
# UNIVERSITÀ POLITECNICA DELLE MARCHE

# FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

Studio e applicazione di algoritmi per la realtà aumentata per lo sviluppo di un'applicazione finalizzata al Retail.

Study and application of augmented reality algorithms for the development of a retail oriented app.

Relatore:

Prof. Aldo Franco Dragoni

Tesi di Laurea di:

Leonardo Galeazzi

Correlatore:

Dott. Paolo Sernani

*Anno Accademico 2019/2020*

# INDICE

## Premessa

È chiamata realtà aumentata un'esperienza del mondo circostante arricchita di oggetti non reali bensì generati da un calcolatore. I fini di questa sovrapposizione posso essere molteplici con il principale quello di fornire maggiori informazioni a chi osserva tale mondo aumentato. Lo scopo di questa tesi è illustrare il funzionamento delle tecnologie che rendono possibile la realtà aumentata e di illustrare lo sviluppo di un'applicazione basata su di essa.

## Introduzione

In un romanzo del 1901 dello scrittore L.Frank Baum(1) un ragazzo è in grado tramite degli occhiali "elettrici" di conoscere il carattere della persona che ha di fronte. Sulla fronte dei buoni infatti appare la lettera "B" su quella dei cattivi la lettera "C", sui saggi la lettera "S" e sugli ingenui la lettera "I". Sessant'anni dopo, nel 1961, l'informatico Ivan Sutherland, successivamente professore di ingegneria elettronica ad Harvard e di informatica al Caltech, inventa "La Spada di Damocle", il primo visore per la realtà virtuale. Essendo questo in grado di far trasparire almeno in parte il mondo reale all'utilizzatore è di fatto considerabile allo stesso tempo il primo visore per la realtà aumentata, essendo qui che giace la differenza tra i due concetti. La prima applicazione del dispositivo fu quella di mostrare al suo utente un cubo sospeso a mezz'aria di fronte a lui. Nel 1992 l'inventore Louis B. Rosenberg formula il concetto di perceptual overlay. Egli afferma che qualcosa come un righello, uno strumento che aiuta il suo utilizzatore a compiere un azione come disegnare una linea dritta può essere inteso come un esempio di perceptual overlay, di sovrapposizione percettiva. In quel caso la percezione sensoriale che è stata sovrapposta al mondo afferma essere quella reale, tattile e visiva, del righello stesso che ha fornito supporto per compiere l'azione di disegnare la riga. L'autore teorizza poi come allo stesso modo una sovrapposizione percettuale di tipo virtuale potrebbe essere di utilità per svolgere una determinata mansione. Usando una metafora e riferendosi a questi oggetti virtuali con il nome di Virtual Fixtures nota come non avendo una massa questi sono esenti dai limiti imposti dalla fisica e meccanica di un oggetto reale e possono essere modificati a piacimento e istantaneamente da un computer [31]. L'esperimento che seguì il concetto fu tuttavia di diversa natura da quanto teorizzato. In questo un utente comanda un esoscheletro composto da un braccio robotico e utilizza un visore composto di lenti. Tramite il sistema ottico del visore l'utente pecepisce l'illusione della presenza del braccio robotico al posto del proprio e sotto tale illusione effettua il semplice test di inserire delle piccole sbarre di metallo all'interno di alcuni fori. Fu notato come sotto l' illusione la produttività nell'eseguire il compito mediante l'esoscheletro aumentò del 70%. Sin da allora insieme ai notevoli progressi della tecnologia sono stati molti anche i progressi fatti nel mondo dell'Augmented Reality. Aziende come Google e Microsoft hanno sviluppato i loro visori per la realtà aumentata, il cui funzionamento sarà brevemente coperto.

# Capitolo 1: Realtà aumentata: storia, concetto e requisiti

## 1.1 Realtà aumentata

Nel suo Survey of Augmented Reality [4] Azuma ritiene come fosse limitativo definire la realtà aumentata legandola imprescindibilmente al concetto di Head Mounted Display ovvero di visore. Il concetto era infatti nato a partire da questa tecnologia e per anni legata ad essa era anche stata la ricerca in merito. Nel tentativo di dare allora una definizione più generale e non ancorata a nessun

specifico hardware Azuma definì un sistema per l'Augmented Reality come un sistema avente tre caratteristiche:

1-La generale capacità di combinare il virtuale al reale.

2-La capacità di far interagire un utente con il mondo aumentato in tempo reale.

3-La capacità di registrare accuratamente nel mondo oggetti virtuali.

Di particolare interesse nella realtà aumentata è la capacità di far sì che gli oggetti inseriti si integrino con il mondo in modo da risultare quanto più indistinguibili da esso.

## 1.2 Requisiti

Per implementare il concetto di realtà aumentata è necessario l'utilizzo di una vasta gamma di tecnologie hardware e software come anche l'utilizzo di appositi algoritmi. La necessità di visualizzare oggetti virtuali proiettati sul mondo reale richiede ovviamente l'utilizzo di un display e di dispositivi di input con il principale una fotocamera. Tramite l'utilizzo di una fotocamera è infatti possibile ottenere un video feed in tempo reale del mondo e sul quale è possibile poi agire inserendo delle entità virtuali. Non di seconda importanza sono un processore in grado di effettuare le opportune operazioni di calcolo e dei sensori con i quali è possibile, insieme alla fotocamera, riconoscere le caratteristiche dell'ambiente inquadrato. Un riconoscimento di queste caratteristiche permette ,ad esempio, l'individuazione di piani fisici, come un muro, eventualmente designabili poi come la posizione di un oggetto virtuale da mostrare all'utente. I requisiti hardware sono allora complessivamente tre:

1-Un processore

2-Un display

3-Sensori

## 1.3 Display e apparato computazionale

Allo stato attuale della tecnologia fare una distinzione tra le tecnologie utilizzate come display implica anche effettuare una distinzione tra gli strumenti computazionali. Di particolare rilevanza infatti sia per mostrare all'utente il mondo aumentato sia per effettuare il calcolo che permette ciò sono due strumenti: gli head mounted displays (HMS), visori per la realtà aumentata e gli hand-held displays ovvero smartphones.

*Figura 1: A sinistra un visore per la realtà aumentata, a destra un comune smartphone.*

### 1.3.1 Head Mounted Displays

Le tecnologie utilizzate come display nel mondo della realtà aumentata ricadono sotto due tipologie: Ottiche e Video. Generalmente considerata la meno costosa e più facilmente implementabile la variante ottica presenta un proiettore montato sul dispositivo indossato che proietta l'immagine virtuale su un combinatore ottico. Il combinatore ottico è in parte trasparente, permettendo quindi all'utente di vedere il mondo reale e in parte riflettente, in grado ovvero di riflettere all'occhio umano l'oggetto virtuale proiettato dal proiettore. Lo scopo è ovviamente quello di far sì che l'occhio percepisca le due realtà come una sola. I parametri di interesse in questo tipo di tecnologia sono gli indici di riflessione e trasmissione del combinatore ottico come anche, nel caso più sofisticato, la scelta di quali lunghezze d'onda della luce riflettere e quali trasmettere [4].

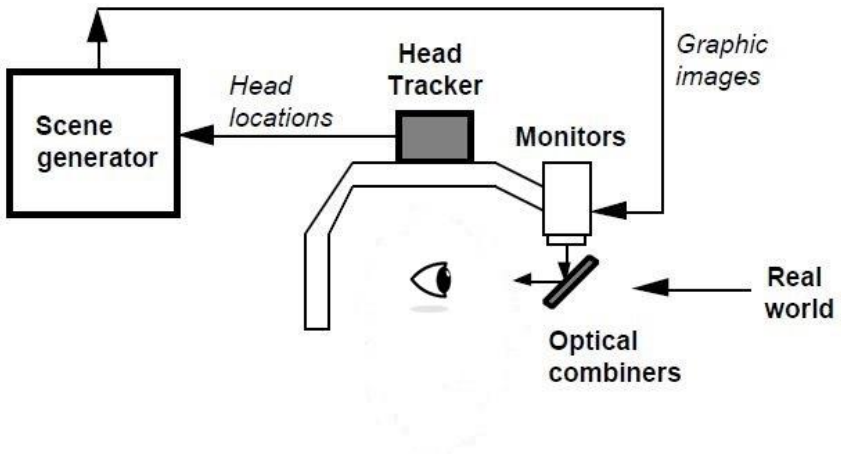


*Figura 2: Esempio di variante ottica di un HMS.*

Nel caso della variante video invece l'HMS è chiuso , non è possibile vedere il mondo circostante.È presente una fotocamera montata sul visore che acquisisce un feed live dell'ambiente di fronte all'utente, il video viene poi modificato in tempo reale per aggiungere le entità virtuali

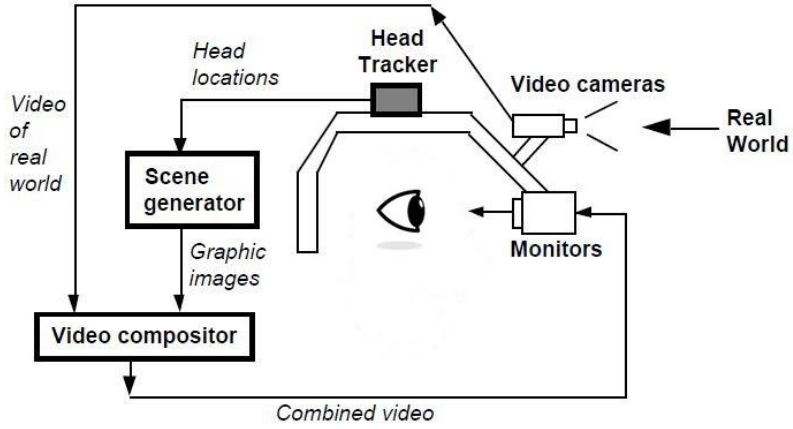


*Figura 3: Esempio di variante video di un HMS.*

e viene restituito all'utente su un monitor posto all'interno dell'apparato. Questo sistema rispetto all'ottico presenta il problema di un ritardo nel mostrare all'utente la versione finale del mondo aumentato. Sia l'acquisizione del video infatti sia la generazione del contenuto virtuale presentano ritardi che vanno a sommarsi e possono potenzialmente causare un'esperienza di ritardo significativa.

## 1.4 Problematiche principali

Dal punto di vista concettuale, sono due le questioni da affrontare nella realizzazione di un'applicazione di realtà aumentata: il "Tracking" e la "Registration". Queste sono interdipendenti e vanno entrambe risolte qualunque sia la particolare scelta implementativa fatta tra le tecnonolgie e algoritmi legati alla realtà aumentata.

### 1.4.1 Registration

Con il termine registration si intende il corretto allineamento dell'oggetto virtuale nel mondo reale per garantire l'illusione della coesistenza tra mondo reale e virtuale. La grande capacità dell'occhio umano nel distinguere dettagli rende complicato il problema della registration. Va tuttavia notato come il grado di accuratezza richiesto per la registration dipende dall'utilizzo inteso dell'applicazione facentene uso. In un'applicazione della realtà aumentata alla chirurgia è chiaro come il grado di accuratezza della registration di un oggetto finalizzato alla diretta assistenza all' operazione chirurgica debba essere estremamente elevato. Meno stringente è invece il requisito nel caso di utilizzo dell'applicazione per mostrare informazioni all'interno di una finestra virtuale posta a fianco di una statua in un contesto educativo. Si parla di errore di registration per descrivere l'offset visivo tra la posizione assunta dall'oggetto virtuale e quella intesa. Azuma [4] spiega come gli errori dal punto di vista della registration possono essere di tue tipologie: statici o dinamici. I primi sono errori che si presentano quando il punto di vista dell'utente e la posizione effettiva degli oggetti rimangono

fermi. I dinamici avvengono invece nel momento in cui c'è movimento. Un banale motivo per la presenza di un errore di registration è un erroneo tracking.

## 1.5 Tracking

Una comune necessità nelle applicazioni di realtà aumenta è stimare la "pose" ovvero posizione e rotazione della fotocamera all'interno dell'ambiente. Questo può essere fatto a partire da una singola immagine, in tal caso si parla di Detection, con l'oggetto quello appunto identificato. Nel caso tuttavia più complesso di una sequenza temporale di immagini come un video feed si parla di Tracking [1]. Questo può a sua volta essere di tue tipologie. Il tracking può ovvero essere effettuato a partire da ogni frame senza l'utilizzo della pose ottenuta dal frame precedente, in tal caso si parla di Tracking by Detection. Questo metodo ha impiego in genere quando i frame presentano immagini non correlate, un esempio è la presenza di un ambiente rapidamente variabile che sia per movimento della fotocamera o meno. Nel caso invece in cui lo spostamento della fotocamera è minimo tra frame successivi viene utilizzato un metodo di calcolo della sua attuale pose che si basa sul risultato del calcolo nel frame precedente. In questo caso si parla di Tracking by Tracking. Entrambi i metodi possono essere utilizzati per soddisfare il requisito del Tracking in un'applicazione di realtà aumentata. Le tecniche per effettuare tracking vengono generalmente suddivise in tre categorie: sensor based, vision based e hybrid type based [5].

### 1.5.1 Sensor-based Tracking

Il progresso fatto in ambito della microelettronica ha fatto sì che sia ormai comune la presenza di IMUs, sistemi di misura inerziali all'interno di dispositivi come smarphones[6]. Una IMU è composta principalmente da tre sensori: accelerometro, magnetometro e un giroscopio.

**Accelerometro**: L'accelerometro è un sensore in grado di misurare forze e quindi accelerazioni a cui è sottoposto il dispositivo che lo contiene. Questo misura in ogni momento la somma di due accelerazioni, la gravitazionale e l'eventuale accelerazione imposta sul dispositivo. Nei dispositivi mobili viene preso come riferimento il piano del dispositivo e vengono misurate le forze lungo i tre assi x y e z.

**Giroscopio:** Un giroscopio ha il compito di calcolare e restituire un vettore tridimensionale rappresentante la velocità angolare del dispositivo attorno ai i tre assi del suo sistema di riferimento. Nonostante i progressi fatti nella tecnologia dietro al funzionamento di questo tipo di sensori essi rimangono soggetti ad un errore non trascurabile specialmente in ambito di smarphones e miscroelettronica.

**Magnetometro**: Il magnetometro misura l'intensità del campo magnetico pensabile come un vettore tridimensionale che punta il nord magnetico della Terra. Questo tipo di sensori presenta un errore dovuto in modo significativo all'interferenza causata dall'ambiente circostante. La presenza di elementi aventi un proprio campo magnetico statico o una distorsione del campo magnetico nell'ambiente in cui è presente il giroscopio possono essere causa di errore. Nel primo caso l'errore

prodotto è un semplice offset e può essere compensato nel secondo caso la compensazione può risultare più complessa .

Nel caso di dispositivo mobile con "sensor-based tracking" si intende un processo per il quale la pose del dispositivo è ottenuta inizialmente dall'utilizzo di informazioni GPS e una bussola che danno rispettivamente posizione e orientamento. Il movimento viene poi successivamente tracciato grazie alle informazioni fornite dalla IMU nel dispositivo.

### 1.5.2 Vision-based Tracking

Le tecniche che ricadono nella categoria Vision-based prevedono l'utilizzo di un processamento di immagini per individuare la pose della fotocamera relativamente a un certo sistema di riferimento solidale con il mondo, per effettuare ovvero il tracking della fotocamera che ha acquisito tali immagini [5]. Questa tecnica fa allora riferimento alla disciplina della Computer Vision uno dei quali obiettivi è capire come ottenere informazioni 3d a partire da dati 2d tempo varianti come la sequenza di frame componente un video acquisito da una fotocamera [7]. Il vision-based tracking in quanto facente parte del mondo della computer vision è allora a sua volta divisibile in due categorie: feature-based tracking e model-based tracking [8]. Il primo prevede di individuare elementi 2d come primitive geometriche, punti, segmenti, cerchi, contorni di oggetti. L'obiettivo di questo metodo è trovare una corrispondenza tra gli elementi 2d individuati e delle coordinate nel mondo tridimensionale [5]. Mediante poi un apposito calcolo è possibile trovare da questa informazione la pose della fotocamera nello spazio. Il secondo metodo invece prevede l'utilizzo a priori di un modello 2d o 3d degli oggetti da individuare. Nonostante questa distinzione ne esiste anche una seconda, più comune nel mondo della realtà aumentata che prevede di distinguere tra Marker Tracking e Markerless Tracking. Il Marker tracking fa affidamento sulla presenza di figure geometriche predesignate e inserite nell'ambiente, detti markers, per stimare la pose della camera. I markers vengono riconosciuti e grazie alle loro caratteristiche è possibile mediante opportuni algoritmi di computer vision ottenere la pose della fotocamera. In questo tipo di approccio la capacità di effettuare la stima e la precisione con cui viene fatto dipende dalla tipologia e numero di markers utilizzati [9].Per ottenere un tracking che permetta di effettuare una registration dell'oggetto virtuale con preciso allineamento al mondo reale è in genere necessario un grande numero di markers presenti nell'ambiente e inquadrati dalla fotocamera. Il numero di markers richiesti diminuisce con la diminuzione della precisione richiesta per il tracking.

*Figura 4:Due dei markers utilizzati dalla libreria ARToolKit.*

Una delle prime librerie per lo sviluppo di software per la realtà aumentata e a fare uso del Marker Tracking è stata la libreria ArToolKit, in figura (4) si hanno due dei markers usati in essa. Ogni marker nel caso di ArToolKit presenta una particolare figura al suo interno. La stima della pose è fatta grazie ad un algoritmo che confronta regioni geometriche invarianti di tali figure inquadrate con quelle

delle registrate a priori nel sistema. Nel caso di dispositivi mobili il processo del marker tracking passa attraverso tre fasi [6]:

1) **Marker detection**: Il marker viene cercato nelle regioni dello spazio in cui si ritiene vi sia maggiore probabilità di trovarlo e tali regioni vengono selezionate.
2) **Marker identification**: Le regioni selezionate vengono analizzate e il marker eventualmente riconosciuto.
3) **Marker tracking e pose estimation:** Vengono individuate quattro coppie di spigoli nel marker e viene eseguita una trasformazione omografica per effettuare il tracking.

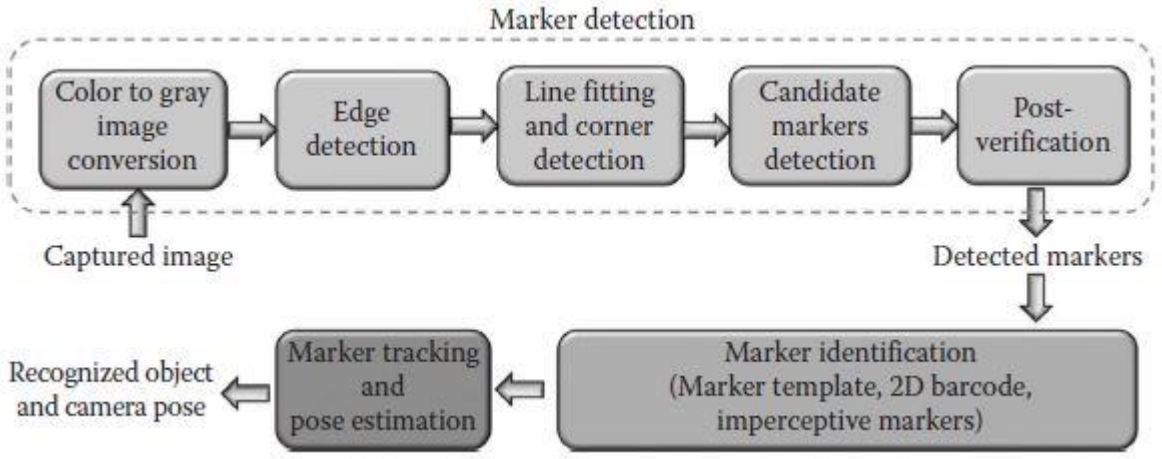


*Figura 5: Lifecycle di un processo di Marker Tracking.*

La figura (5) mostra il processo completo di tracking passante per i tre stadi descritti. Nella prima fase di detection vengono utilizzate tecniche per eliminare falsi positivi e per evitare quindi che le relative regioni vengano processate inutilmente nelle fasi successive del processo. Lo svantaggio notevole di un approccio marker based è l'elevata sensibilità all'occlusione. L'altra tipologia di tracking, la markerless non prevede l'utilizzo di appositi markers ma consiste nello stimare la pose a partire da caratteristiche intrinseche dell'ambiente, linee, spigoli, punti[5]. Sebbene computazionalmente più complessa essa offre maggiore robustezza nel tracking rendendo possibile questo anche in condizioni in cui parte della visuale della fotocamera viene occlusa. L'occlusione dei markers nel caso marker based infatti comporta una perdita del tracking della fotocamera [10]. Un primo utilizzo del Markerless tracking fu quello di Park, You e Neumann [10]. Nel loro paper illustrano il funzionamento di un loro sistema facente uso di Markers solamente per l'inizializzazione e in grado di mantenere il Tracking anche in condizioni in cui un normale sistema Marker based lo avrebbe perso. Questa stabilizzazione è fatta mediante il riconoscimento di caratteristiche tipiche e intrinseche dell'ambiente ed è quindi un primo esempio di approccio Markerless al problema di stimare la pose di una fotocamera nel tempo. Il nome markerless tracking fa riferimento alla semplice assenza della necessità della presenza di markers, questo può allora, utilizzando la differenziazione illustrata in precedenza, essere model-based o feature-based. Un esempio di metodologia utilizzata per effettuare markerless tracking è lo SLAM e in particolar modo algoritmi come il Kalman filter. Nella tesi verranno successivamente trattati in modo più approfondito il problema dello SLAM, l'algoritmo di Kalman e altri algoritmi simili.

### 1.5.3 Hybrid Tracking

Un approccio ibrido prevede l'utilizzo congiunto di sensori e fotocamera per eseguire il tracking. Nel caso di dispositivi mobili è stato inizialmente pensato di utilizzare l'informazione fornita dal GPS sulla posizione del dispositivo per iniziare il tracking per poi affidarsi a tecniche visual-based [6]. Le prime applicazioni di approcci ibridi non fecero uso di una tecnica markerless tuttavia ma marker-based [5]. Solo successivamente iniziarono a emergere studi su come utilizzare insieme alla varia sensoristica dei dispositivi anche un approccio markerless utilizzando SLAM e gli algoritmi che permettono di implementarlo. Questo è l'approccio utilizzato oggi nell'ambito dei dispositivi mobili in cui vengono utilizzate le informazioni provenienti dalle IMUs e viene utilizzato SLAM per effettuare il tracking.

