로 정의되어 있을 수도 있다. 플랫폼 변경 시 쉽게 대응할 수 있도록 독립적인 데이타 타입을 정의해야 한다.

3.2 인터페이스 계층

광학 문자 인식기는 호출하는 사용자에게 객체가 어떻게 호출되고, 데이터가 어떻게 저장되며, 어떻게 구현되어 있는가 하는 문제로부터 완전히 격리시켜 객체에 포함된 정보의 오용과 손상을 방지시키고, 독립성 및 이식성을 증가시키기 위한 캡슐화를 구현해야 한다. 즉, 사용자에게는 문자 인식기의 내부 데이터를 숨기고, 인터페이스 함수들만을 제공하여 접근 가능하도록 하여야 한다. 이와 같은 내부 구현이 드러나지 않은 최소한의 인터페이스에 의한 접근허가는 사용자가 해당 모듈을 사용하기 쉽게 해주고, 각 모듈(컴포넌트)간의 결합을 약하게 해 줌으로써 유지 보수비용을 줄일 수 있다. 캡슐화 구현 시 일반적으로 얻을 수 있는 장점은 다음 아래와 같다[22].

첫째, 접근 권한을 지정함으로써 속성과 연산에 대한 접근 여부를 제어할 수 있다.

둘째, 접근 권한으로 선언된 함수 및 데이터는 외부에서 직접적으로 접근이 되지 않으므로, 이들에 어떠한 변경이 발생하여도 영향을 받지 않게 된다.

셋째, 시스템 수정 시 파급 범위를 한정시킴으로써 시스템 개발 및 유지 보수 과정에서 발생하는 변경에 효율적으로 대처할 수 있는 이점을 제공한다.

본 논문에서는 광학 문자 인식기를 위한 인터페이스로 총 4개의 함수를 정의 하였다. 표 4는 문자 인식을 위한 최소한의 인터페이스 함수를 보여주고 있다. 아래 함수들을 사용하기 위한 순서를 보면, 먼저 응용 프로그램에서 문자 인식 시스템을 호출하기 위해 시스템의 동작에 필요한 초기화 작업을 수행해 주어야 한다. 따라서 제일 먼저 Construct 함수를 호출하여 인식에 필요한 전체 메모리를 확보하고 초기화 시켜준다. 또한, 시스템에서 사용하는 데이터 있을 경우, 파일에 저장 되어 있는 LoadDatas 함수를 호출하여 데이터 파일을 읽고 해당 메모리를 할당한 후 데이터를 넘겨받는 방식으로 구조화해야 한다. 이렇게 되면, 인식을 위한 각 처리 단계별 초기화 함수가 호출되어 각 처리 과정에 필요한 준비가 완료된 것이고, 이제 Recognition 함수를 통해서 문자 인식기를 호출 하여 입력 이미지를 바탕으로 인식 결과를 얻을 수 있다. 즉, 인식을 위해 필요한 모든 메모리 영역을 확보하고, 필요한 데이터를 메모리로 읽은 다음, 확보한 영역 안에서 인식이 이루어지도록 해야 한다. 또한 응용프로그램 종료가 이루어지면 Destruct 함수를 통하여 할당 받았던 모든 메모리 영역을 반환하도록 구조적으로 문자 인식기 인터페이스를 구성해야한다.

표 4 인터페이스 함수

함수 명	설명
Construct	엔진을 초기화 하고 인식에 필요한 메모리를 할당한다.
Destruct	엔진이 종료될 때, 할당된 메모리를 해제시킨다.
Recognition	문자 인식 시스템을 호출한다.
LoadDatas	데이터가 파일형태로 존재할 때 데이터를 읽고 해당 메모리를 할당하여 포인터를 넘겨준다.

