UNIVERSITÀ DI PADOVA **FACOLTÀ DI INGEGNERIA**

TESI DI LAUREA

SISTEMI DI TELECAMERE INTELLIGENTI PER LA VIDEOSORVEGLIANZA

Laureando: Alberto Salamone

Relatore: Prof. Emanuele Menegatti
Correlatore: Ing. Stefano Ghidoni

Corso di Laurea Magistrale in Ingegneria Informatica

Data Laurea 26/10/2010

Anno Accademico 2009/2010

Ringraziamenti

Alla mia famiglia per il costante appoggio durante questo lungo cammino
ad Irene per essermi sempre stata vicina
a Dino per avermi sempre dato una mano nei momenti di bisogno
e a Bert per i "robe da Sala".

Sommario

Questa tesi è stata svolta nell'ambito di un progetto del Laboratorio di Sistemi Autonomi Intelligenti (IAS-LAB) dellUniversità degli Studi di Padova. Il progetto consiste nella creazione di un sistema autonomo di videosorveglianza distribuito. Il presente lavoro di tesi si è focalizzato principalmente su due aspetti:

- la realizzazione di un fall detector per la rilevazione di cadute di persone tramite l'elaborazione delle immagini riprese da una telecamera ambientale;
- l'incapsulamento del fall detector sviluppato all'interno di un nodo di elaborazione di un sistema software basato su Network Multimedia Middleware (NMM). NMM è un middleware per la gestione di stream video e audio in una rete distribuita.

Inizialmente, viene illustrato lo stato dell'arte attuale per quanto riguarda i sistemi di rilevazione delle cadute, soprattutto focalizzato in ambito domestico. Vengono inoltre illustrate le principali tecniche sviluppate finora differenziandole fra tecniche di analisi e tecniche ad apprendimento.

Dopo la presentazione della soluzione innovativa realizzata per il fall detector, nei successivi capitoli viene presentato NMM (Network-Integrated Multimedia Middleware), il middleware adottato all'interno dello IAS-LAB per la realizzazione del sistema distribuito. NMM viene descritto ponendo molta attenzione sulle difficoltà incontrate durante l'installazione e la preparazione dell'ambiente di lavoro cercando di fornire una sorta di piccola "guida all'uso" sperando di facilitarne il futuro utilizzo.

Indice

Sommario **iii**

Introduzione **xi**

I Fall Location . . . xiii
II Conseguenze delle cadute . . . xiii
III Stato dell'arte dei fall detector . . . xv
IV Principi e Algoritmi per la rilevazione di cadute . . . xvi
IV.I Analytical methods . . . xvi
IV.II Machine learning methods . . . xvii

1 Ambiente di lavoro **19**
1.1 OpenCV . . . 21

2 Fall Detector **23**
2.1 Motion Detection . . . 23
2.2 Feature Extraction . . . 26
2.2.1 Gradiente . . . 26
2.2.2 Aspect Ratio . . . 27
2.2.3 Distanza fra proporzione fra BBox (Beta) . . . 27
2.2.4 Tempi di immobilità . . . 27
2.2.5 Ampiezza e direzione del movimento . . . 28
2.3 Algoritmo implementato . . . 29
2.4 Valutazione per i Fall Detector . . . 31
2.4.1 Criteri di qualità . . . 31
2.4.2 Set up degli esperimenti . . . 32
2.4.3 Esperimenti e risultati . . . 33

3 NMM (Network-Integrated Multimedia Middleware) **39**
3.1 Architettura generale di NMM . . . 40
3.2 L'organizzazione a node . . . 42
3.3 Registry Service . . . 45

4 Installazione di NMM **47**
4.1 NMM . . . 47
4.1.1 Prerequisiti hardware . . . 47
4.1.2 Configurazione della rete . . . 48
4.1.3 Configurazione software . . . 48
4.1.4 Testing del sistema . . . 53
4.1.5 Configurazione del controllo per la sicurezza . . . 54
4.1.6 Come utilizzare NMM . . . 55
4.2 SDK (Software Development Kit) . . . 58

5 NMM e Fall Detection **63**
5.1 DedistNode . . . 63
5.2 FallDetectionNode . . . 67
5.3 VideoEventReceiverNode . . . 69
5.4 Problemi con NMM . . . 72

6 Conclusioni **75**

Appendix **81**

A Librerie per l'installazione di NMM **81**
A.1 Informations on external libraries . . . 83
A.1.1 a52dec . . . 83
A.1.2 faad . . . 84
A.1.3 ffmpeg . . . 84
A.1.4 I1394 . . . 86
A.1.5 libmp3lame . . . 86
A.1.6 libraw1394 . . . 87
A.1.7 libmad . . . 87
A.1.8 libdvdnav . . . 87
A.1.9 libdvdread . . . 88
A.1.10 libogg . . . 88
A.1.11 libvorbis . . . 89
A.1.12 libshout . . . 89
A.1.13 fftw . . . 90
A.1.14 libliveMedia . . . 90
A.1.15 mpeg2dec . . . 91
A.1.16 cdparanoia . . . 92
A.1.17 libpng . . . 92
A.1.18 asoundlib . . . 92
A.1.19 Xlib . . . 93
A.1.20 libjpeg . . . 93
A.1.21 ImageMagick . . . 93

A.1.22 ImageMagick for Windows 94
A.1.23 mplayer 94
A.1.24 vlc 94
A.1.25 transcode 95
A.1.26 ogmtools 95
A.1.27 libxml++ 95
A.1.28 libx264 96
A.1.29 DVB API 5.1 96
A.1.30 ulxmlrpcpp 96
A.1.31 openssl 97
A.1.32 expat 97

Bibliografia **99**

Elenco delle figure

1 Percentuale di cadute in un periodo di 12 mesi, monitorando soggetti di età compresa fra i 65 e gli over 80. xii
2 Location delle cadute. xiii
3 Andamento dei ricoveri ospedalieri dovuti a cadute in un ospedale australiano. . . xiv

1.1 Varie tipologie di specchio per videocamera omnidirezionale. 19
1.2 A sinistra un esempio di dome mentre a destra una videocamera omnidireazionale. 20
1.3 Presentazione dell'Omnidome. 20

2.1 Andamento del gradiente. 26
2.2 Andamento dell'aspect ratio. 27
2.3 Diagramma di flusso dell'algorimo implementato. 30
2.4 Scenari di caduta. 33
2.5 Slot sequenziali per i video di test. 34
2.6 Esempio di frame acquisito. 34
2.7 Esempio di frame con bounding box del tracking. 35
2.8 Esempio di frame con caduta prima della rilevazione. 35
2.9 Esempio di frame con caduta con rilevazione. 35
2.10 Esempio di frame con caduta. 36
2.11 Esempio di frame con caduta. 36
2.12 Esempio di frame con caduta. 36
2.13 Esempio di tracking non ideale. Il soggetto è segmentato in 3 bounding box. . . 38
2.14 Ingrandimento della figura precedente. 38

3.1 Esempio di architettura Client - Server. 40
3.2 Esempio di architettura basata su middleware. 40
3.3 Architettura di NMM. 41
3.4 Esempio di un flow graph con NMM. 43
3.5 Esempio di un flow graph distribuito con NMM. 44
3.6 Divisione e sincronizzazione del flusso audio-video tramite NMM. 44

5.1 Immagine omnidirezionale dello stream in input per il nodo. 64
5.2 Immagine dedistorta dello stream in output dal nodo. 64
5.3 Grafo NMM per l'applicazione sviluppata. 71

6.1 Rilevazione della caduta da parte dell'Omnidome. 76
6.2 Visuale del dome. 76

Introduzione

La caduta degli anziani rappresenta un problema di salute pubblica in quanto possibile causa di fratture invalidanti [1] e può comportare anche conseguenze psicologiche che riducono l'indipendenza della persona [2].

Nel 1987 il "Kellogg International Working Group on the prevention of falls in the elderly" ha definito la caduta come un "non intenzionale giungere sul pavimento, o un qualsiasi livello inferiore, come conseguenza di un colpo violento, di una perdita di coscienza , di un'insorgenza improvvisa di paralisi, come un ictus o un attacco epilettico". Da allora sono state utilizzate questa o molte altre simili descrizioni per definire la caduta. A seconda dell'oggetto di studio, tuttavia, alcuni ricercatori hanno utilizzato una definizione più ampia che comprende le cadute che si verificano a causa di vertigini e sincope.

La definizione del Kellogg è appropriata quindi, per gli studi volti a identificare i fattori che compromettono le funzioni "sensomotorie" e del controllo dell'equilibrio, mentre la definizione più generica è opportuna per gli studi che si concentrano anche su cause cardiovascolari.

Anche se le cadute sono spesso identificate come incidenti, è stato dimostrato statisticamente che l'evento "caduta" può essere significativamente descritto da una distribuzione di Poisson, ciò implica che i processi causali sono coinvolti e che i "fall incident" non sono semplicemente eventi casuali, ma eventi legati a fattori come età, sesso, peso, struttura fisica, ecc...[3].

I primi studi per la rilevazione di cadute si sono basati su interviste nelle quali veniva chiesto al soggetto quante volte era caduto in un periodo passato, solitamente 12 mesi. Questa metodologia ovviamente non si è rivelata vincente in quanto molte cadute non venivano ricordate visto il lungo periodo di tempo coinvolto. La riduzione del periodo considerato però forniva una finestra temporale poco significativa.

Studi più recenti invece, hanno sottoposto i soggetti ad un periodo di monitoraggio, indicativamente sempre di 12 mesi, per poter identificare più accuratamente le cadute. Com'è lecito aspettarsi questi studi hanno rilevato una frequenza di caduta ben maggiore rispetto le prime analisi. Per tenere traccia del numero di cadute durante il periodo di monitoraggio sono state utilizzate varie tecniche: "calendari di caduta", questionari mensili, accertamenti telefonici, o email.

Ogni metodo ha vantaggi e svantaggi in termini di precisione, di costo e tempo di impegno. I calendari sono molto precisi in quanto i soggetti sono pregati di indicare ogni giorno se si sono o meno verificate cadute. Tuttavia, i dettagli specifici sulle circostanze non possono essere accertati fino a quando il diario non viene restituito alla fine del mese.

I questionari mensili hanno un vantaggio nel fatto che tutte le informazioni si possono trarre da un modulo unico.

Le interviste telefoniche consentono il livello di approfondimento voluto dal ricercatore, ma possono richiedere molte chiamate e disponibilità elevata da parte del soggetto. Tuttavia, anche con la metodologia più rigorosa, è molto probabile che si verifichino circostanze nelle quali risulta difficile rilevare tutti i dati necessari. Dopo una caduta, i più anziani sono spesso disorientati e doloranti e non ricordano i fattori scatenanti. Inoltre si verificano anche episodi di "negazione", ovvero viene addossata la colpa della loro caduta a fattori esterni e quindi non valutano l'incidente verificatosi come un'effettiva caduta non riportandola sulla documentazione.

Nel 1977, Exton-Smith ha esaminato l'incidenza delle cadute in 963 persone di età superiore ai 65 anni [4]. Egli ha rilevato che nelle donne dai 65 ai 69 anni, la percentuale di coloro che sono cadute aumenta con l'età di circa il 30%, mentre supera il 50% in quelle oltre gli 85 anni. Negli uomini dai 65 ai 69 anni, la percentuale di coloro che sono caduti aumenta con l'età del 13% arrivando circa al 30% per i soggetti sopra gli 80 anni e oltre.

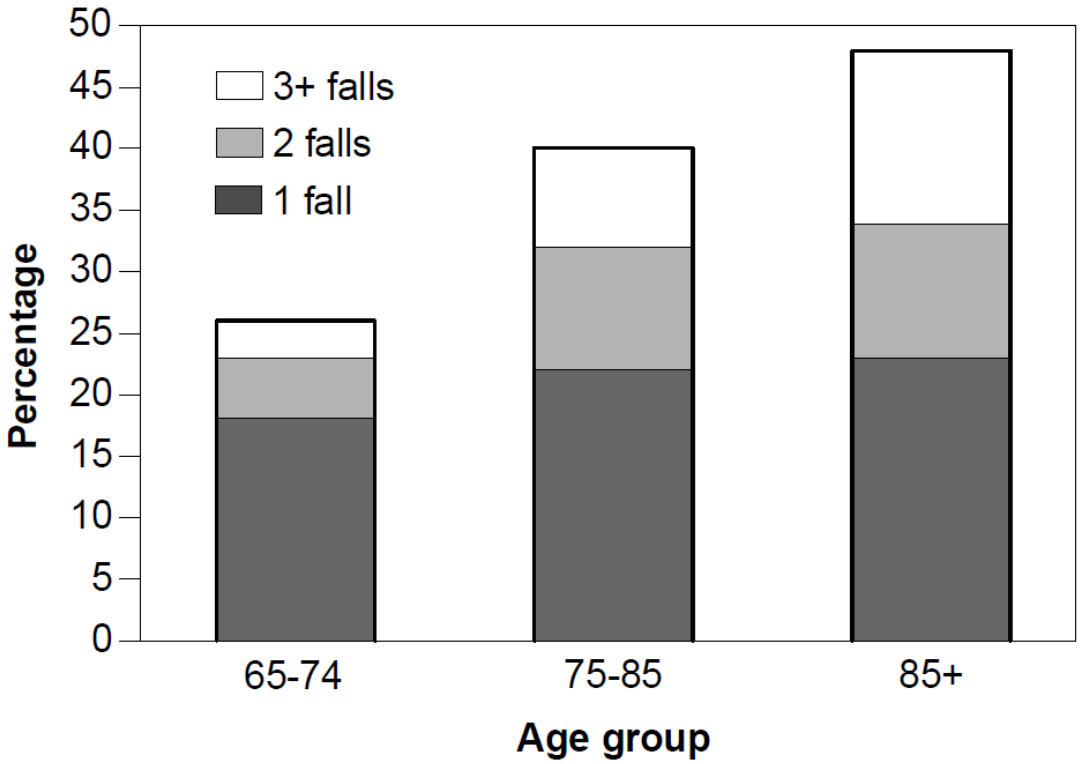

Figura 1: *Percentuale di cadute in un periodo di 12 mesi, monitorando soggetti di età compresa fra i 65 e gli over 80.*

I Fall Location

Nel caso di anziani che vivono nella propria abitazione in maniera indipendente, circa il 50% delle cadute si verificano nell'ambito delle loro case e negli immediati dintorni. Solitamente si verificano negli ambienti comunemente utilizzati come la camera da letto, salotto, cucina e bagno. Il resto delle cadute invece si verifica in ambienti pubblici o in case altrui.

Comunemente le cause di caduta in ambienti pubblici includono crepe nel pavimento, disallineamenti, terreno o superfici scivolose [5].

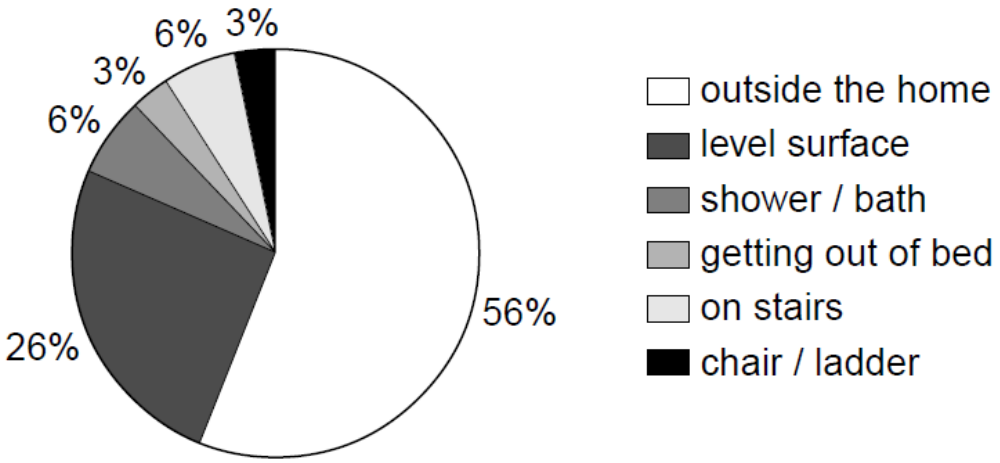

Figura 2: *Location delle cadute.*

L'ubicazione delle cadute è anche legata all'età e al sesso. Nel caso di donne anziane che vivono autonomamente per esempio, si è scoperto che il numero di cadute verificatasi al di fuori della casa diminuisce con l'età, con un corrispondente aumento del numero di cadute verificatesi all'interno della casa.

II Conseguenze delle cadute

Le cadute sono la principale causa di ricovero ospedaliero in persone dai 65 anni in su e rappresentano il 4% di tutti i ricoveri in questa fascia d'età.

Dai 40 anni, il tasso di ricoveri dovuti a cadute aumenta costantemente del 4,5% all'anno per gli uomini (raddoppio ogni 15,7 anni) e del 7,9% all'anno per le donne (raddoppio ogni 9,1 anni). In coloro che hanno compiuto 85 anni ed oltre, i livelli hanno raggiunto il 4% annuo negli uomini e 7% l'anno nelle donne [6].

L'andamento appena descritto è visibile in figura 3.

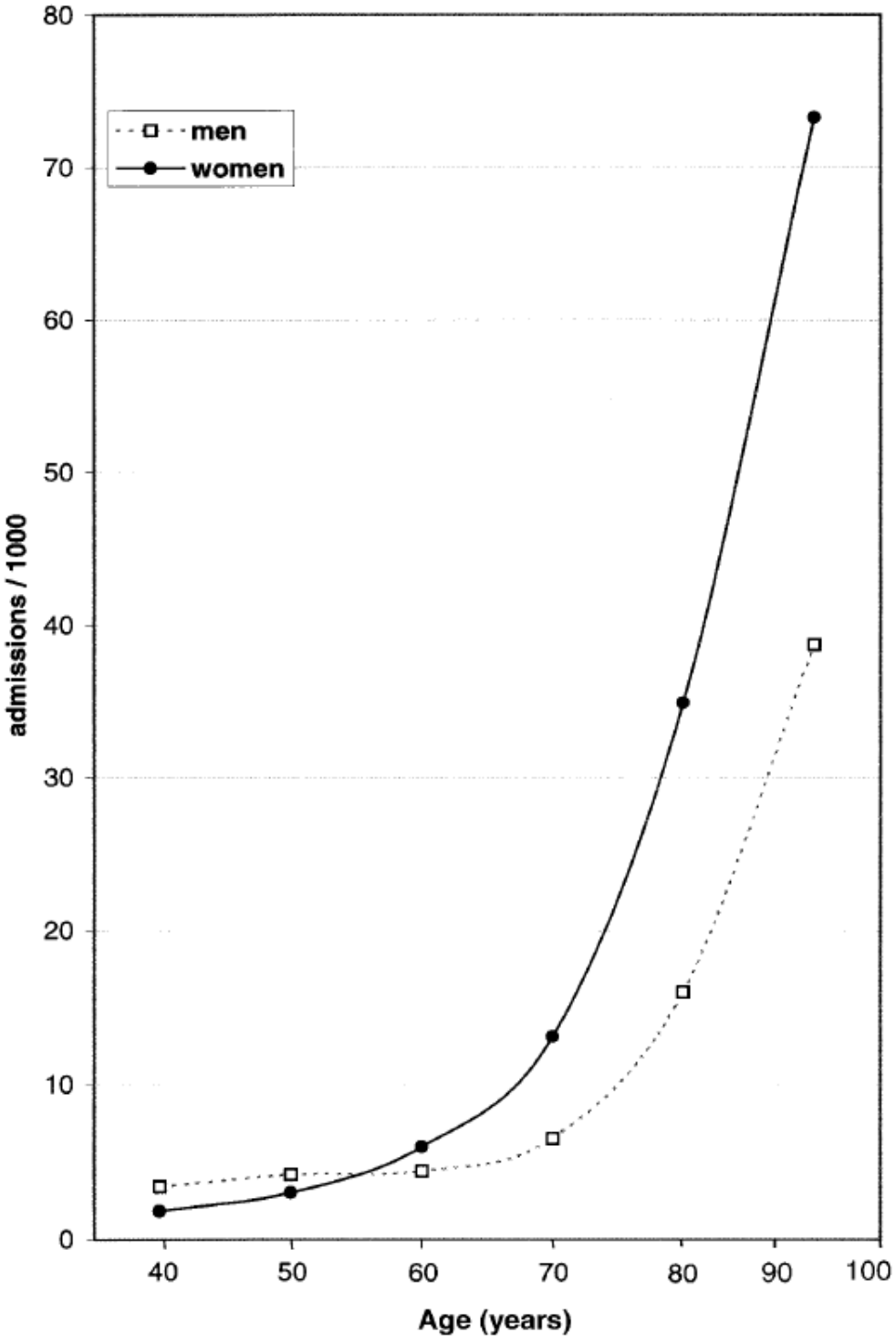

Figura 3: *Andamento dei ricoveri ospedalieri dovuti a cadute in un ospedale australiano.*

Le più comuni lesioni che richiedono ricovero ospedaliero comprendono fratture del femore, fratture delle gambe, fratture del radio, ulna e altre ossa del braccio e fratture del collo e tronco. In termini di morbilità[1] e mortalità, la più grave di queste lesioni è la frattura dell'anca. Gli anziani recuperano lentamente da fratture dell'anca e sono sensibili a complicazioni postoperatorie. Le fratture dell'anca possono risultare anche causa di morte e di quelli che sopravvivono, spesso non si recupera la completa mobilità [7].

