Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

APPLICAZIONI DI REALTÀ AUMENTATA IN AMBITO MUSEALE

Relatore:
Chiar.mo Prof.
ALESSANDRO AMOROSO

Presentata da:
ANDREA IANNÌ

II sessione
2014-2015

Alla mia famiglia
e a tutti coloro che hanno creduto in me...

Introduzione

La realtà aumentata, unitamente a quella mista, stanno rapidamente prendendo pieno all'interno di molti aspetti della vita umana. Scopo di questo lavoro è di analizzare tecnologie e tecniche esistenti al fine di applicarle ad un caso reale, la rilevazione e la sovrapposizione di un oggetto digitale tridimensionale ad uno presente in un museo.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione e definizione del problema

Questo capitolo rappresenta un'introduzione alle tecnologie che sono state utilizzate e implementate all'interno del lavoro descritto in questo documento.

1.1 Vari tipi di realtà

Con il termine realtà fisica si può definire l'insieme di tutte le cose reali che ci circondano[1], come ad esempio gli oggetti presenti su una scrivania o tutto ciò che è percepibile usando i cinque sensi che ha l'essere umano.
Le nuove tecnologie hanno introdotto altri concetti di realtà [1]:

1. realtà aumentata;
2. realtà mista;
3. realtà virtuale.

Agli estremi si trovano due entità differenti tra loro, mentre quella al centro rappresenta una loro fusione. Nella Figura 1.1 è mostrato un esempio dei vari tipi di realtà.

[1]Da wikipedia: https://it.wikipedia.org/wiki/Realt%C3%A0

Figura 1.1: Confronto fra i vari tipi di realtà.
a) Scena percepita dagli occhi; b) realtà aumentata; c) realtà mista; d) realtà virtuale.

Fonte immagini http://wallpapersphotography.com, http://www.flipkart.com, https://www.cgstud.io e http://www.clipartbest.com.

1.1.1 Realtà virtuale

La realtà virtuale è composta da un ambiente tridimensionale totalmente generato da un computer. In questo caso tutti gli oggetti, le forze con le quali interagiscono e il mondo in cui si trovano sono gestiti da una macchina e mostrati tramite appositi device.
Tali dispositivi possono fornire due tipi di esperienza. La prima, detta non immersiva o classica, prevede l'uso di un display che fa da tramite fra il mondo fisico e quello digitale. Rientrano in questa categoria i classici videogiochi tridimensionali, dove ci trova ad esplorare un mondo 3D riuscendo a vederlo come se si guardasse sempre attraverso una finestra (il display) che viene via via aggiornata in base agli spostamenti del protagonista. Questa tipologia non permette un'interazione naturale con l'ambiente ed è facile capire che ciò che si sta guardando non fa parte della realtà in cui ci si trova.
Il secondo tipo di realtà virtuale è invece definito immersivo. Nonostante esista da parecchio tempo, questa tipologia di realtà virtuale sta ritornando in auge negli ultimi anni, poiché hanno iniziato a diffondersi dispositivi indossabili che consentono di "immergersi" all'interno di questi mondi virtuali. Per far ciò vengono usate diverse tecniche, come ad esempio l'Oculus rift[2] che

[2]https://www.oculus.com/en-us/rift/

mostra delle immagini stereoscopiche tramite l'uso di un casco (HMD), facendo in modo che la telecamera segua i movimenti della testa di chi utilizza il sistema. Da qui sono nati problemi dovuti all'interazione con questi ambienti, un ostacolo che è stato riscontrato è che ad un progresso di tale entità non si associa un adeguato aggiornamento e perfezionamento riguardo alle metodologie di interazione con oggetti 3D, dell'utilizzo di interfacce grafiche e del generale design ambientale, con la diretta conseguenza di indebolire o addirittura annullare l'effetto presenza dell'HMD. Questo genere di problematiche però non verranno approfonditi in quanto non rientrano all'interno degli argomenti qui trattati.

1.1.2 Realtà aumentata

Per realtà aumentata si intende un arricchimento della percezione sensoriale tramite l'utilizzo di dispositivi specifici, che possono essere display indossabili o semplici smartphone. Il concetto chiave di questa tecnologia è il fornire qualcosa in più a ciò che si ha davanti nel mondo reale, e non sostituirlo con altro.
Rientrano in questo ambito tutte quelle tecnologie che attraverso di fotocamere o altri sensori aumentano o intensificano la percezione sensoriale. Quindi anche l'aggiunta di segnali audio o informazioni sul luogo in cui ci si trova basate sul GPS.

1.1.3 Realtà mista

La realtà virtuale e quella aumentata, nonostante a primo impatto diano l'impressione di essere due entità opposte, sono in realtà due concetti separati e distinti, i quali possono coesistere nello stesso momento [2]. La realtà mista può essere vista come un continuo, in cui la realtà virtuale e quella aumentata si trovano adiacenti e dove uno integra l'altro.

Lo scopo è quello di inserire oggetti digitali all'interno delle realtà percepita, integrandoli con essa. Quando si disegna un sistema di questo genere devono essere presi in considerazione tre aspetti:

1. come combinare il mondo virtuale con quello reale;
2. come interagire in tempo reale;
3. come posizionare gli oggetti digitali nell'ambiente tridimensionale fisico.

Oltre questi tre punti esiste anche un altro elemento: la mobilità. Potrebbe essere previsto infatti che chi utilizza il sistema abbia la possibilità di muoversi all'interno di un ambiente più o meno grande. Questo genere di applicazioni fa aumentare la complessità del problema di posizionamento 3D, in quando è possibile vedere la stessa scena da angolazioni differenti.

1.2 Componenti

Scopo di questa tesi è quello di ottenere un sistema che utilizza la realtà mista con l'obiettivo di aggiungere oggetti tridimensionali ad una scena, prevedendo anche una componente di mobilità, cioè deve essere consentito di poter girare intorno all'oggetto in questione.
Per semplificazione un sistema del genere può essere visto come l'unione di componenti differenti, un dispositivo che possa generare la scena mostrare, un sistema in grado di definire il punto in cui vanno disegnati gli oggetti digitali e qualcosa che visualizzi il risultato finale.

1.2.1 Generatore della scena

È il dispositivo o software che ha il compito di fare il rendering della scena. Non rappresenta un problema, in quanto spesso è necessario caricare solo qualche oggetto tridimensionale di piccole dimensioni, quindi è un compito che può essere svolto dalla maggior parte dei dispositivi.

1.2.2 Sistema di tracciamento

È una parte chiave del sistema, infatti serve a stabilire dove devono essere visualizzati gli oggetti digitali e per mantenere l'illusione che siano perfettamente in linea con quelli reali. Se ciò non avvenisse si perderebbe la sensazione che i due mondi siano collegati tra loro.

1.2.3 Display

Rappresenta la componente in cui viene mostrata la scena, cioè dove la realtà si fonde col mondo virtuale. Esistono dispositivi indossabili come gli HMD. Ma il sistema potrebbe anche funzionare su un semplice computer o smartphone dotato di videocamera, aumentando la realtà attraverso di esso.

Display ottici trasparenti

Sono rappresentati da quella classe di display che permettono di vedere attraverso di essi. Grazie ai quali è possibile avere una visione diretta dell'ambiente, ma allo stesso tempo viene sovrapposta un'immagine virtuale, come mostrato in Figura 1.2.

Questo tipo di display richiedono una doppia calibrazione, cioè una rispetto

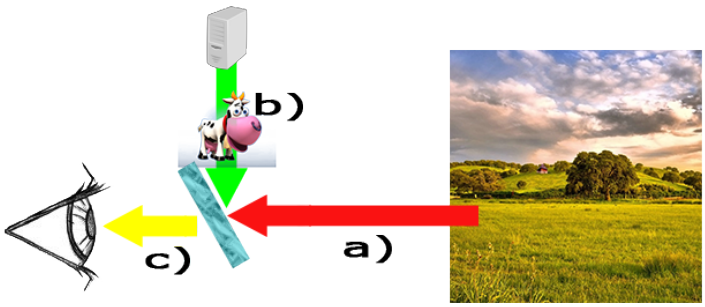

Figura 1.2: Funzionamento di un display trasparente, la luce (a), prima di arrivare all'occhio (c), passa per un display trasparente nel quale viene sovrapposto l'oggetto tridimensionale digitale (b).

agli oggetti reali della scena e una rispetto al punto di vista di chi la usa,

poiché l'immagine finale deve risultare correttamente sovrapposta rispetto la posizione dalla quale viene osservata.
Esistono dispositivi che montano display trasparenti di questo genere, il loro scopo è quello di visualizzare oggetti digitali all'interno di un contesto reale. Un esempio è dato dal Microsoft Hololens, mostrato in Figura 1.3.
Questo dispositivo è dotato di diversi sensori come fotocamera ed infrarossi, e grazie ad essi è in grado di rilevare come è composto l'ambiente ed aggiungere degli elementi virtuali. Ha a disposizione anche un sistema ottico trasparente che permette di sovrapporre gli oggetti digitali a ciò che si ha di fronte. Rientra nella categoria degli HMD e ha lo scopo di fornire l'illusione che all'interno di una stanza ci sia più di ciò che gli occhi percepiscono (Figura 1.4).

Figura 1.3: Il dispositivo hololens (a) con un dettaglio dei sensori (c) e del sistema ottico (b).

Fonte immagine http://www.microsoft.com/microsoft-hololens.

Video mostrato attraverso un display

Una tecnica molto popolare è quella di acquisire un'immagine tramite l'utilizzo di un fotocamera e di mostrarla, dopo aver aggiunto gli oggetti digitali, su un comune display, come in Figura 1.5. Un esempio potrebbe essere

quello di associare ad un HMD di realtà virtuale due fotocamere posizionate all'altezza degli occhi, in modo da fornire un'immagine stereoscopica.

La popolarità di questa tecnica non è dovuta a dispositivi indossabili, ma

Figura 1.4: Un esempio del sistema olografico di hololens, grazie ai sensori che ha a disposizione è stato possibile posizione un video sulla parete, un meteo tridimensionale sul ripiano e altri oggetti digitali all'interno della stanza.

Fonte immagine http://www.microsoft.com/microsoft-hololens.

ai comuni smartphone, i quali, grazie alla fotocamera di cui sono dotati, possono fare da tramite fra il mondo reale e quello virtuale-misto, naturalmente consentendo di avere un unico piano focale.

Stanno però iniziando a diffondersi dispositivi dotati della tecnologia Intel realsense, che è in grado di rilevare la profondità dell'ambiente e le dimensioni degli oggetti sfruttando due videocamere e un sensore di profondità (Figura 1.6). Grazie a questa tecnologia si potrebbe ottenere un maggiore livello di precisione rispetto ad una comune fotocamera. Esistono anche altri dispositivi di questo tipo, ma realsense è già presente in alcuni sistemi notebook e desktop, e si prevede che presto sarà inserito anche nei tablet pc.

Proiezione diretta

Una realtà mista può essere ottenuta anche proiettando delle immagini direttamente sulla scena come in Figura 1.7. Questo consente di integrare direttamente la parte virtuale con l'ambiente reale, ma dipende anche dalle condizioni di luce e dalle superfici sulle quali si proietta. Inoltre resta sempre il problema della messa a fuoco su piani non omogenei.

Figura 1.5: Il video viene acquisito tramite la fotocamera del dispositivo (a) e nel display viene mostrata l'immagine acquisita con l'oggetto 3D sovrapposto (b)

Figura 1.6: Dispositivo Intel realsense, è dotato di più fotocamere e di un sensore di profondità, grazie ai quali è in grado i mappare gli oggetti che ha di fronte.

Fonte immagine http://windows.hdblog.it.

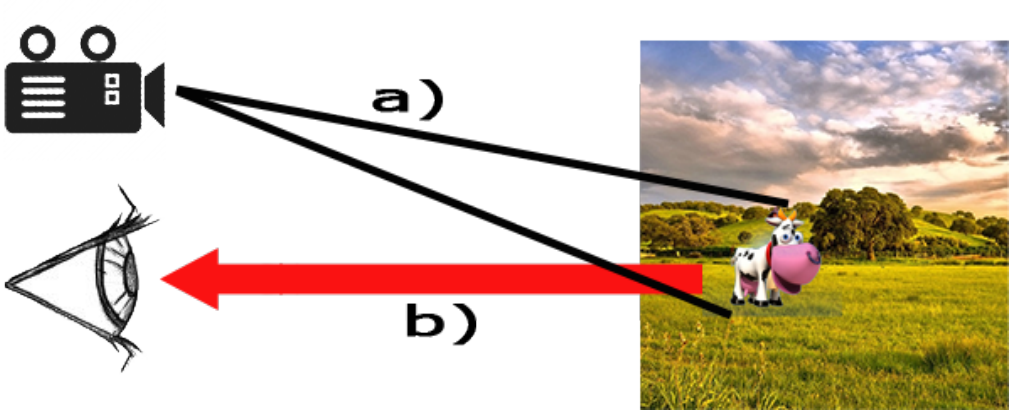

Figura 1.7: Viene proiettata direttamente un'immagine nella scena da un proiettore esterno (a), questo consente di vedere tutti gli elementi già composti (b).

1.3 Lavori correlati

Per poter posizionare un'oggetto digitale all'interno di un ambiente fisico, e soprattutto fare in modo che interagisca con un altro realmente presente è necessario riconoscere quest'ultimo e avere un sistema di posizionamento coerente col mondo esterno. Esistono diverse tecniche per rilevare un oggetto fisico in un'ambiente tridimensionale, la più semplice è quella di associare ad ognuno di essi un marker, che possono anche essere creati dinamicamente, come nel caso del museo di storia naturale di Londra [3], dove è stata creata una sala apposita dove sono presenti una serie di led che fanno da marker, il problema di questo genere di tecniche è che necessitano di modiche all'ambiente, come l'introduzione dei marker o dei led, e, in questo caso specifico, di un hardware specifico, che viene distribuito ai visitatori, per visualizzare gli oggetti tridimensionali.
Dal museo di Londra[3] arriva un altro caso di utilizzo della realtà aumentata in ambito culturale, in questo caso è stato ideato una sorta di museo elettronico sparso per tutta la città. Ogni volta che il GPS rileva che ci trova in una posizione specifica, viene proposta una foto di quel punto nel passato, facente parte dell'archivio storico di Londra, i fruitori possono quindi leggere una breve descrizione allegata all'immagine, oppure sovrapporre la scena a quella acquisita dalla fotocamera, così da poterne valutare le differenze, come mostrato in Figura 1.8.
Un caso analogo è stato implementato a Firenze, dove, sempre basandosi sul GPS, vengono mostrati i dipinti che rappresentano ciò che si sta guardando, permettendo di vedere come è cambiata la città o come l'artista l'ha interpretata.

Un primo tentativo di usare i marker in ambito museale è stato fatto da Rekimoto [4] nel 1998. Egli propose di associare ad ogni opera un marker, dal quale poi ricavarne posizione e traslazioni. Questa tecnica consente di mostrare in scena un oggetto di dimensioni fisse in una posizione tridimen-

[3]http://www.museumoflondon.org.uk/london-wall/

Figura 1.8: Applicazione Street museum del museo di Londra, nella figura viene sovrapposta una foto storica alla realtà.

Fonte immagine www.mymodernmet.com.

sionale precisa. Un metodo simile è stato utilizzato di recente dal catalogo IKEA [5], la copertina del quale può essere usata per vedere come un oggetto, in vendita presso la catena sopra citata, appaia all'interno di una stanza, come in Figura 1.9.
Come il catalogo, una qualunque immagine può essere usata come marker, operazione però semplice solo con elementi bidimensionali, come dipinti, infatti quando si tratta di oggetti a tre dimensioni, come sculture, le cose si complicano.

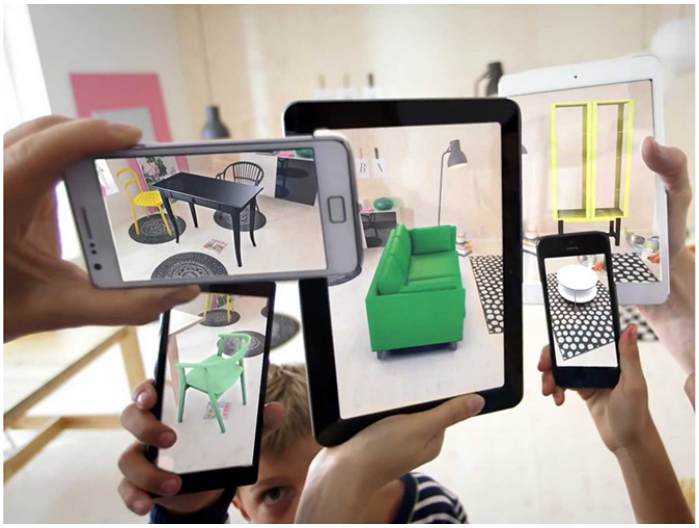

Figura 1.9: IKEA ha creato un'applicazione di realtà aumentata in cui il catalogo viene usato come marker, questo consente di avere un oggetto di cui se ne conoscono le dimensioni all'interno della stanza, permettendo di inserire rappresentazioni digitali con dimensioni reali in un ambiente fisico.

Fonte immagine www.businessinsider.com.

1.4 Scopo del lavoro e problemi correlati

L'obiettivo finale di questo lavoro è quello di sovrapporre ad un oggetto reale un corrispondente elemento tridimensionale (Figura 1.10). La controparte digitale è stata ottenuta tramite una termografia computerizzata (TAC) dell'oggetto fisico da Morigi et al. [6] e mostra come è fatto al suo interno.
Identificare un oggetto tridimensionale utilizzando una semplice fotocamera non è un'operazione banale, è possibile osservarlo da più punti di vista, che possono variare anche in altezza o distanza, oltre che in rotazione. Anche la luce può cambiare per via di fattori come l'orario, la nuvolosità, o la presenza di altri fattori.

Oltretutto sono spesso necessarie grandi risorse di memorizzazione e di calcolo per questo genere di operazioni.
Si potrebbe provare ad alleggerire il carico computazionale cercando prima

Figura 1.10: Globo di Coronelli (a), si tratta dell'oggetto reale che si vuole riconoscere. A destra (b) si trova invece la sua ricostruzione in 3D, ricavata da una TAC del globo.

