David : 일반화된 영상처리 개발 툴

David : A Well Formed Image Processing SDK(System Development Kit)

▌이봉규 / 제주대학교 전산통계학과 교수, (주)바이오인포마이너 대표
▌양요한 / 제주대학교 전산통계학과 석사 과정, (주)바이오인포마이너 개발팀장

잘 정의되고 일반화된 영상처리 SDK (System Development Kit)는 영상처리 응용시스템 개발자들이 손쉽게 응용분야를 개발할 수 있도록 해준다. 그러나 일반화된 SDK에 대한 명확한 정의나 연구가 국내에서 이루어지지 않아 아직까지 영상처리를 위한 일반화된 SDK가 개발된 사례는 없다. 이에 본 개발팀에서는 영상처리에 관련된 다양한 응용 시스템 개발에 공통적으로 사용이 가능한 일반화된 영상처리 SDK인 "David"의 개발을 통하여 기존의 다른 영상처리 툴과 차별화되는 새로운 개념의 S/W에 접근하였다.

이 글에서는 구현된 영상처리 SDK인 "David"에 대한 구현 배경 및 구현 방법을 간략히 소개한 후, 구현된 실제 S/W를 통하여 새롭고 다양한 기능들을 소개한다.

서 론

현재 세계적으로 영상처리(Image Processing) 응용분야에 대한 연구/개발이 지속적으로 이루어지고 있으며, 기술적인 진보도 상당해 이루어지고 있다. 또한 멀티미디어 정보의 활용이 증가하고 있어 광학과 펜 기반 문자 인식, 멀티미디어 내용 기반 정보 검색과 이해 등에 대한 흥미와 기대감이 증가하고 있는 추세이기 때문에 향후 영상처리 응용 기술 개발은 확대될 것이다[1,2]. 이런 다양한 응용분야 중에서도 특히 생체인식(Biometrics), 영상자동검색(Contents Based Retrieval) 등은 차세대 선도기술로 중요성이 날로 증가하고 있다[3]. 따라서 영상처리 응용 분야 중 하나인 생체인식분야를 통해 영상처리 응용개발에 적합한 SDK (System Development Kit, 본 논문에서 SDK는 영상처리에 관련된 개발 툴만을 대상으로 함)의 중요성을 알 수 있다.

생체정보인 지문, 홍채, 손등정맥, 얼굴 등을 이용하여 개인에 대한 보안/인증을 수행하는 생체인식에 관한 연구는 자체의 편리성, 정확성, 유용성으로 인하여 획기적인 기술로 평가받고 있다[4]. 이런 추세를 반영하듯, 다양한 생체정보에 대한 인식방법[5,6], 이를 활용한 제품[6] 등이 국내외적으로 폭넓게 연구/개발이 진행되고 있으며, 많은 결과들이 소개되고 있다. 이런 연구/개발의 추세는 자연스럽게 영상처리를 위한 SDK에 대한 필요성을 증가시키고 이에 최근 국내외에서는 생체인식을 포함한 영상처리 응용 전반에 사용할 수 있는 일반화된 SDK (Well Formed SDK)를 개발하려는 노력을 활발히 전개하고 있다[7]. 그러나 이런 노력에도 불구하고 아직까지는 일반화된 SDK를 개발한 사례는 없는데, 그 이유는 이런 종류의 SDK인 경우, 기존의 영상 처리 툴(편집 기능 위주)과는 근본적으로 다른 특성이 있기 때문이다.

생체인식을 포함한 모든 영상처리 응용 시스

템들은 그림 1과 같은 공통된 개발 흐름을 가지고 있다[8]. 개발의 예를 생체인식으로 든다면 우선 입력장치(카메라, 생체정보 측정장치 등)를 통해 인식에 사용할 생체영상을 컴퓨터에 저장하고(그림 1-①) 2) 저장된 영상을 영상 프로세싱 기법을 활용하여 인식에 필요한 특징을 추출한 후(그림 1-②) 3) 추출된 특징을 데이터베이스 (Data Base, DB)에 저장된 각 개인의 특징들과 비교하여 인식하는 과정(그림 1-③)을 반복하는 흐름을 가진다. 이런 절차는 보통 인식 대상에 무관하게 공통적으로 적용되며 특히 2)번 단계의 경우 다양한 영상처리 기법으로 최적의 특징을 찾는 것이 생체인식과정에서 핵심적인 요소이다.