Infine, le cadute possono portare anche alla disabilità o ad una ridotta mobilità che spesso comporta una maggiore dipendenza da altri e quindi una maggiore probabilità di ricovero in case di cura nel caso di anziani.

III Stato dell'arte dei fall detector

I metodi di monitoraggio statistico delle cadute visti in precedenza, si basano sulla compilazione di questionari e comunque sulla rilevazione dell'evento ormai già verificatosi da tempo. Non c'è quindi la possibilità di poter intervenire prontamente.

Nasce da questa esigenza quindi, un desiderio di fornire degli strumenti utili al pronto intervento in caso di caduta.

Nella maggior parte dei lavori accademici, i ricercatori hanno basato la loro strumentazione per la rilevazione di cadute su sensori come gli accelerometri, a partire da Lord e Colvin [8] nel 1991, seguiti da Williams [9] 1998, con un dispositivo autonomo integrato in una cintura che rilevava gli effetti dell'impatto sul pavimento e con un sensore al mercurio per rilevare l'inclinazione della persona e capire se fosse sdraiata.

Noury [10], ha progettato un sensore autonomo, da fissare sotto l'ascella, in grado di rilevare quando la velocità supera una certa soglia, la successione di istanti da una posizione verticale ad una posizione sdraiata e l'assenza di movimenti dopo la caduta. Il dispositivo riesce a raggiungere una sensibilità e una specificità intorno all'85% testandolo su 15 scenari di cadute differenti effettuate da 5 persone per 5 volte.

Zhang [11] ha posto un accelerometro tri-assiale in un cellulare. Il dispositivo è in grado di identificare la seguente sequenza di eventi: un'attività quotidiana, la caduta e quindi la persona che rimane immobile. Questa soluzione si è rivelata molto interessante dato che oggigiorno il cellulare è un dispositivo posseduto dalla maggior parte della popolazione.

Mathie [12] con un accelerometro triassiale posto in vita, utilizzando una serie di parametri tra cui l'angolo di inclinazione, l'accelerazione, la durata di una postura, l'energia spesa metabolica e le attività precedenti e successive alla caduta, è riuscito ad ottenere un sistema con sensibilità superiore al 98% e specificità tra 88% e 94%.

Tutti i lavori illustrati finora si basano su dispositivi e sensori da applicare al soggetto interessato. Queste soluzioni ovviamente comportano dei problemi operativi, dovuti spesso ad un'ergonomia insufficiente, che generano il "rifiuto" delle attrezzature da parte di chi le indossa.

[1] In medicina del lavoro e in medicina delle assicurazioni per morbilità si intende il rapporto percentuale tra il numero di giornate di assenza dal lavoro per malattia e il numero di giornate lavorative previste (ossia quelle effettuate più quelle mancate a causa della malattia)

Un'alternativa meno invasiva dal punto di vista personale, è utilizzare un approccio basato sull'image processing, ovvero sfruttare sistemi di videocamere che riprendono l'ambiente domestico ed analizzano le immagini con algoritmi specifici per la rilevazione delle cadute.

Wu [13] presso la University of Vermont-USA ha constatato che le velocità verticale e orizzontale risultano essere 3 volte superiori durante una caduta che per un qualsiasi altro movimento controllato, e che entrambe le velocità sono della stessa ampiezza, al momento della caduta mentre sono fortemente dissimili durante un movimento controllato. Questa osservazione ha ispirato Nait-Charif [14] e Rougier [15] a lavorare sul tracciamento dei movimenti della testa basando la rilevazione di cadute su un particle-filter. Mihailidis [16] dall'Università del Toronto-Canada, ha piazzato una videocamera sul soffitto e ha realizzato un sistema di fall detection che, testato su 21 volontari, è stato capace di rilevare il 77% delle cadute.

Alcuni prototipi sono stati implementati con successo e molti sono stati resi disponibili sul mercato in ambito commerciale. Tuttavia, non vi è uno sviluppo industriale significativo di sensori di caduta e l'uso di questi dispositivi nel quotidiano è abbastanza raro probabilmente per molteplici ragioni: un'affidabilità nei risultati abbastanza ridotta, ergonomia insufficiente, "rifiuto" delle attrezzature da parte di chi le indossa e alto tasso di falsi allarmi che generano segnalazioni inappropriate.

IV Principi e Algoritmi per la rilevazione di cadute

I metodi per la fall detection sono principalmente basati su modelli analitici, ma esistono anche quelli basati su "machine learning techiniques" ovvero metodi ad apprendimento. Vediamo rapidamente le principali differenze.

IV.I Analytical methods

Quasi tutte le cadute terminano con una posizione sdraiata sul pavimento, il più semplice approccio quindi è di rilevare questa posizione tramite sensori d'inclinazione. Questo metodo è appropriato per monitorare soggetti isolati che stanno svolgendo qualche attività, ma meno adatto per la rilevazione delle cadute di una persona anziana nel proprio ambiente domestico; infatti il semplice atto di sdraiarsi su un divano o di dormire su un letto, potrebbe generare un falso allarme. Una soluzione complementare è rilevare la posizione sdraiata sul pavimento, introducendo anche dei sensori di pressione, ma se la caduta non termina sul pavimento o se avviene in una zona non monitorata dai sensori, questa non verrà rilevata.

Quando una persona cade solitamente colpisce il pavimento o un ostacolo. L'impatto si traduce in un'inversione del verso del vettore di accelerazione lungo la direzione di caduta. Quest'inversione può essere rilevata tramite uno "shock detector" che è sostanzialmente un accelerometro con delle soglie prefissate. Anche se la maggior parte delle cadute si verifica nel "piano frontale" (avanti o indietro), la direzione della traiettoria di caduta è ovviamente variabile da una caduta all'altra, quindi anche la posizione del sensore sul corpo relativa al punto di impatto può modificare i parametri registrati al momento della caduta.

Altro parametro per rilevare la caduta può essere la mancanza di movimento. Dopo una caduta seria, in cui una persona potrebbe aver subito gravi traumi, solitamente questa non si muove e non cambia posizione. Tenuto conto di questa considerazione si potrebbero utilizzare sensori di movimento e di rilevazione delle vibrazioni piazzati in un'estremità del corpo (per esempio sulla caviglia), oppure semplici sensori ad infrarosso distribuiti nell'ambiente domestico. Punto cruciale di questo approccio è la scelta del tempo di latenza (il ritardo prima di prendere la decisione), che deve essere sufficientemente lungo per ridurre i falsi positivi, e si tradurrà quindi in un'attesa più lunga prima dell'intervento.

Durante la caduta c'è un periodo di "caduta libera" durante la quale la velocità verticale aumenta linearmente col tempo a causa dell'accelerazione gravitazionale [3]. Se si misura la velocità verticale dei movimenti volontari di una persona allora è possibile discriminarla rispetto quella relativa alla caduta potendo quindi fissare una soglia appropriata. La difficoltà sta nella scelta di questa soglia: se è troppo bassa il dispositivo tenderà a rilevare eventi negativi (falsi positivi); se la soglia è troppo alta potrebbero non essere rilevati eventi desiderati (falsi negativi). Questa soglia ha anche una variabilità che dipende da soggetto a soggetto. Per superare questa difficoltà, si pu ricorrere a un periodo di apprendimento supervisionato o meno. Durante il primo caso, verrà chiesto a chi indossa il dispositivo di effettuare una serie di movimenti volontari al fine di identificare la velocità normale di esecuzione. Nel secondo caso è sufficiente registrare i movimenti della persona durante lo svolgimento di normali attività e di procedere quindi con un'analisi statistica.

Anche se queste tecniche sono ben consolidate in ambienti controllati, come per esempio nei laboratori, devono essere modificate in ambienti non controllati dove non è possibile manipolare né l'illuminazione, né la definizione della scena (è ovviamente necessario che il soggetto in esame sia nel campo visivo). Inoltre, poichè il soggetto si muove in uno spazio tridimensionale, sarebbe utile l'impiego di tecniche più complesse come la "stereovisione", ovvero l'utilizzo di due videocamere prospettiche che inquadrano la stessa scena da posizioni leggermente sfalsate, in questo modo si riesce a percepire il senso di profondità.

Negli ultimi anni, queste tecniche stanno diventando sempre più realizzabili sia per un fattore tecnologico, sia soprattutto per uno economico, grazie alla nascita di telecamere a basso costo (web cam), la possibilità di trasmettere le immagini in modalità wireless su distanze brevi e la disponibilità degli algoritmi necessari. Tuttavia l'accettazione di questa tecnologia rappresenta un grande problema, in quanto richiede il posizionamento di telecamere nello spazio di vita della persona e in particolare in camera da letto e nel bagno, con conseguenti problemi di privacy.

IV.II Machine learning methods

Senza alcun modello analitico, è possibile applicare un approccio "intuitivo" per lo sviluppo di "machine learning systems" basati sulla rilevazione di cadute a partire da un primo periodo di osservazione (periodo di training) seguito poi da un periodo di classificazione. E' necessario settare dei criteri per la classificazione che risultino sufficientemente significativi.

Se si procede attraverso una periodo di training supervisionato, si può realizzare una rete neurale, che sarà poi utilizzata per classificare automaticamente scenari futuri. Solo le situazioni

incontrate durante la formazione potranno essere classificate, le altre saranno inserite in una classe “others”.

Se la formazione non è supervisionata il classificatore funzionerà bene se il tempo di training è abbastanza lungo, idealmente dovrebbe essere inversalmente proporzionale alla frequenza di caduta. D’altra parte però molto probabilmente le prime cadute non verranno rilevate in quanto la classe non sarà ancora nota al sistema [3].

Capitolo 1

Ambiente di lavoro

L'attività di tesi è stata svolta all'interno dello IAS-LAB dell'Università di Padova.

Il laboratorio è provvisto di varie telecamere prospettiche, ma principalmente è stato utilizzato l'Omnidome.

L'Omnidome integra una telecamera motorizzata (dome) ed una telecamera omnidirezionale, che cooperano attivamente grazie ad un software intelligente di elaborazione delle immagini allo scopo di supervisionare tutto l'ambiente circostante sfruttando i vantaggi di entrambi i tipi di telecamera.

La videocamera omnidirezionale consiste in una semplice videocamera prospettica puntata verso l'alto, con applicato in cima uno specchio convesso. Quest'ultimo consente alla videocamera di ampliare il proprio campo visivo mettendo quindi a disposizione una visione a 360 gradi dell'ambiente circostante. La compressione di una porzione così ampia di mondo nel proprio campo visivo però, comporta una perdita di risoluzione inevitabile. Inoltre la convessità dello specchio introduce una leggera deformazione nella parte inferiore dell'immagine.

Nell'immagine 1.1 è possibile vedere varie tipologie di specchi per la videocamera omnidirezionale. A seconda dello specchio utilizzato varia il campo visivo o addirittura varia la tipologia di videocamera da utilizzare (come nell'esempio "a3" e "d" della figura).

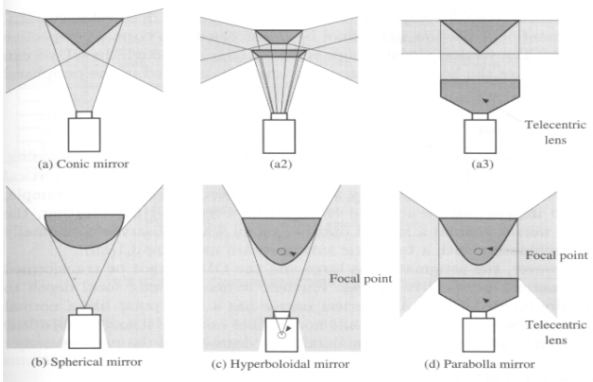

Figura 1.1: *Varie tipologie di specchio per videocamera omnidirezionale.*

La telecamera motorizzata, a differenza dell'omnidirezionale, offre una risoluzione decisamente superiore, ma un campo visivo molto più limitato.

Le informazioni estratte dalle immagini omnidirezionali sono utilizzate con lo scopo di controllare attivamente il dome e inquadrare ad elevata risoluzione le zone o le persone o gli oggetti di interesse. La telecamera motorizzata è attivata solo nel caso siano rilevati nell'ambiente eventi di interesse attraverso la telecamera omnidirezionale. In assenza, la telecamera viene lasciata inattiva.

Figura 1.2: *A sinistra un esempio di dome mentre a destra una videocamera omnidireazionale.*

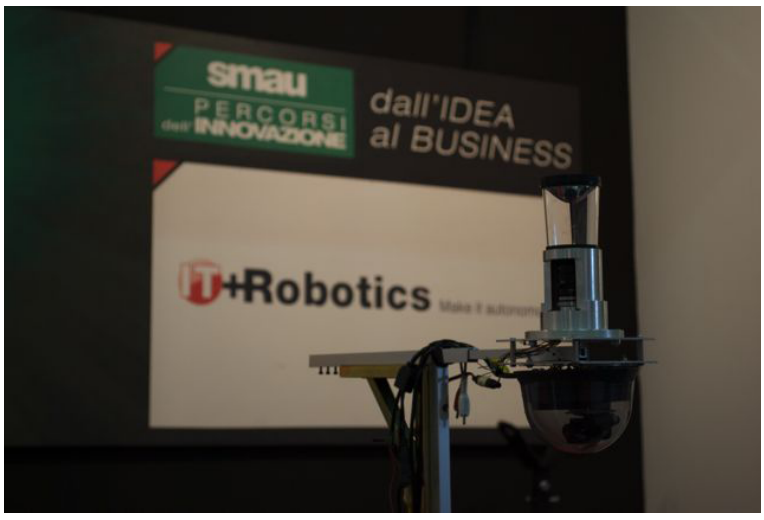

Figura 1.3: *Presentazione dell'Omnidome.*

I software attuali di inseguimento automatico di un soggetto in movimento non sono in grado di inseguire in modo affidabile il soggetto con telecamere motorizzate, mentre il soggetto può essere facilmente seguito in immagini omnidirezionali e usare questa informazione per

controllare automaticamente i movimenti della telecamera motorizzata. Infatti, le telecamere omnidirezionali permettono una visione globale in ogni istante di tutto l'ambiente circostante.

In laboratorio inoltre è stato installato un middleware NMM (Network-Integrated Multimedia Middleware) per avere la possibilità di interagire con le varie videocamere semplicemente con la creazione di nodi. La parte relativa a NMM verrà ampliamente trattata nel capitolo 3.

1.1 OpenCV

L'applicazione è stata sviluppata in C++, inoltre sono state utilizzate le librerie OpenCV (Open Source Computer Vision), emergenti nel campo della computer vision e sviluppate da Intel sotto una licenza di tipo OpenSource, compatibili con la GNU GPL.

È bene però prima fare chiarezza sull'uso e lo scopo di queste librerie. La capacità di interpretare ed utilizzare correttamente le informazioni acquisite da una videocamera o fotocamera attualmente presenta molti problemi insoluti. Convertire un'immagine in informazioni "oggettive" astraendone il contenuto dalla pura rappresentazione luminosa, sebbene sia un'operazione banale per un cervello umano adulto è, a tutt'oggi, un problema di elevata complessità per un sistema automatico. Oltretutto il campo di ricerca è evidentemente molto giovane, con meno di trent'anni di esperienza. In quest'ottica si inserisce la necessità di una base comune di potenti strumenti analitici, primo dei quali una libreria che raccolga le funzionalità degli algoritmi più utilizzati e citati in letteratura, oltre che una serie di formati di rappresentazione dei dati secondo standard aperti e condivisi.

Le librerie OpenCV nascono appunto a questo scopo; lo sviluppo prende le mossa da un gruppo di ricerca sponsorizzato da Intel. È infatti parzialmente basata sulla Intel Image Processing Library (IPL): tale prodotto è oggi integrato nella libreria commerciale IPP (Intel Integrated Performance Primitives), con cui conserva piena compatibilità e verso la quale rende disponibili un completo ventaglio di funzioni più specifiche. Tra i punti di forza sottolineiamo inoltre la politica di licenza utilizzata, in stile BSD e definita nella "Intel License Agreement For Open Source Computer Vision Library", completamente compatibile con la licenza GPL. A grandi linee questo permette una libera ridistribuzione sia in forma sorgente che binaria, anche all'interno di prodotti commerciali, a condizione di mantenere le note di copyright e di non utilizzare il nome Intel a scopo promozionale di prodotti derivati. Inoltre un'altra potenzialità offerta è la caratteristica di essere cross-platform: cioè possono essere compilate e usate sia sotto sistema operativo Microsft Windows che GNU/Linux. Questa caratteristica le rende molto appetibili per i requisiti di portabilià.

Capitolo 2

Fall Detector

In questo capitolo verrà illustrata la soluzione implementata per la fall detection. Inoltre verranno riportati i risultati ottenuti mediante test eseguiti in laboratorio.

2.1 Motion Detection

Il primo passo per la fall detection è ovviamente il rilevamento accurato della persona.

A tal fine è stato utilizzato l'algoritmo di background substraction realizzato da Francesco Bolgan nella sua tesi [17] per estrarre in ogni frame i blob rappresentanti appunto le persone in movimento.

L'idea di base del background subtraction è quella di identificare il livello di sfondo per un determinato video, segmentando ogni frame in altri due frame chiamati rispettivamente:

- Foreground Mask
- Background Mask

Una volta identificato il layer dello sfondo verrà semplicemente fatta la differenza fra il frame corrente e quello di background, ottenendo quindi le differenze fra essi che rappresentano gli oggetti in movimento nella scena.

Nella sua tesi, Bolgan ha realizzato due algoritmi per l'estrazione del movimento, l'adaptive background subtraction e il frame differencing-background subtraction.