Fonte immagine http://manfrediana2.racine.ra.it.

di capire da che lato si sta guardando l'oggetto, ad esempio utilizzando sensori di localizzazione, anche se il GPS non ha una precisione tale da poter essere d'aiuto in un caso indoor. Ma esistono sensori che tramite l'emissione di onde radio da punti diversi permettono ad un dispositivo di capire in che posizioni si trovi in base alla potenza del segnale ricevuto (Figura 1.11).

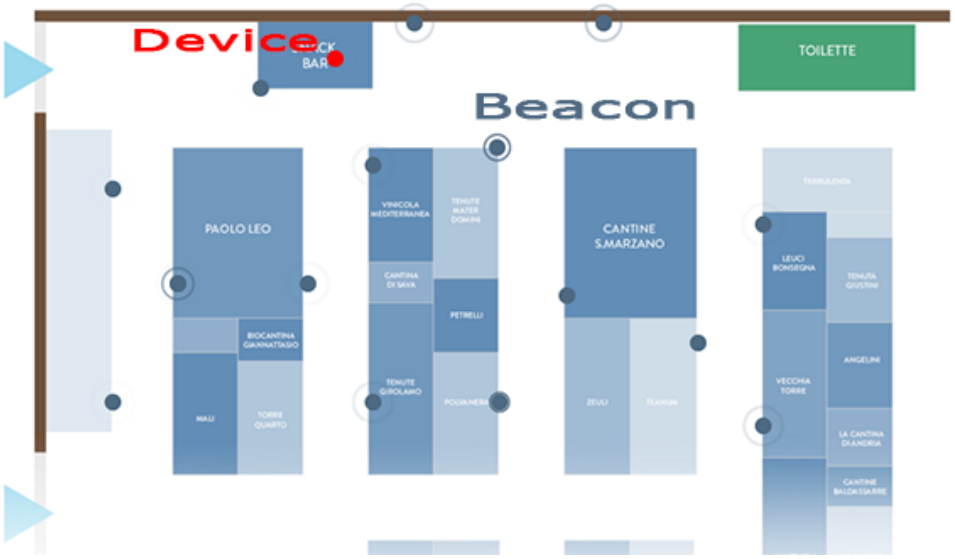

Figura 1.11: Un sistema di posizionamento e navigazione indoor basato su di un'infrastuttura di dispositivi compatibili con lo standard Apple iBeacon e compatibile con gli smartphone con supporto alla tecnologia Bluetooth 4.0LE. I beacon, in blu, sono posizionati in punti diversi dell'edificio, il dispositivo, in rosso, è in grado di stimare la sua posizione in base alla potenza del segnale ricevuta dai sensori.

Fonte immagine http://www.nextome.org.

Capitolo 2

Algoritmi per il riconoscimento degli oggetti fisici e il loro posizionamento nello spazio

In questo capitolo è stata fatta un'analisi degli algoritmi utilizzabili per il problema del rilevamento degli oggetti fisici. In particolare sono stati approfonditi algoritmi per la memorizzazione e il riconoscimento di pattern negli oggetti e per la registrazione della posizione in un ambiente tridimensionale. Per poter rilevare un oggetto è necessario conoscerne le caratteristiche estetiche, come forma e colori. Serve quindi che questi fattori siano estratti tramite un qualche algoritmo o definiti manualmente.

2.1 Hardware specializzato

I migliori risultati nel riconoscimento di oggetti 3D si ottengono usando un hardware specializzato, solo che questo obbligherebbe un museo ad acquistare e manutenere tali apparecchiature, che spesso non risultano molto economiche.
Alcune di queste tecnologie, come hololens o realsense, non sono ancora nemmeno in commercio, quindi non ancora utilizzabili. E anche avendole a dispo-

sizione richiederebbero dei cambiamenti strutturali ai luoghi dove dovranno essere utilizzate, come un stand su cui posizionarle, o addirittura la costruzione di un ambiente apposito per il loro utilizzo, come è avvenuto al museo di storia naturale di Londra, dove è stata progettata una stanza in cui sono presenti dei marker basati su dei led ed ai visitatori vengono forniti dei dispositivi appositi per poter assistere allo spettacolo tridimensionale.

2.2 Riconoscimento degli oggetti tramite le caratteristiche locali

Riconoscere un oggetto all'interno del mondo reale richiede la memorizzazione e la ricerca di caratteristiche che non siano invariabili in caso di occlusione parziale o in base a trasformazioni geometriche (rotazione, scala o traslazione).

2.2.1 SIFT

Il metodo qui descritto, chiamato "scale invariant feature transform" (SIFT) [7, 8], trasforma l'immagine di un oggetto in una collezione di vettori riferiti a caratteristiche locali, ognuno dei quali, è particolarmente robusto a cambi di illuminazione, rumore ed a variazioni del punto di vista. Inoltre queste caratteristiche sono invarianti rispetto le trasformazioni geometriche, come per esempio cambiamenti di scala o rotazioni.
L'algoritmo può essere riassunto sinteticamente in 4 fasi principali:

1. individuazione degli estremi locali (massimi o minimi) nello scale-space;
2. localizzazione dei punti chiave;
3. assegnazione di uno (o più) orientamenti canonici;
4. generazione dei descrittori.

Come primo passo l'algoritmo identifica quei punti che massimizzano e minimizzano una funzione basata su una differenza di filtri gaussiani applicati all'immagine. Ogni caratteristica viene usata in seguito per generare un vettore che identifichi la regione locale da cui è stato estratto, per ogni immagine l'algoritmo genera migliaia di queste componenti.
Le chiavi SIFT così estratte sono utilizzate per identificare in seguito il modello in questione tramite un approccio nearest-neighbor. Le chiavi potenziali sono identificate tramite una tabella hash basata sulla trasformata di Hough ed in seguito viene usato un algoritmo dei minimi quadrati per una stima finale dei parametri del modello.

Individuazione degli estremi locali

Un aspetto cruciale degli oggetti fisici è che questi possono essere visti da distanza differenti, quindi possono essere rappresentati da scale differenti. Se si vuole descrivere la struttura di una di queste entità diventa quindi cruciale poterlo su una sua rappresentazione multi-scala.
Un metodo prevede di rappresentare l'immagine come una famiglia di segnali gradualmente smussati, in modo da eliminare via via i dati appartenenti a scale differenti. Esso viene chiamato rappresentazione lineare dello scale-space.
È stato dimostrato Lindeberg [9] che il kernel gaussiano, che serve a smussare le immagini, è quello che dà i migliori risultati in un'analisi nello scale-space, cioè nel caso in cui l'immagine venga ridimensionata.
Per questo motivo è stato scelto di prendere i valori che massimizzano e minimizzano una differenza di funzioni gaussiane in tale spazio. Ciò viene fatto creando una piramide di immagini ottenute tramite convoluzione con un filtro gaussiano più volte e ricampionando ad ogni livello. Ogni volta che viene effettuata questa operazione viene restituita un'immagine di dimensioni inferiori rispetto alla precedente, andando via via a formare una piramide.
In seguito viene applicata una funziona chiamata DoG, che sta per differenza

di gaussiane, come in Figura 2.1, dove sostanzialmente viene fatta una sottrazione tra coppie di livelli adiacenti della suddetta piramide, da cui verranno estratti i punti di massimo o minimo.

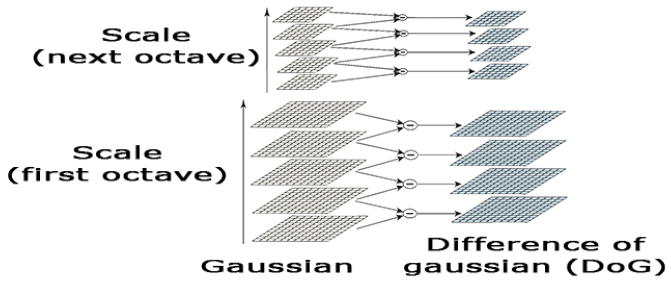

Figura 2.1: Per ogni ottava dello scale-space, l'immagine iniziale viene ripetutamente convoluta con un filtro gaussiano per produrre l'insieme di immagini a sinistra. Quelle adiacenti vengono sottratte per produrre la differenza-di-gaussiane sulla destra. Dopo ogni ottava, la figura viene sotto-campionata di un fattore uguale a 2, e il processo viene ripetuto.

Localizzazione dei punti chiave

I massimi e i minimi sono ottenuti confrontando ogni pixel della piramide con i suoi vicini allo stesso livello, che sono 8. Quando un pixel viene scelto si controlla ai livelli inferiori o superiori se quello corrispondente è ancora un massimo (o un minimo) assoluto, Figura 2.2. Questo consente di scartare la maggior parte di punti in poco tempo.
È anche consigliato applicare dei filtri come l'eliminazione di quei punti con basso contrasto per evitare bordi e linee e per garantire di mantenere solo i punti più stabili di applicare una soglia alla funzione gaussiana. Il risultato di questi due accorgimenti è mostrato in Figura 2.3.

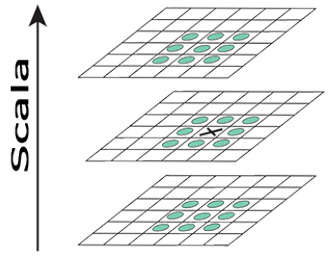

Figura 2.2: I massimi e minimi delle immagini DoG vengono rilevati confrontando un pixel (contrassegnato con X) con i suoi 26 vicini (indicati da cerchi) nelle regioni 3x3 della scala corrente e di quelle adiacenti.

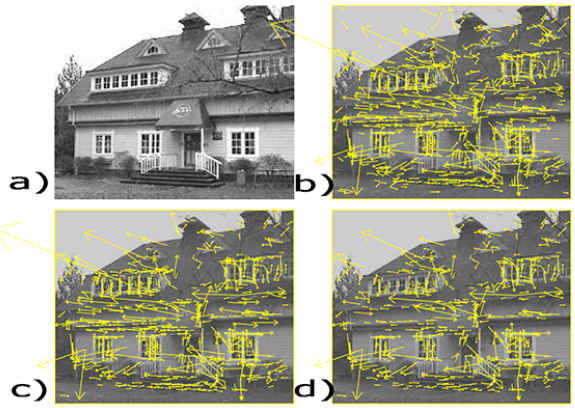

Figura 2.3: (a) Immagine originale. (b) La posizione dei primi 832 punti chiave ottenuti dalla differenza delle gaussiane. (c) Dopo aver applicato la soglia di contrasto minimo, restano 729 punti chiave. (d) Gli ultimi 536 punti chiave che rimangono dopo aver applicato la soglia alla funzione gaussiana.

Assegnazione di un orientamento canonico

Per garantire che il sistema sia robusto alla rotazione viene assegnato ad ogni punto chiave scelto un orientamento canonico, basandosi sulle sue caratteristiche intrinseche. Per ognuno di essi viene generato un istogramma basato sul gradiente di un suo intorno composto da 36 bin, uno per ogni 10 gradi, assegnando come orientamento il picco massimo, e ricordando anche

tutti quelli superiori all'80% di tale valore, un punto può quindi avere più orientazioni.

Generazione dei descrittori

Vengono infine generati dei descrittori basati sull'orientamento medio del gradiente di sotto regioni di un certo cluster di pixel rappresentate un intorno ai punti chiave, che verranno poi usati per identificare l'oggetto. La Figura 2.4 mostra la robustezza del metodo rispetto alle principali trasformazione che un'immagine può subire (rotazione, traslazione e scala).

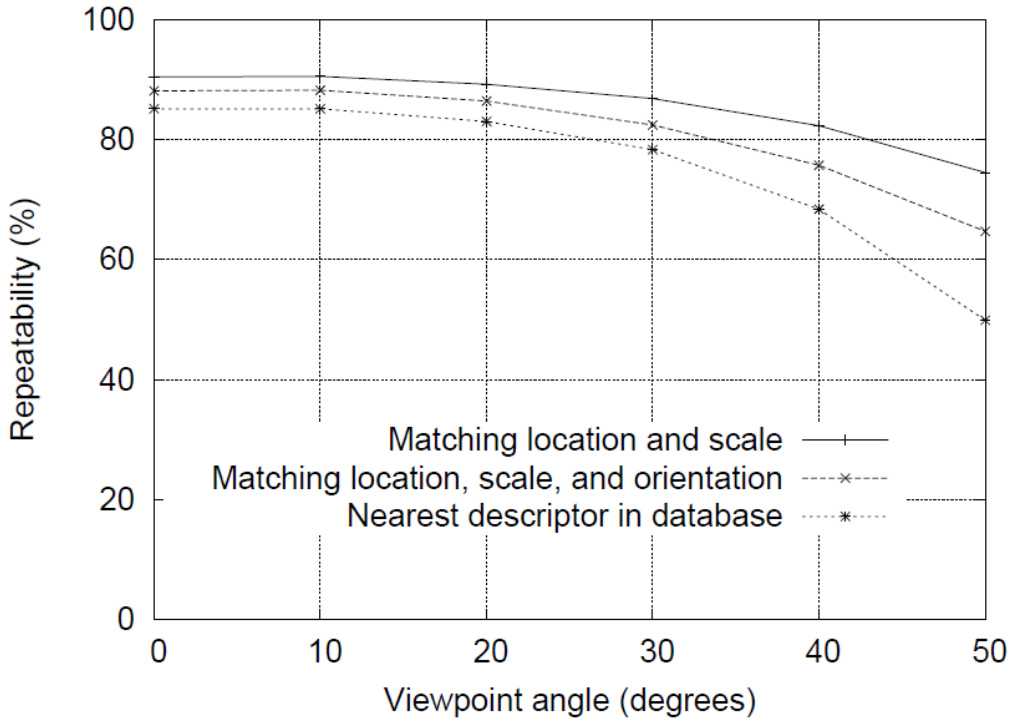

Figura 2.4: Questo grafico mostra la stabilità di ripetibilità di un punto chiave variando la posizione, l'orientamento e la corrispondenza finale in un database in funzione di una distorsione affine. Il grado di distorsione è espressa in termini di rotazione del punto di vista.

2.2.2 SURF

Ispirato a SIFT, l'algoritmo "speed up robust feature" (SURF) [10, 11] è stato progettato per l'utilizzo specifico nella computer vision, concentrandosi principalmente sulle prestazioni, in modo da poter eseguire applicazioni realtime.
Il problema principale di SIFT è che impiega troppo tempo per analizzare una singola immagine, SURF nasce per velocizzare le fasi di rilevazione e generazione dei descrittori, introducendo anche un tipo di quest'ultimi che occupano meno memoria. Questo però va a scapito della robustezza dei cambi di illuminazione o rotazione dell'oggetto.
Per velocizzare il processo è stato introdotto in concetto di immagine integrale, definito da Viola et al. [12], che può essere vista come una rappresentazione intermedia della figura da processare. Per ottenerla vanno sommati i valori in scala di grigio dei pixel in una certa area rettangolare. Ad esempio il valore dell'immagine integrale $I_\Sigma(\alpha)$ nella posizione dell'immagine $\alpha = (x, y)$ è data dalla somma di tutti i pixel facenti parte dell'area rettangolare fra l'origine e il punto α.

$$I_\Sigma(\alpha) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{2.1}$$

Dove I rappresenta l'immagine di input. Una volta che è il calcolo è stato fatto saranno necessarie tre somme per ottenere l'intensità di ogni area rettangolare, come in Figura 2.5. Tale operazione consente di velocizzare di molto il processo di estrazione dei punti chiave e di matching.

Punti chiave basati sulla matrice hessiana

Il calcolo dei punti chiave viene fatto tramite l'utilizzo di una matrice hessiana, approssimata mediante l'uso di immagini integrali.
L'algoritmo si serve di un filtro gaussiano del secondo ordine che permette un'analisi spaziale su diversi ordini di scala. Nella pratica è però necessario discretizzare ed approssimare le gaussiane e quindi potrebbero avere una

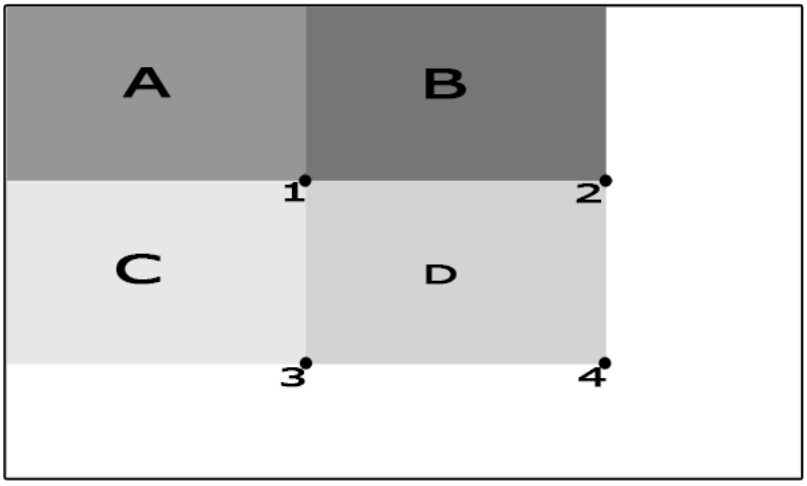

Figura 2.5: Utilizzando immagini integrali, in sole tre somme e quattro accessi alla memoria è possibile calcolare la somma delle intensità all'interno di un'area rettangolare di qualsiasi dimensione. La somma dei pixel all'interno rettangolo può essere calcolata con quattro riferimenti ad un array. Il valore dell'immagine integrale in posizione 1 è la somma dei pixel nel rettangolo A. Il valore di posizione 2 è A + B, in posizione 3 è A + C, e in posizione 4 è A + B + C + D. La somma dentro D può essere calcolata come 4 + 1- (2 + 3).

perdita di ripetibilità nel rilevamento dei punti. Per velocizzare il processo vengono usati dei box-filter, che sono delle approssimazioni di gaussiane che possono lavorare anche con delle immagini integrali, i quali possono essere variati di dimensione, evitando così perdite dovute al sotto-campionamento dell'immagine.

