UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Vislib

Elaborato di esame di: **Alberto Nobile, Giovanni Giovannozzi, Daniele Cisamolo, Gianluca Rossi**

Consegnato il: **1 luglio 2005**

Sommario

Si sono studiati le funzionalità e l'uso delle librerie Vislib in modo da poterle utilizzare per acquisire in maniera corretta le immagini provenienti da una telecamera con collegamento USB. Successivamente con l'ausilio di due applicativi presenti nelle librerie abbiamo verificato il corretto funzionamento del pacchetto.

1. Introduzione

Il pacchetto VisLib utilizzato per l'elaborato è liberamente scaricabile dal sito ActivMedia [1]. Tale libreria è disponibile unicamente per sistemi Linux e consta di una serie di strumenti per l'acquisizione di immagini tramite una webcam. Lo standard di riferimento utilizzato dal sistema per la comunicazione con la telecamera è il noto "video for linux" (v4l [2]) che offre alcuni tool utili per la manipolazione delle immagini acquisite.

Obiettivo dell'elaborato è verificare il corretto funzionamento delle librerie Vislib con una telecamera USB marcata Philips. Il fine ultimo è la gestione della visione su un robot.

2. Il problema affrontato

Dopo aver caricato il modulo corretto per il funzionamento della telecamera, si è eseguito il programma extractionHSI, uno dei numerosi programmi d'esempio forniti con le librerie.

Avviato il software l'errore riscontrato è stato:

```
VisLib 1.8.1
detected TrueColor visual (depth 24)
MaxHeight 480
MaxWidth 640
Channels 1
ioctl (VIDIOCGCHAN) failed
Window height 480 X 0
Window width 640 Y 0
Frames quantity: 2
VIDIOCMCAPTURE-Init-Frame#0: Invalid argument
fatal error - vlGrabInit () failed
```

Nell'ultima riga si può notare la presenza di una chiamata alla funzione vlGrabInit() che fallisce l'esecuzione. Si è partiti da tale errore per indagare sulle cause del malfunzionamento.

Il file contenente la funzione in esame è grab.c situato nella directory *vislib/src*, deputato all'acquisizione delle immagini. vlGrabInit() ha il compito di inizializzare il framegrabber impostando la risoluzione, il formato delle immagini da acquisire e tutti i parametri essenziali per il funzionamento, per questo motivo ogni programma che si appoggia alle vislib invoca tale metodo per avere accesso alla webcam.

Per impostazioni predefinite, il formato adottato dal software è RGB32, mentre, dopo alcuni test, si è scoperto che la telecamera in nostro possesso è in grado di acquisire le immagini unicamente in formato YUV 4:2:0 [3] altrimenti noto come YV12. Questo modello di webcam si appoggia al modulo *pwc* che basa il suo funzionamento sullo YUV, scomodo per i nostri scopi.

3. La soluzione adottata

Il primo passo è stato verificare i vari formati messi a disposizione dalle librerie v4l sulle quali vislib si appoggia, tra i tanti si è rinvenuto VIDEO_PALETTE_YUV420P, conforme con lo spazio dei colori utilizzato dalla nostra telecamera. A questo punto si è presentato un nuovo inconveniente, l'immagine risultante veniva suddivisa in quattro fotogrammi identici in bianco e nero.

Il secondo passo è stato creare una nuova funzione di conversione che facesse al caso nostro poiché le funzioni presenti all'interno del pacchetto non fornivano la trasformazione da YUV a RGB.

Modificando il codice e inserendo la trasformazione, il risultato ottenuto è stato soddisfacente.

4. Modalità operative

È stato modificato la funzione vlGrabInit() in modo che inizializzasse il framegrabber in modo che acquisisse le immagini nel formato YV12.

```
//v_mmap.format = VIDEO_PALETTE_RGB32;              /* RGB 32 in 4 bytes: B,G,R,NULL */
 v_mmap.format = VIDEO_PALETTE_YUV420P;
```

Come si può notare è stata commentata la linea che impostava il framegrabber per acquisire le immagini in formato RGB32 ed è stata aggiunta una linea con l'impostazione corretta.