L'algoritmo di adaptive background subtraction (d'ora in poi sarà chiamato algoritmo ABS) si basa sulla costruzione di una immagine background che si possa aggiornare nel tempo e che possa servire da confronto nell'estrazione del foreground. A tale scopo, data una sequenza abbastanza lunga di frame (indicati nel seguito con $frame_n$, n l'indice dell'immagine n-esima), si eseguono i seguenti passi (tutte le formule proposte operano sui singoli pixel):

1. inizializzazione. Si assume come background iniziale una immagine "speciale", come ad esempio la scena che si vuole riprendere priva di qualsiasi soggetto in movimento o

stazionario, oppure, se ciò non fosse possibile, semplicemente il primo frame a disposizione (si indica poi nel seguito con $scarto_n$ lo scarto quadratico medio relativo al frame n-esimo e con $stddev_n$ la deviazione standard relativa al frame n-esimo):

$$background_0 = frame_0$$
$$scarto_0 = (frame_0)^2$$
$$stddev_0 = frame_0$$

2. estrazione del foreground. Si confronta il frame corrente con il background a disposizione con la seguente formula:

$$|frame_n - background_{n-1}| > cstddev_{n-1}$$

I pixel che soddisfano tale condizione sono considerati facenti parte del foreground;

3. aggiornamento dello scarto. Si aggiorna lo scarto con la seguente formula:

$$scarto_n = \alpha \cdot scarto_{n-1} + (1 - \alpha) \cdot |frame_n - background_{n-1}|^2$$

4. aggiornamento del background e della deviazione standard.

$$background_n = \alpha \cdot background_{n-1} + (1 - \alpha) \cdot frame_n$$
$$stddev_n = \sqrt{scarto_n}$$

A questo punto si ha a disposizione il foreground, memorizzato in una maschera binaria i cui pixel posti a uno identificano i pixel appartenenti a soggetti in movimento, sul quale si possono poi eseguire operazioni morfologiche per attenuare il rumore.

L'algoritmo ibrido frame differencing-background subtraction (nel seguito indicato come algoritmo FDBS) invece, non si basa più sul background per l'individuazione del movimento, ma su due differenze di frame consecutivi, e al background lascia solo il compito di "complementare" e migliorare il foreground.

Per estrarre solo i pixel appartenenti a soggetti in movimento le due differenze sono filtrate da una soglia (indicata nel seguito come T_1).

I passi dell'algoritmo sono i seguenti (tutte le formule proposte operano sui singoli pixel):

1. inizializzazione. Si pone la soglia T_1 uguale ad un predeterminato valore non nullo, e si inizializza il background come nel primo passo dell'algoritmo ABS:

   $background_0 = frame_0$

   $T_1 = cost$

2. si opera sul seguente confronto:

   $$|frame_n - frame_{n-1}| > T_1 AND |frame_n - frame_{n-2}| > |T_1|$$

   I pixel che risultano maggiori della soglia in entrambe le due differenze entrano a far parte del foreground, memorizzato in una maschera binaria;

3. si esegue sulla maschera binaria ottenuta al passo precedente una segmentazione considerando adiacenti due pixel che distano meno di una predeterminata distanza, la quale varia a seconda della scena ripresa e dalle dimensioni dei soggetti presenti. Si ottengono così vari gruppi di pixel (nel seguito segmenti), dai quali si estraggono le regioni di interesse (nel seguito ROI), ossia i più piccoli rettangoli che circoscrivono ogni singolo segmento;

4. si effettua una differenza tra il background e il frame corrente per ogni singola ROI:

   $$\forall ROI, |frame_n - background_{n-1}| > T_2$$

   (T_2 è un'altra soglia che può essere diversa dalla precedente T_1). In questo modo si modifica la maschera ottenuta al secondo passo con tutti i pixel del soggetto in movimento, completando o restringendo il foreground;

5. si aggiorna il background solo per quei pixel non appartenenti al foreground:

   $$background_n = \alpha \cdot background_{n-1} + (1 - \alpha) \cdot frame_n$$

Per maggiori approfondimenti e precisazioni, si rimanda alla tesi di Bolgan.

2.2 Feature Extraction

Per l'identificazione di cadute, si possono estrarre varie feature analizzando l'oggetto rilevato e il suo bounding box. Degli esempi possono essere il gradiente orizzontale e verticale, l'aspect ratio, distanza fra bounding box attuale e precedente, tempi di immobilità ed eventualmente anche ampiezza e direzione del movimento. Illustriamo brevemente queste features.

2.2.1 Gradiente

Il gradiente, calcolato tramite filtro di Sobel disponibile nelle OpenCV, dà una rappresentazione dei bordi presenti nell'immagine. A seconda del kernel selezionato, è possibile individuare i bordi orizzontali o i bordi verticali. Fissando gli assi cartesiani di riferimento sul piano immagine (x parallelo alla base e y parallelo all'altezza), solitamente quando si verifica una caduta i maggiori cambiamenti si hanno principalmente in una direzione x o y. Per ogni pixel quindi viene calcolato il gradiente orizzontale (G_x) e quello verticale (G_y).

Nella tabella (paper [18]) sottostante vengono confrontati i due gradienti.

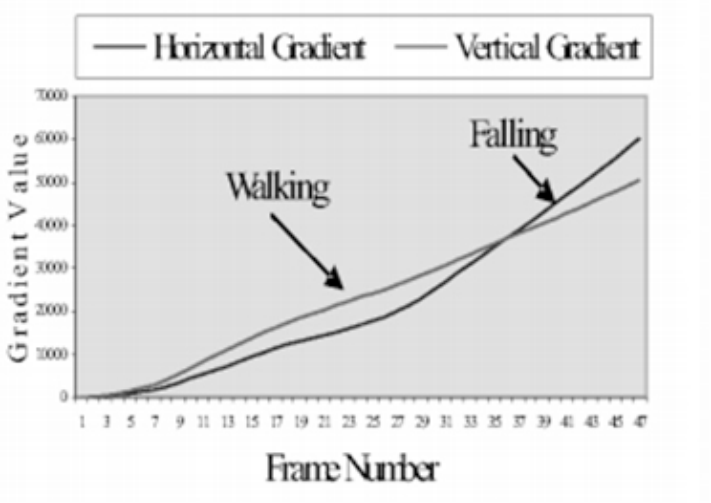

Figura 2.1: *Andamento del gradiente.*

2.2.2 Aspect Ratio

L'aspect ratio di una persona semplicemente il rapporto fra l'altezza del bounding box che la contiene e la base. In figura (paper [18]) sottostante viene riportato l'andamento dell'aspect ratio durante lo svolgimento di normali attività.

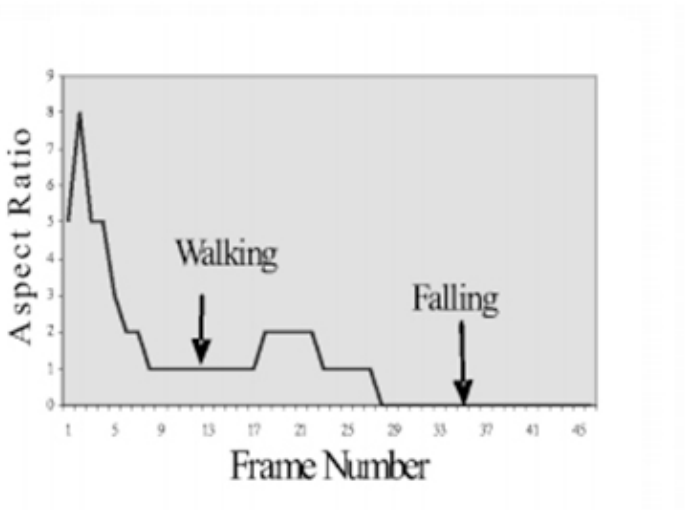

Figura 2.2: *Andamento dell'aspect ratio.*

2.2.3 Distanza fra proporzione fra BBox (Beta)

Il parametro beta rappresenta la distanza di proporzioni fra il bounding box attuale e quello del frame precedente. Questo parametro quindi esprime quanto velocemente si verifica un cambiamento nelle proporzioni di un blob rilevato.

Rappresentando con w la base del boundingbox attuale e w' la base di quello precedente e similmente con h e h' le altezze, allora il parametro beta verrà calcolato semplicemente come:

$$\beta = \sqrt{(w - w')^2 + (h - h')^2}$$

2.2.4 Tempi di immobilità

Se la caduta è seria e quindi potrebbe avere causato gravi danni al soggetto, solitamente la persona rimane a terra per un certo lasso di tempo. Questo periodo di immobilità in seguito alla caduta può essere utilizzato come parametro di conferma per la detection. Ovviamente bisogna trovare un giusto trade off per l'attesa, altrimenti si rischia di avere troppi falsi positivi oppure di non intervenire tempestivamente.

2.2.5 Ampiezza e direzione del movimento

Sempre sfruttando il gradiente, è possibile avere una stima dell'ampiezza e della direzione del movimento.

Identificando con $S_x(x, y)$ e $S_y(x, y)$ il valore di gradiente orizzontale e verticale ottenuto tramite filtro di Sobel per ogni pixel, allora si può calcolare l'ampiezza del movimento per ogni pixel in questo modo:

$$\text{M(x,y)} = \sqrt{S_x^2(x, y) + S_y^2(x, y)}$$

Eventualmente, in presenza di un sistema di telecamere stereo, e quindi con la possibilità di avere coordinate tridimensionali dei blob in movimento, è possibile calcolare anche la direzione del movimento in questo modo:

$$\phi(x, y) = \arctan \frac{S_y(x,y)}{S_x(x,y)}$$

2.3 Algoritmo implementato

Vediamo ora l'algoritmo implementato per la rilevazione delle cadute.

La prima analisi riguarda l'aspect ratio del bounding box (bbox). Se h/b risulta essere minore di 1 allora il bbox è un possibile candidato per la caduta e passerá alle fasi successive di analisi.

Tramite il filtro di Sobel ottengo 2 immagini chiamate "sobelX" e "sobelY" dove ho la rappresentazione dei bordi orizzontali e verticali calcolati con i gradienti.

Basandosi sull'esperienza dei paper [18] e [19], in fase di caduta il gradiente Gy risulta minore rispetto al Gx, quindi al verificarsi di questa condizione avvio il conteggio di caduta.

Ogni oggetto trackato ha un "fallCount" inizializzato a 0, che rappresenta il numero di frame consecutivi per i quali è stata rilevata una caduta. La gestione di questo contatore, dopo il primo incremento è la seguente:

- Se l'oggetto si muove o le sue dimensioni variano il conteggio si ferma e il contatore viene riportato a 0, infatti potrebbe esserci stato un falso allarme e se l'oggetto è ancora in movimento o le sue dimensioni continuano a cambiare vuol dire che non è a terra.
- Se invece le dimensioni e la posizione rimangono stabili allora il fallCount viene incrementato.

Quando il contatore raggiunge la soglia limite (fissata sperimentalmente a 10 frame), è rilevata la caduta.

Per il controllo della variazione di dimensioni del bbox, viene utilizzato il parametro beta. Se beta supera un certo valore di soglia (fissato sperimentalmente e facendo riferimento ai paper analizzati) , allora vengono considerate variate le dimensioni del bbox. Per la variazione di posizione viene fatto un ragionamento simile al calcolo del beta, ovvero calcolo la distanza euclidea fra l'attuale posizione e quella del frame precedente.

Riassumendo, vengono analizzati solo i bbox che risultano essere rettangoli che non si sviluppano in altezza, a questo punto il controllo del gradiente fa partire il conteggio e il contatore viene incrementato o resettato analizzando i movimenti dell'oggetto.

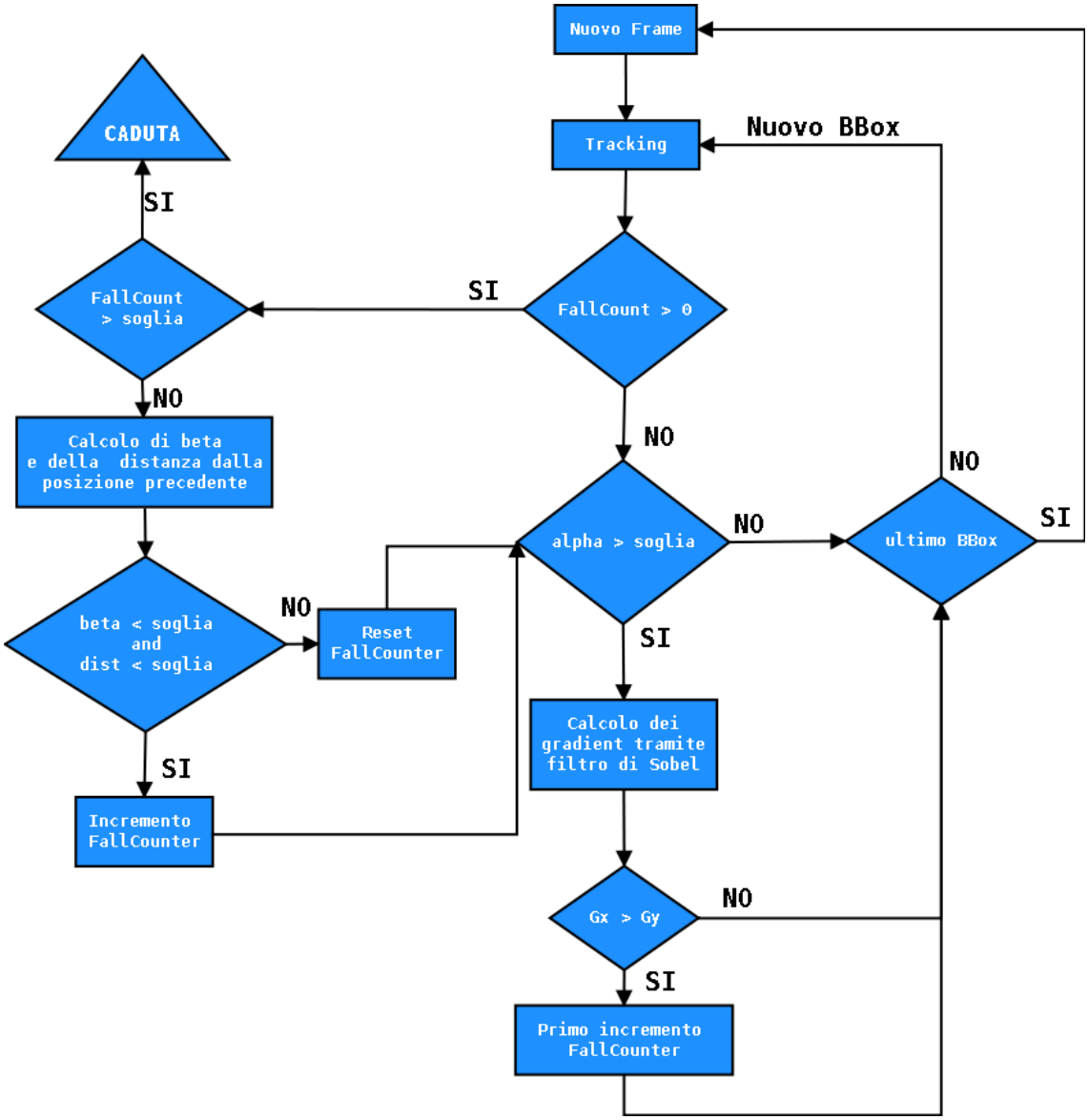

Figura 2.3: *Diagramma di flusso dell'algorimo implementato.*

2.4 Valutazione per i Fall Detector

Attualmente è difficile confrontare le performance di differenti rilevatori di caduta analizzando solo i risultati dei singoli test, infatti questi non tengono conto dei diversi approcci adottati e soprattutto non è garantito che le prove eseguite siano le stesse dato che in letteratura non è mai stato definito uno "standard" per i test e quindi ogni sviluppatore ha eseguito molto probabilmente, delle prove differenti per valutare la qualità della soluzione realizzata.

Sarebbe importante quindi riuscire a delineare una serie di criteri oggettivi comuni da adottare in futuro per la valutazione dei sensori di caduta.

2.4.1 Criteri di qualità

La rilevazione risulta positiva ovviamente se il fall detector riconosce la caduta quando essa si verifica, negativa altrimenti. La qualità del detector però non è definibile soltanto da un output binario positivo o negativo proveniente da un singolo test, ma è necessaria un'analisi statistica eseguita su una serie di test.

Definiamo quindi 4 possibili classi:

- Veri positivi (true positive TP): si verifica una caduta e il detector la rileva;
- Falsi positivi (false positive FP): il detector rileva una caduta ma non se ne sono verificate;
- Veri negativi (true negative TN): si verificano normali movimenti senza cadute e il detector non rileva nulla;
- Falsi negativi (false negative FN): si verifica una caduta ma questa non viene rilevata.

Per valutare la risposta a queste 4 situazioni appena delineate si possono utilizzare i seguenti criteri:

- Sensitivity: capacità di rilevare le cadute. Il sistema potrebbe rilevare anche dei falsi positivi, ma l'importante è che le cadute vengano rilevate. Valuta quindi quanto il sistema è sensibile alle cadute.

$$Sensitivity = \frac{TP}{TP+FN}$$

- Specificity: capacità di rilevare solo le cadute. Il sistema dev'essere in grado di discriminare bene le cadute rispetto gli altri eventi e quindi evitare i falsi positivi.

$$Specificity = \frac{TN}{TN+FP}$$

Questi due parametri utilizzati anche da Noury in [3], sono sembrati i più adatti per la valutazione di qualità di un sistama di fall detection, dato che misurano sia la capacità di rilevare le cadute, che la capacità di rilevare solo le cadute.

2.4.2 Set up degli esperimenti

Le modalità di caduta sono molto varie, quindi risulta essere necessario testare il detector con un certo numero di scenari di caduta e con un certo numero di "pseudo" situazioni di caduta.

Poiché la maggior parte delle cadute avvengono durante i movimenti intenzionali di una persona, esse si verificano principalmente sul piano frontale, avanti o indietro: inciampare su un ostacolo durante una camminata, scivolare all'indietro su un terreno bagnato, o il semplice alzarsi da una poltrona nel caso di anziani.

Se una persona si sbilancia nella direzione di marcia, cercherà inizialmente di recuperare l'equilibrio con alcuni passi in avanti, amplificando così il movimento. Probabilmente in fase di caduta, le braccia verranno portate in avanti per tentare di proteggersi. Potrebbe anche cadere sulle ginocchia rendendo la caduta più "verticale". Se si verifica uno squilibrio all'indietro la persona cercherà di sedersi per attenuare l'intensità dell'impatto.

In alcuni casi però la caduta si può verificare lateralmente, sia durante l'alzarsi, sia dopo una perdita dell'equilibrio tentando di appoggiarsi ad un muro.

Ci sono anche movimenti di vita quotidiana durante i quali l'ampiezza o l'intensità del movimento possono essere simili a quelle incontrate nelle situazioni accidentali come per esempio, le azioni di sdraiarsi o di sedersi, se effettuate in fretta.

La tabella successiva [3], basandosi su quanto appena detto, cerca di delineare quindi una serie di scenari di caduta possibili per la valutazione dei fall detector.

Category	Name	Outcome
Backward fall (both legs straight or with knee flexion)	Ending sitting	**Positive**
	Ending lying	**Positive**
	Ending in lateral position	**Positive**
	With recovery	**Negative**
Forward fall	On the knees	**Positive**
	With forward arm protection	**Positive**
	Ending lying flat	**Positive**
	With rotation, ending in the lateral right position	**Positive**
	With rotation, ending in the lateral to the left position	**Positive**
	With recovery	**Negative**
Lateral fall to the right	Ending lying flat	**Positive**
	With recovery	**Negative**
Lateral fall to the left	Ending lying flat	**Positive**
	With recovery	**Negative**
Syncope	Vertical slipping against a wall finishing in sitting position	**Negative**
Neutral	To sit down on a chairthen to stand up (consider the height of the chair)	**Negative**
	To lie down on the bed then to rise up	**Negative**
	Walk a few meters	**Negative**
	To bend down, catch something on the floor, then to rise up	**Negative**
	To cough or sneeze	**Negative**

Figura 2.4: *Scenari di caduta.*

Sebbene l'obiettivo principale del sensore sia quello di rilevare le cadute di anziani, è effettivamente improponibile eseguire dei test con loro.

Il soggetto che si presta per la caduta non dovrà essere vincolato in alcun modo, deve poter adattare liberamente la propria andatura e la propria velocità nello svolgimento di qualsiasi scenario prestabilito.

Se possibile, l'ideale sarebbe realizzare almeno 3 test per ogni scenario ottenendo quindi 60 varianti di cui metà con cadute e metà con attività normali.

2.4.3 Esperimenti e risultati

Il software prodotto è stato testato su molteplici video acquisiti in laboratorio.

