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

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Adattamento della libreria VisLib all'uso con telecamere V4L2

Elaborato di esame di:

Giovanni Chiodi, Flavio Maccarrone, Stefano Peli

Consegnato il: **03 giugno 2009**

Sommario

Il lavoro svolto riguarda l'adattamento della libreria VisLib all'uso con telecamere V4L2, utilizzando i programmi "uvccapture" e "luvcview". In particolare si tratta di far comunicare la libreria Vislib con "Video for Linux Two" (V4l2), il sistema video di I/O contenuto nel Kernel che permette l'interfacciamento con webcam specifiche, tramite opportuni driver, e webcam generiche che rispettano lo standard UVC.

1. Introduzione

Il tema fondamentale del lavoro svolto riguarda l'ampliamento delle funzionalità della libreria VisLib (Visual Library) al fine di renderla compatibile con il sistema "Video for Linux Two".

1.1. Visual Library

Si tratta di una libreria C veloce e flessibile, utilizzata per l'acquisizione e l'elaborazione delle immagini con visione single-camera. È stata scritta da Terry Fong e Sebastien Grange, due componenti del "Virtual Reality and Active Interfaces (VRAI)Group", dell'istituto di tecnologia della federazione svizzera.

Questa libreria contiene funzioni per:

- l'acquisizione, la visualizzazione e la scelta del formato delle immagini;
- l'elaborazione 2D delle immagini;
- l'object-tracking,

La versione utilizzata nel progetto è la 1.9.4, strutturata per Linux e in grado di lavorare con qualsiasi *framegrabber* (dispositivo elettronico che "cattura" una singola immagine digitale partendo da un segnale video analogico o da uno stream video digitale) che supporti i driver Linux Bt8xx (dalla versione Bt848-1.0 in poi) e il gestore grafico X Window (o X11).

L'obiettivo da raggiungere con questo lavoro è poter gestire il sistema V4l2 direttamente dalla libreria VisLib, per mezzo di opportune integrazioni nella stessa.

1.2. Video for Linux

"Video for Linux" (o "video4Linux" o "V4L") è una API per Linux, strettamente integrata nel kernel, che consente di acquisire dati di tipo video. Essa supporta diversi dispositivi, tra i quali webcam e schede TV.

Storicamente esistono due versioni di questa API:

- V4L, la versione originale, introdotta a partire dal kernel 2.1.x;
- V4L2, la seconda generazione di questo sistema che corregge una serie di errori della precedente, e introdotta dal kernel 2.5.x.

> ➢ **Nonostante la seconda versione sia più ricca di funzionalità della prima, rispetto a quest'ultima perde la compatibilità con un certo numero di driver. Da qui la convivenza ancora attuale delle due versioni.**

1.3. Webcam utilizzata

Il dispositivo utilizzato è una webcam Philips, modello SPC 1005NC, di seguito sono elencate le specifiche fornite dalla casa produttrice.

Specifiche tecniche	
Descrizione Prodotto	Philips SPC 1005NC - webcam
Tipo di dispositivo	Videocamera web - fisso
Dimensioni (L x P x H)	4 cm x 8.8 cm x 8.2 cm
Peso	110 g
Localizzazione	Nuova Zelanda, Taiwan, Svizzera
Macchina fotografica	Colore
Tipo sensore ottico	CMOS - 1300000 pixel
Illuminazione minima	5 lux
Iride obiettivo	F/2.8
Interfacce	USB
Zoom digitale	5
Supporto audio	Sì : microfono incorporato
Caratteristiche	Compatibile con USB 2.0, bilanciamento del bianco, compatibile Skype, tecnologia tracking automatico immagine, pulsante snapshot
Cavi inclusi	1 x cavo USB - 1.5 m
Requisiti di sistema	Microsoft Windows XP, Microsoft Windows Vista
Accessori in dotazione	Auricolare
Certificato per Vista	Certificato per Vista

La webcam appena descritta ha inoltre il pregio di essere compatibile con lo standard UVC (Usb Video Device Class), requisito indispensabile nell'ambito del progetto corrente.

1.4. Piattaforma utilizzata

Per il progetto in questione si è scelto il sistema operativo Linux. In particolar modo viene utilizzata la distribuzione Debian GNU/Linux versione 5.0 (nome in codice *lenny*). Questo caratterizzerà ovviamente i comandi che verranno descritti nei paragrafi relativi alle modalità di installazione dei componenti software di questo progetto.