## 1.6 Markerless Tracking

La tecnica markerless può essere suddivisa, utilizzando la definizione data in precedenza, in model-based o feature-based. La differenza tra le due sta nella necessità o meno dell'utilizzo di un database come riferimento. Nonostante l'applicazione sviluppata in questo lavoro di tesi non fa utilizzo di un tracking che necessita di un modello a priori dell'ambiente, utilizzando invece la tecnologia dello SLAM per il tracking e mapping simultaneo, molti degli algoritmi utilizzati sono in ogni caso i medesimi. Ai fini di introdurre questi con quanta più completezza per poi approfondire il loro funzionamento successivamente nella tesi verrà illustrato un approccio misto spesso utilizzato in applicazioni mobili per la realtà aumentata. In questo è inizialmente costruito un modello di alcune caratteristiche dell'ambiente.

### 1.6.1 Funzionamento

Questa tecnica di tracking a livello più alto è costituita da due fasi, fase offline e online. Nella fase offline vengono individuati i landmark principali che compongono l'ambiente [12]. Su questi vengono individuati dei punti di interesse detti local features che vengono poi trasformati in vettori descrittivi contenenti informazioni. Viene creato poi un database contenente questi vettori descrittivi che fungono da definizioni delle relative features e utilizzabili per effettuare il riconoscimento nella fase online. In qesto stadio vengono anche definiti tutti gli oggetti virtuali che possono poi essere renderizzati nella particolare applicazione di realtà aumentata utilizzata. La fase online ha invece il compito di calcolare la pose della fotocamera mediante il riconoscimento delle features presenti nei frame dello stream video acquisito, questo in tempo reale e mediante un confronto con le features rappresentate nel database. Questa fase ha un funzionamento che si estende in tre step consecutivi: interest point detection, feature description e feature-based tracking [6]. I primi due costituiscono quello che viene chiamato local feature extraction con la parola local a indicare che una feature non rappresenta l'intera immagine in cui si trova ma è generalmente una delle tante presenti all'interno dell'immagine. L'idea è quella di individuare inizialmente dei punti di interesse all'interno di un'immagine. Vengono definiti successivamente dei vettori descrittivi contenenti opportune informazioni su tali punti di interesse. Questi vettori vengono poi predisposti ad essere utilizzabili per un confronto con informazioni sulle features ottenute da immagini di interesse. L'immagine presente nel database che ha il più elevato numero di feature corrispondenti all'immagine acquisita è considerata come un target acquisito e grazie a questo viene computata mediante opportuni algoritmi una pose iniziale della fotocamera [6].

Vengono successivamente confrontate features tra frame consecutivi e le corrispondenze vengono utilizzate per effettuare il vero e proprio tracking in cui viene mantenuta l'informazione nel tempo della pose.

### 1.6.2 Interest point detection

Per effettuare la fase di detection degli interest points viene utilizzato un interest point detector. Questo è un operatore che ad ogni pixel dell'immagine associa un punteggio rappresentante quanto ritiene essere significativo e che seleziona poi i pixel aventi un punteggio localmente massimo. Esistono standard per la valutazione, caratteristiche ovvero che vengono ricercate in un punto e che contribuiscono all'assegnazione di un punteggio ad esso da parte del detector. Queste sono [13]:

**Ripetibilità**: Date due immagini contenenti la stessa scena ma prese in condizioni diverse un'alta percentuale dei punti individuati in una deve essere presente anche nell'altra.

**Particolarità**: I punti vicini al punto in questione costituiscono una regione piuttosto diversa dal resto dell'immagine circostante così da far sì che la feature non sia confusa con un'altra.

**Quantità**: Le features ottenute dai punti devono essere in numero sufficientemente elevato da permettere la rappresentazione anche di piccoli oggetti mediante il riconoscimento e matching delle features tra frame successivi.

**Efficienza**: I punti individuati devono essere tali da garantire un rapido riconoscimento tra immagini nel caso di applicazioni con limiti temporali stringenti.

I detectors sono generalmente caratterizzabili in base alla loro performance e complessità computazionale. Esistono detectors light-weight , caratterizzati come suggerisce il nome da una bassa complessità computazionale [6]. Questi sono tali da massimizzare l'efficienza e quindi garantire prestazioni in tempo reale su dispositivi con limitate capacità computazionali, la qualità della performance ottenuta è tuttavia pessima. Vi sono poi detectors di alta qualità che garantiscono contemporaneamente un'elevata ripetibilità e particolarità ma sono computazionalmente onerosi. Questi sono stati inizialmente adattati per l'utilizzo in dispositivi mobili i quali non avevano capacità computazionale sufficiente per usufruire di tali detectors. Un notabile esempio di detector light-weight è il FAST (Features from Accelerated Segment Test). Questo deriva a sua volta da un algoritmo computazionalmente più complesso detto SUSAN (Smallest Univalue Segment Assimilating Nucleus), un semplice metodo di corner e edge detection che permette di individuare interest points [13][14]. Il funzionamento consiste nel considerare ogni pixel come un nucleo di un certo intorno circolare la cui intensità ovvero saturazione in scala di grigi viene inoltre presa come riferimento. Ogni intorno viene poi suddiviso in due aree una data da pixels di intensità simile e una da pixels di intensità diversa da quella del nucleo. L'informazione sulla grandezza di ognuna delle due parti relativamente alla grandezza dell'intero intorno circolare è l'informazione fondamentale e quella utilizzata per scegliere punti di interesse all'interno dell'immagine. In parti omogenee dell'immagine ,dal punto di vista di intensità , viene infatti trovato che i nuclei hanno la partizione relativa ad intensità simile che costituisce gran parte del loro intorno circolare. In prossimità di spigoli o angoli invece la proporzione cala a un approssimativo 50% e 25% rispettivamente. In figura (6) è illustrato un esempio di tale processo. Utilizzando questa informazione allora il metodo ha una maggiore informazione sulla posizione di angoli e spigoli all'interno dell'immagine e può restituire un insieme di interest points in modo opportuno. Il SUSAN è tuttavia ritenuto eccessivamente

complesso a livello computazionale per applicazioni in tempo reale e ciò ha portato all'invenzione del detector FAST [14]. L'idea dietro a questo è considerare attorno ad ogni pixel non un intorno circolare bensì una sola circonferenza di pixels in particolare di 16 pixels. Il processo è quello di verificare se sono presenti n pixel consecutivi di questi 16 ad avere tutti un'intensità maggiore o minore del nucleo e in tal caso viene attribuito a questo un punteggio pari ad n [6]. Questo viene inoltre marcato come potenziale interest point. La selezione e individuazione finale degli interest points è fatta valutando regioni dell'immagine. Se un nucleo, pixel centrale, ha un punteggio che è un massimo locale nella propria regione vicina allora questo è considerato un interest point.

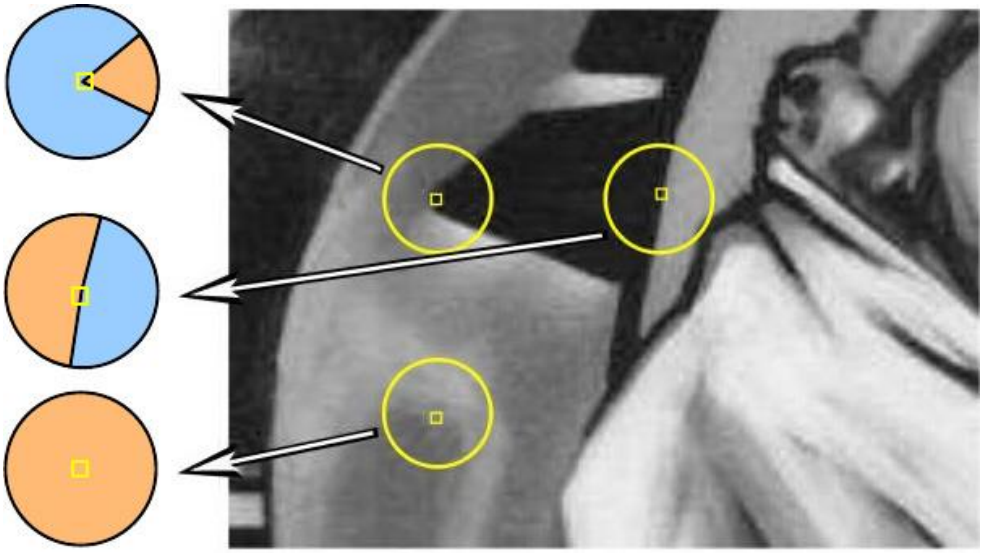

*Figura 6: Esempio del SUSAN detector.*

L'elevata efficienza del FAST è dovuta alla presenza di semplici operazioni aritmetiche per effettuare le operazioni di confronto tra intensità. Un detector di alta qualità e uno dei maggiormente utilizzati è invece il SURF(Speeded Up Robust Features) il cui funzionamento è tuttavia più complesso. Due dei detectors più utilizzati e tra i primi ad essere sviluppati sono l'Harris Detector e il Förstner Operator il funzionamento dei quali sarà descritto in dettaglio nel secondo capitolo della tesi.

### 1.6.3 Feature description

Questa è la fase in cui una volta ottenuta una lista di interest points, di local features, questi vengono descritti mediante vettori utilizzabili poi per effettuare un matching con quelli presenti nel database costruito nella fase offline. Vi è la necessità di descrivere in genere non solo l'interest point ma anche il suo intorno [27]. Se due interest points hanno descrittori della loro area adiacente che coincidono o sono considerati sufficientemente simili allora viene affermato che i due interest points corrispondono allo stesso punto nello spazio. In questo caso gli operatori che effettuano questa operazione sono detti descriptors e i maggiormente utilizzati sono il SURF descriptor e il SIFT (Scale Invariant Feature Transform). Questi due presentano entrambi tuttavia una complessità computazionale troppo elevata per applicazioni in tempo reale e descrittori più semplici come il BRIEF (Binary robust independent elementary features), FREAK (Fast Retina Keypoint) BRISK (Binary Robust Invariant Scalable Keypoints) e LDB (Local difference binary) sono stati sviluppati a tale proposito. Uno dei primi descriptors pensati fu come detto il SIFT, questo è anche quello a partire dal quale i successivi furono sviluppati con l'intento di garantire un'uguale performance ma ad un costo computazionale minore. Quello che fa è compiere una trasformazione di immagini in

coordinate invarianti di scala relative alle varie local features precedentemente individuate in tali immagini[15]. Il descriptor come prima cosa assegna ad ogni interest point un modulo e un orientamento associando ad ogni punto di fatto un vettore. Quello che viene poi fatto è prendere una regione attorno ad ogni interest point divisa ad esempio in una griglia di 16x16 zone nel caso proposto da Lowe. Viene considerato un vettore gradiente dell'intensità in ogni zona e ogni vettore viene pesato con una funzione finestra, finestra di Gauss in particolar modo, rappresentata nell'immagine (6) dalla circonferenza blu, l'immagine è scalata al caso di 8x8 e 2x2 zone rispettivamente per semplicità. Per ogni regione data da 16x16 di queste aree viene considerata una griglia 4x4 data da istogrammi di orientamento che riassumono le informazioni della griglia 16x16. Negli istogrammi ogni freccia ha una lunghezza data dalla somma dei moduli dei gradienti lungo tale direzione. Il descrittore è infine preso come un vettore contenente come valori quelli rappresentati dalla lunghezza delle frecce che costituiscono gli istogrammi di orientamento. Per ogni interest point viene allora preso un descrittore dato da 4x4x8, 128 elementi con 8 il numero di frecce in ogni istogramma. Nella fase offline osservando le stesse features in frame acquisiti sotto angolazioni differenti è stata creata una mappa con questi vettori e con le loro posizioni 3d come parte dell'informazione. Questa mappa è poi il modello utile per gli step successivi del processo di tracking [11]. La grandezza dei vettori ottenuti mediante il SIFT è stata tuttavia trovata eccessiva per effettuare un veloce matching ed è uno dei motivi

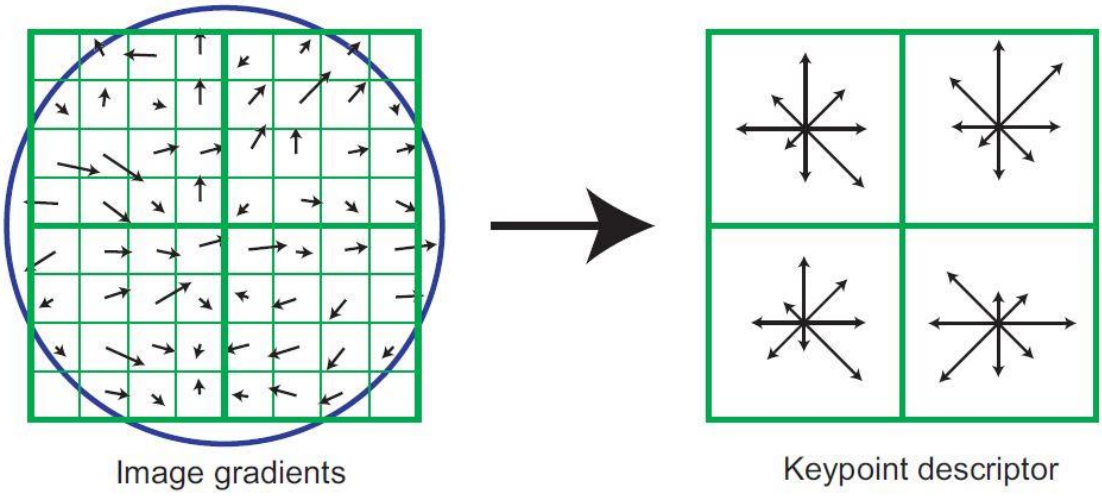


*Figura 7: Esempio del SFIT descriptor.*

della menzionata elevata complessità computazionale del metodo[6].

### 1.6.4 Feature-based tracking

Una volta ottenuti i descrittori delle varie local features individuate in un frame l'ultimo step è quello di eseguire il tracking ovvero ottenere una stima iniziale della pose della fotocamera e mantenerla successivamente nel tempo durante un eventuale movimento. Viene effettuata una ricerca di corrispondenze tra le local features acquisite e descritte del frame corrente con quelle descritte nel database. Se un numero sufficientemente elevato di corrispondenze 2D-3D ovvero tra immagine live e modello nel database vengono trovate è possibile stimare la pose iniziale della fotocamera mediante l'ausilio di algoritmi come RANSAC(Random Sample Consensus) e PROSAC (Progressive

Sample Consensus) [12]. Mediante il matching tra frame successivi delle features e mediante sempre questi due algoritmi è possibile poi mantenere il tracking[11].

## 1.7 ARCore

ARCore è un software development kit ovvero una libreria offerta da Google che permette lo sviluppo di applicazioni facenti uso di realtà aumentata [21]. Questa è anche la libreria alla base dello sviluppo dell'applicazione descritta in questa tesi. Nonostante gli esatti algoritmi di computer vision dietro al suo funzionamento non siano stati resi noti al pubblico in questo ultimo paragrafo di questo capitolo verranno illustrati brevemente il suo funzionamento e le sue caratteristiche. Una discussione approfondita degli algoritmi che si ritengono essere alla base del funzionamento di ARCore e utili per la comprensione dei processi fondamentali della realtà aumentata verrà fatta nel secondo e prossimo capitolo. ARCore utilizza tre processi per realizzare il concetto di realtà aumentata[22]:

1- **Motion tracking :** processo che permette allo smartphone di eseguire il proprio tracking nel tempo.
2- **Environmental understanding:** permette allo smartphone di identificare posizione e dimensione di ogni tipo di superficie.
3- **Light estimation:** questa permette allo smartphone di stimare le caratteristiche della luce all'interno dell'ambiente.

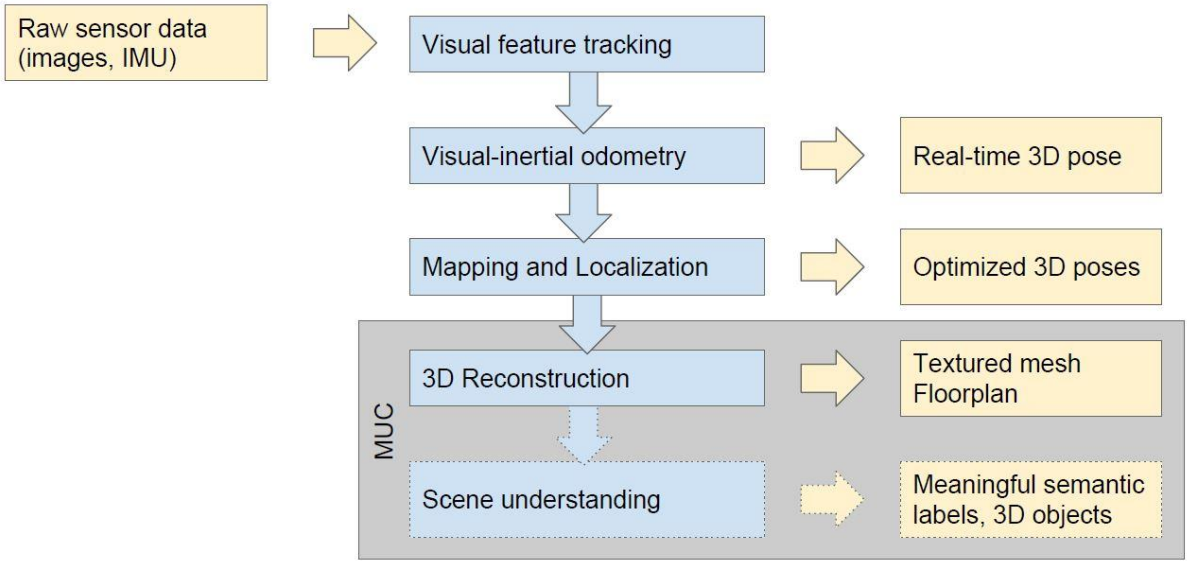


*Figura 8:Schema del funzionamento di ARCore [24].*

Nell'immagine (8) è possibile vedere il completo schema del funzionamento di ARCore [23]. Il primo step è quello del riconoscimento di features presenti nei vari frame del video acquisito. Successivamente la computazione della pose è effettuata mediante un processo di visual-inertial odometry. I due processi sono complementari, le IMUs forniscono accelerazioni lineari e angolari da cui integrando è possibile ottenere posizione e rotazione del dispositivo. Questo è in termini molto semplici quello che si intende con odometry. Questa presenta tuttavia un errore che aumenta nel tempo e dovuto alla presenza di rumore nei sensori componenti le IMUs. Tecniche come la loop closure allora vengono impiegate per migliorare la stima della pose e in particolare un ruolo chiave in questo processo di ottimizzazione viene fatto dall'utilizzo di SLAM[18]. Per fare questo è

importante l'utilizzo di una fotocamera. Per un esempio dell'essenzialità del concetto di loop closure basti pensare ad alla costruzione di un modello di una stanza. La loop closure permette di riconoscere un muro come unico se inquadrato prima da un lato e successivamente dall'altro. In sua assenza nel modello 3d della stanza sarebbero stati inseriti erroneamente due muri per rappresentarne uno. Questo processo di ottimizzazione è utilizzabile tuttavia solamente per ambienti di medie dimensioni e smette di funzionare appena l'ambiente a cui si espone l'applicazione facente uso di ARCore diventa eccessivamente vasto. Questo avviene poiché la richiesta di capacità computazionale raggiunge velocemente i limiti computazionali del dispositivo su cui viene eseguita l'applicazione. Una volta ottenuta la pose è possibile fare una costruzione di un modello tridimensionale dell'ambiente. ARCore non è tuttavia attualmente in grado di costruire un modello completo dell'ambiente ma utilizza prevalentemente informazioni riguardanti superfici piane, verticali e orizzontali.

# Capitolo 2: Algoritmi per detection description e tracking

In questo capitolo verrà illustrato il funzionamento di alcuni degli algoritmi e alcune delle tecniche fondamentali per effettuare il Markerless tracking in applicazioni di realtà aumentata. Vi è motivo di credere che i seguenti algoritmi sono parte integrante della libreria Google ARCore utilizzata per lo sviluppo dell'applicazione in questo lavoro di tesi.

## 2.1 SLAM

Slam è un processo nato per permettere ad un robot posizionato in un ambiente sconosciuto di creare una mappa virtuale dell'ambiente in cui si trova e di determinare la sua posizione all'interno di tale mappa.[2]

### 2.1.1 Localization

La prima parte del problema Slam è la localization, si conosce ovvero la posizione di alcuni landmarks nell'ambiente ma non la pose del robot. Un calcolo continuo della pose permette di conoscere la corretta traiettoria nel tempo del robot all'interno dell'ambiente. Un problema nella fase di localization potrebbe essere una deviazione dovuta ad esempio all'azione di agenti esterni. Il robot in tal caso può pensare di aver girato a destra ad esempio quando in realtà ha girato a sinistra. Questo crea una falsa stima da parte del robot della sua posizione attuale all'interno dell'ambiente.

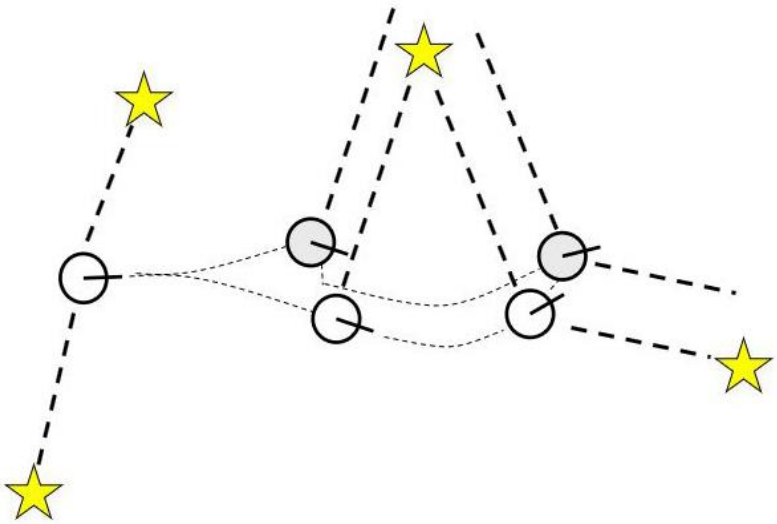

*Figura 9: Esempio di localization.*

Se dalla fotocamera tuttavia il robot nota l'assenza di un landmark dove sarebbe dovuto essere, individua la posizione del landmark reale e utilizza l' errore per correggere la sua pose.

### 2.1.2 Mapping

Nel caso del mapping il robot conosce con certezza a priori la sua pose in ogni istante temporale e individua tramite osservazione la posizione degli elementi caratteristici dell'ambiente. Crea in questo modo una mappa progressiva dell'ambiente. Anche in questo caso è presente un errore ed è tra la posizione nello spazio dei landmarks individuata dal robot e la reale. Queso errore è in questo caso dovuto all'incertezza nella misura dei sensori coinvolti.