Per tutti i filmati i primi frame sono di solo sfondo, ovvero non compare nessuno oggetto in movimento, questo ovviamente per favorire il buon funzionamento del background substraction.

Dato che alcuni filmati potevano anche contenere più di una caduta, sono stati considerati suddivisi in "slot temporali", ovvero un primo slot di camminata o di azione normale, uno slot di caduta, e poi eventualmente ancora camminata e ancora caduta. In questo modo è stato possibile differenziare gli istanti temporali dove cercare falsi/veri positivi/negativi.

Negli slot temporali di camminata se il sistema non rilevava niente veniva classificato come un TN altrimenti se trovava una caduta era un FP. Negli slot di caduta invece se rilevava qualcosa era un TP se non rilevava la caduta invece era un FN.

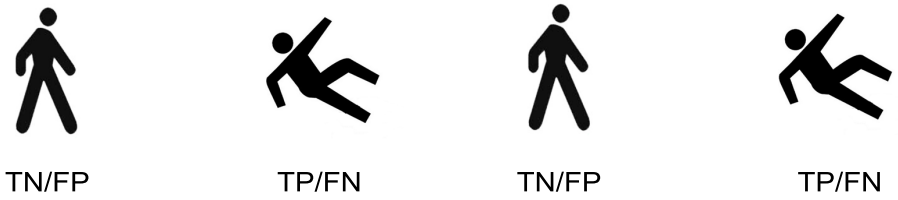

Figura 2.5: *Slot sequenziali per i video di test.*

Riporto ora dei frame d'esempio (da fig.2.6 a fig.2.12) che illustrano il funzionamento del fall detector e una valutazione delle prestazioni. Le immagini acquisite dalla videocamera omnidirezionale vengono dedistorte e mostrate "panoramiche".

Il rilevamento della caduta verrà segnalato con un bounding box rosso intorno alla persona interessata, inoltre verrà segnalato l'id del bbox. Questa identificazione è stata pensata in modo da poter trasmettere informazioni sulla persona trackata (identificata appunto da un id numerico) fra varie telecamere.

Volendo è anche possibile abilitare la visualizzazione di tutti i bbox trackati (verranno mostati a video come rettangoli verdi meno marcati), avendo cosi anche un'idea di tutti gli elementi che il software identifica come in movimento, come viene mostrato in fig.2.7.

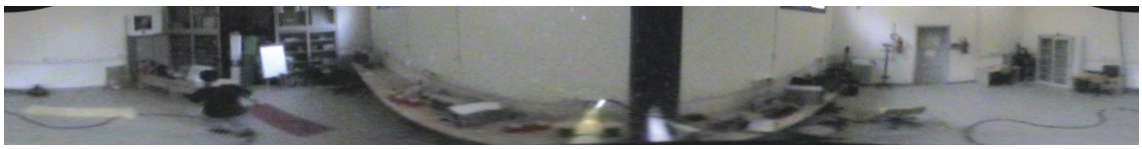

Figura 2.6: *Esempio di frame acquisito.*

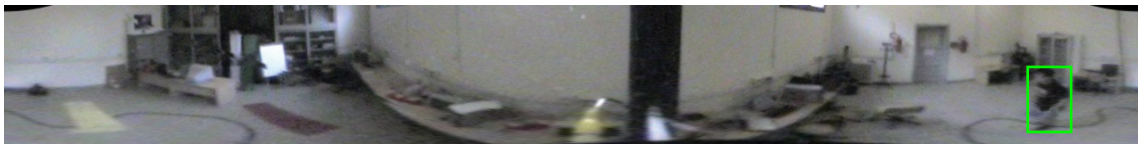

Figura 2.7: *Esempio di frame con bounding box del tracking.*

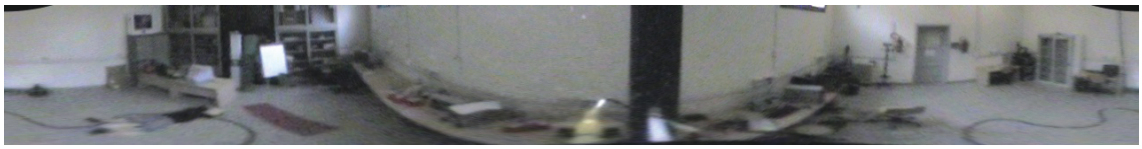

Figura 2.8: *Esempio di frame con caduta prima della rilevazione.*

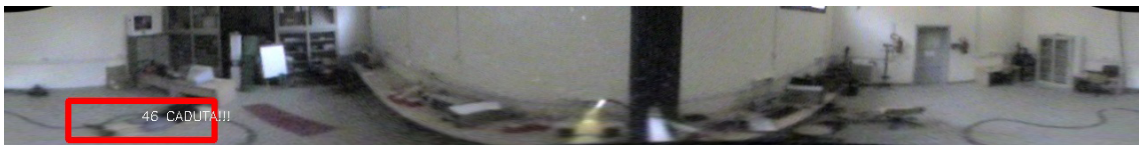

Figura 2.9: *Esempio di frame con caduta con rilevazione.*

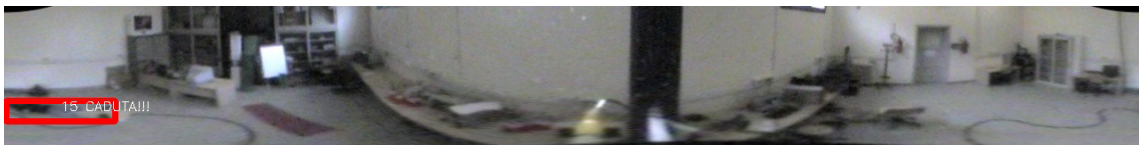

Figura 2.10: *Esempio di frame con caduta.*

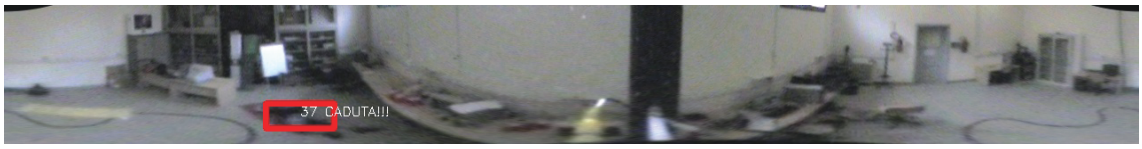

Figura 2.11: *Esempio di frame con caduta.*

Figura 2.12: *Esempio di frame con caduta.*

Vediamo ora i risultati ottenuti con i vari esperimenti effettuati in laboratorio.

VIDEO	TP	TN	FP	FN
caduta1	2	3	0	0
caduta2	1	2	1	0
caduta3	2	3	0	1
caduta4	1	2	0	0
caduta5	2	1	2	0
caduta laterale1	1	2	0	0
caduta laterale2	1	2	0	0
caduta posteriore1	1	2	0	0
caduta posteriore2	1	2	0	0
caduta posteriore3	1	1	1	0
folla1	2	0	2	0
folla2	1	2	0	0
folla3	3	3	1	1
camminata1	0	1	0	0
camminata2	0	1	0	0
camminata3	0	1	0	0
	19	28	7	2

Tabella 2.1: *Risultati ottenuti in laboratorio.*

SENSITIVITY 90,48%

SPECIFICITY 80,00%

I video da *caduta1* fino a *caduta5* riprendono varie tipologie di cadute generiche. I *caduta laterale* e *caduta posteriore* invece, si concentrano su cadute fatte sul fianco e di schiena. I *folla*, sono dei test fatti con più persone nella scena, questi sono stati utili per vedere il comportamento del sistema quando ci sono più soggetti da trackare. Infine i *camminata* sono dei video privi di cadute nei quali un soggetto si muove nella stanza liberamente in modo da testare eventuali falsi allarmi.

Analizzando i risultati possiamo notare che la *sensitivity* è molto alta. Il valore risulta essere elevato in quanto si sono verificati pochi falsi negativi e questo vuol dire che il sistema nei test eseguiti ha quasi sempre rilevato una caduta quando questa si verificava.

La *specificity* invece, influenzata dai falsi positivi, risulta più bassa, ma sempre ad un livello accettabile ed in linea con i lavori già realizzati. Il sistema si è rivelato sensibile quindi a falsi allarmi. La principale causa è imputabile al sistema di tracking, infatti molto spesso quando una persona entrava nella scena rilevata dall'Omnidome, non le veniva associato un solo bounding box (situazione ideale di tracking), ma il soggetto veniva "segmentato", ovvero il tracking asso-

ciava più bounding box a varie parti del corpo, per esempio uno per il busto, uno per le braccia, uno per la testa e uno per le gambe. Questo, ovviamente, influisce negativamente sul sistema di fall detection.

Figura 2.13: *Esempio di tracking non ideale. Il soggetto è segmentato in 3 bounding box.*

Figura 2.14: *Ingrandimento della figura precedente.*

Una ristrutturazione al sistema di tracking, oppure l'utilizzo di un altro approccio (molto interessante sembrerebbe l'utilizzo del tracking basato sugli istogrammi), migliorerebbe di molto il fall detector, riducendo sicuramente il numero di falsi positivi e di falsi negativi.

Capitolo 3

NMM (Network-Integrated Multimedia Middleware)

Network-Integrated Multimedia Middleware o più semplicemente NMM è un progetto nato da un'idea del professor Philipp Slusallek della Saarland University, il quale assieme al Dr. Marco Lohse e a Michael Repplinger ha fondato nel 2005 una spin-off dell'università chiamata Motama e situata a Saarbrücken (Germania). Quest'azienda che utilizza NMM si defnisce specializzata in progettazione e sviluppo di sistemi multimediali distribuiti. In particolare offre tecnologie e soluzioni ingegneristiche, e sviluppo software richiesto dal cliente con supporto e documentazione (presente anche sul sito).

In realtà NMM è open-source, ma per particolari applicazioni Motama fornisce un'adeguata assistenza commerciale. Ciò signifca che i file sorgente sono disponibili in rete, ma esiste un supporto tecnico aziendale nel caso si abbiano esigenze specifche e non gestibili con applicativi propri integrati con i file disponibili.

Come si può comprendere dal nome, NMM è un middleware che fornisce servizi per la comunicazione multimediale sulla rete. In particolare cito alcune delle applicazioni per cui è stato pensato e nelle quali viene o potrebbe venire utilizzato:

- "Passatempi" domestici multimediali basati sulla rete: accesso trasparente a dispositivi remoti e a computazioni remote (es: elaborazione di video), accesso trasparente ad alta velocità ad archivi remoti e integrazione con dispositivi mobili.
- Elaborazione distribuita e parallela di agenti mediatici: sistemi distribuiti anche su larga scala, transcodifica multimediale, indicizzazione di dati multimediali.
- Streaming distribuiti: sistemi client-server, server per lo streaming distribuito, servizi di streaming per i singoli utenti.
- Installazioni multimediali: sistemi di visualizzazione su larga scala (video wall), sistemi distribuiti per la variazione dei contenuti (es: CAD), installazioni multimediali artistiche.

3.1 Architettura generale di NMM

Lo sviluppo tecnologico ha portato ad un pesante utilizzo di dispositivi multimediali connessi alla rete. La normale diffusione dei dati o degli stream di solito è basata su di un approccio client-server con quest'ultimo incaricato di contenere i file e renderli disponibili. I client però non sempre necessitano esclusivamente dei server e a volte solo una comunicazione per il controllo o lo scambio dati con gli altri client permetterebbe di soddisfare completamente le loro richieste.

L'idea di fondo di NMM sta proprio nella realizzazione di uno strato in grado di rendere trasparente l'utilizzo alla totalità degli utenti delle risorse nella rete, come indicato anche nella definizione di middleware. Le figure rendono l'idea di come NMM trasformi le comunicazioni. Si può osservare infatti che nella prima vi è una distinzione tra client e server, mentre nella seconda l'application si trova connessa tra System1 e System2 senza distinzione dei ruoli di ciascuno di essi sulla rete.

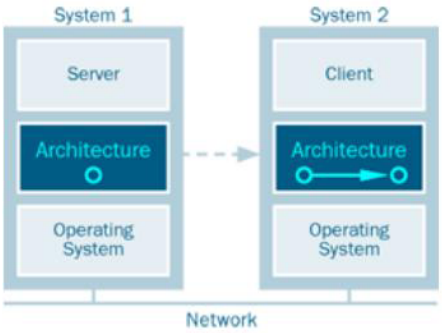

Figura 3.1: *Esempio di architettura Client - Server.*

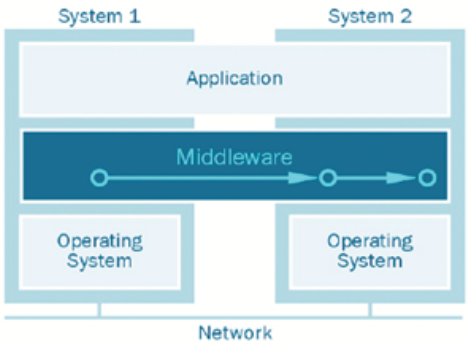

Figura 3.2: *Esempio di architettura basata su middleware.*

Vediamo ora nel dettaglio l'architettura di NMM con l'aiuto della figura 3.3.

Abbiamo un micro-core open-source che sta alla base del middleware e permette l'integrazione di varie tecnologie di rete e dispositivi. Per gestire le comunicazioni abbiamo un supporto per i protocolli standard come TCP, UDP, RTP, ecc. e per i formati standard come XML.

Con un elevato numero di plugin, NMM fornisce un alto livello di astrazione e di trasparenza grazie ad un'organizzazione dei collegamenti simile ad un grafo distribuito.

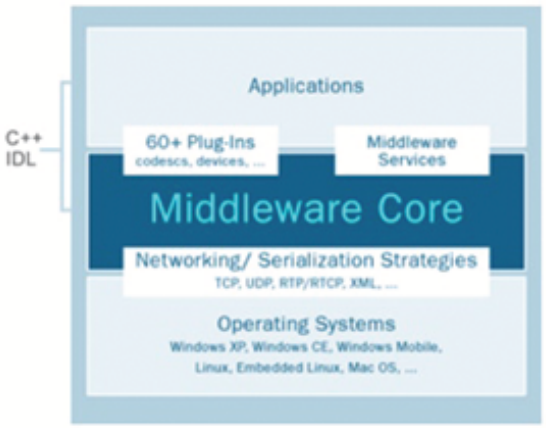

Figura 3.3: *Architettura di NMM.*

In piccolo a sinistra nella figura, viene segnalato inoltre l'utilizzo di un IDL[1] e del C++, infatti sono presenti in NMM un sistema di interfacce di controllo basate su di un Interface Definition Language e delle API object-oriented implementate in C++. Vengono poi fornite tutte le specifiche necessarie per l'utilizzo in diverse piattaforme come Windows XP, Windows CE, Windows Mobile, Linux, Embedded Linux, Mac OS, ecc.

[1]L'interface Description Language è l'astrazione usata per separare le interfacce degli oggetti dalle loro implementazioni. Utilizzato per descrivere le interfacce degli oggetti e i tipi dei parametri, non è un linguaggio di programmazione (non serve per implementare gli oggetti o per realizzare client che accedano agli oggetti).

3.2 L'organizzazione a node

Nelle reti in cui NMM è integrato, ogni componente hardware o software viene identificato come un'entità a sé stante e gli viene dato il nome di node. Ogni node per comunicare con gli altri ha a disposizione delle porte di input e/o output chiamate jack. Ogni jack è predisposto per essere utilizzato nelle comunicazioni di un certo formato video o audio specifico (per esempio yuv o rgb per il video).

Ogni node in base al numero di jack che possiede e alle funzionalità per cui è stato programmato, può essere classificato come:

- Source: produce dati (si interfaccia con l'esterno di NMM) e li inoltra ad un output jack.
- Sink: riceve dati da un input jack e li consuma (anche qui si esce dalla rete di NMM).
- Filter : ha un input jack ed un output jack con la possibilità di modificare i dati che lo attraversano, ma non il format.
- Converter : ha un input jack ed un output jack con la possibilità di modificare i dati ed il format dei dati che lo attraversano.
- Multipexer : ha vari input jack ed un unico output jack.
- Demultiplexer : ha un unico input jack e vari output jack.

Tutte queste classificazioni e questa struttura permettono la creazione di un diagramma di flusso (flow graph) in grado di organizzare la rete e tutti i suoi componenti. Ogni node però può essere collegato ad un altro node solamente se i jack che partecipano alla comunicazione hanno lo stesso format.

Nella figura 3.4 vediamo un realizzazione di un flow graph all'interno di una rete. In questo esempio vi è il node di tipo source che fa audio render collegandosi ad una fonte audio come potrebbe essere un file o uno stream su Internet. Con il suo output jack invia i suoni nel formato audio/mpeg al converter node chiamato audio decoder, il quale trasforma il format in audio/raw e lo rende disponibile ad un sink node attraverso la comunicazione tra i loro jack. In questo modo abbiamo reso disponibile da una parte all'altra della rete una sorgente sonora, infatti l'ascolto potrà avvenire in qualsiasi sink node connesso.

Due node attraverso i rispettivi jack possono scambiare dati se di un particolare format, ma tutte le comunicazioni avvengono con un sistema di messaggi uniforme. Nonostante ciò è possibile classificare due tipologie di messaggistica: a buffer ed a event. I primi sono utilizzati per la comunicazione di dati multimediali, mentre i secondi sono specifici per le informazioni di controllo.

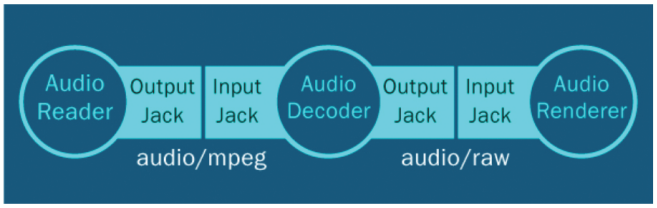

Figura 3.4: *Esempio di un flow graph con NMM.*

Basandoci sulla figura 3.4 si può pensare ad esempio che i suoni siano distribuiti tramite buffer, mentre un controllo remoto del volume tramite event.

In base invece a come interagiscono due node si possono definire le comunicazioni instream ed out-of-band. La prima è il tipico scambio di dati tra due node e può essere a sua volta suddivisa in downstream (dai source node ai sink node) ed upstream (dai sink node ai source node). La comunicazione out-of-band è invece quella parte di scambio dati che avviene tra i node di NMM e tutto il mondo esterno, come applicazioni, file, Internet, ecc. Molte volte queste interazioni utilizzano la modalità ad event come visto per il controllo remoto del volume nell'esempio precedente.

Gli event possono essere inviati manualmente oppure, in maniera più sicura e conveniente, utilizzando i metodi messi a disposizione dalle interface create seguendo l'architettura object-oriented. Le varie interfaces sono descritte in un particolare IDL chiamato NMM IDL molto simile all'IDL definito da CORBA[2]. Ogni file contenente le interfaces comincia con la lettera maiuscola "I". Partendo poi da questi file il compilatore IDL di NMM crea le classi di interfaccia per l'interazione con gli oggetti e le classi di implementazione dove risiedono le funzionalità per i node.

Consideriamo ora le potenzialità dei node anche per la distribuzione sulla rete. Nella figura 3.5 viene riportato l'esempio della 3.4 con aggiunto il posizionamento dei node. Vediamo che sono state introdotte alcune interfaces e una suddivisione dei componenti in tre sistemi differenti ognuno associato ad un host differente. In accordo con la definizione di middleware, i System 1, System 2 e System 3 possono trovarsi in qualsiasi punto sulla rete senza distinzione di sorta. In particolare notiamo che le tre interfaces appartengono al System 1, il source node al System 2 e il converter node con il sink node al System 3. L'esempio si sviluppa come in figura 3.4 anche se in maniera distribuita, con l'aggiunta del System 1 che supervisiona tutto l'insieme dei node. Grazie agli INode ha il pieno controllo sul flusso dei dati e può decidere di settare le varie caratteristiche di ogni interazione; ad esempio può alzare o abbassare il volume del suono e decidere quando far partire un particolare stream. Sulla rete viaggiano senza distinzione i comandi per i node e i dati veri e propri ottenendo così il sistema di messaggi uniforme citato in precedenza. Infine è bene notare le doppie frecce presenti tra i node e i rispettivi INode, le quali evidenziano come la comunicazione può avvenire in entrambi le direzioni.