2. Il problema affrontato

Il problema trattato è di natura strettamente informatica, in quanto coinvolge aspetti di interfacciamento tra moduli software. Nello specifico la situazione che si presenta è schematizzabile tramite il seguente diagramma:

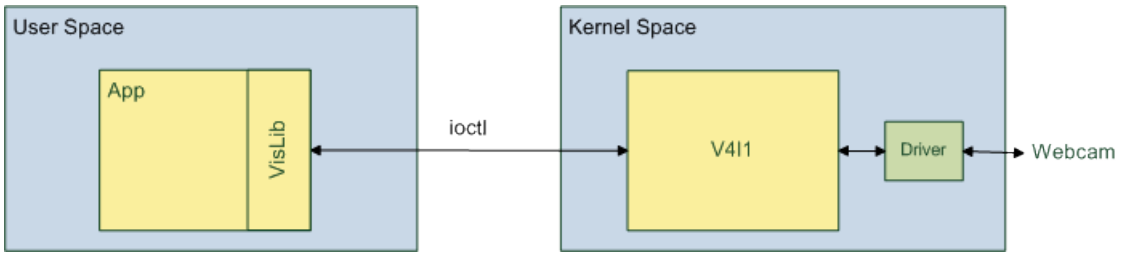

Figura 1 Funzionamento generico di VisLib

Come si può notare dalla Figura 1 la libreria VisLib è in grado di sfruttare le primitive offerte dal sistema "Video for Linux", incluso nel kernel. Pertanto il funzionamento di dispositivi hardware come le webcam è garantito.

Con il passaggio alla versione 2.6 del kernel di Linux, è stato introdotto anche il nuovo sistema video di I/O "V4l2". Ciò ha determinato ovvi problemi di interfacciamento, dal momento che VisLib non supporta alcuno standard di comunicazione con questo nuovo sistema.

La situazione che ipoteticamente si vorrebbe raggiungere è descritta nello schema sottostante.

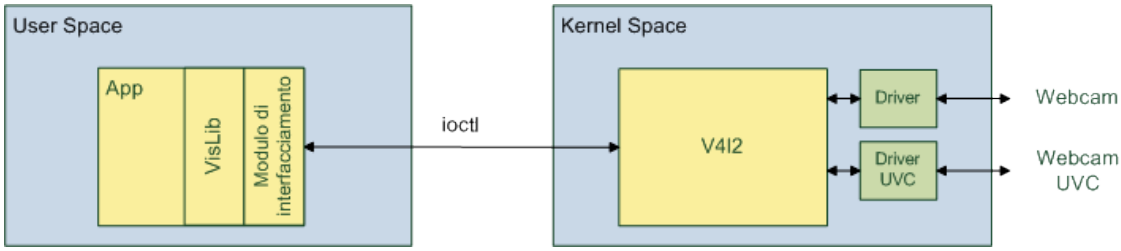

Figura 2 Funzionamento di VisLib previsto

Come si evince dalla Figura 2 tale situazione prevedrebbe la presenza di un opportuno modulo di interfacciamento, che consenta di risolvere il problema di comunicazione con il lato kernel.

3. La soluzione adottata

Il primo passo dell'ipotetica soluzione è consistito nell'analisi dei moduli relativi ai software applicativi "UVCcapture" e "luvcview", ovvero i software che permettono l'interfacciamento di dispositivi webcam per la "cattura di immagini" nonché per la gestione di file video.

Da questa analisi è emerso che entrambe le applicazioni svolgono a grandi linee le medesime operazioni (lo giustifica il fatto che entrambe si appoggiano sullo stesso file C "v4l2uvc.c"). La spinta verso la scelta di "luvcview" rispetto all'altra applicazione è data dal fatto che la prima risulta essere più recente e quindi più funzionale e performante della seconda.

Il passo successivo è quello di determinare le porzioni di software che gestiscono i meccanismi di acquisizione delle immagini tramite webcam, al fine di isolarle dal resto del software, ridondante rispetto ai nostri scopi.

Infine si passa all'ampliamento della libreria VisLib tramite l'integrazione delle funzionalità identificate al passo precedente.

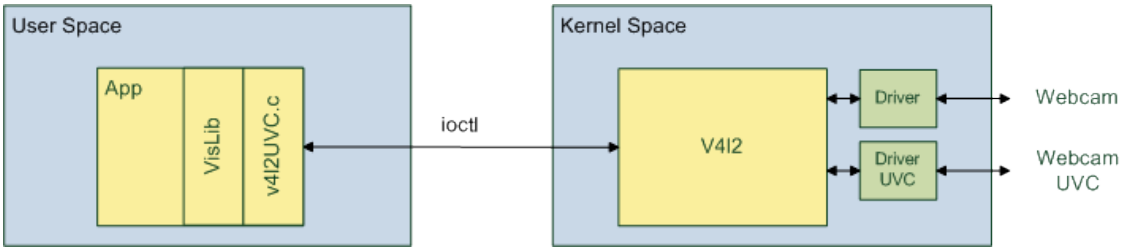

Figura 3 Funzionamento di VisLib previsto

In Figura 3 è mostrato quello che potrebbe essere lo stato finale della operazioni contemplate dalla soluzione.