이런 2)번 단계에서 처리과정을 거쳐 만들어지는 중간 결과물들은 자신의 이전 단계에서 처리된 결과를 입력으로 받는 등 밀접한 관계를 가지고, 각 단계는 자신에게 적용된 영상 처리기법에 대한 세부정보를 유지해야 한다. 따라서 SDK의 경우는 하나의 S/W내에서 입력 장치로부터의 영상 입력, 단일 영역에서 입력 영상에 대해서 복잡한 영상 처리기법을 반복적으로 여러 번 수행이 가능하도록 다단계 처리를 위한 영상 관리와 처리된 복수개의 결과물에 대한 연관성을 제어/저장하는 기술이 필수이다. 기존의 단순 편집 기능을 위주로 한 영상 툴에서는 다단계의 복잡한 영상 프로세싱을 제공하고 있지 않으며, 다단계 반복 처리에 따른 각각의 결과를 조정하고 제어하는데 한계를 가진다.

이렇게 기존의 SDK에서 문제점이 나타나는 근본적인 이유는 영상처리를 위한 SDK의 구현에 기존의 영상처리 S/W에서 사용했던 내부 자료 구조 및 제어 방식을 그대로 사용하거나 제한된 영역에서 유용한 구현 방법을 이용하기 때문이다. 이런 접근 방법으로는 일반화된 SDK를 구현하는데 한계가 있다. 기존의 SDK 개발 방법을 통해 이들 문제점을 지적해보면 다음과 같다.

현재까지 SDK 개발을 위해 사용되었던 방법은 2가지 정도로 분류해 볼 수 있다. 첫 번째 방법은 기존의 편집위주 S/W에서 사용되었던 내부의 구조나 방법을 사용하여 그 위에 단순히 영상처리 기법만 구현하여 추가하는 방법이다[9]. 이 방법을 사용한 개발 사례로 대표적인 것이 Matlab, Photoshop[10] 등을 이용한 것이다. 즉 Matlab에서 동작하는 프로그램을 작성하거나(M file 포맷), Photoshop에서 작동하는 플러그인 루틴이 대표적인 예이다. 이런 접근 방법은 범용성을 지닌 검증이 된 S/W를 그대로 사용할 수 있다는 장점이 있기는 하지만 영상처리 응용을 개발하는데 충분한 기능을 제공하는 것에 한계를 가지는 것은 물론 다단계의 영상 처리에는 적합하지 않다는 근본적인 문제를 지니고 있는 방법이다(Figure 2-①). 두 번째 방법으로는 독립적인 별도의 SDK를 개발하는 방법이다. 이 방법을 이용한 대표적인 사례가 HIPS[11], Khoros [12], XITE [13] 등이다. 이들 구현 SDK들은 라이브러리형태이거나 독립된 플랫폼의 2가지의 형식을 가진다.

이런 접근 방법의 경우, 첫 번째와는 달리 자체적인 전용 구조를 이용하여 영상을 처리하고 있으며, 목적에 적합한 다수의 복잡한 처리루틴을 제공하는 것이 특징이다. 그러나 이런 방법 역시 영상처리에 적합한 범용의 구조보다는 처리 과정에 중점을 두었기 때문에 영상처리를 위해 필요한 각 처리 요소간의 연관성 제어

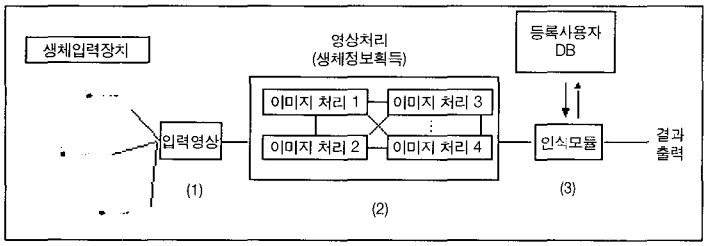


그림 1 생체인식 과정

는 할 수 없으며, 결과영상의 저장방식도 기존의 영상 저장형식을 그대로 사용하기 때문에 처리 결과에 대한 관리가 어려운 것은 사실이다.

이렇듯 기존의 개발 방법으로는 최종목표인 일반화된 SDK를 구성하는데 많은 제약이 따르게 되는데, 그 근본적인 이유는 영상처리 시스템의 특성을 정확히 분석하여 영상처리에 적합한 SDK의 내부 자료구조를 명확히 제시하고 이를 바탕으로 필요한 세부기능들을 구현하는 방법을 사용하지 않았기 때문이다. 즉, 일반화된 SDK를 구현하기 위한 가장 기본적인 연구가 없었기 때문에 나타나는 문제점들이다.

본 개발팀은 지금까지 개발되었던 영상처리 관련 SDK에서 공통적으로 나타나는 문제점을 근본적으로 해결하여 일반화된 SDK를 구현하기 위한 첫 단계로써 다단계 영상처리를 효과적으로 지원할 수 있는 내부구조로써 새로운 레이어(Layer) 구조와 이 구조에 대한 저장 및 연관성제어 방법 등을 개발하고(특허 출원 중인 기술) 이것을 바탕으로 실제 영상처리 SDK를 개발하는 흐름을 따랐다. 이러한 개발과정을 통해 영상처리 응용기술 개발에 전용으로 사용될 새로운 개념의 SDK인 "David"을 구성한 것이다.

