인쇄 기능을 추가한 DVR 재생기

박현준 배상학 공기석 나보균
한국산업기술대학교 컴퓨터공학과
{herotic, totorivers, kskong, bkna}@kpu.ac.kr

DVR Replayer with Print Function

Hyun-jun Park, Sang-hak Bae, Ki-sok Kong, Bo-kyun Na
Korea Polytechnic University

요 약

DRPF는 DVR 카메라에 의해 저장된 영상을 제어하여 사용자가 원하는 고지서 양식으로 변환한 후 프린터로 출력할 수 있게 한다. 현재 각 공공기관에서 발부되는 고지서는 따로 사진을 스캔하여 수 작업한 문서를 인쇄하는 실정이다. 본 논문에서는 미리 정해진 양식에 맞게 영상 이미지와 텍스트를 자동적으로 작성한 고지서를 인쇄할 수 있는 프로그램을 구현한다. 또한 DB와 연동하여 방대한 양의 정보를 제어하고 변환한 양식을 효율적으로 관리할 수 있다. GPL를 따르는 공개 소프트웨어를 지향하기 때문에 기업은 물론 개인도 자유롭게 이용 가능하다.

1. 서 론

현대의 사회는 급속한 인구의 증가로 인해 사람과 사람 사이에 발생하는 예측할 수 없는 사고, 사건 등은 이미 통제를 넘어선 수위에 있다. 이러한 사건, 사고를 예견하고 방지한다는 것은 어디까지나 한계가 있기 때문에 제 3자의 눈에 의해 24시간 356일 동안 감시할 수 있는 방법이 요구되어 왔다.

이런 필요성에 의해 Digital Video Record(이하 DVR)이 등장하게 되었다. DVR은 기존의 아날로그 방식의 CCTV의 이미지 선명도의 취약함과 녹화시간 제한의 한계를 극복하여 깨끗한 영상, 녹화 용량의 극대화를 성공적으로 이루었다.

본 연구의 목적은 DVR로 녹화된 영상을 감시하고 여기에 추가하여 각종 고지서들을 출력할 수 있는 DVR Replayer를 설계, 제작하는 데 있다.

현재 시중에 소개된 윈도우 기반의 DVR Replayer는 대부분 고가이고 사용자가 원하는 형태로 인쇄를 할 수 있는 제품이 전무한 상태이다. 이에 반해 본 연구 프로젝트는 리눅스 기반의 GPL(General Public License)을 따르는 공개 프로그램이기 때문에 누구나 무료로 사용, 확장할 수 있으며 각종 고지서 출력에 특화된 기능을 사용할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 GTK+를 사용한 GUI(Graphical User Interface)에 관련된 연구 사항을 소개하고 3장에서는 이 시스템의 설계와 구현에 대해 설명하였다. 4장에서는 본 논문의 결론과 기대효과, 향후 발전 가능한 방향에 대해 논의 하고자 한다.

2. 관련 연구

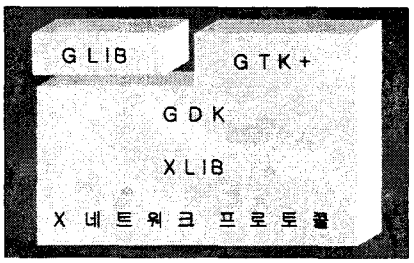
2.1 GTK+

DVR Replayer with Print Function(이하 DRPF)는 GUI 설계를 위해 GTK+ 라이브러리를 사용하였다.

GTK+(GIMP Toolkit)은 원래 GIMP(General Image Manipulation Program)을 만들기 위한 툴킷으로 개발되었다. 지금은 리눅스 기반의 GUI free software 제작에 쓰이게 되었다[1].

GTK+는 Xlib의 함수들에 대한 기본적인 wrapper인 GDK(GIMP Drawing Kit)를 기반으로 하고 있다. GTK+는 본질적으로 객체지향적인 API(Application Programming Interface) 이다. 비록 완전히 C로 쓰였지만 클래스의 개념과 callback 함수(함수에 대한 포인터)가 갖추어져 있다. 여기에 연결리스트(linked list)를 다루기 위한 몇 가지 함수들을 포함하고 있는 glib를 추가적으로 사용할 수 있다. 예를 들어 g_strerror()와 같은 함수들은 다른 유닉스들에 대해 표준이 아니거나 쓰일 수 없지만 GTK+에서는 이식성을 높이기위한 대체 함수들로 쓰일 수 있다. 또한 g_malloc이 디버깅 가능을 강화했듯이 libc의 버전에 보강이 이루어지기도 했다[2].

아래 그림 1은 GTK+를 포함한 gnome 기반의 그래픽 관련 라이브러리들을 보여주고 있다.



