ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica, Informatica e delle Telecomunicazioni

AUGMENTED REALITY: TECNOLOGIE OPEN E FUTURE IMPLEMENTAZIONI

Elaborata nel corso di: Ingegneria Del Software

Tesi di Laurea di:
MARCO PARI

Relatore:
Prof. ANTONIO NATALI

ANNO ACCADEMICO 2013–2014
SESSIONE I

PAROLE CHIAVE

OpenCV

Open Scource

Tecnologie

Realtà Aumentata

Futuro

Alla mia famiglia

Indice

Introduzione

Questa tesi nasce dalla seguente domanda:
Come puó uno sviluppatore approcciarsi al mondo della realtá aumentata?
Per rispondere a questa domanda ho preso come punto fisso la concezione generale di realtá aumentata e come tecnologia di riferimento lo standard open che negli ultimi anni é andato affermandosi nel settore come riferimento, ovvero le librerie OpenCV.
A questo proposito verranno trattati tramite una generale introduzione questi due argomenti per poi finire nei dettagli utili al raggiungimento dell'obiettivo prefissato.
Per entrambi gli argomenti si passerá da una spiegazione generale, per poi passare alla storia di queste tecnologie ed ai futuri utilizzi delle medesime, per concludere con il progetto di tesi da me sviluppato riguardante un'applicazione scritta in C# che si occupa dello scatto di foto attraverso una WebCam, per poi fare il confronto con una collezione di immagini contenute nel programma e se il programma riscontra un match tra le immagini salvate e quella scattata. Se il match ha riscontro positivo viene visualizzato un pdf relativo alla foto della collezione.

Capitolo 1

Realtà Aumentata

Introduciamo ora l'argomento realtà aumentata dandone una breve descrizione per poi inserirla in un contesto applicativo.

1.1 Generalità

1.1.1 Cosa significa Realtà Aumentata?

Per realtà aumentata (in inglese augmented reality, abbreviato AR), o realtà mediata dall'elaboratore, si intende l'arricchimento della percezione sensoriale umana mediante informazioni, in genere manipolate e convogliate elettronicamente, che non sarebbero percepibili con i cinque sensi. Gli elementi che 'aumentano' la realtà possono essere aggiunti attraverso un dispositivo mobile, come uno smartphone, con l'uso di un PC dotato di webcam o altri sensori, con dispositivi di visione (per es. occhiali a proiezione sulla retina), di ascolto (auricolari) e di manipolazione (guanti) che aggiungono informazioni multimediali alla realtà già normalmente percepita. Le informazioni 'aggiuntive' possono in realtà consistere anche in una diminuzione della quantità di informazioni normalmente percepibili per via sensoriale, sempre al fine di presentare una situazione più chiara o più utile o più divertente. Anche in questo caso si parla di AR. Nella realtà virtuale (virtual reality, VR), concetto più diffuso forse anche al gran numero di film e libri che la trattano come argomento, le persone si trovano immerse in un contesto virtuale, in cui l'unico elemento reale sono i loro sensi.

Nell'AR, invece, la persona continua a vivere la comune realtà fisica, ma usufruisce di informazioni aggiuntive o manipolate della realtà stessa. Quindi l'AR è il reciproco della realtà virtuale. La mediazione avviene tipicamente in tempo reale, per non alterare il senso di immediatezza della nostra percezione delle informazioni, ma questo non è un requisito obbligatorio. Le informazioni circa il mondo reale che circonda l'utente possono diventare interattive e manipolabili digitalmente. Esistono due tipi principali di AR:

* AR su dispositivo mobile. Il telefonino (o smartphone di ultima generazione) deve essere dotato necessariamente di Sistema di Posizionamento Globale (GPS), di magnetometro (bussola) e deve poter permettere la visualizzazione di un flusso video in tempo reale, oltre che di un collegamento Internet per ricevere i dati online. Il telefonino inquadra in tempo reale l'ambiente circostante; al mondo reale vengono sovrapposti i livelli di contenuto, dai dati da Punti di Interesse (PDI, altrimenti detti POI dall'inglese Point of Interest) geolocalizzati agli elementi 3D.

* AR su computer. È basata sull'uso di marcatori, (ARtags), di disegni stilizzati in bianco e nero che vengono mostrati alla webcam, vengono riconosciuti dal computer, e ai quali vengono sovrapposti in tempo reale i contenuti multimediali: video, audio, oggetti 3D, ecc.

ARTags o Codice QR

> Normalmente le applicazioni di AR sono basate su tecnologia Adobe Flash o HTML5. Anche Google stà esplorando questa nuova possibilità di interazione con l'utente utilizzando un proprio formato chiamato WebGL reso Open Source ma non ancora disponibile su tutto i browser.

La 'pubblicità aumentata' (Augmented advertising) è esplosa nel 2009 attraverso numerose campagne di marchi (Toyota, Lego, Mini, Kellogs, General Electrics), cantanti (Eminem, John Mayer) o riviste (Colors, Esquire Magazine o Wallpaper). Spam Magazine, nata nel 2012 in

Italia, è la prima rivista gratuita totalmente in AR (sia per quanto riguarda i contenuti editoriali sia per quelli commerciali).

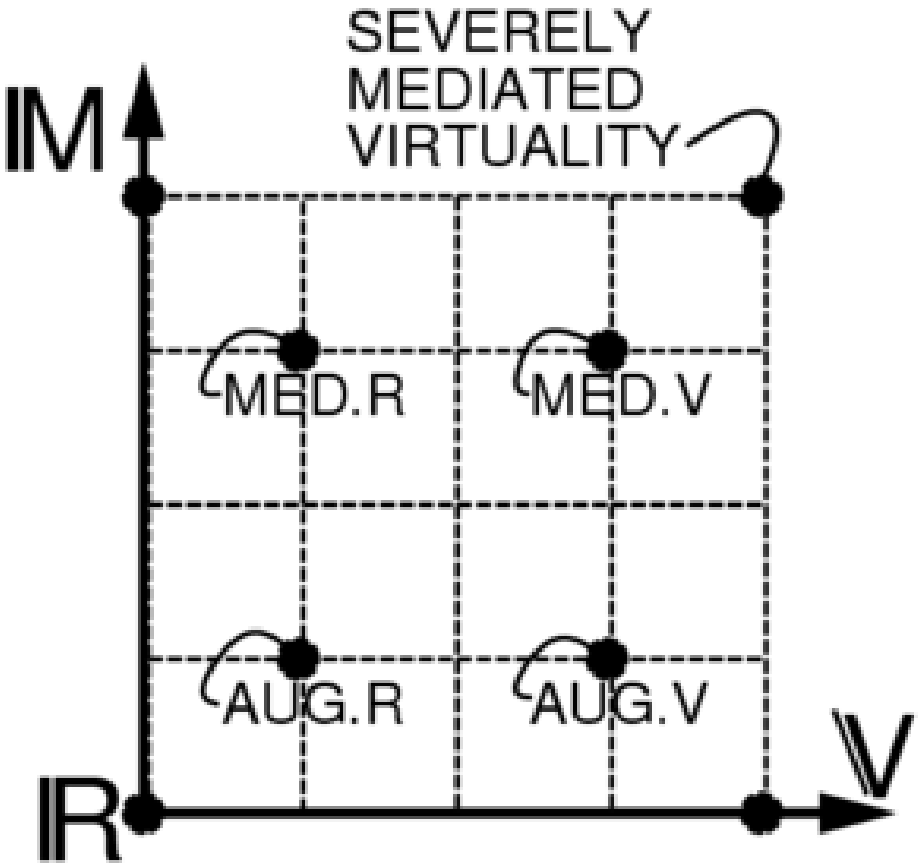
1.2 Definizione Formale

Il conio del termine augmented reality è attribuito a Thomas Caudell e David Mizell, ricercatori dell'azienda costruttrice di aeromobili Boeing e autori di '*Augmented reality: an application of heads-up display technology to manual manufacturing processes*' (1992). In questo documento, l'AR è definita come sovrapposizione di materiale generato dal computer al mondo reale, e contrapposta al già esistente concetto di realtà virtuale come alternativa più economica e meno esosa di risorse computazionali, ideale per assistere i meccanici Boeing nelle operazioni di costruzione e manutenzione degli aeromobili mediante l'uso di head-mounted display (HMD) in grado di visualizzarne gli schemi progettuali in semitrasparenza.

Milgram's Reality-Virtuality Continuum

La definizione di Caudell è poi approfondita da Paul Milgram e Fumio Kishino che, nella stesura di '*A Taxonomy of Mixed Reality Visual Displays*' (1994), introducono il Milgram's Reality-Virtuality Continuum (Continuum della Realta-Virtualita di Milgram), ossia l'intervallo continuo tra un ambiente puramente reale (R) ed uno puramente virtuale (V), i cui punti interni sono detti mixed reality. La mixed reality, quindi, si frappone tra i due estremi del continuum e rappresenta una composizione tra di essi, mediata dal computer, in cui reale e virtuale si incontrano e si fondono. Il variare della predominanza dell'uno o dell'altro, ossia la percezione complessiva della composizione come ambiente reale modificato da elementi virtuali o, viceversa, come ambiente virtuale modificato da elementi reali, implica una classificazione per tale composizione come augmented reality nel primo caso,

o come augmented virtuality nel caso opposto. Inquadrata all'interno del continuum, l'augmented reality è quindi un'istanza di mixed reality percepita da un osservatore come ambiente prevalentemente reale modificato da elementi virtuali. Tale definizione, però, non è sufficiente, in quanto si limita a considerare la prevalenza di elementi virtuali o reali senza tener conto di quanto essi siano poi simili o differenti dalla realtà. Due istanze di augmented reality in cui l'incidenza delle modifiche effettuate dal calcolatore è tutto sommato simile, possono essere percepite in modo completamente diverso dall'utente in base alla natura degli elementi coinvolti, e da quanto essi siano integrati o percepiti come distanti dal mondo reale a cui vengono sovrapposti. Inoltre, è anche possibile allontanarsi dalla realtà senza l'utilizzo di alcun elemento virtuale: prismi o sistemi ottici, ad esempio, possono distorcerne o alterarne la percezione senza che vi sia alcun tipo di composizione tra reale e virtuale. In quest'ultimo caso, in particolare, la tassonomia individuata da Milgram si rivela insufficiente, in quanto ciò che viene percepito è difficilmente classificabile come un ambiente puramente reale, e al tempo stesso non è classificabile come mixed reality per l'assenza di elementi virtuali. Steve Mann, nella stesura di '*Mediated Reality with implementations for everyday life*' (2002), estende il Reality-Virtuality Continuum ad una seconda dimensione con l'aggiunta di un secondo asse chiamato Mediality, che introduce un concetto da lui definito come mediated reality (realtà mediata), ovvero una generalizzazione della mixed reality di Milgram che consente di classificare realtà modificate in modo differente, e non necessariamente dipendente, dalla semplice presenza di elementi virtuali, come il già citato esempio di un prisma ottico. La mediated reality consente inoltre di introdurre il concetto di diminished reality (realtà diminuita), ovvero la rimozione digitale di elementi reali dal flusso video catturato, non classificabile nel continuum di Milgram in quanto non vi è alcuna presenza di elementi virtuali.