3.3 엔진 지원 계층

문자 인식기에서는 인식을 위해 데이터를 저장하여 사용하는 경우가 있다. 예를 들어, 신경망을 사용한 인식기의 경우, 인식을 위해 신경망 가중치 벡터를 사용하고 신경망 가중치 벡터의 크기와 개수만큼 메모리를 할당해서 사용하게 된다. 만약 신경망 가중치 벡터가 정해져 있고 그것에 맞게 메모리를 할당하고, 개수에 맞게 연산을 수행하도록 정적인 구조로 프로그램이 되어 있다면 여러 경우의 문제가 생길 수 있다. 즉, 신경망 가중치 벡터의 종류와 크기는 학습양의 따라 그리고 분류를 원하는 문자의 종류에 따라 늘어날 수도 있고 줄어들 수도 있다. 이 경우 위에서 말한 것과 같이 정적으로 프로그래밍을 할 경우 벡터들이 바뀔 때마다 인식기 프로그램을 다시 수정해야하고 업데이트 하거나 새로 학습 과정을 추가할 경우 쉽게 확장하지 못하는 문제점이 발생한다. 다시 말해, 데이터가 변경될 때마다 인식 시스템 프로그램을 다시 수정해야 하고 새로 학습 과정을 추가 할 경우 쉽게 확장하지 못하는 문제점이 발생한다. 따라서 그림 4와 같은 플러그-인(Plug-In) 데이터 구조와 같이 헤더를 두어 이 헤더 정보를 읽어 오면서 그에 맞게 메모리 할당하고 연산을 하도록 구조화하는 방법을 사용하여야 한다. 따라서 하나의 플러그-인 데이터 구조를 통해 모든 기본적 구조가 잡히고 전체 프로그램 구조를 바꾸지 않고 이 파일만 수정함으로써 업데이트와 확장이 가능하도록 하였다.

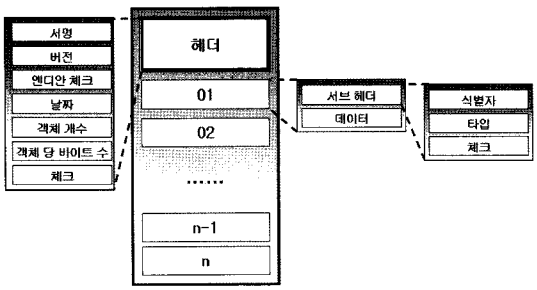

그림 4 플러그인(Plug-In) 데이터 구조

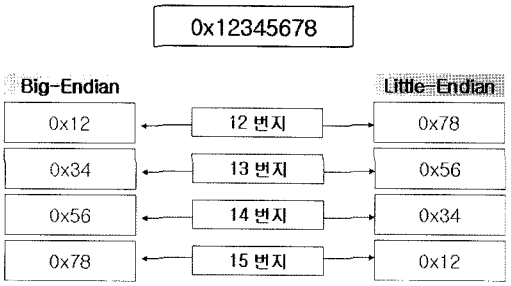

그림 5 빅 엔디안과 리틀 엔디안

또한 메모리에 저장된 데이터를 읽고 쓸 때, 프로세서에 따라 그림 5와 같이 다른 형식이 있을 수 있다. 리틀 엔디안(Little Endian) 방식은 위 비트의 값을 메모리의 하위주소에 저장하고, 빅 엔디안(Big Endian) 방식은 값의 상위비트의 값을 메모리의 하위주소에 저장한다. 따라서 프로세서에 따른 메모리 저장 방식(Endian) 체크 부분을 두어, 데이터를 메모리에 저장 시 각 플랫폼이 지원해 주는 방식에 적응적으로 대응할 수 있도록 작성되어야 한다.

플러그-인 데이터 헤더 및 서브 헤더에 데이터 체크 부분을 두어 데이터가 읽혀질 때 발생될 수 있는 에러 즉, 데이터의 개수에 따른 데이터 바이트 수, 총 데이터 크기와 같은 정확성 여부를 체크하여야 한다. 플러그-인 데이터 구조 헤더 부분에는 위에서 언급한 것 외에도 서명, 파일의 버전, 엔디안 체크, 생성 날짜, 데이터의 개수, 객체당 데이터 개수 등의 기본 데이터들이 저장되어 있다.