(그림1) Gnome 라이브러리 구성도

2.2 GdkPixbuf

GTK+ 는 스크린을 업데이트 하기위해서 이중 버퍼링을 사용하므로 좀더 부드럽게 렌더링할 수 있다.

DRPF에서 그래픽을 표현하기 위해 GdkPixbuf 라이브러리를 사용하였다. GdkPixbuf는 GTK+의 일부분이다. GdkPixbuf는 GIF, PNG, JPEG, BMP 등의 다양한 포맷으로 이미지 로딩과 저장을 핸들한다. GdkPixbuf가 GTK+의 일부이기 때문에, GTK+ 내에서 GdkPixbuf 라

이브러리를 사용할 수 있다. 또한 메뉴, 툴바, 버튼 등의 "alpha-blended" 아이콘을 렌더링하는데 사용할 수도 있다[1].

2.3 Pango

Pango는 새로운 레이아웃용 프레임웍이며 국제화된 텍스트의 렌더링을 위한 라이브러리이다. Pango로 인해 전 세계적으로 텍스트의 표준을 가져올 수 있었다. Pango는 왼쪽에서 오른쪽 이외의 방향으로 흐르는 텍스트를 핸들할 수 있고 복잡한 언어도 쉽게 관리할 수 있다. 또한 사용되는 콘텍스트에 따라 다른 형식을 취하는 문자들도 핸들링한다. 그리고 양방향 텍스트를 지원하여 왼쪽에서 오른쪽, 오른쪽에서 왼쪽 두 가지를 혼합할 수 있다 Arabic과 Tamil 같은 다양하고 복잡한 스크립트를 쉐이핑(shaping) 하기위한 풀러그인이 있다. 안티 앨리어스 텍스트의 렌더링 뿐만 아니라 Xft와 XRender 확장도 핸들한다[3].

2.4 CUPS

CUPS(Common Unix Printing System)는 사용자의 네트워크에 범용 프린팅 솔루션을 제공한다. 그동안 Unix/Linux의 프린팅 문제에 대한 해결책이 없었다. 구식의 lpd(Line Printer Daemon)를 사용해 왔으며, IPP(Internet Printing Protocol)을 지원하지 못하였고 여러 개의 프린터를 사용할 수도 없었다. 그러나 CUPS로 효율적이고 신뢰성 있는 방식의 프린팅 관리가 가능해졌다. CUPS는 IPP를 지원하며, LPD, SMB(Server Message Block: Microsoft Windows에 부착된 프린터), JetDirect에 대한 인터페이스를 가지고 있다. 또 네트워크 프린터 브라우징을 제공하며 PPD(PostScript printer description) 파일을 사용한다[4].

3. 구 현

3.1 개발환경

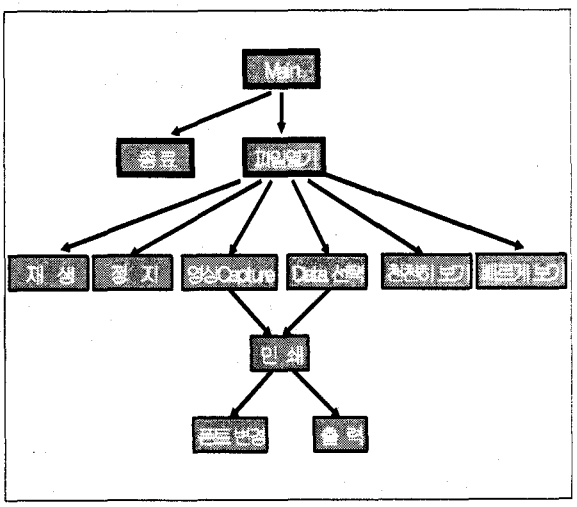
DRPF는 리눅스(Redhat Fedora core 4) 환경에서 vi를 사용 하였고[5], GTK+ 기반의 GUI 설계를 위해서 glade GUI 제작 툴을 사용하였다[6]. DB와 연동을 위해 MySQL을 사용하였다[7]. 디버깅은 gdb를 사용하였다. 아래 표 1은 DRPF를 개발하기 위해 사용한 컴퓨터 사양이다.

(표 1) DRPF 제작 컴퓨터 사양

종 류	품 명
CPU	AMD Athlon 64 Processor 3000+(1.8GHz)
MainBoard	Unitec NF4U
RAM	1GB PC3200(DDR400)
Graphic Card	NVIDIA Geforce 7800 GT

3.2 소프트웨어 구성도

DRPF는 GTK+ 쓰레드 기반으로 구현되어 있다[1]. 사용자가 버튼을 클릭하면 시그널이 발생하고 시그널에 대한 동작이 구현되어 있는 콜백함수를 수행하여 사용자가 원하는 동작을 수행하게 된다[8]. 아래 그림 2는 DRPF를 구현하기 위한 소프트웨어 구성도이다.



