# 모바일 장치를 위한 미래 적응형 장치 인터페이싱 기술의 개발

권오철[1O] 김주성[1] 이창건[1)]
서울대학교 컴퓨터공학부[1]
ockwon@rubis.snu.ac.kr, jskim@rubis.snu.ac.kr, cglee@snu.ac.kr

# The Development of Future Adaptive Device Interfacing Technology for Mobile Device

Oh-Chul Kwon[1O] Joo-Sung Kim[1] Chang-Gun Lee[1]
The School of Computer Science and Engineering, Seoul National University[1]

**요 약**

다가오는 유비쿼터스 세상의 모바일 장치는 스마트폰의 자체의 기능을 넘어서 사용자 주변에 있는 다른 IT 장치와의 융합 서비스를 제공하는 미래의 핵심 생활 기반으로 발전하게 될 것이다. 모바일 장치에서 사용되고 있는 Legacy 표준 디바이스 인터페이스에는 많은 기능적 제약이 따르며 모바일 장치의 사용자 인터페이스가 가지는 물리적 한계를 극복하고 미래에 등장하게 될 새로운 사용자 인터페이스 장치와의 자유로운 연동을 위해서는 새로운 형태의 장치 인터페이스의 개발이 필요하다. 본 논문에서는 안드로이드 모바일 플랫폼 환경을 기반으로 진행 중인 미래 적응형 장치 인터페이싱 기술에 대한 연구를 설명하고 향후 연구 방향에 대해 기술한다.

## 1. 서 론

모바일 장치는 인류가 만들어낸 도구 중 가장 빠른 속도로 발전을 거듭해 나가고 있다. 3세대에서 4세대로 넘어가고 있는 고속 이동통신 기술의 발전과 더불어 최근 스마트폰의 급속한 보급을 통해 모바일 기술은 새로운 전기를 맞고 있다. 다양한 형태의 장치로 개발되어 저마다의 고유한 기능들을 제공하던 휴대형 장치들은 이제 하나의 통합된 모바일 장치의 형태로 발전하여, 이제 사용자들은 스마트폰 하나로 대부분의 기능들을 누릴 수 있게 되었다. 하지만 가까운 미래에서는 이런 기능적 통합 과정마저도 한계에 다다를 것으로 예상되고 있다. 그 이유는 바로 모바일 장치가 가지고 있는 물리적인 제약에서 비롯된다.

모바일 장치가 인간과 가장 가까운 전자 장치로 발전하게 된 가장 큰 이유는 바로 휴대성이다. 작고 가벼운 모바일 장치는 언제 어디서나 인간과 함께하며 가장 긴밀한 정보 소통을 하고 있다. 하지만 사용자 인터페이스 장치로써의 모바일 장치는 결코 훌륭한 기기가 아니다. 모바일 장치의 작은 LCD 화면과 키패드 또는 터치스크린과 같은 입력장치들은 많은 물리적 한계를 가지고 있어서 사용자 인터페이스 구현에 있어서 많은 제약을 가져오기 때문이다.

따라서 미래의 모바일 장치는 이러한 한계를 극복하고 사용자에게 보다 나은 서비스를 제공하기 위해 사용자 주변의 사용자 인터페이스 장치와의 융합 서비스를 제공하는 모바일 융합 장치로 발전하게 될 것이다. 그리고 이러한 IT 융합 서비스들은 미래 유비쿼터스 세상의 생활 기반이 될 것이다.

그러나 빠른 하드웨어 분야의 발전에도 아직 휴대 단말기기 환경에서 제공되고 있는 사용자 인터페이스와 이를 활용한 응용 프로그램들은 2G 환경에서 크게 발전하지 못하고 있다. 기존의 UI 디바이스 표준 인터페이스를 통한 디바이스 Plug&Play 기능은 Legacy 표준 디바이스 인터페이스를 지원하는 장치에만 국한되어 기술적으로 진보된 디바이스가 존재함에도 Legacy 표준 디바이스 인터페이스의 제약으로 인해 진보된 기능의 탑재와 활용이 제한되어 왔기 때문이다. 또한 응용 프로그램에서도 Legacy 표준 디바이스 인터페이스만을 사용하여 미래의 장치의 진보된 기능을 사용하는데 많은 제약이 따를 수밖에 없다. 이와 같은 문제점을 해결하기 위해 Legacy 표준 인터페이스를 기반으로 개발된 기존의 응용 프로그램이 미래 UI 장치를 활용하게 할 뿐 아니라

1) 교신저자임

※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2010-C1090-1011-0008)
※ 이 연구를 위해 연구 장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다.