본 해설의 구성은 다음과 같다. 2장에서는 먼저 개발된 전용 내부 구조인 레이어와 그에 따른 제어 방법을 설명하고 3장에서는 실제 구현된 일반화된 SDK인 David의 다양한 기능을 예제를 통해 직접 소개한다. 그리고 4장에서 결론을 맺는다.

영상처리 전용 내부구조 - 레이어 및 리스트

일반화된 SDK개발을 위해 필요한 새로운 내부 자료구조가 갖추어야 할 가장 중요한 특성은 단일 영역에서 영상을 다단계처리 하고 그 결과들은 저장/제어할 수 있어야 한다는 것이다. 지금까지는 한 영역에 있는 영상을 처리하면 다른 영역에 그 결과가 나타나는 방식이므로 다단계 처리를 할 경우 상호 연관성을 파악하기가 어려웠기 때문에 일반적인 SDK의 개발이 어려웠다. 또한 영상처리의 경우 단순 영상 편집과는 달리 사용되었던 처리 방법이나 관련 파라미터에 대한 내용이 매우 중요한 기능을 가지는 것이 일반적이지만 기존의 툴에서는 처리 방법에 대한 내용을 직접적으로 제어할 수 없었기 때문에 정확히 개발과정을 재현하는 것이 어려웠다. 이에 본 연구팀은 이런 문제점을 근본적으로 해결할 수 있도록 기존의 개념과는 완전히 다른 새로운 레이어 및 레이어 연결구조를 개발하였다. 개발된 구조는 입력영상을 하나의 영역에서 처리가 가능토록 하여 단계간의 연관성의 파악이 가능하기 때문에 기존에는 구현이 힘들었던 일반화된 SDK를 손쉽게 구현할 수 있도록 한다.

전용구조 – 기본 개념

기존의 영상 편집 S/W나 영상처리 S/W의 레이어는 "한 화면에 겹쳐 싸여있는 아세테이트지" [10] 라고 정의하는데 이는 한 화면에 배경-〉영상 1 -〉 영상2를 겹쳐 배열하여 복합적인 영상을 만드는 것으로 마치 여러 장의 사진을 겹쳐놓은 것과 같은 형태를 띠도록 하는 것으로 본 논문에서의 정의와는 다른 것으로 입력되는 영상의 형태가 제한되어 있기 때문에 (*n*비트로 표현되는 그레이(Gray) 값) 입력영상을 다양한 영역(주파수 영역, 복소수 영역 등)으로 변환시켜야 하는 영상처리들에 그대로 적용시키는 것이 어렵다.

이에 비해서 단일 영역에서의 다단계 영상처리를 위해 본 논문에서 제안하는 레이어는 입력된 영상을 처리할 경우 나타나는 모든 영역에 대해 처리가 가능한 내부구조를 가지고 있는 새로운 구조이다 (그림 2). 그림 에서 보듯이 전체 구조는 3가지의 세부구조(제어, 영상 저장, 인자전

달)로 나뉜다. 제어요소는 현재 레이어가 가리키는 영상의 크기(가로, 세로), 채널의 수와 영상 저장형식을 담는 일반적인 필드(Field)와 현재 활성화되었는지 여부, 화면에 나타낼지 여부와 레이어에 적용된 처리와 관련된(처리 후 원래 영상과 크기가 다를 수 있음) 변수로 구성된다. 영상 저장부분은 실제 처리된 영상을 저장하는 영역인데 어떤 영상이 어떤 변환에 사용될지는 사전에 알 수 없기 때문에 처리방법에 무관하게 결과를 모두 받을 수 있도록 실수영역 영상 (Real Part Image), 허수영역 영상 (Imaginary Part Image), 컬러 (Color) 영상, 그레이 영상에 관련된 저장 영역을 모두 가지고 있다. 인자 전달 요소는 현재 레이어에 적용했던 처리기법의 실제 이름과 사용했던 파라미터를 가지고 있다. 이렇듯 복합적인 구조를 가지고 있기 때문에 처리방법에 따라 결과를 저장/제어하는 내부 구조를 추가/변경 할 필요 없이 제안구조 하나만으로 모든 정보를 저장하고 관리할 수 있는 것이다. 이러한 특성을 가진 레이어는 새로운 이벤트가 발생할 경우 동적으로 생성된다.