3.4 엔진 계층

모듈화는 프로그램을 작성할 때 큰 시스템을 한 번에 작성하는 것이 아니라 기능별로 나누어 부분별 작성을 한 다음, 각각의 작은 프로그램들을 서로 연결시켜 하나의 완성된 프로그램을 만드는 방법이다. 이 때, 기능별로 나누어진 각 모듈은 각각 하나의 기능을 수행하며 그 기능을 수행하기 위하여 필요한 모든 코드와 변수를 포함하도록 해야 한다[22].

광학 문자 인식기 개발 시 공통되는 부분을 하나의 모듈로 분리 하고, 각 단계별 프로세스를 분리시켜 인터페이스를 통해서만 접근하도록 모듈화해야 한다. 만약 각 단계가 모듈화 되어 있지 않다면, 하나의 프로세스에 문제가 있거나, 새로운 기능이 추가 또는 필요 없는 기능이 삭제될 때 재사용을 기대하기 힘들어 지므로, 각 단계를 모듈화 해야하고, 각 모듈은 인터페이스만을 제공하게 하여, 독립성을 유지시켜 주어야 한다. 본 논문에서는 공통으로 사용되는 이미지 모듈을 분리시키고, 각 기능별로 모듈화시켰다. 그리고 각 모듈은 필수적인 제약 사항만을 가지도록 하였다. 그림 6에서는 각 모듈이 하나의 입력과 출력을 바탕으로 한 문자 인식 엔진 데이터 흐름도를 보여주고 있다.

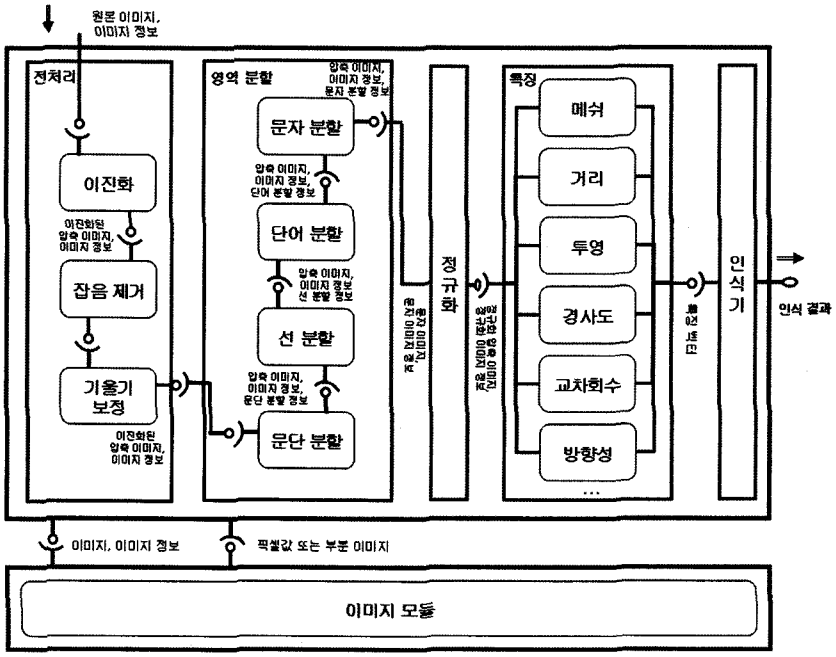

그림 6 CBOCR 엔진 흐름도

이미지 모듈은 문자 인식 시스템 인식 모듈을 제외한 전체에서 공통적으로 사용되는 부분을 분리한 모듈이다. 개별 픽셀 값을 얻어오거나 저장하며, 부분 이미지를 얻어오거나 저장하는 기능으로 구성되어 있다. 이미지 모듈은 전처리 과정, 영역 분할 과정, 정규화 과정 그리고 특징 추출 과정에서 공통적으로 이미지를 이용한 처리를 수행하므로, 공통 모듈로 분리하여 확장 및 변경이 용이하도록 하였다. 이미지 모듈은 값을 얻고자 하는 이미지를 입력으로 하고 원하는 화소 값 혹은 부분 이미지를 출력한다.

