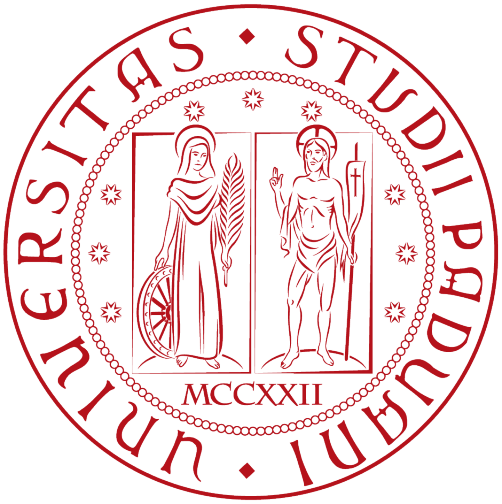
Università degli Studi di Padova
Dipartimento di Matematica
CORSO DI LAUREA IN INFORMATICA



Sviluppo core back-end per la configurazione del sistema e l'analisi di dati di tracking.

Giulio Lovisotto

Relatore: Professoressa Ombretta Gaggi

Anno Accademico 2012/2013

Contents

List of Tables

List of Figures

1 Introduzione

1.1 Scopo del documento

Il seguente documento intende descrivere in maniera dettagliata il contenuto formativo del progetto di stage svolto da Giulio Lovisotto presso la ditta Pathflow s.r.l. L'esposizione verrà strutturata e ordinata seguendo le classiche fasi dello sviluppo software: verrà prima esposta l'analisi, poi i dettagli della progettazione ed infine le informazioni riguardanti i test.

1.2 Pathflow

**********DA SISTEMARE**********
Pathflow è una start up attualmente incubata in H-Farm e WCap. L'idea originale, proposta da Alberto Gangarossa era di creare qTracker. Con il tempo l'idea si è evoluta fino allo stato attuale. *manca qalcosa sui premi* Il team di Pathflow si costruisce in giugno 2013 quando vengono coinvolti nel progetto Marco Baratto, Matteo Noris, Riccardo Greguol e in seguito Alon Muroch. Pathflow riceve un grant da parte di WCap, e in seguito da altri finanziatori (tra i quali H-Farm). L'azienda chiude la fase di seed ad agosto 2013 dopo aver raccolto 130k. Il prodotto entra in fase di beta testing ad ottobre 2013 in 2 store di Telecom Italia situati a Roma. Qua c'è il sito di Pathflow `http://pathflow.co`
****

1.3 Progetto

Il progetto riguarda la realizzazione di un sistema basato su telecamere per la raccolta di dati di tracciamento all'interno dei retail store. Il sistema ha come fine quello di elaborare statistiche a partire dai dati raccolti. Le statistiche in questione serviranno a fornire alle figure professionali che si occupano del marketing (quali visual merchandiser e marketing manager) la possibilità di migliorare l'esperienza di acquisto del cliente e di ottimizzare i flussi di vendita.

Le informazioni verranno raccolte su una piattaforma cloud e rese disponibili agli utilizzatori del sistema tramite delle dashboard, che forniscono dei report e dei grafici che presentano le statistiche in maniera chiara e comprensibile.

Dato che il sistema è complesso e vasto nel suo insieme esso è stato diviso in 3 parti:

- **Video Analytics**
- **Back-end Core**
- **Front-end**

Il progetto proposto per lo stagista consiste nella realizzazione del modulo di back-end core.
Le funzionalità di tale sottosistema comprendono una parte di configurazione del sistema e una parte di generazione statistiche e dati grafici.

Nello specifico esso deve permettere l'impostazione dei valori di configurazione delle telecamere, e salvare tali dati in maniera persistente, evitando che vengano perse le opzioni salvate. Tale funzionalità si può suddividere ulteriormente tra vera e propria calibrazione delle telecamere e impostazione di criteri di trasformazione dei dati di tracking (il significato verrà approndito in seguito).
La parte di statistiche comprende la generazione di un *heatmap* della planimetria del locale. Essa a partire dai dati di tracciamento dev'essere in grado di produrre una visualizzazione grafica delle diverse concentrazioni di movimento all'interno dell'area del locale. La planimetria del locale verrà fornita dal cliente in formato vettoriale .DXF.

Deve essere inoltre in grado di generare alcune statistiche che riguardano: il conteggio delle persone presenti (*counting*), il tempo di attesa davanti alla cassa (*waiting line*), il tempo di ritorno nel locale (*dwell time*), e il tempo trascorso all'interno di alcune aree preimpostate (*bounce rate*).

Tale software è inteso per l'utilizzo solo da parte di un utente amministratore (interfaccia grafica molto semplice), che al momento dell'installazione si preoccupa di configurare i dati necessari per il corretto funzionamento del sistema nel suo insieme. Il sistema verrà poi impostato per eseguire periodicamente il trasferimento dei dati generati nel server locale (quello che risiede nello store) verso il server cloud. Il software dovrà funzionare sulle principali piattaforme (Mac OS/Linux/Windows).

2 Pianificazione

In questa sezione viene riportato un diagramma di Gantt riassuntivo della pianificazione del lavoro nel periodo di stage. Come si può vedere le ore sono state suddivise secondo 4 obiettivi, che rappresentano parti di sistema indipendenti che verranno sviluppate in maniera autonoma.

Come si può riscontrare nelle attività riportate nel diagramma lo svolgimento dello sviluppo non seguirà un classico modello a cascata. E' infatti previsto che dopo una prima parte di analisi generale del sistema (che servirà a comprendere le relazioni tra le varie parti e le caratteristiche e funzionalità del software nella sua interezza), l'implementazione concreta delle varie parti avvenga secondo un modello che prevede la suddivisione del lavoro in piccoli incrementi. Ciò è reso possibile dalla completa indipendenza delle varie parti in cui il core è stato suddiviso. Il modello di ciclo di vita utilizzato può essere paragonato ad uno *scrum*: è infatti emerso nei primi tempi che tale modello era preferibile rispetto alla rigidità dei modelli più classici, in quanto permetteva lo sviluppo continuo dei vari incrementi e la possibilità di mostrare passo per passo i risultati ottenuti. Per chiarezza si riporta una breve descrizione degli obiettivi presenti nel piano di lavoro che verranno maggiormente approfonditi nelle sezioni successive:

Obiettivo 1 Generazione di un *heatmap* a partire dall'input di un file .DXF che descrive la planimetria del locale e da un file .CSV che contiene una lista di coordinate di tracciamento

Obiettivo 2 Realizzazione di un tool per la calibrazione delle telecamere

Obiettivo 3 Integrazione nel core di una base di dati per la consistenza dei dati e delle configurazioni

Obiettivo 4 Recupero di informazioni statistiche significative a partire dai dati di tracciamento

In figura 1 è presente il diagramma di Gantt di cui sopra. Le voci del menù che portano la voce verifica sono da intendersi come ore dedicate all'esecuzione dei test precedentemente definiti.