[2]CORBA (Common Object Request Broker Architecture) uno standard sviluppato da OMG per permettere la comunicazione fra componenti indipendentemente dalla loro distribuzione sui diversi nodi della rete o dal linguaggio di programmazione con cui siano stati sviluppati.

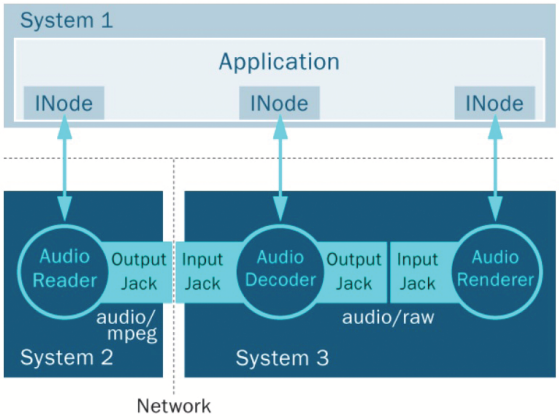

Figura 3.5: *Esempio di un flow graph distribuito con NMM.*

Citiamo ora alcuni dei benefici derivanti da un flow graph distribuito:

- Accesso a file collocati sulla rete senza la necessità di un file system distribuito (NFS: Network File System).
- Ogni connessione e ogni stream di dati tra i node è controllato da NMM e le sue interface.
- L'applicazione agisce su ogni node distribuito con comandi remoti, rendendo uniforme la comunicazione.

Concludiamo la descrizione della struttura generale di NMM con particolare attenzione alla sincronizzazione distribuita, infatti ne è prevista una in grado di gestire ogni tipo di comunicazione. Con la figura 3.6 vediamo che con NMM la gestione dei flussi audio e video può essere separata, permettendo l'ascolto sincronizzato su di un node, del video visto su di un node differente. Per ottenere questa funzionalità è necessario il settaggio dell'NTP (Network Time Protocol) in modo tale che un host faccia da server e tutti gli altri da client. Un host server si sincronizzerà con un clock esterno, mentre agli host client basterà sincronizzarsi con esso. Nel sito [20] è disponibile l'intera procedura da seguire per ottenere quando descritto in maniera distinta per Linux, Windows e Mac OS.

Figura 3.6: *Divisione e sincronizzazione del flusso audio-video tramite NMM.*

3.3 Registry Service

Il registry service di NMM è quell'insieme di classi e metodi che permettono la scoperta, la prenotazione e l'istanziazione dei node disponibili negli host locali e remoti. In ogni host un unico registry server è in grado di amministrare tutti i node disponibili tenendo traccia di una descrizione completa di ogni node (se ad esempio è di tipo sink), dei loro nomi, delle interface fornite e dei format di input e output supportati.

Una qualsiasi applicazione usa un registry client per inviare le richieste ai registry server che si possono trovare in esecuzione sia in locale che in remoto. Le richieste vengono inviate su di una particolare porta (normalmente la numero 22801) e quando il server le riceve, prenota lo specifico node. Se le richieste sono locali NMM prevede l'ottimizzazione dei tempi utilizzando un indirizzamento locale e non remoto. Per la creazione e configurazione di flow graph distribuiti complessi, un'applicazione può richiedere ogni node separatamente oppure ricercare le risorse disponibili utilizzando le descrizioni dei node che contengono le informazioni sui collegamenti.

Ogni host per usufruire delle risorse di rete e rendere disponibili le proprie deve lanciare un'applicazione chiamata serverregistry. La sua esecuzione non è invece necessaria nel caso in cui vi sia un host che operi in locale, infatti NMM rende comunque disponibile un registry server locale per il completo utilizzo delle risorse.

Ogni serverregistry necessita di una fase di inizializzazione in cui vengono ricercati i dispositivi e i software disponibili sulla rete.

Nello specifico vediamo ora il codice Linux per ottenere quest'inizializzazione:

```
user@linux:~/nmm/bin> ./serverregistry -s


Create config file with plugin information ...
Loading plugins...
AC3DecodeNode available
AVDemuxNode available
AVIReadNode available
... and many further nodes


Config file successfully written.
```

Vediamo come l'eseguibile *serverregistry* con l'opzione "-s" permetta la ricerca dei node disponibili, ottenendo un file riassuntivo che riporta il risultato ottenuto. In seguito a questa prima fase può essere lanciato l'applicativo *serverregistry* senza opzioni che si metterà in ascolto, in modo tale da rendere effettivamente disponibili le risorse dell'host sulla rete e ricevere quelle remote.

Capitolo 4

Installazione di NMM

Questo capitolo sarà strutturato come una piccola guida "HowTO" per cercare di rendere semplice ed agevole l'installazione e la configurazione dell'ambiente per lavorare con NMM.

Una volta installato il middleware, verrà anche spiegato come installare l'SDK utile per la creazione del proprio node.

Tutta la procedura si riferisce in ambiente Linux e precisamente utilizzando Ubuntu 10.04 come distribuzione. Inoltre non è possibile lavorare con virtual machine in quanto l'emulazione delle periferiche hardware come la scheda video e quella audio creano problemi per la gestione del middleware.

4.1 NMM

4.1.1 Prerequisiti hardware

Per l'installazione in ambiente Linux è necessaria la presenza di una scheda video con l'estensione XV (X-Video) configurata correttamente. Quest'estensione indica un meccanismo di video output per il sistema X Window, il quale è un protocollo di visualizzazione a finestre utilizzato soprattutto nei sistemi Unix-like come il nostro. Per sapere se dobbiamo effettuare delle modifiche lanciamo il comando

```
xvinfo
```

ed osserviamo il suo output. Questo comando stampa a video le informazioni relative alla situazione della scheda video rispetto all'estensione X-Video. Se compaiono i settaggi della scheda allora non sono necessarie modifiche, altrimenti restituirà una riga con un errore. In questo caso dovremmo verificare la corretta installazione del package xvinfo relativo alla distribuzione Linux utilizzata. In genere però questo settaggio non crea problemi.

NMM necessita poi di una scheda o un chip audio correttamente funzionante e in grado di riprodurre suoni a diverse frequenze di campionamento (ad esempio 44,1 KHz). Allo stato attuale ogni personal computer possiede una scheda o un chip audio con integrata questa funzionalità. Resta il fatto però che il prerequisito implica il corretto riconoscimento da parte del sistema operativo del componente hardware.

4.1.2 Configurazione della rete

Vediamo ora come dev'essere configurata la rete per un corretto funzionamento di NMM. Prima di tutto le porte 22801 e quelle dell'intervallo 5000-6000 devono essere aperte, cioé non devono essere bloccate da firewall o altri sistemi di protezione. È poi necessario settare il file /etc/hosts per la risoluzione dei nomi. In esso dobbiamo inserire il dominio di appartenenza, il nome dell'host e infine, se non è presente un DNS (Domain Name System)[1], dobbiamo inserire anche l'indirizzo IP perchè NMM non è in grado di riconoscerli direttamente.

4.1.3 Configurazione software

Prima di installare NMM è necessario preparare l'ambiente di lavoro. Nello specifico, lavorando con Ubuntu 10.04, molti dei pacchetti richiesti per il corretto funzionamento del middleware erano assenti, è stata quindi necessaria una preventiva installazione. Controllando sul sito del progetto sotto la voce librerie esterne

```
http://www.motama.com/nmmdocs_external-libraries.html
```

è presente una lista di tutte le librerie richieste.

Il sito propone 3 strade per l'installazione:

- scaricare l'archivio reso disponibile dalla Motama contenente le librerie precompilate e scompattarle in una cartella. Queste librerie non necessariamente coprono tutte le esigenze ed inoltre non è assicurato il corretto funzionamento in qualsiasi sistema. Il metodo è semplice, ma a volte può creare dei problemi rendendo quindi necessario l'utilizzo di una delle altre due possibilità.
- Scaricare, compilare ed installare le librerie utilizzando i codici sorgente.
- Installare i package della particolare distribuzione Linux utilizzata.

Viene raccomandato l'utilizzo della prima soluzione, che prevede il semplice download di un pacchetto già precompilato di librerie (nmm-2.2.0-optional-external-libs-linux.tar.gz) e la sua installazione seguendo queste istruzioni:

[1] Per la traduzione dei nomi degli host in indirizzi IP e viceversa

```
cd /home/alberto/

tar xvfz nmm-2.2.0-optional-external-libs-linux.tar.gz
```

Nonostante il consiglio, questa alternativa non ha dato i risultati sperati, infatti dopo l'installazione di NMM molti nodi non risultavano disponibili.

Seguendo la seconda alternativa invece, la configurazione ha avuto buon esito, vediamola più nel dettaglio.

Le librerie richieste per il buon funzionamento sono le seguenti:

a52dec	libpng
faad	asoundlib
ffmpeg	Xlib
l1394	ibjpeg
libmp3lame	ImageMagick
libraw1394	mplayer
libmad	vlc
libdvdnav	transcode
libdvdread	ogmtools
libogg	libxml++
libvorbis	libx264
libshout	DVB API 5.1
fftw	ulxmlrpcpp
libliveMedia	openssl
mpeg2dec	expat
cdparanoia	

Ci sono librerie per la gestione dell'audio come per esempio "asoundlib", per la gestione del video come 'Xlib" e per l'interazione con varie periferiche. Ovviamente per un'installazione veramente completa andrebbero installate tutte, ma utilizzando un po' di criterio, se non è previsto l'utilizzo di videocamere firewire per esempio, le librerie "1394" si possono omettere. In appendice vengono riportate tutte le informazioni e le descrizioni delle librerie.
Una volta individuate le librerie l'installazione è stata fatta semplicemente utilizzando il Gestore di Pacchetti Synaptic di Ubuntu, facendo attenzione ad installare dalla lista disponibile i pacchetti **-dev* presenti. Alcune librerie, comunque non sono disponibili tramite Synaptic e quindi è richiesta l'installazione manuale. Solitamente un semplice

```
./configure

make

sudo make install
```

è sufficiente per l'installazione, in caso contrario si rimanda al sito del produttore per le istruzioni.

Una volta preparato l'ambiente si può procedere con l'installazione vera e propria di NMM. Dal sito del Motama bisogna scaricare il file *nmm-2.2.0.tar.gz* (ovviamente la versione più aggiornata disponibile), scompattarlo in una cartella per esempio in

```
/home/alberto/nmm
```

Da qui poi dare i comandi in successione

```
alberto@xps:~/nmm\$ ./configure

alberto@xps:~/nmm\$ make

alberto@xps:~/nmm\$ sudo make install
```

Il *./configure* permette di vedere quali sono le funzioni base di NMM abilitate. Inizialmente dovrebbero esserci delle voci segnate *disabled*, ma dopo l'installazione dei pacchetti esterni fatta in precedenza le voci interessate dovrebbero essere segnate tutte *enabled*.

Dopo l'esecuzione dei seguenti passi, se non si sono riscontrati errori NMM dovrebbe essere installato correttamente.

Come prima verifica si può provare il comando

```
serverregistry -s
```

che restituisce la lista dei nodi disponibili. Viene riportata di seguito il risultato del comando.

```
serverregistry and Network-Integrated
Multimedia Middleware (NMM) Version 2.2.0

Copyright (C) 2005-2010
Motama GmbH, Saarbruecken, Germany
http://www.motama.com
```

```
See licence for terms and conditions of usage

No plugin information available! If you start NMM for
the first time this information is created automatically.
Note: If you ever change your hardware configuration or
update the NMM version you must delete the file
/home/alberto/.nmm/plugins.2.2.0.xps._usr_local
or run 'serverregistry -s' again.

Create config file with plugin information ...
Loading plugins...
AC3DecodeNode                                   available
AC3ParseNode                                    available
ALSAPlaybackNode                                available
ALSARecordNode                                  available
AVDemuxNode                                     available
AVMuxNode                                       available
AnalyseDataIdNode                               available
AudioMuxNode                                    available
BrightnessNode                                  available
BufferDropNode                                  available
BufferShapingNode                               available
BufferTimestampControlNode                      available
BufferTimestampNode                             available
CopyNode                                        available
DVBS2ReadNode                                   not available
DVBSimulatorNode                                available
DevNullNode                                     available
DummyAudioSinkNode                              available
DummyVideoSinkNode                              available
FFMPEGDeinterlaceNode                           available
FFMpegAVIReadNode                               available
FFMpegAVIWriteNode                              available
FFMpegAudioDecodeNode                           available
FFMpegAudioEncodeNode                           not available
FFMpegDecodeNode                                available
FFMpegEncodeNode                                available
FLACReadNode                                    available
FramerateConverterNode                          available
GenericReadNode                                 available
GenericWriteNode                                available
H264DecodeNode                                  available
IVTVReadNode                                    not available
IcecastNode                                     available
```

```
IdNode                                  available
JPEGDecodeNode                          available
JPEGEncodeNode                          available
LogoNode                                available
M4AReadNode                             available
MP3ReadNode                             available
MPEGAudioDecodeNode                     available
MPEGAudioEncodeNode                     available
MPEGDemuxNode                           available
MPEGReadNode                            available
MPEGTSDemuxNode                         available
MPEGTSReadNode                          available
MPEGTimeShiftingNode                    available
MPEGTimeShiftingNode2                   available
MPEGVSHDetectionNode                    available
MPEGVideoDecodeNode                     available
MagickManipulationNode                  available
MagickReadNode                          available
MagickWriteNode                         available
MessageSeekNode                         not available
NetSourceNode                           not available
OGMDemuxNode                            available
OSDManagerNode                          available
OggVorbisDecodeNode                     available
OverlayNode                             available
PCMDecodeNode                           available
PCMEncodeNode                           available
PNGReadNode                             available
PNGWriteNode                            available
PlaybackNode                            available
RGBtoRGBConverterNode                   available
RGBtoYV12ConverterNode                  available
RawNode                                 available
RecordNode                              available
ScopeNode                               available
TSDemuxNode                             available
TVCardReadNode                          available
TVCardReadNode2                         available
TimeDisplayNode                         available
TimedBufferDropNode                     available
URLNode                                 available
VISCACameraReadNode                     not available
VideoCropNode                           available
VideoGrabNode                           available
VideoMuxNode                            available
```

```
VideoScalerNode                                    available
VoiceDetectorNode                                  not available
WavReadNode                                        available
WavWriteNode                                       available
WhiteNoiseNode                                     available
XDisplayNode                                       available
YUVDeInterlaceNode                                 available
YUVtoRGBConverterNode                              available
YUVtoYUVConverterNode                              available
Finished loading plugins ...

Config file successfully written.
```

Questa lista può essere presa d'esempio per un primo controllo di buona installazione di NMM.

Un problema riscontrato dopo l'installazione, era l'assenza di certi nodi fondamentali come *l'ALSAPlaybackNode* o *l'XDisplayNode*, responsabili rispettivamente di gestire lo stream in uscita dell'audio e del video. Essendo i nodi di default in tutti gli esempi, la loro assenza non permetteva l'esecuzione di nessun esempio e quindi non era possibile testare veramente il sistema.

Il problema è stato risolto semplicemente disistallando NMM e reinstallandolo senza apportare nessuna modifica aggiuntiva. Quindi:

```
alberto@xps:~/nmm\$ sudo make uninstall

alberto@xps:~/nmm\$ ./configure

alberto@xps:~/nmm\$ make

alberto@xps:~/nmm\$ sudo make install
```

Questa soluzione è stata adottata su due diversi pc, uno con Ubuntu 10.04 (32 bit) e uno con Ubuntu 9.04 (64 bit) entrambi con lo stesso problema. Sicuramente questo punto meriterebbe un approfondimento per capire quale fosse realmente in problema, ma in mancanza di tempo e avendo ottenuto comunque NMM installato correttamente sul mio pc, ho tralasciato questo dettaglio.

A questo punto ricontrollando la lista dei nodi dovrebbero essere disponibili tutti quanti.

4.1.4 Testing del sistema

Alla fase di installazione segue una fase di testing nella quale verifichiamo il corretto funzionamento dei servizi di base di NMM. La fase di testing può essere effettuata anche su NMM semplicemente compilato, concentriamoci però su NMM compilato ed installato.

Per il primo test eseguiamo tramite *clic*[2] un semplice player audio. Entrando nella directory di NMM seguendo il path

```
nmm-2.2.0/apps/clic/gd/linux/playback/audio/
```

si trovano dei file *.gd* già pronti per l'uso. L'estensione .gd stà per "graph description". Questi semplici file testuali descrivono con una sintassi molto intuitiva il grafo contenente i nodi NMM che si vuole mandare in esecuzione.

Non ci resta che lanciarne uno, per esempio:

```
clic wavplay.gd -i <file audio.wav>
```

dove con *clic* eseguiamo il file *wavplay.gd* con in input un qualsiasi file audio .wav.
Quest'operazione dovrebbe consentirci di sentire il file.

Il secondo test prevede la stessa procedura del primo con la sola differenza che tentiamo di visualizzare un video.
Da

```
nmm-2.2.0/apps/clic/gd/linux/playback/video/
```

diamo

```
clic noise.gd
```

il quale dovrebbe mostrarci una finestra con del rumore video.

4.1.5 Configurazione del controllo per la sicurezza

In NMM è presente anche un metodo di accesso controllato per garantire la sicurezza dei dati trasmessi. Un intruso quindi non potrà entrare nella rete se non è stato precedentemente autorizzato. Questa fase non è strettamente necessaria al corretto funzionamento di NMM. Il file con la funzione di contenere i privilegi di rete si chiama nmmrc sample, si trova in una cartella dentro il file decompresso inizialmente e dev'essere copiato con il nome .nmmrc nella propria cartella home. In esso, settando *allowedreadpaths* identifichiamo quali plugin sono abilitati a

[2]Applicazione per la creazione e l'esecuzione di flow graph con NMM. Il suo utilizzo verrà spiegato meglio successivamente.

leggere dati, settando *allowedwritepaths* identifichiamo quali plugin sono abilitati a scrivere dati e settando *passwd* restringiamo le interazioni tra i processi di NMM solo a quelli con la stessa passwd.

Tutte queste attenzioni non sono logicamente necessarie se si opera in locale e si è protetti da firewall. Tutti i privilegi e le restrizioni di ogni utente Linux rimangono validi ed invariati.

4.1.6 Come utilizzare NMM

Per utilizzare le funzionalità messe a disposizione da NMM possiamo seguire due procedure. La prima e più semplice verrà illustrata in questa sezione e si basa sull'uso dell'eseguibile ./clic; la seconda consiste nella scrittura, compilazione ed esecuzione di codice C++ che utilizza librerie, namespace e interfacce di NMM.

Clic (Command Line Interaction and Configuration) fornisce uno strumento per l'utilizzo dei node già esistenti con la possibilità di configurare alcuni dei loro parametri. Partendo dalla conoscenza delle funzioni di ogni node, ./clic riesce a creare un flow graph da una sua descrizione testuale. Con ./clic diventa molto semplice scrivere ed eseguire un'applicazione che non necessita di interazioni con un utente, basta creare correttamente un file con estensione .gd (graph description) e farlo eseguire da ./clic.