3.5 고정 소수점 데이타 타입 지원

대부분의 이동 통신 기기의 플랫폼에는 실수 연산용 프로세서(Float Pointing Unit)가 없으므로 소프트웨어적인 에뮬레이터를 통해 연산을 할 경우에는 연산 시간이 오래 걸리게 되는 단점이 있어 실시간 문자 인식기를 구현하는데 어려움이 따른다. 따라서 실수형 데이터를 정수형으로 변환하여 실시간 처리가 가능 하도록 해야 한다.

실수형 데이터들의 연산을 정수형 데이터들의 연산으로 바꾸기 전 고려해야 될 가장 중요한 사항은 소수점 이하의 데이타를 표현하는 방법에 관한 것이다. 이를 위해서는 정수형 타입을 가상의 소수점 위치 정보를 가진 데이터 타입으로 나타내어 사용해야 한다. 즉 정수형 데이터를 고정 소수점 데이터처럼 사용한다. 예를 들면, 1101.01(binary)의 경우 그림 7처럼 표현하여 상위 비트에는 부호 비트와 정수 부분으로 구성하고 하위 비트에 소수점 이하 부분으로 정하여 변환한다. 이때 문자 인식기에서 사용되는 데이터 값들의 동적 범위(Dynamic Range)를 측정하여 오류가 최소화 되는 적절한 위치를 선택해야 한다. 가상 소수점을 너무 높게 선택하면 오버플로우가 발생 하고, 너무 낮게 선택하면, 데이터 손실의 문제가 발생한다.

0	0…1101	0100000000…000

그림 7 10진수 6.25의 고정 소수점 변환

3.6 라이브러리 의존성 제거

이동통신 플랫폼은 플랫폼 상에서 제공해주는 라이브러리가 다르므로 이동통신 환경에서 소프트웨어 개발 시 라이브러리 의존성을 제거해 주어야 한다. 특히, 문자 인식기에서는 Math 라이브러리에 의존적이기 때문에, 이 Math 라이브러리의 의존성을 제거해 주어야 한다. 식 (1)에 나타난 시그모이드(sigmoid) 함수는 신경망을 이용한 인식 시스템에서 흔히 사용하는 활성화 함수이다.

$$f(net) = \frac{1}{1+\exp(-net)} \quad f(net) \text{의 범위 : } -1 \sim +1 \quad (1)$$

여기서 그림 8에 나타난 지수 함수의 사용은 프로그램 상에서 많은 수행 시간을 갖게 되고 이동통신 플랫폼의 경우 지수함수를 계산할 수 있는 라이브러리 또는 프로세스를 제공한다고 확신하기 어렵다. 따라서 Math 라이브러리의 의존성을 제거하기 위해 지수 함수의 선형 근사 방안이 필요하다.

선형 근사 알고리즘은 그림 9에서 보는 바와 같이, 나누고자 하는 구간을 설정하고 선형 근사 함수를 이용하여 원 시그모이드 함수의 가장 가까운 1차 근사 방정식을 구한다. 이 때 지수 함수와 근사 방정식의 최대 오차 부분을 구하고 그 점을 중심으로 새로운 구간을 나누게 된다. 각각의 구간에 대해서 다시 선형 근사 함수를 적용하여 같은 방법으로 그 구간에 대한 최대 오차를 구하고 구간들에 대한 최대 오차 값들을 다시 비교하여 큰 부분을 다시 나누는 과정을 반복하여 평균 오차가