4. Modalità operative

Il passo preliminare del processo di ampliamento delle funzionalità di VisLib è la messa in opera di quest'ultimo; pertanto, dopo aver scaricato il pacchetto "vislib-V1.9.4.tgz" dal sito del laboratorio, si effettua la sua compilazione al fine di verificare eventuali dipendenze e identificare i pacchetti mancanti.

Una volta rilevate le dipendenze e installati i pacchetti mancanti si giunge ad una compilazione senza errori, indispensabile per procedere con le operazioni successive.

Idealmente una possibile strategia per avvicinarsi al risultato è quella di isolare le porzioni di software necessarie alla "cattura" del fotogramma in un nuovo file, distinto dagli originali.

Quindi la parte conclusiva del lavoro, nonché la più delicata, concerne nell'adattamento delle parti di software identificate per renderle integrabili nella libreria VisLib.

4.1. Componenti necessari

Come anticipato nel paragrafo precedente, oltre agli applicativi necessari, gli errori di compilazione hanno determinato una serie di installazioni dei pacchetti mancanti, illustrati di seguito in ordine cronologico.

4.1.1. VisLib

Libreria fondamentale del progetto. Già descritta nella parte introduttiva.

4.1.2. libX11-dev

Questo pacchetto, conosciuto anche come "Xlib", fornisce un'interfaccia lato client per il sistema X Window, predisponendo tutte le API necessarie al funzionamento di quest'ultimo. Inoltre tale pacchetto contiene gli header di sviluppo per la libreria contenuta in "libx11-6".

La versione utilizzata all'interno del progetto è la 2:1.2.1-1.

4.1.3. libxt-dev

Questo pacchetto fornisce una libreria base di widget, sfruttata da molti altri strumenti software. Tale pacchetto contiene gli header di sviluppo per la libreria contenuta in "libxt6".

La versione utilizzata all'interno del progetto è la 1:1.0.5-3.

4.1.4. libxext-dev

Questo pacchetto fornisce un'interfaccia lato client per numerose estensioni del protocollo del sistema X Window, oltre a supportare diverse API per il funzionamento dello stesso. Tale pacchetto contiene anche gli header di sviluppo per la libreria contenuta in "libxext6".

La versione utilizzata all'interno del progetto è la 2:1.0.4-1.

4.1.5. x11proto-xext-dev

Questo pacchetto fornisce gli header di sviluppo che descrivono il protocollo della connessione dati punto punto ("Wire Protocol") utile alle librerie lato client contenute nella libreria principale "Xext".

La versione utilizzata all'interno del progetto è la 7.0.2-6.

4.1.6. linux-headers-2.6.26-2-686

Questo pacchetto fornisce i file descrittivi della specifica architetturale del kernel 2.6.26-2-686 di Linux, generalmente utilizzati per la compilazione "out-of-tree" dei moduli del kernel stesso.

La versione da utilizzare dipende da quella del kernel in uso, e perciò anche dalla distribuzione Linux sulla quale si lavora. Per il progetto è stata utilizzata la versione 2.6.26-2-686.

4.1.7. libv4l-0

Questo pacchetto è in realtà una collezione di tre librerie: “libv4lconvert”, “libv4l1” e “libv4l2”.

Esso aggiunge un piccolo layer astratto per l’utilizzo dei dispositivi video4linux2 da parte di programmi che supportano solo video4linux1. L’utilità è quella di supportare una grande varietà di dispositivi, per permettere alle applicazioni che sfruttano questi ultimi di non dover ricorrere alla scrittura di codice separato.

L’utilizzo di questo pacchetto nel progetto è però solo temporaneo; viene infatti sfruttato per il caricamento del file “v4l1compat.so”, necessario a verificare il momentaneo funzionamento di VisLib.

La versione utilizzata nel progetto è la 0.5.9-1.

4.1.8. subversion

Subversion, conosciuto comunemente come *svn*, è un sistema di controllo delle versioni molto simile a CVS (*Concurrent Version System*). In realtà esso contiene proprio le funzionalità maggiormente sfruttate dagli utilizzatori di CVS.

Questo pacchetto include il client Subversion, *svn* appunto, che permette di creare e rendere disponibile in rete una repository Subversion, funzionalità di ampio interesse nel progetto per poter scaricare l’applicativo *luvcview*.

La versione utilizzata è la 1.5.1dfsg1-2.