Rappresentazione dello scale-space e rilevamento dei punti chiave

Per via del fatto che spesso un oggetto può essere visto da distanze differenti è necessario che i punti chiave siano rilevati in più fattori di scala. Questa fase viene velocizzata utilizzando i box-filter e le immagini integrali, grazie ai quali non è più necessario applicare iterativamente lo stesso filtro

a diversi livelli precedentemente filtrati come avveniva in SIFT, ma possono essere applicati dei filtri di dimensioni diverse senza variare quelle dell'immagine, come mostrato in Figura 2.6. Tale operazione fornisce anche il vantaggio di evitare il problemi di aliasing dovuti al sotto-campionamento.

Per rilevare i punti chiave viene applicata una soppressione non-maxima

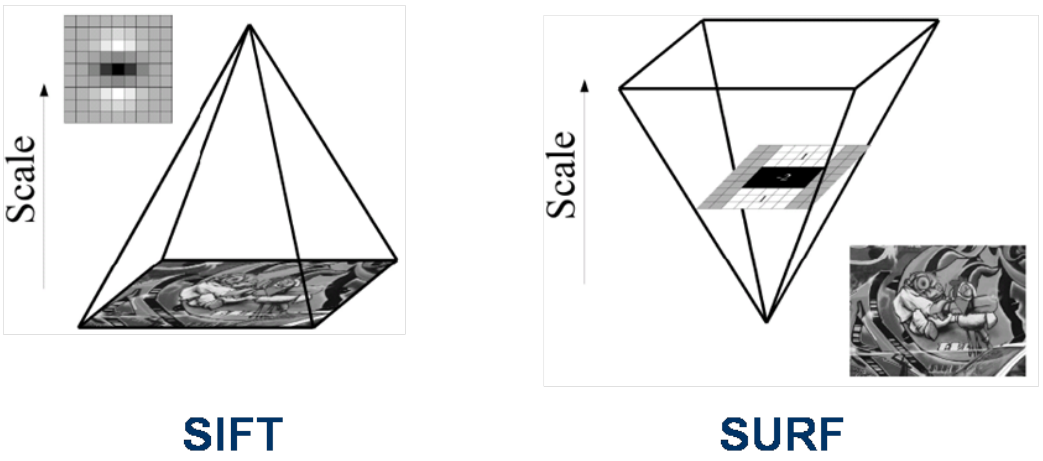

Figura 2.6: Invece di ridurre iterativamente le dimensioni dell'immagine (sinistra), l'uso di immagini integrali permette up-scaling del filtro a costi costanti (a destra).

introdotta da Neubeck e Van Gool [13], che è sostanzialmente un algoritmo di ricerca dei massimi locali, in un vicinato 3x3x3 nello scale-space.

Descrittori dei punti chiave

I vettori SURF si basano sulle risposte ai filtri di Haar del primo ordine piuttosto che sull'istogramma del gradiente, sfruttando il vantaggio ottenuto dall'uso delle immagini integrali. Per individuare l'orientamento principale vengono calcolate le risposte del filtro di Haar lungo le direzione x e y entro una finestra circolare attorno al punto.

Come descrittore si utilizza una finestra quadrata centrata nel punto stesso, di dimensioni basate sul livello di scala alla quale la caratteristica è stata rilevata e orientata nelle direzione precedentemente rilevata. Per memorizzare il

risultato la finestra viene suddivisa in 16 regioni, alle quali viene nuovamente applicato il filtro di Haar per ottenere un vettore per ognuna di esse, che verranno poi concatenati per ottenere il descrittore finale.

2.2.3 ORB

Fa parte di questa classe di algoritmi ORB [14] (Oriented FAST and Rotated BRIEF). Esso sfrutta l'algoritmo di rilevazione dei punti chiave FAST [15] e quello di descrizione BRIEF [16]. Entrambi queste tecniche hanno un basso costo computazionale e dei buoni risultati.
ORB differisce da questi due algoritmi perché introduce:

- una componente per orientamento in FAST veloce ed accurata;
- un metodo di computazione efficiente dell'orientamento in BRIEF;
- un metodo di analisi e correlazione delle caratteristiche con orientamento in BRIEF;
- un metodo di apprendimento per de-correlare caratteristiche BRIEF sotto invarianza rotazionale, che porta a migliori prestazioni nelle applicazioni basate sul nearest-neighbor.

Rilevamento delle caratteristiche oriented-FAST

Il primo passo dell'algoritmo è la rilevazione delle caratteristiche FAST, per farlo quest'ultimo richiede soglia limite di intensità tra il pixel centrale e quelli che li circondano. Il risultato è un insieme di punti molto vicini ai bordi, ma non viene fornita alcune indicazione se questi punti siano vicino agli angoli oppure no (che rappresentano una parte importante di un oggetto, che spesso è invariante e facile da rilevare). Per migliorare questo risultato viene quindi dato in input un valore di soglia che permette di ottenere un numero maggiore di punti rispetto agli N voluti, in seguito vengono ordinati basandosi sul rilevatore di angoli di Harris [17], e vengono mantenuti solo i primi N.

Dato che FAST non fornisce delle caratteristiche multi-scala, come invece fanno SIFT e SURF, viene generata una piramide di immagini a scale differenti e viene applicato l'algoritmo ad ogni livello.
Per ottenere l'orientamento calcola l'intensità ponderata del baricentro di un area circolare, chiamata patch, avente come centro uno degli angoli rilevati. La direzione del vettore da quest'ultimo fino al baricentro viene presa come orientamento del punto chiave.

rBRIEF: Rotation-Aware Brief

Un descrittore BRIEF è costituito da una piccola stringa descrittiva di un area di un'immagine costruita da un set di test di intensità binari. Presa una patch p filtrata con un filtro che la smussa per attenuare le componenti troppo forti, un test binario τ è definito come:

$$\tau(p; x, y) = \begin{cases} 1 & p(x) < p(y) \\ 0 & p(x) \geq p(y) \end{cases} \tag{2.2}$$

Dove $p(x)$ rappresenta l'intensità di p nel punto x. Una caratteristica può essere definita come un vettore di n test binari:

$$f_n(p) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_i, y_i) \tag{2.3}$$

ORB usa una distribuzione gaussiana nel centro della patch e un valore di $n = 256$. Per effettuare lo smussamento viene usata un'immagine integrale.
L'algoritmo BRIEF solitamente fallisce se l'immagine viene ruotata più di qualche grado, Figura 2.7. Per risolvere questo problema si è scelto di ruotare i punti chiave secondo l'orientamento a loro assegnato prima di avviare la procedura.

BRIEF ha una proprietà importante che ogni piccola caratteristica ha una grande varianza e una media di circa 0.5. Ma una volta che questa viene orientata lungo la direzione punto chiave perde tale proprietà e diventa più distribuita. Il vantaggio di avere un'alta varianza è che rende una caratteristica più semplice da discriminare, in quanto risponde in modo differente

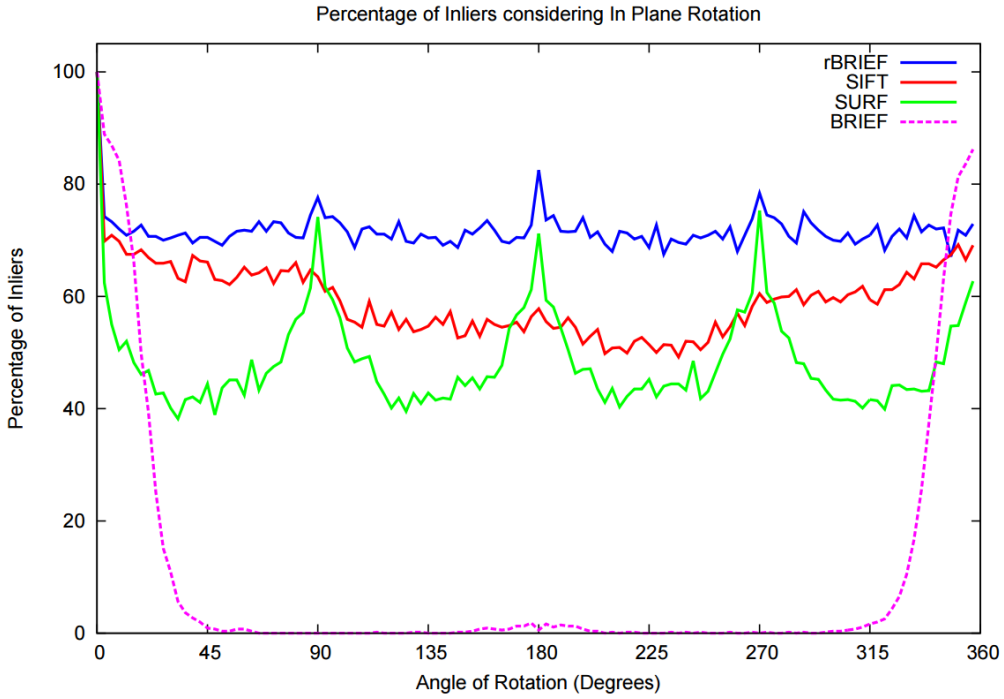

Figura 2.7: Verifica delle performance di SIFT, SURF, BRIEF with FAST, and ORB (oFAST +rBRIEF) sotto rotazioni sintetiche con l'aggiunta di un rumore gaussiano con deviazione standard 10.

agli input. Un'altra proprietà utile è quella di avere dei test non correlati, cosicché ognuno di essi possa contribuire al risultato. Per risolvere tutti questi problemi, ORB esegue una ricerca greedy tra tutti i possibili test binari per trovare quelli che hanno sia alta varianza che media vicino a 0.5, oltre ad essere non correlati fra loro. L'algoritmo finale è stato chiamato rBRIEF.
Per il matching dei descrittori viene usato il multi-probe LSH [18] che migliora la LSH tradizionale. Gli autori affermano che ORB è molto più veloce SURF e SIFT e descrittori di ORB funzionano meglio di quelli di SURF. Questo rende l'algoritmo una buona scelta per dispositivi a bassa potenza.

2.3 Posizionamento nello spazio tridimensionale

Una volta riconosciuto un oggetto è necessario capire anche da che posizione lo si sta guardando, quindi se ci si trova dietro, di fronte, a sinistra o a destra. Più precise saranno queste informazioni, più realistica potrà essere la l'integrazione del mondo reale con quello virtuale.

2.3.1 Uso delle caratteristiche SIFT

Una possibile soluzione al problema è stata presentata da Gordon e Lowe [19]. L'idea è quella di utilizzare le caratteristiche locali invarianti per riuscire a fare un match tra i punti 2D e quelli 3D.
Il sistema funziona in due stage differenti. Durante il primo vengono estratte le caratteristiche SIFT da una serie di immagini di riferimento e vengono stabilite le varie corrispondenze. Il processo è fatto in modo per rimanere lineare rispetto al numero di immagini di input, tramite l'uso di un'indicizzazione veloce approssimata e collegando solo le immagini che formano un albero di copertura.
Le corrispondenze rilevate vengono usate per generare le metriche su come è formato il mondo fisico, e allo stesso vengono estratta i parametri di calibrazione della videocamera.
Una volta che è stato ottenuto un modello del mondo fisico è necessario che siano specificate la posizione, l'orientamento e le dimensioni dell'oggetto digitale.
Il secondo stage del sistema prevede il riconoscimento e una stima accurata della posizione tramite il riconoscimento dei descrittori e il loro collegamento al modello virtuale precedentemente creato. I risultati sono mostrati in Figura 2.8.

Figura 2.8: La tazza di caffè è riconosciuta in ogni fotogramma e viene calcolata la sua posa. La teiera virtuale viene sovrapposta in cima all'oggetto. Gli ultimi due fotogrammi mostrano che la tazza viene riconosciuta anche in presenza di occlusioni parziali, senza avere necessità di fotogrammi precedenti.

Acquisizione della geometria dell'oggetto

Il primo stage prende in input una sequenza non ordinata di immagini dell'oggetto da modellare. Sono necessarie almeno due immagini della scena, ma maggiore è il numero, migliore sarà il risultato ottenuto. Il sistema usa queste immagini per generare un modello 3D sparso della scena, e, allo stesso, tempo per riconoscere il punto da cui si sta guardando l'oggetto e ottenere i parametri di calibrazione della fotocamera. Infine si possono inserire degli oggetti tridimensionali all'interno dell'ambiente digitale così ottenuto. I passi dell'algoritmo sono i seguenti:

1. le caratteristiche locali invarianti vengono estratte dalle immagini;
2. un sistema robusto di matching viene applicato per trovare le corrispondenze tra le caratteristiche estratte;
3. in subset di match multi-vista è scelto come input per un algoritmo per la generazione della struttura e il rilevamento del movimento;
4. le caratteristiche rimanenti sono o posizionate al loro posto tramite triangolazione o, se anomale, rimosse;
5. viene eseguito il posizionamento, l'orientamento e il ridimensionamento di un oggetto digitale relativamente alle coordinate dell'oggetto generato.

Riconoscimento del modello e camera tracking

Per riconoscere il modello nel mondo reale vengono eseguiti principalmente tre passaggi:

1. si estraggono le caratteristiche SIFT dal frame attuale;

2. le nuove caratteristiche vengono confrontate con quelle vecchie usando l'algoritmo best bin first (BBF) [20], fornendo un set di corrispondenze da 2D a 3D;

3. le corrispondenze trovate vengono usate per calcolare l'attuale posa della camera tramite un approccio che combina gli algoritmi RANSAC [21] e Levenberg-Marquardt [22].

I risultati finali ottenuti, tabella 2.1, sono però non adatti ad un'applicazione real-time, infatti gli autori sono riusciti ad ottenere solo 4 frame per secondo con un processore da $1.8Ghz$, quando ne sono necessari almeno 30 per una sensazione di realismo.

feature extraction (SIFT algorithm)	$150ms$
feature matching (BBF algorithm)	$40ms$
camera pose computation	$25ms$
frames per second	4

Tabella 2.1: Tempi di calcolo medi per una sequenza video con dimensione frame 640×480.

2.3.2 Posizionamento tramite uso di hardware specializzato

Se si vuole sapere se si è in prossimità di un oggetto che ha una posizione fissa all'interno di una stanza, ad esempio dentro un museo, la localizzazione

GPS non è affidabile, ma si può anche usare una delle tecnologie di localizzazione indoor esistenti. La maggior parte degli smartphone sono infatti dotati di bluetooth ed esistono tecnologie che utilizzano le onde radio per determinare la posizione.
Questi dispositivi funzionano tramettendo onde radio bluetooth a bassa consumo, la loro batteria è in grado di durare anni e sono utilizzabili con la maggior parte dei dispositivi mobili in commercio. Un esempio del loro funzionamento è stato fornito da Martin et al. [23].
Le periferiche bluetooth LE annunciano se stessi agli altri dispositivi nelle vicinanze inviando pacchetti di breve durata ad intervalli fissi. Non serve alcuna associazione, in quanto i pacchetti vengono mandati in broadcast. Ogni sensore ha associato un ID unico ed è fatto per funzionare a distanze molto brevi.
Un sistema del genere utilizza un server esterno che conosce la posizione approssimativa dei beacon e quindi è in grado di fornire una stima della posizione di chi riceve il loro segnale tramite la triangolazione della potenza rilevata da quest'ultimo, come mostrato in Figura 2.9. I sistemi attualmente

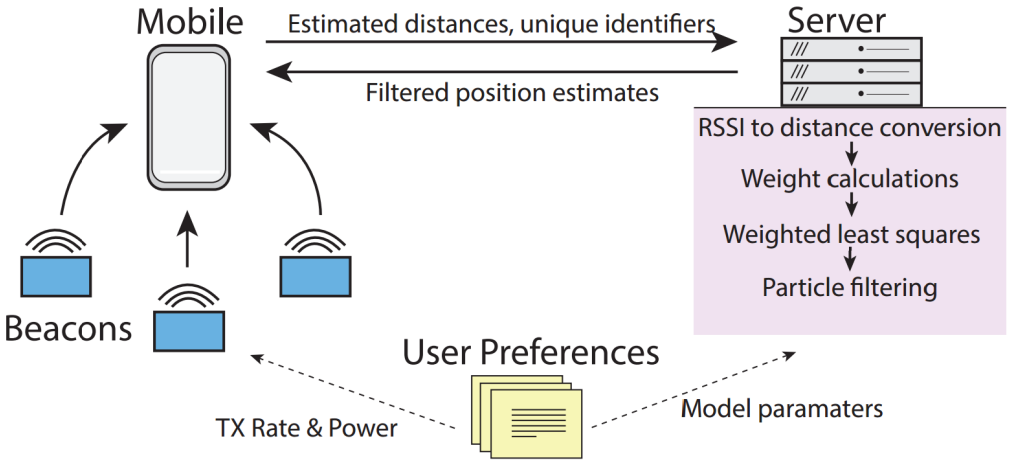

Figura 2.9: Bluetooth LE position estimation flow

in vendita forniscono anche strumenti per calibrare automaticamente il po-

sizionamento all'interno di una stanza, in modo da facilitare il lavoro degli sviluppatori.
Queste tecnologie possono essere usate anche per ridurre il carico computazionale in applicazioni simili a quella qui discussa. Infatti si potrebbero creare dei dataset di caratteristiche dell'oggetto divisi per posizione dell'osservatore e cercare al loro interno solo quelli visibili dal punto rilevato tramite i beacon, riducendo quindi il numero di elementi con cui fare i confronti.

Capitolo 3

Strumenti utilizzati

In seguito è presente una descrizione degli strumenti hardware e software utilizzati per lo svolgimento di questo lavoro.

3.1 OpenCV

OpenCV (open source computer vision) [24, 25] è una libreria sotto licenza BSD, quindi gratuita sia per uso accademico sia commerciale. Dispone di interfacce C ++, C, Python e Java e supporta Windows, Linux, Mac OS, iOS e Android.
OpenCV è stato progettato per l'efficienza computazionale e con un'attenzione particolare verso le applicazioni in tempo reale. Scritta in codice ottimizzato C / C ++, la libreria può usufruire di un'elaborazione multi-core. Se utilizzata in concomitanza con OpenCL, si può sfruttare l'accelerazione hardware di piattaforme di calcolo eterogenee.
Adottato in tutto il mondo, OpenCV vanta una comunità composta da più di 47 mila persone e il numero stimato di download è superiore ai 9 milioni. Il range di utilizzo va dall'arte interattiva, passando per l'ispezione di miniere, fino alla robotica avanzata.