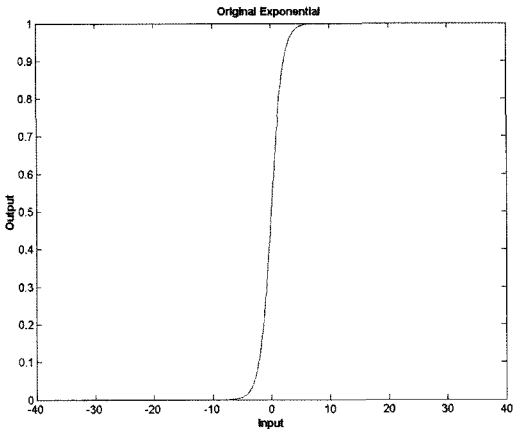

그림 8 Exponential 특성 곡선표

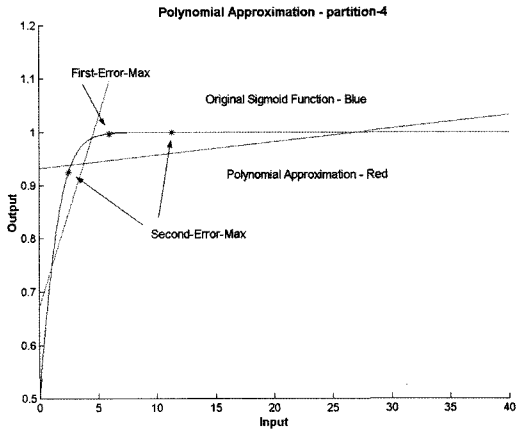

그림 9 지수 함구 근사

최소가 될 때까지 구간을 나눈다.

본 논문에서 구현한 문자 인식기에서는 이러한 방식을 이용해 128개의 구간으로 나눠 근사한 방정식을 구한 후 실험을 통해 신뢰할 만한 오차를 가지는 최소 구간 근사 방정식을 사용하였다.

4. 실 험

인식기는 신경망을 바탕으로 구현되었으며, 실험을 위한 환경은 AMD Baton 2500, 512 Mbytes 메모리 환경에서 구현 하였다. 또한, 카메라 영상을 바탕으로 한국전자통신연구원(ETRI)에서 개발한 개인 컴퓨터용 문자 인식기를 비교 대상으로 하여, 성능 측정을 하였다. 그리고 이동통신기기 환경인 한국 표준 플랫폼 WIPI(Wireless Internet For Interoperability)와 노르웨이 트롤텍사에서 개발한 PDA GUI Qt, 그리고 MS PocketPC환경에 이식하여 재사용이 얼마나 용이한지 시험하였다. WIPI와 PocketPc의 경우 메모리 저장방식이 리틀 엔디안(Little Endian) 방식을 사용하며, 메모리 힙 사이즈는 최대 1Mbyte 환경이었다. Qt 보드의 경우 ARM Processor에 빅 엔디안(Big Endian) 형식을 사용하는 환경이며, 메모리 힙 사이즈는 최대 64Mbytes 환경이었다.

4.1 실험 결과

표 5는 기존 한국전자통신 연구원에서 개발한 개인 컴퓨터용 문자 인식기와 본 논문에서 제안한 아키텍처를 바탕으로 변환된 문자 인식기의 수행 시간을 보여준다. 표에서도 볼 수 있듯이, 변환된 문자 인식기의 경우 실수 형 데이터를 정수형으로 변환하여 사용한 결과, 기존의 경우보다 성능이 향상된 것을 볼 수 있다.

표 5 수행 시간 (초단위)

	이진화 과정	선분할 단어 분할	문자 당 인식 과정	전체
기존 ETRI방법	0.202	0.078	0.416	0.697
제안한 방법	0.146	0.042	0.302	0.490

실험에 사용된 이미지는 640 * 480 크기의 컬러 이미지를 사용하였으며, 최대 지원 이미지는 1600 * 1200 크기로 제한하였다. 많은 양의 텍스트가 포함된 이미지의 경우 PocketPC LCD 사이즈 제한으로 인하여, 다음 줄로 표현되었다.