Tassonomia di Mann

Più nel dettaglio, la tassonomia di Reality, Virtuality, Mediality di Mann è descritta da un piano cartesiano con origine Reale (R), ovvero la realtà pura e non modificata, ed i cui assi sono Virtuality (V) per le ascisse e Mediality (M) per le ordinate. Quest'ultimo indica il livello di mediazione, ovvero quantifica l'entità delle modifiche effettuate. Più ci si allontana dall'origine e più la crescente mediazione causa una percezione di complessivo allontanamento dalla realtà. Per valori sufficientemente grandi di M, le regioni di augmented reality e augmented virtuality sono classificate, per sottolineare larilevanza delle modifiche, come mediated reality e mediated virtuality. Nel punto diagonalmente opposto all'origine, alla massima distanza sia lungo V che lungo M, in particolare, si ha una percezione di un mondo pura-

mente virtuale così fortemente mediato da essere completamente diverso dalla realtà (severely mediated virtuality). Va sottolineato che la tassonomia di Mann non è strettamente limitata al campo informatico e multimediale, ma è un concetto del tutto generale che, data la sempre crescente pervasività del virtuale nella vita di tutti i giorni dovuta alla rapida diffusione delle nuove tecnologie e di nuovi mezzi di comunicazione (e-mail, SMS), trova applicazioni anche in ambito sociologico e antropologico. Inoltre, come già detto in precedenza, visto che il concetto di Mediality consente di introdurre modifiche non necessariamente dovute alla presenza di elementi virtuali, non riguarda esclusivamente istanze di realtà mediata tramite l'utilizzo di un calcolatore. Per tale motivo spesso viene utilizzato il più specifico termine Computer-mediated reality, in modo da restringere la sua applicazione al campo delle modifiche effettuate da un computer. Inquadrata all'interno di tale tassonomia, quindi, l'augmented reality è definibile come un'istanza di Computer-mediated reality percepita da un osservatore come ambiente prevalentemente reale modificato da elementi virtuali (definizione del tutto equivalente a quella nel continuum di Milgram) per valori sufficientemente piccoli di Mediality tali da mantenere una percezione di complessiva similitudine alla realtà. Ronald Azuma, nella stesura di '*A Survey of Augmented Reality*' (1997), documento successivo a quello di Milgram ma precedente a quello di Mann, fornisce un'ulteriore definizione di AR tuttora largamente accettata, tramite l'individuazione di tre punti fondamentali:

* *Combinazione di reale e virtuale*
* *Interattiva in real time*
* *Registrata in 3D*

Il primo punto è fondamentalmente equivalente alla definizione di Milgram, ma il secondo e il terzo specificano due dettagli finora ignorati: l'AR deve offrire interattività in tempo reale, e gli elementi virtuali devono essere combinati a quelli reali (registration) in tre dimensioni. Si tratta, quindi, di una definizione più rigorosa e stringente, che esclude svariati esempi citati in precedenza come la sovraimpressione di testo o interfacce grafiche ad una trasmissione televisiva, e che invece rientrano nella più lasca definizione di AR data da Milgram e Mann.

Nella trattazione di questa tesi, si farà riferimento principalmente alla definizione data da Azuma, inquadrata all'interno della tassonomia di Mann come caso particolare di Computer-mediated reality, con qualche sporadico excursus su applicazioni non propriamente rientranti in tale definizione, ma comunque definibili come AR in modo lasco.

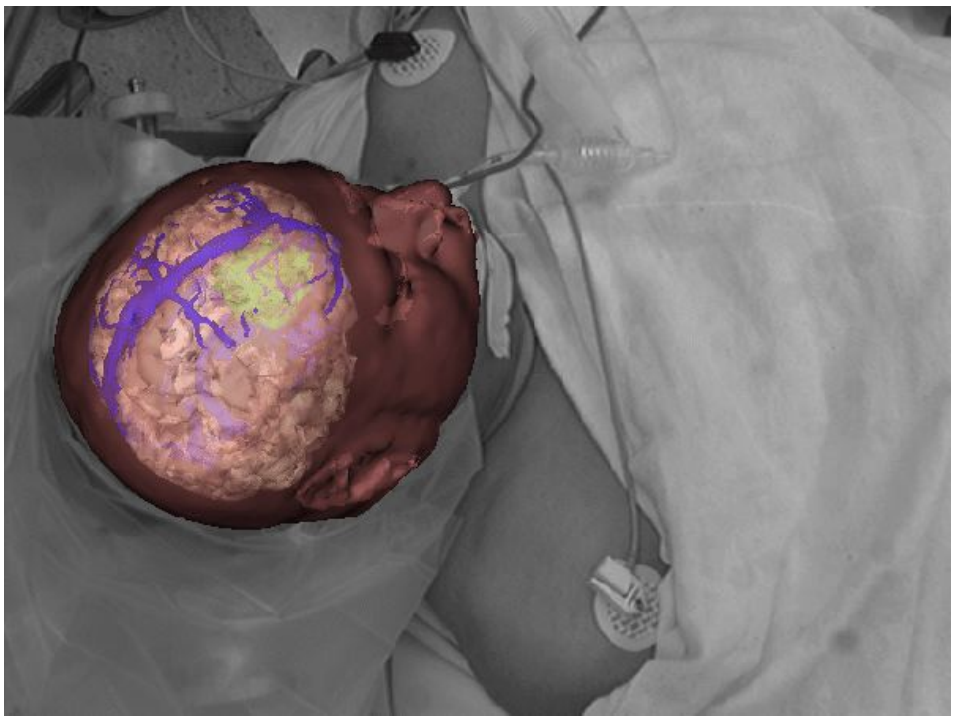
1.3 Applicazioni

In questa sezione verranno spiegati in maniera generale i vari campi in cui l'AR trova un'impiego concreto, che vada oltre il semplice fine ludico. Esso rimane comunque un'importante forma di utilizzo di questa tecnologia, ma si vuole rendere l'idea che quello non è e non sarà in futuro l'unico campo di utilizzo.

1.3.1 Ambito Medico

I medici potrebbero utilizzare l'AR come visualizzazione e aiuto alla formazione per un intervento chirurgico. Potrebbe essere possibile raccogliere set di dati 3D di un paziente in tempo reale, utilizzando sensori non invasivi come la risonanza magnetica (MRI), tomografia computerizzata (CT), o ecografia. Questi insiemi di dati potrebbero essere sovrapposti e combinati in tempo reale con una vista del paziente reale. In effetti, darebbe al medico una 'vista a raggi X' dell'interno di un paziente. Questo sarebbe molto utile durante chirurgia minimamente invasiva, che riduce il trauma di un'operazione utilizzando piccole incisioni o senza incisioni. Un problema delle tecniche minimamente invasive è che riducono la capacità del medico di vedere all'interno del paziente, rendendo più difficile la chirurgia. L'AR potrebbe fornire una vista interna senza necessità di grandi incisioni. L'AR potrebbe anche essere utile nelle attività generali di visualizzazione e individuazione per medici in sala chirurgica. Un esempio lampante è che i chirurghi possono rilevare alcune caratteristiche ad occhio nudo che non si possono vedere in RM o TAC, e viceversa. L'AR darebbe ai chirurghi accesso ad entrambi i tipi di dati simultaneamente, rendendo una miglior visione globale dello stato di un paziente. Si potrebbero anche guidare le attività di precisione, come ad esempio la visualizzazione di un punto in cui praticare un foro nel cranio per

un intervento chirurgico al cervello o dove eseguire una biopsia di un piccolo tumore. Le informazioni dai sensori non invasivi sarebbe visualizzata direttamente sul paziente, mostrando esattamente dove per eseguire l'operazione.

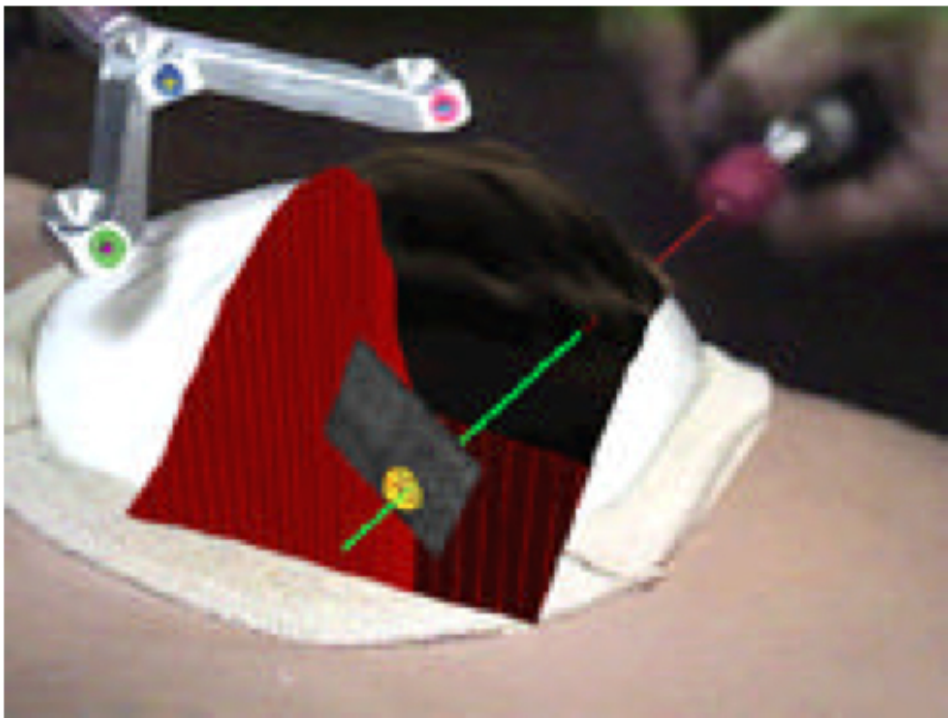


Cervello del Paziente senza aprirne il cranio

L'AR potrebbe anche essere utile per scopi di addestramento. Istruzioni virtuali potrebbero ricordare ad un chirurgo inesperto dei passaggi necessari, senza il bisogno di distogliere lo sguardo da un paziente e di consultare un manuale. Oggetti virtuali potrebbero anche individuare gli organi e specificare i percorsi per evitare di disturbare. Diversi progetti stanno esplorando questo campo di applicazione. Al UNC Chapel Hill, un gruppo di ricerca ha effettuato prove di scansione del ventre di una donna incinta con un sensore ad ultrasuoni, generando una rappresentazione 3D del feto all'interno del grembo materno e la visualizzazione che in un Head Mounted Display (HMD).

Head Mounted Display

L'obiettivo è quello di dotare il medico della capacità di vedere il movimento, il calciare del feto mentre giace all'interno del grembo materno, con la speranza che questo un giorno possa diventare un 'stetoscopio 3D'. Sforzi più recenti si sono concentrati su un ago per biopsia di un tumore al seno. La figura sottostante mostra il mockup di un'operazione di biopsia mammaria, dove gli oggetti virtuali identificano la posizione del tumore e guidano l'ago fino al suo bersaglio. Altri gruppi al laboratorio AI del MIT, General Electric e altrove stanno indagando sulla visualizzazione dei dati di risonanza magnetica o TAC, direttamente registrati sul paziente.

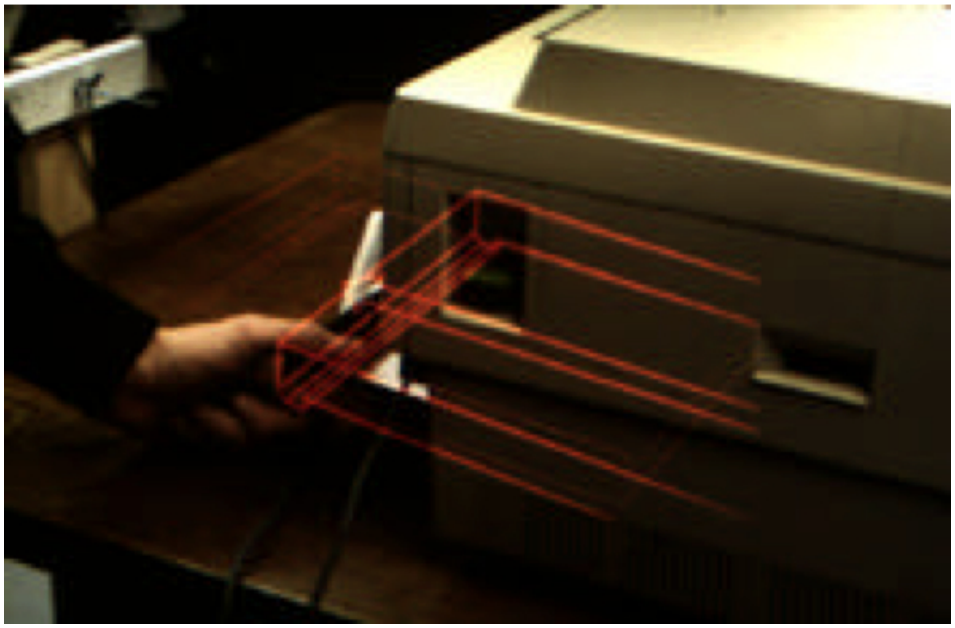


AR guida il chirurgo fino al tumore

1.3.2 Fabbricazione e Riparazione

Un'altra categoria di applicazioni per l'AR è il montaggio, la manutenzione e la riparazione di macchinari complessi. Le istruzioni potrebbero essere più facili da capire se fossero disponibili, non come manuali con testo e immagini, ma piuttosto come disegni 3D sovrapposte al dispositivo reale, che mostra passo-passo le operazioni che devono essere fatte e come farle. Questi disegni 3D possono essere animati, rendendo le indicazioni ancora più esplicite. Diversi progetti di ricerca hanno dimostrato prototipi per questo settore.Il gruppo di Steve Feiner in Columbia ha costruito un'applicazione per la manutenzione di una stampante laser. Le figure sottostanti, da diversi punti di vista, mostra dove il reticolo generato al computer sta dicendo all'utente di rimuovere il vassoio della carta.