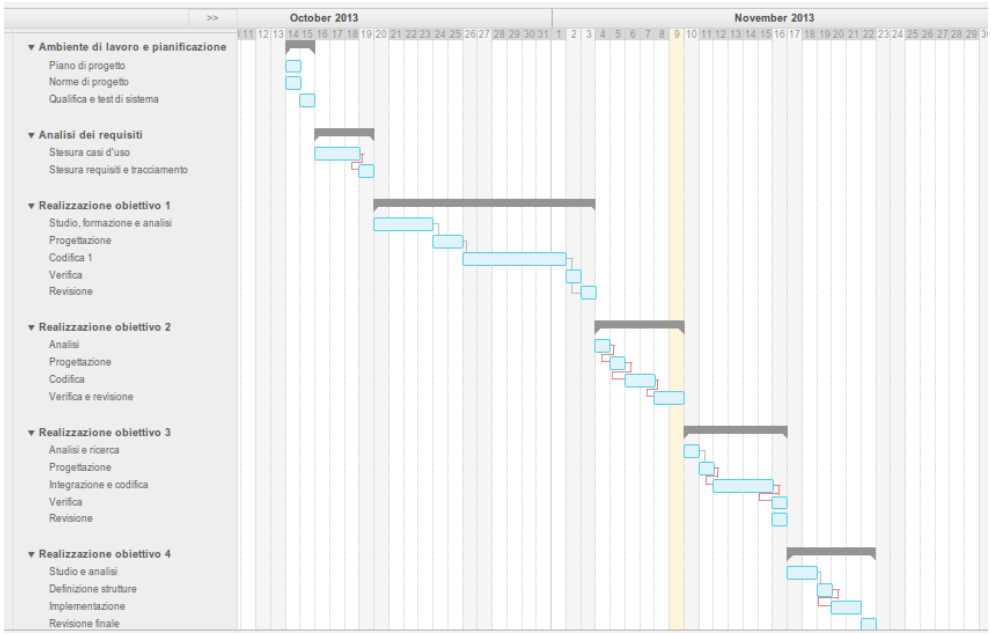


Figure 1: diagramma di Gantt che descrive la pianificazione

3 Tecnologie Utilizzate

In questa sezione vengono brevemente introdotte le tecnologie utilizzate nello sviluppo dell'applicazione, e vengono esposte le motivazioni che hanno portato alla scelta di tali tecnologie.

3.1 OpenCV

OpenCV è una libreria open source il cui obiettivo è quello di fornire un'API per il supporto al real-time computer vision. Essa è sviluppata da Intel ed attualmente manutenuta da *itseez* (`http://itseez.com/`). OpenCV è rilasciata con licenza BSD open source, ed è libera per quanto riguarda l'utilizzo accademico e commercial. E' scritta in C/C++ e fornisce le interfacce per diversi linguaggi: C++, C, Python, MATLAB e Java.

E' cross platform e supporta i più comuni sistemi: Windows, Linux, Mac OS, iOS e Android.
OpenCV viene ufficialmente rilasciata nel 1999, attualmente l'ultima versione stabile ufficiale è la 2.4.6.
OpenCV è una libreria molto ampia e completa, è attualmente utilizzata in diverse applicazioni complesse (molte che riguardano la *video surveillance*), inoltre ha alle spalle anni di sviluppo e una comunità di utenti molto vasta.
L'intero sistema che l'azienda Pathflow vuole realizzare è basato su tale libreria, nelle specifiche del progetto di stage è stato specificato che essa verrà utilizzata per quanto riguarda le funzioni relative alla *computer vision*.

3.2 Qt

Dato che il progetto prevede la realizzazione di una minimale interfaccia grafica si è scelto di utilizzare il framework Qt.
Le valutazioni che hanno portato a tale scelta, sono le seguenti:

- **Portabilità:** Qt è una libreria cross platform. Nelle specifiche del progetto è emerso che sarebbe stato fondamentale che il software fosse portabile. Altra caratteristica che ha influito è che Qt offre un buon supporto per effettuare manipolazioni grafiche e disegno, e dopo una ricerca si è rivelata la migliore alternativa tra le librerie disponibili (sono state valutate altre librerie alternative per il solo disegno, ma esse si sono rivelate meno complete e spesso scarsamente documentate).
- **Licenza:** Qt è infatti disponibile con licenza GNU LGPL v2.1 ed è quindi possibile utilizzarlo senza che sia necessario rilasciare il sorgente del prodotto.

3.3 MySQL

Il database scelto per il salvataggio dei dati è MySQL. Le motivazioni di tale scelta sono nuovamente la necessità di avere un *DBMS* cross platforme e open source. In fase di studio sono state valutate anche altre alternative, che vengono qui brevemente elencate con i relativi difetti che hanno comportato la preferenza per MySQL:

- **memsql** è un database che nasce per fornire supporto al real-time analytics. Viene preso in esame per le sue caratteristiche di velocità che renderebbero l'elaborazione del grande carico di dati molto veloce. Tali caratteristiche sono dovute al fatto che memsql è un database che risiede in memoria (RAM), e che non esegue direttamente le query, ma le converte in codice C++ (con GCC) e poi esegue il codice oggetto. Sebbene tali caratteristiche sono molto favorevoli si è preferito non utilizzare memsql in quanto essendo un progetto relativamente nuovo non ha ancora un supporto completo delle caratteristiche di un database relazionale e in quanto c'era la necessità di appoggiarsi su una soluzione solida che avrebbe dato certezze nel tempo.
- **mongoDB** è un database di tipo non relazionale (*document-oriented*). Esso si basa su documenti stile JSON con schemi dinamici. Anche mongoDB è stato preso in considerazione per le sue caratteristiche di velocità che avrebbero fatto comodo per l'elaborazione dei moltissimi dati di tracciamento. Si è però preferito basarsi su un database di tipo relazionale in quanto la tipologia di dati che andava memorizzata era strettamente di tale tipo.