테스트 결과 중 많은 오 인식 결과를 보였는데, 이는 전처리 과정과 영역 분할 과정을 거치면서 두 문자가 붙어 한 문자로 인식되거나 한 문자가 두 문자로 인식되어, 오인식의 결과를 낳았다.

표 6은 본 논문에서 제안한 아키텍처를 바탕으로 변환 된 문자 인식기의 코드 라인 수에 따른 각 플랫폼별 변경된 코드 라인 수를 나타내주고 있으며, 전체 각 계층별 코드 라인 재사용 비율을 나타내고 있다. 이것은 데스크탑 환경에서 개발 되어진 소스 코드를 기준으로 측정한 결과이다. 표 6에서 나타낸 코드 라인 수는 헤더 파일 및 주석을 포함한 것이다.

여기서 보는 바와 같이 시스템에 의존적인 부분 플랫폼 의존 계층은 72% 재사용이 가능 하였으며, 이를 제외한 나머지 부분은 모두 재사용 가능 하였다. 하지만 플랫폼 의존 계층 중 특히 이미지 관리 부분과 타입 정의 부분은 거의 재사용이 이루어지지 않았다. 또한 PocketPC, WIPI와 Qt에서 플랫폼 의존 계층의 재사용 코드 라인 수가 같은 값으로 나온 것은 수정되는 부분이 동일하였기 때문이다.

표 6 각 플랫폼 별 변경된 코드 라인 수

	플랫폼 의존	인터 페이스	엔진 지원	엔진 계층	재사용율
전체 줄 수	347	60	59	6133	
PocketPC	97	0	0	0	98.53 %
WIPI	97	0	0	0	98.53 %
Qt	97	0	0	0	98.53 %

위 실험 결과에서 알 수 있듯이, 기존의 ETRI에서 개발한 개인 컴퓨터 환경에서의 인식결과와 마찬가지로 PocketPC, WIPI 그리고 Qt의 실험 결과에서 모두 동일한 인식 결과를 얻을 수 있었으며, 실수 형 데이터를 정수형으로 변환하여 인식 성능 또한 향상되었음을 알 수 있었다. 서로 다른 이동통신 플랫폼 환경이며 다른 메모리 저장 방식을 지원하는 환경하에서 광학 문자 인식기의 의존적인 부분만을 수정함으로써, 전체적으로 98.55% 이상의 코드 재사용이 가능하였다.

5. 결 론

본 논문에서는 이동통신 플랫폼에서 문자 인식기 재사용을 위한 소프트웨어 아키텍처를 설계하였다. 본 논문에서 제안된 소프트웨어 아키텍처는 계층형 아키텍처로 설계되었으며, 플랫폼 의존 지원 계층, 인터페이스 계층, 엔진 지원계층 그리고 엔진 계층으로 총 4개의 계층으로 나누었다. 엔진 지원 계층에서는 문자 인식기에서 데이터 이용 시 플랫폼마다 다른 메모리 저장 방식(Big Endian, Little Endian)에 적응적인 대응을 위한 플러그-인(Plug-In) 데이터 구조를 설계하였으며, 인식 성능을 향상시키기 위해 부동 실수형 데이터를 가상의 고정 소수점이 있는 정수형 데이터로 변환하였다. 또한 라이브러리의 의존성을 없애기 위해 선형 근사 방법을 문자 인식기에 적용하였다.

실험은 기존 ETRI에서 개발된 문자 인식기를 본 논

문에서 제안한 아키텍처를 바탕으로 변환하였으며, 인식을 위한 분류기는 신경망을 이용하였다. 그리고 변환된 문자 인식기를 다양한 이동통신 플랫폼(WIPI, Qt, PocketPC)에서 실험 하였다. 실험 결과 문자 인식 소프트웨어 코드의 98% 이상이 재사용 가능함을 확인하였으며, 세 플랫폼 모두 성능이 향상된 동일한 인식 결과를 얻을 수 있었다.

참 고 문 헌

[1] 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.

[4] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Second Edition. Addison-Wesley, Boston, 2003.