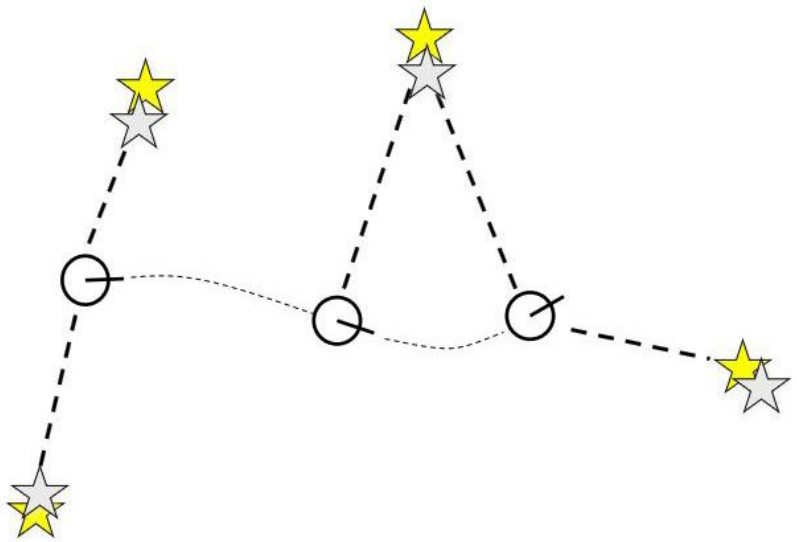

*Figura 10: Esempio di mapping.*

### 2.1.3 Slam (Simultaneous Localization and Mapping)

L'unione del mapping alla localization permette di comprendere cosa succede quando il robot utilizza lo slam. Questo crea una mappa dell'ambiente e la utilizza contemporaneamente quando possibile per correggere l'errore della sua pose nello spazio. È da notare come la mappa restituita dallo SLAM è meno accurata di una prodotta nelle stesse condizioni dal mapping in cui la pose del robot è infatti conosciuta a priori. Allo stesso modo la stima della pose nel caso dello SLAM è peggiore di quella ottenuta dalla localization in cui è conosciuta a priori la mappa dell'ambiente. Se si parla dello SLAM come un problema da risolvere la sua risoluzione si trova allora di fronte a questo "dilemma". Si vogliono stimare accuratamente due quantità ma per stimarne una si ha bisogno dell'altra.

### 2.1.4 Il problema

Il problema parte con la definizione di quelli che sono i dati a disposizione. Si hanno a disposizione un vettore di sfrozi di controllo consecutivi $u_k = (u1,\ldots,uk)$ dati nel tempo e un vettore di risultati $z_k = (z1,\ldots.zk)$ dati dalle continue osservazioni del robot a partire da uno dei suoi sensori come ad esempio una fotocamera. Quello che si vuole sono invece una mappa, indicata con m, dell'ambiente e la traiettoria nel tempo del robot al suo interno, ovvero una sua serie di sue corrette pose in un numero discreto di consecutivi istanti temporali . Questa è data da un vettore $x_k = (x0,x1,\ldots,xk)$ dove ogni elemento è una pose del robot al relativo istante i-esimo. Dal fatto che tra tre pose sono stati

eseguiti due comandi per far muovere il robot e che la prima pose x0 è utilizzata come l'origine del sistema di riferimento impiegato successivamente si ha un elemento in più nel vettore $x_k$ rispetto ai vettori $u_k$ e $z_k$. I dati ottenuti come ad esempio la pose sono in genere soggetti ad errore come detto e per questo motivo per risolvere il problema dello Slam viene preso un approccio probabilistico. Anziché affermare che il robot è ad una certa distanza, limitandosi al caso di una semplice distanza per semplicità ai fini di un esempio, si utilizza una distribuzione gaussiana con questa grandezza sull'asse delle ascisse e sull'asse delle ordinate la probabilità che il robot abbia effettivamente tale distanza dal punto di riferimento considerato.

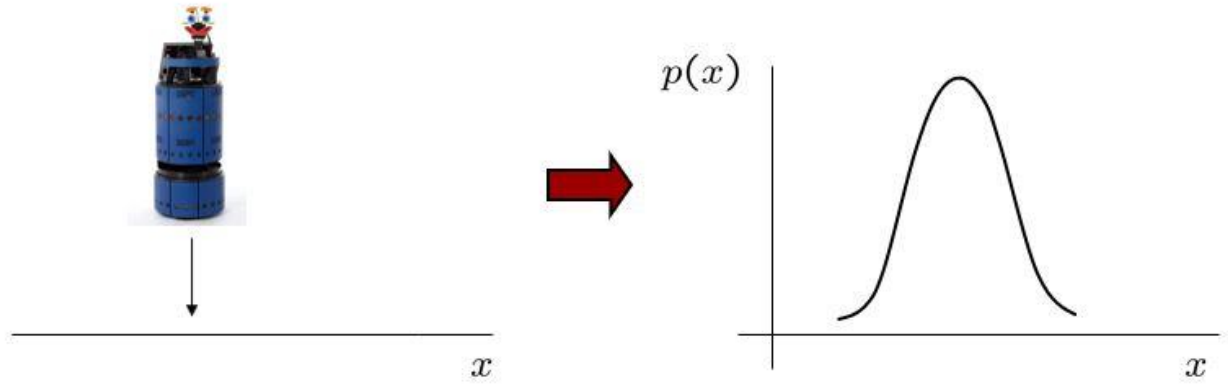


*Figura 11: Stima mediante una gaussiana.*

La distribuzione di probabilità a cui si è interessati in questo caso è indicata con p(xk, m | zk, uk)[2][3]. Il simbolo | sta ad indicare che si vuole stimare il valore delle variabili che precedono il simbolo a partire dalla conoscenza dei valori di quelle che lo seguono. La soluzione del problema SLAM consiste allora nello stimare questa distribuzione di probabilità. Per fare ciò vi sono diversi metodi e in base al tipo di informazioni che si hanno a priori viene in genere preferito un metodo piuttosto che un altro. È da fare infine un'importante distinzione tra Online SLAM e Full SLAM. Con Online SLAM ci si riferisce al problema di voler conoscere solamente l'ultima posizione e non l'intera traiettoria nel tempo del robot. Questa informazione è invece quello che si vuole conoscere nel Full SLAM oltre ovviamente alla mappa che è uno dei parametri di interesse in entrambi i casi.

### 2.1.5 Bayes filter

Il concetto alla base dei metodi utilizzati per risolvere il problema dello SLAM è il filtro di Bayes. Questo è un approccio probabilistico dedicato a stimare una sconosciuta funzione di densità di probabilità. In questo caso come visto una delle grandezze la cui densità è da stimare è lo stato del robot, questo a partire da due informazioni che sono le osservazioni zk e i controlli uk. Lo stato può riferirsi ad esempio alla pose, ma può riferirsi anche a qualsiasi informazione che si desidera conoscere. Inizialmente ogni stato è equiprobabile, si ha allora una distribuzione uniforme. Se il metodo ha successo si ha alla fine un picco visibile nella distribuzione attorno ad uno stato che è idealmente il reale. Non si ha tuttavia come detto la certezza sul fatto che il sistema sia in tale stato ma si utilizza appunto la distribuzione di probabilità ottenuta. La distribuzione del corrente stato a partire dall'informazione delle osservazioni e comandi precedenti viene indicata con $p(x_t \mid z_{1:t}, u_{1:t})$, la si chiama "belief " dello stato $x_t$ e la si indica con $bel(x_t)$. È possibile utilizzare il teorema di Bayes e scrivere l'uguaglianza $bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t})$.

Nel primo termine, $p(z_t \mid x_t, z_{1:t-1}, u_{1:t})$, è possibile fare una presupposizione detta la Markov assumption. Siccome si conosce l'attuale stato del sistema è possibile affermare che non è necessario in realtà conoscere alcun informazione sul passato, si può stimare ovvero l'osservazione $z_t$ a partire dalla sola conoscenza dello stato attuale $x_t$. L'espressione diventa allora

$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t})$. Nel secondo termine si vuole stimare lo stato corrente del sistema a partire da tutti gli sforzi di controllo e dalle osservazioni fino all'istante t-1 esimo. Si ipotizza ovvero la situazione in cui si ha una stima completa fino all'istante t-1 e si dà un ultimo sforzo di controllo. Dal teorema della probabilità totale è possibile, introducendo la nuova variabile $x_{t-1}$, ovvero lo stato all'istante precedente all'attuale, scrivere

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t})\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\, dx_{t-1}.$$

L'integrazione è fatta su ogni possibile valore che può assumere la variabile $x_{t-1}$.Viene ora riapplicata la Markov assumption relativamente al primo membro della funzione integranda. La differenza è che la conoscenza è ora dello stato all'istante t-1, non è possibile allora fare a meno dell'informazione fornita da $u_t$ .L'espressione diventa quindi

$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t})\, dx_{t-1}$ . Il primo termine integrando è chiamato Motion model e sta ad indicare che conoscendo lo stato del robot all'istante t-1 e conoscendo lo sforzo di controllo dato tra l'istante t-1 e l'istante t si può stimare la probabilità dello stato a tempo t. Allo stesso tempo per il secondo membro si può affermare che siccome si è interessati alla stima dello stato all'istante t-1 non è di interesse l'informazione $u_t$ . L'espressione viene perciò ulteriormente ridotta a

$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$ .Il termine $\eta\, p(z_t \mid x_t)$ è detto Observation model. È possibile notare ora come il secondo termine integrando $p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})$ non è altro che la belief dello stato all'istante t-1, all'istante precedente all'attuale. L'espressione finale è ovvero

$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t)\, bel(x_{t-1})\, dx_{t-1}$ . Questa ottenuta in questo modo è una formula ricorsiva che permette di stimare lo stato del sistema utilizzando: lo stato all'istante precedente $x_{t-1}$, l'attuale sforzo di controllo $u_t$ e l'attuale osservazione $z_t$. Il filtro di Bayes viene a volte rappresentato a partire da questa formula con due step. Il primo step è detto Prediction Step ed è la relazione $\underline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t)\, bel(x_{t-1})\, dx_{t-1}$ . Il secondo step è il Correction step e non è altro che la relazione vista scritta ora come $bel(x_t) = \eta\, p(z_t \mid x_t)\, \underline{bel}(x_t)$. Il prediction step prende in considerazione il comando eseguito e il correction step l'osservazione dei sensori. Se si ha allora una distribuzione di probabilità dello stato attuale, dando uno sforzo di controllo e ottenendo un'osservazione dai sensori è possibile calcolare lo stato all'istante successivo. Questo è quello che fa il filtro di Bayes. Nel contesto dello SLAM la variabile di stato $x_t$ fa riferimento alla pose del robot e alla mappa dell'ambiente. Esistono diverse implementazioni del filtro di Bayes e si dividono in due famiglie principali, la famiglia dei Kalman filters e quella dei Particle Filters. Questi si differenziano in base ai modelli che permettono di utilizzare e in base alle presupposizioni che si fanno sulla natura della distribuzione di probabilità.

## 2.1.6 Kalman filter

Questo è una particolare implementazione del filtro di Bayes ed è considerata la più utilizzata. Se utilizzabile, se le ipotesi necessarie sono ovvero assumibili, è inoltre affermabile che il filtro di Kalman è il migliore stimatore tra le varianti del filtro di Bayes. Le assunzioni che richiede sono che il modello usato sia lineare e che la distribuzione di probabilità sia gaussiana. Con modello lineare si intende che il motion model e l'observation model sono funzioni lineari ovvero che valgono le relazioni

(1) $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$
(2) $z_t = C_t x_t + \delta_t$

con A B e C delle matrici variabili ad ogni istante temporale e $\epsilon_t$, $\delta_t$ termini che indicano eventuale rumore. La matrice $C_t$ svolge il ruolo particolare di indicare cosa ci si aspetta di osservare all'istante corrente dato lo stato attuale. Quello che si vuole fare nel filtro di Kalman è utilizzare nella formula ricorsiva vista

(3) $bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t)\, bel(x_{t-1})\, dx_{t-1}$

un motion model e un observation model di tipo gaussiano. Si prendono questi due ovvero della forma:

(4) $p(x_t \mid x_{t-1}, u_t) = \det(2\pi R_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)\right)$

(5) $p(z_t \mid x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1} (z_t - C_t x_t)\right)$. Viene fatta inoltre l'assunzione che la precedente belief $bel(x_{t-1})$ sia di tipo gaussiano. Siccome tutte le distribuzioni sono gaussiane la belief $bel(x_t)$ sarà gaussiana a sua volta. I termini $R_t$ e $Q_t$ rappresentano l'incertezza per stato e osservazione rispettivamente. Quello che fa il filtro di Kalman è calcolare una media pesata con l'incertezza tra la predizione e l'osservazione.

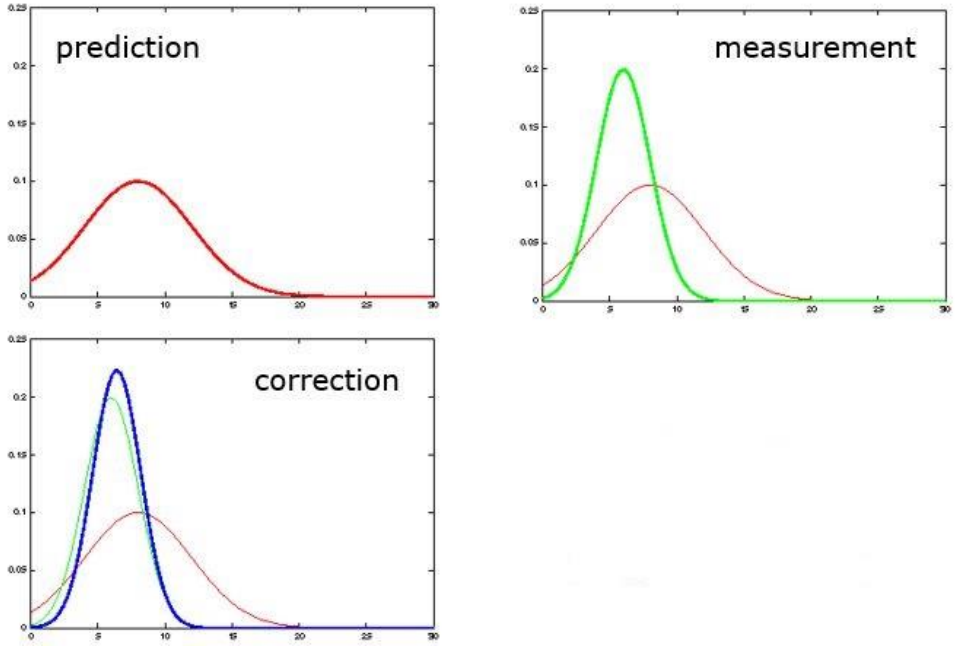


*Figura 12: Processi di prediction e correction.*

Dalle curve è possibile notare come nell'ultimo step della correction la media è più vicina a quella della curva con minore covarianza associata. Il processo è iterativo, data la situazione corrente, la stima successiva e l'osservazione successiva si ripete allo stesso modo il ragionamento.L'idea è che migliore è il sensore utilizzato dal robot, minore è l'incertezza della misura e più il robot si fiderà di questa. Peggiore è il sensore e più il robot si fiderà della sua predizione del believ. Il vero e proprio algoritmo di Kalman che esegue il calcolo di queste gaussiane è il seguente:

1: Kalman_filter($\overline{\mu}_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2: $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$

3: $\underline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

4: $K_t = \underline{\Sigma}_t C_t^T (C_t \underline{\Sigma}_t C_t^T + Q_t)^{-1}$

5: $\mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t)$

6: $\Sigma_t = (I - K_t C_t) \underline{\Sigma}_t$

7: return $\mu_t, \Sigma_t$

I passi 2 e 3 sono il prediction step, i passi dal 4 al 6 il correction step. Il collegamento con quanto già visto è dato dal fatto che se si utilizzano i modelli (4) e (5) per calcolare una nuova gaussiana con (3) si ottengono esattamente le stime di media e covarianza dello step 7. L'algoritmo è allora semplicemente un modo più compatto di scrivere ciò. I primi due step restituiscono la media e la varianza stimate dopo che si è dato lo sforzo di controllo e fatto muovere il robot. Il passo 1 è il modello lineare visto precedentemente. Nel passo 2 si ha la covarianza che è pensabile come

l'incertezza della stima. Questa è calcolata come l'incertezza all'istante precedente più l'incertezza che viene aggiunta dal movimento. L'equazione chiarisce come il movimento aggiunge incertezza al sistema che viene indicata con il termine di rumore $R_t$. Il correction step effettua la media pesata detta e questo è descritto dal termine $K_t$ calcolato nello step 4 e chiamato Kalman gain. Il termine $Q_t$ indica l'incertezza dell'osservazione dei sensori. Nel caso di alta incertezza su quello che si osserva allora il termine sarà molto grande, la somma nello step 4 sarà grande, il suo inverso molto piccolo e di conseguenza si ha un piccolo Kalman gain. Le conseguenze di questo possono essere chiaramente viste poi allo step successivo dell'algoritmo. Nello step 5 viene calcolata la nuova media come la media predetta più il Kalman gain per la differenza tra quello che si osserva e quello che ci aspetta di oservare. Questo termine di guadagno sta allora ad indicare quanto viene presa in considerazione quest'utlima differenza nella computazione della nuova media con la conseguenza già descritta e mostrata nei grafici in figura (12). Se i sensori non avessero dato alcuna informazione si avrebbe avuto un Qt tendente all'infinito, un $K_t = 0$ e al passo 5 una media uguale a quella già predetta, $\mu_t = \overline{\mu}_t$. I sensori non hanno ovvero fornito nessuna nuova informazione utile ai fini della correzione. È possibile notare inoltre cosa succede sempre a partire dallo step 4 se l'incertezza sull'osservazione è invece bassa e in particolare 0. In tal caso si ha $Q_t = 0$ da cui, dallo step 4,

$K_t = \underline{\Sigma}_t C_t^T (C_t \underline{\Sigma}_t C_t^T)^{-1} = \underline{\Sigma}_t C_t^T ((C_t^T)^{-1} \underline{\Sigma}_t^{-1} C_t^{-1}) = C_t^{-1}$ ,sostituendo questo allo step 5 si ha infine

$\mu_t = \overline{\mu}_t + C_t^{-1} (z_t - C_t \overline{\mu}_t) = C_t^{-1} z_t$. Si è detto come la matrice indicasse cosa ci si aspetta di osservare all'istante corrente dato lo stato corrente, la sua inversa darà allora l'informazione opposta. Essa indica cosa ci si aspetta come stato data l'osservazione è ovvero grazie alla sola osservazione che è in grado di restituire la media finale della belief dello stato. Lo step 6 fa infine un update della covarianza utilizzando anch'essa il Kalman gain e la matrice covarianza della believ stimata nel prediction step.

### 2.1.7 Extended Kalman filter

L' Extended Kalman Filter fa la medesima cosa ma viene utilizzato nella situazione in cui le equazioni che si hanno non sono lineari. Il caso è ovvero quello in cui anziché avere le equazioni (1) e (2) queste sono della forma

(1) $x_t = g(x_{t-1}, u_t) + \epsilon$
(2) $z_t = h(x_t) + \delta_t$

dove g e h sono delle funzioni non lineari. L'extended kalman filter prevede allora prima di linearizzare le due funzioni e poi di applicare l'algoritmo di Kalman visto. Si utilizza un'approssimazione di Taylor troncata al prim'ordine e valutata nella media della stima precedente indicata con $\mu_t$, si ha

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\delta g(\mu_{t-1}, u_t)}{\delta x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$h(x_t) \approx h(\bar{\mu}_t) + \frac{\delta h(\bar{\mu}_t)}{\delta x_t}(x_t - \bar{\mu}_t)$. I termini $\frac{\delta g(\mu_{t-1}, u_t)}{\delta x_{t-1}}$ e $\frac{\delta h(\bar{\mu}_t)}{\delta x_t}$ sono delle matrici Jacobiane nel caso generale in cui g(x) è un vettore di funzioni. Questo è valido ovviamente solamente attorno ad un punto di linearizzazione nel punto ovvero attorno al quale si è approssimata la funzione con il suo sviluppo in serie di Taylor. Il calcolo va rifatto quindi se si è interessati alla linearizzazione attorno ad un altro punto. Indicando le due matrici Jacobiane con G e H rispettivamente si ha sempre un motion model e un observation model di tipo gaussiano, linearizzati e della forma

$p(x_t \mid x_{t-1}, u_t) \approx$

$$\det(2\pi R_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_t - g(\mu_{t-1}, u_t) - G_t(x_{t-1} - \mu_{t-1}))^T R_t^{-1}(x_t - g(\mu_{t-1}, u_t) - G_t(x_{t-1} - \mu_{t-1}))\right)$$

$p(z_t \mid x_t) \approx \det(2\pi Q_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)\right)^T Q_t^{-1}(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t))\right)$.

L'algoritmo in questo caso è

1: Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2: $\bar{\mu}_t = g(\mu_{t-1}, u_t)$

3: $\underline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

4: $K_t = \underline{\Sigma}_t H_t^T (H_t \underline{\Sigma}_t H_t^T + Q_t)^{-1}$

5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

6: $\Sigma_t = (I - K_t H_t)\underline{\Sigma}_t$

7: return $\mu_t, \Sigma_t$.

L'unica cosa che è cambiata è che le funzioni lineari sono state sostituite dalle non lineari e le matrici A e C dalle matrici G e H. La complessità dell'algoritmo è $O(k^{2.4} + n^2)$ con k la dimensione del vettore z e n quella del vettore x od ogni istante temporale. Questo algoritmo rimane allora efficiente fintanto che la dimensione dei vettori di stato e osservazione è piccola.

## 2.2 Triangolazione e localizzazione

Sistemi basati sulla visione per effettuare l'operazione di tracking si basano sulla relazione tra punti nell'ambiente osservato e la fotocamera che li osserva[11]. Con un processo chiamato triangolazione è possibile, date immagini di stessi punti nello spazio da più di una nota posizione della fotocamera, calcolare la posizione in 3d di questi punti. Questo processo è di particolare utilità per costruire un modello 3d dell'ambiente. Il processo chiamato localizzazione prevede invece il processo opposto. Osservando ovvero più punti di cui si conosce la posizione nello spazio da un'unica posizione della fotocamera è possibile calcolare la sua posizione e rotazione relativamente

a questi. Quest'ultimo processo è allora chiaramente utile per effettuare il tracking della fotocamera ma anche per fare una stima iniziale della sua pose.