Vediamo ora un esempio di file .gd che utilizza un node chiamato WavReadNode per prelevare un file .wav. Questo file viene preso nell'host e viene fatto ascoltare in locale grazie ad un node chiamato ALSA PlaybackNode:

```
% This graph description plays a WAV file using the
% ALSAPlaybackNode
% It can only be run on Linux.
% Command:
% clic wavplay.gd -i <local WAV file>
WavReadNode
! ALSAPlaybackNode
```

Le prime righe che cominciano per '%' sono commenti. Il simbolo '!' indica il collegamento tra due node. Una volta salvato il file come wavplay.gd, lo possiamo eseguire con:

```
clic wavplay.gd -i /home/alberto/Scrivania/test.wav
```

Come già visto ./clic prende wavplay.gd con in input (indicato da '-i') song.wav e semplicemente manda in esecuzione il grafo.
Ecco un secondo esempio più interessante che converte un file MPEG in un file AVI dividendo il flusso audio da quello video:

```
% This graph description converts an MPEG video file with AC3
% audio into an AVI file. Furthermore the desired bitrate of
% the video is specified using the connection format.
% Use the -i option of clic to specify the mpeg video file
% and -o option to specify the name of the output file.
GenericReadNode
! MPEGDemuxNode
{
  { ["mpeg_video0"] ! MPEGVideoDecodeNode
                    ! FFMpegEncodeNode
                      @ Format ("video/mpeg4", bitrate = 1200000)
                    !
    ["video"]
  }

  {
    ["ac3_audio0"] !
    ["audio"]
  }
} AVMuxNode
! AVIWriteNode
```

In questo caso si ha un generico node che legge un file dato in ingresso e che invia i dati ad un demultiplexer node; qui il flusso audio viene diviso da quello video. Quest'ultimo flusso è inviato ad un MPEGVideoDecodeNode attraverso un canale (jack tag) chiamato "mpeg video0"; esso viene poi manipolato da un FFMpegEncodeNode con il settaggio del format come indicato dopo "@"; infine il flusso variato arriva ad un AVMuxNode attraverso il canale chiamato "video". Quest'ultimo node riceve nel suo secondo jack di input il flusso dal canale "audio" che giunge a sua volta da quello chiamato "ac3 audio0"; quindi dall'MPEGDemuxNode all'AVMuxNode il suono non viene manipolato e arriva senza attraversare particolari node. L'ultimo passaggio globale è tra l'unico output dell'AVMuxNode e l'input dell'AVIWriteNode, il quale scrive un file AVI con nome e percorso inseriti da riga di comando.

È utile sapere inoltre che i nomi dei canali sono arbitrari e possono essere scelti a piacere; se non è assegnato nessun nome è come se si utilizzasse "default".I nomi dei node invece sono quelli specifici di NMM ed ognuno di essi richiamerà i metodi opportuni. Infine si può notare come le parentesi graffe nel codice delimitino i blocchi relativi a ogni node.

Come scritto nei commenti del codice, per l'utilizzo di questo file con ./clic è necessaria in ingresso l'opzione '-i' per indicare percorso e nome del file MPEG di input, e l'opzione '-o' per indicare percorso e nome del file AVI di output da salvare:

```
./clic mpeg_to_avi.gd -i /home/alberto/video/movie.mpeg
-o /home/alberto/video/movie.avi
```

Vediamo ora in maniera sintetica quali opzioni ./clic riconosce all'interno di un file .gd per ogni node:

- setLocation(string): l'elemento string dato in ingresso a quest'opzione individua l'host specifico dove è situato un particolare node (attenzione perchè nell'host deve essere in esecuzione serverregistry).
- setPort(int): l'intero int indica il numero della porta in cui serverregistry è in ascolto. Di default è usata la porta 22801.
- setSharingType(sharing-type): quest'opzione accetta in ingresso un tipo specifico di NMM chiamato "sharing-type" che permette di configurare se un node può essere o meno riusato da un'applicazione.
- Interface methods: permettono di richiamare i metodi definiti in NMM-IDL per uno specifico node con i relativi parametri, evitando il settaggio da riga di comando. Abbiamo così la possibilità di creare un flow graph completo direttamente nella graph description, ./clic dovrà solo invocare il file .gd specifico senza opzioni aggiuntive.
- setEventStrategy: definisce il particolare protocollo usato nella comunicazione. Di default è il TCP.

Le prime tre opzioni vanno inserite in un graph description appena dopo un nodo precedute dal simbolo '#'. Un esempio può essere:

```
WavReadNode # setLocation("host1")
# setSharingType(SHARED)
! ALSAPlaybackNode
```

La quarta opzione va inserita subito dopo le prime tre (se presenti) e preceduta dal simbolo '$'. Ognuna di esse però deve essere seguita dallo stato[3] in cui verrà utilizzata:

```
WavReadNode \$ setFilename("/home/alberto/audio/song.wav") INITIALIZED
! ALSAPlaybackNode \$ setDownstreamMaxSize(1,"default") CONSTRUCTED
```

[3]I tipi possibili sono: EXCLUSIVE, EXCLUSIVE THEN SHARED, SHARED, EXCLUSIVE OR SHARED (che è quello di default), SHARED OR EXCLUSIVE, EXCLUSIVE THEN SHARED OR SHARED, SHARED OR EXCLUSIVE THEN SHARED. Si rimanda a [20] per ulteriori chiarimenti.

Infine la quinta opzione va inserita per ultima preceduta dal simbolo @:

```
WavReadNode @ setEventStrategy("UDPStrategy")
! ALSAPlaybackNode
```

Passiamo ora ad un ulteriore utilizzo di ./clic associato al Graph Builder. Esso permette la creazione automatica di un flow graph direttamente da riga di comando con diverse tipologie di input. Tutti i formati supportati da NMM sono disponibili in questa modalità e per richiamarli la struttura del comando è:

```
./clic <url>?<option>=<value>
```

Un esempio specifico può essere:

```
./clic file://host1/home/alberto/mp3/song.mp3
```

In questo caso specifico, ./clic permette l'ascolto in locale del file .mp3 presente nell'host1 nel percorso indicato. Con l'url che comincia per 'file' possiamo richiamare i tipi: .wav, .ogg, .mp3, .ac3,.mpeg, .avi, .ogm, .vdr, .png e .jpg, sempre che i node necessari per ognuno di essi siano resi disponibili in fase di installazione. 'file' è solo una della varie tipologie di sorgente, infatti sono disponibili anche 'audiocd', 'dvd', 'tv', 'dvbtv', 'ivtv', 'mpegtv', 'http'. Un'ulteriore funzionalità per il Graph Builder messa a disposizione da ./clic è la distribuzione dei flussi audio e video:

```
./clic <url>?<option>=<value> -A <host for audio output>
-V <host for video output>
```

Inserendo qui i nomi degli host specifici dalla postazione in cui ci troviamo riusciamo a prendere un file o uno stream anche in remoto ed inviare sulla rete i flussi audio e video divisi, ovunque vogliamo. ./clic penserà a tutto, compresa la sincronizzazione.

4.2 SDK (Software Development Kit)

Motama mette a disposizione un kit chiamato NMM-SDK o NMM Software Development Kit per lo sviluppo di software propri. In realtà questo kit consiste in una semplice interfaccia in NMM-IDL, un plugin che implementa questa interfaccia e un flow graph che utilizza questi due

componenti, fornisce cioè un esempio su come uno sviluppatore dovrebbe fare per creare i propri plugins riconoscibili dal registry service (cioè visibili in un flow graph) e di come utilizzarli.

Procediamo con l'installazione di NMM-SDK. Il prerequisito abbastanza logico da tenere presente prima di cominciare è la necessità di avere NMM installato correttamente. Iniziamo allora con lo scaricamento dal web del pacchetto opportuno (nmm-sdk-2.2.0) e con lo scompattarlo.

Dentro la cartella

```
...nmm-sdk-2.2.0/nmm/plugins/audio/filter
```

si trova un nodo d'esempio AudioVolumeFilterNode che testato con clic consente l'apertura di un wav e la regolazione del volume tramite shell. La via più semplice e consigliata anche sul sito di NMM per creare un proprio nodo è modificare questo o comunque prendere spunto da esso.

Guardando i .cpp risulta di facile comprensione la sintassi NMM, il primo problema è stato riuscire ad includere in fase di compilazione i nostri file esterni (.cpp e .h) al file AudioVolumeFilter.cpp.

Per farlo bisogna includere i file desiderati al make.am

```
libnmmFallDetectionNode_la_SOURCES = \
    FallDetectionNode.cpp \
    cameraacquisition.cpp \
    frame_grabber_device.cpp \
    seqreader.cpp \
    video_capture_device.cpp \
    actualmoment.cpp \
    facedetector.cpp \
    falldetector.cpp \
    imageacquisition.cpp \
    main.cpp \
    motdetapplication.cpp \
    motiondetector.cpp \
    cameramotion.cpp \
    camerapositioning.cpp \
    trackedcontour.cpp \
    unwrapdata.cpp

treepkginclude_HEADERS = \
    (IDLHEADERS) \
    FallDetectionNode.hpp \
    visibility.hpp \
    cameraacquisition.h \
    frame_grabber_device.h \
    seqreader.h \
```

```
    video_capture_device.h \
    actualmoment.h \
    facedetector.h \
    falldetector.h \
    imageacquisition.h \
    motdetapplication.h \
    motiondetector.h \
    cameramotion.h \
    camerapositioning.h \
    trackedcontour.h \
    unwrapdata.h
```

Una volta modificato il make.am dalla directory dell'sdk bisogna utilizzare delle funzioni di autotools[4] per generare il make.in e successivamente il makefile

```
chmod +x autogen.sh && ./autogen.sh
```

a questo punto abbiamo tutto pronto, basta dare un configure, ricordandosi che ovviamente l'sdk richiede NMM, quindi bisogna passarlo in questo modo

```
./configure --with-nmm=/usr/local
```

ora abbiamo i makefile generati, basta dare dalla directory principale dell'sdk

```
make
sudo make install
```

e se tutto si conclude senza errori, dando

```
serverregistry -s
```

dovremmo vedere il nome del nostro nodo, per esempio

```
FallDetectionNode        available
```

[4]Per chi non conoscesse autotools o comunque ne voglia sapere di più, consiglio questa guida online http://sourceware.org/autobook/autobook/autobook_toc.html

Si ricorda che nel caso si utilizzino librerie esterne è necessario aggiungerle al make.am. Per esempio nel caso della OpenCV le righe da aggiungere sono le seguenti:

```
INCLUDES = \
    $(all_includes) \
    -I/usr/local/include/opencv $(all_includes) -I/usr/include/opencv
LIBS =  -lhighgui -lcxcore -lcv
```

Capitolo 5

NMM e Fall Detection

In questo capitolo vedremo nel dettaglio la realizzazione dei nodi NMM implementati per la rilevazione delle cadute.

Ovviamente il nodo "FallDetectionNode", non è altro che l'integrazione del software già sviluppato, ma verrà descritta tutta la parte relativa ad NMM.

5.1 DedistNode

Per essere in grado di collegare fra di loro dei nodi all'interno di un grafo, i loro formati di input e di output devono corrispondere.

Come illustrato nel capitolo dedicato a NMM, esistono varie classi standard di nodo.

Se il nodo realizzato è una sottoclasse di *GenericSourceNode, GenericFilterNode, GenericConverterNode, GenericProcessorNode, GenericDemultiplexerNode, GenericMultiplexer o GenericSinkNode* (in breve una sottoclasse di un nodo "generico"), definire i formati di input e output non risulta molto complesso.

Principalmente bisogna concentrarsi su due punti. In primo luogo, bisogna precisare i formati supportati dal nodo (ovvero quelli che il nodo può ricevere come input) alla voce *doInit()*. In secondo luogo, si deve definire i formati di output per il nodo con *doInitOutput()*.

Nel caso del nodo sviluppato, dato che deve ricevere uno stream video da una videocamera omnidirezionale con una risoluzione di 640 x 480 e deve inoltrare ad eventuali altri nodi del grafo NMM immagini dedistorte di risoluzione 1280 x 156, è stato necessario estendere il *GenericProcessorNode*, per avere la possibilità di variare il formato di output rispetto quello di input.

Questo primo nodo, quindi, si occuperà della trasformazione da immagine omnidirezionale a cilindro panoramico. Questa funzione è stata inserita in un nodo specifico in quanto potrà tornare molto utile per future applicazioni riguardanti l'Omnidome; in questo modo, non ci si dovrà preoccupare di gestire l'immagine omnidirezionale, ma si potrà lavorare direttamente su quella dedistorta.

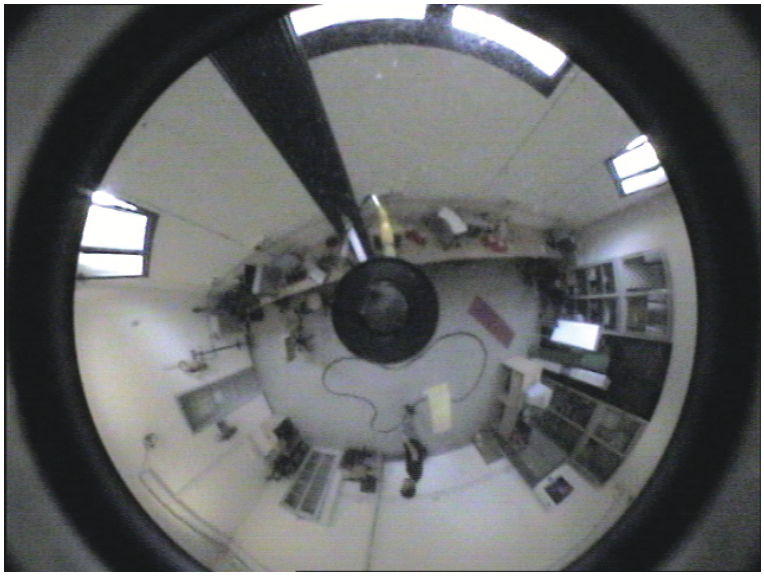

Figura 5.1: *Immagine omnidirezionale dello stream in input per il nodo.*

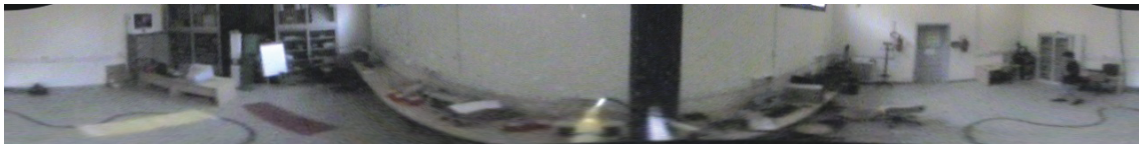

Figura 5.2: *Immagine dedistorta dello stream in output dal nodo.*

Vediamo ora come definire i formati.

```
Result DedistNode::doInit()
{
// register all supported output formats
NMM::Format *in_format;
in_format = getOwnInputProperty()->addNewFormat("video/raw");
in_format->addIntParamValue(Format::x_resolution_param, 640);
in_format->addIntParamValue(Format::y_resolution_param, 480);
in_format->addWildcard(Format::ratio_x_param);
in_format->addWildcard(Format::ratio_y_param);
in_format->addWildcard(Format::framerate_param);
in_format->addStringParamValue(Format::channelorder_param,"rgb");
in_format->addStringParamValue(Format::endian_param, "little");
in_format->addStringParamValue(Format::colorspace_param,"rgb24");
in_format->addIntParamValue(Format::bitperpixel_param,24);
in_format->addStringParamValue(Format::format_param,"packet");
getOwnInputProperty()->setDefaultFormat(in_format);

NMM::Format *out_format;
out_format = getOwnOutputProperty()->addNewFormat("video/raw");
out_format->addWildcard(Format::x_resolution_param);
out_format->addWildcard(Format::y_resolution_param);
out_format->addWildcard(Format::ratio_x_param);
out_format->addWildcard(Format::ratio_y_param);
out_format->addWildcard(Format::framerate_param);
out_format->addStringParamValue(Format::channelorder_param,"rgb");
out_format->addStringParamValue(Format::endian_param, "little");
out_format->addStringParamValue(Format::colorspace_param,"rgb24");
out_format->addIntParamValue(Format::bitperpixel_param,24);
out_format->addStringParamValue(Format::format_param,"packet");
getOwnOutputProperty()->setDefaultFormat(out_format);

in_format->setIOPartner(out_format);
out_format->setIOPartner(in_format);


return SUCCESS;


}
```

Creiamo due formati, uno di input “in_format” e uno di output “out_format”. Ovviamente inizializziamo uno stream video per entrambi.

Per tutti i parametri del formato è possibile aggiungere una “wildcard” o settare la tipologia richiesta. Per esempio, con riferimento al codice riportato sopra, sono state inserite delle wildcard per il framerate, questo vuol dire che il mio nodo accetterà un qualsiasi framerate sia in input che in output. Diversamente accade invece per la risoluzione, dove sono stati settati i parametri richiesti per il nodo, dato che l’immagine omnidirezionale deve essere dedistorta le

sue dimensioni sono molto importanti e quindi non posso accettare una qualsiasi risoluzione in input.

All'interno di un nodo, è possibile definire vari formati. L'input e l'output poi verranno associati con le due istruzioni finali:

```
in_format->setIOPartner(out_format);
out_format->setIOPartner(in_format);
```

In questo modo diciamo al nodo che in caso di di ricezione di uno stream con parametri compatibili all'"in_format" dovrà settare come formato d'uscita "out_format".
Vediamo ora l'inizializzazione dello stream di output. A differenza del *doInit* nel *doInitiOutput* lo stream di input è già stato inizializzato ed è quindi possibile acquisire delle informazioni su di esso inizialmente ignote ed accettate comunque tramite wildcard. Se per esempio dobbiamo fissare la risoluzione in uscita in funzione di quella in entrata, ora abbiamo la possibilità di modificare e di inizializzare lo stream di output con i parametri desiderati.

```
Result DedistNode::doInitOutput()
{

// get input format
const Format* in_format = getInputFormat();
if (in_format == NULL) {
ERROR_STREAM("No input format" << endl);
return FAILURE;
}

// get format parameters
width = in_format->getIntValue(Format::x_resolution_param);
height = in_format->getIntValue(Format::y_resolution_param);
if ((width <= 0) || (height <= 0))
{
return FAILURE;
}

bitperpixel = in_format->getIntValue(Format::bitperpixel_param);
framerate = in_format->getFloatValue(Format::framerate_param);
channelorder = in_format->getStringValue(Format::channelorder_param);
colorspace = in_format->getStringValue(Format::colorspace_param);
format = in_format->getStringValue(Format::format_param);
endian = in_format->getStringValue(Format::endian_param);
// cout << "INFORMAZIONI SUL FORMATO\n"<<"bitperpixel "
// << bitperpixel <<endl<<"channelorder "<<channelorder<<endl
// << "colorspace " << colorspace<< endl<<"format "<<format
// <<endl<<"framerate "<<framerate<<endl;

// set output format
// this is equal to input format for all filter nodes
Format* o_format = getOwnWorkingOutputProperty()->addNewFormat("video/raw");
o_format->addIntParamValue(Format::x_resolution_param, 1280);
```

```
o_format->addIntParamValue(Format::y_resolution_param, 156);
o_format->addIntParamValue(Format::ratio_x_param, 1280);
o_format->addIntParamValue(Format::ratio_y_param, 156);
o_format->addFloatParamValue(Format::framerate_param, framerate);
o_format->addStringParamValue(Format::channelorder_param, channelorder);
o_format->addStringParamValue(Format::colorspace_param, colorspace);
o_format->addStringParamValue(Format::format_param, format);
o_format->addStringParamValue(Format::endian_param, endian);
o_format->addIntParamValue(Format::bitperpixel_param, bitperpixel);
o_format->addFloatParamValue(Format::framerate_param, framerate);

getOwnWorkingOutputProperty()->addNewFormat(o_format);
getOwnWorkingOutputProperty()->setDefaultFormat(o_format);

return SUCCESS;
}
```

In questo caso, inizialmente mi faccio restituire tutti i parametri dell'input stream. Questo è necessario se si erano definite delle wildcard, dato che, settando la possibilità di accettare qualsiasi parametro in input, non sappiamo a priori cosa riceveremo. Nel mio caso è stato utilizzato come verifica di correttezza nella lettura del formato iniziale.