Punto di vista Esterno

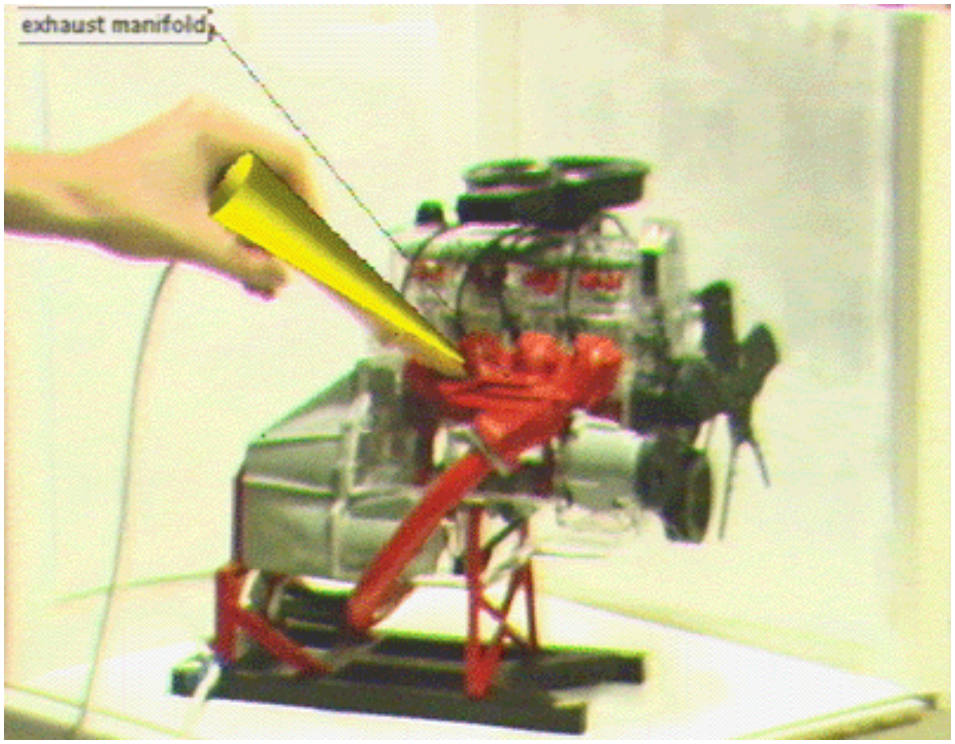


Punto di vista dell'Operaio

Un gruppo a Boeing sta sviluppando una tecnologia AR per guidare i tecnici nella costruzione di un cablaggio, che fa parte del sistema elettrico di un aeroplano. La memorizzazione di queste istruzioni in forma elettronica fa risparmiare spazio e ridurre i costi. Attualmente, i tecnici usano tavole di grandi dimensioni di layout fisico per costruire tali imbracature, e Boeing utilizza diversi magazzini in cui conserva tutte queste tavole. Tale spazio potrà essere svuotato per altro uso, se l'applicazione avrà successo. Boeing sta utilizzando un programma di reinvestimento Technology (PRT o Technology Reinvestment Program, TRP) in modo da portare questa tecnologia nella propria fabbrica.

1.3.3 Annotazione e Visualizzazione

L'AR potrebbe essere utilizzata per annotare oggetti e ambienti con informazioni pubbliche o private. Le applicazioni che utilizzano informazioni pubbliche presuppongono la disponibilità di banche dati pubbliche da cui attingere. Ad esempio, un display portatile potrebbe fornire informazioni sul contenuto di scaffali di una biblioteca quando l'utente va in giro la libreria. Presso il Centro Europeo di Ricerca per la Computer-Industry (ECRC), un utente può puntare a parti di un modello di motore e il sistema di AR visualizza il nome della parte che viene puntata. La figura sottostante mostra questo, quando l'utente punta al collettore di scarico su un modello di motore appatr l'etichetta 'collettore di scarico'.



Parti di Motore vengono evidenziate

In alternativa, queste annotazioni potrebbero essere note private collegate a oggetti specifici. I ricercatori in Columbia l'hanno dimostrato, partendo con l'idea di attaccare le finestre di un'interfaccia utente standard su posizioni specifiche nel mondo, e collegato a oggetti specifici come promemoria. La figura 8 mostra una finestra sovrapposta come un'etichetta su uno studente. Indossa un dispositivo di tracciamento, in modo che il computer conosce la sua posizione. Come lo studente si muove intorno, l'etichetta segue il suo percorso, fornendo all'utente AR con un promemoria di ciò che gli serve per parlare con gli studenti circa.

AR applicata al Social

AR porterebbero aiuto in compiti generali di visualizzazione. Un architetto con un HMD potrebbe essere in grado di guardare fuori dalla finestra e vedere come la proposta di nuovo grattacielo cambierebbe il panorama. Se un database contenente informazioni sulla struttura di un edificio è disponibile, AR potrebbe dare algi architetti una 'vista a raggi X' all'interno di un edificio, mostrando dove sono i tubi, le linee elettriche e i supporti strutturali dentro le mura. I ricercatori dell'Università di Toronto hanno costruito un sistema chiamato ARGOS (o Augmented Reality through Graphic Overlays on Stereovideo), che tra l'altro è usato per rendere delle immagini più facili da capire in condizioni di difficile visione.

1.3.4 Programmazione dei Percorsi per Robot

Radiocomandare un robot è spesso un problema difficile, specialmente quando il robot è lontano, con notevole ritardo nel collegamento di comunicazione. In questa circostanza, invece di controllare diretta-

mente il robot, può essere preferibile controllare invece una versione virtuale del robot. L'utente progetta e specifica le azioni del robot, manipolando la versione virtuale locale, in tempo reale. I risultati vengono visualizzati direttamente sul mondo reale. Una volta che il piano è testato e determinato, allora l'utente dice al robot reale di eseguire il piano specificato. Questo evita oscillazioni indotte causate dai ritardi. Le versioni virtuali possono anche prevedere gli effetti di manipolazione sull'ambiente, quindi servono come una pianificazione e uno strumento di anteprima per aiutare l'utente a svolgere l'attività desiderata. Il sistema ARGOS ha dimostrato che un'AR stereoscopica è una metodologia più facile e più accurata di una pianificazione del percorso del robot con interfacce tradizionali monoscopiche.

1.3.5 Aerei Militari

Per molti anni, aerei militari ed elicotteri hanno usato Head-Up Displays (HUD) e Helmet-Mounted Sights (HMS) per sovrapporre la grafica vettoriale alla vista del mondo reale del pilota. Oltre a fornire la navigazione di base e le informazioni di volo, su queste immagini sono talvolta registrati con gli obiettivi nell'ambiente, fornendo un modo per puntare le armi del velivolo. Per esempio, la torretta in un elicottero da combattimento può essere associata all'HMS del pilota, in modo che il pilota possa mirare con la torretta semplicemente guardando il bersaglio. Le future generazioni di aerei da combattimento saranno sviluppati con un HMD incorporato nel casco del pilota.

1.3.6 Intrattenimento

Al SIGGRAPH '95 (Special Interest Group on GRAPHics and Interactive Techniques), diversi espositori hanno mostrato 'Set Virtuali', che fondevano attori reali con sfondi virtuali, in tempo reale e in 3D. Gli attori dovevano solamente stare di fronte a un grande schermo blu o verde, mentre una macchina fotografica di movimento controllato dal computer registrava la scena. Dal momento che la posizione della telecamera viene tracciata e movimenti degli attori sono predeterminati, è possibile inserire digitalmente l'attore in uno sfondo virtuale 3D. Per esempio, l'attore potrebbe apparire all'interno di un grande anello

da filatura virtuale, in cui la parte anteriore dell'anello copre l'attore mentre la parte posteriore dell'anello è coperta dal attore. L'industria dello spettacolo vede questo come un modo per ridurre i costi di produzione: la creazione e la memorizzazione di set praticamente è potenzialmente più economica rispetto alla costruzione di sempre nuovi set fisici partendo da zero. Il progetto ALIVE dal MIT Media Lab compie un ulteriore passo avanti, popolando l'ambiente con le creature virtuali intelligenti che rispondono alle azioni dell'utente.

Roger Rabbit Aggiunto Virtrualmente

1.3.7 Direzioni Future Nel Mondo Del Lavoro

Questa sezione identifica le aree e gli approcci che richiedono ulteriori ricerche per la produzione di sistemi con un'AR migliorata.

* *Approcci Ibridi:* sistemi di monitoraggio futuri potranno essere ibridi, combinando approcci in grado di coprire le debolezze di quelli attuali. Lo stesso può essere vero per altri problemi dell'AR. Ad esempio, le strategie di registrazione correnti in genere si concentrano su una singola strategia. Sistemi futuri possono essere più robusti se molte tecniche vengono combinate. Un esempio è combinare le tecniche di visione a base di previsione. Se parti dei nastri non sono disponibili, il sistema passa alla

previsione per ridurre gli errori di registrazione, piuttosto che interrompersi completamente. Gli errori risolti con previsione a loro volta producono una più precisa localizzazione dalla stima iniziale.

* *Sistemi Real-Time e Computazione a Tempo Critico:* Molti sistemi in ambiente vituale (VE, Virtual Environments) non sono realmente eseguiti in tempo reale. Invece, è comune montare il sistema, spesso su UNIX, e poi vedere quanto velocemente viene eseguito. Questo può essere sufficiente per alcune applicazioni in VE. Poiché tutto è virtuale, tutti gli oggetti vengono sincronizzati automaticamente con l'altro. L'AR è una storia diversa. Ora il virtuale e il reale devono essere sincronizzati, e il mondo reale viene 'eseguito' in tempo reale. Pertanto, i sistemi ad AR efficaci devono essere costruiti con prestazioni in tempo reale. Timestamp esatti devono essere disponibili. I sistemi operativi non possono arbitrariamente scambiare il processo AR in qualsiasi momento, per durate arbitrarie, come farebbero con un qualsiasi processo in VE. I sistemi devono essere costruiti per garantire il completamento entro i tempi specificati, piuttosto che 'eseguire il più velocemente possibile.' Queste sono le caratteristiche dei simulatori di volo e di alcuni sistemi VE. La costruzione e il debug di sistemi real-time è spesso doloroso e difficile, ma i requisiti per un'AR richiedono performance in real-time.

* *Studi sulla Percezzione e sulla Psicofisica:* La Realtà Aumentata è un settore maturo per studi psicofisici. Quanto lag può rilevare un utente? Quanto errore di registrazione è rilevabile quando la testa è in movimento? Oltre a domande sulla percezione, sono anche necessari esperimenti psicologici che esplorano problemi di prestazioni. Quanto una previsione sulla head-motion può migliorare le prestazioni degli utenti su un compito specifico? Quanto errore di registrazione è tollerabile per una specifica applicazione prima che la performance di tale compito degrada notevolmente? L'errore più grande ammissibile è mentre l'utente muove la testa o quando la si ferma? Inoltre, non si sa molto su possibili illusioni ottiche causate da errori o conflitti nella vi-

sualizzazione simultanea di oggetti reali e virtuali. Sono stati condotti alcuni esperimenti in questo campo. Jannick Rolland, Frank Biocca e loro studenti hanno condotto uno studio sugli effetti causati da spostamenti oculari in video HMD. Hanno trovato che gli utenti parzialmente adattati allo spostamento degli occhi, ma avevano anche effetti collaterali negativi dopo la rimozione della HMD. Il gruppo di Steve Ellis al NASA Ames ha svolto lavori in profondità percepita in un HMD. ATR ha anche condotto uno studio.

* *Portabilità:* Alcune potenziale applicazioni per l'AR hanno dei requisiti che permettono all'utente la possibilità di muoversi attraverso grandi ambienti, anche all'aperto. Uno di essi è rendere le apparecchiature contenute nelle dimensioni e portatili. La tecnologia di tracciamento esistente non è in grado di tracciare un utente esterno con la precisione richiesta.

* *Display Multimodali:* Quasi tutti i lavori sull'AR sono concentrati sul senso visivo: oggetti grafici virtuali e sovrapposizioni. Ma l'aumento potrebbe applicarsi a tutti gli altri sensi. In particolare, l'aggiunta e rimozione audio 3D è una capacità che può essere utile in alcune applicazioni AR.