또한 제안하는 레이어 구조는 생성된 순서에

구조명	구성 레코드 명	구성 필드 수	설 명
레이어	제어요소	16	현재 레이어의 식별자, 이미지의 크기 등 정보, 다음 레이어의 위치
	이미지 저장	22	정수/실수/복소수 데이터 저장영역 (영상 처리 결과)
	인자전달	2	현재 레이어에 적용된 영상처리 루틴에 사용된 인자들

그림 1 생체인식 과정

따라서 서로 연결된 구조를 가지는데 이런 연결리스트 (리스트, Linked List)를 통해 특정 입력 영상에 관련된 영상처리 결과물들을 다른 입력 영상들에 대한 것과 구별하여 관리함으로써 단일영역에서 입력 영상을 다단계 처리하는 것이 가능하다. 단일 영역에서 한 영상의 다단계 처리결과를 모두 나타내기 위해서는 현재 영역에 나타내어야 할 영상들과 다른 영역에 표시할 영상을 구별하여 관리해야 한다. 영상처리 SDK의 경우 상이한 여러 개의 입력 영상을 동시에 처리하는 경우가 빈번하기 때문에 이런 구별이 불가능할 경우 한 영역에 관련있는 여러 개의 처리결과를 한꺼번에 나타낼 수 없게 된다. 따라서 단일영역에서 영상을 다단계 처리하는 기능을 위해서는 영상이 입력되면 이 입력을 저장한 레이어로부터 처리가 될 때마다 레이어가 동적으로 생성되어 결과를 저장한 후 입력으로 사용한 레이어와 결과 레이어를 서로 연결시켜 하나의 리스트로 만들어 줌으로써 레이어들을 입력에 따라 그룹화하는 기능은 필수적이다. 그림 3은 입력 영상을 4단계로 처리한 후 생성되어진 레이어들이 상호 연관되어진 리스트의 구조를 보여준다. (입력 영상 0번 레이어, 마지막 결과는 4번 레이어).

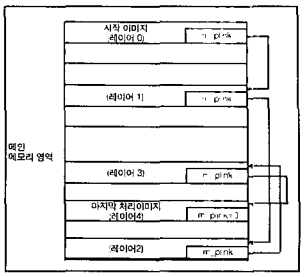


그림 3 5개의 레이어들이 연결된 리스트 구조

제안구조의 장점

획득영상에서부터 최종 처리 결과에 이르는 각각의 처리결과와 사용된 정보 등은 레이어에 저장되며, 이들 레이어들은 하나의 리스트를 형성하기 때문에 사용된 영상처리 루틴들은 데이터(영상)의 흐름 (Data Flow)에 의해서 제어되는 구조이다. 따라서 제안구조를 이용하면 개별 처리결과 뿐 아니라 상호연관성도 고려할 수 있기 때문에 다른 영상처리 툴 등에서 구현하지 못했던 단일파일에 전체 처리 결과를 저장하는 것과 새로운 처리 단계를 삽입/삭제하는 과정을

자동으로 제어 할 수 있는 SDK를 쉽게 구현할 수 있다.

현재까지 작업한 결과(다단계 영상 처리 결과)를 가지고 있는 특정 리스트의 모든 내용을 단일 파일에 통합 저장하기 위해서는 처음 입력 영상에서(레이어 0)에서부터 시작하여 마지막 레이어에 이르기까지 링크를 추적하면서 순차적으로 각 레이어를 저장하는 방법을 사용할 수 있다. 저장방법은 먼저 리스트 자체에 대한 정보인 리스트 관련 정보(리스트 식별자 등, 현재 처리되고 있는 리스트가 여러개일 수 있기 때문에 이들 간의 식별이 필요함)를 헤더형식으로 저장한다. 그런 다음 리스트의 첫 번째 레이어부터 시작하여 링크를 추적하면서 각각의 레이어가 가진 제어정보, 이미지저장 요소, 인자전달요소를 순차적으로 저장한다. 이 작업은 리스트의 끝(링크가 NULL)에 도달할 때 까지 계속된다.