Successivamente inizializziamo lo stream di output settando gli stessi parametri di quello di input tranne la risoluzione, dove mettiamo 1280x156.

A questo punto abbiamo definito gli stream di input e di output del nostro nodo e possiamo concentrarci sulla computazione. Ovviamente un nodo NMM richiede anche altre funzioni come *"doDeinitOutput"* ecc..., ma queste sono molto semplici da interpretare e capire guardando l'SDK fornita da Motama.

5.2 FallDetectionNode

Questo nodo si occupa del tracking e della rilevazione delle cadute. E' definito come un *GenericFilterNode* dato che riceve uno stream già dedistorto e non deve eseguire modifiche su di esso.

I due metodi "doInit()" e "doInitOutput()" sono molto simili a quelli definiti nel DetectionNode, salvo piccole modifiche sui parametri (per esempio la risoluzione avrà una wildcard).

Il cuore del nodo è il metodo "processBuffer". All'interno di questa funzione vanno inserite tutte le operazioni che il nodo deve eseguire ad ogni frame nel caso di un nodo video. Il problema principale è stato ottenere un'immagine OpenCV dal buffer in input.

Riporto ora le parti salienti di codice per far capire come strutturare il processBuffer.

```
Buffer* FallDetectionNode::processBuffer(Buffer *in_buffer)
{
     if (!inputImg)
     inputImg = cvCreateImage( cvSize(width, height), IPL_DEPTH_8U, 3  );
```

```
    if(!outputImg)
    outputImg = cvCreateImage( cvSize(1280, 156), IPL_DEPTH_8U, 3  );

   //acquisisco il frame e lo metto in un'immagine OpenCV
   inputImg->imageData = in_buffer->getData();

   //QUI VANNO INSERITE LE ISTRUZIONI PER
   //PROCESSARE L'IMMAGINE

   //creazione del buffer di output
   out_buffer_size = outputImg->imageSize;
   Buffer* out_buffer = getNewBuffer( out_buffer_size );
   out_buffer->setUsed(out_buffer_size);
   out_buffer->setData( outputImg->imageSize, outputImg->imageData );

   // rilascio del buffer
   in_buffer->release();

   return out_buffer;
 } //buffer
```

InputImg e outputImg sono delle IplImage OpenCV.

Usando il *"getData()"* è possibile ottenere il frame contenuto nel buffer di input e associarlo quindi all'immagine OpenCV. A questo punto è possibile eseguire una qualsiasi operazione di processing. Una volta ottenuta l'immagine finale, ovvero quella da far restituire dal nodo, basta istanziare un buffer di output e restituirlo.

Arrivati a questo punto abbiamo un nodo NMM pronto per essere installato ed eseguito in un grafo. Per l'installazione si rimanda al capitolo precedente sull'installazione dell'SDK. Per l'esecuzione, come spiegato nel capitolo di NMM, bisogna creare un file .gd che lanci il nostro nodo.

Il file utilizzato per la mia applicazione è il seguente:

```
TVReadCardNode2
! YUVtoRGBConverterNode
! DedistNode
! FallDetectionNode
! RGBtoYV12ConverterNode
! XDisplayNode
```

Il primo nodo, il "TVReadCardNode2", si occupa dell'acquisizione dell'immagine da framegrabber. Questo restituisce uno stream video codificato in yuv, mentre il "DedistNode" vuole in ingresso uno stream rgb. La conversione dal primo al secondo formato, è affidata al "YUVtoRGBConverterNode".

A questo punto abbiamo in cascata i miei due nodi e successivamente "RGBtoYV12 ConverterNode", infatti dobbiamo riportare lo stream in yuv prima di poterlo inoltrare all'"XDisplayNode" che visualizzerà a schermo l'immagine.

5.3 VideoEventReceiverNode

Come visto nel capitolo di presentazione dell'ambiente di lavoro, l'Omnidome è dotato oltre che di una videocamera omnidirezionale, di una videocamera prospettica motorizzata.

Una volta rilevata la caduta sarebbe quindi utile che il "FallDetectionNode" gestisse anche il movimento del dome in modo da inquadrare ad alta risoluzione la persona caduta.

Dato che stiamo lavorando con un middleware, la soluzione più corretta sarebbe realizzare un nodo che si occupi del movimento della camera prospettica. Il "FallDetectionNode" dovrebbe quindi riuscire a comunicare con questo nodo per fornirgli la posizione del soggetto da inquadrare.

Per realizzare la comunicazione fra nodi, sono stati utilizzati gli eventi forniti da NMM, il "FallDetectionNode" dovrà quindi creare un evento quando rileva una caduta e lanciarlo nella rete dove ci sarà poi un "listener" che catturerà l'evento e si occuperà di muovere la videocamera.

Vediamo come prima cosa la creazione dell'evento.

All'interno del "processBuffer" del "FallDetectionNode", basta inserire le seguenti righe:

```
if(fall.existsTarget() && targetFall.front().sendEvent){
//caduta rilevata

fall.getTargetPosition(&hAngle, &vAngle);
TInValue<int>* id = new TInValue<int>((targetFall.front().ID));
TInValue<float>* angleH = new TInValue<float>(hAngle);
TInValue<float>* angleW = new TInValue<float>(vAngle);
TInValue<Time>* eTimestamp = new TInValue<Time>(clock.getTime());

// creo  downstream Event
Event* e = new Event("FALL_DETECTED_EVENT", 4);
e->setValue(0,id);
e->setValue(1,angleH);
e->setValue(2,angleW);
e->setValue(3, eTimestamp);
CEvent* ce = new CEvent();
ce->insertEvent(e);
sendEventDownstr(ce);

}
```

Come si vede creo quattro parametri che saranno gli argomenti dell'evento e poi semplicemente creo l'evento (chiamato "FALL_DETECTED_EVENT"), setto i parametri e lo lancio. Il send può essere fatto "downStream" o "upStream", ovvero l'evento può essere lanciato dal nodo corrente verso la fine del grafo o verso l'inizio.

Ora bisogna realizzare il nodo che attenda l'evento. Questo nodo sarà un "filterNode" dato che non dovrà processare lo stream, ma dovrà semplicemente inoltrarlo al nodo successivo del grafo e percepire l'evento.

Vediamo questo nodo un po' nel dettaglio.

Nel costruttore del nodo bisogna specificare il tipo di evento che il listener deve aspettare per attivarsi:

```
VideoEventReceiverNode::VideoEventReceiverNode(const char* name )
: GenericFilterNode(name){

  //registrazione evento in ascolto
  registerEventListener("FALL_DETECTED_EVENT",
      new TEDObject4<EventListener,
      int, float, float, Time,
      Result(EventListener:: * )(const int, const float,
      const float, const Time),Result>
  (&m_event_listener, &EventListener::fallDetected));

}
```

Il mio nodo dovrà aspettarsi un evento con quattro parametri, un int, due float e un Time. Alla ricezione dell'evento dovrà chiamare la funzione "fallDetected" della classe "EventListener". Questa classe è implementata sullo stesso file del "VideoEventReceiverNode" e contiene le istruzioni da eseguire in caso di rilevazione di evento, nel mio caso, dovrà muovere il dome.

```
  Result EventListener::fallDetected(const int id,
   const float angleH, const float angleW)
  {

  //QUI SONO STATE INSERITE LE ISTUZIONI PER GESTIRE
  // I MOVIMENTI DEL DOME.

   return SUCCESS;
  }
```

Tutte le altre funzioni come "processBuffer" e "doInit", dovranno semplicemente restituire lo stream iniziale nel caso del processBuffer e inizializzare un formato di ingresso e uno di uscita con wildcard nel caso del doInit.

Ci si deve inoltre ricordare di inserire il nodo appena realizzato nel grafo.

```
TVReadCardNode2
! YUVtoRGBConverterNode
! DedistNode
! FallDetectionNode
! RGBtoYV12ConverterNode
! VideoEventReceiverNode
! XDisplayNode
```

Riassumiamo in breve l'intero grafo con un'immagine:

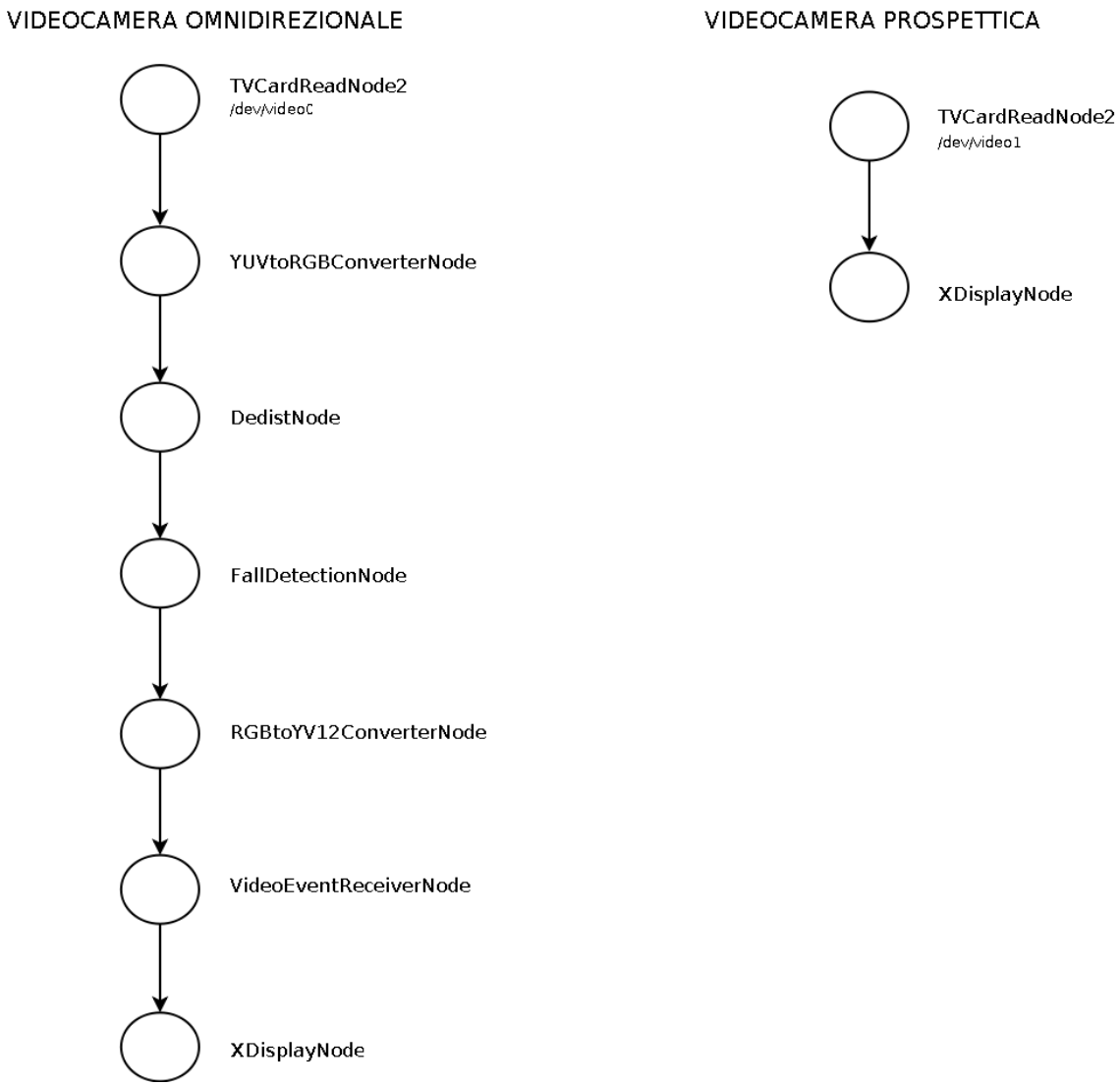

Figura 5.3: *Grafo NMM per l'applicazione sviluppata.*

e riportiamo il file clic finale facendo qualche precisazione.

```
TVReadCardNode2 $setDevice(``/dev/video0'') INITIALIZED
! YUVtoRGBConverterNode
! DedistNode
! FallDetectionNode
! RGBtoYV12ConverterNode
! VideoEventReceiverNode
! XDisplayNode $setPosition(0,0) INITIALIZED

TVReadCardNode2 $setDevice(``/dev/video1'') INITIALIZED
! XDisplayNode $setPosition(0,540) INITIALIZED
```

Il pc dove viene eseguita l'applicazione dovendo acquisire da due videocamere è dotato di due framegrabber. Il metodo "setDevice" di TVCardReadNode2 consente appunto di selezionare quale device si intende utilizzare per quel grafo. Come spiegato nei capitoli di NMM, se si utilizzano metodi di interfaccia è necessario definire anche lo stato del nodo (in questo caso "INITIALIZED").

Per rendere più gradevole la visualizzazione su schermo, è anche stato utilizzato il metodo "setPosition" di XDisplayNode, in modo da sistemare le finestre video sullo schermo.

Per concludere, faccio notare che con un unico file di clic, è possibile avviare più grafi contemporaneamente.

5.4 Problemi con NMM

Di seguito vengono riportati i principali problemi riscontrati durante l'utilizzo di NMM.

Come molti progetti open-source, la documentazione non è molto approfondita e spesso tralascia dettagli molto importanti sull'utilizzo del middleware e sulla realizzazione dei nodi. Il forum di supporto fortunatamente è abbastanza frequentato dai programmatori di Motama e solitamente si riescono ad avere delle risposte in tempi relativamente brevi, anche se non sono mai molto approfondite, perché in caso di problemi "seri", non risolvibili con un post breve, si viene rimandati all'assistenza a pagamento.

Superato lo scoglio della documentazione guardando soprattutto i nodi già realizzati, ci si ritrova di fronte ad un problema veramente rilevante di NMM: molte funzioni sono solo dichiarate, ma non implementate. Per fare un breve esempio, il TVCardReadNode2 dispone di un metodo *setResolution* tramite il quale si dovrebbe poter impostare la risoluzione di acquisizione da videocamera. Dopo diversi tentativi falliti, controllando il codice ho riscontrato la mancanza di implementazione di alcune funzioni. Questo problema si presentava in particolar modo nei metodi di interfaccia dei nodi richiamabili da clic.

Concludo riportando un problema riscontrato nell'utilizzo delle OpenCV internamente ad un nodo NMM.

Tale problema si verificava quando all'interno del *processBuffer* veniva istanziata una IplImage (immagine OpenCV). Nel distruttore del nodo dato che veniva allocata della memoria,

erano inseriti dei *cvReleaseImage* necessari al rilascio della memoria occupata dalle immagini, questo però creava dei problemi in fase di terminazione, infatti premendo “q” (eseguendo un grafo con clic vengono stampati a video dei comandi per gestire l’esecuzione del grafo, per esempio “q” per terminare, “p” per mettere in pausa, “s” per riprendere, ecc...) il programma terminava segnalando degli errori sulla gestione della memoria non eseguendo il distruttore.

Contattando gli sviluppatori tramite il forum, sono riuscito a risolvere il problema: è bastato omettere il release delle immagini, infatti NMM nella terminazione si occupa automaticamente della deallocazione della memoria, per questo motivo all’esecuzione del release manuale NMM lanciava un errore.

Capitolo 6

Conclusioni

La caduta degli anziani rappresenta un problema di salute pubblica in quanto possibile causa di fratture invalidanti e di conseguenze psicologiche che riducono l'indipendenza della persona. Il problema però non è limitato soltanto agli anziani, si pensi per esempio a luoghi isolati come un parcheggio coperto di notte o un magazzino: in caso di caduta o svenimento per un malessere, il soggetto potrebbe rimanere senza soccorsi per un periodo molto lungo dato che il luogo non è molto frequentato.

La rilevazione della caduta diventa quindi un interessante problema scientifico essendo un processo anomalo che si può identificare utilizzando, per esempio, i vari approcci descritti in questa tesi nell'introduzione. La presenza di un sistema di identificazione di cadute consentirebbe un pronto intervento in caso di incidenti.

Il Laboratorio di Robotica Autonoma (IAS-LAB) dell'Università degli Studi di Padova si sta occupando della creazione di un sistema di videosorveglianza distribuito e ha deciso di integrare, fra le varie funzionalità, un fall detector.

Questo lavoro di tesi si è focalizzato principalmente su due punti:

- Realizzazione di un fall detector per la rilevazione di cadute tramite image processing;
- Integrazione del fall detector sviluppato in un nodo NMM, un middleware per la gestione di stream audio e video in una rete distribuita.

Per la realizzazione del progetto si è utilizzato un sensore video brevettato dall'IT+Robotics, denominato Omnidome, che si compone di una videocamera omnidirezionale e di un dome. Grazie alla prima è possibile controllare un ambiente molto vasto mentre con la seconda si possono inquadrare ad alta risoluzione le zone interessate in base all'applicazione.

Il fall detector, tramite filtro di Sobel, valuta il rapporto fra gradiente orizzontale e gradiente verticale del soggetto trackato. In caso di caduta è stato rilevato sperimentalmente [18] che la somma del gradiente orizzontale fatta su ogni pixel del bounding box è superiore rispetto quella del gradiente verticale. Partendo da questa considerazione si è sviluppato l'intero sistema. Altri parametri rilevanti per la rilevazione di caduta sono il tempo d'immobilità, l'aspect ratio e la variazione di dimensioni del bounding box da un frame all'altro.

Per testare il sistema sono stati ricreati in laboratorio differenti scenari di caduta, cercando di riprodurre tutte le possibili configurazioni: frontale, sui fianchi, di schiena e anche varie situazioni di tracking ovvero scenari con una o più persone in movimento.
I risultati ottenuti sono visibili in tabella 2.1. La sensitivty raggiunge il 90% mentre la specificity l'80%, percentuale leggermente inferiore per il verificarsi di falsi positivi. I valori ottenuti sono comunque accettabili ed in linea con i lavori già esistenti.

Un esempio dell'esecuzione del programma è visibile nelle immagini successive: in figura 6.1 si può vedere l'intera scena ripresa dalla videocamera omnidirezionale col rilevamento della caduta mentre in figura 6.2 il soggetto è inquadrato automaticamente dal dome.

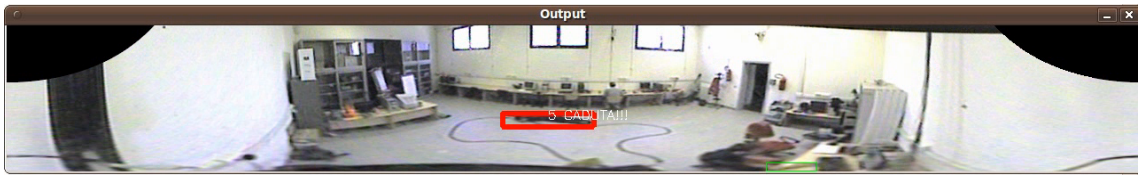

Figura 6.1: *Rilevazione della caduta da parte dell'Omnidome.*

Figura 6.2: *Visuale del dome.*

L'integrazione del progetto su un middleware come NMM ha permesso di semplificare la gestione di vari aspetti del lavoro: l'acquisizione da varie videocamere è stata effettuata con la semplice

istanza di un nodo TVCardReadNode2 e l'interazione fra vari nodi è stata molto agevolata dagli eventi NMM. La possibilità di poter creare singoli nodi adibiti ad una specifica funzione risulta molto comoda in quanto consente di realizzare applicazioni complesse facendo interagire fra di loro varie parti indipendenti.
Qualora si volesse approfondire ed integrare questo lavoro, sarebbe preferibile innanzitutto una revisione al sistema di tracking: l'approccio del background subtraction a seconda dell'applicazione potrebbe rivelarsi sconveniente, ad esempio in presenza di illuminazione che generi ombre oppure di oggetti rilevati come foreground che fermano il loro moto per un lungo periodo entrando a far parte del background.

Fra i vari algoritmi di tracking presenti in letteratura, quello ad istogrammi sembra molto promettente anche se necessita di un'inizializzazione nella quale va indicato manualmente il soggetto da trackare. Per sopperire a questo inconveniente, si potrebbe realizzare un algoritmo ibrido dove si delegano le prime fasi di inizializzazione ad un tracking basato su background subtraction, facendo subentrare in seguito quello ad istogrammi.