4.1.9. libsdl1.2-dev

Questo pacchetto comprende i file di sviluppo di SDL. *Simple DircetMedia* Layer è una libreria multimediale multi piattaforma, scritta in C, che crea un livello astratto al di sopra di varie piattaforme software grafiche e sonore e dunque può controllare video, audio digitale, CD-ROM, suoni, thread, caricamento condiviso di oggetti, timer e networking. Questa astrazione permette ai programmatori di scrivere un’applicazione multimediale una sola volta e farla girare su molti sistemi operativi. È indispensabile per compilare tutte le applicazioni che utilizzano SDL.

La versione utilizzata nel progetto è la 1.2.13-4.

4.1.10. uvccapture

Questo software ha l’obiettivo di catturare un’immagine da una webcam USB, ad uno specifico intervallo di tempo, e di salvarla in un file di estensione JPEG.

La versione utilizzata nel progetto è la 0.5.

4.1.11. luvcview

Questo è un software per webcam UVC che permette di registrare video, con la possibilità di salvarli anche in formato AVI. È ideale per effettuare piccoli test sul funzionamento di una webcam: per questo, e per altri motivi spiegati nei paragrafi successivi, viene preferito ad *uvccapture* nello svolgimento del progetto in questione.

La versione utilizzata nel progetto è la 1:0.2.4-2.

4.2. Modalità di installazione

Prima di procedere all’integrazione di parti di codice in VisLib viene eseguita tutta una serie di operazioni: le prime atte a verificare la corretta funzionalità della libreria C considerata, le seconde dedite all’installazione e all’esatta configurazione dei pacchetti e dei software applicativi precedentemente elencati e descritti. Di seguito sono riportati i passi compiuti per effettuare quanto appena detto.

4.2.1. Installazione di VisLib

Per prima cosa viene scaricato (direttamente dal sito del laboratorio) il file compresso "vislib-V1.9.4.tgz", contenente i file necessari per installare la libreria VisLib. Tale archivio viene quindi decompresso per mezzo del solito comando di decompressione:

```
# tar xzf vislib-V1.9.4.tgz
```

L'esecuzione di questa operazione porta alla creazione automatica della cartella "vislib", alla quale si può accedere non solo per compilare la relativa libreria, ma anche per ottenere alcune informazioni, come ad esempio quelle contenute nella documentazione ("vislib/doc/tutorial.pdf"). La prima attività, quella relativa alla compilazione, è però indispensabile per procedere all'installazione di VisLib. I comandi sono quelli previsti per una generica compilazione:

```
# cd vislib/
# make
```

È molto probabile che in questa fase vengano segnalati degli errori, dovuti principalmente alla mancata installazione di alcuni pacchetti software sul sistema operativo in utilizzo.

> ➢ **Gli errori rilevati in fase di compilazione possono risultare diversi da quelli presi in considerazione in questo progetto, ma è ovvio! Come altre applicazioni anche VisLib richiede pacchetti che possono essere presenti o meno, ma questo dipende dallo stato del sistema operativo in uso.**

In particolare i difetti rilevati a questo punto delle operazioni sono i seguenti:

```
error: X11/Instrinsic.h: Nessun file o directory
error: X11/StringDefs.h: Nessun file o directory
error: X11/extensions/XShm.h: Nessun file o directory
error: X11/keysym.h: Nessun file o directory
```

Per risolverli viene dapprima letto il file "install", contenuto nella cartella principale di Vislib.

```
# less install
```

Da questo, che consiglia di visitare il sito web della Active-Media, si evince che alla base di tutto c'è la mancanza del driver Bt848, responsabile dell'API Video for Linux. Vengono quindi cercati i file mancanti, evidenziati dal log della compilazione, o tramite la funzionalità di ricerca dell'interfaccia testuale Linux "aptitude", o tramite il motore web di ricerca dei pacchetti di Debian Linux. Una volta identificati, i pacchetti necessari vengono installati, e ancora una volta viene preferita l'interfaccia "aptitude". In particolare si ha che:

- per far fronte alla mancanza del file keysym.h viene installato il pacchetto libx11-dev

  ```
  # aptitude install libx11-dev
  ```

- per la mancanza dei file Intrinsic.h e StringDefs.h viene installato il pacchetto libxt-dev

  ```
  # aptitude install libxt-dev
  ```

- per la mancanza del file XShm.h vien installato il pacchetto x11proto-xext-dev che lo contiene

  ```
  # aptitude install x11proto-xext-dev
  ```

È possibile che, nonostante tutti i precedenti errori siano stati risolti, una successiva compilazione ne identifichi diversi altri. Questo perché i nuovi pacchetti generano altre dipendenze da pacchetti che come quelli appena installati non si trovano nel sistema operativo corrente. È il caso del pacchetto linux-headers 2.6.26-2-686, che viene immediatamente installato:

```
# aptitude install linux-headers-2.6.26-2-686
```