제안된 구조를 이용한 SDK의 또 다른 장점은 현재 리스트에 새로운 처리과정을 삽입하거나 처리과정을 제거할 경우, 삽입/제거된 데이터에 영향을 받는 부분을 동적으로 처리 할 수 있다는 것이다. 이 기능은 기존에는 구현하기 힘들었던 것으로 제안 구조의 우수성을 보여주는 대표적인 예이다. 이런 특징의 장점은 다단계 처리에서 발생할 수 있는 예를 통하여 알아볼 수 있다. 현재 입력 영상 -> A 변환 -> B 변환 -> C 변환으로 다단계 처리하는 과정에서, A 변환과 B 변환 사이에 새로운 B' 변환을 삽입할 경우 (즉 A 변환으로 처리한 입력 영상을 B' 으로 처리하고 그 결과를 B로 처리하는 경우), 기존에 B단계와 C 단계에서 얻어졌던 결과는 각기 입력이 바뀌기 때문에 B 단계와 C 단계를 다시 재 계산해야한다 (삭제의 경우도 마찬가지임). 기존에는 이런 경우 전체 단계를 처음부터 다시 재 계산하는 방법을[12] 사용하였기 때문에 단계가 복잡할 경우 많은 시간이 소요되었다. 또한 이런 재 계산 단계를 사용자의 명령에 의해서 진행했기 때문에 정확성을 기하기 어려웠다. 그러나 제안 구조에서는 이런 기존의 문제점을 연결 리스트의 장점을 이용하여 자동으로 해결한다. 만약 특정 단계 뒤에 새로운 처리가 삽입될 경우, 바로 전 단계의 레이어에 있는 결과 데이터를 입력으로 하여 선택된 처리를 수행한 후 사용자가 지정한 위치에 레이어를 삽입한다(연결 리스트 삽입 알고리즘으로 구현 가능). 그런 다음에는 자동적으로 현재 삽입된 레이어에 있는 결과를 입력으로 하여 다음 단계에 있는 변환을 수행하여 해당 레이어의 결과를 갱신한다. 이런 과정을 리스트의 끝이 될 때까지 수행하는 것이다. 이렇듯 제안된 구조를 사용할 경우에는 전체 단계를 처음부터 다시 처리하는 것이 아니라, 삽입된 위치 이후에 있는 단계들만 재 계산해주면 되기 때문에 효율적이다. 아울러 삽입이 발생한 경우 자동으로 필요한 부분을 재 계산해주기 때문에 항상 정확한 결과를 유지할 수 있다.

구현된 SDK - David

구현 플랫폼

David의 실제 구현은 윈도 (Windows) 기반의 PC(Personal Computer)상에서 비주얼 C++ (Visual C++) 6.0을 사용하였다. 이처럼 실제 구현을 위한 플랫폼(Platform)을 윈도로 함으로써, UNIX기반의 W/S를 (Workstation) 기반으로 구현된 대부분의 영상처리 SDK구현에 비하여 여러 가지 장점을 가질 수 있다. 우선 하드웨어적인 측면에서 본다면 PC는 W/S에 비해서 성능향상(계산 속도의 증가) 속도가 빠르기 때문에 대량의 계산을 요구하는 영상처리에 보다 적합하다. 또한 PC의 운영체제인 윈도는 다양하고 강력한 개발도구를 (비주얼 C++ 등) 다수 제공하고 있기 때문에 S/W 개발 측면에서도 PC가 보다 나은 개발조건을 가졌다고 볼 수 있어 윈도기

반의 PC를 플랫폼으로 사용하게 되었다. 더욱이 현재 영상처리 응용 시스템을 개발하는 개발자 대다수가 윈도 환경에 익숙해져 있기 때문에 운용적인 면에서도 타당하다. 표 1은 구현에 대한 세부적인 내용을 정리한 것이다. 또한 표 2는 David에서 구현한 기존에는 볼 수 없었던 기능들을 요약한 것이다.

표 1. 구현된 SDK의 세부사항

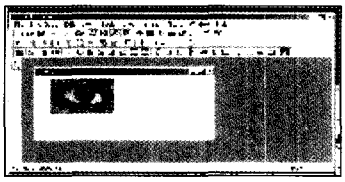
구 분	내 용
구현언어	비주얼 C++ 6.0, 개발 라이브러리, 기타 윈도상의 개발 툴
개발목표	제안한 레이어 구조 및 제어방법을 윈도환경에서 실현한 응용 S/W
개발용 PC 사양	팬티엄 II 200MHz 이상, 128M 바이트 이상의 기억장치, 스캐너/디지털 카메라를 위한 USB port, 윈도 계열 운영체제

표 2. 구현된 개별사항

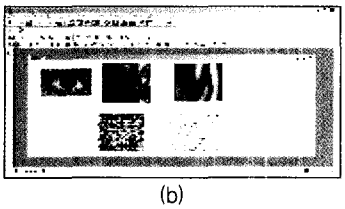
구현기능	내 용
1. 레이어 및 리스트 구조의 구현	제안하는 레이어 및 리스트를 실제 구현하였는가
2. 단일 영역에서 영상 다단계 처리	한 영역(다큐먼트)에서 영상을 다단계 처리 가능한가
3. 처리결과의 단일파일 저장	다단계 처리결과를 한 파일에 저장하고 제어 가능한가
4. 동적인 레이어의 삽입과 그에 따른 연관성 제어	제안하는 연관성 처리가 동적으로 이루어지는가
5. 영상에 대한 매크로 기능	한 영상에 적용된 처리절차를 그대로 다른 영상 적용 시킬 수 있는가

작업영역 (Workspace)