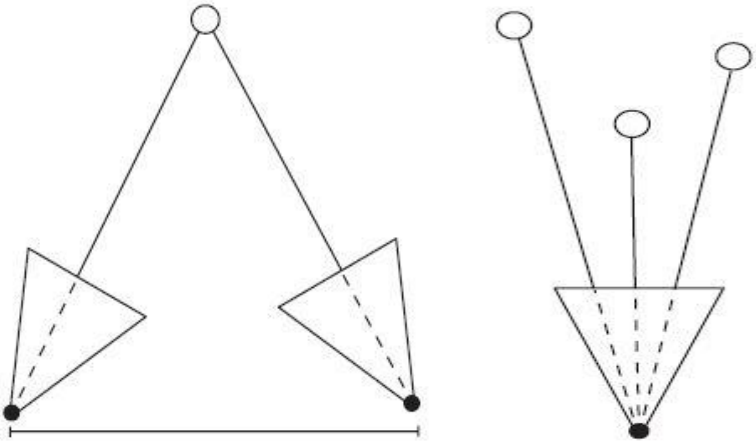

*Figura 13: Triangolazione e localizzazione*

In un'applicazione di realtà aumentata è possibile considerare nota la pose iniziale della fotocamera considerandola come l'origine degli assi nel mondo. Applicando un processo di triangolazione è poi possibile misurare la posizione di alcune features caratteristiche individuate nello spazio. Non perdendo traccia di queste features tra frame successivi del video è possibile poi utilizzare il processo di localizzazione per mantenere l'informazione sulla pose della fotocamera relativamente a tali punti.

### 2.2.1 Stereo normal case

Un modello spesso utilizzato per spiegare il concetto di triangolazione è lo stereo normal case[25]. Questa è caratterizzato dalla presenza di due immagini nello stesso piano e con un offset lungo un solo asse.

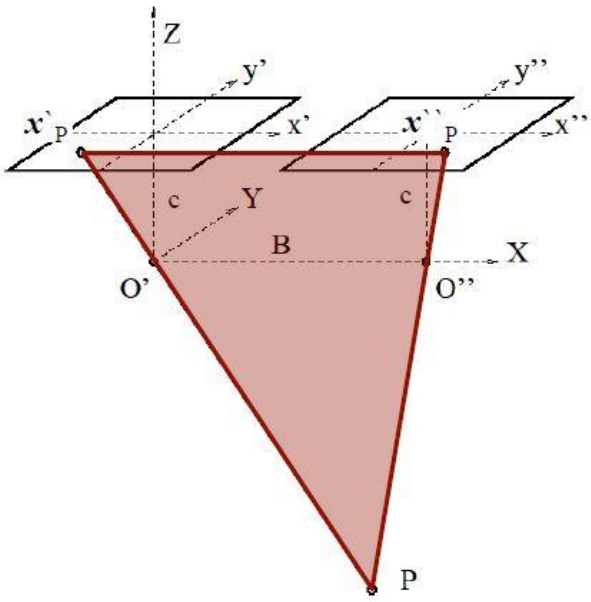


*Figura 14:Stereo normal case.*

Un'altra caratteristica è che le due immagini sono orientate nello stesso modo non vi è quindi alcuna differenza rotazionale tra le due. È possibile anche immaginare la presenza di due fotocamere posizionate l'una a fianco all'altra che puntano lungo la stessa direzione ma con una di queste traslata lateralmente rispetto all'altra. Le due immagini sono della stessa scena quindi ma prese in posizioni differenti. Lo scopo finale della triangolazione è quello di prendere due punti corrispondenti tra le due immagini e ricavare la posizione nello spazio dell'entità corrispondente a tali punti. Si ha che $x'_p$ e $x''_p$ sono le due proiezioni sui due piani immagine rispettivamente del punto p nello spazio. Conoscendo la loro posizione si vogliono trovare le coordinate 3d del punto P. È possibile in queste condizioni geometriche semplificate ridurre il problema al caso più semplice bidimensionale considerando il triangolo $x'_p$ $x''_p$P. Con un leggero cambio nei simboli utilizzati ci si riduce alla situazione in figura (15).

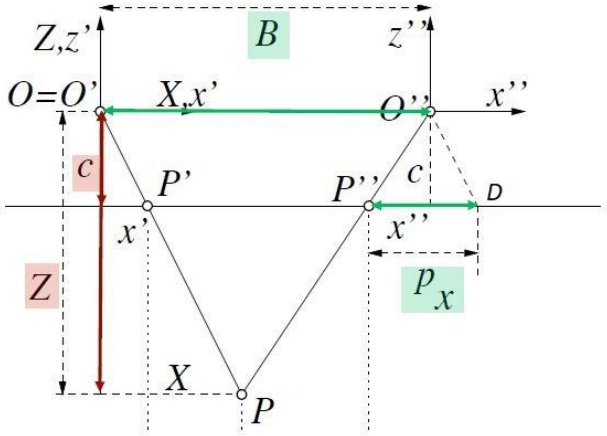


*Figura 15:Caso bidimensionale.*

Traslando il segmento O'P verso destra si ottiene il segmento tratteggiato P''D la cui lunghezza è indicata con $P_X$ e che non è altro che la differenza tra le coordinate x dei due punti nelle rispettive immagini. A questo punto è possibile legare le distanze in gioco affermando che

$\frac{Z}{c} = \frac{B}{-(x''-x')}$, relazione giustificabile mediante il teorema di Talete. La quantità al denominatore è detta "x-parallax" e non è altro che la differenza della coordinata x dei due punti corrispondenti nelle due immagini. Da questa equazione è possibile ottenere la distanza Z semplicemente moltiplicando ambo i membri per la costante c. Siccome sia B che c sono costanti la distanza Z dipende esclusivamente dall'x-parallax. La stessa cosa è fattibile per la distanza X per cui vale la relazione $\frac{X}{x'} = \frac{Z}{c} = \frac{B}{-(x''-x')}$. Per ottenere la coordinata y si considera il piano xy mostrato in figura (16).

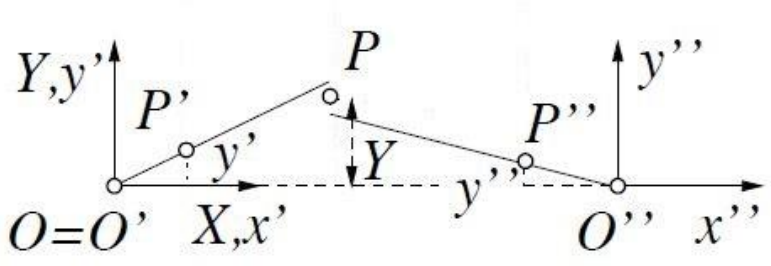


*Figura 16:Piano xy.*

Nel caso ideale dello stereo normal case ci si aspetta che la coordinata y sia la stessa per entrambi i punti corrispondenti. Nel caso di differenza tra i valori delle due coordinate viene semplicemente presa la media come indicato in figura(16). La relazione valida in questo caso presenta allora una media ed è $\frac{Y}{\mathrm{X}} = \frac{\frac{y'+y''}{2}}{x'}$ da cui avendo X dall'espressione precedente è possibile ricavare la Y come $Y = \frac{y'+y''}{2} \frac{B}{-(\mathrm{x''}-\mathrm{x'})}$ . Mediante le tre equazioni viste è allora possibile, conoscendo le coordinate x e y dei due punti sulle due immagini e la costante c, stimare la posizione del punto di interesse nello spazio tridimensionale.

## 2.3 Harris Corner Detector

### 2.3.1 Il concetto di corner detection

L'Harris corner detector è uno dei primi metodi utilizzati per effettuare la descritta interest point detection all'interno di un'immagine e sul quale si basano altri detector computazionalmente più efficienti. Una prima caratteristica di questo come suggerisce il nome è quella di utilizzare di angoli come elemento da riconoscere per definire un interest point. Con corner detection si intende l'identificazione di angoli all'interno dell'immagine, è allora di interesse in tale processo differenziare tra le diverse caratteristiche presenti all'interno di un'immagine. L'idea di come fare ciò è illustrabile da un semplice algoritmo che prevede l'utilizzo di una "finestra"[26]. Questa viene traslata verso l'asse y dato un semplice sistema di riferimento bidimensionale e se la zona dell'immagine a cui è sovrapposta cambia allora è possibile la presenza di un angolo in tale punto. Se la zona sottesa dalla finestra originale e la traslata sono le stesse allora è più probabile la presenza di uno spigolo. Con tale algoritmo un angolo è identificato se la finestra restituisce zone diverse tra loro qualsiasi sia la direzione in cui è avvenuto il suo movimento. Un'illustrazione di ciò è in figura (17).

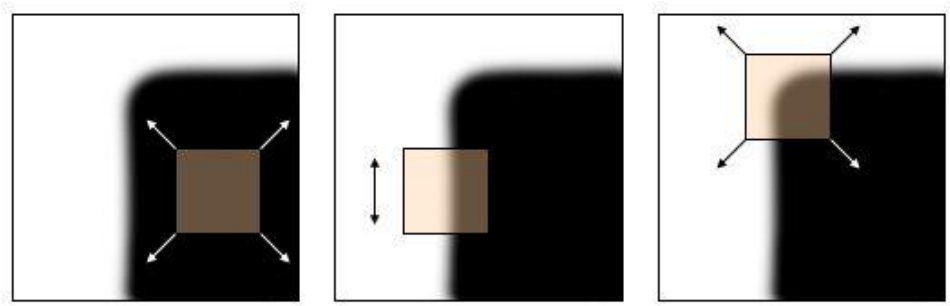

*Figura 17:Semplice esempio di corner detection.*

Nel primo caso qualsiasi sia il movimento il cambiamento è nullo, nel secondo caso è nullo lungo la direzione y e nel terzo è presente lungo ogni direzione. Questo problema consiste nello studio della funzione in due variabili $E(u,v)= \Sigma_{x,y}\, w(x,y)\, [I(x+u, y+v) - I(x,y)]^2$ dove u e v sono l'incremento lungo x e y rispettivamente e $w(x,y)$ la particolare funzione finestra utilizzata. La funzione rappresenta infatti il cambiamento della zona descritta dalla finestra ad una traslazione di di un fattore u e v lungo i due assi. Questo algoritmo non è computazionalmente efficiente ed ha una complessità computazionale di O(larghezza della finestra$^2$ * range della traslazione$^2$ * larghezza dell'immagine$^2$). Questa complessità può portare alla necessità di effettuare un numero di computazioni del valore della funzione $E(u,v)$ dell'ordine del miliardo per ottenere una stima della posizione degli angoli all'interno dell'immagine. Per una maggiore efficienza allora si abbandona l'idea di voler conoscere l'esatto valore della funzione nel proprio dominio ma ci si accontenta di una stima. Si sviluppa in serie di Taylor la funzione E(u,v) e in particolar modo si sviluppa la funzione intensità I come $I(x+u, y+v) = I(x, y) + \frac{\delta I(x,y)}{\delta x} u + \frac{\delta I(x,y)}{\delta y} v = I(x, y) + I_x v + I_y v$. La funzione E(u,v) approssimata diventa allora $E(u,v) \approx \Sigma_{x,y}\, w(x,y)[I_x u + I_y v]^2$ scrivibile in forma matriciale come $E(u,v) \approx \Sigma_{x,y}\, w(x,y)\, [u\ v]\, \begin{bmatrix} I_x \\ I_y \end{bmatrix}\, [I_x\ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$. Dalla dipendenza dalle sole variabili x e y è possibile portare fuori dalla sommatoria i vettori riga e colonna composti da u e v e computando il prodotto interno si ottiene $E(u,v) \approx [u\ v]\{ \Sigma_{x,y}\, w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \} \begin{bmatrix} u \\ v \end{bmatrix} = [u\ v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$. Il termine all'interno delle parentesi graffe viene indicato con M e utilizzando questa approssimazione è utilizzabile per caratterizzare in modo adeguato l'area sottesa dalla finestra. La matrice M è visibilmente simmetrica e dal teorema spettrale è allora anche diagonalizzabile. Viene chiamata R la matrice le cui colonne sono date dagli autovettori relativi a M. Questi sono ortogonali e costituiscono le basi degli autospazi relativi agli autovalori di M. Tramite questa matrice R è possibile scrivere $E(u,v) \approx [u\ v]\, R \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix}\, R^{-1} \begin{bmatrix} u \\ v \end{bmatrix}$. L'approssimazione matriciale $E(u,v) \approx [u\ v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$ se sviluppata assume come grafico un iperboloide ellittico le cui curve di livello sono appunto delle ellissi. L'espressione vista è ulteriormente sviluppabile come $E(u,v) \approx [u\ v]\, R \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix}\, R^{-1} \begin{bmatrix} u \\ v \end{bmatrix} = (R^T \begin{bmatrix} u \\ v \end{bmatrix})^T \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} (R^T \begin{bmatrix} u \\ v \end{bmatrix}) = \lambda_1 u'^2 + \lambda_2 v'^2$ che se uguagliato ad un numero ad esempio 1 non è altro che un ellisse ruotata dalla matrice R come mostrato in figura (18).

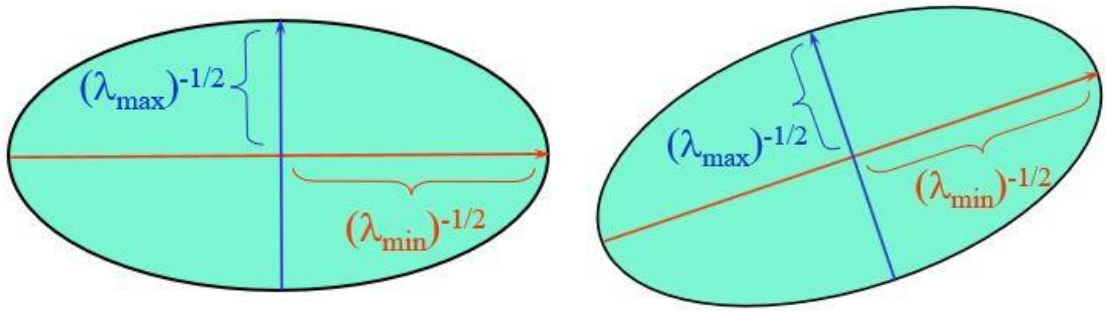


*Figura 18: Ellisse ruotata e ottenuta come curva di livello.*

La lunghezza dei suoi assi è data dagli autovalori della matrice M mentre la sua orientazione dalla matrice R. Quello che viene fatto con questi autovalori è confrontare il loro valore con il valore 0 e utilizzare l'informazione ottenuta per stimare se la zona relativa alla finestra è una zona uniforme, uno spigolo o un angolo.

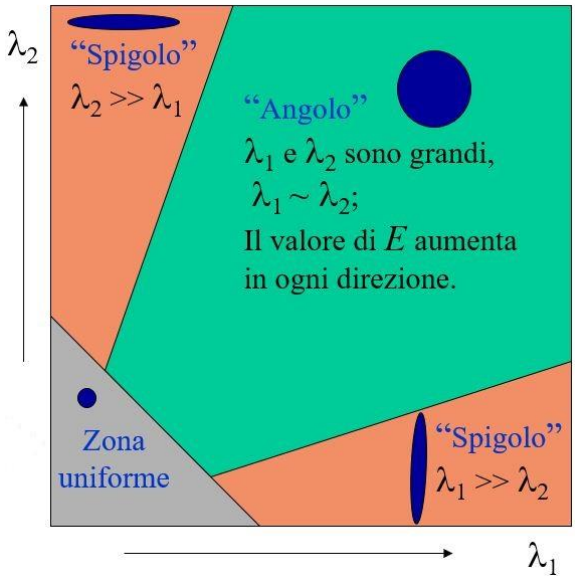


*Figura 19:Schema del processo utilizzato per la caratterizzazione delle zone.*

Il valore assunto degli autovalori sta ad indicare il cambiamento lungo due direzioni ortogonali associate a questi. Un angolo viene identificato laddove vi è un cambiamento significativo dell'immagine lungo queste due direzioni e quindi quando i due autovalori assumono entrambi un valore grande[13]. In maniera alternativa se uno dei due autovalori è più grande dell'altro allora è probabile la presenza di uno spigolo. Nel caso sono entrambi molto piccoli è invece probabile la presenza di una zona uniforme con nessun particolare cambiamento lungo le varie direzioni. Una scelta comunemente fatta è tuttavia quella, ai fini di ottenere una maggiore leggerezza computazionale, di inglobare le informazioni relative a entrambi gli autovalori in un unico indice. Questo è un indice di quanto è probabile che la zona considerata nell'immagine contenga un angolo ed è detto "Cornerness" e verrà indicato con C.

### 2.3.2 Harris corner detector

La scelta fatta da Harris e Stephens è quella di calcolare la Cornerness come $C = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$ [26] ovvero come $C = \det(M) - \lambda\,\mathrm{trac}(M)$. In quanto M è una matrice diagonale il prodotto dei due autovalori costituisce il suo determinante mentre la loro somma la sua traccia. Il vantaggio computazionale di calcolare questo indice in questo modo è chiaramente dovuto al fatto che non vi è alcun bisogno in partenza di diagonalizzare la matrice M né di trovare quindi i suoi autovalori. Si calcola il determinante di M con una delle regole che lo permettono e la traccia semplicemente sommando per sua definizione gli elementi sulla diagonale. È chiara dalla formula utilizzata per calcolare la Cornerness la correlazione tra un elevato valore degli autovalori e un elevato valore di questa. L'Harris corner detector segue i seguenti passi:

1- Si acquisisce un'immagine in input.
2- Si calcolano le derivate parziali $I_x$ $I_y$ della funzione intensità associata all'immagine.
3- Si calcolano le componenti $I_x^2$ $I_y^2$ $I_{xy}$.
4- Viene applicata una funzione filtro di tipo gaussiano per costruire la matrice M insieme alle componenti precedentemente calcolate.
5- Viene calcolata la Cornerness come $C = \det(M) - \lambda\,\mathrm{trac}(M)$.
6- Si individuano aree all'interno dell'immagine la cui Cornerness è superiore ad una certa soglia.
7- Si scelgono infine come interest points i punti in cui la funzione Cornerness ammette un massimo locale[13].

## 2.4 Förstner Operator

Il Förstner Operator è uno dei principali e validi corner detectors insieme all'Harris corner detector. Ai fini di realizzare un metodo per identificare interest points all'interno di un'immagine è utile pensare al motivo per cui viene fatto ciò. Uno dei motivi principali è la possibilità di ricercare con facilità questi interest points in altre immagini e affermare che questi, che sono pensabili anche come pixels, corrispondono tra le immagini, ovvero rappresentano lo stesso punto nella realtà. Conoscendo questa corrispondenza e dove le immagini sono state prese nello spazio è possibile ,mediante il metodo di triangolazione visto al paragrafo 2.3, trovare la posizione nello spazio corrispondente a questi punti. Un metodo per trovare corrispondenze senza l'utilizzo di descriptors è quello di tagliare una piccola porzione di un'immagine e ricercarla in una seconda immagine. Uno dei metodi che implementa questa idea è quello della Cross Correlation[25].

### 2.4.1 Cross correlation

La prima delle forti ipotesi da cui parte questa tecnica è la presenza di sola traslazione tra le due immagini, si suppone ovvero l'assenza di rotazione tra le due. La seconda ipotesi è che i valori di intensità variabili possono essere solamente luminanza e contrasto. L'obiettivo come detto è quello di prendere una porzione di un'immagine, un template, ed eseguire l'operazione di ricerca di questo in una seconda immagine. Il processo è detto template matching. Dato un template centrato in un punto $P_m$ il problema diventa quello di trovare i due offset u e v all'interno dell'immagine che

garantiscono il corretto posizionamento del template all'interno dell'immagine stessa. Siccome per ipotesi è assente una rotazione allora la posizione del template all'interno dell'immagine può essere rappresentata dalla posizione del pixel nel suo angolo in alto a sinistra ad esempio. La funzione intensità della prima immagine viene chiamata $g_1(i,j)$ mentre quella del template $g_2(p,q)$. L'assunzione di traslazione come unica trasformazione ammissibile significa affermare che l'unica trasformazione tra coordinate ammessa è $T_G : \begin{bmatrix} p \\ q \end{bmatrix}_m = \begin{bmatrix} i \\ j \end{bmatrix}_m - \begin{bmatrix} u \\ v \end{bmatrix}$ m = 1,….,M.

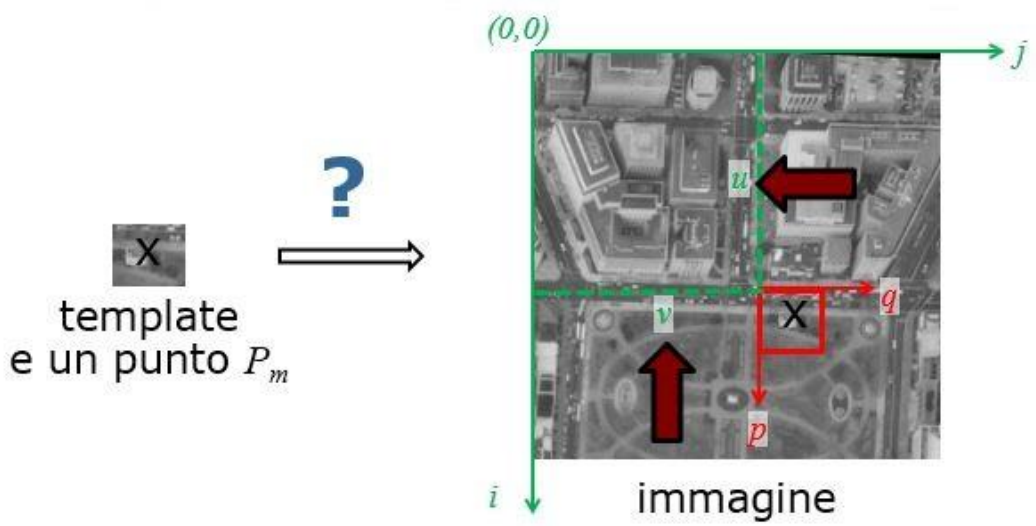


*Figura 20: Ricerca dell'offset che garantisce il template matching.*

Il problema ha allora solo due incognite, u e v costituenti l'offset. Oltre ad una rotazione nel modello utilizzato è previsto anche un cambio di intensità, questo è rappresentabile mediante una trasformazione detta radiometrica e in questo caso viene previsto che sia del tipo $T_I : g_2(p,q) = a + b\ g_1(i,j)$. Da questa trasformazione si aggiungono alle incognite u e v i parametri a e b, con un numero di parametri totali da stimare pari allora a 4. Trovare le prime due variabili implica trovare l'offset che massimizza la somiglianza tra i livelli di intensità di template e immagine. Questa somiglianza è misurabile mediante una funzione detta "cross correlation function", l'offset viene preso come quello che massimizza la funzione. Questa è

$P_{12}(u,v) = \frac{\sigma_{g1g2}(u,v)}{\sigma_{g1}(u,v)\sigma_{g2}}$ . Il termine $\sigma_{g1}$ è la deviazione standard dei valori della funzione $g_2$ del template. Il termine $\sigma_{g1}(u,v)$ è la deviazione standard della funzione $g_1$ dell'immagine ma ristretta ad un'area pari a quella del template e posizionata all'offset corrente (u,v). Il termine $\sigma_{g1g2}(u,v)$ è infine la covarianza tra i valori delle funzioni intensità nell'area in cui template e immagine sono in quel momento sovrapposte. Come tecnica per effettuare la ricerca del template nell'immagine il metodo della cross correlation prevede di prendere il template, farlo scorrere lungo l'immagine e ad ogni istante ovvero per ogni offset (u,v) calcolare il valore della cross correlation function $P_{12}(u,v)$. Viene costruito in questo modo un grafico dai valori della funzione mostrato in figura (21). Nel grafico è in genere presente un picco e il corrispondente offset $(\hat{u}, \hat{v})$ corrisponde alla posizione in cui template e immagine idealmente corrispondono. La funzione di cross correlation assumerebbe idealmente il valore 1 come picco in tale offset tuttavia poiché le due immagini originali, dalla prima delle quali è stato preso il template, sono state prese in posizioni differenti il

valore di intensità tra le due è in genere diverso. Questo è ovvero diverso tra template e immagine anche dove questi due coincidono e questo risulta in un valore di picco minore di 1. È da notare come nel caso template e immagine sono identiche ad esempio figure uniformemente colorate dello stesso colore e senza alcun carattere distintivo la cross correlation function assume il valore di 1 ad ogni offset. Non si osserva in tal caso alcun picco nel rispettivo grafico e non viene trovata alcuna posizione di corrispondenza.