Per recuperare la funzione di conversione ci si è serviti del codice sorgente di Xine, un ottimo riproduttore multimediale disponibile per Linux. Il codice era liberamente disponibile su un sito dedicato agli svilupppatori[4]

Di seguito viene riportata la funzione originale che acquisisce l'ultima immagine catturata dal framegrabber:

```
int
vlGrabLastImage (vlImage *pic)
{
 int i=0;
 int row,col;
 vlPixel *data;
 unsigned char *p;

 if (!pic) {
  VL_ERROR ("vlGrabLastImage: error: NULL image\n");
  return (-1);            /* failure */
 }

 /* initialize the new picture */
 if (0 > vlImageInit (pic, RGB, grab_cols, grab_rows)) {
  VL_ERROR ("vlGrabLastImage: error: could not initialize image\n");
  return (-1);            /* failure */
 }

 /* temp variable */
 data = pic->pixel;

 /* extract RGB components from mmbuf */
 p = double_buf;
```

```
 for (row=0; row<grab_rows; row++) {
  for (col=0; col<grab_cols; col++) {
    data[i+2] = *p++;           /* blue */
    data[i+1] = *p++;           /* green */
    data[i] = *p++;             /* red */
    p++;                        /* NULL byte - alpha paramter */
    i+=3;
  }
 }

 return (0);                    /* success */
}
```

Come si può notare i dati vengono presi carattere per carattere dall'immagine fornita da video for linux e vengono inseriti nell'array contenente l'immagine RGB senza operare alcuna conversione. È stata quindi modificata tale funzione affinché venissero inseriti nell'array data i valori corretti ottenuti con un'opportuna conversione.

Di seguito si riporta il nuovo codice:

```
int
vlGrabLastImage (vlImage *pic)
{


 vlPixel *data;
 unsigned char *p;

 if (!pic) {
  VL_ERROR ("vlGrabLastImage: error: NULL image\n");
  return (-1);          /* failure */
 }

 /* initialize the new picture */
 if (0 > vlImageInit (pic, RGB, grab_cols, grab_rows)) {
  VL_ERROR ("vlGrabLastImage: error: could not initialize image\n");
  return (-1);          /* failure */
 }

 /* temp variable */
 data = pic->pixel;

 /* extract RGB components from mmbuf */
 p = double_buf;
   int    j,i=0;
    int    y, u, v;
    int    r, g, b;

    int    sub_i_uv;
    int    sub_j_uv;

    int    uv_width, uv_height;
    unsigned char *src_y, *src_u, *src_v;

    int width = grab_cols;
    int height = grab_rows;
    uv_width  = width / 2;
    uv_height = height / 2;
```

```
    for (i = 0; i < height; ++i) {
      /*
       * calculate u & v rows
       */
      sub_i_uv = ((i * uv_height) / height);

      for (j = 0; j < width; ++j) {
                /*
                * calculate u & v columns
                */
                sub_j_uv = ((j * uv_width) / width);

    /***********************************************
     *
     *  Colour conversion from
     *    http://www.inforamp.net/~poynton/notes/colour_and_gamma/ColorFAQ.html#RTFToC30
     *
     *  Thanks to Billy Biggs <vektor@dumbterm.net>
     *  for the pointer and the following conversion.
     *
     *   R' = [ 1.1644         0    1.5960 ]   ([ Y' ]   [  16 ])
     *   G' = [ 1.1644   -0.3918   -0.8130 ] * ([ Cb ] - [ 128 ])
     *   B' = [ 1.1644    2.0172         0 ]   ([ Cr ]   [ 128 ])
     *
     *  Where in xine the above values are represented as
     *
     *   Y' == image->y
     *   Cb == image->u
     *   Cr == image->v
     *
     ***********************************************/
                src_y = p;
                src_u = p + width * height;
                src_v = p + width * height * 5 / 4;

                y = src_y[(i * width) + j] -16;
                u = src_u[(sub_i_uv * uv_width) + sub_j_uv] -128;
                v = src_v[(sub_i_uv * uv_width) + sub_j_uv] -128;

                r = (1.1644 * y) + (1.5960 * v);
                g = (1.1644 * y) - (0.3918 * u) - (0.8130 * v);
                b = (1.1644 * y) + (2.0172 * u);

                clip_8_bit (r);
                clip_8_bit (g);
                clip_8_bit (b);

                data[(i * width + j) * 3 + 0] = r;
                data[(i * width + j) * 3 + 1] = g;
                data[(i * width + j) * 3 + 2] = b;

      }
    }

  return (0);                           /* success */
}
```