미래 UI 장치에 특화된 새로운 인터페이싱 기술까지 기존 디바이스 인터페이스 프레임워크에 수용하는 미래 적응형 장치 인터페이싱 기술을 개발하는 것이 본 연구의 최종 목표이다.

미래 적응형 장치 인터페이싱 기술은 현재의 Legacy 표준을 지원하는 다양한 사용자 인터페이스 장치뿐만 아니라 미래의 보다 발전된 형태의 사용자 인터페이스 장치의 기능까지 지원하는 진화 가능한 장치 인터페이스로써 본 연구를 통해 미래의 휴대 단말 장치와 다양한 사용자 환경의 주변 장치와의 융합 서비스를 위한 기반 기술을 확보하고 이를 적용한 다양한 융합 서비스를 개발하고자 한다.

## 2. 관련 연구

작은 화면과 불편한 입력 장치를 가진 모바일 장치의 물리적 한계를 극복하기 위해 외부의 사용자 인터페이스 장치를 활용하는 형태의 연구는 다양한 분야에서 서로 다른 방향으로 활발히 진행되어 오고 있다.

[2, 5] 에서는 모바일 장치에 모션 센싱 기술을 접목한 연구가 진행되었다. 이를 이용하면 사람의 동작을 인식하여 마우스 포인터의 제어나 간단한 형태의 입력이 가능하다. 또한 [8, 10] 에서는 모바일 장치와 가전기기들과의 통신을 통해 실내 환경을 제어하고자 하는 연구가 진행되었다. 하지만 이러한 연구들은 모바일 장치가 단순히 원격 조정 장치의 형태로만 활용되었을 뿐, 모바일 장치에 더 나은 외부의 사용자 인터페이스 장치를 융합한 형태의 기술은 아니다.

[3] Samsung 의 Instinct HD Phone 이나 [4] Apple 의 iPhone 에서는 TV-OUT 기능을 통해 모바일 장치의 작은 화면을 극복하고자 하는 시도를 하였다. 또한 [6, 7] 에서는 빔 프로젝터를 활용하여 모바일 장치의 작은 화면을 큰 스크린에 보이도록 하였다. 하지만 이러한 시도들도 단순히 화면이 확대되었을 뿐 그 화면에 보이는 내용 자체의 변화는 없다. 더 나은 화질과 해상도를 지원하는 외부의 디스플레이 장치가 있다고 해도 모바일 장치만을 고려하여 설계된 모바일 플랫폼과 응용 프로그램에서는 이를 인식하지 못하여 지원할 수 없기 때문이다.

## 3. 레거시 디바이스 인터페이스의 구조

이 절에서는 모바일 장치에서 일반적으로 사용하고 있는 임베디드 리눅스에서의 저수준 파일 입출력 함수를 통한 디바이스 파일 접근과 하드웨어 제어 방식에 대해 요약하여 기술하고 이러한 방식의 문제점에 대해서 설명한다.

[그림 1]과 같이, 응용 프로그램은 시스템에 준비된 함수나 프로그램 자체적으로 선언된 함수를 사용하여 특정한 기능을 하는데 이러한 응용 프로그램은 사용자 공간에서 프로세스 형태로만 동작하므로 직접적으로 하드웨어를 제어할 수 없다. 따라서 커널 내의 파일 시스템 구조에 의해 하드웨어를 제어하는 디바이스 드라이버 함수와 연결된 디바이스 파일이라는 특수한 파일을 사용한다. 즉, 응용 프로그램은 하드웨어를 제어하기 위해 저수준 파일 입출력 함수를 사용해 디바이스 파일에 데이터를 읽거나 씀으로써 디바이스 드라이버 함수가 호출된다. 이러한 디바이스 드라이버는 커널 내부에 구현되어 있거나 모듈 형태로 커널에 적재된다.