그림 4는 구현된 David의 초기 화면과 실제 한 영역에서 영상을 다단계 처리하는 과정을 보여준다. 이 S/는 다양한 버전(Version)의 윈도(98/ME/2000/XP)에서 모두 동작할 수 있으며, GUI(Graphical User Interface)를 지원한다. 그림의 (a)는 시작한 후 나타나는 초기 화면이다. 그림에서 보듯이 입력 영상을 읽어들이면 내부적으로는 새로운 레이어가 자동 생성되어 입력 영상을 저장하고 화면에 출력하게 된다. (b)는 입력영상을 이용한 다단계 영상처리 결과를 보여준다. 입력영상에 대해서 처리할 경우, 각 단계마다 레이어들이 자동으로 생성되면서 현재 단계에서의 영상처리 결과를 저장하게 된다. 이런 일련의 과정을 거치면 (b)에서 보이는 것과 같이 다단계 처리된 모든 결과를 화면에서 확인할 수 있게 된다. 여기서 주목할 것은 바로 (b)에서 보듯이 모든 처리 결과 및 입력 영상이 단일 영역에 표시된다는 것이다. 기존의 SDK의 경우 입력 영상이 처리될 경우 새로운 영역(도큐먼트)이 생성되어 그 결과를 저장하고 화면에 표시하는 반면, 제안된 레이어 및 리스트를 이용한 본 S/W인 경우는 특정 입력 영상에 관련된 모든 처리결과는 한 영역에 표시되기 때문에 개발자의 경우 처리과정의 흐름을 파악하기가 용이하기 때문에 응용 시스템 개발 공정이 단순화되어 개발의 효율성을 높일 수 있다.



(a)



(b)

그림 4 David의 작업 영역 및 사용자 인터페이스

저장 파일 형식-"Dav"

단일 영역에서 처리된 다단계 처리 결과를 하나의 파일로 저장하는 기능 역시 제안된 구조를 통해 쉽게 구현할 수 있다. David에서는 Xite [13], Khoros [12] 등과 마찬가지로 자체의 영상저장 형식인 "Dav"를 구현하고 있다. 구현된 "Dav" 형식은 기존의 영상저장 방법과는 달리 한 영역에 있는 모든 처리 결과들(레이어 리스트 전체)에 대해서 데이터, 영상처리방법, 사용 파리미터, 영상

의 특성을 모두 저장한다. 이런 방법이 가능한 것은 구현 SDK의 경우 특정 영역에 있는 모든 결과를 레이어 리스트로 관리하기 때문에 개별 레이어의 연관성을 알 수 있기 때문이다. 입력 영상을 시작으로 리스트내의 모든 레이어를 추적하면서 현재 도달한 레이어가 가지는 모든 내용을 순차적으로 파일에 기록하기 때문에 작업한 모든 내용을 한 파일에 기록할 수 있는 것이다. 그림 5는 단일 파일 저장을 위한 컴퍼넌트(Component)인 "save_layer_list" 클래스의 동작과 관련된 예를 보여준다. 그림의 (a),(b)에서는 입력 영상을 5단계로 처리한 후의 화면 결과를 "eye.dav"라는 단일 파일에 저장하는 것을 보여준다. (c)에서는 저장결과를 다시 읽어봄으로써 저장된 파일이 현재 영역의 모든 결과를 올바르게 유지하고 있음을 증명하고 있다.

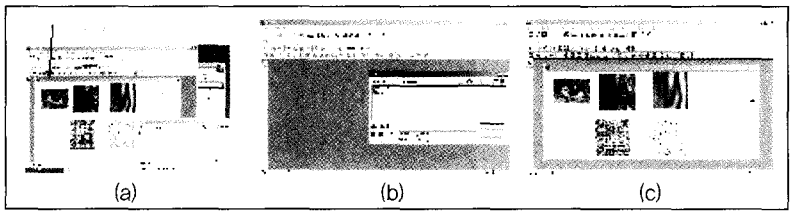


그림 5 단일 파일 저장 예

단계별 처리결과의 연관성 제어

David을 구성하는 컴퍼넌트 중 하나인 "image_control" 클래스는 처리결과의 연관성 자동제어에 해당되는 부분으로 기존에는 없던 새로운 기능으로 (Khoros의 경우는 제한적인 구현) 제안된 레이어 및 리스트의 유용성을 직접적으로 볼 수 있는 기능이다 (그림 6). 우선 그림의 (a)에서는 입력된 영상을 5단계로 처리 한 후의 결과를 보여준다. 5단계에 적용된 처리방법은 모두 다르지만 모두 같이 레이어 구조에 저장되고 상호 리스트를 유지하고 있다.