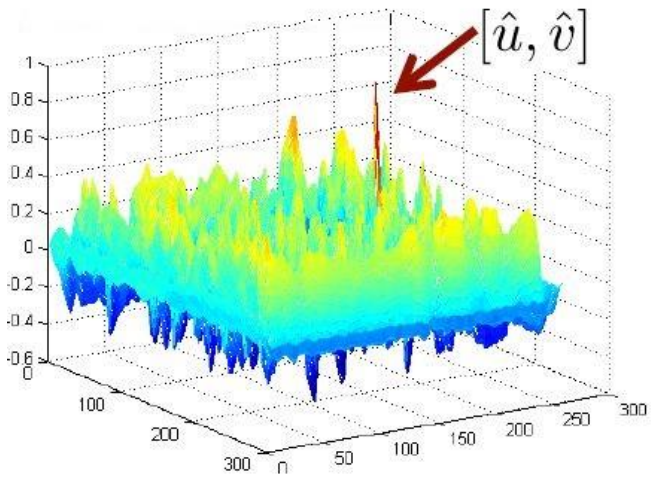


*Figura 21:Grafico della cross correlation function.*

La corrispondenza con questo metodo è inoltre limitata dalla discretizzazione assunta considerando un pixel come unità di musura. È possibile estendere i risultati per ottenere una precisione maggiore e "sub-pixel".

### 2.4.2 Least Squares Matching

Volendo gestire trasformazioni più complesse di quelle assunte possibili come ipotesi per la cross correlation si può generalizzare questa e ottenere il metodo del Least Squares Matching. Alcune trasformazioni che è desiderabile tenere in considerazione possono essere ad esempio la rotazione, lo scaling, o altri tipi di trasformazioni radiometriche[29]. Nonostante, in quanto una generalizzazione, è in grado di risolvere il problema di trovare una corrispondenza anche in presenza di trasformazioni più complesse necessita di un arbitrario valore iniziale. Il problema che deve risolvere il Least Squares Matching è non lineare, si considera sempre un'immagine ed un template le cui funzioni di intensità sono chiamate g e f rispettivamente. Si chiamano le coordinate x e y prese ad ogni m con m l'm-esimo pixel tra gli M totali. L'equazione che definisce il problema è $g(x_m,y_m) = f(x_m-u,y_m-v)+n(x_m,y_m)$ $m=1,....,M$ con n il termine che indica il rumore. È possibile riscrivere questa per ottenere un modello detto il Gauss Markov model ovvero $g(x_m,y_m)- n(x_m,y_m) = f(x_m-u,y_m-v)$ (1). L'obiettivo è quello di stimare l'offset $(\hat{u}, \hat{v})$ che minimizza la somma dei termini di rumore al quadrato. Per risolvere il problema è necessario linearizzare la funzione f che non è generalmente lineare. Il fatto che la funzione f non è in genere lineare è intuibile dal fatto che in una generica immagine l'intensità tra pixel varia seguendo una legge non lineare. Nel caso preso in considerazione viene fatta l'ipotesi che f sia continua e differenziabile , l'unica trasformazione che viene poi considerata è una traslazione. La linearizzazione di f viene fatta attorno al valore iniziale che è come detto necessario e in questo caso indicato con i generici (x,y), si scrive $f(x+\Delta u, y + \Delta v)$

$\approx f(x,y) + \frac{\delta f}{\delta u}|_{(x,y)} \Delta u + \frac{\delta f}{\delta v}|_{(x,y)} \Delta v$ . Da questa si ottiene un Gauss Markov model linearizzato della forma $g(x_m,y_m)- n(x_m,y_m) = f(x_m,y_m) + \frac{\delta f}{\delta u}|_{(x_m,y_m)} \Delta\hat{u} + \frac{\delta f}{\delta v}|_{(x_m,y_m)} \Delta\hat{v}$ . Scritto in questo modo si parla di un pixel, l'm-esimo, estendendo il ragionamento a tutti i pixel si può scrivere

$$\begin{bmatrix} g_1 \\ \vdots \\ g_m \\ \vdots \\ g_M \end{bmatrix} - \begin{bmatrix} f_1 \\ \vdots \\ f_m \\ \vdots \\ f_M \end{bmatrix} - \begin{bmatrix} n_1 \\ \vdots \\ n_m \\ \vdots \\ n_M \end{bmatrix} = \begin{bmatrix} f_{x,1} & f_{y,1} \\ \vdots & \vdots \\ f_{x,m} & f_{y,m} \\ \vdots & \vdots \\ f_{x,M} & f_{y,M} \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix}$$ nominando la prima differenza come $\Delta l$ , il terzo vettore come v, la matrice jacobiana con A e l'ultimo termine come $\Delta x$. Si può definire in questo modo il sistema di equazioni $(A^T\Sigma_{ll}^{-1}A)\ \Delta\hat{x} = A^T\Sigma_{ll}^{-1}\ \Delta l$ con la matrice $\Sigma_{ll}^{-1}$ la matrice inversa della matrice covarianza associata all'incertezza che si ha nella misura del valore dell'intensità. Il rumore che si ha nella misura è in questo caso costante per ogni pixel ed è possibile allora scrivere la matrice covarianza come la matrice identità moltiplicata per un fattore di rumore, $\Sigma_{ll} = \sigma_n^2 * I$ e l'equazione precedente diventa $(A^T\ \sigma_n^2\ I\ A)\ \Delta\hat{x} = A^T\ \sigma_n^2\ I\ \Delta l$ ovvero $(A^TA)\ \Delta\hat{x} = A^T\ \Delta l$. Da questa è possibile calcolare il vettore update $\Delta\hat{x}$ come $\Delta\hat{x} = (A^TA)^{-1}\ A^T\ \Delta l$. Il processo seguito per risolvere il problema ovvero l'equazione (1), il Gauss Markov model non lineare, è iterativo. Si calcola come prima cosa con l'espressione vista il primo cambiamento $\Delta\hat{x}$ dell'offset da quello del valore iniziale scelto. Si itera poi il processo fino alla sua convergenza linearizzando la funzione ad ogni iterazione e si ottiene infine la soluzione come somma tra valore iniziale dell'offset e tutti gli incrementi calcolati. Un risultato notevole di questa tecnica è l'indipendenza del risultato finale dal rumore presente nella misura dell'intensità, è possibile vedere infatti come il termine $\sigma_n^2$ non compare nell'ultima espressione di $\Delta\hat{x}$. La matrice jacobiana A è una matrice di dimensione 2xM, il prodotto $A^TA$ è di conseguenza una matrice 2x2 dove ogni elemento è la somma di M elementi dall'operazione di prodotto righe per colonne. Si ha

$$A^TA = \begin{bmatrix} \Sigma_m f^2_{x,m} & \Sigma_m f_{x,m}f_{y,m} \\ \Sigma_m f_{x,m}f_{y,m} & \Sigma_m f^2_{y,m} \end{bmatrix}$$ con le somme fatte su ogni pixel e con la notazione utilizzata si intende la derivata di f lungo x o y calcolata al pixel m-esimo e per calcolare questi quattro elementi è sufficiente conoscere le derivata prima della funzione f. In questo caso l'incertezza che si ha nella stima dello shift lungo x e quindi della stima finale dell'offset lungo x è data dalla matrice $\Sigma_{\hat{x}\hat{x}} = \sigma_n^2\ (A^TA)^{-1} = \sigma_n^2 \begin{bmatrix} \Sigma_m f^2_{x,m} & \Sigma_m f_{x,m}f_{y,m} \\ \Sigma_m f_{x,m}f_{y,m} & \Sigma_m f^2_{y,m} \end{bmatrix}^{-1}$. È chiaro da questa relazione come maggiore è il gradiente della funzione f minore è la matrice della covarianza e quindi minore è l'incertezza sulla stima. Nel caso di immagine uniformemente colorata, da cui lo sarà anche il template, il gradiente è nullo e l'incertezza è infinita. In quel caso si è detto infatti come non è possibile trovare alcun offset relativo alla corrispondenza cercata. È utile considerare anche la matrice covarianza del gradiente $\Sigma_{\Delta f\Delta f} = \begin{bmatrix} \sigma^2 f_x & \sigma f_x \sigma f_y \\ \sigma f_x \sigma f_y & \sigma^2 f_y \end{bmatrix}$ con $\sigma^2 f_x = \frac{1}{M}\Sigma_m f^2_{x,m}$ , $\sigma^2 f_y = \frac{1}{M}\Sigma_m f^2_{y,m}$ , e $\sigma f_x f_y = \frac{1}{M}\Sigma_m f_{x,m}f_{y,m}$ e legata a quella

dello shift dalla relazione $\Sigma_{\hat{x}\hat{x}} = \frac{\sigma_n^2}{M} \Sigma^{-1}_{\Delta f\, \Delta f}$ con $\sigma_n^2$ la varianza del rumore e supposta costante, il valore $\frac{\sigma_n^2}{M}$ è quindi in particolar modo una costante. Da questa relazione si deduce oltre all'osservazione già fatta sul gradiente che maggiore è il numero di pixels nell'immagine e minore è il rumore relativo alla funzione f e migliore è la stima dell'offset. Un esempio delle informazioni ottenibili con queste equazioni può essere visto utilizzando l'immagine in figura (22) dove si suppone di voler trovare la corrispondenza con un template preso dal centro di questa. Quest'immagine presenta una semplice transizione tra pixels scuri e chiari lungo una dimensione e questo comporta un gradiente con componente lungo x nulla

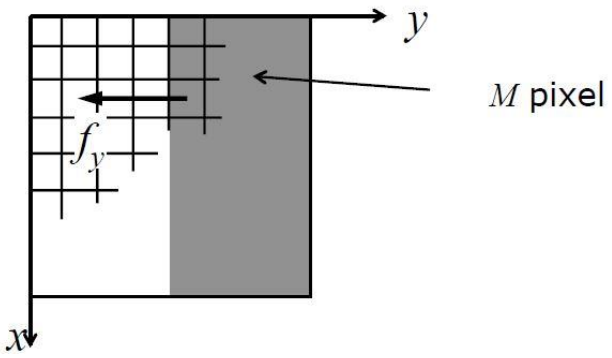


*Figura 22: Caso di netto contrasto.*

ed una lungo y non nulla. In particolar modo allora la somma delle componenti lungo x al quadrato e i prodotti in cui compare $f_x$ restituiscono il valore 0 e la covarianza dello shift diventa $\Sigma_{\hat{x}\hat{x}} = \sigma_n^2 \begin{bmatrix} \Sigma_m f^2_{x,m} & \Sigma_m f_{x,m} f_{y,m} \\ \Sigma_m f_{x,m} f_{y,m} & \Sigma_m f^2_{y,m} \end{bmatrix}^{-1} = \frac{\sigma_n^2}{M} \begin{bmatrix} \sigma^2 f_x & \sigma f_x \sigma f_y \\ \sigma f_x \sigma f_y & \sigma^2 f_y \end{bmatrix}^{-1} = \frac{\sigma_n^2}{M} \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2 f_y \end{bmatrix}^{-1}$. Siccome è necessario invertire quest'ultima matrice per stimare lo shift non è possibile stimare questo lungo la direzione x, in quanto è presente uno 0 come primo elemento della matrice ma è possibile stimarlo lungo y essendo $\sigma^2 f_y$ diverso da 0. L'informazione ottenuta è allora che lungo la direzione y è possibile localizzare un template dell'immagine molto bene, lungo la x no. Intuitivamente infatti prendendo un template dalla zona che presenta entrambe le intensità e facendolo scorrere lungo la direzione x all'interno dell'immagine l'informazione ottenuta è nulla. Non è possibile in quel modo trovare alcun match tra i due, e quindi nessun offset lungo l'asse x, la matrice fa notare che è possibile trovare un offset solo lungo y . Il ragionamento fatto è stato fatto prendendo in considerazione solamente la traslazione come trasformazione e non la rotazione e si è supposta l'assenza di alcuna trasformazione radiometrica. È possibile in questo caso tuttavia generalizzare e aggiungere la presenza di queste trasformazioni. Una rotazione può essere espressa come $T_G$: $\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} a1 & a2 \\ a4 & a5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_m + \begin{bmatrix} a3 \\ a6 \end{bmatrix}$ mentre la trasformazione radiometrica come $T_I$:a7 f+a8 con a7 il cambiamento del contrasto e a8 quello della luminanza. Questo caso è analogo al precedente ma generalizzando si è passati dal dover inizializzare con un valore iniziale e poi stimare non più 2 parametri bensì 8. La linearizzazzione si complica in quanto è ora necessario derivare rispetto non solo più a u e v ma rispetto a tutte le 8 variabili.

## 2.4.3 Förstner Operator

Data la conoscenza di come devono essere le informazioni a disposizione per effettuare un buon matching è ora possibile trovare interest points come quei punti che garantiscono una buona probabilità di trovare una corrispondenza tra immagini differenti[28]. Questo si declina in diverse caratteristiche desiderabili e ricercate in un interest point da individuare. Queste sono:

1- Una discreta differenza tra l'interest point e l'area ad esso circostante.
2- Una robustezza nel riconoscimento di questo in un'altra immagine. Si vuole ovvero essere in grado di ritrovare l'interest point anche nel caso in cui l'angolo con cui è stata acquisita l'immagine è cambiato o è cambiata l'illuminazione dell'ambiente.
3- Si vuole essere in grado di localizzare quanto più accuratamente nello spazio l'interest point.

Nel metodo del least squares matching nel caso di sola traslazione si era arrivati ad un sistema di equazioni lineari della forma $(A^TA)\,\Delta\hat{x} = A^T\,\Delta l$ che esplicitato è $\begin{bmatrix} \Sigma_m f^2_{i,m} & \Sigma_m f_{i,m}f_{j,m} \\ \Sigma_m f_{i,m}f_{j,m} & \Sigma_m f^2_{j,m} \end{bmatrix} \begin{bmatrix} \Delta\hat{i} \\ \Delta\hat{j} \end{bmatrix} = \begin{bmatrix} \Sigma_m f_{im}\Delta g_m \\ \Sigma_m f_{jm}\Delta g_m \end{bmatrix}$. Chiamando la prima matrice N è possibile vedere questa come la media sugli M pixels del prodotto tra il gradiente e il suo trasposto il tutto moltiplicato per M e scrivere $N = M\,\overline{\Delta f \Delta f^T} = M\,\overline{\begin{bmatrix} \Sigma_m f^2_{i,m} & \Sigma_m f_{i,m}f_{j,m} \\ \Sigma_m f_{i,m}f_{j,m} & \Sigma_m f^2_{j,m} \end{bmatrix}} = M \begin{bmatrix} G\,f_i^2 & G\,f_if_j \\ G\,f_if_j & G\,f_j^2 \end{bmatrix}$ con G un operatore rappresentante un box filter, una funzione che così scritta restituisce la media precedente. Sotto queste condizioni e assumendo che le componenti del gradiente hanno media nulla è possibile affermare anche da quanto già visto che la matrice N non è altro che M moltiplicato per la matrice covarianza del gradiente $\Sigma_{\Delta g \Delta g}$. Si è visto come valeva la relazione $\Sigma_{\hat{x}\hat{x}} = \frac{\sigma_n^2}{M}\,\Sigma^{-1}_{\Delta f\,\Delta f}$ la covarianza dello shift può essere allora scritta come $\Sigma_{\hat{x}\hat{x}} = \sigma_n^2\,N^{-1} = \Sigma_{\hat{x}\hat{x}} = \frac{\sigma_n^2}{M}\,\Sigma^{-1}_{\Delta g \Delta g} = \frac{\sigma_n^2}{M}\,(\overline{\Delta f \Delta f^T})^{-1}$. Un'immagine rumorosa dà una maggiore varianza dell'errore e una peggiore stima dello shift. Maggiore è invece il numero di pixels migliore è la stima. Questa dipende inoltre anche dal gradiente, più grandi sono le somme dei suoi valori elevati al quadrato e più accurata questa è. L'obiettivo è quello di trovare interest points corrispondenti ad un'incertezza sulla stima dello shift bassa ovvero tali per cui gli autovalori della matrice $\Sigma_{\hat{x}\hat{x}}$ siano piccoli. Questi devono anche avere come detto una certa particolarità, devono essere migliori dei punti nell'area a loro circostante. Va specificato come piuttosto che un punto viene considerata una zona attorno ad un punto. È possibile utilzzando queste informazioni esplicitare un algoritmo per estrarre questi interest points. Il primo step è computare la matrice covarianza del gradiente $\Sigma_{\Delta g \Delta g} = \begin{bmatrix} \sigma^2 g_x & \sigma g_x \sigma g_y \\ \sigma g_x \sigma g_y & \sigma^2 g_y \end{bmatrix} = \frac{1}{M} \begin{bmatrix} \Sigma_m f^2_{i,m} & \Sigma_m f_{i,m}f_{j,m} \\ \Sigma_m f_{i,m}f_{j,m} & \Sigma_m f^2_{j,m} \end{bmatrix}$ il che significa come prima cosa calcolare gli elementi del gradiente per ogni pixel. Si è detto come si è interessati ad avere autovalori quanto più piccoli e in particolare l'algoritmo cercherà l'autovalore con valore minore. Il secondo step è allora quello di calcolare il minor autovalore della matrice e questo è calcolato come $\lambda_{min}(\Sigma_{\Delta g \Delta g}) = \frac{\sigma^2 g_x + \sigma^2 g_y}{2} - \frac{1}{2}\sqrt{(\sigma^2 g_x - \sigma^2 g_y)^2 + 4\,\sigma g_x g_y}$. Dalle relazioni viste è dimostrabile come il minor autovalore della matrice covarianza del gradiente corrisponde al maggior autovalore della matrice covarianza dello shift. In particolare la relazione che li lega è $\lambda_{max}(\Sigma_{\hat{x}\hat{x}}) = \frac{\sigma_n^2}{M}\,\lambda^{-1}_{min}(\Sigma_{\Delta g \Delta g})$. L'autovalore

massimo $\lambda_{max}(\Sigma_{\hat{x}\hat{x}})$ viene poi confrontato con una certa soglia superiore per effettuare la decisione sulla scelta del punto come interest point. L'autovettore corrispondente all'autovalore $\lambda_{min}(\Sigma_{\Delta g \Delta g})$ ha una direzione che è quella lungo la quale la stima dell'offset è peggiore. In modo simile $\lambda_{max}(\Sigma_{\hat{x}\hat{x}})$ misura l'incertezza che si ha lungo la direzione lungo la quale la stima è peggiore. Porre un limite superiore a questo valore significa limitare allora questa incertezza ad un certo valore scelto. Si impone allora la relazione $\lambda_{max}(\Sigma_{\hat{x}\hat{x}}) \leq T\sigma_{\hat{x}}$ con $T\sigma_{\hat{x}}$ la soglia e pari ad esempio a 0.5 pixels. Per ogni pixel vengono allora calcolate le quantità dette, e viene confrontato l'autovalore con la soglia. Tutti i pixels per cui questa relazione è soddisfatta vengono considerati buoni al fine di venire localizzati in un'altra immagine. Una situazione possibile è quella in cui più pixels in una stessa area soddisfano questa condizione. In tal caso viene selezionato il pixel con il minor $\lambda_{max}(\Sigma_{\hat{x}\hat{x}})$ e questo viene preso come interest point. Mediante l'utilizzo degli autovalori è inoltre possibile e utile eliminare regioni corrispondenti a caratteristiche come spigoli. Questi sono riconoscibili poiché solo uno dei due autovalori della matrice N assume un alto valore. Scegliendo solamente aree in cui entrambi gli autovalori sono elevati è allora possibile individuare caratteristiche con la qualità di essere facilmente localizzabili.

## 2.5 SIFT: Scale invariant feature transform

Il SIFT è un feature point detector e descriptor, il suo compito è ovvero individuare interest points e creare una una loro descrizione così da permettere un confronto di questi tra immagini differenti di una stessa scena 3d. Questa descrizione è in genere data un vettore n-dimensionale con le varie tecniche per effettuare una description caratterizzate da una loro dimensione, nel caso del SIFT di 128 [27]. Un descriptor in genere descrive non solo un interest point ma anche un suo intorno nell'immagine. Una caratteristica importante del SIFT descriptor è l'invarianza alla rotazione, ruotando gli oggetti nell'immagine si ottengono ovvero sempre gli stessi descriptors per i vari interest points. Questo metodo presenta anche una discreta robustezza a cambiamenti di illuminazione nella scena che rappresentano le immagini. È da notare come sebbene il SIFT prevede una sua tecnica per l'individuazione di interest points questo step può essere effettuato mediante altri metodi come ad esempio il Förstner Operator. In tal caso gli interest points possono poi essere sottoposti al processo di description mediante la relativa funzionalità del metodo SIFT. Ai fini di illustrare questo metodo con completezza verranno approfondite entrambe le funzionalità del SIFT.