[5] M. J. Davis and R. B. Williams "Software Architecture Characterization," *ACM SIGSOFT Software Engineering Notes*, vol.22, no.3, pp.30-38, May 1997.

[6] W. B. Frakes and K. Kang, "Software Reuse Research: Status and Future," *IEEE Transactions On Software Engineering*, vol.31, no.7, pp.529-536, July 2005.

[7] D. E. Perry, and A. L. Wolf, "Foundations for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes*, vol.17, no.4, pp.42-52, Oct 1992.

[8] SEI in Carnegie Mellon University, "How Do You Define Software Architecture," http://www.sei.cmu.edu/architecture/definitions.html

[9] J. Chen, "Architecture for Systematic Development of Mechatronics Software Systems," Licentiate Thesis, Royal Institute of Technology, Sweden, ISSN 1400-1179, ISRN KTH/MMK-01/06-SE.

[10] P. Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*, vol.12, No.6, pp.42-50, Nov. 1995.

[11] R. C. Gonzales and R. E. Woods, Digital Image Processing, Second Edition. Prentice Hall, New Jersey, 2002.

[12] N. Otsu, "A Threshold Selection Method from Gray-level Histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol.9, no.1, pp.62-66, March 1979.

[13] O. D. Trier and A. K. Jain, "Goal-Directed Evaluation of Binarization Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.17, no.12, pp.1,191-1,201, December 1995.

[14] Y. C Shin, R. Sridhar, V. Demjanenko, P. W. Palumbo and J. Hull, "Contrast Enhancement of Mail Piece Images," *Proceeding Machine Visoin Applications in Character Recognition and Industrial Inspection, San Jose, USA*, pp.27-23, February 1992.

[15] S. C. Hinds, J. L. Fisher and D. P. D'Amato, "A Document Skew Detection Method Using Run- length Encoding and the Hough Transform," *Proceeding 10th International Conference on Pattern Recognition*, Atlantic City, USA, pp.464-468, June 1990.

[16] S. H. Kim, C. B. Jeong, C. Y. Suen, "Word segmentation of printed text lines based on gap clustering and special symbol detection," *IEEE International Conference on Pattern Recognition*, Quebec City, Canada, pp.320-323, August 2002.

[17] T. Akiyama and N. Hagita, "Automated Entry System for Printed Documents," *Pattern Recognition*, vol.23, no.11, pp.1141-1153, October 1990.

[18] S. W. Lee, D. J. Lee, H. S. Park, "A New Methodology for Gray-Scale Character Segmentation and Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.18, no.10, pp.1045-1050, October 1996.

[19] Geetha Srikantan, Stephen W. Lam and Sargur N. Srihari, "Gradient-Based Contour Encoding for Character Recognition," *Pattern Recognition*, vol. 29, no.7, pp.1147-1160, July 1996.

[20] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, Second Edition, Wiley-interscience Publication, New York, 2000.

[21] S. N. Srihari, J. J. Hull and R. Choudari, "Integrating Diverse Knowledge Sources in Text Recognition," *ACM SIGOA Conference on Office Information Systems*, Philadelphia, USA, pp.463-479, January 1983.

[22] 윤청, 소프트웨어 공학, 생능 출판사, 경기도, 1999.

김 세 호

2004년 관동대학교 컴퓨터교육과 학사. 2006년 중앙대학교 컴퓨터공학부 석사. 2006년~현재 중앙대학교 컴퓨터공학부 박사과정. 관심분야는 패턴인식, 임베디드 소프트웨어 아키텍쳐 등

이 찬 근

1996년 중앙대학교 전자계산학과 학사. 1998년 KAIST 전산학과 석사. 2005년 Univ. of Texas at Austin 전산학과 박사. 2005년~2007년 미국 인텔 소프트웨어 엔지니어, 2007년~현재 중앙대학교 컴퓨터공학부 조교수. 관심분야는 실시간 소프트웨어, 수행시간 모니터링, 소프트웨어 테스팅 등