In realtà, anche dopo una corretta installazione, i file di tale pacchetto faticano ad essere trovati durante la compilazione di VisLib. Accedendo alla cartella "src" di sistema la presenza del pacchetto appena istallato viene però constatata:

```
# cd /usr/src/
# ls
```

Dopo un'attenta analisi del problema se ne identifica la causa: il file "grab.c" della sottocartella "src" di VisLib mostra un comando di inclusione errato poiché il cammino descritto per il recupero del file "pwc-ioctl.h", necessario a catturare una foto istantanea da un video, è inesistente. Tale complicazione viene risolta creando per comodità un link ai file del pacchetto appena scaricato:

```
# ln -s linux-headers-2.6.26-2 linux
```

ed editando il file "grab.c" secondo questo opportuno indirizzo:

```
# include "/usr/src/linux/include/media/pwc-ioctl.h"
```

Completato il passo precedente viene individuato un ultimo errore di dipendenza, risolto installando l'ultimo pacchetto necessario, libxext-dev:

```
# aptitude install libxext-dev
```

A questo punto la compilazione di VisLib non genera più alcun errore: la libreria risulta correttamente compilata.

4.2.2. Collaudo webcam

Il dispositivo webcam Philips SPC 1005NC, le cui specifiche sono state descritte nella fase introduttiva, viene collaudato per il suo utilizzo durante l'intero progetto. Per farlo vengono eseguite due semplici operazioni:

- connessione del cavo della webcam ad una porta USB del calcolatore in utilizzo;
- controllo del riconoscimento del dispositivo da parte del sistema operativo, attraverso lo studio dell'output del seguente comando:

  ```
  # lsusb
  ```

  che deve dare come risultato

  ```
  # Bus 00x Device 00x: ID 0471:0332 Philips
  ```

Se l'output è corretto il sistema assegna alla webcam un "device", identificato dalla parola *video* seguita da un numero (ad esempio *video0*).

> ➢ **Se la creazione del "device" non avviene automaticamente, questa deve essere eseguita manualmente, oppure tramite software adibiti alla gestione delle periferiche plug and play USB.**

4.2.3. Verifica di funzionamento di VisLib

Verificata la corretta installazione della webcam, non resta che testare il funzionamento dei programmi di esempio che accompagnano la libreria VisLib. Per farlo viene eseguito un file di esempio, contenuto appunto in "examples", sottocartella di VisLib. La scelta ricade sull'esempio identificato dal nome "acquisizionePiccola". Eseguire tale applicativo è semplice:

```
# cd examples/
#./acquisizionePiccola -d /dev/video0
```

Tuttavia viene generato il seguente errore:

```
vlInit: error: Impossibile inizializzare il framegrabber
```

Questo non è causato da un malfunzionamento di VisLib, ma dal fatto che la webcam collegata, che l'applicativo cerca di utilizzare, richiede API Video4Linux2, delle quali VisLib è momentaneamente sprovvista. Risolvere questo problema è praticamente l'obiettivo del progetto.

Resta però un punto interrogativo: il corretto funzionamento di VisLib. Per affrontare questa situazione viene presa in considerazione una soluzione momentanea: l'utilizzo del pacchetto software libv4l-0. Quest'ultimo, la cui funzione è esplicata nel paragrafo relativo ai componenti installati, contiene infatti il file "v4l1compat.so", necessario all'applicativo di esempio per poter funzionare. Si procede con l'installazione del pacchetto:

```
# aptitude install libv4l-0
```

Per maggiore sicurezza viene ricompilata la libreria VisLib. Quindi viene utilizzata la funzionalità del linker dinamico di Linux per caricare e collegare il file "v4l1.compat.so" della libreria appena installata all'applicativo "acquisizionePiccola":

```
# LD_PRELOAD=/usr/lib/libv4l/v4l1compat.so
```

Viene eseguito il file di esempio:

```
#./acquisizionePiccola
```

che in questo caso va a buon fine senza dare nessun errore, ma anzi catturando un'immagine dalla webcam.

Volendo effettuare più prove si può esportare al livello di variabile globale il link dinamico creato, tramite il comando:

```
# export LD_PRELOAD=/usr/lib/libv4l/v4l1compat.so
```

Il tutto però, come precedentemente anticipato, è solo un'operazione temporanea, atta a verificare il corretto funzionamento della compilata libreria VisLib. Una volta realizzato l'obiettivo del progetto, tale procedura non si renderà più necessaria.