### 2.5.1 Detection

SIFT utilizza un approccio a tre step per effettuare la detection all'interno di un'immagine e si basa su uno dei metodi di detection alternativi al Förstner Operator chiamato Difference-of-Gaussians. Il primo step è quello di prendere l'immagine ed effettuare ad essa un processo detto Gaussian smoothing che prevede di sfocare l'immagine mediante una funzione gaussiana per eliminare rumore in questa[30]. Supponendo di aver preso più copie della stessa immagine e di aver applicato a queste un processo di smoothing mediante funzioni gaussiane di diversa grandezza il secondo step consiste nel confrontare queste immagini due a due[27]. Per una coppia viene calcolata la differenza tra le due, pixel per pixel e questa, se rappresentata in un immagine a sua volta, tende a mostrare il contorno di oggetti che presentano ad esempio un discreto contrasto con la zona a loro circostante. Vengono ovvero mostrate le aree dell'immagine originale in cui i valori di intensità cambiano. Un esempio di ciò è mostrato in figura (23). Questo step è pensabile come l'applicazione di un filtro

passa banda all'immagine di partenza, vengono mantenute solamente le frequenze comuni tra le varie copie dell'immagine alle quali sono state applicate diverse funzioni gaussiane per il processo di smoothing.


*Figura 23: Immagine differenza in cui è possibile notare il bordo dell'immagine.*

Il terzo e ultimo step è quello di trovare massimi locali tra le varie immagini ottenute come differenze. Un processo da seguire come nel caso del Förstner Operator è quello di sopprimere successivamente i massimi locali non facilmente localizzabili. Al termine di questo procedimento si è in possesso di un certo numero di interest points nell'immagine e si procede con la creazione dei descriptors.

### 2.5.2 Description

Quello che viene fatto nella fase di description dal SIFT per ogni interest point è associare a questo il vettore $\begin{bmatrix} \mathrm{p} \\ s \\ r \\ \mathrm{f} \end{bmatrix}$ detto SIFT feature. Con l'elemento p si indica le coordinate in pixel all'interno dell'immagine in cui è presente l'interest point, l'elemento s descrive il fattore di scala dell'immagine, r l'orientamento. L'elemento f è invece il vero e proprio descriptor e dato da 128 elementi. Nonostante si mantiene l'informazione su un eventuale riscalamento e rotazione è importante da notare come il descriptor è indipendente da questi e rimane sempre lo stesso a parità di interest point. Il SIFT analizza la distribuzione degli elementi del gradiente della funzione intensità nell'area circostante di un interest point. Come discusso nel primo capitolo quello che viene fatto è poi prendere un istogramma riassuntivo dell'entità dell'intensità lungo le varie direzioni e prendere i valori delle sue frecce come componenti del descriptor. Se si ruotasse l'immagine originale anche gli istogrammi sarebbero ruotati, prendendo un descriptor come i valori dell'istogramma allora questo avrebbe dei valori ordinati diversamente. Ai fini del confronto non si avrebbe allora l'invarianza alla rotazione desiderata. Per risolvere il problema e ottenere l'invarianza quello che viene fatto è prendere la dimensione dominante in un istogramma e normalizzare tutte le sue direzioni rispetto a questa. Una volta computato un descriptor per ogni interest point è possibile trovare corrispondenze tra questi in immagini differenti. Quello che viene fatto data un immagine e un interest point è trovare quello in una seconda immagine il cui descriptor ha la minor distanza rispetto al descriptor del primo interest point. Si afferma in tal caso che i due interst points coincidono. Basandosi sulla sola distanza tra i vettori f tuttavia nella pratica è comune la presenza di false corrispondenze, sorge allora il problema di eliminare i falsi positivi e questo è risolvibile mediante l'utilizzo dell'algoritmo RANSAC.

## 2.6 RANSAC: Random sample consensus

Il RANSAC è un algoritmo comunemente utilizzato per risolvere problemi caratterizzati dalla necessità di rimuovere falsi positivi. Questo algoritmo prevede come prima cosa la definizione del problema che si è interessati a risolvere e per il quale la presenza di falsi positivi tra i dati può essere problematica. Il processo inizia risolvendo il problema con alcuni dei dati, utilizzando ad esempio due interst points corrispondenti per effettuare la triangolazione. Viene successivamente visto in che misura i rimanenti points corrispondenti supportano tale risultato e si procede così molteplici volte prendendo di volta in volta una diversa coppia di corrispondenze. L'idea chiave è di individuare la partizione ottimale tra falsi positivi e corretti positivi al fine di usare i positivi per risolvere il problema. L'algoritmo è quindi composto da tre semplici step:

1- Si prende un campione di un certo numero di corrispondenze tra interest points.
2- Si risolve il problema preposto mediante le corrispondenze prese.
3- Si assegna un punteggio ai points non utilizzati per la risoluzione del problema in base a quanti di questi, dato il risultato ottenuto, sarebbero falsi positivi piuttosto che corretti positivi.

Questo procedimento viene poi ripetuto numerevoli volte e viene tenuto il risultato corrispondente al maggior numero di corretti positivi tra gli interest points non utilizzati. Come esempio si supponga di avere un certo numero di punti in un piano e di voler far passare una retta tra due punti. Si inizia prendendo due dei punti tra quelli a disposizione, si fa passare una retta tra i due e questo costituisce la risoluzione del problema postosi.

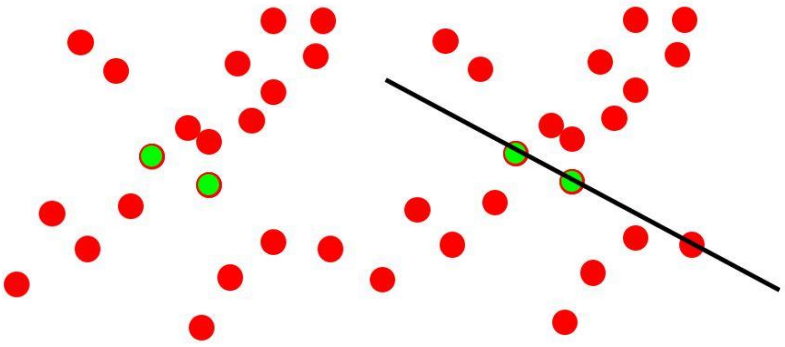

*Figura 24:Scelta dei dati iniziali e risoluzione del problema.*

Si definisce una distanza δ e si afferma che i punti con una distanza dalla retta minore di δ sono i corretti positivi mentre i rimanenti sono considerati falsi positivi. Dato questo esempio, mostrato in figura (25), si ottiene un risultato con associati 4 punti che supportano l'ipotesi iniziale della scelta dei due punti presi.

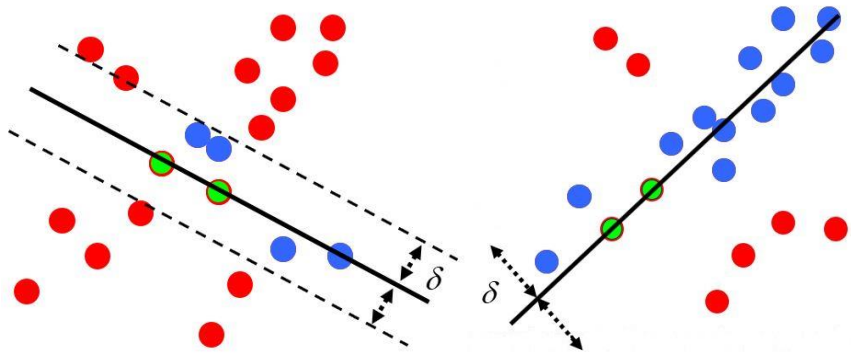


*Figura 25:Valutazione dei dati rimanenti.*

A questo punto si ripete il procedimento cambiando i due punti iniziali e si troveranno a volte un numero maggiore di punti considerabili corretti positivi, ad esempio 12 come sempre mostrato in figura (25) e che caratterizza il risultato come migliore del precedente. Procedendo come soluzione si prenderanno la coppia di punti e la retta passante per questi corrispondenti al maggior numero di corretti positivi. Di fondamentale importanza sono:

**s**: Il numero di punti inizialmente selezionati. Questo è dato dall'algoritmo specifico utilizzato per trovare la soluzione ovvero dal problema che si deve risolvere.

**e**: Il rateo di falsi e corretti positivi.

**T:** Il numero di prove necessarie per arrivare alla miglior soluzione. Questo dipende da **e**. Nel caso di nessun falso positivo è sufficiente un' iterazione dell'algoritmo RANSAC altrimenti il numero delle iterazioni necessarie cresce all'aumentare dei falsi positivi. Questo può essere calcolato come

$T = \frac{\log(1-p)}{\log(1-(1-e)^s)}$ con p la probabilità di aver scelto il set iniziale di punti corrispondente alla soluzione ottima del problema ovvero quella corrispondente al minor numero di falsi positivi tra i punti rimanenti. È possibile notare come il parametro che influisce in maniera più significativa su T è s. Maggiore è allora il numero di punti necessari per risolvere il problema particolare postosi maggiore sarà il numero di iterazioni da far compiere all'algoritmo RANSAC per ottenere il risultato migliore.

**δ:** La distanza che definisce la soglia utilizzata per distinguere i corretti positivi dai falsi.

RANSAC è quindi un algoritmo in grado di irrobustire il processo di matching di interest points tra immagini differenti poiché permette di rimuovere le false corrispondenze generalmente trovate utilizzando metodi come il SIFT.

# Capitolo 3: Sviluppo e funzionamento dell'applicazione

## 3.1 Panoramica

Con l'intento di applicare il concetto della realtà aumentata a uno scopo specifico si è deciso di sviluppare un'applicazione mobile in grado di assistere un consumatore nella scelta dell'acquisto di un prodotto. Si è sviluppata ovvero un'applicazione in grado di riconoscere un prodotto presente in un bacino di prodotti definito a priori in un database e di fornire all'utente un menù interattivo con il quale ottenere informazioni sul prodotto specifico.

### 3.1.1 Obiettivi

Gli obiettivi preposti e realizzati come funzionalità dell'applicazione sono:

1- Permettere all'utente di puntare la fotocamera del proprio smartphone sull'etichetta o su una superficie distintiva e caratteristica di un prodotto e far apparire un menù virtuale a fianco del prodotto.
2- Offrire all'utente tutte le funzionalità classiche di un menù. Permettere ovvero di aprire o chiudere una delle funzionalità previste premendo un apposito bottone virtuale sul touchscreen e di tornare al menù quando lo si desidera o quando l'operazione richiesta è terminata.
3- Le funzionalità che si vogliono offrire mediante il menù virtuale sono principalmente tre. La prima consiste nella possibilità di andare ad una pagina descrittiva del prodotto come ad esempio la sua pagina ufficiale sul sito dell'azienda proprietaria. La seconda funzionalità è quella di mostrare all'utente un breve video promozionale specifico del prodotto. La terza funzionalità è quella di far apparire un secondo menù in sovrimpressione nel quale sono presenti: l' informazione sul prezzo del prodotto sul sito Amazon.it, un link utilizzabile per andare a tale pagina e un link ad un sito che confronta i prezzi di esso su vari siti che ne offrono la vendita.
4- Una volta avviato il video promozionale l'utente può allontanarsi dal prodotto e il video , mostrato in sovrimpressione al menù virtuale, non scompare ma si mantiene nelle prossimità della sua posizione originale e viene semplicemente messo in pausa. Appena l'utente inquadra nuovamente il prodotto il video riprende da dove era rimasto.

5- Mostrare il menù virtuale, in condizioni normali ovvero in assenza di play del video, solamente se il prodotto è inquadrato e farlo scomparire dal momento in cui l'utente sceglie di spostare la fotocamera.

Nonostante l'approccio seguito è quello di avere già presente e disponibile all'applicazione un database contenente le informazioni associate ad ognuno dei prodotti un seguito logico allo sviluppo dell'applicazione e di interesse sarebbe quello di ottenere tali informazioni dal web mediante processi online in background.

## 3.2 Unity

Unity è un motore grafico e allo stesso tempo uno strumento software impiegato per lo sviluppo multipiattaforma di applicazioni. Nato con finalità principale quella dello sviluppo di videogiochi in particolar modo in ambito mobile ha esteso le sue potenzialità nel development iniziando a supportare lo sviluppo di applicazioni per la realtà virtuale e aumentata mediante librerie come la già introdotta ARCore e la più generale ARFoundation successivamente approfondita. Le componenti che concorrono al funzionamento di una generica applicazione sviluppata con Unity [19] sono:

**Scenes**: Una volta creato un progetto Unity all'interno di questo è possibile creare più di una Scene. Una Scene è pensabile come l'applicazione stessa in quanto ha ad essa associata un suo menù e variabili d'ambiente. Procedendo inoltre con la compilazione ad esempio per piattaforma Android il file di installazione .apk creato è relativo alla scene selezionata precedentemente all'operazione di compilazione.

**GameObject:** Un GameObject è l'elemento basilare di una scene in Unity, sono le entità che costituiscono l'applicazione e possono essere di varia natura. Una luce, un qualsiasi oggetto bidimensionale o tridimensionale , una finestra di testo sono tutti esempi di entità che in Unity rientrano sotto la categoria di GameObjects.

**Components**: I Gameobjects sono tuttavia elementi statici, non possono eseguire azioni né vi sono azioni intrinseche associate ad essi. Questi necessitano che venga allegato a loro una component. Questa è di fatto un file script scritto in uno dei linguaggi supportati da Unity e dalla piattaforma target per l'esecuzione dell'applicazione in sviluppo. L'assegnazione di uno script ad un Gameobject ne determina il funzionamento, descritto dal codice componente lo script stesso. L'assegnazione in Unity di una componente ovvero di uno script ad un Gameobject può avvenire in modo esplicito ovvero aggiungendo lo script in maniera grafica dall'editor di Unity o in maniera implicita ovvero mediante il codice dello stesso script. Una componente fondamentale è la Transform che contiene la posizione e orientazione del Gameobject a cui è assegnata relativamente all'origine nel mondo. Di particolare utilità è il meccanismo che fa sì che le strutture dati se definite opportunamente nello script, che come detto costituisce una certa componente, sono inizializzabili nel menù chiamato Inspecor di Unity che mostra le varie componenti associate al Gameobject selezionato.

**Prefabs:** Unity permette il salvataggio di Gameobjects creati insieme a tutte le componenti a loro associate con relativi valori eventualmente assegnati. Queste componenti sono utili al fine di aggiungere un Gameobject precedentemente creato nella scena o rimuoverlo da essa. Da notare

come dal punto di vista dell'applicazione se un Gameobject non è presente nella scene è come se non esistesse.

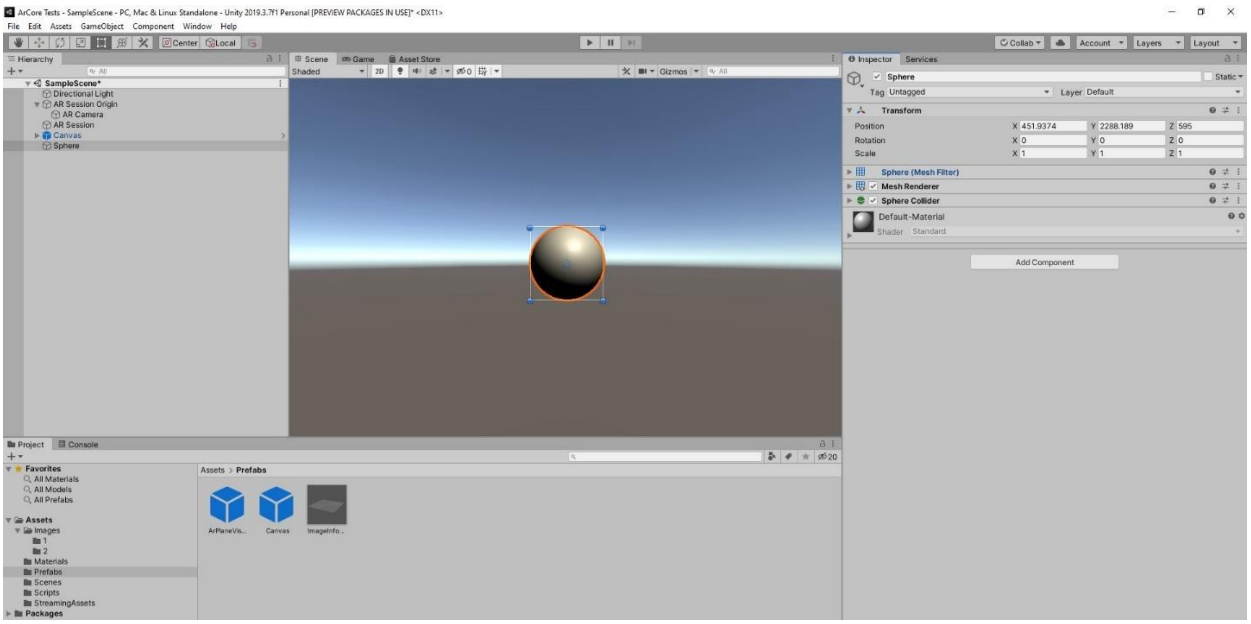


*Figura 26:Scene Unity e varie componenti.*

Va notato come queste sono le componenti di un'applicazione Unity da un punto di vista concettuale, l'implementazione sotto forma di strutture dati è varia e non approfondita in questo contesto.

## 3.3 ARFoundation

Di vitale importanza per lo sviluppo dell'applicazione proposta, nel contesto di Unity, è stato ARFoundation. ARFoundation è definito come un pacchetto di Unity che comprende diverse API di basso livello, comprende inoltre funzionalità aggiuntive come il management del lifecycle della componente AR session e la creazione di Gameobjects utilizzati per rappresentare elementi caratteristici riconosciuti dell'ambiente. Entrambi i termini sono relativi ad Unity e il loro significato verrà illustrato in seguito. Le low level API di cui si fa riferimento, afferma il manuale di ARFoundation [16], facenti parte del namespace UnityEngine.Experimental.XR consistono in un insieme di classi dette Subsystems. Una definizione alternativa di ARFoundation è quella di un insieme di classi derivate dalla classe MonoBehaviour, classe genitore di ogni script facente uso del linguaggio C#, utilizzato per lo sviluppo dell'applicazione in questo lavoro di tesi. Nella documentazione relativa alla prima versione di ARFoundation [16] viene affermato come questo insieme di classi supportano i seguenti concetti:

1- Rilevamento di superfici piane.
2- Point clouds, terminologia utilizzata in ARFoundation per riferirsi ad un insieme di interest points individuati in un frame[17].
3- Reference points: coppie di posizioni-orientamento rispetto alle quali viene fatto il tracking della fotocamera.

4- Light estimation: la stima della media della temperatura di colore e intensità della luce presente nell'ambiente.
5- World tracking: il tracking discusso ovvero quello della fotocamera, del dispositivo, il calcolo della sua pose cioè posizione e orientamento nello spazio.

La più recente versione [20] vede diverse funzionalità aggiuntive alcune delle quali:

6- Face tracking: il riconoscimento e tracking di facce umane.
7- 2d Image Tracking: funzionalità alla base dello sviluppo dell'applicazione illustrata in questa tesi e che consiste nel riconoscimento e tracking di immagini.
8- 3d Object Tracking: medesima operazione del caso 2d ma nel caso più complesso di dover riconoscere oggetti tridimensionali.

Alla base del funzionamento di ARFoundation vi sono ARCore e ARKit due software development kits, o pensabili anche come APIs sviluppate da Google e Apple rispettivamente al fine di permettere a sviluppatori interessati di creare applicazioni per la realtà aumentata[18]. La prima è finalizzata allo sviluppo su dispositivi Android mentre la seconda su dispositivi iOS. Quello che fa ARFoundation è offrire un'unica interfaccia di alto livello per entrambe, utilizzando ovvero le sue strutture dati e i suoi metodi non si è consapevoli di quale delle due API si sta utilizzando. Essendo ARFoundation il pacchetto utilizzato per lo sviluppo in questa tesi verranno ora descritte le principali classi che ne costituiscono il funzionamento.

### 3.3.1 Componenti fondamentali

In ogni scene di Unity in cui si vuole sviluppare un'applicazione facente uso di realtà aumentata ARFoundation prevede l'obbligatoria presenza di tre gameobjects: una ARSession, una ARSessionOrigin e una AR Camera, queste corrispondono a loro volta a delle omonime classi costituenti omonimi script.

**ARSession**: Questa componente controlla il ciclo di vita dell'esperienza di realtà aumentata nell'applicazione. Come un qualunque gameobject all'interno di Unity può essere disabilitata o abilitata a runtime mediante l'utilizzo di apposito codice in uno degli script che concorrono al funzionamento dell'applicazione. La disattivazione dell' ARSession fa sì che l'applicazione non esegua la rilevazione delle caratteristiche dell'ambiente come le superfici. Con session si fa riferimento in ARFoundation allora a questo processo di continua identificazione. Può infine esserci una sola ARSession nella scene di Unity per un corretto funzionamento dell'esperienza di realtà aumentata nell'applicazione.

**Trackable:** Un trackable è definito in ARFoundation come una qualsiasi entità individuabile e di cui è possibile eseguire il tracking. Esempi sono superfici, point clouds, facce e oggetti tridimensionali. Si parla di trackable anche riferendosi agli oggetti virtuali, gameobjects in Unity, che si desidera aggiungere all'ambiente. Di questi viene infatti mantenuto il tracking proprio come per le caratteristiche individuate nell'ambiente.

**ARSessionOrigin:** Lo scopo primario di questa componente è quello di eseguire la trasformazione della posizione di un trackable dal sistema di coordinate utilizzato dal dispositivo al sistema utilizzato da Unity. Questo viene fatto così che Unity sappia modificare posizione e rotazione del

trackable se necessario in modo tale che l'azione abbia il risultato inteso durante l'esecuzione dell'applicazione.

**Trackable manager**: È previsto che per ogni trackable di cui si intende eseguire il tracking sia presente come componente del gameobject ARSessionOrigin il relativo trackable manager. Da tenere sempre in considerazione è che nonostante ARSession e ARSessionOrigin siano dei gameobjects nella scena sono considerabili anche come components. Si è detto infatti che ogni gameobject è di fatto vuoto e richiede una component , uno script, per funzionare. Ebbene entrambi sono definiti allora da una component col medesimo nome e a loro associata.

**Raycasting:** Processo gestito dal Raycast manager questo consiste nella proiezione nello spazio di un vettore virtuale e nella identificazione di collisioni di questo vettore con dei trackables. I trackables con cui può avvenire una collisione non sono necessariamente presenti nell'ambiente reale come ad esempio potrebbe essere un muro. Questi possono essere come detto anche dei gameobjects virtuali precedentemente inseriti nell'ambiente durante l'esecuzione dell'applicazione. L'origine del vettore virtuale è dato da un punto specificato dall'utente sullo schermo del dispositivo.