각 레이어들은 입력영상을 선두로 생성된 순서에 따라서 번호(0, 1, 2, 3, 4, 5)로 번호가 부여되어 있다. 이런 번호를 통해 특정 레이어에 대한 선행자와 후속자를 알 수 있는데 예를 들면 2번 레이어의 경우는 1번 레이어의 결과 영상을 입력으로 받아 처리한 후 자신의 결과로 보관하면서 동시에 3번 레이에에 입력으로 전달한다. 이런 상황에서 만약 입력 영상(0번)에 새로운 처리 방법을 적용(1과 1 사이에 새로운 6번 레이어가 삽입)한 경우 기존의 1번에서 5번까지의 레이어는 자신이 받아들이는 입력과 출력이 바뀌게 되며 따라서 처리 결과도 달라진다. 즉 현재 표시된 내용을 변경해야 하는 것이다. 기존의 개발 환경에서는 이럴 경우 사용자가 일일이 수작업을 통해서 만들어 주어야하기 때문에 어려운 점이 많았다. 그러나 David에서는 레이어 리스트의 연관성을 이용하여 이 작업을 자동으로 진행한다. 즉 특정 레이어가 새로 삽입되면 삽입된 레이어 이후에 있는 리스트 요소들에 대해서 자동으로 계산과정을 다시 수행함으로써 이 문제를 쉽게 해결하는 것이다. (b)와(c)가 이런 특성을 보여준다. (b)의 경우 새로운 레이어 6이 삽입되면 1에서 5번까지의 레이어가 자동으로 다시 처리되는 결과를 보여준다. (c)에서는 새로 삽입된 6번이 삭제되면 이전으로 돌아간다.

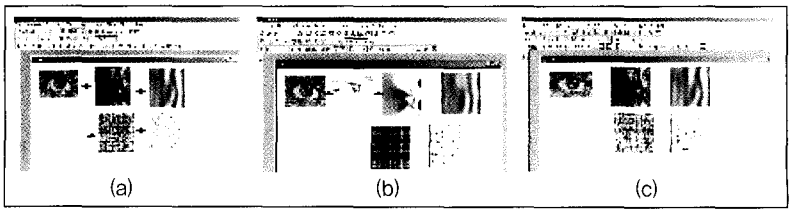


그림 6 처리 단계에 대한 자동 제어 예

매크로 처리 파일 - "TSQ" 포맷

입력 영상을 다단계 처리하여 만든 리스트의 모든 내용은 화면에 있는 모든 내용을 "Dav" 형식의 단일 파일에 저장이 가능하다. 이런 저장 기능이외에도 David에는 현재 처리되는 리스트에서 영상 처리 정보만(변환종류, 파라미터들)을 따로 뽑아 파일로 저장했다가 새로운 영상에 이전의 처리 결과를 일괄적으로 적용할 수 있도록 하는 영상처리에 대한 매크로 기능을 구현하기 위해 "TSQ" 형식을 지원한다. 이 파일 포맷은 현재 처리되고 있는 다단계의 처리과정 상에서 사용되었던 영상처리 루틴에 대한 id, 파라미터를 순차적으로 기록하여 디스크에 저장한다. 이 기능은 이미 몇몇의 개발 툴에서도 매크로 명령어 (내부 라이브러리를 이용한 프로그래밍)를 구현하는 형식으로 제공하고 있기는 하다. 그러나 David은 그 기능 상 기존의 매크로 기능과는 달리 현재 작업창에서 확인되는 처리결과를 데이터 흐름에 맞추어 저장/반복이 가능하기 때문에 보다 유용하고 확장된 면을 보인다 (그림 7). 그림 7의 (a)에서는 현재 입력된 영상에 대한 4단계로 처리한 결과와 이런 결과를 얻는데 사용되었던 처리에 대해서 결과 영상을 제외하고 (변환방법, 사용 파라미터) 등에 대한 정보만을 현재 리스트에서 추출하여 파일의 형식으로 저장하는 과정을 보여준다. 이 과정을 거치게 되면 현재 리스트를 만들 때 사용되었던 처리 정보가 파일로 저장된다. 그림의 (b),(c)는 새로운 입력 영상에 대해서 저장했던 매크로 처리 파일을 적용한 결과를 보여준다. 새로운 입력 영상에 대해서 이전의 4단계 처리를 그대로 적용하기 위해서는 사용자는 단순히 이전에 저장했던 매크로 파일을 읽어서 실행만 시키면(그림 7-b), 이전의 처리과정을 다른 영상에 적용한 결과를 바로 얻을 수 있는 것이다 (그림 7-c).