3.1.1 Funzionalità di OpenCV

OpenCV offre diverse funzionalità e strutture dati pensate per l'analisi delle immagini e la computer vision. Al suo interno sono già implementati diversi algoritmi aventi scopi del genere, fornendo quindi gli strumenti necessari a creare applicazioni di questo tipo.

Modulo core, image processing e interfaccia utente

La parte centrale di OpenCV è detta *core*, ed è formata dagli elementi base della libreria necessaria alla manipolazione delle immagini al livello di un singolo pixel.
Ogni immagine è formata da un insieme di valori, ognuno dei quali rappresenta l'intensità del colore in quel punto (Figura 3.1). Per poter memorizzare questi insiemi di elementi viene usata una classe chiamata Mat. Grazie ad essa non è necessario allocare manualmente la memoria o deallocarla in seguito. La classe Mat è fondamentalmente composta da due parti: un'intestazione (contenente informazioni quali la dimensione della matrice, il metodo utilizzato per la memorizzazione, l'indirizzo in cui è memorizzata, e così via) e un puntatore ai valori dei pixel.
Queste strutture dati sono alla base di ogni funzionalità della libreria e sono presenti strumenti per poter leggere file, caricare video in realtime e applicare filtri alle immagini.
Oltre che analizzare le immagini vengono forniti strumenti grafici come trackbar o semplici bottoni.

Modulo calib3d

Un modulo molto importate fornito da OpenCV è il calib3d, al suo interno sono presenti strumenti per trasformare una semplice immagine 2D in un mondo tridimensionale.
Un esempio è dato dalla possibilità di generare una calibrazione di una fo-

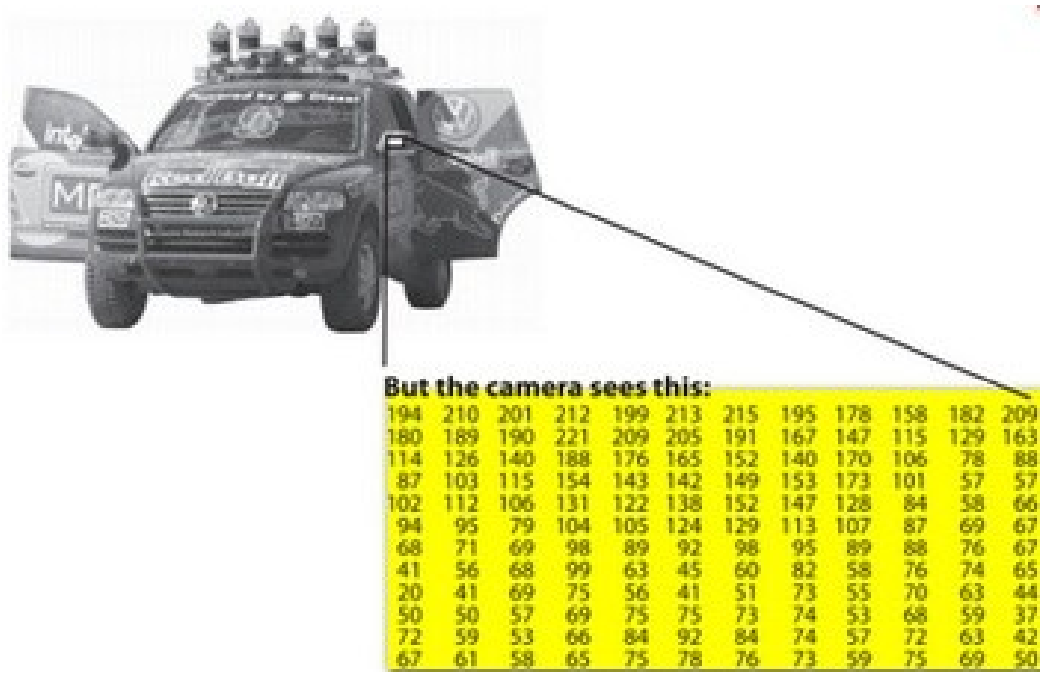

Figura 3.1: Un esempio di come è composta un'immagine digitale, non è nulla di più che una matrice di valori che rappresentano l'intensità di grigio in un pixel.

Fonte immagine http://opencv.org.

tocamera: queste, per via dell'abbassamento dei costi alla fine del ventesimo secolo, sono diventate un elemento comune all'interno della vita quotidiana. Come ogni cosa, questa economicità ha provocato un rovescio della medaglia: una distorsione significativa dell'immagine. Fortunatamente tale alterazione è costante e con una calibrazione e una rimappatura può essere corretta. Inoltre, grazie a questo processo è possibile anche determinare la relazione tra le unità naturali della fotocamera (pixel) e le unità del mondo reale (ad esempio millimetri).

Altre funzionalità

Sono incluse anche funzionalità per l'analisi dei video come il motion tracking, o per il rilevamento di oggetti, compresi algoritmi come SIFT, SURF

e ORB. Ma anche funzionalità più avanzate come un modulo per il machine learning o uno per il funzionamento parallelo utilizzando CUDA.

3.1.2 Utilità in questo lavoro

La libreria è stata utilizzata all'interno di questo lavoro per analizzare i fotogrammi ripresi dalle fotocamere o da video pre-generati, in esso sono presenti infatti molti strumenti utilizzabili per applicazioni di realtà aumentata o mista.
OpenCV mette a disposizione, già implementati, algoritmi come SIFT, SURF e ORB, utili per il rilevamento degli oggetti. Ma anche per la stima e la correzione delle distorsioni su immagini di cui si conosce la forma, come scacchiere o marker.
Essendo compatibile con diversi sistemi operativi non è stato un problema riutilizzare lo stesso codice su dispositivi differenti. Facilitando quindi il lavoro di sviluppo.

3.2 OpenGL

Per la parte grafica è stato scelto di utilizzare uno stardard compatibile con diversi sistemi operativi e che fa un uso diretto delle funzionalità della GPU.
OpenGL è l'ambiente principale per lo sviluppo di software per dispositivi portatili, per applicazioni 2D interattive e grafiche 3D. Dalla sua introduzione nel 1992, è diventata la più diffusa e supportata interfaccia di programmazione per la grafica 2D e 3D del settore, portando alla pubblicazione di migliaia di applicazioni per una vasta gamma di piattaforme. OpenGL incorpora un ampio set di funzionalità per il rendering, il texture mapping, gli effetti speciali, e altre tecniche visualizzazione.

3.3 Blender

Per la creazione dei modelli tridimensionali è stato usato il software gratuito e open source Blender, che è una suite per la creazione di modelli e animazioni 3D. Supporta la totalità della pipeline di modellazione 3D ed è utilizzabile per applicazione di animazione, simulazione, rendering, motion tracking, video editing e la creazione di giochi.
Il software è stato utilizzato per la creazione dei vari modelli, per l'analisi di quelli ottenuti tramite la TAC effettuata da Morigi et al. [6] e per generare animazioni tridimensionali. Parte del lavoro è stato svolto infatti all'interno di un ambiente 3D, in quanto non è sempre possibile poter testare ogni risultato in un caso reale, soprattutto quando l'ambiente in questione è una stanza all'interno di un museo, dove non è sempre possibile apportare i cambiamenti, anche se provvisori, di cui si ha bisogno in questi casi.

3.4 Hardware utilizzato

Non è stato usato nessun hardware particolare, ma solo delle webcam con risoluzione 640x480 pixel e uno smartphone Apple iPhone 5S. Oltre che in 3D, la scena è stata anche ricostruita su scala ridotta per poter fare dei test in laboratorio su un caso reale.

Capitolo 4

Soluzioni analizzate

Durante il lavoro di progettazione ci si è resi conto che esistevano diverse soluzioni al problema del rilevare la posizione di un oggetto all'interno di un'immagine, quindi è stato deciso di analizzare, implementare e testare ognuna di esse, in modo da poter scegliere quella più adatta al caso specifico.

Lo scopo del progetto è quello di mostrare ai visitatori di un museo il lavoro svolto da Morigi et al. [6] per restaurare un antico globo di Vincenzo Coronelli (Figura 4.1). Per fare ciò è stato scelto di sviluppare un'applicazione che, utilizzando la realtà mista, mostri i risultati da loro ottenuti facendo una TAC all'oggetto.

Figura 4.1: Globo celeste costruito da Vincenzo Coronelli all'inizio del diciottesimo secolo, situato all'interno della biblioteca Manfrediana di Faenza. Un elemento che salta molto all'occhio sono i riflessi presenti sulla teca.

4.1 Simmetria radiale

Una fotografia rappresenta una visione di un mondo tridimensionale su un solo piano. Se ci si sofferma ad osservare l'oggetto obiettivo di questo lavoro si capisce che esso non è altro che una grossa sfera, cioè una figura geometrica che diventa un cerchio all'interno di una foto.
Loy e Zelinsky [26] hanno proposto un algoritmo in grado di rilevare aree circolari di una certa dimensione all'interno di immagini digitali. Il metodo serve a rilevare quei punti dove vi è un alta simmetria radiale.

Si dice che una figura piana possiede una simmetria radiale di ordine p (numero naturale diverso da zero) se, fissato un angolo α di ampiezza:

$$\alpha = \frac{360°}{p} \tag{4.1}$$

Lo stesso risultato è mostrato graficamente in Figura 4.2.

Questo calcolo viene fatto sopra un set di uno o più raggi R che dipendono dalle dimensioni della caratteristica che si vuole rilevare. Il risultato è una matrice dove ogni punto rappresenta l'intensità del contributo alla simmetria

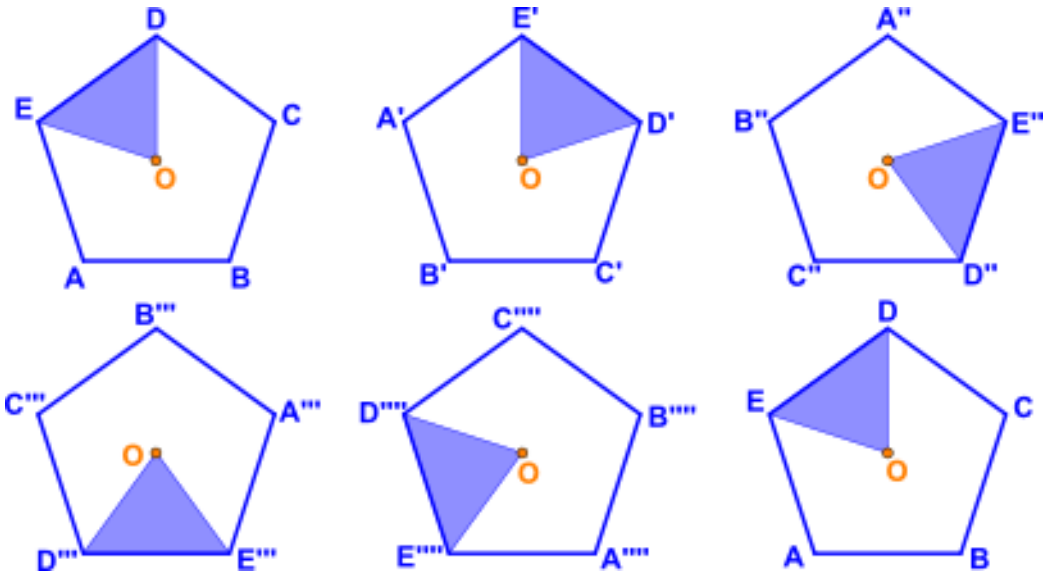

Figura 4.2: Simmetria radiale di un pentagono, applicando una rotazione centrata sempre nel punto O di α gradi si riottiene sempre la stessa figura.

Fonte immagine http://www.youmath.it.

radiale di raggio $r \in R$ centrata nel pixel in questione, cioè ad una distanza r da ogni punto.

4.1.1 Principio teorico

Sostanzialmente vengono formate due matrici per ogni raggio r, quella degli orientamenti O_r e quella delle intensità M_r. I valori vengono poi generati a partire dal gradiente dell'immagine di partenza g. Per ogni punto p si calcola un valore positivo p_{+ve} e uno negativo p_{-ve}, dove:

- p_{+ve} equivale al punto a distanza r in cui il vettore del gradiente $g(p)$ sta puntando;
- p_{-ve} equivale al punto a distanza r in cui il vettore opposto del gradiente $g(p)$ sta puntando.

Un esempio grafico è fornito nella Figura 4.3.

Sia O_r che M_r vengono inizializzati a 0, e ogni coppia di punti p_{+ve} e p_{-ve}

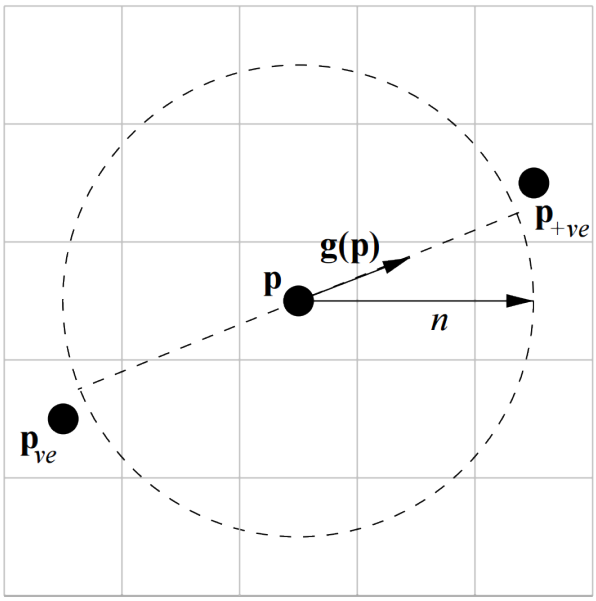

Figura 4.3: La posizione dei pixel p_{+ve} e p_{-ve} generate dal gradiente g(p) per un raggio di $r = 2$.

forniscono rispettivamente una posizione in cui avverrà un incremento o un decremento. La differenza è che nella prima matrice il valore è uguale a ± 1, nella seconda è uguale a $\pm\,||g(p)||$.
Come passo finale si calcola una matrice S che viene ricavata dalla somma di R matrici S_r. Quest'ultime non sono altro il prodotto delle matrici O_r e M_r normalizzate con valori compresi fra $[-1, 1]$.

4.1.2 Caso specifico

Nel caso specifico del problema qui riportato si sta cercando un grosso cerchio, con diametro probabilmente molto vicino ai due terzi della larghezza dell'immagine, se si tratta di un fotogramma in verticale, o da un terzo fino alla metà se è in orizzontale, lo si può notare in Figura 4.4, dove sono state evidenziate le porzioni occupate dal globo se inquadrato da vicino.

Si è anche interessati solo ai valori di p_{+ve}, in quanto si sta cercando il

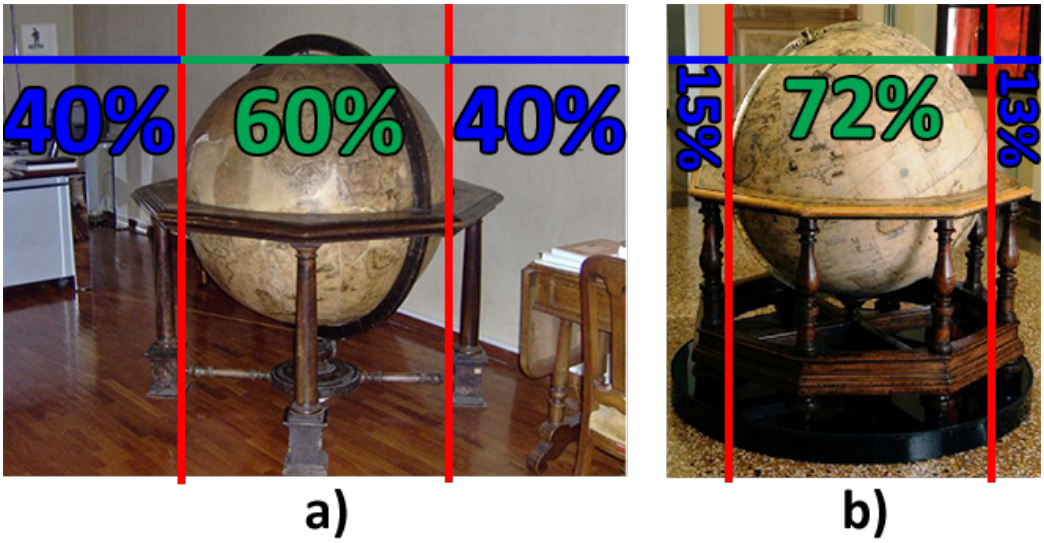

Figura 4.4: Le proporzioni di un globo di Coronelli rispetto ad una foto in orizzontale (a) e rispetto ad una in verticale (b).

centro della sfera, che è ottenibile seguendo il vettore estratto dal gradiente dei bordi, che corrisponde alla derivata prima. L'unico problema è quello di trovare la corretta dimensione dell'oggetto poiché l'algoritmo richiede un insieme di dimensioni possibili come parametro di input.
Un'ottimizzazione si potrebbe ottenere sfruttando il vantaggio dato dal fatto che la sfera occupa la gran parte dell'immagine, grazie a questa assunzione si potrebbe lavorare su una versione molto rimpicciolita della figura, in modo da poter migliorare le prestazioni.
L'algoritmo è stato sviluppato e testato su fotografie del globo in questione, i risultati sono mostrati in Figura 4.5. Durante i test è sorto un primo problema dovuto al fatto che il supporto esterno dell'oggetto ha un'intensità maggiore del globo stesso, quindi, dalla visuale dell'immagine proposta, si ottiene una circonferenza spostata più a sinistra rispetto a quella effettiva.