4.2.4. Installazione e configurazione di uvccapture e luvcview

Le porzioni di software da integrare nella libreria VisLib vengono editate partendo da quelle offerte da altre applicazioni.

Inizialmente viene considerato l'applicativo *uvccapture*, introdotto precedentemente, scaricato e installato con i seguenti comandi:

```
# wget 'http://staticwave.ca/source/uvccapture/uvccapture-0.5.tar.bz2'
# tar xjf uvccapture-0.5.tar.bz2
# cd uvccapture-0.5/
# make
```

In fase di compilazione non sorgono particolari problemi legati alla dipendenza da pacchetti non installati; questo accade perché uvccapture utilizza librerie genericamente già presenti sui sistemi operativi Linux, o comunque precedentemente installate per risolvere gli errori di compilazione di Vislib. Nel caso sorgessero errori si consiglia di risolvere le dipendenze associate a uvccapture.

Per maggiori Informazioni viene consultato il file di lettura:

```
# less README
```

Utilizzando l'applicazione appena installata sorgono però alcune complicazioni, poiché la funzionalità di "cattura" delle foto genera immagini sfalsate con la webcam in uso.

Prima di trovare una soluzione al problema riscontrato viene considerata un'alternativa: *luvcview*. La scelta di questo applicativo software non è casuale. Leggendo attentamente la documentazione relativa alle due applicazioni si scopre infatti che tra loro esiste uno stretto legame: uvccapture si basa su luvcview, in particolare per mezzo del file "v4luvc.h", che nella seconda applicazione risulta avere una versione più recente che nella prima. Si procede perciò con l'installazione di luvcview. Viene innanzitutto creata una directory adibita a contenere l'applicativo considerato:

```
# mkdir luvcview
# cd luvcview/
```

Quindi ne viene scaricato il relativo pacchetto (è in questa fase che viene installato il software subversion, necessario per il download del file):

```
# aptitude install subversion
# svn checkout http://svn.quickcamteam.net/svn/luvcview
```

Viene poi eseguito il generico comando di compilazione:

```
# make
```

che porta alla luce diversi errori circa la mancanza della libreria SDL (si veda l'introduzione per maggiore completezza). Quest'ultima viene cercata e installata grazie alle solite funzionalità di aptitude:

```
# aptitude search sdl | grep dev
# aptitude install libsdl1.2-dev
```

La successiva ricompilazione mostra però ancora errori:

```
# make
error: uvcvideo.h: Nessun file o directory
error: dynctrl-logitech.h: Nessun file o directory
```

Una semplice lettura del file README di luvcview spiega come risolverli; si tratta di scaricare questi file e altri necessari:

```
# wget http://svn.berlios.de/svnroot/repos/linux-uvc/linux-uvc/trunk/uvcvideo.h
# wget http://svn.berlios.de/svnroot/repos/linux-uvc/linux-uvc/trunk/uvc_compat.h
# wget http://svn.quickcamteam.net/svn/qct/Linux/Common/include/dynctrl-
logitech.h
# make
```

La compilazione va a buon fine; non resta che utilizzare luvcview con la webcam prima collaudata. Viene eseguito il seguente comando:

```
# ls -l /dev/video0
```

dal quale si evince che per utilizzare correttamente il tutto è richiesta l'appartenenza al gruppo *video*. Per constatarlo viene verificato l'output del comando:

```
# groups
```

Tutto risulta a posto, quindi si procede all'utilizzo, semplicemente come prova, dell'applicazione luvcview:

```
#./luvcview -d /dev/video0
```

E con questo si constata l'effettivo funzionamento del dispositivo webcam e dell'applicazione luvcview.

4.3. Estensione di VisLib

L'estensione di VisLib è sicuramente l'operazione più delicata ed importante dell'intero progetto, poiché coinvolge svariati aspetti di programmazione a basso livello ed in generale modifiche di svariate parti di codice preesistente, spesso non del tutto chiaro.

La situazione che si presenta prima di effettuare le modifiche relative all'integrazione di VisLib è schematizzata nella figura sottostante.

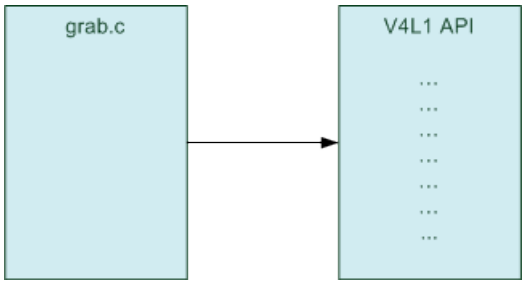

Figura 4 Struttura del grab prima delle modifiche

È possibile notare che il file "grab.c", contenente la logica necessaria alla cattura delle immagini, si appoggia sulle funzioni messe a disposizione da Video for Linux 1.