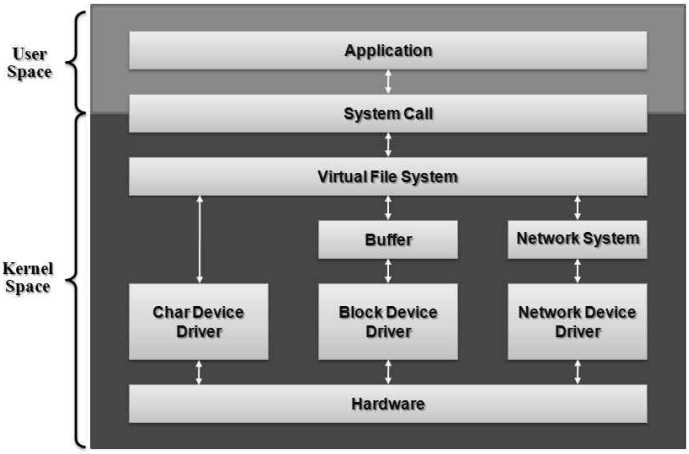


그림 1. Legacy 디바이스 드라이버의 구조

커널은 디바이스 파일에 기록된 디바이스 타입과 주번호를 이용해 커널 내에 등록된 디바이스 드라이버 함수를 연결한다. 커널에는 디바이스 드라이버를 관리하는 구조체가 있고, 그 구조체는 또다시 file_operation 구조체를 필드로 포함하고 있다. 이를 정리해보면 응용 프로그램에서는 디바이스 파일을 열어 타입 정보와 주 번호를 얻어오고 커널에서 관리하는 장치 배열에 등록된 디바이스 드라이버의 인덱스를 얻는다. 이제 이 인덱스 값을 이용하면 장치 배열에서 앞서 언급한 file_operation 구조체의 주소를 얻을 수 있다. 이 file_operation 구조체에 주목할 필요가 있는데, 이 구조체에는 디바이스 드라이버가 디바이스 드라이버를 등록하는 함수를 사용하여 저수준 파일 입출력에 대응하는 함수를 설정한 내용을 담고 있다. 또한 이 구조체를 통해 응용 프로그램의 저수준 파일 입출력 함수와 디바이스 드라이버의 함수가 1:1로 연결되어 동작한다.

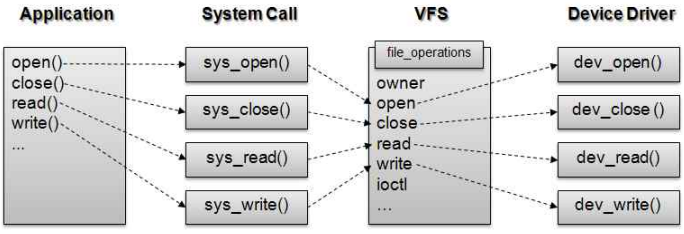


그림 2. 디바이스 드라이버의 함수 호출 과정

위의 [그림 2]와 같이, 응용 프로그램과 디바이스 드라이버는 file_operation 구조체를 통해 연결된다. 응용 프로그램이 저수준 파일 입출력 함수를 사용하여 디바이스 파일에 접근하면 커널은 등록된 문자 디바이스 드라이버의 file_operation 구조체의 정보를 참고하여 디바이스 파일에 접근한 함수에 대응하는 함수를 호출한다.