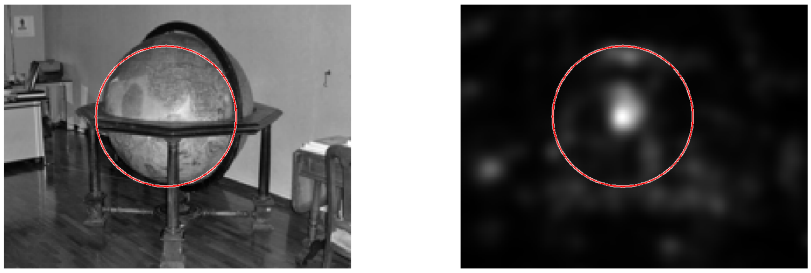

Figura 4.5: Algoritmo di simmetria radiale testato su un'immagine di un globo. A destra c'è l'area circolare rilevata e a sinistra l'intensità dei valori di simmetria all'interno della figura.

4.1.3 Aree circolari in OpenCV e problematiche di questi metodi

Oltre che il metodo visto in precedenza è stato testato anche un algoritmo implementato all'interno della libreria OpenCV. Questo, definito come trasformata circolare di Hough (in contrapposizione a quella lineare), è sempre basato sul gradiente ed è stato descritto da Yuen et al. in [27]. Utilizza la trasformata di Hough e richiede un raggio minimo e massimo come parametro di input.
I risultati ottenuti sono simili a quelli presenti in Figura 4.6, la maggior parte di volte il globo viene riconosciuto ed evidenziato correttamente. Come nel caso precedente, è però necessario fare una stima delle dimensioni che il globo potrebbe avere all'interno della figura, inoltre potrebbero sorgere problemi dovuti ai riflessi degli altri oggetti sulla teca di protezione.
Un altro problema comune ad entrambi i metodi è dovuto al fatto che anche se il globo viene correttamente riconosciuto non si hanno informazioni su

come questo sia orientato. Cioè non è possibile distinguere se lo si sta osservando da sinistra o da destra, o dal basso o dall'alto e così via. Oltretutto il problema non ha una soluzione intuitiva come il precedente caso. Questo genere di soluzioni è quindi risultato non adatto allo scopo prefissato.

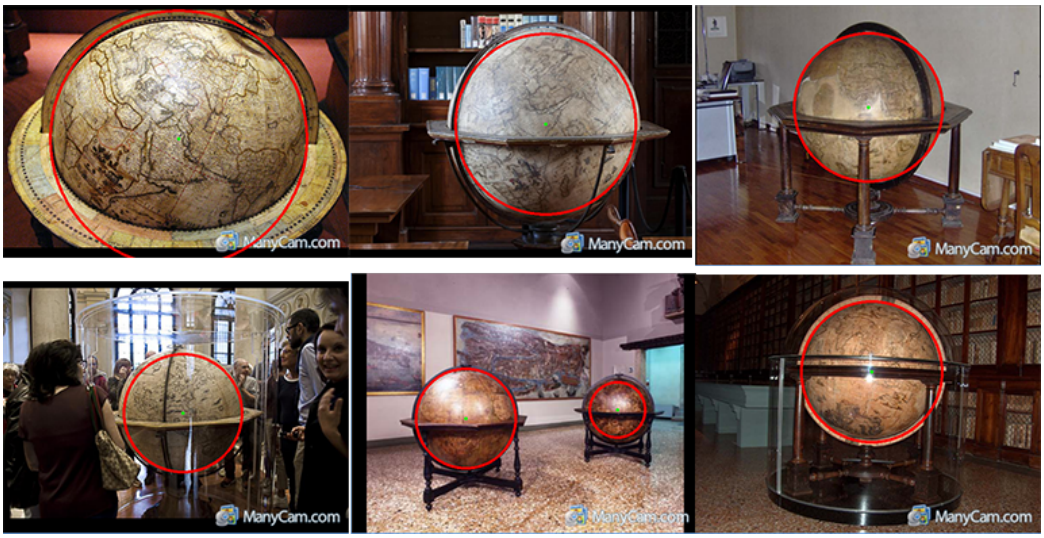

Figura 4.6: Alcuni risultati dell'algoritmo che utilizza la trasformata di Houngh per rilevare le aree circolari già implementato all'interno della libreria OpenCV.

4.2 Riconoscimento del globo utilizzando le caratteristiche locali

Il metodo qui proposto si basa su algoritmi di rilevamento delle caratteristiche locali di un oggetto tramite l'uso di algoritmo come SIFT, SURF o ORB, ed è stato implementato usando l'ultimo di quelli citati, in quanto ha un buon rapporto tra la qualità dei risultati e il costo computazionale in generale.
Questa scelta è stata fatta testando tutti e 3 gli algoritmi su un set di 10 immagini differenti con risoluzione 640x480 pixel. Per ognuna di esse sono stati estratti i punti chiave e generati i descrittori. In seguito l'immagine è

stata ruotata, sono stati nuovamente generati i descrittori e confrontati con quelli precedentemente estratti al fine di trovare delle correlazioni. Sono stati calcolati i tempi necessari ad effettuare questi calcoli su 100 run per ogni immagine, i risultati sono mostrati in Figura 4.7. ORB si è dimostrato dalle 10 alle 100 volte più di SIFT e SURF ad effettuare tutto il procedimento, quindi la scelta è stata ovvia.

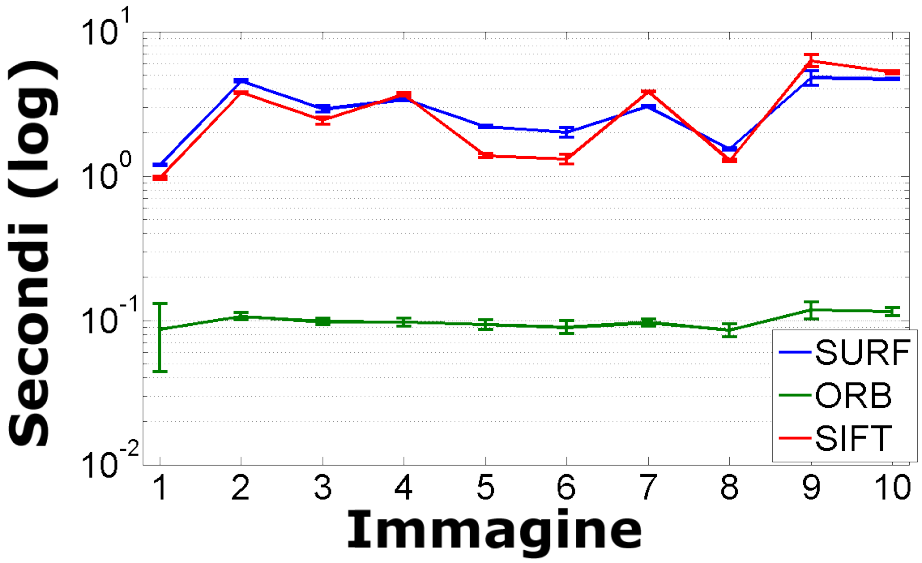

Figura 4.7: Tempi di esecuzione e confronto di SIFT, SURF e ORB su 10 immagini. Sono stati eseguiti 100 run per ogni immagine di cui in figura sono stati mostrati gli intervalli di confidenza. La scala dei secondi è logaritmica. L'hardware utilizzato è un Intel i3 $1.8GHz$ *quadcore con* $4GB$ *di RAM.*

Un esempio intuitivo di applicazione della soluzione in questione può essere fatto utilizzando un qualunque poligono con complessità minima, come un cubo. Si immagini che questo abbia in ogni faccia un disegno differente, allora si potrebbero estrarre le caratteristiche di ognuna di esse e riutilizzarle in seguito per stabilire quali parti siano visibili e come siano orientate. Conoscendo l'esatta forma della struttura con le corrette proporzioni non è difficile sovrapporre un modello 3D a quello reale.
Sostanzialmente andranno eseguiti i seguenti passi:

- estrazione delle caratteristiche ORB e generazione dei descrittori dalla scena;
- match dei descrittori della scena con quelli del modello tridimensionale usando il FLANN matcher [28];
- stima della posizione con gli algoritmi PnP e RANSAC;
- applicazione di un filtro lineare di Kalman per eliminare eventuali pose sbagliate.

Data la difficoltà di ricreare esattamente un modello tridimensionale del globo oggetto di questo lavoro, è stato deciso di utilizzare un modello più semplice (anche un cubo opportunamente orientato può andare bene) a cui sono impostate come texture delle foto dell'oggetto reale riprese da angolazioni differenti.
In seguito, sono stati estratti i descrittori dal modello e memorizzati in un file per rendere più rapido il calcolo. Ogni volta che si usa l'applicazione vengono estratte le caratteristiche ORB dai frame ripresi dalla fotocamera e confrontati con quelli in memoria. Se viene trovata una corrispondenza di usa l'algoritmo PnP, che serve per effettuare una trasposizione delle coordinate da 2D a 3D, per ottenere un sistema di coordinate che coincide con quello che la fotocamera inquadra.

4.2.1 Risultati ottenuti e problemi

Il metodo riesce a stimare la posizione dell'oggetto all'interno dell'ambiente tridimensionale con precisione, ma soffre per via dei cambi di illuminazione che possono esserci durante l'arco della giornata, oltre ai problemi derivanti dai riflessi dovuti alla teca. Altro problema è dato dal fatto che non tutti i dispositivi sono in grado di eseguire i calcoli richiesti in tempo reale. Quindi si rischia non riuscire ad ottenere quella sensazione di realisticità cercata.

Anche il tenere in memoria i descrittori del modello virtuale può diventare un problema, il file di prova, che è stato preso da un solo lato del modello

3D, occupava infatti più di 20MB.

4.2.2 Ottimizzazione tramite l'utilizzo di sensori beacon

Per migliorare le prestazioni si è pensato di utilizzare dei sensori beacon, in particolare sono stati testati gli Estimote. Questi sensori tramite l'emissione ad intervalli regolari di un segnale radio, permettono di creare un sistema di localizzazione indoor. In Figura 4.8 i sensori sono stato posizionati sulle pareti e tramite l'applicazione disponibile sul dispositivo mobile è possibile stimare la posizione all'interno della stanza di chi la sta usando.

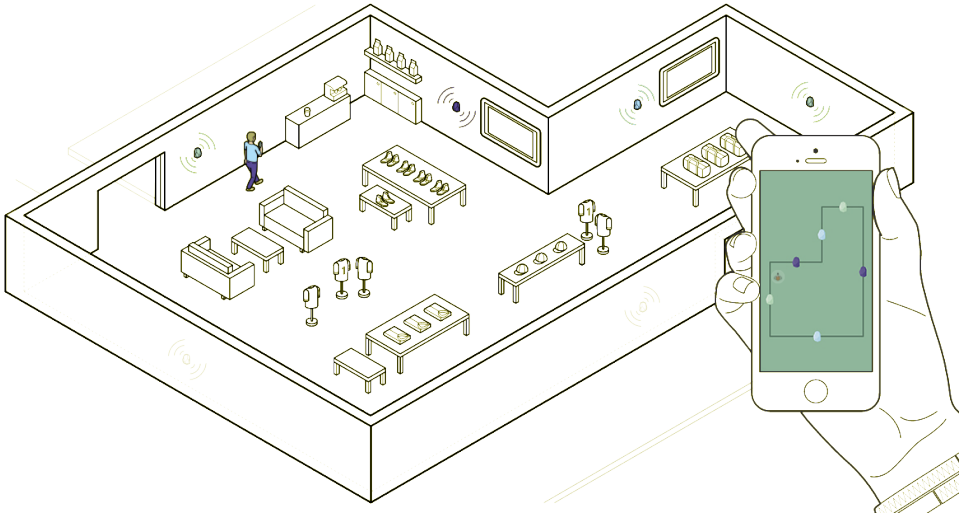

Figura 4.8: Esempio di funzionamento dei beacon Estimote, tramite l'applicazione dei sensori sulle parete è possibile stimare la posizione di una persona che si sta muovendo all'interno della stanza.

Fonte immagine http://estimote.com.

Utilizzando questi sensori è possibile avere una posizione approssimative dell'osservatore rispetto all'oggetto. Si può sfruttare questo vantaggio creando più file contenenti i descrittori del modello virtuale visto da angolazioni

differenti, invece che mettere tutte le caratteristiche all'interno di uno solo. Fatto ciò basta richiamare via via solo i file necessari in base al punto in cui ci si trova, diminuendo quindi di molto l'insieme all'interno del quale effettuare i confronti.

4.3 Sistema basato sui marker

I migliori risultati, come è ovvio, si ottengono usando un hardware specializzato nel riconoscimento di oggetti 3D, solo che questo obbligherebbe un museo ad acquistare e manutenere tali apparecchiature, che spesso non risultano molto economiche. Non tutti i dispositivi inoltre sono in grado di estrarre le caratteristiche chiave dai frame e confrontarle con quelle presenti in un dataset in tempo reale.

Si è deciso quindi di pensare delle modifiche strutturali molto lievi dell'area in cui è esposto l'oggetto. In particolare si è presa in considerazione l'applicazione di alcuni marker di dimensioni ridotte ($12cm$x$12cm$) alla base dell'oggetto. Per rilevare la presenza di questi elementi, e quindi valutare la posizione reale dell'oggetto, è stato pensato di utilizzare gli smartphone degli utenti stessi, ai quali verrà fornita un'applicazione apposita. Il vantaggio di questa scelta è che, essendo i marker semplici immagini stampate su carta bianca, e quindi oggetti passivi, non richiedono alcuna alimentazione, né tantomeno una grande manutenzione.

4.3.1 Funzionamento dei marker

Solitamente un marker è disegnato come un rettangolo, che racchiude al suo interno delle aree bianche e nere, circondate da un contorno di quest'ultimo colore [29], Figura 4.9.

In questo progetto sono stati usati dei marker quadrati di dimensione 5x5, che significa che contengono una matrice quadrata con larghezza 5 di valori binari. Gli elementi al suo interno rappresentano un codice formato da 5 word, ognuna di 5 bit.

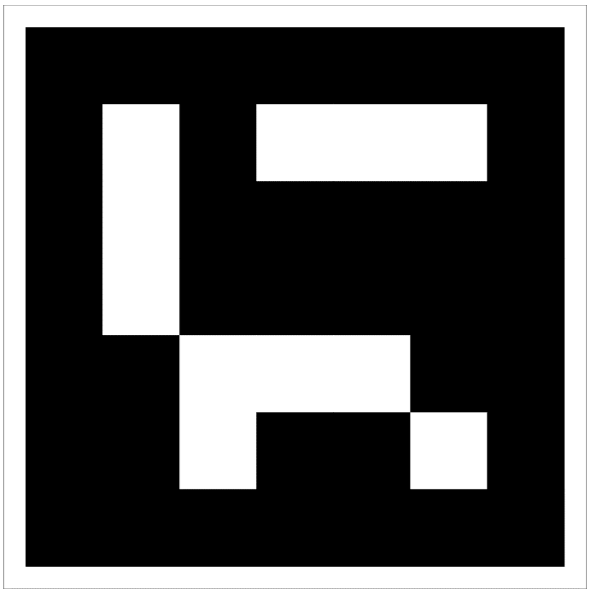

Figura 4.9: Marker 5x5 rappresentate il numero 270.

Per favorire la codifica vengono usati solo due bit per l'informazione, mentre i tre restanti servono per rilevare eventuali errori. Avendo solo due elementi per ognuna delle 5 word, possono essere rappresentati un massimo di 210 valori, cioè 1024 ID differenti.
La codifica utilizzata è una versione modificata del codice di Hamming [30], dove il primo bit è invertito. Questo serve a fare in modo che il numero 0, non sia più rappresentato dal valore 00000, ma dal valore 10000, togliendo la possibilità di avere un marker totalmente nero, tabella 4.1. Questo accorgimento è necessario perché altrimenti ci sarebbero dei problemi di riconoscimento quando nell'ambiente sono presenti oggetti neri.

Un esempio grafico è presente in Figura 4.10, come primo passo viene fatta un divisione in quadratini, aventi tutti le stesse dimensioni. Il passo successivo consiste nel controllare il bordo, assicurandosi che sia totalmente vuoto. Fatto ciò si deve procedere riga per riga nella parte interna e confrontare i valori con la tabella 4.1.
Nell'esempio dato la prima riga coincide col valore 01, la seconda e la terza

BINARIO	CODIFICA				
00	1	**0**	0	**0**	0
01	1	**0**	1	**1**	1
10	0	**1**	0	**0**	1
11	0	**1**	1	**1**	0

Tabella 4.1: Codifica utilizzata per le righe dei marker

equivalgono al valore 00, la penultima a 11 e l'ultima a 10. Considerando questi bit come se fossero ordinati dal più significativo in giù si ottiene il numero binario 0100001110 che in decimale equivale al valore 270.

Altra cosa che rende questo metodo robusto è il fatto che il marker non

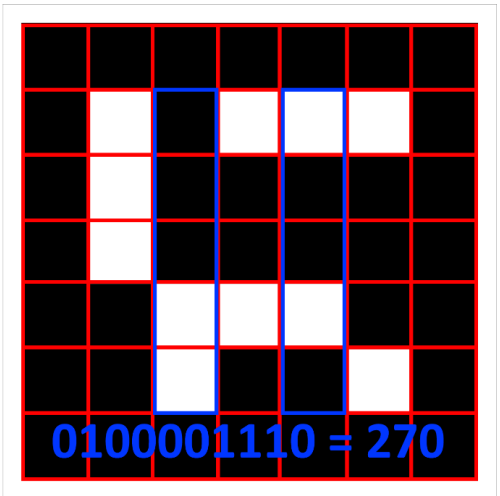

Figura 4.10: Esempio di decodifica di un marker 5x5.

può essere letto se ruotato. Con un marker quadrato possono infatti esserci 4 possibili interpretazioni, ma, per via dei bit di parità, solo una di queste darà un risultato valido, Figura 4.11.

Figura 4.11: Possibili interpretazioni di un marker, di queste solo la prima ha una codifica valida.

4.3.2 Rilevamento dei marker

In questo paragrafo è spigato come estrarre un marker da un'immagine. Questa operazione si divide sei fasi:

1. conversione dell'immagine in scala di grigio;
2. applicazione una soglia binaria;
3. rilevamento dei contorni;
4. scelta dei possibili marker;
5. rilevazione e decodifica dei marker tra i candidati;
6. stima della posizione nell'ambiente tridimensionale.