Come è possibile intuire, la situazione che si desidera ottenere è quella che vede coinvolte le funzioni fornite da Video for Linux 2, come segue.

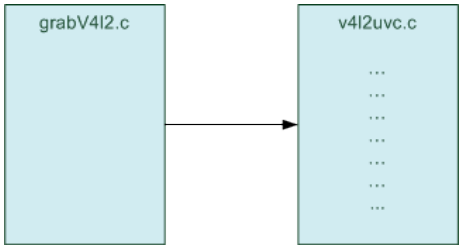

Figura 5 Struttura del grab dopo le modifiche

Il vecchio file "grab.c" è sostituito da un nuovo file "grabV4l2.c", che a differenza del primo si appoggia sulle funzioni di V4l2, nello specifico sul file "v4l2uvc.c".

Prima di iniziare con il lavoro vero e proprio si opta per l'effettuazione di una prova di cattura dell'immagine (una sorta di studio di fattibilità); per fare questo viene creato un file "shot.c" che richiama le funzioni di luvcview e viene verificata l'effettiva possibilità di catturare e salvare un'immagine tramite la webcam. La prova consistite nella semplice chiamata del file con il comando

```
# ./shot
```

dopo l'opportuna compilazione.

A questo punto del lavoro si è riscontrato un problema consistente nell'acquisizione di un'immagine fallata, ossia non perfettamente centrata: questo probabilmente era dovuto ad una non perfetta sincronizzazione iniziale tra webcam e V4L2. Il problema è stato risolto con uno shot preventivo a vuoto, che ha garantito la centratura del fotogramma acquisito.

4.3.1. Fase di integrazione dei file necessari

Il primo passo tangibile del lavoro riguarda l'inclusione degli header e dei file contenenti le funzioni all'interno dello spazio di lavoro della libreria VisLib, in modo da mettere a disposizione di quest'ultima le funzionalità necessarie alla cattura ed al salvataggio delle immagini.

Nel dettaglio, i file che vengono aggiunti nella directory VisLib/include sono i seguenti:

```
avilib.h
color.h
dynctrl_logitech.h
huffman.h
ultils.h
uvc_compat.h
uvcvideo.h
v4l2uvc.h
```

Mentre i file aggiunti nella directory VisLib/src risultano essere:

```
avilib.c
color.c
utils.c
v4l2uvc.c
```

Tutti i file vengono estratti dal pacchetto luvcview, ampiamente discusso nei paragrafi precedenti. Eventualmente, al momento del rilascio di una nuova versione del pacchetto, per effettuare anche l'aggiornamento della libreria VisLib basterà sostituire i file appena descritti.

4.3.2. Modifiche rilevanti degli header

A questo punto del lavoro vengono apportano anche delle modifiche al file "vlGrabV4l2.h", creato a partire dal preesistente file "vlGrab.h" e contenuto nella directory vislib/include:

- aggiunta l'istruzione

```
#include <linux/videodev.h>
```

che consente di accedere alle costanti di Video for Linux 2 (V4L2_PIX_FMT_YUYV, V4L2_PIX_FMT_MJPEG);

- sostituito l'include guard per mezzo dell'istruzione

```
ifndef __GRAB_V4L2_H__
```

che evita al compilatore di includere più volte lo stesso file in fase di compilazione.

4.3.3. Creazione del nuovo file "grabV4l2.c"

Il punto saliente del lavoro è rappresentato dalla creazione di un nuovo file grabber che sfrutti V4l2 anziché V4l. Per far questo viene scelto di partire dal file originale "grab.c" effettuandone una copia e rinominandola in "grabV4l2" (vedi Figura 5) e successivamente apportare le modifiche necessarie.

Se il codice che opera i controlli sulle operazioni (blocchi if) può rimanere sostanzialmente invariato, al contrario tutte le chiamate di funzione che vanno verso V4l devono essere reindirizzate verso V4l2; nel dettaglio quelle coinvolte sono le seguenti:

```
init_videoIn()
uvcGrab()
initLut()
close_V4l2()
freeLut()
Pyuv422torgb24()
V4l2SetControl()
```

4.3.4. Problemi riscontrati sul formato delle immagini

Un aspetto che merita particolare attenzione è il formato delle immagini sfruttato da V4l2. Durante le modifiche, infatti, si riscontra un fastidioso problema, che causa la cattura di immagini dai colori errati ed aventi la metà inferiore completamente nera.