* *Problemi Politici e Sociali:* Gli aspetti tecnologici non sono gli unici che devono essere considerati quando si costruisce un'applicazione reale. Ci sono anche delle dimensioni sociali e politiche da considerare quando nuove tecnologie cadono nelle mani degli utenti reali. Talvolta la percezione è ciò che conta, anche se la realtà tecnologica è diversa. Per esempio, se i lavoratori percepiscono il laser come un rischio per la propria salute, possono rifiutare di utilizzare un sistema con laser nel display, anche se tali laser sono sicuri per gli occhi. Ergonomia e facilità d'uso sono considerazioni fondamentali. Se l'AR è veramente una soluzione più economica delle alternative nelle sue applicazioni proposte deve essere determinato, non lo è a prescindere. Un altro fattore importante è se la tecnologia è percepita come una minaccia per i posti di lavoro, in sostituzione di lavoratori, soprattutto con molte aziende che subiscono recenti licenziamenti.

L'AR può fare bene in questo senso, perché è inteso come uno strumento per rendere più facile il lavoro dell'utente, piuttosto che qualcosa che sostituisce completamente il lavoratore umano. Anche se il trasferimento di tecnologia non è normalmente un oggetto di pubblicazioni accademiche, si tratta di un problema reale. Preoccupazioni sociali e politiche non dovrebbero essere ignorate durante i tentativi di spostare l'AR fuori dal laboratorio di ricerca e nelle mani degli utenti reali.

1.3.8 Direzioni Future Nella Quotidianità

Infine mi sembra doveroso inserire l'AR in un contesto generalizzato, ponendomi la domanda: Cosa cambierà per gli utenti nella vita di tutti i giorni grazie a queste tecnologie? La risposta non è semplice, visto che nascono continuamente nuove idee su come utilizzare queste nuove tecnologie per rendere le mansioni svolte quotidianamente più facili o più immediate, ma uno scorcio di futuro lo possiamo intravedere attraverso quel device, ancora in fase di testing e che verrà rilasciato all'inizio del 2014, prodotto da Google che prende il nome di Google Glass:

Google Glass

Si tratta, pratica, di una versione miniaturizzata degli HMD precedentemente trattati, ma progettati per poter essere utilizzati quotidianamente senza interferire nelle interazioni sociali o in quelle pratiche che richiedono concentrazione, come la guida. Le potenzialità di questo strumento stanno nell'avere la conoscenza presente su internet a portata 'd'occhio' visto che basta pronunciare un comando predefinito (nell'immagine si nota la frase 'ok glass' da pronunciare per attivare le funzioni vocali, altrimenti il display rimarrà trasparente con solamente l'ora corrente visualizzata) per accedere a tutte le funziuonalità disponibili. La parola 'pronunciare' non è stata utilizzata casualmente visto che, a parte un piccolo trackpad posizionato sopra l'orecchio destro, il principale metodo di input saranno proprio i comandi vocali. Le API per gli sviluppatori sono già state distribuite, anche se solo una stretta cerchia di developer (in continua espansione) ha a disposizione un paio di Glass su cui fare pratica.

Capitolo 2

OpenCV

In questo capitolo viene introdotto l'argomento OpenCV la sua importanza all'interno della tesi.

2.1 Generalità

2.1.1 Cosa significa OpenCV?

OpenCV (Open Computer Vision) è una libreria riguardante funzioni principalmente utilizzate nella computer vision a real time. La visione artificiale è l'insieme dei processi che mirano a creare un modello approssimato del mondo reale (3D) partendo da immagini bidimensionali (2D), quindi una trasformazione dei dati provenienti da una fotocamera o videocamera verso una decisione o una loro nuova rappresentazione. Una tale trasformazione è effettuata per un qualsiasi scopo. E' possibile inserire nei dati, anche delle informazioni riguardanti il contesto in cui si opera. Essendo ormai l'acquisizione di immagini un problema banale con le tecnologie attulamente disponibili, che battono l'occhio umano sotto ogni punto di vista (risoluzione, velocità o sensibilità), rimane l'importante problema di dare un senso all'immagine che viene catturata. Questa elaborazione per quanto

possa sembrare estremamente facile per il cervello umano, è molto più complessa in ambito informatico. Sono rilasciate sotto licenza BSD (Berkeley Software Distribution, prima licenza UNIX), rendendone libero l'utilizzo a scopi commerciali e di ricerca. In pratica si tratta di una famiglia di licenze software permissive delle quali molte vengono considerate Open Source, che a grandi linee permette la 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 promozianale di prodotti derivati. Originariamente scritte in C ora hanno un'interfaccia C++ e tutte le nuove implementazioni sono in questo linguaggio. Esiste anche un interfaccia completa in Python, Java e MATLAB/OCTAVE (versione 2.5 in poi), le cui API possono facilmente essere reperite su internet. Frammenti scritti in altri linguaggi come C#, Ch, Ruby sono stati scritti per incoraggiare l'utilizzo di queste librerie ad una più ampia utenza. Le librerie possiedono più di 2500 algoritmi ottimizzati, utilizzati in tutto il mondo con più di 2,5 milioni di download e 40000 gruppi di utenti attivi. Esempi di applicazioni possono essere la Human-Computer Interaction (HCI); l'identificazione, segmentazione e il riconoscimento di oggetti; riconoscimento di volti e gesti; tracking di singoli o più movimenti, ecc.

2.1.2 Storia

Lanciato ufficialmente nel 1999, il progetto OpenCV è stato un'iniziativa del gruppo di ricerca Intel per applicazioni che utilizzavano la CPU in maniera intensa, parte di una serie di progetti tra cui Raytracing in tempo reale e 3D display walls. Il maggior contributo al progetto includono un numero di ottimizzazioni fatte da esperti della Intel in Russia così come dalla Intel's Performance Library Team. Nei suoi primi giorni di vita, l'obiettivo del progetto era descritto come una ricerca sulla Visone Avanzata che non solo doveva essere open, ma doveva fornire del codice ottimizzato per creare una buona base per future infrastrutture. Lo scopo era quello di creare una conoscenza diffusa sulla visione distribuendo una struttura comune su cui gli sviluppatori avrebbero potuto costruire le porprie applicazioni, in modo ch il codice fosse immediatamente leggibile e trasferibile. Le

applicazioni commerciali basate sulla visione avanzata possono essere create attraverso del codice ottimizzato portatile reso disponibile gratuitamente con una licenza che non deve essere necessariamente gratuita e open. La prima versione alpha di OpenCV è stata rilasciata al pubblico durante una conferenza della IEEE riguardante la Computer Visione e il Pattern Recognition nel 2000, e ben cinque beta sono state rilasciate tra il 2001 e il 2005. La prima versione stable 1.0 è stata rilasciata nel 2006. A metà 2008, openCV ottenne il supporto di Willow Garage ed è tutt'ora in sviluppo attivo. Una versione 1.1 prerelease è stata distribuita ad Ottobre 2008. La seconda major release di OpenCV è stata rilasciata ad Ottobre 2009. OpenCV 2.0 include grossi cambiamenti nell'interfaccia C++, nuove funzioni, più facile comprensione, miglior implementazione delle funzioni già esistenti in termini di performance(specialmente su sistemi a più core). Ufficilamente viene rilasciata una nuova release ogni 6 mesi e lo sviluppo è ora portato avanti da un team russo indipendente supportato da corporazioni commerciali. Successivamente, durante Settembre 2010, in OpenCV viene inserita una logica CUDA (architettura di elaborazione in parallelo realizzata da NVIDIA che permette netti aumenti delle prestazioni di computing grazie allo sfruttamento della potenza di calcolo delle GPU). Nell'Agosto 2012 il supporto dei OpenCV è stato preso da una fondazione no-profit, la OpenCV.org che ne mantiene attivo lo sviluppo e il sito di riferimento.

2.2 Librerie Grafiche

Con il termine di libreria grafica si identificano generalmente almeno tre famiglie di librerie con scopi differenti:

* I toolkit, librerie primitive per la creazione di oggetti grafici di interfaccia;

* Librerie di rendering e multimedia, come DirectX e OpenGL, orientate alla massima performance nella creazione di effetti poligonali o vettoriali il cui utilizzo più comune è teso all'ottenimento di elevate prestazioni grafiche sfruttate nei videogame o nelle applicazioni multimediali;

* Librerie di gestione di hardware grafico come digitalizzazioni e frame grabber;

Le OpenCV, anche se includono alcune funzionalità tipiche di ciascuna delle famiglie citate, non fanno parte di nessuno di questi gruppi. L'utilizzo primario è infatti quello collegato alla visione artificiale, il cui problema principale, come già detto, è quello di estrarre dalle immagini dati significativi e trattabili in modo automatico. Tale campo di studio trova le sue applicazioni nella robotica, nei sistemi di videosorveglianza evoluti e nei sistemi di monitoraggio e sicurezza, oltre che in ogni sistema di archiviazione automatica di informazioni visive.
Di seguito alcuni esempi di ambienti di sviluppo open che utilizzano le librerie OpenCV:

* 2D and 3D feature toolkits
* Egomotion estimation
* Facial recognition system
* Gesture recognition
* Human-computer interaction (HCI)
* Mobile robotics
* Motion understanding
* Object identification
* Segmentation and Recognition
* Stereopsis Stereo vision: depth perception from 2 cameras
* Structure from motion (SFM)
* Motion tracking
* Augmented reality

Per supportare alcune delle aree sopracitate, OpenCV include una libreria di apprendimento automatico statistico che contiene:

* Boosting (meta-algorithm)
* Decision tree learning
* Gradient boosting trees
* Expectation-maximization algorithm
* k-nearest neighbor algorithm
* Naive Bayes classifier
* Artificial neural networks
* Random forest
* Support vector machine (SVM)

Tutte le funzioni della libreria sono accomunate dal prefisso 'cv', mentre i nomi delle classi utilizzate hanno prefisso 'Cv' (con la C maiuscola), ad eccezione della IplImage, ereditata come si vede dal nome dalla libreria Ipl. La libreria inoltre è dotata di diverse strutture dati come CvMat (matrici numeriche), IplImage (Buffer immagine), CvMemStorage (Buffer dinamici), CvSeq (liste dinamiche), grafi e alberi, con le necessarie funzioni di gestione. E' inoltre disponibile un completo ventaglio di funzioni di disegno diretto: linee, ellissi, poligonali, testo, ecc. Nel caso in cui ci fosse la necessità di ricorrere a librerie dedicate, specifiche per la gestione dell'hardware adottato, le funzioni della libreria permettono un facile scambio di dati con oggetti provenienti da altre librerie. Tra le funzioni più avanzate, OpenCV mette a disposizione molti degli algoritmi più raffinati oggi disponibili. Citando alcuni degli algoritmi più importanti va detto che uno dei problemi più comuni affrontati nel trattamento immagini riguarda la delimitazione di oggetti, o in generale la loro identificazione a scopo di misura, conteggio o identificazione, come ad esempio nel riconoscimento dei caratteri (OCR) oppure nella trasformazione di immagini da raster a vettoriali per i quali è presente l'algoritmo di Canny per l'identificazione degli spigoli dell'immagine (cioè zone di confine tra aree differenti). Tale algoritmo è attualmente considerato uno dei migliori disponibili allo scopo, ma oltre al metodo di Canny sono disponibili

per scopi analoghi, i filtri di Sober e Laplace ed una completa serie di funzioni di classificazione geometrica dei contorni, utili a trasformare in informazione numerica le immagini trattate. In materia di analisi di immagini in movimento (motion detection, object traking, ecc.) sono presenti applicazioni degli algoritmi di Lucas e Kanade, e di Horn e Schunk, utili per il calcolo dei flussi ottici per la rilevazione del movimento e per il block matching (rilevazione di elementi uguali in posizioni differenti tra immagini successive). Ancora per quanto riguarda il patten matching, è presente un classificatore basato sul metodo di Haar ed ottimizzato per il riconoscimento di volti umani e utilizzabile anche per il riconoscimento di oggetti dopo una adeguata istruzione. Infine esiste un'implementazione del filtro di Kalman e dell'algoritmo di condensation, utilizzati nella riduzione del rumore e nei problemi di previsione.

2.3 Struttura

La struttura di OpenCV consta delle seguenti 6 parti:

1: CXCORE: contiene la definizione di tutte le strutture dati e le funzioni per gestire immagini e video.

2: CV: contiene tutte le funzioni per il processamento e l'analisi delle immagini, la calibrazione, il tracking e il pattern recognition.