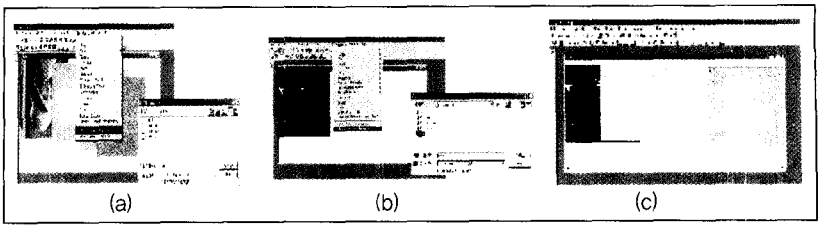


그림 7 매크로 가능 예

결 론

이번 글에서는 영상처리 응용 시스템 개발을 지원하는 일반화된 SDK를 구성하는데 필요한 자료구조(레이어 및 리스트)와 이들 자료 구조의 저장/제어 방법에 대해서 간략히 소개하였다. 또한 개발된 자료구조를 이용하여 실제 실재 영상 획득, 영상의 다단계처리, 인식을 종합적으로 지원하기위해 개발된 "David"의 기능을 소개하였다.

레이어 및 리스트 구조의 개발은 영상처리에서 사용되는 여러 영상들의 관리를 용이하게 해주기 때문에 효율적인 영상처리 SDK의 구성을 가능케 하였다. 또한 현재 처리되고 있는 영상들에 대한 모든 처리과정을 단일 파일에 저장시킴으로써 개발과정자체를 기록할 수 있도록 하였다. 이 방법의 개발로 인하여 영상처리 응용기술 개발에 대한 효율성을 증가시키는 계기가 되었다. 즉, 지금까지 구현된 적이 없었으나 영상처리 SDK에 필수적인 기술이었던 입력영상과 처리된 영상간의 관련성 문제를 제안된 레이어 및 리스트를 통해 해결함으로써 부가적으로 구현된 기능이 자동 연관성 제어이다. 획득된 영상을 여러 단계로 처리한 결과들 중간에 새로운 단계를 삽입하거나 임의의 단계를 수정할 경우, 삽입되거나 수정된 단계 이후의 내용을 자동적으로 다시 계산해주는 이런 기능은 개발자로

하여금 자유롭게 처리 단계를 변화시키거나 삽입/삭제시킬 수 있도록 한다.

이외같이 본 해설에서는 지금까지 연구/개발된 적이 없는 독창적이고 새로운 형태의 영상처리 SDK를 만드는데 필요한 핵심기술과 실제 구현된 S/W를 통해 영상처리 SDK에 대한 필요성이 증가하는 현재의 추세에 부합하고, 아울러 새롭고 획기적인 개념을 바탕으로 한 영상처리 응용시스템 개발 SDK 구성에 필요한 독창적인 기술을 소개하는 것이다.

David에 대한 보다 자세한 사항은 개발팀에 문의하실 수 있습니다.
http://www.bioinfominer.com Tel : 064+702+1832

[참고문헌]

[1] Landgrebe, D. A. 1997. On Progress Toward Information Extraction Methods for Hyperspectral Data. SPIE 42nd Annual Meeting, San Diego CA.

[2] 장동혁. 2001. 디지털 영상처리의 구현. 정보게이트.

[3] Tan, T. N. 1998. Rotation Invariant Texture Features and Their Use in Automatic Script Identification. PAMI, 20(7). 751-756.

[4] Koehler, G. 1998. Biometrics: A case study- Using Finger Image access in an Automated branch. Proc. of CTST' 98, 1. 535-541.

[5] Daugman, J. 1998. Phenotypic versus genotypic approaches to face recognition. Springer-Verlag. 108-123.

[6] Daugman, J. 1999. Biometric decision landscapes. Univ. of Cambridge Computer Lab.

[7] 문지현, 안도성, 김학일, 2001. 지문 인식 시스템 성능 평가를 위한 플랫폼 구현. 2001년 정보과학회 춘계 학술대회 논문집, 28(1). 601-603.

[8] Wayman, J. L. 2000. National Biometric test center collected works. San Jose state Univ.

[9] Mathwork Inc. 2002, Mathtool.net, http://www.mathtool.net.

[10] Adobe. 2002. Adobe PhotoShop 7.0 Manual. Adobe Inc.

[11] Matthew C. 2002. The HIPS package, http://www.permutationcity.co.uk

[12] Konstantinides K. and Rasure, J. R. 1994. The Khoros Software Development Environment for Image and Signal Processing. IEEE Trans. Image Processing, 3(3). 243-252.

[13] XITE; 2002. http://www.ifi.uio.no/forskning/grupper/dsb/Software/Xite/

[14] Argiro, D. and Rasure, J. An X windows based application programming system. in Proc. Xhibition 92: A Window on Distributed Computing(San Jose).

[15] Dyer, D. S. 1990. A dataflow toolkit for visualization. IEEEE Comp. Graphics Applications, 10(4). 60-69.

[16] Upson. C. 1989. The application visualization system : a computational environment for scientific visualization, IEEE Comp. Graphics Applications. 30-42.