3.4 MySQL wrapped

Per facilitare il *system management* è stato scelto di usare una semplice libreria che fornisce le funzionalità di *Object Relational Mapping* per un database di tipo MySQL. Tra le varie opzioni si è scelto di usare MySQL Wrapped, molto semplice e leggera. Essa inoltre fornisce un tool di generazione codice C++ a partire dall'output del comando *mysqldump*. In tale modo si è potuto progettare prima lo schema del database e in seguito creare le rispettive classi con lo script *sql2class*. MySQL Wrapped crea un ulteriore livello di astrazione a partire dall'API di connessione in C fornita da MySQL.

3.5 dxflib

Il file .DXF è un file di output di programmi di grafica vettoriali quali AutoCAD. Esso contiene al suo interno delle entità che definiscono gli elementi che descrivono un immagine (generalmente 2D).
Il formato dei file .DXF è noto e ben documentato nei manuali forniti da AutoCAD. Per le funzionalità di parsing del file .DXF si è scelto di utilizzare dxflib: una libreria in C++ open source. Essa si occupa della lettura del file e definisce delle interfacce che possono essere ridefinite per ottenere il comportamento desiderato. I dettagli verranno trattati meglio in seguito.

3.6 CMake

CMake è un sistema di build cross platform e open source. Nasce per fornire un sistema di management del processo di build del software. Per ottenere ciò CMake utilizza dei metodi che non dipendono dal tipo di compilatore presente nel sistema operativo, ma da alcuni file di configurazione autonomi. CMake genera dei makefiles che possono in seguito essere eseguiti con il comando di make relativo all'ambiente utilizzato (*make* per Linux, *nmake* per Visual Studio etc). CMake supporta sia build di tipo *in-place che* out-of-place, supporta inoltre linking di tipo statico e dinamico.

Per il suo utilizzo è sufficiente creare dei file denominati CMakeLists.txt che contengono le direttive per la creazione del makefile.
Cmake è stato rilasciato sotto licenza di tipo *New BSD License*.
L'utilizzo di CMake come sistema di build è stato imposto nelle prime fasi dello sviluppo, per mantenere la coerenza con quanto era già stato fatto e per rispettare le caratteristiche di cross platforming dell'applicativo da realizzare.

4 Analisi dei Requisiti

4.1 Casi d'uso

In questa sezione verranno elencati i casi d'uso del sistema che è oggetto dello stage. Dato che il sistema è stato pensato per rispondere solo all'intervento di un utente che si occupa di amministrazione e installazione l'unico attore che prenderemo in considerazione è **utente**. Per ogni caso d'uso verranno riportate:

1. DESCRIZIONE: contenuto del caso d'uso
2. FLUSSO PRINCIPALE DEGLI EVENTI
3. PRECONDIZIONI: asserzioni che sono valide prima dell'effettiva esecuzione del caso d'uso
4. POSTCONDIZIONI: asserzioni che sono valide dopo l'esecuzione del caso d'uso
5. SCENARI ALTERNATIVI: eventuali scenari alternativi che differiscono dal normale flusso del caso d'uso

Ogni caso d'uso ha un identificativo stile UCn dove n indica una posizione gerarchica. Ogni caso d'uso è posizionato all'interno della gerarchia che parte dal caso d'uso più generale UC0 (radice dell'albero). Per ogni caso d'uso figlio valgono le precondizioni del padre.

4.1.1 UC0: Scenario principale

descrizione L'utente ha avviato il sistema il quale è pronto a rispondere agli eventi. L'utente può scegliere di accedere alla calibrazione delle telecamere, accedere alla loro configurazione, oppure generare le statistiche a partire dai dati di tracking attualmente presenti nel sistema (figura 2)

flusso principale degli eventi

1. L'utente può calibrare le telecamere (UC1)
2. L'utente può configurare le telecamere (UC2)
3. L'utente può generare e visualizzare le statistiche (UC3)

precondizione Il sistema è avviato e funzionante

4.1.2 UC1: Calibrazione telecamere

descrizione L'utente può iniziare la calibrazione delle telecamere, oppure può visualizzare un video *undistorted* che usa i parametri di calibrazione per correggere i frame prodotti dal video. (figura 3)

flusso principale degli eventi

1. L'utente può iniziare la calibrazione (UC1.1)
2. L'utente può visualizzare il video corretto con i parametri di calibrazione (UC1.2)

precondizione Il sistema è avviato e funzionante

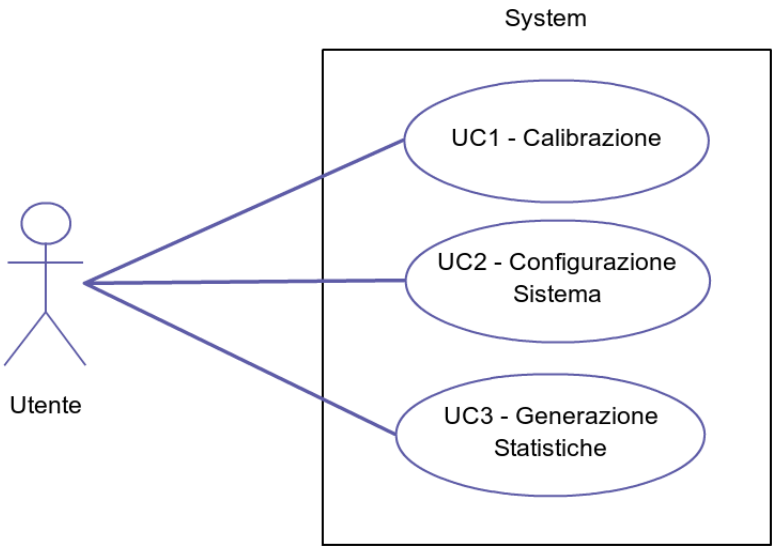


Figure 2: UC0 - Scenario principale

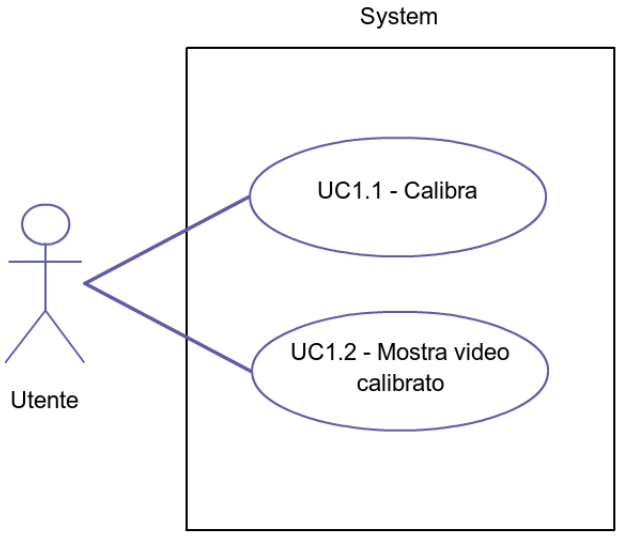


Figure 3: UC1 - Calibrazione telecamere

4.1.3 UC1.1: Calibra

descrizione L'utente sta selezionando le opzioni per la calibrazione delle telecamere. Può selezionare: numero di immagini da utilizzare nella calibrazione, il frame step indirizzo IP nella rete della camera da calibrare, e poi iniziare l'effettiva calibrazione. (figura 4)