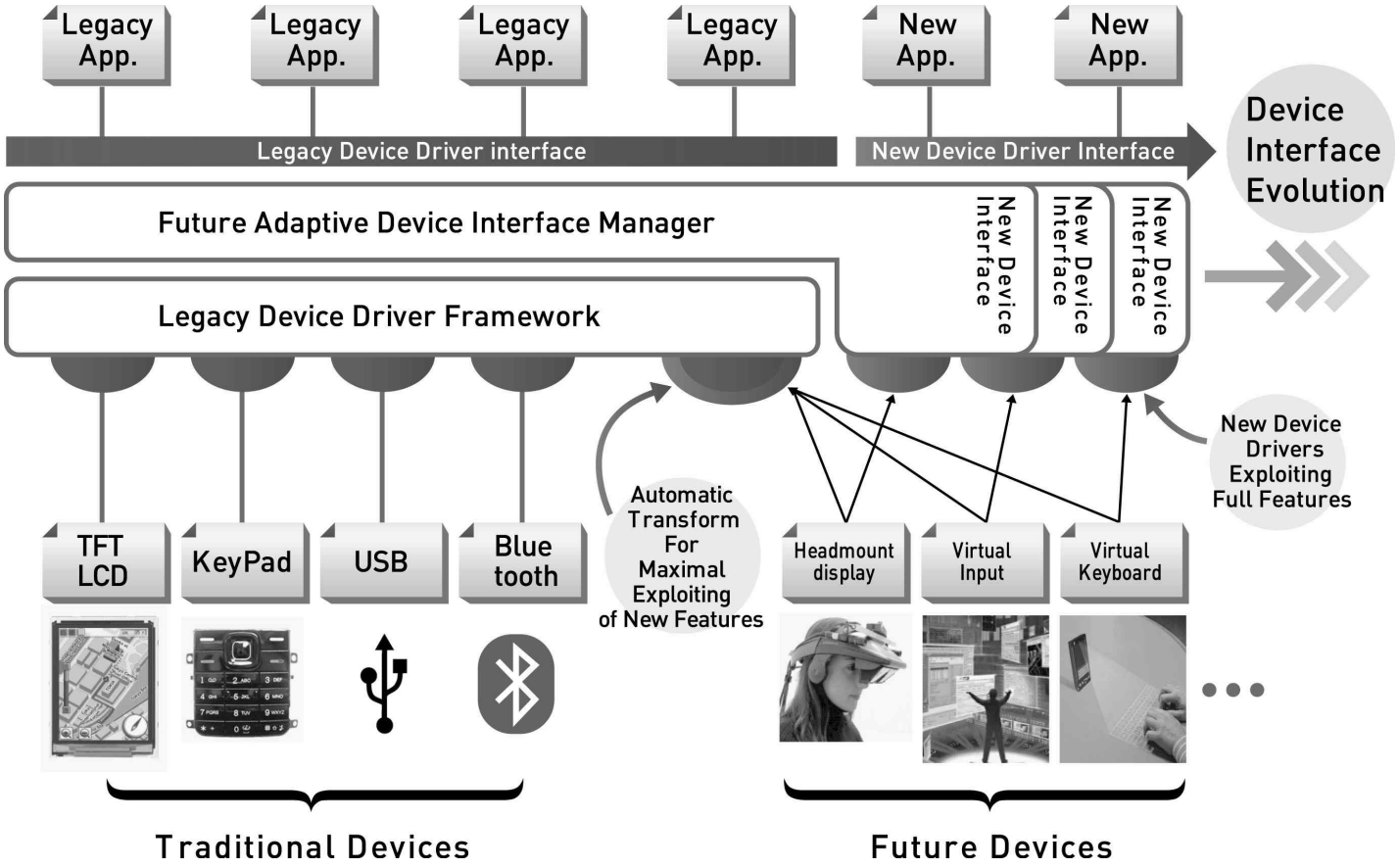


그림 3. 미래 적응형 장치 인터페이스의 구조

file_operation 구조체는 소유자를 나타내는 owner 필드를 제외하면 모두 함수 포인터이며 관습적으로 사용되는 필드를 제외하고 NULL 포인터로 할당한다. “xxx_”로 시작되는 함수는 file_operation 에 대응하는 디바이스 드라이버 함수를 의미한다.

리눅스에서는 이러한 디바이스 제어 방식에는 몇 가지 문제점들이 있는데 가장 큰 문제점은 Plug&Play 가 Legacy 표준 디바이스 인터페이스에 맞춰 개발된 디바이스에만 국한된다는 점이다. 또한 정해진 함수의 영역 내에서 모든 기능을 처리해야하는 제약 조건 때문에 디바이스 드라이버 개발자들은 제한된 함수만을 사용하여 디바이스를 제어할 수 있도록 구현해야만 하며, 새로운 기능들을 활용하기 위해서는 별도로 만들진 응용 프로그램에서 데이터를 재가공하여 디바이스 드라이버를 활용하여야 한다. 이러한 제약 사항은 모바일 장치에서 미래의 사용자 인터페이스 장치를 사용하는데 있어서 가장 큰 장해 요소이며, 미래의 사용자 인터페이스의 장치까지도 수용할 수 있는 새로운 형태의 장치 인터페이스의 연구의 가장 큰 동기가 되었다.

## 4. 미래 적응형 장치 인터페이스

### 4.1 연구 목적

기존 사용자 인터페이스 장치의 표준 인터페이스를 통한 장치 Plug&Play 기능만으로 다양한 사용자 인터페이스 장치를 지원하기에는 현실적인 어려움이 따른다. 기존의 장치 Plug&Play 기능은 Legacy 표준 장치 인터페이스를 지원하는 장치에만 국한되어 있어서 기존의 장치 인터페이스에서 지원하지 않는 기능을 탑재한 기술적으로 진보된 장치가 있다 해도 이러한 진보된 기능의 탑재가 제한되기 때문이다. 또한 응용 프로그램이 Legacy 표준 장치 인터페이스만을 사용할 때 새로운 디바이스의 진보된 기능을 사용하는데 많이 제약이 따른다.