Vista la ingente quantità di codice aggiunto, si è ritenuto necessario riportare per completezza l'intera funzione. La funzione *clip_8_bit* è stata aggiunta nel seguente modo:

```
#define clip_8_bit(val)
{
  if (val < 0)
    val = 0;
  else
    if (val > 255) val = 255;
}
```

4.1. Componenti necessari

- Una macchina linux con kernel compilato per supportare v4l
- Una webcam Philips ToUcam Pro PCVC840K/00
- I driver *pwc* per la webcam disponibili sul sito dello sviluppatore [5] (tali driver possono essere utilizzati anche per altri modelli di webcam)
- Il pacchetto delle librerie Vislib

4.2. Modalità di installazione

Il pacchetto Vislib è liberamente scaricabile presso il sito di Activmedia. Una volta ottenuto il pacchetto bisogna decomprimerlo in una directory a scelta dell'utente, si consiglia */usr/local/* procedendo con i seguenti comandi:

 # cd /usr/local

 # tar –xzvf /percorso/di/vislib-1.8-v4l.tgz

In seguito bisogna modificare il codice contenuto nel file */usr/local/vislib-1.8/src/grab.c* secondo quanto scritto sopra.

Infine bisogna compilare il tutto con i seguenti comandi:

 # cd /usr/local/vislib-1.8/

 # make

Per verificare il corretto funzionamento del pacchetto, lanciare uno dei programmi dimostrativi presenti nella directory */usr/local/vislib-1.8/examples/* come ad esempio *extractionRGB*.

4.3. Modalità di taratura

Come impostazioni predefinite è stata impostata la dimensione di acquisizione delle immagini a 640x480, agendo sul seguente codice del file */usr/local/vislib-1.8/include/vislib.h*:

```
/* image properties */
#define VL_ROWS_DEFAULT                 480
#define VL_COLS_DEFAULT                 640
```

5. Conclusioni e sviluppi futuri

Al termine delle procedure sopra elencate, si è potuto constatare il corretto funzionamento delle librerie vislib con la telecamera USB in dotazione. Potrebbe essere auspicabile l'inserimento delle modifiche nel

codice sorgente rilasciato da Activmedia in modo da essere compatibile con tutte le telecamente che acquisiscono nativamente le immagini nel formato YV12.

Bibliografia

La bibliografia deve essere indicata seguendo uno standard ben preciso. Lo stile da usare è lo stile "Biblio", che numera automaticamente i riferimenti.

[1] Sito di Activmedia - www.activmedia.com

[2] Sito di V4l - http://www.exploits.org/v4l/

[3] Sito della documentazione del formato YUV - http://fourcc.org/

[4] Codice sorgente di Xine con la funzione di conversione da YUV 4:2:0 a RGB 32 - http://koders.com/c/fid2E0141F56262807E06AC5F8934A416A9E179D91D.aspx?s=yv12torgb

[5] Sito dello sviluppatore pwc - http://www.saillard.org/linux/pwc/

Indice