flusso principale degli eventi

1. L'utente seleziona il numero di immagini da utilizzare (UC1.1.1)
2. L'utente seleziona il frame step (UC1.1.2)
3. L'utente seleziona l'indirizzo della camera da calibrare (UC1.1.3)
4. L'utente avvia la calibrazione calibrare (UC1.1.4)

postcondizione I file che contengono i file di calibrazione della telecamera (distortion e instrinsics) sono stati generati e salvati nel file system

scenari alternativi

1. L'utente interrompe la procedura, il sistema torna in attesa di eventi
2. La procedura di calibrazione fallisce, il sistema segnala l'errore e torna in attesa di eventi

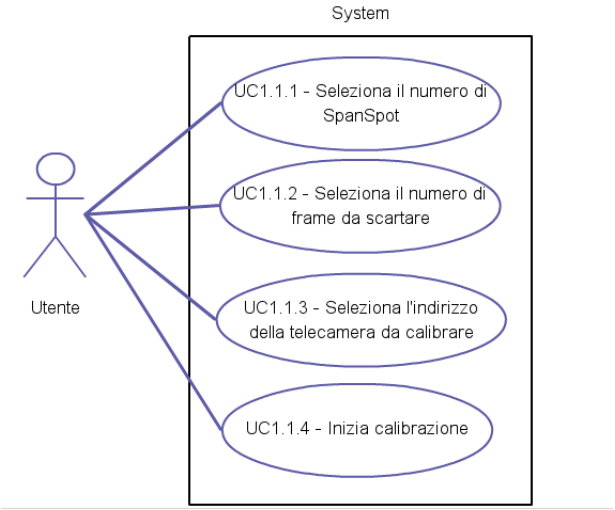


Figure 4: UC1.1 - Calibra

4.1.4 UC1.1.1: Seleziona il numero di immagini da utilizzare

descrizione L'utente seleziona il numero di immagini da utilizzare per la calibrazione

postcondizione Il sistema conosce il numero di immagini da utilizzare da utilizzare nella calibrazione

scenari alternativi

1. L'utente interrompe la procedura, il sistema ritorna in attesa di eventi

4.1.5 UC1.1.2: Seleziona il frame step

descrizione L'utente seleziona il frame step per la calibrazione

precondizione L'utente ha già selezionato il numero immagini da utilizzare

postcondizione Il sistema conosce il frame step per la calibrazione

scenari alternativi

1. L'utente interrompe la procedura, il sistema ritorna in attesa di eventi

4.1.6 UC1.1.3: Inserisce l'indirizzo della telecamera da calibrare

descrizione L'utente inserisce l'indirizzo della telecamera da calibrare

precondizione L'utente ha già selezionato il numero immagini da utilizzare e il frame step per la calibrazione

postcondizione Il sistema conosce l'indirizzo della telecamera da calibrare

scenari alternativi

1. L'utente interrompe la procedura, il sistema ritorna in attesa di eventi

4.1.7 UC1.1.4: Inizia calibrazione

descrizione L'utente inizia la calibrazione effettiva

precondizione L'utente ha già selezionato le opzioni di calibrazione (UC1.1.1 - UC1.1.2 - UC1.1.3), il sistema quindi conosce i parametri da utilizzare

postcondizione Il sistema genera i file di calibrazione intrinsics e distortion e li salva nel file system

scenari alternativi

1. La calibrazione non termina con successo quindi l'operazione viene interrotta e l'errore segnalato. Il sistema ritorna in attesa di eventi

4.1.8 UC1.2: Mostra video calibrato

descrizione L'utente visualizza il video corretto con i valori di calibrazione

precondizione L'utente ha già effettuato con successo la generazione dei file di calibrazione che sono presenti nel sistema

4.1.9 UC2: Configurazione telecamere

descrizione L'utente può impostare le opzioni di configurazione delle telecamere, aggiungendone di nuove, rimuovendole, modificandole. Può inoltre ottenere i file necessari alla configurazione, convertendo un file .DXF in .PNG, salvando un frame dalla telecamera, o calcolare la *homography matrix* (figura 5)

flusso principale degli eventi

1. L'utente inserisce una nuova telecamera nel sistema (UC2.1)
2. L'utente rimuove una telecamera esistente (UC2.2)
3. L'utente modifica la configurazione di una telecamera (UC2.3)
4. L'utente cattura e salva un frame della telecamera (UC2.4)
5. L'utente converte un file .DXF in .PNG (UC2.5)
6. L'utente calcola la matrice omografica relativa ad una telecamera (UC2.6)
7. L'utente seleziona una telecamera (UC2.7)

precondizione Il sistema è avviato e funzionante.

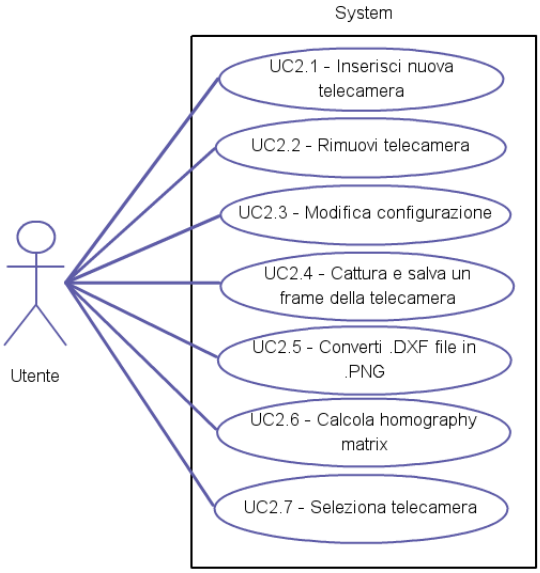


Figure 5: UC2 - Configurazione telecamere

4.1.10 UC2.1: Inserisci nuova

descrizione L'utente inserisce il nome della nuova telecamera

postcondizione Il sistema ha inserito la nuova telecamera nel database

scenari alternativi

1. Il nome inserito non è valido, il sistema interrompe l'operazione e torna in attesa di eventi.

4.1.11 UC2.2: Rimuovi telecamera

descrizione L'utente rimuove la telecamera selezionata

precondizione L'utente ha selezionato una telecamera tra quelle esistenti

postcondizione Il sistema ha rimosso la telecamera dal database

4.1.12 UC2.3: Modifica configurazione

descrizione L'utente ha selezionato una telecamera e ne modifica le opzioni di configurazione, quali: file distortion, intrinsics, valore dell'altezza del frame, valore della larghezza del frame, percorso del file contentente la *homography matrix* (figura **??**).