La conversione in scala dei grigi è necessaria perché solitamente i marker contengono solo i colori nero e bianco, quindi rende più semplice l'operazione. La conversione in valori binari dell'immagine consiste nel trasformare ogni pixel o in bianco o nero e serve a trovare i contorni dei marker. Ci sono due modi per fare ciò:

- si impone una soglia tra bianco e nero. Questo metodo è molto semplice, ma dipende molto dall'illuminazione della scena;
- si calcola una soglia in base al colore dei pixel nella scena. È più lento ma più efficace.

Il risultato di tale conversione è mostrato in Figura 4.12.

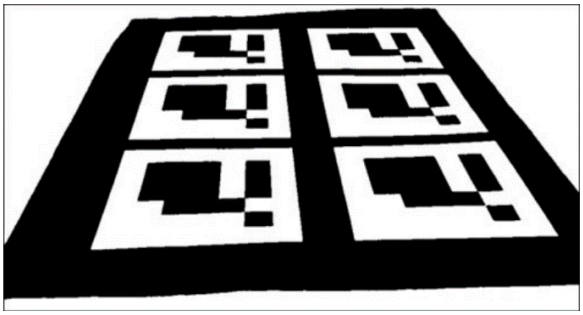

Figura 4.12: Immagine binarizzata, si nota una netta divisione tra le parti in bianco dei marker e quelle in nero.

Il passo successivo prevede la rilevazione dei contorni, per fare ciò è stato usato l'algoritmo di Suzuki et al. [31], escludendo i contorni più piccoli di una certa soglia in quanto potrebbero non contenere dei marker, o comunque questi non sarebbero riconoscibili.

Per scegliere quali potrebbero essere dei marker controllano i vari poligoni e se questi hanno più, o meno, di 4 vertici vengono esclusi.
Trovati i possibili candidati viene applicata una trasformazione per rimuovere la visione prospettica, in modo da poterli vedere frontalmente. Un esempio è mostrato in Figura 4.13.

In seguito viene applicato l'algoritmo di Otsu [32] che consente di avere una divisione più netta tra le parti in bianco e in nero.

L'ultimo punto prevede di dividere l'immagine ottenuta in una sezione 7x7 e calcolare il numero del marker, facendo attenzione a tutte le possibili rotazioni. Se viene restituito un valore l'elemento in questione viene accettato, altrimenti viene scartato.

4.3.3 Stima della posizione 3D

Per facilitare il calcolo viene prima definito un perimetro del marker sull'immagine congiungendo i 4 vertici.

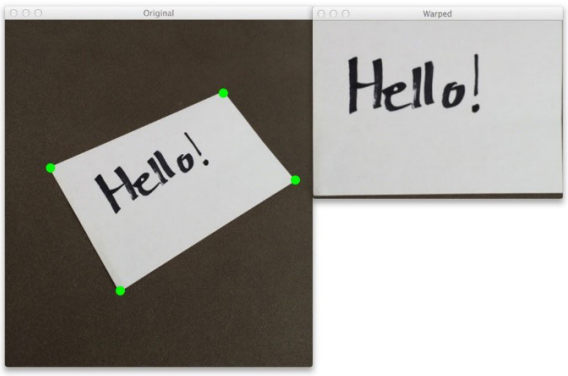

Figura 4.13: Passaggio dalla visualizzazione prospettica a quella frontale di un elemento di un'immagine.

Fonte immagine http://www.pyimagesearch.com.

Conscendo la posizione precisa dei vertici nell'immagine si può calcolare la trasformazione tra la camera e il marker in un ambiente 3D. Infatti le due operazioni da trovare tra il punto di vista e l'oggetto sono la traslazione e rotazione.

Considerando la Figura 4.14 è possibile intuire un metodo per calcolare le matrici di rotazione a traslazione che descrivono la posizione di ogni marker. Il punto C indica il punto di vista, ed i punti $P_1 \dots P_4$ rappresentano i vertici del poligono così come inquadrati dalla camera. Conoscendo già la forma iniziale dei marker è possibile traslare e ruotare i punti nelle posizioni $p_1 \dots p_4$, che rappresentano il poligono senza nessuna trasformazione applicata. Fatto ciò basta applicare delle operazioni inverse agli oggetti che si vuole aggiungere alla scena per posizionarli così come sono posizionati i marker.

4.3.4 Posizionamento dei marker

Per scegliere la posizione e la dimensione adatta dei marker sono stati fatti diversi test. Inizialmente il lavoro è stato svolto in un mondo tridimensionale totalmente virtuale, mostrato in Figura 4.15, in cui è stata ricostruita la struttura che si vuole riconoscere, applicando ai 4 lati un marker, in modo

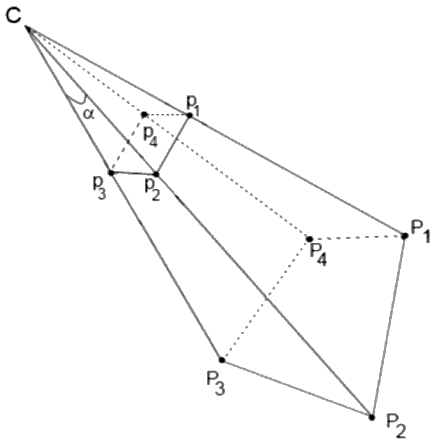

Figura 4.14: Traslazione dei vertici di una figura vista da una camera C in vertici che rappresentano le dimensioni e la forma reali del poligono.

da averne almeno uno sempre visibile.

Successivamente è stato costruito un modellino in miniatura a cui sono stati

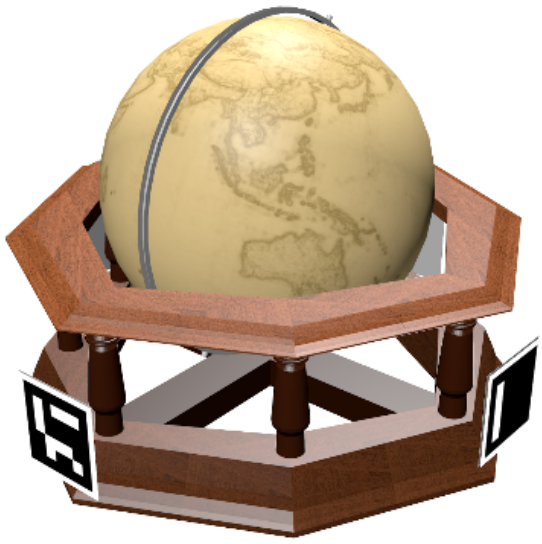

Figura 4.15: Modello 3D del globo su cui sono stati fatti i test iniziali, i marker sono stati posizionati ai 4 lati, in modo da averne almeno uno sempre visibile.

applicati dei marker, sempre ai 4 lati. Il problema di questo oggetto è che

andava posizionato a qualche *cm* dalla fotocamera per simulare la presenza del globo di dimensioni reali, ciò impediva l'uso di una risoluzione video HD (720p), in quanto la messa a fuoco era troppo lenta per un oggetto così vicino, imponendo quindi l'uso di marker di dimensioni eccessive rispetto a quelle del modello stesso.
Si è deciso quindi di fare dei test sull'oggetto reale, utilizzando dei marker di dimensioni da 10x10*cm* a 20x20*cm*, decidendo infine di utilizzarli con dei lati pari a 12x12*cm* e di posizionarli alla base dell'oggetto, applicandoli in un rialzo facente parte della struttura protettiva dell'opera, in modo da non dover effettuare cambiamenti fisici, ma solo incollare i marker.
Posizionando i marker come descritto si ottiene una sorta di cerchio che circonda l'oggetto. Questa scelta, oltre ai motivi descritti in precedenza, è stata adottata per poter stabilire anche la posizione da cui si sta guardando il globo (davanti, dietro, di fianco ecc...). Per raggiungere questo obiettivo i marker avranno come numero identificativo l'angolo di visione dell'oggetto stesso, e questo, essendoci 360° in un cerchio, rende più che sufficiente il limite di ID imposto dai marker 5x5, cioè 1024.
Una rappresentazione grafica è presente in Figura 4.16. La linea rossa indica il perimetro dell'oggetto. A questo verranno applicati diversi marker verticali indicanti l'angolo di rotazione. Mentre la traslazione e la rotazione dovuti al punto di vista saranno calcolati come in precedenza.

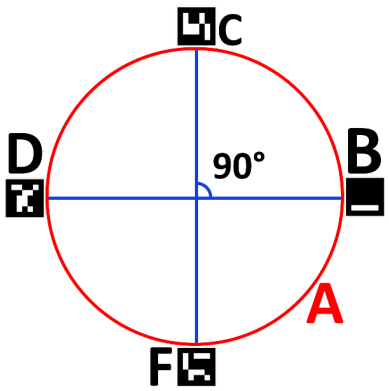

Figura 4.16: Posizionamento dei marker attorno la il perimetro A. B indica il valore 0, C il valore 90, D 180 e F 270. Tutti i marker sono ruotati nel con la base nel verso uscente dal cerchio. La figura mostrata serve solo a far capire dove verranno posizionati i marker, in quanto non saranno in orizzontale sul pavimento, ma in verticale a contatto con l'oggetto

Capitolo 5

Soluzione adottata

Per raggiungere l'obiettivo finale sono state analizzate tre diverse soluzioni.

La prima prevedeva che venissero rilevati gli oggetti di forma circolare all'interno dei frame acquisiti dalla fotocamera. Durante i test il metodo ha dato dei risultati soddisfacenti nel riconoscere il globo, se correttamente tarato, ma non è in grado di fornire alcuna indicazione su quale lato dell'oggetto si sta osservando. Inoltre la scelta dei parametri di input dipende molto dalla distanza dall'oggetto e dall'illuminazione della stanza, richiedendo un numero non banale di variabili da prendere in considerazione, come una raggio minimo e massimo della sfera (che può variare in base alla distanza).

Il secondo metodo utilizzava algoritmi per il riconoscimento delle caratteristiche locali per rilevare la posizione e l'orientamento nelle 3 dimensioni dell'oggetto. Si è anche pensato di usarlo in concomitanza con dei sensori radio per stimare la posizione dell'osservatore rispetto all'oggetto in questione. Ma per via delle capacità computazionali richieste e dello spazio occupato in memoria dal dataset delle caratteristiche dell'oggetto è stato deciso di scartarlo, inoltre l'acquisto e la manutenzione dei sensori avrebbe richiesto un'ulteriore spesa per il museo in questione.

Lasciata per ultima perché considerata leggermente invasiva è la soluzione che alla fine è stata valutata come più idonea. Questa prevede l'utilizzo

di alcuni marker per stimare la posizione dell'oggetto rispetto l'osservatore. Rilevare uno o più marker e calcolare le trasformazioni a loro applicate per trovarsi in quella posizione, è un'operazione che non richiede grosse risorse di calcolo e, rappresentando un sistema di codifica, nemmeno di memorizzazione, quindi è un compito che può essere svolto con un dispositivo mobile. È stato studiato un metodo per rendere la soluzione meno invasiva possibile, posizionando i marker in modo da non richiedere modifiche strutturali e non ostruire la visione dell'oggetto. Inoltre i marcatori sono essenzialmente solo dei fogli di carta bianca con dei quadratini neri stampati, quindi hanno un impatto economico minimo.

5.1 Posizione finale dei marker

La scelta della posizione è stata fatta cercando di minimizzare l'impatto visivo dato dai marker, garantendo comunque un funzionamento adeguato del sistema. Per ottenere questi risultati si è scelto di non apportare modifiche strutturali all'ambiente dove si trova l'oggetto, ma di apporre dei marker di dimensioni ridotte (12x12cm) ai lati del rialzo presente alla base della struttura (Figura 5.1).

Il progetto iniziale prevedeva di apporre solo 4 marker, indicanti gli angoli

Figura 5.1: Due basi analizzate della struttura di protezione di un globo. Può essere rotonda (a) o quadrata (b), ed è alta circa 13 cm.

0°, 90°, 180° e 270°. Dato che questi non sempre venivano rilevati in ogni fotogramma per via delle dimensioni ridotte, si è deciso senza quindi di apporre un marker ogni 10 gradi, per un totale di 36 figure. Questo garantisce che siano rilevati anche più di un marker ogni fotogramma, consentendo anche di correggere alcuni errori conseguenti al calcolo della posizione di un singolo elemento non corretto.
Il sistema di stima del posizionamento basato sulla rotazione funziona però solo sui globi a base circolare, su quelli a basa quadrata è stato necessario apportare qualche modifica. Ad ogni faccia vanno inseriti 15 marker, tutti differenti e con una uguale distanza fra loro. A quelli centrali corrispondono i codici 0, 90, 180 e 270, e hanno la stessa funzione del precedente metodo. È in quelli ai lati che è stato necessario apportare qualche modifica. Quella principale è l'angolo di rotazione da loro indicato, che non sarà più uguale a codice del marker, ma sarà uguale a quello dell'elemento centrale della faccia in cui si trova, mentre la posizione sarà traslata della distanza che c'è fino a quest'ultimo. Questo farà in modo che tutti i marcatori indichino la posizione in cui si trova quello centrale e di quanto questo è ruotato, consentendo quindi di fare una stima del punto di vista basata su più marker.
Una rappresentazione tridimensionale di entrambi i metodi è presente in Figura 5.2 e in Figura 5.3.

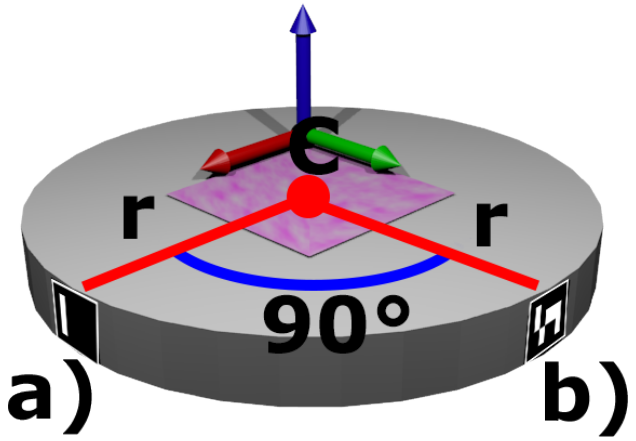

Figura 5.2: L'obiettivo è quello di trovare il punto C partendo dalla posizione dei marker. Per farlo ci si sposta lungo l'asse x di $-r$ e si ruota il sistema di coordinate di un angolo uguale al codice identificativo del marker, che è 0° per a) e 90° per b).

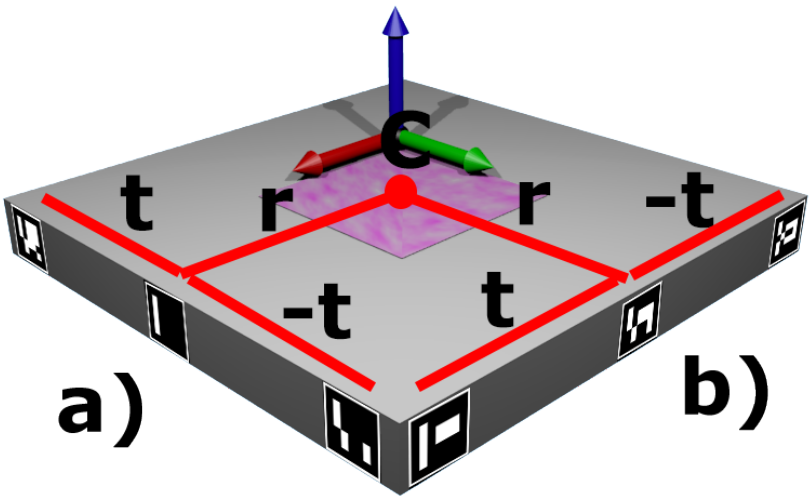

Figura 5.3: L'obiettivo è quello di trovare il punto C partendo dalla posizione dei marker. Inizialmente ci si sposta di $-r$ lungo l'asse x e lungo l'asse y di una distanza t che corrisponde a quella tra il marker rilevato e quello centrale. Infine si fa ruotare tutto il sistema di coordinate così ottenuto di un angolo uguale per tutti quelli appartenenti alla stessa faccia, che è 0° in a) e 90 in b). Questo consente di arrivare sempre al punto C e di avere lo stesso sistema di coordinate per tutti i marker.

5.2 Stima della posizione dell'oggetto nello spazio

I marker non forniscono alcuna informazione su dove si trova l'oggetto cercato, ma, una volta rilevati, è possibile stimare le trasformazioni applicate rispetto al punto di vista. Conoscendo già le dimensioni dei marker e le operazioni di traslazione rotazione e scala da applicare per posizionarli sotto il globo è possibile disegnare un qualunque oggetto digitale in quel punto.
Le trasformazioni geometriche modificano le coordinate di un oggetto per ottenerne un altro simile, ma differente per posizione orientamento e dimensione, alterano quindi la geometria lasciando invariata la topologia. Ogni trasformazione geometrica complessa può essere decomposta in una concatenazione di trasformazioni elementari quali traslazione, rotazione e scala. Ognuna di queste può essere rappresentata da una matrice 4x4 ed è un'operazione lineare, questo consente di utilizzare la combinazione di più matrici per ottenere trasformazioni complesse.
Quando si rileva un marker viene definita una matrice che rappresenta le trasformazioni necessarie per posizionarsi nella sua stessa posizione. OpenGL offre degli strumenti per la gestione di operazioni quali traslazione, rotazione e scala in forma matriciale, quindi è semplice applicare quelle necessarie per raggiungere l'oggetto fisico dal punto in cui si trova il marker e successivamente anche la matrice rilevata precedentemente in modo da ottenere posizione finale dell'oggetto digitale rispetto all'osservatore.