**ARCamera**: Altro gameobject e componente di fondamentale importanza questa ha il compito di renderizzare il trackable che si è interessati a vedere inserito nell' ambiente. Mediante il suo manager ARCamera manager è possibile modificare la modalità di focalizzazione come anche attivare la modalità di Light Estimation. Questa ha anche sempre come componente un AR Pose Driver che ha il compito di aggiornare e modificare l'orientamento e posizione della ARCamera nella scene.

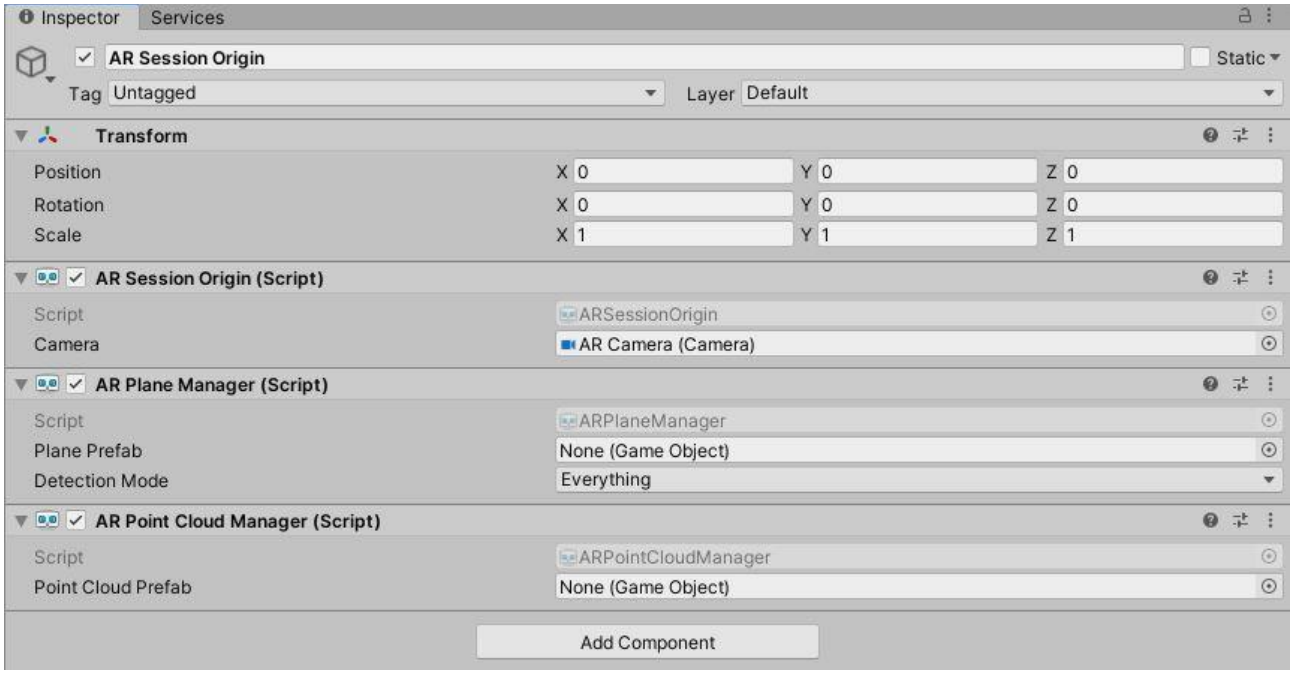


*Figura 27:AR Session Origin e le sue componenti fondamentali.*

## 3.4 Sviluppo

Per lo sviluppo dell'applicazione il cui funzionamento desiderato è stato precedentemente descritto si è fatta la scelta di utilizzare l'Unity Editor e in particolare di creare un progetto 3D in esso. Mediante la funzionalità del package manager si sono aggiunti al progetto pacchetti contenenti le librerie di ARFoundation, ARCore e altre librerie contenenti i Subsystems per la realtà aumentata che mette a disposizione Unity. Ovviamente, per cominciare, sono stati creati i gameobject appena descritti, tra cui AR Session, AR Session Origin, e AR Camera.

### 3.4.1 Image Tracking

L'obiettivo dell'applicazione si è detto essere quello di offire ad un utente una scelta tra azioni che lo informino in modo specifico sul prodotto inquadrato. Il modo in cui si è deciso di implementare ciò è mediante la funzionalità dell' Image Tracking offerta da ARFoundation. Si è deciso ovvero di rappresentare un prodotto con un immagine bidimensionale. Questo ha posto necessariamente dei limiti alla tipologia di prodotto riconoscibile e per cui quindi un set di informazioni personalizzato può essere offerto ad un potenziale cliente. Nel caso infatti di prodotti di natura geometrica non particolarmente lontana dalla forma bidimensionale come libri l'immagine della copertina può essere utilizzata per identificare univocamente il prodotto. Per oggetti come una lampada o una sedia ciò non è in genere possibile. Verranno discussi in seguito nella tesi i limiti dell'utilizzo di questa tecnologia nel riconoscere oggetti tridimensionali. In ARFoundation l'implementazione dell'image tracking richiede l'utilizzo di alcune componenti specifiche una delle quali è la componente AR Tracked Image Manager. Questa viene aggiunta al gameobject ARSession Origin ed ha come funzionalità prefissata quella di istanziare nella scena ovvero nell'ambiente un certo gameobject per ogni immagine individuata.

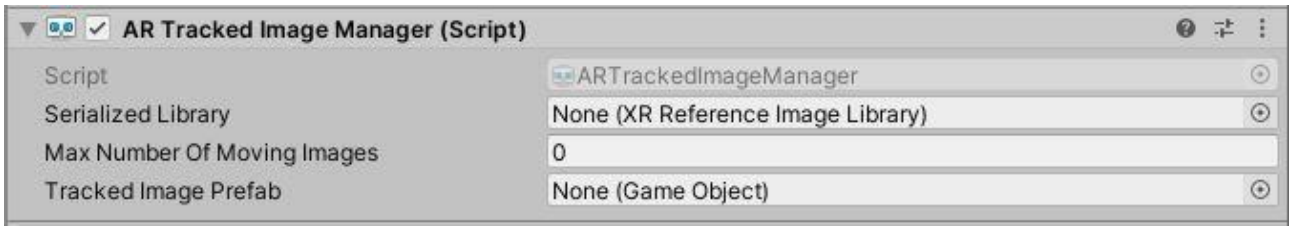


*Figura 28:AR Tracked Image Manager.*

Questo gameobject prende il nome di Tracked Image Prefab e va specificato sotto forma di prefab nell' opportuna casella chiamata in Unity "Serialized Field" all'interno della componente. È possibile specificare il massimo numero di immagini che si desidera riconoscere e di cui si vuole mantenere eventualmente un contemporaneo tracking. Infine va specificato quali sono le immagini di interesse e per fare questo l'AR Tracked Image Manager chiede in input una XR Reference Image Library. Questa è una classe che come attributi prende delle XRReferenceImage ovvero delle immagini con delle informazioni aggiuntive specificabili direttamente in Unity.

### 3.4.2 Problematica iniziale e soluzione adottata

Una volta configurato il manager se si compila ed esegue su uno smartphone l'applicazione quello che si ha è la presenza di uno stream video proprio come nel caso di utilizzo di una fotocamera integrata al dispositivo ma senza il menù classico di interazione per fare foto, video o modificare impostazioni varie. Quando una delle immagini specificate nella XR Reference Image Library viene inquadrata e riconosciuta il prefab designato viene per default istanziato nella scena

sovrapponendosi all'immagine. La posizione di questo varia poi seguendo il movimento dell'immagine se questa si muove. Un problema inizialmente riscontrato è stato nel meccanismo di ARFoundation relativamente all'Image Tracking di istanziare un solo gameobject per ognuna delle immagini riconosciute durante l'esecuzione. La natura problematica di questa meccanismo di default è intuibile dallo scopo dell'applicazione. Questo è infatti quello di fornire informazioni differenti per ciascuno dei prodotti ovvero per ciascuna delle immagini e con un unico prefab istanziato per ogni immagine nel database questo sembrava inizialmente impossibile . La soluzione che si è deciso di adottare ha portato anche alla definizione in maniera più specifica del mezzo con cui queste informazioni vengono presentate all'utente. Si è deciso di fare questo mediante l'istanziazione di un menù virtuale come l'unico gameobject da istanziare a riconoscimento avvenuto. Per fare ciò si è creato come gameobject un menù su Unity, gli sono stati attribuiti come gameobjects figli dei bottoni e si è salvato questo come prefab per poi assegnarlo nella casella vista nel AR Tracked Image Manager. Il meccanismo di distinzione delle informazioni è poi stato scriptato: all'istanziazione del menù ognuno dei tasti ha già a se assegnata l'informazione specifica al prodotto relativo all' immagine riconosciuta. La soluzione di questo problema come lo sviluppo delle caratteristiche specifiche dell'applicazione sono presenti all'interno di un unico script chiamato TrackedImageInfoManager scritto in linguaggio C# e allegato come component alla ARSession Origin.

### 3.4.3 Eventi e gestione

Un'applicazione facente uso di ARFoundation e Image Tracking durante la propria esecuzione gestisce una situazione particolare come evento da gestire. Questo è il cambiamento dell'ambiente inquadrato dovuto anche al semplice e piccolo movimento della fotocamera. Si ipotizza, da quanto visto testando il funzionamento della funzionalità di image tracking, che questo evento sia gestito in modo tale da riacquisire l'immagine se il suo tracking è stato perso o da acquisire un' immagine diversa se questa viene inquadrata. In ogni caso si è deciso di aggiungere alla gestione di tale evento una propria funzione. Si è fatto questo sfruttando le capacità della struttura dati del delegate che mette a disposizione il linguaggio C#.

```
ARTrackedImageManager m_TrackedImageManager;
0 riferimenti
void Awake()
{
    m_TrackedImageManager = GetComponent<ARTrackedImageManager>();

}
0 riferimenti
void OnEnable()
{
    m_TrackedImageManager.trackedImagesChanged += OnTrackedImagesChanged;
}
0 riferimenti
void OnDisable()
{
    m_TrackedImageManager.trackedImagesChanged -= OnTrackedImagesChanged;
}
```

*Figura 29:Registrazione dell'event handler all'evento.*

Nella classe ARTrackedImageManager che costituisce l'omonima componente vista presente nell'AR Session Origin è presente un event facente uso del delgate generico di tipo Action e chiamato trackedImagesChanged. Questo è descritto nella documentazione di ARFoundation come un evento chiamato ad ogni frame e con informazioni riguardanti l'immagine di cui si sta effettuando il tracking. Pertanto, per gestire le informazioni relative al tracking e aggiornare di conseguenza la visualizzazione in app, è sufficiente implementare un handler "OnTrackedImagesChanged" che si registra all'evento nella funzione OnEnable dello script. Si rimuove allo stesso modo dalla gestione dell'evento il proprio metodo quando lo script termina la sua esecuzione. L'handler OnTrackedImagesChanged viene chiamato quindi ad ogni frame e riceve in input le informazioni relative alle immagini target per il tracking. Il fulcro del funzionamento dell'applicazione sta in due cicli in cui per ognuna delle immagini aggiunte o aggiornate viene chiamata la funzione UpdateInfo che prende come parametro una struttura dati che descrive la relativa immagine.

```
foreach (var trackedImage in eventArgs.added)
{
    trackedImage.transform.localScale = new Vector3(0.288f, 1f, 0.192f);
    UpdateInfo(trackedImage);
}

foreach (var trackedImage in eventArgs.updated)
{
UpdateInfo(trackedImage);
}
```

*Figura 30:Cicli alla base del funzionamento dell'applicazione.*

La presenza di questi due cicli come scelta implementativa ha tuttavia creato un secondo problema nella realizzazione del funzionamento inteso per l'applicazione. L'utilizzo della chiamata della funzione all'interno del ciclo comporta la chiamata della funzione tante volte quante sono le immagini complessivamente riconosciute anche se queste non sono più attivamente inquadrate. Questo ha reso impossibile l'utilizzo diretto delle informazioni delle immagini all'interno della funzione UpdateInfo per definire cosa dovesse succedere relativamente al riconoscimento dell'immagine inquadrata in quel momento. Nel caso infatti in cui fossero state complessivamente riconosciute due immagini nel corso dell'esecuzione e si fosse tornati ad inquadare la prima due funzioni UpdateInfo sarebbero state chiamate con ognuna avente le informazioni relative alla propria delle due immagini. Per effettuare una distinzione allora e far sì che vengano eseguite azioni come la personalizzazione del menù in background relativamente alla sola immagine inquadrata si è fatto uso della struttura dati del dizionario messa a disposizione dal linguaggio C#. Per ogni parametro di interesse e di cui è di interesse solamente il valore specifico alla singola immagine è stato creato un relativo dizionario. Ad ognuno di questi viene poi aggiunto in modo opportuno un elemento, ovvero una coppia chiave-valore, al riconoscimento di ogni nuova immagine non precedentemente riconosciuta. Nonostante tutte le funzioni UpdateInfo vengano chiamate ad ogni evento trackedImagesChanged ovvero ad ogni frame è possibile mediante i dizionari eseguire le azioni relative alla sola immagine inquadrata. È possibile ovvero eseguire le azioni desiderate solamente nella chiamata di UpdateInfo che ha ricevuto come parametro le informazioni sull'immagine di interesse in tale momento e in grado di utilizzare quindi le informazioni di questa per personalizzare di conseguenza l'esperienza dell'utente. Un'elenco complessivo delle classi utilizzate sotto forma di gameobjects, components (scripts) è mostrato in figura (31).

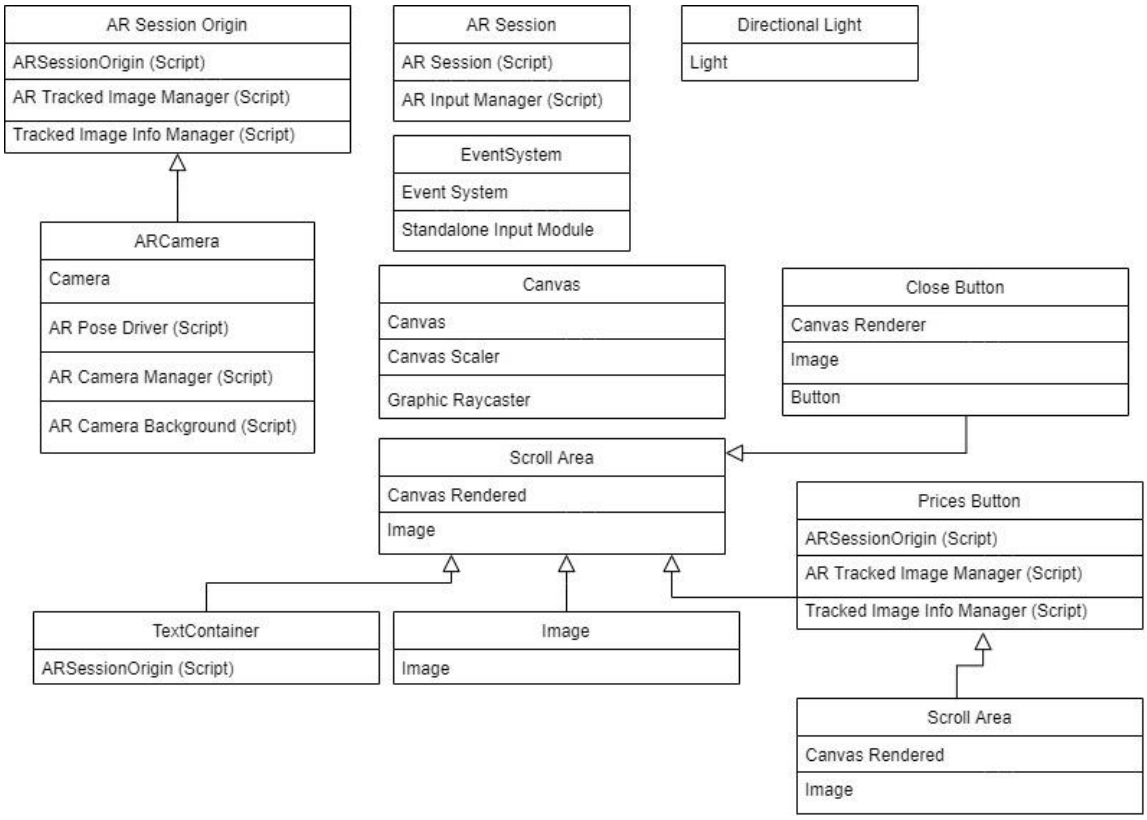


*Figura 31:Class diagram.*

## 3.5 Funzionamento

Il funzionamento dell'applicazione si declina in due step: riconoscimento e interazione.

### 3.5.1 Riconoscimento

All'avvio dell'applicazione all'utente è presentato un feed video identico come detto a quello dato da una fotocamera ma senza nessun comando specifico. Nonostante non sia stata una scelta adottata è possibile in questa fase mediante l'utilizzo di un gameobject di tipo Canvas, creato in precedenza alla compilazione nella scene di Unity, mostrare all'utente una pagina informativa iniziale sul funzionamento dell'applicazione. A questo punto l'utente ha la possibilità di muovere il dispositivo mobile e quello che viene fatto nel frattempo dalle varie componenti dell'applicazione quali l'ARSession è riconoscere elementi caratteristici dell'ambiente in particolar modo superfici piane. All'inquadramento e riconoscimento di una delle immagini inserite nella XR Reference Image Library viene immediatamente istanziato un menù virtuale opportunamente affiancato all'immagine e contenente le tre opzioni principali descritte. Nell'immagine (32) il prodotto è un romanzo di Isaac Asimov la cui immagine di copertina è stata utilizzata per rappresentarlo e stata con successo riconosciuta dall'applicazione una volta inquadrata. Ad istanziamento avvenuto del menù virtuale questo rimane presente e affiancato al prodotto anche se questo si muove, seguendo il suo eventuale movimento in modo da mantenere invariata la sua posizione relativa a questo. Se il prodotto esce dall'inquadratura il suo tracking viene anch'esso perso ed è stata una scelta implementativa quella di associare a tale evento una scomparsa del menù. Questo apparirà nuovamente a fianco del prodotto qualora questo venga rinquadrato e riconosciuto nuovamente. Va notato tuttavia come la funzionalità di default offerta da ARFoundation relativamente all'image

tracking prevede che il gameobject designato all'istanziazione rimanga presente anche qualora la visuale dell'immagine venisse persa. Il gameobject istanziato viene ritrovato nella stessa posizione relativa all'immagine riconosciuta una volta che questi vengono inquadrati nuovamente. Questo suggerisce come una mappa dell'ambiente è stata effettivamente costruita da parte dell'applicazione mediante ARFoundation e ARCore e utilizzando concetti illustrati nel secondo capitolo della tesi tramite la descrizione di specifici algoritmi che li implementano.

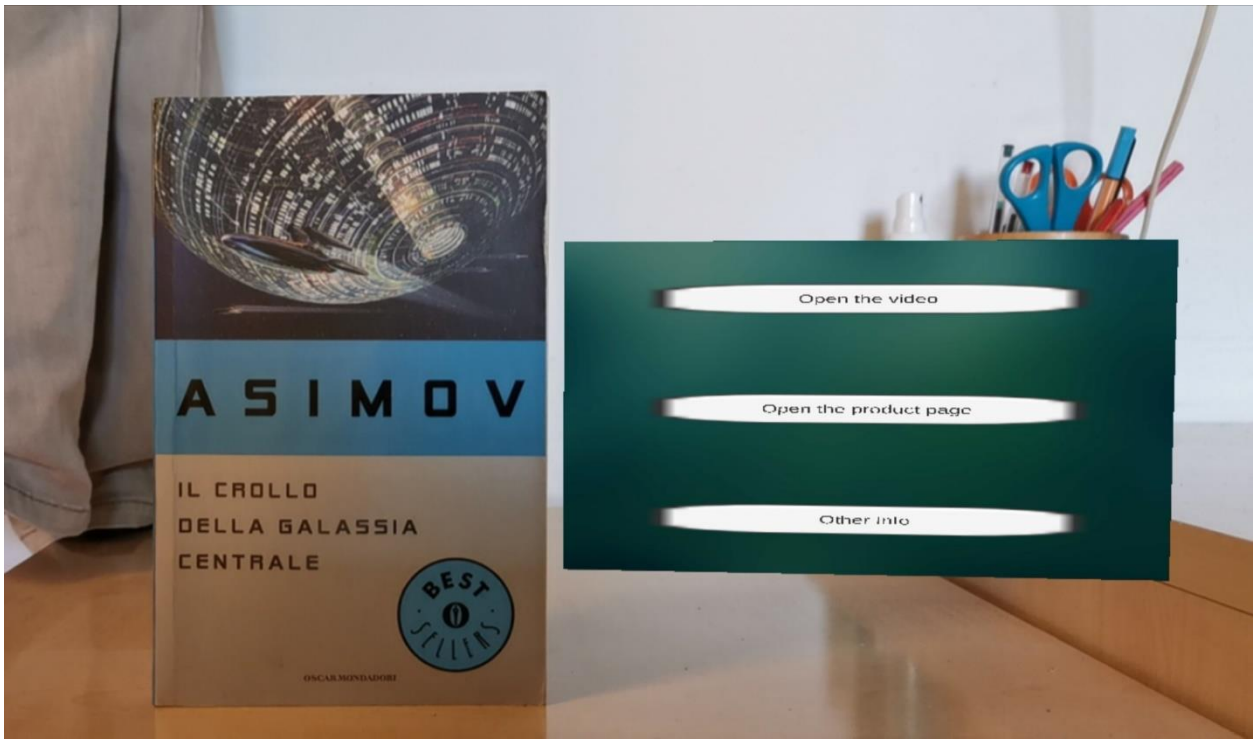


*Figura 32:Istanziazione del menù ad avvenuto riconoscimento del libro.*

### 3.5.2 Video Playback

La prima funzionalità accessibile mediante il menù è quella di vedere un video descrittivo e promozionale del prodotto e che apparirà nella finestra del menù con la conseguente scomparsa dei bottoni virtuali. Per ogni prodotto un video è stato memorizzato in un server e all'interno dello script è presente una logica di distinzione che permette di utilizzare il link relativo al prodotto inquadrato. Una volta che l'utente ha premuto il primo bottone viene attivato l'handler di questo che recupera, mediante l'informazione di qual è il prodotto di interesse, il giusto video dal web. Appena il download del video è stato completato, operazione che soprattutto nel caso di utilizzo di brevi video promozionali dura qualche secondo, inizia il suo playback. Va notato come per un esperienza immediata è possibile mediante Unity inserire all'interno delle risorse utilizzate dall'applicazioni i video promozionali. Per un motivo di efficienza dal punto di vista dello spazio in memoria occupato e per garantire una maggiore continuità con l'eventuale progetto di integrazione di meccanismi online in background è stata adottata la soluzione dipendente dal web. Una volta che il video ha iniziato il suo playback l'utente può allontanarsi o distogliere la visuale dal prodotto, in tal caso entra in funzione il meccanismo di pausa del video che esegue tale azione ogni qual volta il tracking del prodotto viene momentaneamente perso. Alla riacquisizione del tracking ovvero quando l'utente riposiziona il dispositivo in modo da inquadrare il prodotto il video riprende dall'istante in cui era stato messo in pausa. Durante il periodo in cui il tracking del prodotto è stato

perso non lo è stato il tracking propriamente detto e di cui si è parlato fin'ora nei precedenti capitoli. Il menù virtuale con il video in pausa in questo periodo di tempo è infatti intenzionalmente mantenuto istanziato a fianco al prodotto. Anche se ci allontana perdendolo di vista e si ritorna è possibile trovarlo in prossimità della sua posizione iniziale per poi tornare una volta inquadrato mediante un assestamento alla sua esatta posizione precedente. La possibilità di fare ciò denota quindi chiaramente un continuo tracking della fotocamera, ovvero un continuo calcolo della pose di questa relativamente all'ambiente. Allo stesso tempo è chiaramente anche costruita una mappa dell'ambiente che permette all'applicazione di riconoscere la zona in cui era il prodotto prima di averlo perso di vista. Al termine del video si ritorna al menù con la riapparizione dei bottoni.