flusso principale degli eventi

1. L'utente modifica il percorso del file di calibrazione *distortion* (UC2.3.1)
2. L'utente r modifica il percorso del file di calibrazione *intrinsics* (UC2.3.2)
3. L'utente modifica il valore dell'altezza del frame (UC2.3.3)
4. L'utente modifica il valore della larghezza del frame (UC2.3.4)
5. L'utente modifica il percorso del file di configurazione contenente la *homography matrix* (UC2.3.5)

precondizione L'utente ha selezionato una telecamera tra quelle esistenti

scenari alternativi

1. L'utente interrompe la modifica, il sistema ritorna in attesa di eventi

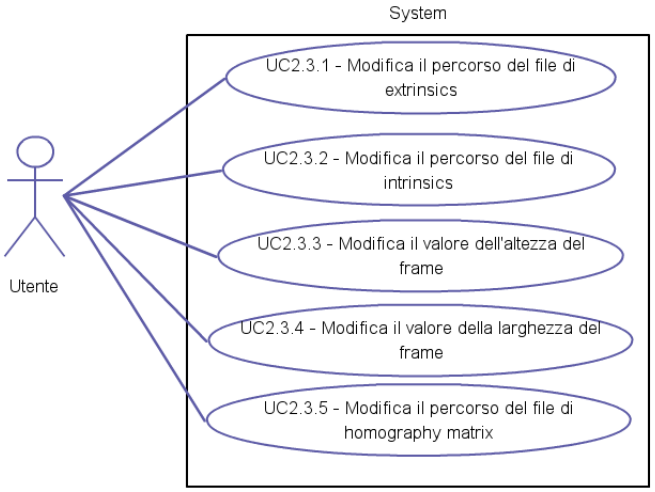


Figure 6: UC2.3 - Modifica configurazione

4.1.13 UC2.3.1: Modifica il percorso del file di distortion

descrizione L'utente seleziona un file nel file system che contiene i valori *distortion* di calibrazione.

postcondizione Il sistema ha modificato il corrispondente valore della telecamera nel database

scenari alternativi

1. L'utente interrompe la selezione del file, il sistema ritorna in attesa di eventi

4.1.14 UC2.3.2: Modifica il percorso del file di intrinsics

descrizione L'utente seleziona un file nel file system che contiene i valori *intrinsics* di calibrazione.

postcondizione Il sistema ha modificato il corrispondente valore della telecamera nel database

scenari alternativi

1. L'utente interrompe la selezione del file, il sistema ritorna in attesa di eventi

4.1.15 UC2.3.3: Modifica il valore dell'altezza del frame

descrizione L'utente seleziona un valore per l'altezza del frame della telecamera.

postcondizione Il sistema ha modificato il corrispondente valore della telecamera nel database

4.1.16 UC2.3.4: Modifica il valore della larghezza del frame

descrizione L'utente seleziona un valore per la larghezza del frame della telecamera.

postcondizione Il sistema ha modificato il corrispondente valore della telecamera nel database

4.1.17 UC2.3.5: Modifica il percorso del file di homography matrix

descrizione L'utente seleziona un file nel file system che contiene i valori che descrivono la *homography matrix* utilizzata per la traduzione delle coordinate.

postcondizione Il sistema ha modificato il corrispondente valore della telecamera nel database

scenari alternativi

1. L'utente interrompe la selezione del file, il sistema ritorna in attesa di eventi

4.1.18 UC2.4: Cattura e salva un frame della telecamera

descrizione L'utente indica al sistema di attivare la telecamera selezionata, catturare un frame e salvarlo.

precondizione L'utente ha selezionato una telecamera tra quelle esistenti

postcondizione Il frame della telecamera selezionata è stato salvato nel file system

scenari alternativi

1. La telecamera non è raggiungibile, il sistema interrompe l'operazione segnala l'errore e ritorna in attesa di eventi

4.1.19 UC2.5: Converti .DXF file in .PNG

descrizione L'utente seleziona nel file system un file .DXF da convertire in formato .PNG

postcondizione Il file selezionato è stato convertito in un immagine che riproduce il contenuto del .DXF

4.1.20 UC2.6: Calcola homography matrix

descrizione L'utente calcola la homography matrix per una particolare telecamera configurata.

postcondizione Il file che contiene la *homography matrix* è stato calcolato e salvato nel file system

4.1.21 UC2.7: Seleziona telecamera

descrizione L'utente seleziona una telecamera tra la lista di telecamere presenti nel sistema

postcondizione La telecamera indicata diventa selezionata e quindi **attiva**

4.1.22 UC3: Generazione statistiche

descrizione L'utente può visualizzare le statistiche relative ai dati presenti nel sistema. Può effettuare la conversione dei dati *raw*, visualizzare l'*heatmap*, o le statistiche (figura 7)

flusso principale degli eventi

1. L'utente seleziona la trasformazione dei dati di tracciamento presenti nel sistema (UC3.1)
2. L'utente seleziona la generazione dell'heatmap (UC3.2)
3. L'utente seleziona la generazione delle statistiche (UC3.3)

precondizione Il sistema è avviato e funzionante.