(그림 2) DRPF 소프트웨어 구성도

3.3 영상파일 디코딩

DRPF에서 영상을 재생하기 위해서는 DVR 카메라에 의해 녹화되어 H.263으로 인코딩된 영상파일을 디코딩한다. 디코딩을 초기화하기 위해 프레임을 저장하기 위한 버퍼를 설정한다[9].

압축된 영상파일을 디코딩하면 YUV411 형식의 이미지가 추출되고 이를 RGB24로 변환한다. 그 후 각각의 프레임들을 GTK의 DrawingArea Widget에 연속적으로 출력하여 영상을 재생한다[10].

3.4 GUI 구현

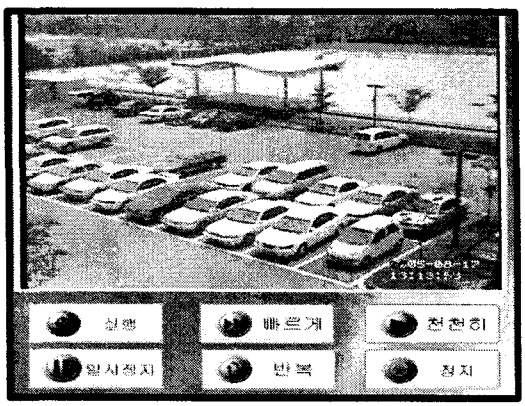
H.263으로 디코딩된 영상 파일을 재생하고 이에 관련된 정보를 제어하기 위한 GUI를 구현하기 위해 GTK+ 라이브러리를 사용한다. 최상위의 Window Widget 아래에 hbox,vbox Widget으로 레이아웃을 설정한 다음 각각의 Drawingarea, button, entry, tree Widget 들이 배치된다.

영상 파일에서 디코딩된 하나의 프레임은 pixbuf라는 버퍼에 저장되어 GTK+의 drawingarea Widget에 출력된다. 아래 그림 3은 하나의 프레임을 drawingarea Widget 에 출력하는 소스 코드이다.

```
gbooleanon_expose_event(GtkWidget *widget,
GdkEventExpose  *event, gpointer user_data)
{
 GdkPixbuf *dapix;
 GdkPixbuf *imagedest;
 gdk_window_clear (widget->window);
 if(dapix != NULL){
  imagedest = gdk_pixbuf_new
        (GDK_COLORSPACE_RGB,FALSE,
        8,320,240);
  gdk_draw_pixbuf(GTK_WIDGET(widget)->window,
  GTK_WIDGET(widget)->style->white_gc,
  imagedest,0,0,0,0,1,1,
  GDK_RGB_DITHER_NORMAL,0,0);
 }
 else{
  g_signal_connect (widget,"expose_event",
  G_CALLBACK (scribble_expose_event), NULL);
  g_signal_connect (widget,"configure_event",
  G_CALLBACK (scribble_configure_event), NULL);
 }
 return FALSE;
}
```

(그림 3) Drawingarea Widget에 이미지 출력

아래의 그림 4는 영상 파일을 읽어와 GTK+로 구현된 GUI 환경에서 영상을 재생하는 그림이다.



(그림 4) 영상 출력 화면

3.5 영상 캡쳐 기능 구현

DrawingArea Widget에 재생중인 사용자가 원하는 하나의 프레임을 캡쳐하여 버퍼에 저장한 다음 이미지 크기를 조정한 뒤에 또 다른 Drawingarea Widget에 출력된다. 아래 그림 5는 버튼을 클릭하였을 때 위 그림 4에서 재생중인 영상에서 한 프레임을 캡쳐하여 저장하는 소스 코드이다.

```
void on_buttonNo_clicked(GtkButton *button,
gpointer user_data){
 GdkPixbuf *imagedest;
 dapix = gdk_pixbuf_get_from_drawable
 (dapix,GTK_WIDGET(drawingarea)->window,cmap,
 0,0,0,0,VDO_WIDTH,VDO_HEIGHT);

 if(dapix == NULL){
  gtk_exit(0);
 }

 if(dapix != NULL){
  imagedest = gdk_pixbuf_new
  (GDK_COLORSPACE_RGB,FALSE,8,320,240);
  gdk_draw_pixbuf(GTK_WIDGET(destarea)->window
  ,GTK_WIDGET(destarea)->style->white_gc,
  imagedest,0, 0, 0, 0, -1, -1,
  GDK_RGB_DITHER_NORMAL, 0, 0);
 }
}
```

(그림 5) 재생중인 영상에서 하나의 프레임을 저장

아래 그림 6은 4개의 Drawingarea Widget 에 캡쳐된 이미지를 표현한 그림이다.