APPENDICE

Appendice A

Librerie per l'installazione di NMM

- *a52dec* - AC3 audio decoder.
- *faad* - Decoder for AAC.
- *ffmpeg* - Various audio and video codecs.
- *l1394* - Management of IEEE1394 devices.
- *libmp3lame* - MPEG layer3 audio encoder.
- *libraw1394* - Low level IEEE1394 support.
- *libmad* - MPEG layer3 audio decoder.
- *libdvdnav* - DVD reading and navigation.
- *libdvdread* - DVD reading.
- *libogg* - OGG file parser.
- *libvorbis* - OGG/Vorbis audio decoder.
- *libshout* - Injection of audio streams into a ShoutCast server.
- *fftw* - Fast Fourier Transformation, used for audio visualization.
- *libliveMedia* - RTP support.
- *mpeg2dec* - MPEG version 1/2 video decoder.
- *cdparanoia* - CDDA Support for Linux.
- *libpng* - PNG file reader.
- *asoundlib* - ALSA headers and library.

- *Xlib* - X11 headers and libraries.
- *libjpeg* - JPEG Support.
- *ImageMagick* - Imaging Support.
- *mplayer* - Parts of the implementation of DirectXDisplayNode is based on MPlayer.
- *vlc* - Parts of the implementation of DirectXDisplayNode is based on vlc.
- *transcode* - Parts of the C implementation of deinterlacing algorithm in YUVDeInterlaceNode.
- *ogmtools* - Parts of the implementation of OGMDemuxNode is taken from the ogmtools package.
- *libxmlpp* - Some optional plug-ins and features of NMM use a modified version of libxml++, which is provided along with NMM.
- *libx264* - H264 encoding support. Used by FFMpegEncodeNode.
- *DVB API 5.1* - DVBS2ReadNode depends requires DVB API 5.1, which is part of the Linux kernel headers.
- *ulxmlrpcpp* - XML-RPC library used by example client of TVCaster using XML-RPC API.
- *openssl* - Dependency of ulxmlrpcpp.
- *expat* - Dependency of ulxmlrpcpp.

A.1 Informations on external libraries

Here is the legend of the tables:

Name	The original package name of the library
Description	A brief description of the library
Version	The version of the library which works with NMM. NMM may or may not be be compatible with newer versions of the library. If a custom version of the library is required, then this is indicated here.
Homepage	The official project homepage of the external library
Source	Name of the package which contains the full source code of the external library as well as build files required for building the library on all supported platforms. The package is either an original source package of the library or a customized source package which is based on an original source package.
Binary	List of platforms for which precompiled binaries of the library are provided.
Licence	The type of licence under which the external library is available. For custom licences (i.e. licences other than the well-known GNU licences), the full licence text is provided in this documentation.
Build Instructions	Instructions for building the library from provided sources and build files the on all platforms

A.1.1 a52dec

Name	a52dec
Description	AC3 audio decoder
Version	0.7.4
Homepage	http://liba52.sourceforge.net/
Source	tar.gz
Binary	Linux, PS3, Mac OS X, Windows
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf a52dec-0.7.4.tar.gz
cd a52dec-0.7.4
./configure --enable-shared --disable-static
make
make install
```

A.1.2 faad

Name	faad2
Description	AAC audio decoder
Version	2.7
Homepage	http://www.audiocoding.com
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENCE Version 2 (Commercial non-GPL licensing possible)

Build Instructions:

```
tar -xvjf faad2-2.7.tar.bz2
cd faad2-2.7
./configure --with-mp4v2
make
mkdir faad_installed
make install DESTDIR=<ABSOLUTE_PATH>/faad_installed
```

A.1.3 ffmpeg

Name	ffmpeg
Description	Various audio and video codecs
Version	0.5
Homepage	http://www.ffmpeg.org
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Build Instructions:

We assume that libmp3lame is installed in /home/bob. This is required to enable support for MP3 encoding using the ffmpeg library. If you do not need this, you can omit the –enable-libmp3lame configure switch and build ffmpeg without libmp3lame. NMM encoder plug-ins which use ffmpeg will not be able to encode audio in MP3 format. We assume that libx264 is installed in /home/bob. This is required to enable support for H264 encoding using the ffmpeg library.

```
Unpack the source file and set environment variables so that
dependencies are found:

tar xjf ffmpeg-0.5.tar.bz2
export LD_LIBRARY_PATH=/home/bob/lib
cd ffmpeg-0.5

Replace the following line in libavformat/avformat.h

#define MAX_STREAMS 20

by

#define MAX_STREAMS 99

If you need AMR support, build FFMPEG as follows:

cd libavcodec
mkdir amr_float
cd amr_float
unzip ../../../26104-510.zip
unzip 26104-510_ANSI_C_source_code.zip
cd ../../../

./configure --enable-shared --disable-static \
--enable-libmp3lame \
--enable-amr_nb --enable-amr_wb \
--extra-cflags=-I/home/bob/include \
--extra-ldflags=-L/home/bob/lib \
--enable-libx264 --enable-gpl
make
make install

If you do not need AMR support, build FFMPEG as follows:

./configure --enable-shared --disable-static \
--enable-libmp3lame \
--extra-cflags=-I/home/bob/include \
--extra-ldflags=-L/home/bob/lib \
--enable-libx264 --enable-gpl
make
make install
```

A.1.4 I1394

Name	I1394
Description	Management of IEEE1394 devices
Version	0.2.7
Homepage	http://sourceforge.net/projects/l1394/
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

We assume that libraw1394 is installed, and header files are available in the standard include paths. If libraw1394 is installed in a non-standard location, you can use the –with-raw1394 switch when running configure.

```
tar xzf l1394-0.2.7.tar.gz
cd l1394-0.2.7
./configure --enable-shared --disable-static
make -k
make -k install
```

A.1.5 libmp3lame

Name	libmp3lame
Description	MPEG layer3 audio encoder
Version	3.96.1
Homepage	http://lame.sourceforge.net
Source	tar.gz
Binary	Linux, Windows, PS3, Mac OS X
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf lame-3.96.1.tar.gz
cd lame-3.96.1
./configure --enable-shared --disable-static
make
make install
```

A.1.6 libraw1394

Name	libraw1394
Description	Low level IEEE1394 support
Version	as provided by SuSE 11.0
Homepage	http://sourceforge.net/projects/libraw1394
Binary	version abailable no
Licence	GNU LESSER GENERAL PUBLIC LICENCE Version 2.1

A.1.7 libmad

Name	libmad
Description	MPEG layer3 audio decoder
Version	0.15.1b
Homepage	http://www.underbit.com/products/mad/
Source	tar.gz
Binary	Linux, Windows, PS3, Mac OS X
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf libmad-0.15.1b.tar.gz
cd libmad-0.15.1b/
./configure --enable-shared --disable-static
make
make install
```

A.1.8 libdvdnav

Name	libdvdnav
Description	DVD reading and navigation
Version	0.10.1
Homepage	http://dvd.sourceforge.net/
Source	tar.gz
Binary	Linux, Windows, PS3, Mac OS X
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf libdvdnav-0.1.10.tar.gz
cd libdvdnav-0.1.10
./configure --enable-shared --disable-static
make
make install
```

A.1.9 libdvdread

Name	libdvdread
Description	DVD reading
Version	0.9.4
Homepage	http://www.dtek.chalmers.se/groups/dvd/
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf libdvdread-0.9.4.tar.gz
cd libdvdread-0.9.4/
./configure --enable-shared --disable-static
make
make install
```

A.1.10 libogg

Name	libogg
Description	OGG file parser
Version	1.1
Homepage	http://xiph.org/ogg/
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf libogg-1.1.tar.gz
cd libogg-1.1
```

```
./configure --enable-shared --disable-static
make
make install
```

A.1.11 libvorbis

Name	libvorbis
Description	OGG/Vorbis audio decoder
Version	1.0.1
Homepage	http://xiph.org/vorbis/
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf libvorbis-1.0.1.tar.gz
cd libvorbis-1.0.1
./configure --enable-shared --disable-static --with-ogg=/home/bob
make -k
make -k install
```

A.1.12 libshout

Name	libshout
Description	Injection of audio streams into a ShoutCast server
Version	2.0
Homepage	http://www.icecast.org/
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf libshout-2.0.tar.gz
cd libshout-2.0
./configure --enable-shared --disable-static --with-ogg=/home/bob
--with-vorbis=/home/bob
make
make install
```

A.1.13 fftw

Name	fftw
Description	Fast Fourier Transformation, used for audio visualization
Version	2.1.5
Homepage	http://www.fftw.org/
Source	tar.gz
Binary	Linux
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
tar xzf fftw-2.1.5.tar.gz
cd fftw-2.1.5
./configure --enable-shared --disable-static
make
make install
```

A.1.14 libliveMedia

Name	libliveMedia
Description	RTP support
Version	2009.06.02 with modifications by Motama
Homepage	http://live555.com/liveMedia/
Source	tar.gz
Binary	Linux, Windows
Licence	GNU LESSER GENERAL PUBLIC LICENCE Version 2.1

Build Instructions:

```
# Unpack
tar xzf live.2009.06.02.tar.gz
cd live

# Add -DUSE_SYSTEM_RANDOM to COMPILE_OPTS in config.linux (to avoid segfault)

# Build static libraries
./genMakefiles linux
make

# Build shared library
gcc -shared -o libliveMedia.so.2009.06.02 */*.a
ln -s libliveMedia.so.2009.06.02 libliveMedia.so
```

```
# install to \$INSTALLDIR
find -name \*.hh -exec cp "{}" \$INSTALLDIR/include/live ";"
find -name \*.h -exec cp "{}" \$INSTALLDIR/include/live ";"
cp libliveMedia.so.2009.06.02 libliveMedia.so \$INSTALLDIR/lib

Build Instructions (Windows):

Open Microsoft Visual Studio 2005.

Unpack the file live.2005.07.13.tar.gz.

Open the solution file liveMedia.sln from the source package.

In the combo box for selecting the build configuration, pick Release.

Choose "Build Solution" from the "Build" menu

Copy the DLL file from the Release directory to a directory that is
listed in your PATH environment variable.
This is required for running NMM-based applications that use this library.

Copy the LIB file from the Release directory to a directory that is listed as
additional library directory in your Visual Studio project files that use this
external library.
(This is only required for development of applications that use this
library directly.
You may not need to do this at all.)
```

A.1.15 mpeg2dec

Name	mpeg2dec
Description	MPEG version 1/2 video decoder
Version	0.4.0b
Homepage	http://libmpeg2.sourceforge.net/
Source	tar.gz
Binary	Linux, Windows, Mac OsX
Licence	GNU GENERAL PUBLIC LICENSE Version 2

Build Instructions:

```
tar xzf mpeg2dec-0.4.0b.tar.gz
cd mpeg2dec-0.4.0/
./configure --enable-shared --disable-static
make
make install
```

A.1.16 cdparanoia

Name	cdparanoia
Description	CDDA Support for Linux
Version	IIIalpha9.8-548
Homepage	hhttp://xiph.org/paranoia/
Source	tar.gz
Binary	Linux, Windows, PS3

A.1.17 libpng

Name	libpng
Description	PNG file reader
Version	1.0.13
Homepage	http://www.libpng.org/
Source	http://xiph.org/paranoia/down.html
Binary	no
Licence	GNU GENERAL PUBLIC LICENCE Version 2

Build Instructions:

```
# building
tar xzf libpng-1.2.19.tar.gz
cd libpng-1.2.19
./configure
make

# installing to $DESTDIR
cp libpng12.so* $DESTDIR/lib
cp libpng.so* $DESTDIR/lib
cp png.h $DESTDIR/include
```

A.1.18 asoundlib

Name	asoundlib
Description	ALSA headers and library
Version	1.0.9a
Homepage	http://www.alsa-project.org
Binary	no
Licence	GNU Lesser General Public License Version 2.1

A.1.19 Xlib

Name	Xlib
Description	X11 headers and libraries
Binary	no

A.1.20 libjpeg

Name	libjpeg
Description	JPEG support
Version	6.2.0-738
Homepage	http://www.ijg.org/
Source	http://www.ijg.org/files/
Binary	PS3

A.1.21 ImageMagick

Name	ImageMagick
Description	Imaging Support
Version	6.4.0.4 (as provided with OpenSuSE 11.0)
Homepage	http://www.imagemagick.org
Source	tar.bz2
Binary	Linux

Build Instructions:

```
tar xvfz ImageMagick-6.3.5-5.tar.gz
cd ImageMagick-6.3.5/
./configure
make
make install
```

A.1.22 ImageMagick for Windows

Name	ImageMagick for Windows
Description	Imaging Support
Version	6.4.6
Homepage	http://www.imagemagick.org
Source	ImageMagick-windows.zip
Binary	Windows
Licence	http://www.imagemagick.org/script/license.php

Build Instructions:

```
1) Run Visual Studio 2005
2) Open Solution ImageMagick-6.4.7/VisualMagick/configure/configure.sln
3) Click "Finish" in Conversion Wizard
4) Build Solution
5) Run Solution (CTRL+F5)
6) Press Next...Next...Finish in VisualMagick configuration tool
7) Open solution VisualMagick/VisualDynamicMT.sln
8) Click "Finish" in Conversion Wizard
9) Build Solution
```

A.1.23 mplayer

Name	mplayer
Description	Parts of the implementation of DirectXDisplayNode is based on MPlayer
Version	unkown
Homepage	http://www.mplayerhq.hu/
Source	Included in nmm/plugins/display/DirectX/DirectXDisplayNode.hpp and .cpp
Binary	Will be built from nmm/plugins/display/DirectX/DirectXDisplayNode.hpp and .cpp
Licence	GPL

A.1.24 vlc

Name	vlc
Description	Parts of the implementation of DirectXDisplayNode is based on vlc
Version	unkown
Homepage	http://www.videolan.org/
Source	Included in nmm/plugins/display/DirectX/DirectXDisplayNode.hpp and .cpp
Binary	Will be built from nmm/plugins/display/DirectX/DirectXDisplayNode.hpp and .cpp
Licence	GPL

A.1.25 transcode

Name	transcode
Description	Parts of the C implementation of deinterlacing algorithm in YUVDeInterlaceNode
Version	unkown
Homepage	http://www.transcoding.org
Source	Included in nmm/plugins/video/filter/yv12_deinterlace.h and .c
Binary	Will be built from nmm/plugins/video/filter/yv12_deinterlace.h and .c
Licence	GPL

A.1.26 ogmtools

Name	ogmtools
Description	Parts of the implementation of OGMDemuxNode is taken from the ogmtools package
Version	unkown
Homepage	http://www.bunkus.org/videotools/ogmtools/
Source	nmm/plugins/av/ogm/ogmstreams.h
Binary	Will be built from sources in nmm/plugins/av/ogm
Licence	GPL

A.1.27 libxml++

Name	libxml++
Description	Some optional plug-ins and features of NMM use a modified version of libxml++, which is provided along with NMM.
Version	unkown
Homepage	http://libxmlplusplus.sourceforge.net/
Source	nmm/utils/xml
Binary	Will be built from sources in nmm/utils/xml
Licence	LGPL

A.1.28 libx264

Name	libx264
Description	H264 video encoder
Version	x264-snapshot-20090401-2245
Homepage	http://www.videolan.org/developers/x264.html
Source	tar.bz2
Binary	Linux
Licence	GPL

Build Instructions:

```
NOTE: Use tcshell (use of bash causes problems with
./configure and make of libx264)

tar xjvf x264-snapshot-20090401-2245.tar.bz2
cd x264-snapshot-20090401-2245
./configure --enable-shared --disable-asm
make
make install
```

A.1.29 DVB API 5.1

Name	DVB API 5.1
Description	Linux DVB API
Version	5.1
Homepage	http://www.linuxtv.org/
Headers	Not provided (part of the Linux kernel headers)
Binary	not provided (part of the Linux kernel)
Licence	GPL

A.1.30 ulxmlrpcpp

Name	ulxmlrpcpp
Version	1.7.4
Homepage	http://ulxmlrpcpp.sourceforge.net
Binary	Linux
Licence	GNU LIBRARY GENERAL PUBLIC LICENSE Version 2

```
Note: You have to use --prefix option to specify install location.
Option $DESTDIR does not work.

tar xfj ulxmlrpcpp-1.7.4-src.tar.bz2
cd ulxmlrpcpp-1.7.4
./configure --with-openssl --prefix=$HOME/ulxmlrpcpp_install
make install
```

A.1.31 openssl

Name	openssl
Version	as provided with SuSE 11.0
Homepage	http://www.openssl.org
Binary	provided with SuSE 11.0

A.1.32 expat

Name	expat
Description	The Expat XML Parser
Version	as provided with SuSE 11.0
Homepage	http://expat.sourceforge.net/
Binary	provided with SuSE 11.0

Bibliografia

[1] R Andersson L Laflamme Sadigh S, A Reimers. Falls and fall related injuries among the elderly: a survey of residentialcare facilities in a swedish municipality. pages 129–140, 2004.

[2] Gostynski M. Prevalence, circumstances and consequences of falls in institutionalized elderly; a pilot study. 1991.

[3] N. Noury, A. Fleury, P. Rumeau, A.K. Bourke, G.O. Laighin, V. Rialle, and J.E. Lundy. Fall detection - principles and methods. pages 1663–1666, 2007.

[4] Exton-Smith AN. Care of the elderly: meeting the challenge of dependency. *Grimley Evans J editors. London: Academic Press*, 1977.

[5] Stephen R. Lord. Falls in older people - risk factors and strategies for prevention. 2001.

[6] Harvey AH Baker SP. Fall injuries in the elderly. clinics in geriatric medicine. volume 1, pages 501–512, 1985.

[7] Cooney LM Jr Marottoli RA, Berkman LF. Decline in physical function following hip fracture. pages 861–866, 1992.

[8] C.J. Lord and D.P. Colvin. Falls in the elderly: Detection and assessment. pages 1938 –1939, oct. 1991.

[9] G. Williams, K. Doughty, K. Cameron, and D.A. Bradley. A smart fall and activity monitor for telecare applications. volume 3, pages 1151 –1154 vol.3, oct. 1998.

[10] N. Noury, P. Barralon, G. Virone, P. Boissy, M. Hamel, and P. Rumeau. A smart sensor based on rules and its evaluation in daily routines. volume 4, pages 3286 – 3289 Vol.4, sep. 2003.

[11] T. et al. Zhang. Fall detection by embedding an accelerometer in cellphone and using kfd algorithm. *International Journal of Computer Science and Network Security*, pages 277–284, 2006.

[12] M.J. Mathie, J. Basilakis, and B.G. Celler. A system for monitoring posture and physical activity using accelerometers. volume 4, pages 3654 – 3657 vol.4, 2001.

[13] Wu G. Fall detection by embedding an accelerometer in cellphone and using kfd algorithm. *Jour. of Biomechanics*, pages 1497–1500, 2000.

[14] H. Nait-Charif and S.J. McKenna. Activity summarisation and fall detection in a supportive home environment. volume 4, pages 323 – 326 Vol.4, aug. 2004.

[15] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Monocular 3d head tracking to detect falls of elderly people. pages 6384 –6387, aug. 2006.

[16] Mihailidis A. An intelligent emergency response system: preliminary development and testing of automated fall detection. *J. Telemed. Telecare*, pages 194–198, 2005.

[17] Francesco Bolgan. Algoritmi di rilevazione del moto per sistemi di videosorveglianza. 2006.

[18] Chittaranjan Mandal Vinay Vishwakarma and Shamik Sural. Automatic detection of human fall in video. pages 616–623, 2007.

[19] Muhammad Jamil Khan and Hafiz Adnan Habib. Video analytic for fall detection from shape features and motion gradients. *World Congress on Engineering and Computer Science*, 2, 2009.

[20] Motama. Sito di nmm - http://www.motama.com/nmm.html.