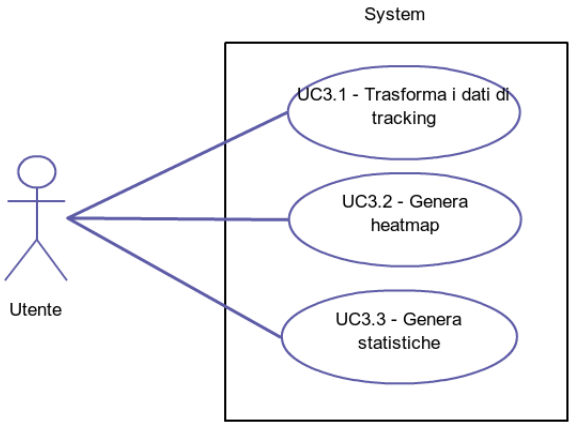


Figure 7: UC3 - Generazione statistiche

4.1.23 UC3.1: Trasforma i dati di tracking

descrizione L'utente indica al sistema di eseguire la trasformazione dei dati *raw* di tracking

postcondizione Il sistema ha trasformato i dati presenti nel database ed ha salvato i dati trasformati.

scenari alternativi

1. La trasformazione fallisce, il sistema segnala l'errore e torna in attesa di eventi

4.1.24 UC3.2: Genera heatmap

descrizione L'utente indica al sistema di generare l'heatmap con la rappresentazione grafica dei dati di tracking

postcondizione Il sistema elabora i dati presenti nel database generando l'heatmap e salvandola nel file system.

scenari alternativi

1. La generazione fallisce, il sistema segnala l'errore e torna nello stato di attesa.

4.1.25 UC3.3: Genera statistiche

descrizione L'utente indica al sistema di generare le statistiche relative ai dati presenti

postcondizione Il sistema ha generato le statistiche e le ha salvate nel database

scenari alternativi

1. La generazione fallisce, il sistema segnala l'errore e torna nello stato di attesa.

4.2 Requisiti

In questa sezione verranno elencati i requisiti software che sono emersi dai casi d'uso individuati e dalle riunioni informative con i leader dello sviluppo. Per chiarezza si è cercato di tenere un rapporto 1:1 tra casi d'uso e requisiti.

Nell'esposizione si userà la seguente convenzione: se un requisito padre ha figli con tipologia diversa tra di loro, il requisito padre assumerà la tipologia più vincolante tra quelle dei figli (esempio: se ci sono 2 figli, uno obbligatorio ed uno desiderabile il padre sarà qualificato come obbligatorio).

4.2.1 Requisiti funzionali

Table 1: Tabella requisiti funzionali

Codice	Descrizione	Tipologia	Fonte
RF1	Il sistema deve permettere all'utente di eseguire la calibrazione delle telecamere	Obbligatorio	UC1
RF1.1	Il sistema deve permettere all'utente di impostare le opzioni di calibrazione	Obbligatorio	UC1 UC1.1
RF1.1.1	Il sistema deve permettere all'utente di selezionare il numero di immagini da utilizzare per la calibrazione	Obbligatorio	UC1 UC1.1 UC1.1.1
RF1.1.2	Il sistema deve permettere all'utente di selezionare frame step per la calibrazione	Obbligatorio	UC1 UC1.1 UC1.1.2
RF1.1.3	Il sistema deve permettere all'utente di selezionare l'indirizzo identificativo della telecamera	Obbligatorio	UC1.1 UC1.1.3
RF1.1.4	Il sistema deve eseguire la calibrazione delle telecamere con le opzioni impostate	Obbligatorio	UC1.1 UC1.1.4
RF1.2	Il sistema deve permettere all'utente di visualizzare un video che utilizza i parametri di calibrazione per correggere i frame	Desiderabile	UC1 UC1.2
RF2	Il sistema deve permettere all'utente di configurare le telecamere	Obbligatorio	UC2
RF2.1	Il sistema deve permettere all'utente di inserire una nuova telecamera	Obbligatorio	UC2 UC2.1
RF2.2	Il sistema deve permettere all'utente di rimuovere una telecamera	Obbligatorio	UC2 UC2.2

Continued on next page

Table 1 – *Continued from previous page*

Codice	Descrizione	Tipologia	Fonte
RF2.3	Il sistema deve permettere all'utente di modificare e salvare in maniera persistente la configurazione di una telecamera esistente	Obbligatorio	UC2 UC2.3
RF2.3.1	Il sistema deve permettere all'utente di modificare il file di distortion di calibrazione di una telecamera	Desiderabile	UC2 UC2.3 UC2.3.1
RF2.3.2	Il sistema deve permettere all'utente di modificare il file di intrinsics di calibrazione di una telecamera	Desiderabile	UC2 UC2.3 UC2.3.2
RF2.3.3	Il sistema deve permettere all'utente di modificare il valore dell'altezza del frame utilizzato per una telecamera	Obbligatorio	UC2 UC2.3 UC2.3.3
RF2.3.4	Il sistema deve permettere all'utente di modificare il valore della larghezza del frame utilizzato per una telecamera	Obbligatorio	UC2 UC2.3 UC2.3.4
RF2.3.5	Il sistema deve permettere all'utente di modificare il file contenente la homography matrix utilizzata per la traduzione delle coordinate di tracking relative ad una telecamera	Obbligatorio	UC2 UC2.3 UC2.3.5
RF2.4	Il sistema deve permettere di salvare un frame preso dal video stream della telecamera	Obbligatorio	UC2 UC2.4
RF2.5	Il sistema deve permettere di convertire un file .DXF in un file .PNG che riproduce graficamente l'immagine contenuta nel file originale	Obbligatorio	Interno
RF2.6	Il sistema deve permettere di calcolare la homography matrix utilizzata per la traduzione delle coordinate di tracking relative ad una telecamera	Obbligatorio	UC2 UC2.6
RF2.7	Il sistema deve permettere di selezionare una telecamera per la modifica	Obbligatorio	UC2 UC2.7
RF3	Il sistema deve permettere all'utente di generare delle statistiche a partire dai dati di tracking attualmente presenti	Obbligatorio	UC3

Continued on next page

Table 1 – *Continued from previous page*

Codice	Descrizione	Tipologia	Fonte
RF3.1	Il sistema deve permettere di trasformare i dati di tracking in coordinate di posizione all'interno di un immagine .PNG che descrive la planimetria del locale, e di salvare tali informazioni in maniera persistente	Obbligatorio	Interno
RF3.2	Il sistema deve permettere di generare un heatmap grafica che fornisca una rappresentazione grafica dei dati di tracking	Obbligatorio	UC3 UC3.2
RF3.3	Il sistema deve permettere di generare delle statistiche numeriche a partire dai dati di tracking e di salvarle in maniera persistente	Facoltativo	UC3 UC3.3
RF3.3.1	Il sistema deve permettere di generare la statistica di *Dwell Time*	Facoltativo	UC3 UC3.3
RF3.3.2	Il sistema deve permettere di generare la statistica di *Counting*	Facoltativo	UC3 UC3.3
RF3.3.3	Il sistema deve permettere di generare la statistica di *Waiting Line*	Facoltativo	UC3 UC3.3