이러한 문제를 극복하기 위해, Legacy 표준 장치 인터페이스를 대상으로 개발된 기존의 응용 프로그램이 미래의 진보된 사용자 인터페이스 장치를 지원할 수 있도록 진보된 미래 UI 장치의 Full feature를 제공하는 새로운 인터페이스 정의 기술할 뿐 아니라 진보된 장치에 특화된 새로운 인터페이스 기술까지 기존 디바이스 인터페이스 프레임워크에 수용하는 미래 적응형 장치 인터페이스를 개발하고자 한다.

### 4.2 미래 적응형 장치 인터페이스의 기능

#### 4.2.1 자동 장치 인터페이스 변환 기술 ( Automatic Device Interface Transform Technique )

위 그림 3과 같이, 기존의 표준 장치 인터페이스를 기반으로 개발된 응용 프로그램들이 진보된 미래의 사용자 인터페이스 장치에 연결될 수 있도록 하고, 진보된 기능들을 최대한 활용할 수 있도록 하기 위해서는 자동 장치 인터페이스 변환 기술이 필요하다. 또한 미래의 진보된

사용자 인터페이스 장치의 기능을 Legacy 표준 인터페이스에 맞게 추상화하여 응용 프로그램의 I/O 요구에 맞게 프로파일에 기반을 둔 진보된 기능으로의 I/O 자동 확장이 필요하다.

4.2.2 장치 인터페이스 진화 기술 ( Device Interface Evolution Technique )

진보된 사용자 인터페이스 장치의 모든 기능을 제공하기 위해서는 새로운 장치 드라이버 인터페이스를 장치 인터페이스에 수용하는 장치 인터페이스 진화 기술이 요구된다. 이 기술에는 세부 기술에는 진보된 미래 사용자 인터페이스 장치의 모든 기능을 제공하는 새로운 인터페이스를 정의하고 이 인터페이스에 맞게 사용자 인터페이스 장치를 추상화하는 기술이 포함된다. 그리고 진보된 미래 사용자 인터페이스 장치를 위한 장치 드라이버를 설계, 구현할 수 있는 프레임워크 및 새로운 드라이버 모듈의 삽입하는 기술이 요구된다. 또한 진보된 새로운 인터페이스를 새로운 응용 프로그램 개발에 직접 제공하기 위한 Legacy 소프트웨어 플랫폼 바이패싱 기술이 포함된다.

### 4.3 연구 진행

4.3.1 연구 환경 : 안드로이드 모바일 플랫폼

[1]2007년 발표한 구글의 오픈 소스 모바일 소프트웨어 플랫폼 안드로이드 모바일 플랫폼은 오픈 소스라는 장점뿐만 아니라 리눅스 커널을 기반으로 한 모바일 장치를 위한 기반 플랫폼으로서 휴대폰과 다양한 IT 장치에 이식할 수 있는 개방성을 갖추고 있다. 안드로이드 플랫폼의 기반은 오픈소스 운영체제인 리눅스이므로 공개된 안드로이드의 소스코드를 통해 개발자는 매우 큰 자율성과 권한을 확보할 수 있으며 대부분의 모바일 응용 개발자에게 익숙한 자바 언어를 사용하기 때문에 응용 구현을 위해 자바 클래스 형태의 풍부한 API를 활용할 수 있다. 또한 안드로이드 플랫폼은 적합한 계층 구조와 모듈들을 가지고 있다. 가장 하위의 리눅스 커널 레이어에서부터 최상위의 어플리케이션 프레임워크 레이어까지 각 계층 간의 역할들과 기능들이 잘 분리되어 있으며 안드로이드를 통해 개발된 기술의 모듈화를 가능하게 한다.

하지만 안드로이드의 소스 코드는 매우 방대한 크기를 가지고 있으며, 안드로이드의 특정한 부분으로 한정한다 할지라도 이를 분석하고 이해하는 작업은 쉽지 않은 일이다. 따라서 장치 인터페이스와 관련된 안드로이드의 소스코드 분석은 미래 적응형 장치 인터페이스 연구 개발 과정 내내 진행될 것이다.