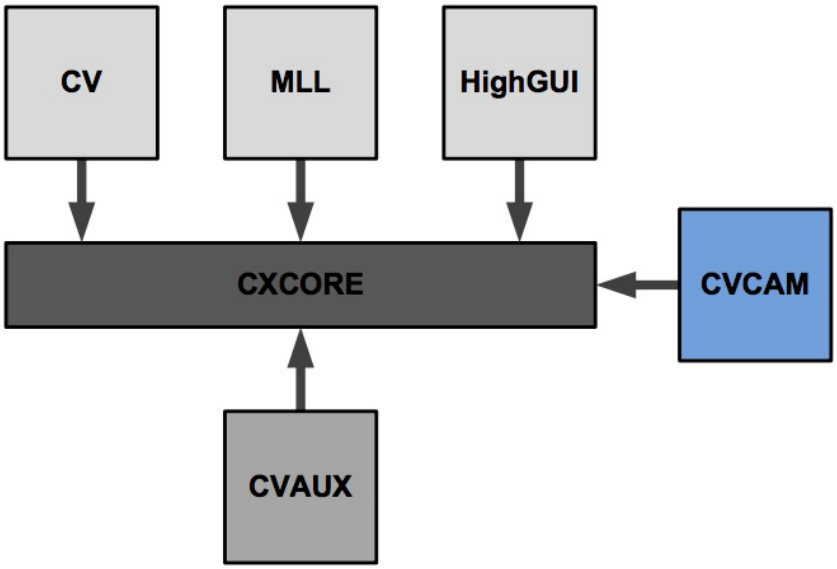
3: MLL (Machine Learning Language): contiene molte funzioni sul Machine Learning, quali il clustering e la classificazione.

4: HighGUI: contiene le definizioni delle interfacce utenti (GUI).

5: CVCAM: contiene le interfacce per le webcam.

6: CVAUX: contiene algoritmi sperimentali per scopi diversi, ad esempio: segmentazione, sottrazione del background, modelli HMM (Hidden Markov Model), ecc.

La struttura di OpenCV precedente può essere riassunta nella schematizzazione seguente.



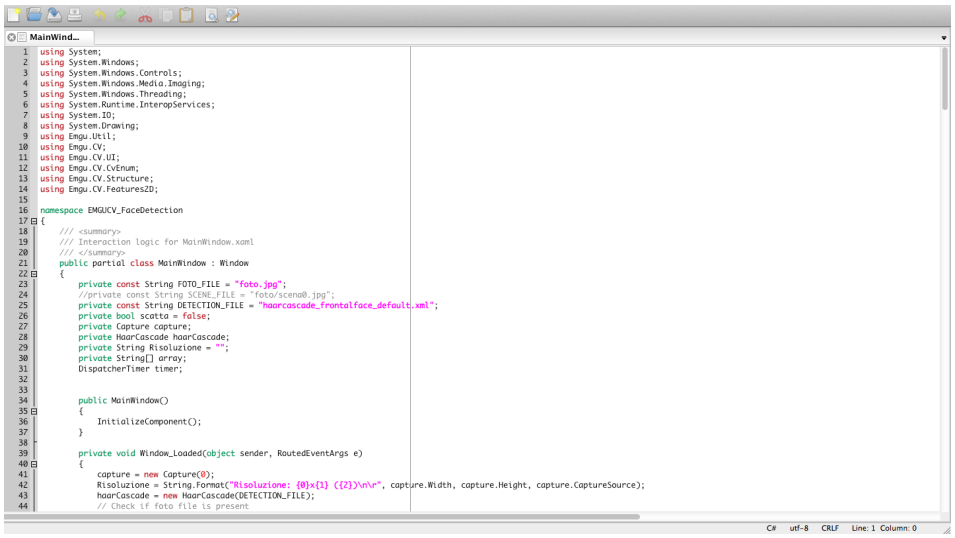
Struttura OpenCV

Per velocizzare OpenCV è possibile utilizzare la libreria IPP (Intel Performance Primitives). Infatti OpenCV è basato, internamente, su tali librerie in quanto molti programmatori Intel ne fanno un forte uso. Le IPP sono delle librerie altamente ottimizzate, che agiscono direttamente a livello delle micro-istruzioni del processore, evitando qualsiasi perdita di tempo. Se le IPP sono installate, OpenCV le rileverà in automatico e le includerà al momento della compilazione, per ottimizzare al massimo il codice. Così se, ad esempio è richiesto il calcolo di una FFT, verrà utilizzata la funzione definita nelle IPP anzichè quella nativa di OpenCV.

2.4 Semplice Esempio

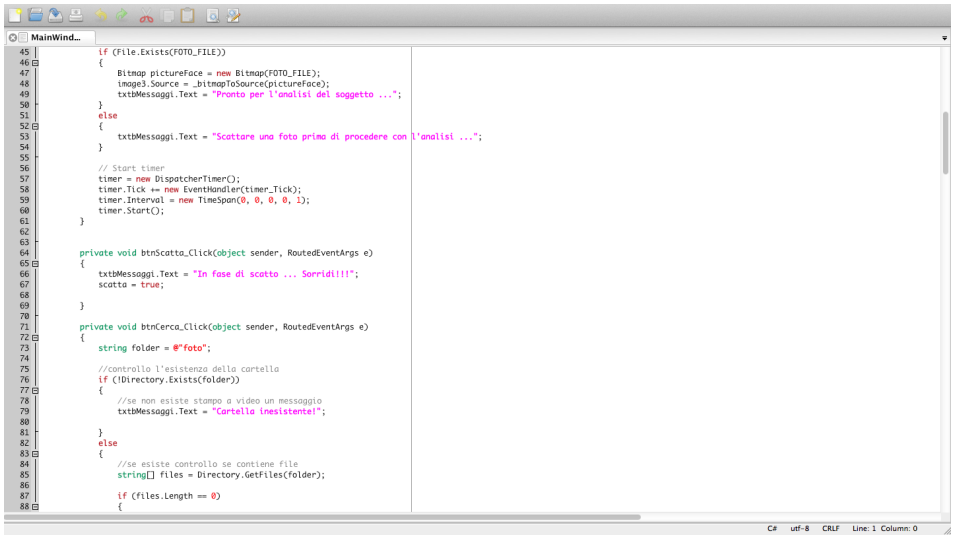
Dopo questa parte di teoria passiamo ad una parte più pratica. Avendo note le potenzialità delle librerie OpenCV per quanto riguarda il face tracking e il riconoscimento facciale, ho proceduto con la creazione di questa applicazione.
Scritta in C#, si tratta di un'applicazione che evidenzia i volti ripresi dalla WebCam e scattando una foto campione è in grado di analizzarla e confrontarla con ciò che è memorizzato in una cartella 'foto' e

stabilire se si tratta dello stesso soggetto. In seguito al riconoscimento del volto viene mostrato un pdf associato ad esso, in modo da fornire in maniera rapida e semplice informazioni sul soggetto.
Di seguito il codice del metodo richiamato al caricamenteo dell'applicazione.



```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media.Imaging;
using System.Windows.Threading;
using System.Runtime.InteropServices;
using System.IO;
using System.Drawing;
using Emgu.Util;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.Features2D;

namespace EMGUCV_FaceDetection
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private const String FOTO_FILE = "foto.jpg";
        //private const String SCENE_FILE = "foto/scena0.jpg";
        private const String DETECTION_FILE = "haarcascade_frontalface_default.xml";
        private bool scatta = false;
        private Capture capture;
        private HaarCascade haarCascade;
        private String Risoluzione = "";
        private String[] array;
        DispatcherTimer timer;


        public MainWindow()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            capture = new Capture(0);
            Risoluzione = String.Format("Risoluzione: {0}x{1} ({2})\n\r", capture.Width, capture.Height, capture.CaptureSource);
            haarCascade = new HaarCascade(DETECTION_FILE);
            // Check if foto file is present
```

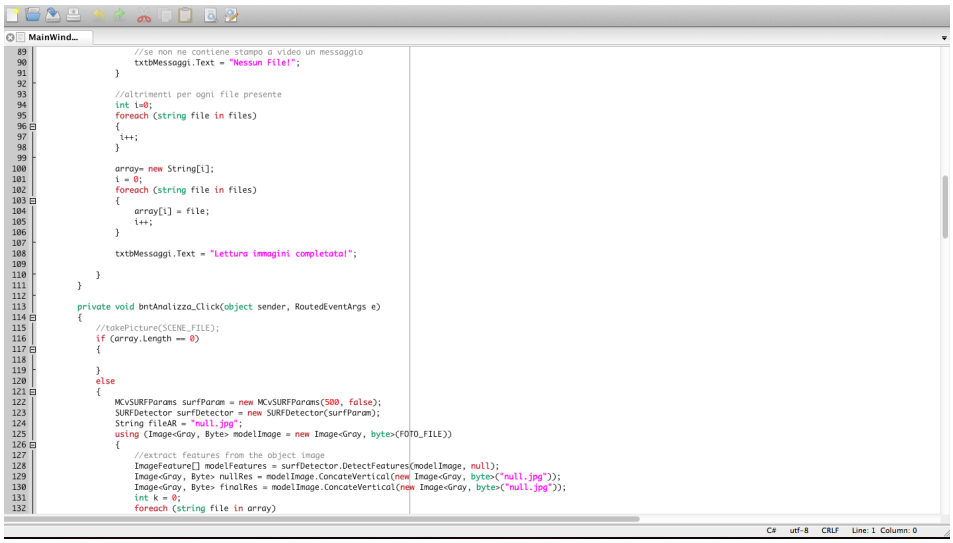
Prima Parte



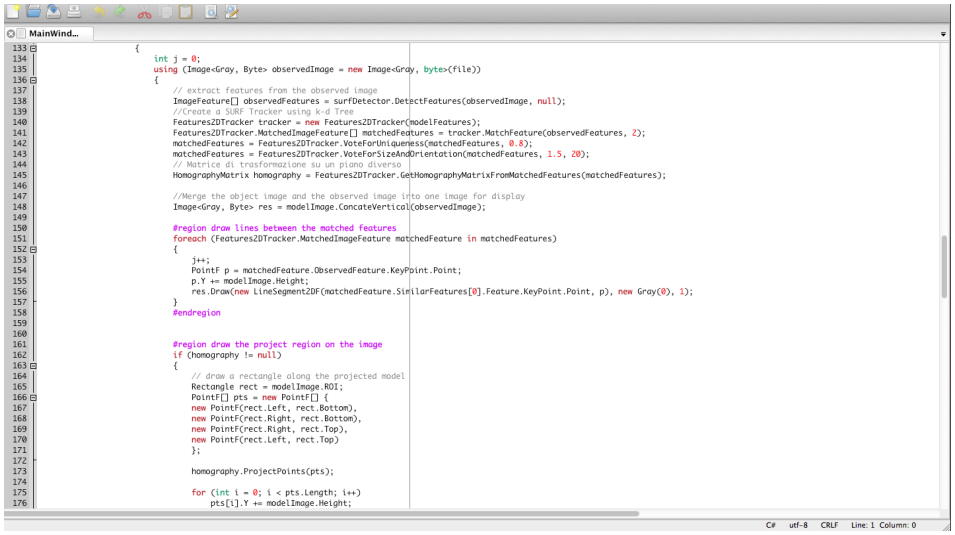
```
            if (File.Exists(FOTO_FILE))
            {
                Bitmap pictureFace = new Bitmap(FOTO_FILE);
                image3.Source = _bitmapToSource(pictureFace);
                txtbMessaggi.Text = "Pronto per l'analisi del soggetto ...";
            }
            else
            {
                txtbMessaggi.Text = "Scattare una foto prima di procedere con l'analisi ...";
            }

            // Start timer
            timer = new DispatcherTimer();
            timer.Tick += new EventHandler(timer_Tick);
            timer.Interval = new TimeSpan(0, 0, 0, 0, 1);
            timer.Start();
        }


        private void btnScatta_Click(object sender, RoutedEventArgs e)
        {
            txtbMessaggi.Text = "In fase di scatto ... Sorridi!!!";
            scatta = true;

        }

        private void btnCerca_Click(object sender, RoutedEventArgs e)
        {
            string folder = @"foto";

            //controllo l'esistenza della cartella
            if (!Directory.Exists(folder))
            {
                //se non esiste stampo a video un messaggio
                txtbMessaggi.Text = "Cartella inesistente!";

            }
            else
            {
                //se esiste controllo se contiene file
                string[] files = Directory.GetFiles(folder);

                if (files.Length == 0)
                {
```

Seconda Parte

L'oggetto principale è il Capture, in pratica è quello che cattura i frame dalla WebCam, è possibile anche definire alcuni parametri supplementari, ma in questo caso prende le impostazioni base. Da notare il numero 0 tra le parentesi quando instanzia l'oggetto, serve ad identificare quale WebCam utilizzare, nell'ipotesi che ne abbiate più di una disponibile, naturalmente il primo pensiero va alla possibilità di utilizarne due contemporaneamente per la visione stereoscopica. Altro oggetto importante è l'HaarCascade tramite il quale è indicato un file XML, dove sono definite le regole per il riconoscimento dei volti. Tra i vari file delle OpenCV che si scaricano, c'è anche un eseguibile per creare questi file XML. Il resto del codice è per controllare se è gia stata scattata una foto, la parte importante è quella relativa alla definizione del Timer che ad ogni Tick richiamerà il metodo associato, visibile nel codice che segue.