### 3.5.3 Pagina web e altre informazioni

La seconda funzionalità si basa su meccanismi simili alla prima. L'handler del bottone fa distinzione tra i vari prodotti e identifica e apre il link della pagina informativa relativa al prodotto inquadrato. La pagina è aperta dal browser predefinito nel dispositivo e l'applicazione segue il classico lifecycle di un'applicazione android, viene messa in pausa e quando l'utente decide di ritornare riprende l'esecuzione. Il terzo ed ultimo pulsante ha infine il compito di aprire un menù in sovraimpressione laterale che mostra all'utente una foto e il prezzo del prodotto disponibile sul sito Amazon.it e due bottoni con hyperlink ad essi associati. Il primo bottone permette nello stesso modo offerto dal secondo bottone del menù di andare mediante il browser predefinito del dispositivo alla pagina amazon del prodotto la cui foto e prezzo sono stati mostrati. È infine possibile mediante un link personalizzato a run-time eseguire la ricerca del prodotto inquadrato in un motore di ricerca di prezzi per tale prodotto al fine di trovare il prezzo migliore in caso di interesse all'acquisto.

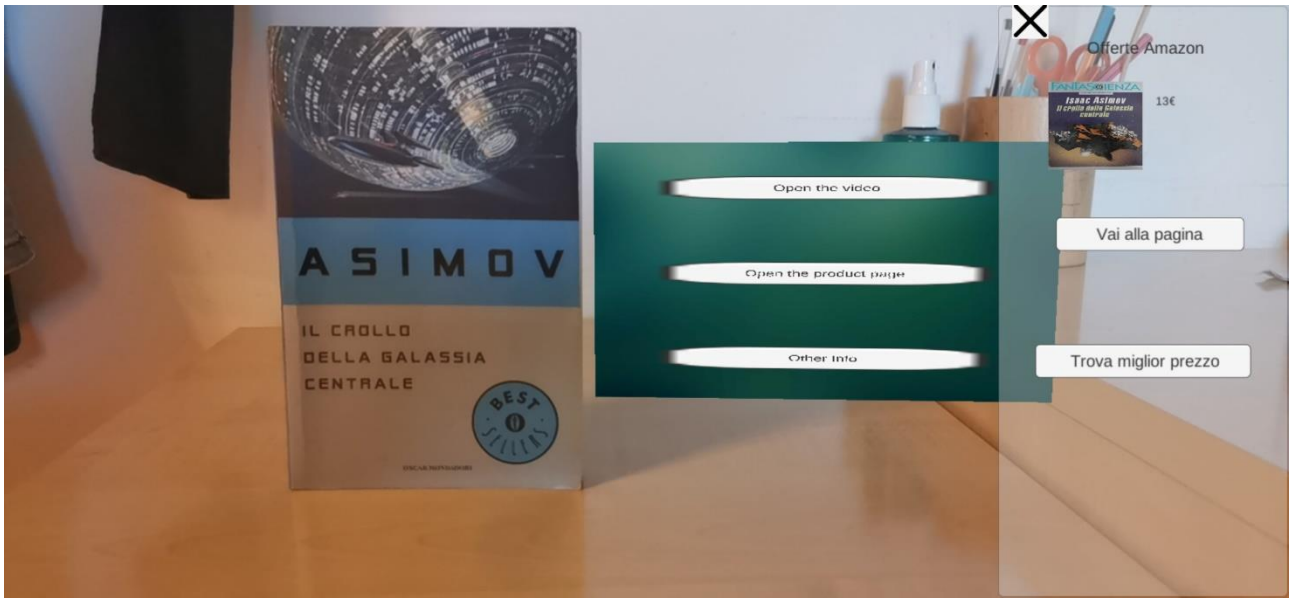


*Figura 33:Menù di acquisto.*

## 3.6 Tests

Come seguito allo sviluppo dell'applicazione si è deciso di testare le capacità della tecnologia offerta da ARFoundation e ARCore tenendo a mente il funzionamento desiderato di un'applicazione per la realtà aumentata. Si è deciso di testare in particolar modo la funzionalità dell'Image Tracking.

### 3.6.1 Riconoscimento di oggetti tridimensionali

Di chiaro interesse in questa tesi relativamente alla realtà aumentata è la capacità dell'applicazione di innanzitutto riconoscere univocamente un prodotto. L'applicazione sviluppata nonostante il suo carattere generale è limitata dalla capacità della funzionalità dell'image tracking di riconoscere un oggetto piuttosto che un altro. Questa è infatti nata per svolgere tale compito su immagini, entità bidimensionali e non conciliabili con oggetti di natura tridimensionale. Si è deciso tuttavia di testare quanto fosse possibile ciò mediante due test. Il primo è stato quello di fare foto a diverse angolazioni dell'oggetto da riconoscere in particolare due sedie differenti in dimensioni e un tavolo. La motivazione dietro la scelta di queste due tipologie di oggetti è stata duplice. Da un lato sono oggetti semplici e tridimensionali, dotati ovvero di una certa profondità ritenuta poter essere problematica nel processo di identificazione e confronto degli interest points. Dall'altro lato sono oggetti privi di superfici piane sufficientemente dettagliate da poter essere utilizzabili come immagini identificative mediante una loro foto. La XR Reference Image Library prevede infatti nel codice che la implementa un test sul sufficiente livello di dettaglio di ciascuna delle immagini che vengono inserite in essa. Un'immagine non sufficientemente dettagliata come la foto di una delle superfici piane di questi oggetti ha portato ad un errore nella fase di compilazione con come motivazione associata quella che tali immagini non erano sufficientemente dettagliate. Il quesito a riguardo rimasto di cui si era interessati a conoscere la risposta era allora se mediante una foto dell'oggetto all'interno del suo ambiente potesse essere poi possibile riconoscerlo. Sono state allora fatte foto di ciascun oggetto da diverse angolazioni e senza rimuovere lo sfondo di questi, ora proiettati sul piano 2-d della loro immagine. Queste sono state poi inserite nella XR Reference Image Library e l'applicazione è stavolta stata compilata con successo in quanto l'ambiente attorno agli oggetti ha reso le immagini acquisite sufficientemente dettagliate. L'ipotesi presa è stata quella che se l'oggetto mantenuto nella stessa posizione relativa all'ambiente in cui sono state prese le sue foto venisse inquadrato nella stessa angolazione ma ad una maggiore distanza il riconoscimento fosse possibile. Questo esperimento non ha avuto tuttavia successo. L'idea del secondo test che si è deciso di fare in questo ambito è nata invece da un aspetto pratico. Si è fatto semplicemente caso ad alcune tipologie di prodotti come bottiglie sulle quali è spesso presente un etichetta. Questi sono oggetti tridimensionali di natura cilindrica sui quali è stata posta un'etichetta, oggetto di natura bidimensionale ma reso tridimensionale una volta "avvolto" attorno alla bottiglia. Da qui è nata l'idea di testare mediante l'immagine di un etichetta bidimensionale la capacità dell'image tracking di riconoscere il prodotto della relativa bottiglia. Questo test è allora sulla capacità di riconoscere mediante l'immagine di un oggetto bidimensionale lo stesso oggetto che ha assunto tuttavia una dimensione aggiuntiva. In questo secondo test è stata allora rimossa l'etichetta da una bottiglia d'acqua da 0,5L, questa è stata posizionata su un tavolo quindi assumendo solamente due dimensioni e una sua foto è stata presa facendo attenzione ad evitare una distorsione dell'immagine. L'etichetta è stata poi riposta sulla bottiglia nella sua posizione originale ed è stato tentato un suo riconoscimento.


*Figura 34:Posizionamento erroneo della finestra virtuale.*

Scegliendo come prefab una semplice finsetra virtuale con il nome dell'immagine al suo centro il risultato trovato è stato che l'immagine è stata riconosciuta ma il prefab istanziato non è stato correttamente posizionato. Anziché essere affiancata all'immagine come specificato nel codice scritto la finestra si posizionava al centro dell'immagine e cambiando periodicamente angolazione. La sua texture non è stata inoltre renderizzata lasciando questa di un colore nero. La motivazione dell'assenza di un corretto allineamento è stata attribuita ad un'incapacità della funzionalità di default dell'image tracking di localizzare le dimensioni dell'immagine curva. È chiaro infatti come la mancanza o erroneità di tali informazioni possa rendere impossibile il corretto posizionamento relativo della finestra. È inoltre possibile che la trasparenza e curvatura della bottiglia abbia impedito il corretto rilevamento della superficie sulla quale l'immagine si trovasse contribuendo al risultato.

### 3.6.2 Dimensione e Illuminazione

Un'altra serie di test è nata dall'intenzione di testare il riconoscimento sotto una varietà di cambiamenti delle immagini e dell'ambiente. Si è stati interessati inizialmente a vedere se banalmente la distanza alla quale il riconoscimento e tracking dell'immagine venisse perso cambiasse all'aumentare delle dimensioni dell'immagine inquadrata. Si è svolto l'esperimento mediante immagini presenti sullo schermo del pc e facilmente ridimensionabili mantenendo invariate le proprie proporzioni. È stato trovato come aspettatosi che la distanza massima di tracking è aumentata all'aumentare delle dimensioni. L'interesse si è poi volto sulla questione illuminotecnica, si è voluto testare il riconoscimento in diverse condizioni di illuminazione, sia intrinseche all'immagine sia dell'ambiente in cui ci si è testato il riconoscimento. Una prima serie di test ha riguardato la luminanza delle immagini mantenendo invariata invece l'intensità luminosa incidente sullo schermo del computer, mediante il quale sono state mostrate le immagini. Le immagini utilizzate sono state scelte opportunamente con una bassa luminanza intrinseca, questa è stata aumentata via software progressivamente ad ogni test e si sono notate due cose. I risultati

tra le varie immagini sono stati i medesimi e in ognuna delle prove è stato necessario modificare l'angolazione del telefono affinché l'immagine venisse riconosciuta. Questo per aumentare la luminanza percepita dalla fotocamera del dispositivo. La seconda cosa notata è stata come ci si aspettava un aumento nella velocità con cui il riconoscimento è avvenuto all'aumentare della luminanza fatto via software. Questa grandezza è stata trovata essere di grande interesse per garantire un facile riconoscimento come anche un mantenimento successivo del tracking di un immagine. Da notare come l'immagine inserita nella libreria è l'originale e priva di alcuna modifica, ad essere stata modificata ai fini del test è stata solo l'immagine sulla quale si è effettuato il test. Risultati simili sono stati ottenuti aumentando la luminosità mediante l'aumento dell'intensità luminosa incidente sullo schermo ottenuta semplicemente utilizzando l'impianto di illuminazione della stanza avente una luce calda quasi monocromatica. Tutto questo sempre in presenza di illuminazione naturale proveniente dalla finestra della stanza.

### 3.6.3 Altri risultati interessanti

Durante i test finora descritti sono stati notati due comportamenti interessanti. Il primo riguarda la velocità con cui una delle immagini inserite all'interno della XR Reference Image Library viene riconosciuta. È stato notato ovvero come sebbene all'avvio dell'applicazione una di queste immagini richiedesse un certo tempo e cambio di angolazione per essere riconosciuta, ad avvenuto riconoscimento la medesima operazione in un istante temporale successivo risultava immediata. Una volta ovvero perso il tracking questo poteva essere riacquisito istantaneamente senza particolari cambi di angolazione semplicemente inquadrando nuovamente l'immagine. Questo fenomeno si è notato poi estendersi a tutte le immagini in modo incrociato. Una volta riconosciuta ovvero una delle immagini con la solita difficoltà iniziale trovata, nella stessa esecuzione dell'applicazione e in un istante temporale successivo, le altre sono state correttamente riconosciute immediatamente. Si è ipotizzata la presenza di una fase iniziale di training che giustifica la difficoltà a riconoscere per la prima volta una delle immagini inserite nella libreria ma che velocizza il riconoscimento successivamente. Il secondo comportamento trovato riguarda infine l'occlusione dell'immagine nello spazio tridimensionale. È stata mostrata a schermo di un computer un'immagine e si è fatta passare una mano tra la fotocamera e lo schermo in modo da testare a che livello di occlusione il tracking venisse perso una volta che l'immagine era stata già riconosciuta. Il risultato trovato è stato che il tracking veniva mantenuto fino al 50% di occlusione del campo visivo dopo di che veniva perso. L'aspetto trovato interessante è stato il fatto che spostando la mano il prefab istanziato rimanesse appoggiato sulla mano come mostrato in figura (36).

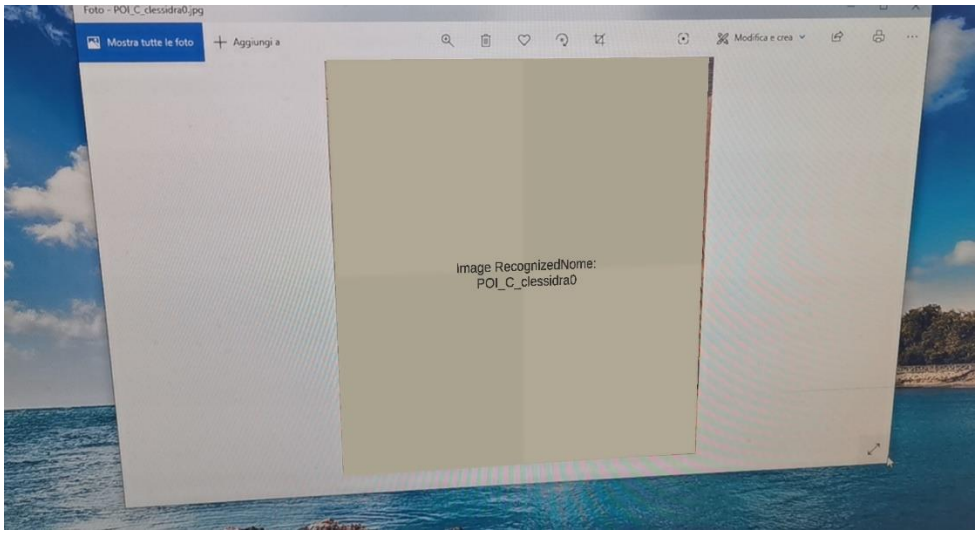


*Figura 35:Corretta istanziazione post riconoscimento.*


*Figura 36:Erroneo posizionamento al movimento della mano dopo un tentativo di occlusione.*

Questo comportamento è stato attribuito al fatto che il piano sul quale ARCore ha individuato l'immagine viene probabilmente sostituito durante l'occlusione dalla mano. Il prefab inizialmente ancorato allo schermo del pc viene allora con tutta probabilità ancorato alla mano durante l'occlusione e la rimozione di essa giustifica un relativo spostamento da parte del prefab istanziato.

# Conclusioni e possibili sviluppi

Con l'intenzione di sviluppare un'applicazione Android facente uso di realtà aumentata ai fini dell'assistenza al cliente si sono approfonditi concetti riguardanti questa disciplina. È stato fatto un quadro generale sulla storia e sul primo sviluppo della tecnologia per l'AR illustrando poi i concetti fondamentali dietro al suo funzionamento. Ciò ha permesso di approfondire come seguito logico alcuni degli algoritmi più utilizzati che permettono di risolvere i vari problemi, la risoluzione sequenziale dei quali permette di realizzare il concetto della realtà aumentata. Questo lavoro ha anche offerto la possibilità di approfondire ARFoundation, libreria di discreto successo nell'ambito dello sviluppo di applicazioni AR e ancora agli albori del proprio sviluppo da parte di Unity e Google. Sono stati testati i suoi limiti e le sue capacità giungendo alla conclusione che nonostante alcune limitazioni ancora presenti offre un'ampia gamma di funzionalità utilizzabili con successo per creare una larga varietà di applicazioni facenti uso dell'AR. Durante lo sviluppo dell'applicazione è stata concepita l'idea di estendere al mondo del web alcuni aspetti di questa in un possibile futuro sviluppo. È pensabile relativamente al menù con prezzo e link amazon ottenere queste due informazioni mediante un'apposita query creata a runtime per garantire dati sempre aggiornati in maniera automatica. Sempre lungo tale linea di pensiero è pensabile costruire a runtime la libreria contenente le immagini da riconoscere e prelevate quindi da un server. Nella sua limitatezza si ritiene quindi che l'applicazione e l'idea dietro essa nascondano una grande potenzialità in un mondo che vede sempre più la presenza di smartphones nelle mani dei suoi abitanti.

## Bibliografia e sitografia

[1] Uchiyama, Hideaki & Marchand, Eric. (2012). Object Detection and Pose Tracking for Augmented Reality: Recent Approaches.

[2] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, 2007, pp. 225-234, doi: 10.1109/ISMAR.2007.4538852.

[3] Stachniss, C. [Cyrill Stachniss]. (2013, 21 Ottobre). SLAM-Course - 00 - Course Introduction (2013/14; Cyrill Stachniss) [File video]. Ottenuto da https://youtu.be/U6vr3iNrwRA .

[4] Ronald T. Azuma. 1997. A survey of augmented reality. Presence: Teleoper. Virtual Environ. 6, 4 (August 1997), 355–385. DOI:https://doi.org/10.1162/pres.1997.6.4.355.

[5] Feng Zhou, H. B. Duh and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR," *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, Cambridge, 2008, pp. 193-202, doi: 10.1109/ISMAR.2008.4637362.

[6] Xin Yang and K.T. Tim Cheng, “Scalable Augmented Reality on Mobile Devices Applications, Challenges, Methods and Software*”, Barfield, Woodrow & Caudell, Thomas. (2001). Fundamentals of Wearable Computers and Augmented Reality.Second edition. pp.195-226*

[7] Huang, Thomas S.. “Computer Vision: Evolution And Promise.” (1996).

[8] Pressigout, Muriel & Marchand, Éric. (2006). Hybrid tracking algorithms for planar and non-planar structures subject to illumination changes. Proceedings - ISMAR 2006: Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality. 52-55. 10.1109/ISMAR.2006.297794.

[9] Zhang, Xiang & Fronz, S. & Navab, Nassir. (2002). Visual marker detection and decoding in AR systems: a comparative study. 97- 106. 10.1109/ISMAR.2002.1115078.

[10] Park, Jun & You, Suya & Neumann, Ulrich. (2002). Natural Feature Tracking for Extendible Robust Augmented Realities.

[11] Jonathan Ventura and Tobias Höllerer, “Urban Visual Modeling and Tracking*”, Barfield, Woodrow & Caudell, Thomas. (2001). Fundamentals of Wearable Computers and Augmented Reality.Second edition. pp.173-194*

[12] Suya You and Ulrich Neumann, “Visual Tracking for Augmented Reality in Natural Environments*”, Barfield, Woodrow & Caudell, Thomas. (2001). Fundamentals of Wearable Computers and Augmented Reality.Second edition. pp.151-171*

[13] Tuytelaars, Tinne & Mikolajczyk, Krystian. (2008). Local Invariant Feature Detectors: A Survey. 10.1561/9781601981394.

[14] Rosten E., Drummond T. (2006) Machine Learning for High-Speed Corner Detection. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg.

[15] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60**,** 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[16] https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/manual/index.html

[17] https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.1/manual/point-cloud-manager.html

[18] https://developers.google.com/ar/discover.

[19] https://docs.unity3d.com/Manual/Components.html

[20] https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@3.0/manual/index.html

[21] https://xinreality.com/wiki/ARCore

[22] https://developers.google.com/ar/discover

[23] Strum , J. [AWE - Augmented World Expo]. (2017, 6 Novembre). Jürgen Sturm (Google): The Computer Vision Technology Underlying ARCore [File video]. Ottenuto da https://youtu.be/1TF7esI3sMQ.

[24] https://www.slideshare.net/AugmentedWorldExpo/jrgen-sturm-google-the-computer-vision-technology-underlying-arcore.

[25] Stachniss, C. [Cyrill Stachniss]. (2015, 9 Luglio). Photogrammetry I - 08a - Matching - Cross Correlation (2015) [File video]. Ottenuto da https://youtu.be/Afvg8SB6Fok .

[26] Cheng, S. [ousam2010]. (2020, 13 Febbraio). 9-Harris corner detector [File video]. Ottenuto da https://youtu.be/5iy_V9FQ72U .

[27] Stachniss, C. [Cyrill Stachniss]. (2016, 12 Gennaio). Photogrammetry II - 10 - SIFT Features and RANSAC (2015/16) [File video]. Ottenuto da https://youtu.be/oT9c_LlFBqs.

[28] Stachniss, C. [Cyrill Stachniss]. (2015, 9 Luglio). Photogrammetry I - 09 - Point Features - Förstner Operator (2015) [File video]. Ottenuto da https://youtu.be/T-6dDOMG2m0.

[29] Stachniss, C. [Cyrill Stachniss]. (2015, 9 Luglio). Photogrammetry I - 08b - Matching - Least Squares Matching (2015) [File video]. Ottenuto da https://youtu.be/JI4QhY8YXAI.

[30] http://www.cse.psu.edu/~rtc12/CSE486/lecture04.pdf

[31] Rosenberg, Louis. (1992). The Use of Virtual Fixtures as Perceptual Overlays to Enhance Operator Performance in Remote Environments. 52.