4.2.2 Requisiti di vincolo

Table 2: Tabella requisiti di vincolo

Codice	Descrizione	Tipologia	Fonte
RV1	Il sistema dev'essere strutturato secondo un'architettura 3-tier	Obbligatorio	Capitolato
RV2	Il sistema deve funzionare in ambiente Linux	Obbligatorio	Capitolato
RV3	Il sistema deve funzionare in ambiente Windows	Facoltativo	Capitolato
RV4	Il sistema deve funzionare in ambiente Mac OS X	Desiderabile	Capitolato
RV5	Il sistema utilizzerà come strumento di build CMake, verranno quindi forniti i relativi file	Obbligatorio	Capitolato

4.2.3 Requisiti di qualita

Table 3: Tabella requisiti di qualita

Codice	**Descrizione**	**Tipologia**	**Fonte**
RQ1	Deve essere prodotta documentazione del codice sorgente del software	Obbligatorio	Interno
RQ2	Il sistema deve garantire la massima indipendenza tra le funzionalità	Desiderabile	Capitolato

4.2.4 Tracciamento requisiti

Table 4: Tabella tracciamento requisiti - casi d'uso

Fonte	**Requisiti**
Interno	RF2.5
Interno	RF3.1
Interno	RQ1
Capitolato	RQ2
Capitolato	RV1
Capitolato	RV2
Capitolato	RV3
Capitolato	RV4
Capitolato	RV5
UC1 Calibrazione telecamere	RF1
UC1.1 Calibra	RF1.1
UC1.1.1 Seleziona il numero di immagini da utilizzare	RF1.1.1
UC1.1.2 Seleziona il frame step	RF1.1.2
UC1.1.3 Inserisce l'indirizzo della telecamera da calibrare	RF1.1.3
UC1.1.4 Inizia calibrazione	RF1.1.4
UC1.2 Mostra video calibrato	RF1.2
UC2 Configurazione telecamere	RF2
UC2.1 Inserisci nuova	RF2.1
UC2.2 Rimuovi telecamera	RF2.2
UC2.3 Modifica configurazione	RF2.3
UC2.3.1 Modifica il percorso del file di distortion	RF2.3.1
UC2.3.2 Modifica il percorso del file di intrinsics	RF2.3.2
UC2.3.3 Modifica il valore dell'altezza del frame	RF2.3.3
UC2.3.4 Modifica il valore della larghezza del frame	RF2.3.4
UC2.3.5 Modifica il percorso del file di homography matrix	RF2.3.5
UC2.4 Cattura e salva un frame della telecamera	RF2.4
UC2.6 Calcola homography matrix	RF2.6
UC2.7 Seleziona telecamera	RF2.7
UC3 Generazione statistiche	RF3
UC3.2 Genera heatmap	RF3.2
UC3.3 Genera statistiche	RF3.3
UC3.3 Genera statistiche	RF3.3.1
UC3.3 Genera statistiche	RF3.3.2
UC3.3 Genera statistiche	RF3.3.3

5 Problematiche

In questa sezione verranno descritti brevemente per chiarezza i problemi che il sistema si propone di risolvere, i quali quindi sono stati oggetto di studio per lo stagista che ha dovuto analizzarli e ricercarne una soluzione.

5.1 Camera Calibration

Per *camera calibration* si intende la procedura volta a calcolare la relazione matematica tra le coordinate di un punto posizionato nello spazio e la sua proiezione sull'*image plane* di una telecamera.

5.1.1 Intrinsics parameters

La visione origina dall'individuazione della luce nell'ambiente. La luce parte da una qualche sorgente per poi viaggiare nello spazio fino ad incontrare un oggetto, in quel momento buona parte della luce viene assorbita, la parte che non viene assorbita viene riflessa e determina il colore dell'oggetto.

Il modello più semplice disponibile per la rappresentazione di una telecamera è il cosiddetto *pinhole camera model*. In tale modello la luce arriva da un oggetto distante, ma solo un singolo raggio entra per ogni particolare punto. Tale punto viene proiettato su di una superficie. Ciò comporta che l'immagine presente sull'*image plane* è sempre a fuoco, e la sua dimensione dipende dalla distanza della sorgente della luce (*focal lenght*).

Si può vedere in figura 8) una rappresentazione di tale modello.

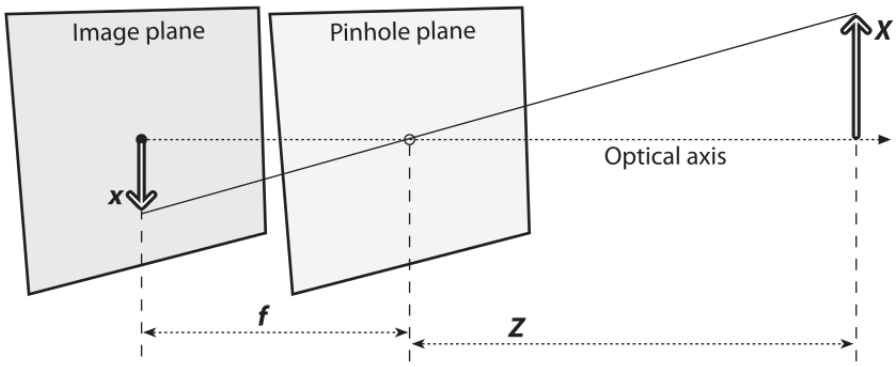


Figure 8: Rappresenzazione di un pinhole camera model

In tale figura *f è la* focal lenght della telecamera, *Z* la distanza tra oggetto e telecamera, *X la dimensione dell'oggetto e* x l'immagine dell'oggetto proiettata sul piano.
Il modello appena descritto può essere reinterpretato come in figura 9