Sfogliando le documentazioni a disposizione ed osservando la parte del codice responsabile della conversione delle immagini si nota che VisLib sfrutta 2 byte per ogni informazione di colore, anziché uno solo come avviene nel metodo Pyuv422torgb24(), pertanto ciascuna terna di colore RGB è rappresentata da 6byte (rr gg bb) anziché da 3 (r g b).

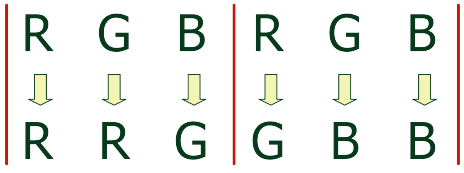

Figura 6 Problema di conversione dell'immagine

In questo modo è possibile ipotizzare che il problema sia dovuto ad una modalità di conversione della struttura delle immagini non corretta (Figura 6), ovvero che la copia avvenga byte a byte e non colore per colore, causando da un lato l'inesattezza dei colori (basta osservare come ad esempio nei 2 byte relativi al rosso della nuova immagine siano inseriti il rosso ed il verde dell'immagine precedente), dall'altro la metà immagine nera (dovuta al fatto che il numero di byte che rappresentano la nuova immagine è doppio rispetto al primo).

Correggendo il codice in modo opportuno, facendo in modo che la copia fosse operata colore per colore si risolve il problema.

4.3.5. Modifiche sul Makefile

Affinché la compilazione faccia riferimento alle nuove versioni dei file prodotti vengono operate opportune modifiche al Makefile, sostituendo le occorrenze di “grab.c” con “grabV4l2.c”.

4.3.6. Testing sugli esempi

Al termine del lavoro si è ritenuto opportuno effettuare un breve test su alcuni esempi contenuti nella directory vislib/examples:

- acquisizionePiccola: consente la cattura di immagini dalla webcam;
- frameratePiccolo: consente la cattura di una serie di frame consecutivi a frequenza regolabile;
- gainControl: fa variare il guadagno della telecamera in un intervallo che va da un valore minimo ad un valore massimo.

Tutte le prove danno esito positivo e comprovano la correttezza delle modifiche apportate al software nel progetto.

4.4. Avvertenze

Nello svolgimento di tutti i passi documentati nei paragrafi precedenti vengono evidenziate delle accortezze. Alcune sono descritte proprio durante la descrizione dei passi da effettuare per raggiungere un determinato risultato, altre sono riproposte qui di seguito.

Nei vari esempi della cartella examples di VisLib il device è stato cablato nel codice, quindi per un corretto utilizzo va modificato in base a come viene riconosciuto dal kernel (esempio /dev/video1).

Nel metodo vlGrabInit() esiste un parametro (input) che non veniva inizialmente utilizzato da VisLib, ma la cui presenza era necessaria per ragioni storiche. In questo contesto tale parametro viene invece preso in considerazione come fattore di scelta tra il formato di acquisizione *mjpeg* o il formato di acquisizione *yuyv*.

5. Conclusioni e sviluppi futuri

L'esperienza svolta ha avuto esito positivo: ora è possibile sfruttare tutte le funzionalità della libreria VisLib utilizzando dispositivi webcam compatibili con lo standard Video for Linux 2.

Tale progetto potrebbe però evolvere andando a toccare diversi punti, tra cui:

- la possibilità di scelta tra l'utilizzo dello standard Video for linux 2 o Video for Linux 1, attualmente soppiantato dal primo nel progetto considerato;
- l'aggiunta di funzionalità avanzate, supportate solo da alcuni dispositivi webcam (alcune logitech), quali *pan*, *tilt* e *zoom*;
- l'aggiornamento della relativa documentazione.

Bibliografia

[1] Sito web del Laboratorio di Robotica Avanzata dell'Università di Brescia, contenente materiale didattico vario, tra cui la libreria *VisLib*, oggetto di studio del progetto svolto - http://www.ing.unibs.it/~arl/

[2] Sito web ufficiale Active Media, contenente informazioni relative alla libreria VisLib – http://robots.activemedia.com

[3] Sito web ufficiale Logitech contenente applicazioni, informazioni e documenti relativi a webcam UVC e non UVC, per diversi tipi di sistema operativo - http://www.quickcamteam.net/

[4] Pagina di riferimento del sito web ufficiale Logitech contenente il software *uvccapture* utilizzato nel progetto - http://www.quickcamteam.net/software/linux/v4l2-software/uvccapture/

[5] Pagina di riferimento del sito web ufficiale Logitech contenente il software *luvcview* utilizzato nel progetto - http://www.quickcamteam.net/software/linux/v4l2-software/luvcview/

[6] Repository di pacchetti Linux Debian, dalla quale è possibile scaricare pacchetti e relative informazioni - http://packages.debian.org/hu/sid/libxext-dev

Indice