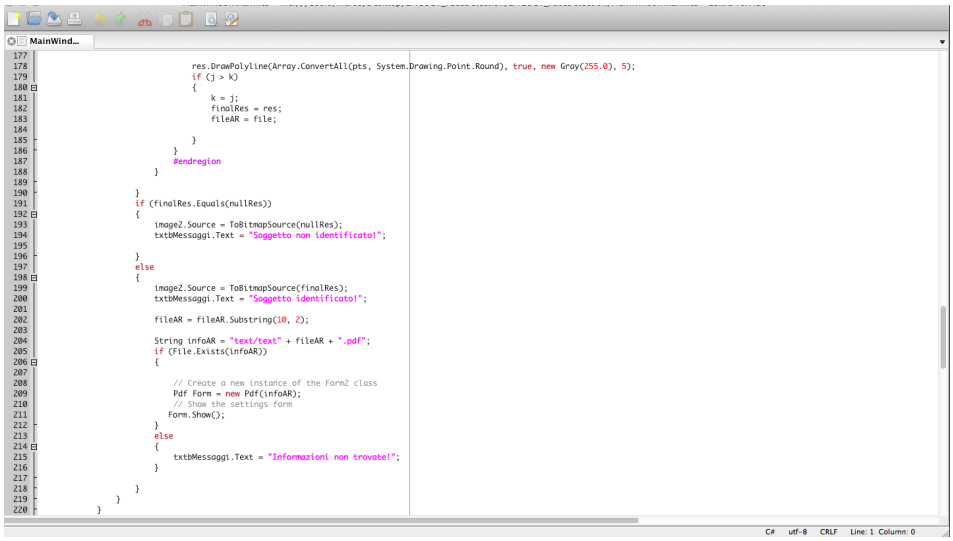
Terza Parte



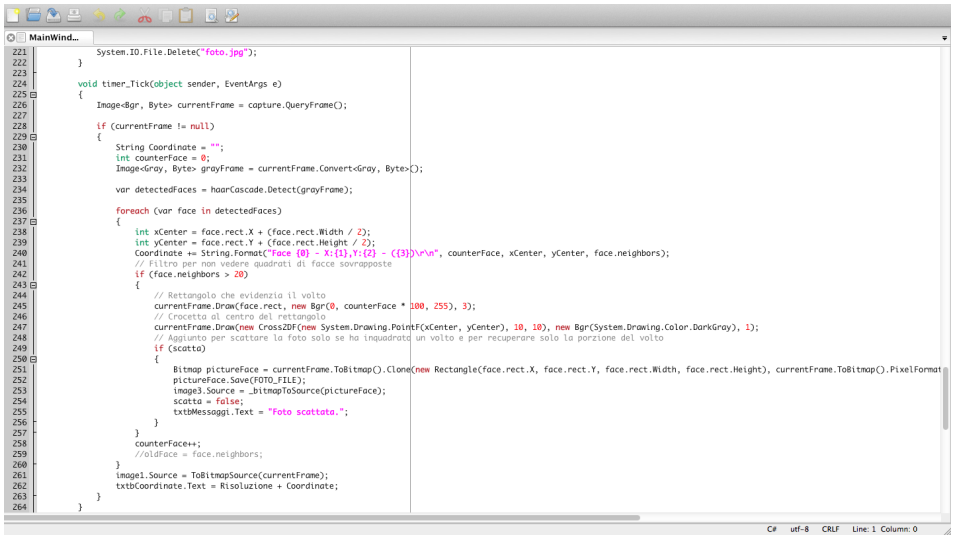
Quarta Parte

La prima operazione che viene eseguita è la cattura di una 'istantanea' tramite il metodo QueryFrame dell'oggetto Capture, l'oggetto Image restituito viene poi elaborato per essere convertito in una immagine in scala di grigi e poi dato in pasto al metodo Detect dell'oggeto HaarCascade di cui parlavo prima. Il risultato è un oggetto che rappresenta tutte le facce, e le loro proprietà, che il metodo è riuscito ad identificare nell'istantanea. Il ciclo foreach serve per elaborare i dati 'di ogni' faccia riconosciuta. In pratica per ogni faccia viene identificata un area rettangolare dell'istantanea che la contiene, recupero queste informazioni per mostrarle poi a video sia sotto forma di coordinate che di rettangolo che evidenzia il volto. Il controllo 'if(face.neighbors>20)' è stato usato usato per evitare che venissero disegnati quadrati sovrapposti, può capitare un po' di tutto anche che riconosca cose strane come volti. Il controllo sulla variabile 'scatta' serve a fare in modo che l'immagine da usare come campione per l'analisi venga scattata, dopo la pressione dell'apposito pulsante, solo quando è evidenziato almeno un volto. Il metodo btnCercaClick serve per caricare in un array tutte le immagini contenute in una cartella 'foto', con le quali faremo poi il confronto con l'istantanea appena scattata. Il metodo bntAnalizzaClick è quello che si occupa di analizzare l'immagine del volto salvata precedentemente con quelle contenute nella cartella 'foto'. Uno degli oggetti fondamentali è il SURFDetector che

viene inzializzato con l'oggetto MCvSURFParams dove è indicata la soglia di keypoint di uguaglianza tra le due immagini, necessari per decretare che si tratta dello stesso soggetto. In ogni confronto, vengono creati due oggetti ImageFeature per le due immagini da campionare a cui vengono poi applicati i metodi dell'oggetto Features2DTracker. Infine ho usato l'oggetto HomographyMatrix per determinare il piano prospettico del volto cercato nell'immagine analizzata, questo oggetto viene definito solo se l'algoritmo riesce a definire una corrispondenza. Il resto del codice serve a concatenare le due immagini in una, unire i punti di corrispondenza delle due immagini ed evidenziare il piano della HomographyMatrix.

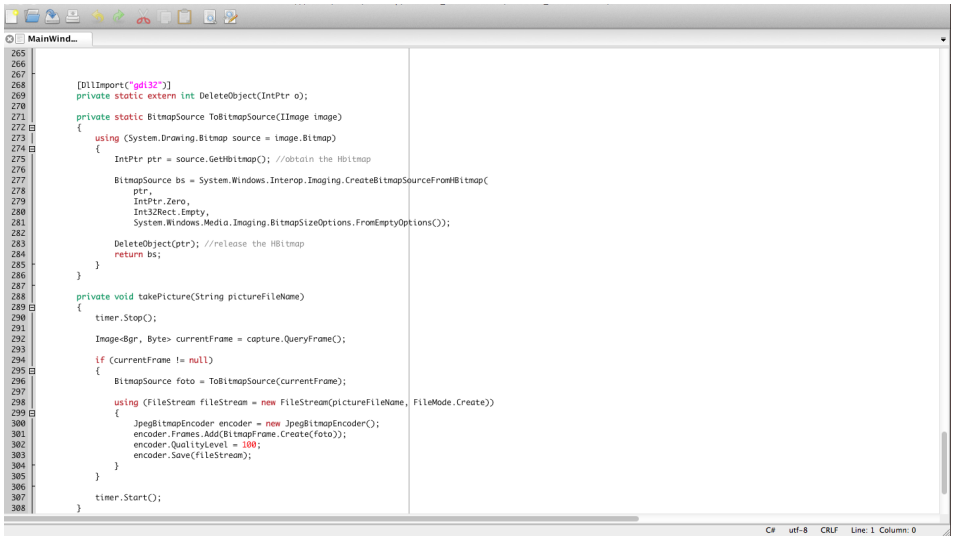


Quinta Parte



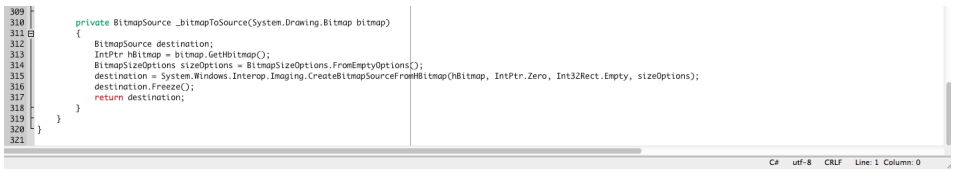
```
            System.IO.File.Delete("foto.jpg");
        }

        void timer_Tick(object sender, EventArgs e)
        {
            Image<Bgr, Byte> currentFrame = capture.QueryFrame();

            if (currentFrame != null)
            {
                String Coordinate = "";
                int counterFace = 0;
                Image<Gray, Byte> grayFrame = currentFrame.Convert<Gray, Byte>();

                var detectedFaces = haarCascade.Detect(grayFrame);

                foreach (var face in detectedFaces)
                {
                    int xCenter = face.rect.X + (face.rect.Width / 2);
                    int yCenter = face.rect.Y + (face.rect.Height / 2);
                    Coordinate += String.Format("Face {0} - X:{1},Y:{2} - ({3})\r\n", counterFace, xCenter, yCenter, face.neighbors);
                    // Filtro per non vedere quadrati di facce sovrapposte
                    if (face.neighbors > 20)
                    {
                        // Rettangolo che evidenzia il volto
                        currentFrame.Draw(face.rect, new Bgr(0, counterFace * 100, 255), 3);
                        // Crocetta al centro del rettangolo
                        currentFrame.Draw(new CrossZDF(new System.Drawing.PointF(xCenter, yCenter), 10, 10), new Bgr(System.Drawing.Color.DarkGray), 1);
                        // Aggiunto per scattare la foto solo se ha inquadrato un volto e per recuperare solo la porzione del volto
                        if (scatta)
                        {
                            Bitmap pictureFace = currentFrame.ToBitmap().Clone(new Rectangle(face.rect.X, face.rect.Y, face.rect.Width, face.rect.Height), currentFrame.ToBitmap().PixelFormat
                            pictureFace.Save(FOTO_FILE);
                            image3.Source = _bitmapToSource(pictureFace);
                            scatta = false;
                            txtbMessaggi.Text = "Foto scattata.";
                        }
                    }
                    counterFace++;
                    //oldFace = face.neighbors;
                }
                image1.Source = ToBitmapSource(currentFrame);
                txtbCoordinate.Text = Risoluzione + Coordinate;
            }
        }
```

Sesta Parte



```
        [DllImport("gdi32")]
        private static extern int DeleteObject(IntPtr o);

        private static BitmapSource ToBitmapSource(IImage image)
        {
            using (System.Drawing.Bitmap source = image.Bitmap)
            {
                IntPtr ptr = source.GetHbitmap(); //obtain the Hbitmap

                BitmapSource bs = System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap(
                    ptr,
                    IntPtr.Zero,
                    Int32Rect.Empty,
                    System.Windows.Media.Imaging.BitmapSizeOptions.FromEmptyOptions());

                DeleteObject(ptr); //release the HBitmap
                return bs;
            }
        }

        private void takePicture(String pictureFileName)
        {
            timer.Stop();

            Image<Bgr, Byte> currentFrame = capture.QueryFrame();

            if (currentFrame != null)
            {
                BitmapSource foto = ToBitmapSource(currentFrame);

                using (FileStream fileStream = new FileStream(pictureFileName, FileMode.Create))
                {
                    JpegBitmapEncoder encoder = new JpegBitmapEncoder();
                    encoder.Frames.Add(BitmapFrame.Create(foto));
                    encoder.QualityLevel = 100;
                    encoder.Save(fileStream);
                }
            }

            timer.Start();
        }
```

Settima Parte



```
        private BitmapSource _bitmapToSource(System.Drawing.Bitmap bitmap)
        {
            BitmapSource destination;
            IntPtr hBitmap = bitmap.GetHbitmap();
            BitmapSizeOptions sizeOptions = BitmapSizeOptions.FromEmptyOptions();
            destination = System.Windows.Interop.Imaging.CreateBitmapSourceFromHBitmap(hBitmap, IntPtr.Zero, Int32Rect.Empty, sizeOptions);
            destination.Freeze();
            return destination;
        }
    }
}
```