5.2.1 Buffer dei marker

Per aumentare la precisione e ridurre lo sfarfallamento dovuto a spostamenti repentini o al cambio di marker l'applicazione non ne usa uno solo, bensì un buffer di 40 elementi. In ogni fotogramma tutte le matrici dei marker rilevati vengono passate alla funzione di visualizzazione. In seguito viene fatto il prodotto tra ognuna di esse e quella della trasformazioni standard da applicare, e il risultato viene inserito in testa a buffer. Quando arriva il

momento di visualizzare l'oggetto tridimensionale viene utilizzata la media di tutte gli elementi memorizzati come matrice di trasformazione. Questo serve anche ad evitare che ogni fotogramma dia l'impressione che l'oggetto digitale scompaia e riappaia un'altra posizione, simulando quindi uno spostamento fluido e continuo, grazie al fatto che nel buffer saranno presenti anche matrici di quelli precedenti.

Purtroppo si è constatato che non in tutti i fotogrammi viene rilevato un marker, nonostante non ci siano spostamenti significativi del dispositivo, provocando ulteriori sfarfallamenti dell'oggetto digitale. Per ovviare al problema l'applicazione continua ad inserire nel buffer la matrice dell'ultimo marker trovato durate i frame dei successivi $500ms$ dall'ultimo rilevamento. Questo fa in modo che l'applicazione continui a mostrare il globo digitale per mezzo secondo anche se non vede alcun più alcun marker.

5.3 Acquisizione, composizione e visualizzazione dell'immagine

Per acquisire l'immagine viene usata la fotocamera del dispositivo mobile in questione, qui è stato usato un iPhone 5S, che dispone di una videocamera con risoluzione FullHD (1920x1080 pixel) e di un processore 64bit dual-core con una frequenza di $1.3GHz$.

L'analisi dell'immagine per ricercare al suo interno i marker viene fatta sfruttando direttamente il processore del dispositivo, quindi è necessario che sia possibile completare il processo almeno 30 volte al secondo, in modo da garantire una visualizzazione continua senza interruzioni. I principali punti su cui si possono risparmiare risorse sono il calcolo adattivo della soglia tra bianco e nero e la risoluzione dell'immagine.
Nei primi test è stato evidenziato come la risoluzione VGA (640x480 pixel) dia dei problemi nel caso in cui i marker siano di dimensioni troppo piccole. Per questo è stato inizialmente scelto di utilizzare una risoluzione di acquisizione HD (1280x720 pixel) con soglia non adattiva. Un secondo test ha

però evidenziato, che riducendo di circa 10 volte la soglia dell'area minima occupabile dai marker, è possibile rilevarli anche con una risoluzione VGA. È stato quindi scelto di usare quest'ultima e una soglia tra bianco e nero non adattiva. Il vantaggio è che, oltre a velocizzare il processo di rilevamento, è stato pure possibile aumentare il numero dei dispositivi compatibili, in quanto non tutti i sono in grado di acquisire video con risoluzione HD.

Ogni volta che un nuovo frame video viene caricato in memoria dal sistema operativo, l'applicazione effettua il processo di rilevamento dei marker su di esso. Fatto ciò ha tutte le componenti per disegnare un nuovo frame, effettuando i seguenti passi:

1. svuota la scena;
2. imposta la proiezione ortogonale per disegnare lo sfondo;
3. disegna l'ultima immagine ricevuta in una finestra;
4. imposta la visione prospettica utilizzando i parametri di calibrazione della videocamera;
5. per ogni marker rilevato calcola il sistema di coordinate derivante e lo inserisce nel buffer delle trasformazioni;
6. calcola la trasformazione media dagli elementi del buffer e la applica all'oggetto digitale;
7. mostra quello contenuto nel framebuffer della scheda video.

Per svolgere questi compiti è stato usato OpenGL. I frame ripresi dalla fotocamera non vengono infatti mostrati subito all'utente, ma viene creato un piano in un ambiente a 3 dimensioni vuoto (punto 1) frontale al punto di vista a cui viene applicata come texture l'immagine acquisita (punto 2). Per evitare di distorcere la figura è necessario che sia il piano così creato che la finestra di OpenGL abbiano lo stesso aspect-ratio dell'immagine originale, cioè se si sta acquisendo un video in VGA si deve operare in 4:3, mentre se lo

si fa in HD le proporzioni diventano 16:9. Al fine di avere l'immagine perfettamente frontale e delle stesse dimensioni al punto 3 viene attivata la visione ortogonale, che consente di mantenere gli oggetti inalterati anche al variare della distanza, come invece avviene nel mondo reale, dove più si allontana qualcosa più diventa più piccola Figura 5.4.

Il passo successivo consiste nell'impostare la visione prospettica, basati però

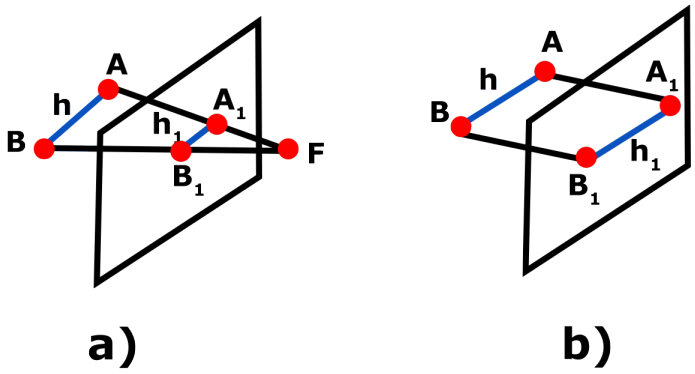

Figura 5.4: Nella proiezione prospettica (a) è presente un punto di fuoco F verso cui ogni punto tende col diminuire della distanza. I punti A e B, distanti tra loro $h > h_1$ tendono ad avvicinarsi tra loro se avvicinati ad F. Nella visione ortogonale (b) invece i punti A e B se riportati sul piano di visualizzazione manterranno comunque la distanza $h = h_1$.

anche sui parametri di calibrazione della fotocamera in questione, che vanno calcolati per ogni dispositivo se si vuole ottenere un risultato perfetto. Per questo lavoro sono stati usati dei parametri presi da libro di Baggio [29]. Gli stessi valori vengono usati anche per generare le matrici di trasformazione dei marker, quindi è importate che siano uguali in entrambi in punti.

Una volta ritornati alla visione prospettica vengono calcolate tutte le matrici di trasformazioni del sistema delle coordinate dei marker rilevati, inserendole in seguito nel buffer. Fatto ciò viene disegnato l'oggetto digitale e applicata la matrice media di tutte quelle presenti nel buffer. L'ultimo passo mostra a video il risultato finale, cioè quello che fin ora è stato caricato

nel buffer della scheda video. La Figura 5.5 mostra graficamente cosa sarà visualizzato alla fine all'interno della finestra. La Figura fa capire anche la necessità che l'aspect ratio della finestra e i parametri di calibrazione sia corretti, altrimenti il globo digitale si sovrapporrebbe a quello fisico nella finestra finale.

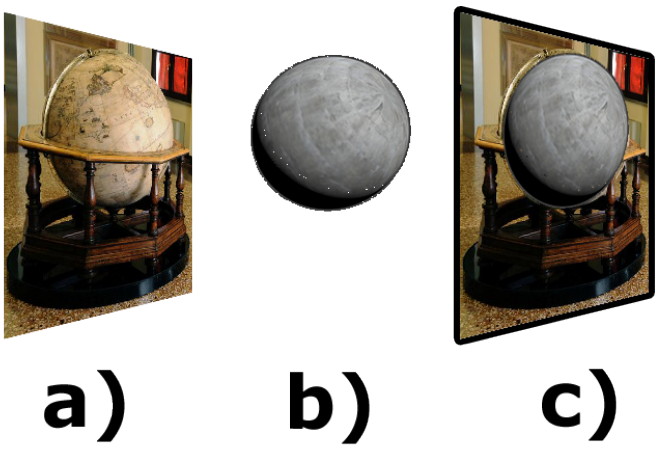

Figura 5.5: Rappresentazione del ambiente 3D creato, viene messa sullo sfondo l'immagine acquisita con la fotocamera (a) usando la proiezione ortogonale. In seguito viene aggiunto l'oggetto tridimensionale (b) e viene posizionato di fronte alla figura, usando però la proiezione prospettica. Il risultato finale è una sovrapposizione delle due (c).

5.3.1 Sezionamento dell'oggetto digitale

Quanto descritto fin ora permette solo di sovrapporre l'oggetto digitale a quello fisico, senza però consentire di vedere al suo interno. Quindi non è ancora possibile vedere la struttura interna ricavata tramite l'utilizzo della TAC. Per ovviare a questo problema è stato sfruttato uno degli strumenti offerti da OpenGL, la gestione della camera. Ogni volta che si sceglie il punto di vista nello spazio tridimensionale è necessario impostare, oltre che posizione e il punto in cui si sta osservando, anche una distanza massima e una minima, ed è proprio quest'ultima che è stata sfruttata per consentire la

visione dell'interno del globo. Facendola variare dalla posizione davanti ad esso fino punto centrale è possibile vederlo come se fosse stato sezionato.

Per consentire di variare gradualmente la distanza è stato deciso di inserire uno slider nell'interfaccia utente, che parte da una posizione in cui si vede il globo nella sua interezza a una in cui se ne vede solo metà. Per garantire un sezionamento omogeneo anche al variare della distanza dell'osservatore dal globo, viene applicata ad un punto nel centro dell'oggetto digitale la matrice media ottenuta da buffer, e calcolata la distanza rispetto la precedente posizione, ottenendo il punto massimo a cui lo slider può arrivare. Per il minimo basta sottrarre il raggio del globo al precedente per ottenere un risultato ottimale.

5.3.2 Evidenziazione delle parti

Prevendendo di associare una descrizione alle varie parti della struttura è nato il problema di come evidenziare quelle citate nel testo. Per farlo è stato scelto di usare il colore rosso, quando è necessario riferirsi ad una componente viene visualizzata solo tale componente della texture, dando l'impressione che la parte dell'oggetto sia come evidenziata.

Dato che nel globo le varie parti sono come racchiuse una dentro l'altra è stata demandata questa duplice funzione allo slider per sezionare l'oggetto, man mano che si sposta verso destra, diminuendo la parte visualizzata, vengono evidenziata parti differenti, partendo dal guscio più esterno fino ad arrivare al perno centrale.

5.4 Architettura dell'applicazione

In questo paragrafo è stata descritta l'architettura della soluzione finale proposta, che è basata su quella proposta da Baggio [29], e sviluppata per un dispositivo iOS, le cui applicazioni contengono sempre almeno un'istanza dell'interfaccia UIViewController. Queste servono a gestire un set di viste che rappresentano l'interfaccia utente.

L'applicazione in questione ha un unico controller della vista che contiene i comandi forniti all'utente e la finestra in cui verrà visualizzata l'immagine acquisita con le relative aggiunte digitali. Come nel diagramma delle classi in Figura 5.6, al suo interno vi sono le tre principali componenti necessarie a far funzionare un'applicazione di questo genere:

- la sorgente del video da analizzare, che può trattarsi di un video pre-registrato o appena acquisito, che fornisce anche i parametri di calibrazione della fotocamera;
- la pipeline di processing, al suo interno vi sono tutti gli strumenti necessari a rilevare i marker a stimare le trasformazioni a loro applicate;
- il motore di visualizzazione, ha il compito di stimare la posizione finale del globo, fare il rendering di tutte le parte e mostrarlo.

La sorgente video serve a ottenere i frame video ripresi con la camera, questo significa che deve essere in grado di scegliere quale periferica di acquisizione utilizzare (anteriore o posteriore), impostare i parametri di acquisizione, come la risoluzione ed raccogliere i frame senza influire sull'andamento dell'UI.
Tutti gli strumenti per l'analisi delle immagini sono stati inclusi nella classe MarkerDetector. L'interfaccia fornita al ViewController è minima, vi è solo un metodo per processare i frame e uno per ottenere i risultati.
L'approccio modulare è voluto per facilitare l'implementazione di soluzioni basate su metodo in questione su piattaforme differenti. Per esempio la classe MarkerDetector è scritta in C++, in modo da poter essere utilizzata su ogni dispositivo che lo supporta, compresi i sistemi desktop. Lo stesso vale per il VisualizationController, il quale usa OpenGL, che è disponibile per una moltitudine di sistemi.

5.4.1 Routine principale

Il processo principale inizia quando la VideoSource riceve un frame video, che a sua volta lo passerà al ViewController, il quale eseguirà i seguenti passi:

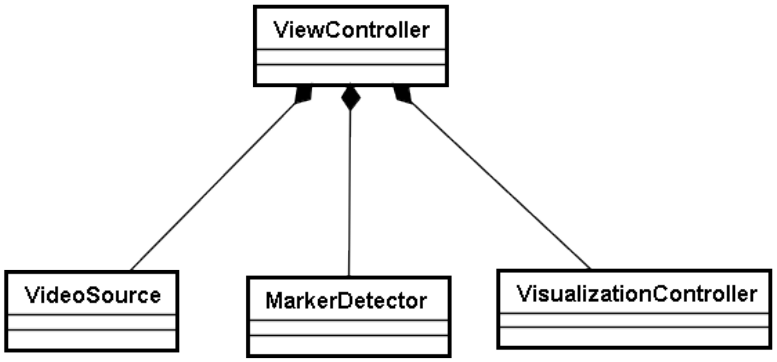

Figura 5.6: Le quattro classi fondamentali dell'applicazione. Il ViewController si occupa di gestire l'interfaccia utente e di mettere in comunicazione le altre componenti. Il VideoSource fornisce l'input video e i dati di calibrazione della camera. Il MarkerDetector serve a rilevare i marker all'interno delle immagini e fornisce le matrici di traslazione. Infine il VisualizationController fa il rendering della scena finale, oltre che a mantenere il buffer delle matrici di trasformazione.

1. invia il nuovo frame al VisualizationController;

2. demanda al MarkerDetector il compito di analizzarlo per rilevare dei marker;

3. invia i marker rilevati al VisualizationController;

4. avvia il rendering della scena.

Il primo e l'ultimo punto differiscono tra loro perché nel primo viene semplicemente inviata l'immagine presa nel buffer del dispositivo al VisualizationController, il quale la inserirà nel buffer video. Questo verrà visualizzato solo nell'ultimo punto, quando anche le altre componenti della scena saranno pronte.
Il secondo step ha il compito di processare l'immagine acquisita e rilevare i marker al suo interno, i quali verranno poi passati al VisualizationController durante il terzo punto.
Il diagramma di sequenza in Figura 5.7 mostra tutte le principali operazioni svolte durante la fase di processing dell'immagine. La routine ha inizio quando il VideoSource riceve l'immagine dalla fotocamera, passandola al ViewController. Questo invia l'immagine al VisualizationController come sfondo e al MarkerDetector per analizzarla. Nel passo successivo vengono presi i marker rilevati e passati al VisualizationController. Infine viene disegnato il tutto. Da notare che ogni comunicazione fra il ViewController e VisualizationController è asincrona, questo per garantire che la visualizzazione dei frame non sia rallentata dal rilevamento dei marker.

5.4.2 Disegno della scena

In Figura 5.8 è presente un diagramma di attività che mostra tutti i passi eseguiti dal VisualizationController prima di disegnare effettivamente la scena. Il processo inizia quando si riceve un comando di draw da ViewController, come primo passo viene disegnato lo sfondo utilizzando l'ultima immagine caricata. In seguito viene fatto un controllo se sono stati rilevati marker se la risposta è negativa ed è stato lo stesso per gli ultimi $500ms$ la routine termina, se invece è stato rilevato un marker meno di mezzo secondo prima viene re-inserita la sua matrice nel buffer e disegnato l'oggetto tridimensionale utilizzando la media delle matrici presenti al suo interno.
Se invece sono stati rilevati dei marker, per ognuno di essi vengono prima applicate le operazioni necessarie a trovare il centro della base ed in seguito

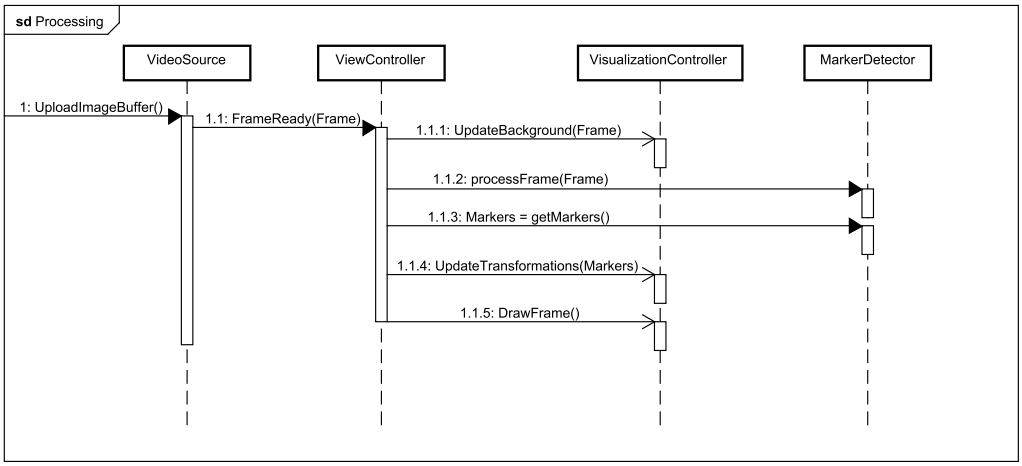

Figura 5.7: La routine di processing dei un frame inizia quando questo viene caricato nel buffer di memoria video (1). L'immagine viene quindi passata al ViewController (1.1), che la invia al VisualizationController (1.1.1) e al MarkerDetector (1.1.2). In seguito prende il risultato dell'analisi (1.1.3) e lo invia al VisualizationController (1.1.4). Come ultimo passo dice a quest'ultimo di visualizzare il tutto (1.1.5).

la matrice di trasformazione collegata al marcatore in questione, il risultato viene inserito nel buffer.

Quando si disegna l'oggetto digitale vengono anche applicate delle trasformazioni costanti relative ed esso, come una traslazione nell'asse z o un resize per farlo coincidere con la controparte fisica (che dipende sia dalla distanza dei marker da questo, che dalle loro dimensioni fisiche).