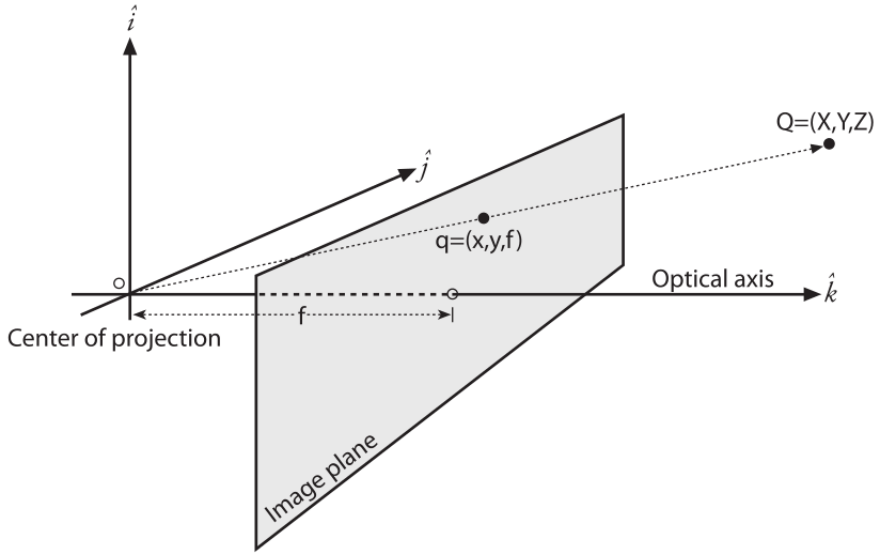


Figure 9: Rappresenzazione di un pinhole camera model rovesciato

nella quale sono stati rovesciati il punto di *pinhole* e l'*image plane*. Ciò ci permette di semplificare le relazioni che valgono tra i punti dell'oggetto e le loro proiezioni. Infatti ora è più chiara la similarità dei triangoli, inoltre l'immagine non è più capovolta come era in figura 8. In tale visione vanno aggiunti altri due nuovi parametri c_x *e* c_y per considerare un eventuale *displacement* del centro delle coordinate sull'*image plane*, dovuto alla possibilità che il centro del chip (della telecamera) non risieda esattamente sopra l'asse indicato in figura 9 come *optical axis*.

In tale visualizzazione c'è una semplice relazione che mappa i punti Q_i *nello spazio (coordinate (*X_i, Y_i, Z_i)) nei punti sul piano di proiezione (coordinate (x_i,y_i)), tale relazione è chiamata *projective transform*. Viene riportata in figura 10

$$q = MQ, \quad \text{where} \quad q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Figure 10: Espressione matematica di una relazione projective transform

Tale modello è utile per capire la meccanica della geometria tridimensionale di visione, ma non è utilizzato nella realtà perche presenta dei limiti. Esso infatti in pratica porterebbe a una frequenza di generazione dei frame molto lenta, dovuta alla bassa quantità di luce che riesce a passare attraverso un *pinhole*. Per ottenere una maggiore quantità di luce e quindi un maggior rapporto di frame/secondo è necessario utilizzare una lente, che consente di raccogliere molta più luce e di curvarla in maniera che converga nel punto di projection. Lo svantaggio di utilizzare una lente è che viene introdotta la *distortion*.

5.1.2 Distortion parameters

La *lens distortion* si può dividere in due principali tipi:

1. *radial distortion*
2. *tangential distortion*

La prima è un risultato della forma fisica della lente. Le lenti infatti tendono a distorcere i pixel vicini ai bordi dei frame, tale fenomeno è chiamato "barrel effect". Viene riportata in figura 11 uno schema che aiuta l'intuizione del motivo per cui tale fenomeno si verifica.

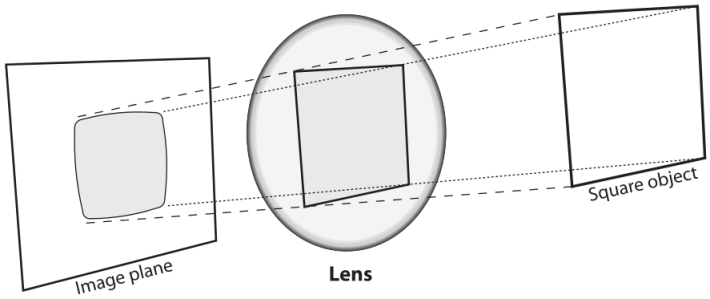


Figure 11: Radial distortion effect

La seconda è un risultato dei difetti di assemblaggio che portano la lente a non essere perfettamente parallela all'*image plane*. Si riporta un immagine che evidenzia tale fenomeno in figura 12.

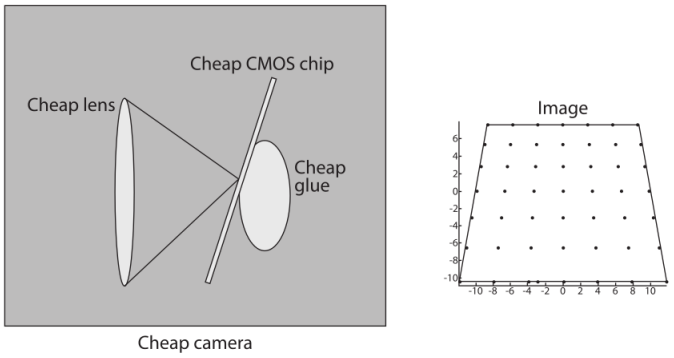


Figure 12: Tangential distortion effect

5.1.3 Calibrazione

Ora che è stato descritto come identificare a livello matematico le proprietà *intrinsics* e *distortion* possiamo vedere le tecniche con le quali si prendono in considerazione tali parametri per correggere i frame catturati dalla telecamera.
Le tecniche di calibrazione si basano sul seguente criterio: viene esposto mostrato alla

camera un oggetto di cui si conosce la struttura, e si valutano le differenze tra le caratteristiche dell'oggetto catturato dalla camera (quindi proiettato sull'*image plane*) con le sue note caratteristiche fisiche.
Per ogni immagine che una telecamera cattura di un oggetto, è possibile descrivere la sua posizione rispetto al sistema di coordinate della telecamera in termini di rotazione e traslazione. Possiamo quindi definire una relazione tra la posizione nello spazio di un oggetto e le sue coordinate nel frame catturato dalla telecamera. Combinando tale relazione con le correzioni derivanti dai parametri di *intrinsics* si ottiene un sistema di equazioni che risolto rappresenta la soluzione al problema della calibrazione.
In linea di principio qualsiasi oggetto può essere usato ai fini di tale operazione, ma è pratico utilizzare un oggetto che presenta caratteristiche regolari (pattern). All'interno del progetto verrà utilizzata una immagine di una scacchiera con dimensioni (numero di celle) predefinite.
Si potrà vedere come OpenCV offre un API di alto livello per la soluzione del problema della calibrazione con l'utilizzo di una scacchiera.

5.2 Tracking data transformation

5.2.1 Homography matrix

5.2.2 Qt coordinate system

5.3 Qt gradients

References

[1] Pathflow website `http://pathflow.co`

[2] OpenCV website `http://opencv.org/`

[3] Gary Bradski, Adrian Kaehler *Learning OpenCV* 2008.

[4] Robert Laganière *OpenCV 2 Computer Vision Application Programming Cookbook* 2011.

[5] Qt website `http://qt-project.org/`

[6] MySQL website `http://www.mysql.com/`

[7] MySQL wrapped website `http://www.alhem.net/project/mysql/`

[8] AutoCAD *DXF Reference* `http://images.autodesk.com/adsk/files/acad_dxf0.pdf`