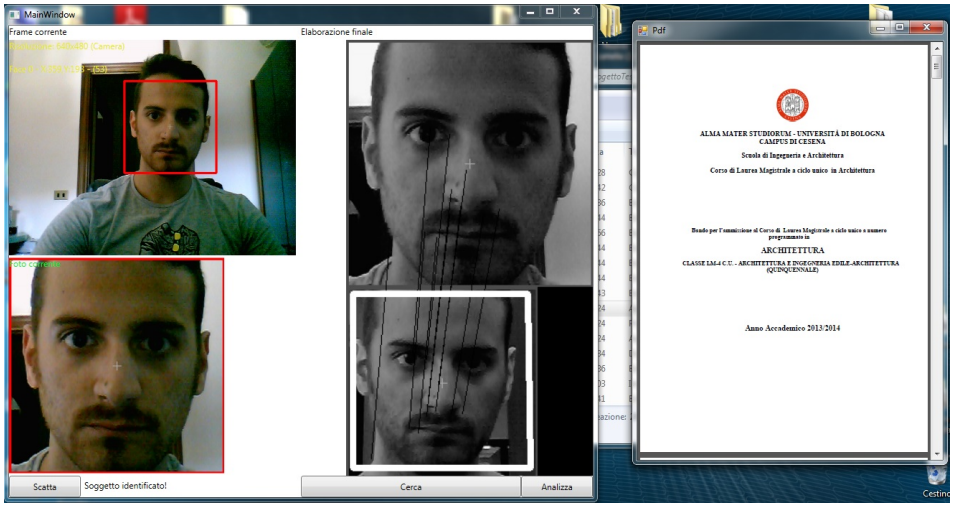
Ottava Parte

La nona parte fa riferimento ad un basilare visualizzatore di pdf che si limita a ricevere in ingresso un identificatore relativo alla foto riscontrata e visualizza il file associato.

Pdf.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace EMGUCV_FaceDetection
{
    public partial class Pdf : Form
    {
        private string infoAR;

        public Pdf(String infoAR)
        {
            this.infoAR = infoAR;
            InitializeComponent();
            Form_Loaded();
        }

        public void Form_Loaded()
        {
            axAcroPDF1.LoadFile(infoAR);
        }
    }
}
```

Nona Parte

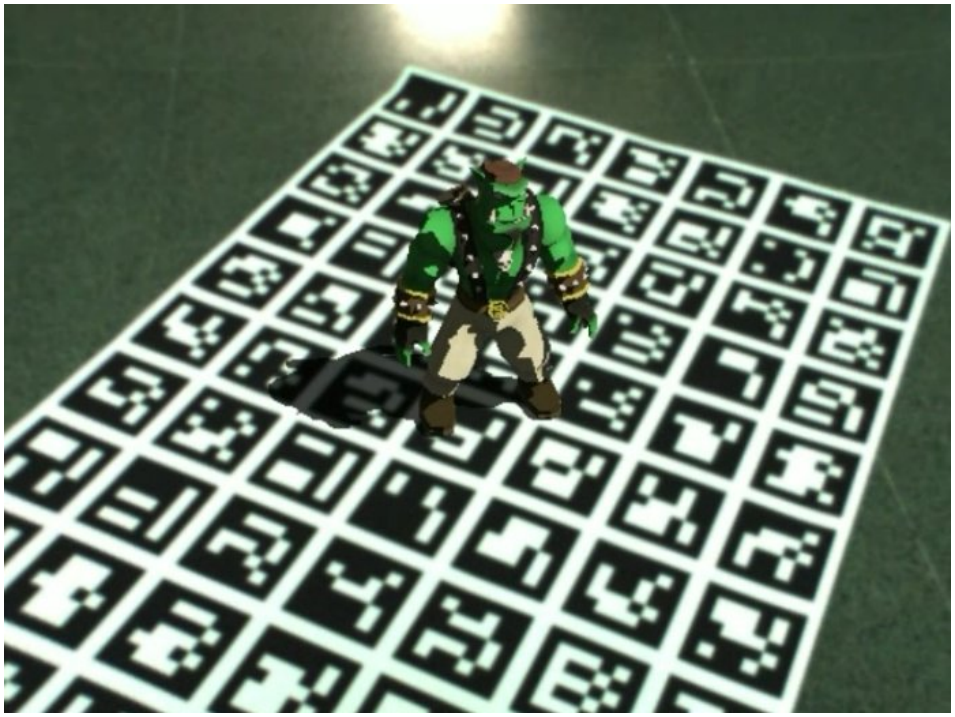


Risultato Finale

Nel caso il soggetto non venga riconosciuto l'immagine sulla sinistra sarà l'istantanea scattata precedentemente (in scala di grigio) e un divieto, messo per simboleggiare il mancato matching.
Tra tutte le immagini nella cartella 'foto' può essere che ce ne siano più di una per lo stesso soggetto, a questo proposito è stata inserita un'apposita variabile che effettui questo controllo e visualizzi solo l'immagine in cui sono stati trovate la maggiori corrispondenze.

2.5 ArUco: un caso pratico

ArUco è una libreria che utilizza le OpenCV per creare dei software per AR. Creata da un gruppo di professori dell'università di Córdoba, chiamato AVA (Aplicaciones de la Visión Artificial) e fondato nel 1998, questa libreria di distinge dai concorrenti per semplicità di utilizzo e risparmio in termini di linee di codice da scrivere per creare un programma. Un semplice programma che dimostra la potenzialità di questa libreria è quello illustrato nella figura sottostante.



Orco in AR

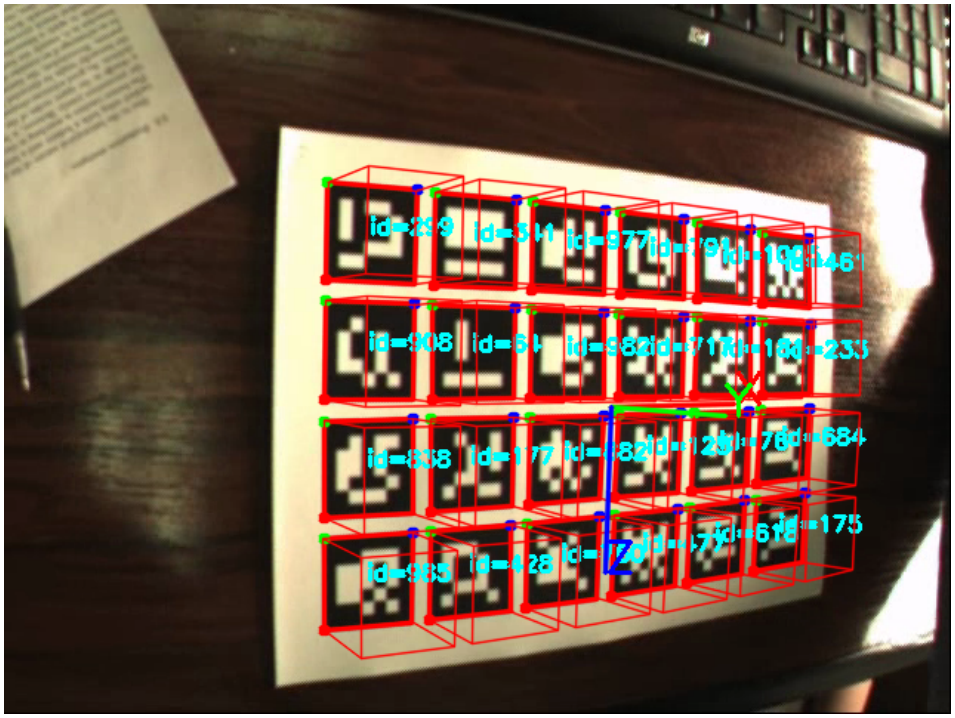
L'orco rappresentato non è presente nella realtà ma è aggiunto grazie all'utilizzo delle librerie ArUco e quindi alle OpenCV. Per rendere un'idea di quanto ben implementate siano queste librerie riporto qui di seguito un esempio:

Queste linee di codice:

```
int main(int argc,char **argv)
{
	try
	{
		CameraParameters CamParam;//Intrinsics and Distorsion Paramters
		MarkerDetector MDetector; //The Detector of Makers
		vector<Marker> Markers;//The set of Markers detected in the image
		cv::Mat InImage=cv::imread(argv[1]);//read image
		CamParam.readFromXMLFile(argv[2]);//read the Camera Parameters
		CamParam.resize( InImage.size());//resizes the parameters
		float MarkerSize=atof(argv[3]);//size of the marker(use your measurer)
		//Detection
		MDetector.detect(InImage,Markers,CamParam,MarkerSize);
		//For each marker
		for(unsigned int i=0;i<Markers.size();i++){
			cout<<Markers[i]<<endl;//print info
			//drow the marker border and id
			Markers[i].draw(InImage,Scalar(0,0,255),2);
			//draw a 3d cube in each marker
			CvDrawingUtils::draw3dCube(InImage,Markers[i],CamParam);
		}
		//show input with augmented information
		cv::namedWindow("in",1);//
		cv::imshow("in",InImage);
		cv::waitKey(0);//wait for key to be pressed
	}catch(std::exception &ex)
	{
		cout<<"Exception :"<<ex.what()<<endl;
	}
}
```

Esempio codice ArUco

Sono l'essenziale per creare un programma che interagisce con la realtà in questo modo:



Risultatao della compilazione del codice soprastante

Le principali feature di ArUco sono:

* *Individuare i marcatori con una singola linea di codice C++*
* *Individuazione dei bordi di un AR(Marcatori composti da diversi marcatori)*
* *Unico Requisito è OpenCv (¿=2.1)*
* *1024 diversi marcatori*
* *integrazione incrociata con OpenGL e OGRE*
* *Veloce, affidabile e multi-piattaforma visto che si basa su OpenCV*
* *Contiene esempi che aiutano chiunque ad eseguire un'applicazione AR in meno di 5 minuti*
* *Licenza BSD*

2.5.1 Calibrazione Camera

La calibrazione della fotocamera è quel processo che si effettua per ottenere i parametri fondamentali della camera. Questi aprametri permettono di determinare dove si trova un determinato punto 3D nello spazio catturato dai sensori della fotocamera. I parametri si dividono in *intrinseci* e *estrinseci*.

I parametri intrinseci sono:

* *fx,fy:*Lunghezza focale delle lenti della fotocamera su entrabi gli assi. Di solito sono espressi in pixel.

* *cx,cy:*Centro ottico del sensore (espresso in pixel).

* *k1,k2,p1,p2:*Coefficienti di distorsione.

In una camera ideale, un punto 3D (X,Y,Z) nello spazio si proietta nel pixel come:

x= (X fx /Z) +cx ; y= (Y fy/Z)+cy.

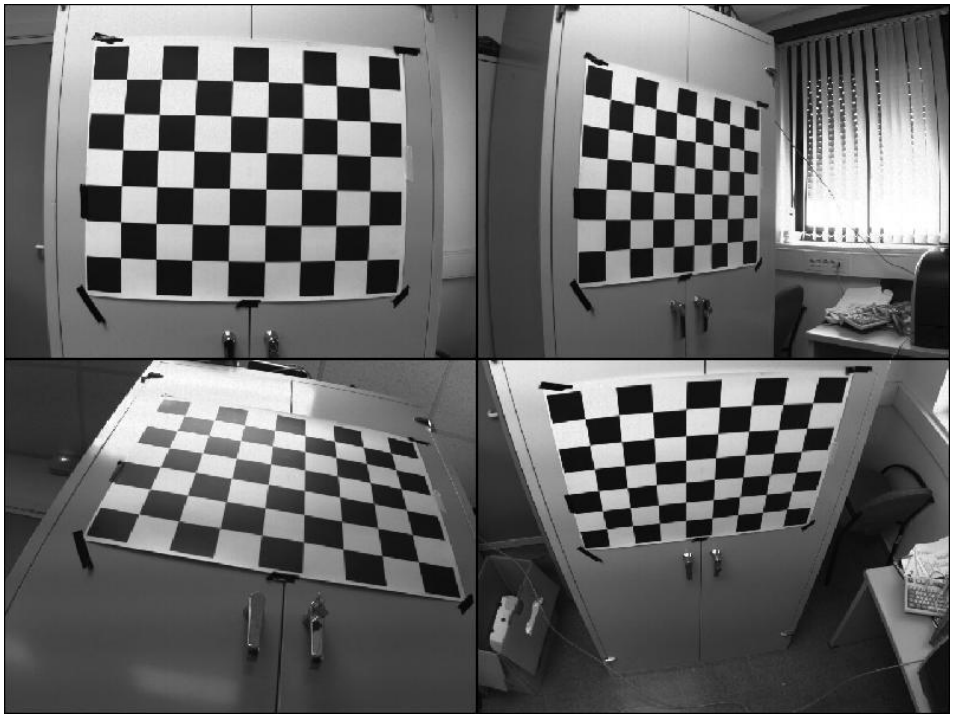
Nonostante ciò, normalmente le lenti delle camere distorcono la scena rendendo i punti lontani dal centro ancora più lontani. Così, le strisce verticali in prossimità dei bordi dell'immagine appaiono leggermente piegate. Di conseguenza, se vogliamo conoscere la proiezione di un pixel, dobbiamo considerare le componenti di distorsione. Ci limitiamo a dire che ci sono due tipi di distorsioni (radiale e tangenziale) e questi sono rappresentati dal paremeters p1, p2, k1, k2.

Distorsione senza calibrazione

Quanto sopra presuppone che si conosca la posizione 3D di un punto in relazione al sistema di riferimento della telecamera. Se si vuole sapere la proiezione di un punto in un sistema di riferimento arbitrario, allora è necessario utilizzare parametri estrinseci. I parametri estrinseci sono fondamentalmente le rotazioni 3D (Rvec = Rx, Ry, Rz) e traslazioni 3D (TVEC = Tx, Ty, Tz) necessari per tradurre il sistema di riferimento telecamera in uno arbitrario.

2.5.2 Calibrazione in OpenCV

La calibrazione è il processo che mira all'ottenimento the paramentri intrinseci della fotocamera e OpenCV permette di farlo facilmente. L'unica operazione da fare è catturare un'immagine di un pannello colorato come una scacchiera di dimensioni note.



Calibrazione

Si dovrebbero prendere almeno 5 immagini diverse. Una frontale e quattro in cui ogni volta un bordo diverso della scacchiera è vicino al bordo dell'immagine. Ponendo il pattern vicino al bordo dell'immagine sarà possibile stimare con precisione la distorsione della fotocamera. Una volta che abbiamo preso la foto con la fotocamera che si desidera utilizzare, utilizzare l'applicazione 'samples/cpp/calibration.cpp' fornita in OpenCV(>= 2.2) per calibrare la fotocamera. Dovremmo indicare il numero di angoli del modello in entrambi gli assi, e la dimensione reale del quadrato. Come output, il programma genera un file .yml che può essere usato in ArUco. Nel processo di ottenimento dei parametri intrinseci, anche OpenCV ottiene dei valori estrinseci dalla fotocamera per ognuna delle posizioni del pattern.

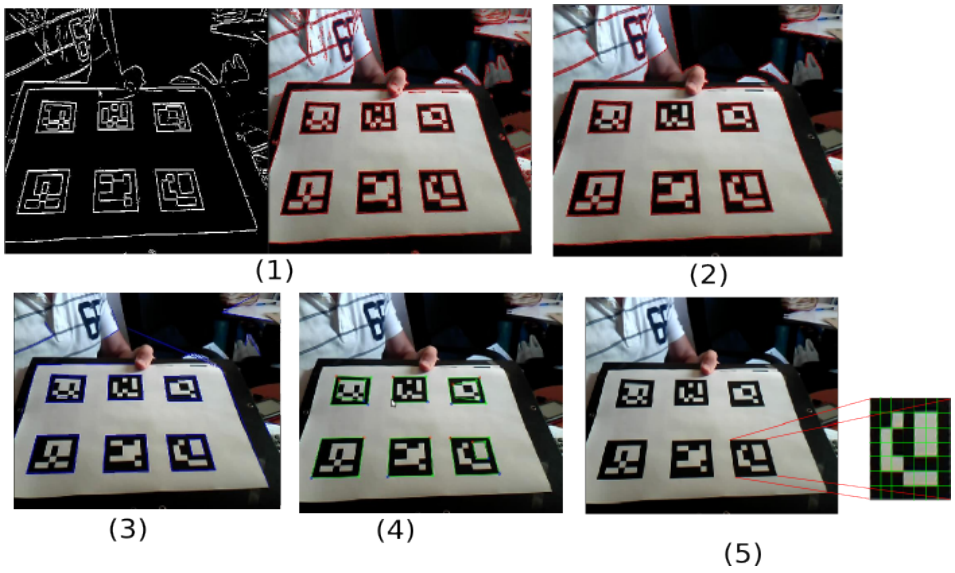
2.5.3 Come Rendere la Realtà Aumentata?

Aumentare la realtà è il processo di aggiunta di informazioni virtuali alle immagini. Per farlo, abbiamo bisogno di sapere dove dipingere

l'informazione virtuale (normalmente utilizzando OpenGL o un altro motore 3D). Questo è il punto in cui i marcatori ci aiuteranno. Un Marker AR, come quello utilizzato in ArUco, è un elemento molto distintivo che può essere facilmente rilevato. L'indicatore sarà utilizzato per calcolare i parametri estrinseci della fotocamera in relazione al marcatore in modo che sarete in grado di eseguire il rendering in 3D sapere dove è il centro (punto di coordiate 0,0,0) del sistema di riferimento globale. Dal momento che il marcatore AR è molto facile da rilevare, è possibile elaborare le immagini e fare il rendering in tempo reale.

2.5.4 Processo di Identificazion in ArUco

L'identificazione del marker in ArUco procede come di seguito:



Identificazione Marker

* Applichiamo una soglia di di adattamento in modo da ottenerei bordi (Figura 1)

* Troviamo i margini. Dopo, vengono trovati anche margini indesiderati oltre a i margini reali che vogliamo utilizzare. Il resto del processo mira ad eliminare questi ulteriori margini indesiderati.

    * Rimuoviamo i bordi composti da un numero troppo piccolo di punti (Figura 2)
    * Approssimazione dei poligoni tenendo solamente le figura con esattamente 4 angoli (per esempio, dei rettangoli) (Figura 3)

* Organizzare i angoli in ordine antiorario.

* Rimuovere i rettangoli troppo vicini. Step necessario in quanto riconosciamo anche i singoli rettangoli contenuti all'interno di un marker. A qeusto punto abbiamo anche il bordo esterno. (Figura 4)

* Identificare i marker

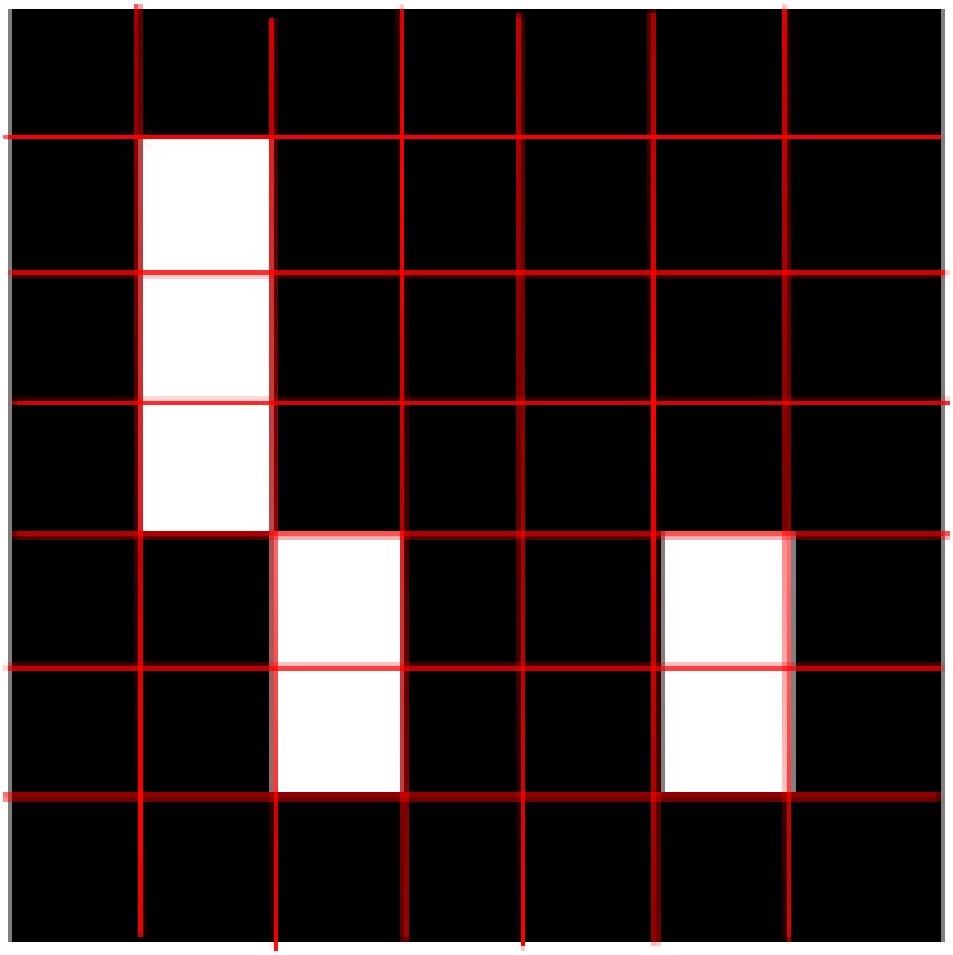
    * Rimuovere la proiezione dovuta alla prospettiva così da ottenere una visione frontale. (Figura 5)
    * Calcolare le soglie utilizzando il metodo Otsu. L'algoritmo Otsu presume che nell'immagine da sogliare siano presenti due sole classi e quindi calcola la soglia ottima per separare queste due classi minimizzando la varianza intra classe.
    * Identificazione del codice interno. Se è un marker, allora ha un codice interno. Il marker è diviso in una griglia 7x7 dove solo la griglia interna 5x5 contiene l'informazione. Le celle mancanti corrispondono al bordo nero esterno. Quindi, prima di tutto controlliamo che ci sia effettivamente un bordo nero. Dopodichè leggiamo la griglia 5x5 interna e vediamo se contiene un'informazione che possiamo considerare valida (leggere correttamente il codice potrebbe richiedere una rotazione)

* Per i marker validi, perfezioniamo gli angoli utilizzando l'interpolazione dei subpixel

* Infine, se vengono forniti i parametri intrinseci della fotocamera, vengono calcolati quelli estrinseci dei marcatori.

2.5.5 Codificazione dei Marker

Ogni marker ha un codice interno dato da 5 parole di 5 bit l'una. La codifica utilizzata è una versione leggermente modificata del codice Hamming. In totale, ogni parola ha solo 2 bit di informazione dei 5 utilizzati. Gli altri 3 sono utilizzati per l'individuazione di errori. Di conseguenza, possiamo avere fino a 1024 id diversi. La caratteristica principale che differenzia la codifica dal codice Humming originale stà nel fatto che il primo bit (bit di parità del bit 3 e 5) è invertito. Cioè, l'id 0 (che nel codice Humming verrebbe identificato come 00000) diventa 10000 con questa codifica. L'idea alla base di ciò è evitare che un rettangolo completamente nero diventi un marker con un id valido e l'obiettivo di ridurre la probabilità di falsi positivi con oggetti nell'ambiete.



Marker

Bibliografia

[1] Gary Bradski and Adrian Kaehler: *Learnin OpenCV: Computer Vision with the OpenCV Library*, O'Reilly

[2] Robert Laganiere: *OpenCv 2 Computer Vision, Application Programming Cookbook*, PACKT Publishing

[3] sito web WebGL: *http://www.chromeexperiments.com/about/*

[4] sito web ArUco: *http://www.uco.es/investiga/grupos/ava/node/26*

[5] sito web OpenCV: *http://opencv.willowgarage.com/wiki/*

[6] Ronald T. Azuma *A Survey of Augmented Reality*

Ringraziamenti

In conclusione di questa tesi volevo ringraziare la facoltá che ha reso possibile la mia formazione e il raggiungimento del titolo finale, tutti i professori e gli impiegati universitari che rendono viva quella struttura che é la Seconda Facoltá di Ingegneria e che permettono a tanti, come me, di realizzare le loro ambizioni e dare un valore aggiunto al loro curriculum, spendibile con orgoglio nel mondo del lavoro. E per ultimi, ma non per importanza, una ringraziamento profondo a tutti coloro che hanno creduto in me, che nonostante le difficoltá incontrate mi sono sempre stati vicino, hanno sempre fatto il tifo per me e mi hanno dato la forza di completare questo difficile percorso.