(그림 6) 캡쳐된 영상 이미지

3.6 인쇄 기능 구현

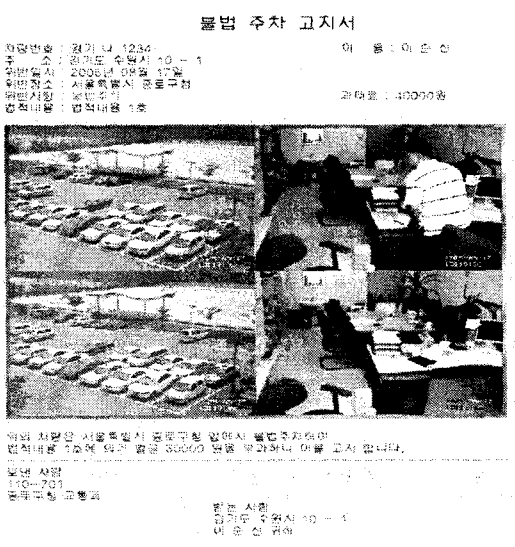
DB에서 추출한 텍스트 정보는 pango 라이브러리를 사용하고 캡쳐된 영상 이미지는 Gdkpixbuf 라이브러리를 사용하여 새로운 Window에서 고지서 형태로 가공되어 표현된다.

DRPF는 가공된 새로운 Window로 부터 jpeg 이미지로 변환하여 CUPS 라이브러리를 사용하여 인쇄한다. 아래 그림 7은 프린트 버튼을 클릭하여 버퍼에 저장되어 있는 이미지를 인쇄하는 소스 코드이다.

```
void on_printOkButton_clicked(GtkWidget *button,
GtkWidget *draw){
 GdkPixbuf *savepix;
 savepix = gdk_pixbuf_get_from_drawable
 (savepix,GTK_WIDGET(draw)->window,cmap,
 0,0,0,0,320,240);

 if(savepix == NULL){
  gtk_exit(0);
 }

 gdk_pixbuf_save(savepix,"savepix.jpg","jpeg",&error,
 "quality","100",NULL);

 cupsprint();
}
```

(그림 7) Window Widget으로부터 jpeg 이미지로 저장

아래 그림 8은 텍스트 정보와 이미지를 하나의 Window에 고지서 형태로 가공하여 표현한 모습이다.



(그림 8) 인쇄 미리보기

4. 결론

4.1 향후 연구 과제

DRPF는 영상제어, 카메라 제어, 네트워크 프린터 제어를 연구할 가치가 있다. 영상제어로는 기존의 수작업에 의한 차량번호의 식별에서 벗어나서 패턴인식 알고리즘을 적용하여 사용자의 편의를 도모할 수 있다. 또한 카메라를 고정된 방향으로만 위치시키지 않고 다른 각도에서도 감시할 수 있는 기능을 추가 할 수 있다. 마지막으로 네트워크에 연결되어 있는 프린터들을 제어하여 인쇄 기능의 확장을 추구할 수 있다.

4.2 DVR Replayer의 미래

추후 DVR Replayer의 경제적, 산업적 효과는 계속해서 증가될 것이다. 그 이유는 사회가 산업화되고 다양한 경제의 수단이 발달하면서 개인과 기업의 보안의 중요성은 비례하여 증가하기 때문이다.

DVR의 증가로 인해 현재 주로 공공기관이나 사업체에만 사용되는 DVR은 일반 가정에서도 사용되게 될 것이다. 기술의 발전에 따라 미래의 DVR Replayer는 사물을 스스로 감시, 분석하여 관련된 정보를 저장하게 될 것이다.

참고문헌

[1] Arthur Griffith, "GNOME/GTK+ Programming Bible", IDG Books Worldwide, 2000.

[2] GTK+ 공식 홈페이지, http://www.gtk.org

[3] PANGO 공식 홈페이지, http:/www.pango.org

[4] CUPS 공식 홈페이지, http://www.cups.org

[5] Neil Matthew, Richard Stones, 배재현(역), "Beginning Linux Programming 제3판", 정보문화사, 2004.

[6] GLADE 개발자 홈페이지, http://glade.gnome.org

[7] 김정환, "전산 실무에 사용하는 데이터베이스 MySQL응용방안", 홍릉과학출판사, 2005.

[8] Donna Martin, 김진구(역), "초보자를 위한 리눅스 GTK+ 프로그래밍 21일 완성", 인포북, 2000.

[9] ITU-T, "Recommendation H.263 : Video Coding for Low Bit Rate Communication", 1996.

[10] 김명호, "멀티미디어 시스템 개론", 홍릉과학출판사, 2005.