4.3.2 진행 경과

[9]앞선 연구에서는 기본적으로 실행 시간에 동적으로 화면의 해상도를 변경할 수 없는 구조를 가지고 있는 안드로이드와 임베디드 리눅스 커널을 개선하기 위해 구조 분석을 통해서 안드로이드 플랫폼의 Window Manager에서 리눅스 커널의 프레임버터에 이르기까지의 정확한 화면 데이터의 이동 경로를 파악하고, 그 구조를 변경하거나 모듈을 추가하여 동적으로 해상도 변경이 가능하도록 하여, 사용자 인터페이스 자동 전환 기술을 가능하게 하는 연구를 수행하였다. 이러한 연구를 통해 안드로이드를 기반으로 하는 모바일 장치에서 화면을 외부의 디스플레이 장치로 전환할 수 있는 기술을 개발하였고 이를 실제 모바일 서비스에 적용하는 단계에 이르렀다. 안드로이드의 Wifi 또는 Bluetooth를 통해 검색된 주변 사용자 인터페이스 장치의 프로파일을 분석하여 휴대단말 장치의 환경에 맞는 서비스 수준을 도출하고 자동적으로 사용자 인터페이스 장치의 전환이 이루어지게 된다.

4.3.3 연구 계획

현재 연구는 외부 디바이스 장치를 활용한 사용자 인터페이스 전환 기술을 접목하여 미래 적응형 장치 인터페이싱 기술 개발에 필요한 설계와 표준화 작업을 진행중에 있다. 연구의 우선적인 초점은 Legacy 디바이스 프레임워크에서 기존에 지원하지 않았던 새로운 기능을 가진 사용자 인터페이스 장치를 지원하기 위한 기술인 Automatic I/O Transform 기술 개발에 있으며 진보된 미래 사용자 인터페이스 디바이스의 성능을 최대로 활용하면서도 응용 프로그램의 사용자 인터페이스 장치의 의존성을 숨기는 미래 디바이스 추상화 기술에 대한 연구도 진행하고 있다.

## 4. 결 론

모바일 장치를 위한 미래 적응형 장치 인터페이스에 대한 연구를 통해 모바일 장치의 유한한 사용자 인터페이스의 공간적 한계를 극복하고 사용자의 위치와 관계없이 모바일 장치 하나만으로 다양한 IT 융합 서비스가 가능해질 것으로 기대한다. 이러한 기술적 적응성과 수용력은 모바일 장치와 연결할 수 있는 현재의 사용자 인터페이스 장치뿐만 아니라 미래에 등장하게 될 사용자 인터페이스 장치의 개발을 촉진하고 차세대 장치 인터페이싱 기술로써 다양한 IT 기기들이 동적으로 선택되고 결합되고 확장될 수 있는 융합 기술의 표준을 제시할 수 있게 될 것이다. 또한 모바일 장치를 위한 미래 적응형 장치 인터페이스 기술이라는 독자적인 IT 융합 기술 연구 분야를 개척하고, 향후 지속적인 연구를 통해 미래의 사용자 인터페이스의 장치까지도 수용할 수 있는 진화 가능한 미래 적응형 장치 인터페이싱 기술을 개발하고자 한다.

## 참고문헌

[1] Google. Android Developers. http://developer.android.com, December 2008.

[2] A. S. Shirazi, C. Winkler, and A. Schmidt. Flashlight interaction: a study on mobile phone interaction techniques with large displays. In MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services, pages 1~2, September 2009.

[3] Apple. iPod and iPhone: TV out support. http://support.apple.com/kb/ht1454, 2008.
[4] Samsung. Samsung Instinct HD Phone. http://www.samsungmobileusa.com/InstinctHD.aspx, 2009.
[5] S. Jeon, J. Hwang, G. J. Kim, and M. Billinghurst. Interaction techniques in large display environments using hand-held devices. In VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology, pages 100~103, November 2006.
[6] A. Greaves, A. Hang, and E. Rukzio. Picture browsing and map interaction using a projector phone. In Proceedings of the 10th international conference on Human computer interaction with mobile devices and services, pages 527~530. ACM, September 2008.
[7] A. Hang, E. Rukzio, and A. Greaves. Projector phone : a study of using mobile phones with integrated projector for interaction with maps. In Proceedings of the 10th international conference on Human computer interaction with mobile devices and services, pages 207~216. ACM, September 2008.
[8] L. Tarrini, R. B. Bandinelli, V. Miori, and G. Bertini. Remote Control of Home Automation Systems with Mobile Devices. In Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction, pages 364~368. Springer-Verlag, September 2002.
[9] Oh-Chul Kwon, Joo-Sung Kim, Chang-Gun Lee, Eun Yong Ha, The Development of Automatic User Interface Transform Technology on Android Mobile Platform, in KIISE 36th Fall Conference, Nov. 2009
[10] J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, and M. Pignol. Generating remote control interfaces for complex appliances. In Proceedings of the 15th annual ACM symposium on User interface software and technology, pages 161~170. Springer-Verlag, October 2002.