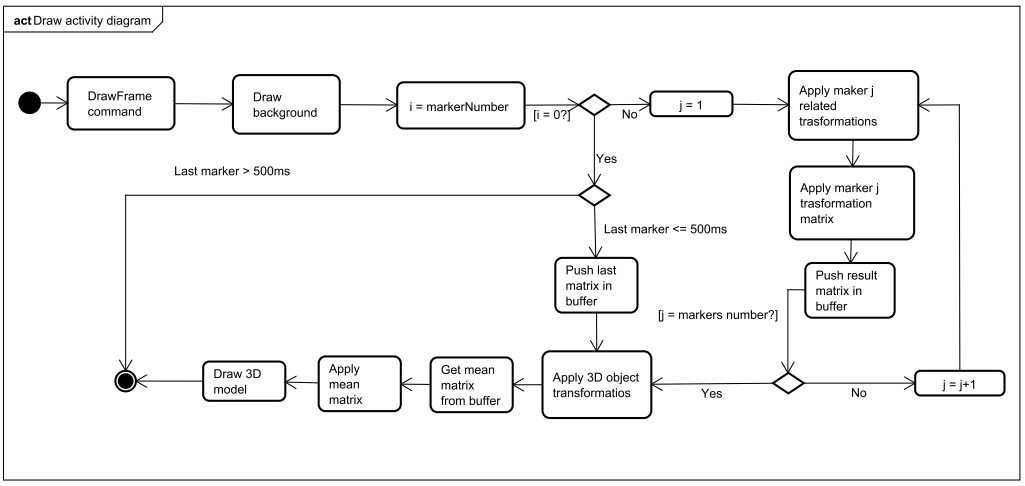

Figura 5.8: Diagramma di attività che mostra i passi seguiti dal VisualizationController quando riceve un comando di draw.

5.5 Interfaccia grafica

Un elemento molto significativo e che spesso decreta il successo di un'applicazione è costituito da come questa si interfaccia con l'utilizzatore finale. È infatti importate che sia semplice comprenderla ed interagire con essa, altrimenti l'utente potrebbe non usarla affatto.

La prima versione dell'interfaccia grafica prevedeva uno spash screen bianco (durante il caricamento e l'apertura della fotocamera) con al centro il nome dell'applicazione, la finestra principale è mostrata in Figura 5.9. Dato che spesso le applicazioni che utilizzano la fotocamera hanno un funzionamento di landscape è stato scelto di attenersi a questo stile, inoltre sono stati inseriti due soli controlli utente, uno switch per attivare le descrizioni a sinistra e uno slider per sezionare l'oggetto tridimensionale in verticale a destra, quindi alla portata dei pollici. Infine, al centro, è stata prevista la finestra in cui viene mostrato il risultato finale.

Il passo successivo è stato quello di far testare l'applicazione a persone non collegate col progetto, questo ha messo in risalto due punti. Il primo ha sottolineato il fatto che i controlli non sono poi così intuitivi senza al-

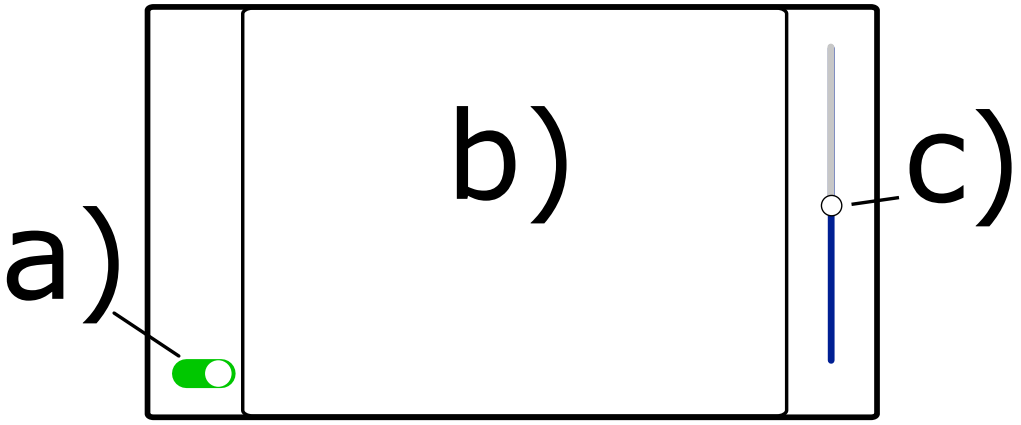

Figura 5.9: Prima interfaccia grafica dell'applicazione, è previsto un funzionamento in landscape e a suo interno vi sono 3 componenti principali: uno switch (a) che serve ad attivare e disattivare le descrizioni, una finestra centrale (b) in cui verrà mostrata la scena e uno slider verticale (c) per sezionare l'oggetto digitale.

cuna descrizione, il secondo invece che la scelta del landscape non è quella corretta, infatti nel globo è maggiore l'altezza rispetto la larghezza, quindi è possibile vederlo più da vicino se si utilizza un'applicazione di tipo portrait, cioè tenendo il dispositivo in verticale.

È stata quindi studiata una nuova versione dell'interfaccia grafica, questa volta in verticale, di cui in Figura 5.10 è presente una sua rappresentazione. La finestra di visualizzazione è stata posizionata nella parte alta, a contatto col bordo, è stata aggiunto un label con scritto "descrizione" accanto lo switch e lo slider è stato girato di 90 gradi in posizione orizzontale, ed entrambi i controlli sono stati infine posizionati nella parte bassa della finestra. In questo modo è possibile tenere il dispositivo con una mano in posizione portrait ed utilizzare i controlli col pollice senza che questo passi mai davanti alla finestra di visualizzazione. Altro vantaggio dato è che adesso la scena si trova sopra la fotocamera, aumentando la sensazione di visione attraverso di essa, anche se di un quantitativo minimo.

Infine sono stati fatti altri due accorgimenti, il primo è stato quello di

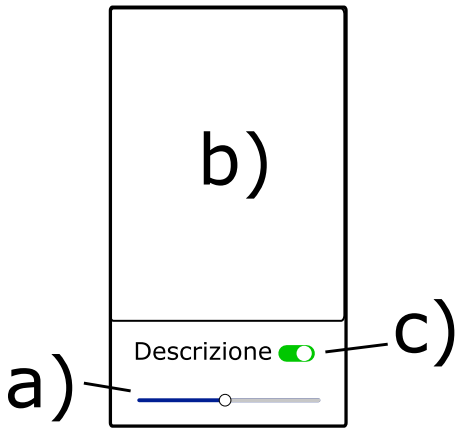

Figura 5.10: Seconda interfaccia grafica dell'applicazione, è previsto un funzionamento in portrait e a suo interno vi sono 3 componenti principali: uno switch (c) che serve ad attivare e disattivare le descrizioni, una finestra centrale (b) in cui verrà mostrata la scena e uno slider orizzontale (a) per sezionare l'oggetto digitale.

sostituire lo spash-screen con una descrizione dell'interfaccia, in modo da sfruttare i tempi di caricamento con qualcosa di utile. Il secondo è stato quello di aggiungere delle icone allo slider nei punti di massimo e di minimo, come in Figura 5.11, dove a sinistra è stato posizionato il globo per intero, a destra il globo sezionato fino a metà. Quest'ultimo punto consente capire in maniera intuiva a cosa serve lo slider in questione, rendendone ancora più semplice l'utilizzo.

Figura 5.11: Per rendere più intuitivo l'utilizzo dello slider sono state aggiunte due icone, una a destra e l'altra a sinistra che rappresentano rispettivamente un globo sezionato fino a metà e uno intero.

Conclusioni e sviluppi futuri

In questa tesi ci si è occupati di analizzare e sviluppare diverse tecniche di realtà aumentata o mista, al fine di poterle applicare ad un caso specifico, il rilevamento di un globo fisico in un video in realtime e la sovrapposizione di una sua controparte digitale.

Per poter rilevare l'oggetto fisico sono state testate tre diverse tecniche:

- la ricerca di una simmetria radiale;
- la ricerca di caratteristiche locali dell'oggetto;
- l'utilizzo di marker per stimare la posizione finale.

La scelta su quale utilizzare si è basata sui costi delle varie soluzioni e su quanto esse risultino invasive all'interno dell'ambiente in questione. Per rendere più economica l'esperienza e evitare l'introduzione di hardware all'interno della stanza è stato deciso di fornire un applicativo per smartphone accettando così le limitazioni che ne conseguono, che riguardano risorse computazionali e memoria disponibile.

Il primo metodo si è dimostrato in grado di rilevare correttamente il globo in molte situazioni con i corretti parametri di inizializzazione. È stato scartato però perché non fornisce alcuna informazione sulla rotazione dell'oggetto fisico e sul punto di osservazione, inoltre i parametri di input variano molto in base alla distanza e all'illuminazione della scena, rendendo quindi necessaria una calibratura troppo complessa.
Il secondo metodo consiste nella ricerca di caratteristiche locali dell'oggetto fisico note, al fine di stabilire dove questo si trovi e come sia orientato. Il

processo si è però rilevato troppo oneroso per uno smartphone, sia per via delle capacità di calcolo spesso ridotte che, soprattutto, per via del fatto che veniva richiesta troppa memoria per mantenere un dataset di tali caratteristiche, che vanno memorizzate per diversi livelli di scala dell'oggetto, cioè per distanze differenti. Inoltre anche in questo caso i cambi di illuminazione hanno rappresentato un problema.
Il terzo metodo prevede l'aggiunta di alcuni marcatori facilmente riconoscibili all'interno della scena, motivo per il quale era stato inizialmente scartato. Dopo alcuni accorgimenti è stato però possibile minimizzare l'effetto visivo dato dalla presenza di questi elementi rendendoli più piccoli e applicandoli su strutture già esistenti senza ostruire la visione delle opere d'arte. Inoltre i marker forniscono anche informazioni sulla posizione dell'osservatore rispetto all'oggetto fisico, rendendo semplice l'aggiunta di quello digitale alla scena. È stato quindi deciso di utilizzare questo metodo per all'interno dell'applicazione finale.

L'ultimo punto svolto è stato quello di progettare e implementare l'applicazione su un dispositivo mobile iOS. Per farlo sono stati utilizzati strumenti come OpenCV per rilevare i maker e OpenGL per visualizzare il risultato finale, facendo anche attenzione a fornire soluzioni differenti per i tipi di base del globo analizzati.

Il progetto può però essere ampliato e migliorato, prevendendo anche l'uso di un hardware specifico come Intel RealSense o Microsoft Hololens. Inoltre ci sono molte altre opere d'arte, come dei dipinti, che mostrano parti non visibili o cancellate se viste attraverso una TAC, un esempio è presente in Figura C.1, dove si può notare come un dipinto di Giulio Romano sia in realtà attribuibile a Raffaello. Lo stesso processo può essere utilizzato per i sarcofagi, per i quali è possibile vedere ciò che si trova al loro interno, come in Figura C.2. Un possibile sviluppo potrebbe quindi prevedere l'aggiunta di altre opere d'arte all'interno dell'applicazione, e tenendo conto anche che è molto semplice identificare oggetti bidimensionali utilizzando le caratteristiche locali, i dipinti quindi non necessiterebbero nemmeno dei marker,

creando così un esperienza più avvolgente ed interattiva per i visitatori dei musei, di modo da valorizzare ulteriormente il nostro patrimonio culturale.

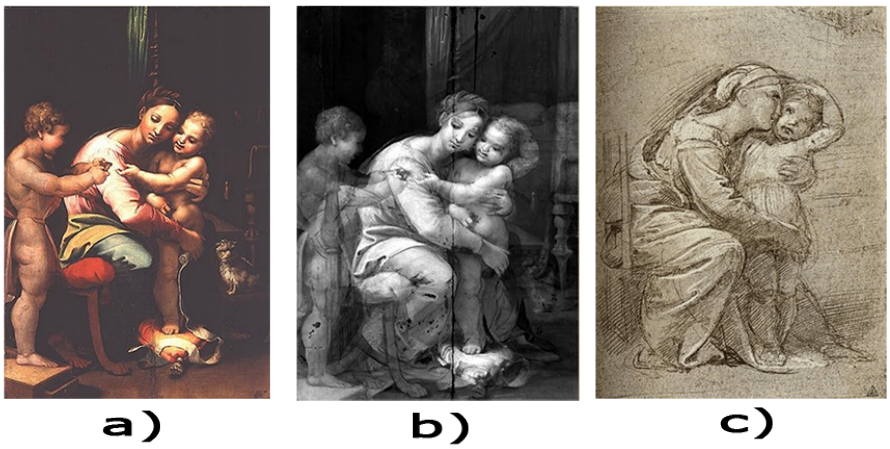

Figura C.1: Madonna con Bambino e San Giovannino conservato alla Galleria Borghese di Roma inizialmente attribuito a Giulio Romano (a). Tramite una TAC (b) è stato possibile capire che il dipinto è formato da due strati e che San Giovannino è stato aggiunto in seguito, oltre al fatto che il braccio del bambino era in un'altra posizione. Questo strato inferiore è attribuibile a Raffaello da un suo disegno su carta conservato all'Ashmolean Museum di Oxford (c).

Fonte immagine http://www.accademiabolognascienzetecniche.it.

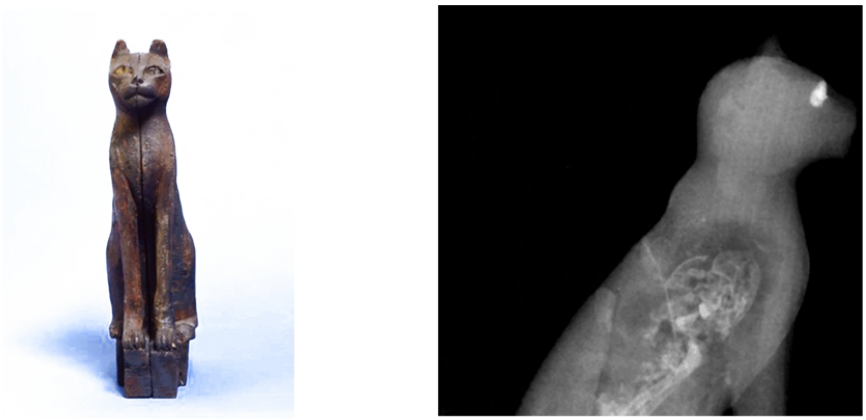

Figura C.2: Un sarcofago egizio di un gatto conservato al museo archeologico di bologna (a), è stato possibile vedere lo scheletro dell'animale utilizzando una TAC (b).

Fonte immagine http://www.accademiabolognascienzetecniche.it.

Bibliografia

[1] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[2] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," in *Photonics for Industrial Applications*, pp. 282–292, International Society for Optics and Photonics, 1995.

[3] A. Barry, J. Trout, P. Debenham, and G. Thomas, "Augmented reality in a public space: The natural history museum, london," *Computer*, no. 7, pp. 42–47, 2012.

[4] J. Rekimoto, "Matrix: A realtime object identification and registration method for augmented reality," in *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific*, pp. 63–68, IEEE, 1998.

[5] N. I. Adhani and R. D. R. Awang, "A survey of mobile augmented reality applications," in *1st International Conference on Future Trends in Computing and Communication Technologies*, pp. 89–96, Citeseer, 2012.

[6] M. P. Morigi, F. Casali, A. Berdondini, M. Bettuzzi, D. Bianconi, R. Brancaccio, A. Castellani, V. D'Errico, A. Pasini, A. Rossi, *et al.*, "X-ray 3d computed tomography of large objects: investigation of an ancient globe created by vincenzo coronelli," in *Optical Metrology*,

pp. 66180A–66180A, International Society for Optics and Photonics, 2007.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[9] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of applied statistics*, vol. 21, no. 1-2, pp. 225–270, 1994.

[10] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–511, IEEE, 2001.

[13] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 850–855, IEEE, 2006.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011.

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, pp. 430–443, Springer, 2006.

[16] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[17] C. Harris and M. Stephens, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, p. 50, Citeseer, 1988.

[18] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: efficient indexing for high-dimensional similarity search," in *Proceedings of the 33rd international conference on Very large data bases*, pp. 950–961, VLDB Endowment, 2007.

[19] I. Gordon and D. G. Lowe, "What and where: 3d object recognition with accurate pose," in *Toward category-level object recognition*, pp. 67–82, Springer, 2006.

[20] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 1000–1006, IEEE, 1997.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C*, vol. 2. Cambridge university press Cambridge, 1996.

[23] P. Martin, B.-J. Ho, N. Grupen, S. Muñoz, and M. Srivastava, "An ibeacon primer for indoor localization: Demo abstract," in *Proceedings*

of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, BuildSys '14, (New York, NY, USA), pp. 190–191, ACM, 2014.

[24] G. Bradski *et al.*, "The opencv library," *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

[25] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with opencv," *Communications of the ACM*, vol. 55, no. 6, pp. 61–69, 2012.

[26] G. Loy and A. Zelinsky, "Fast radial symmetry for detecting points of interest," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 8, pp. 959–973, 2003.

[27] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding," *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[28] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP (1)*, vol. 2, 2009.

[29] D. L. Baggio, *Mastering OpenCV with practical computer vision projects*. Packt Publishing Ltd, 2012.

[30] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[31] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[32] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[33] R. Silva, J. Oliveira, and G. Giraldi, "Introduction to augmented reality," *National Laboratory for Scientific Computation, Av. Getulio Vargas*, 2003.

[34] A. OLWAL, "An introduction to augmented reality," 2009.

Ringraziamenti

Ringrazio tutti i docenti che mi hanno seguito e mi hanno aiutato durante i miei studi sia durante il corso di laurea magistrale che triennale.

Un ringraziamento speciale va alla mia famiglia per avermi sostenuto sia economicamente che, soprattutto, moralmente, alla mia ragazza per aver sopportato "le mie continue perdite di tempo" come lei le ha definite ed ai miei amici e colleghi per il sostegno e l'auito da loro a me fornito.

Così si conclude un periodo importate della mia vita in cui l'università mi ha permesso di crescere, imparare ciò che so e maturare, fornendomi gli strumenti per affrontare la mia vita futura. Adesso volendo citare Orazio *nunc est bibendum, nunc pede libero pulsanda tellus*. E che tutti gli scaramantici lo sappiano, io sulla torre ci sono salito il primo anno.