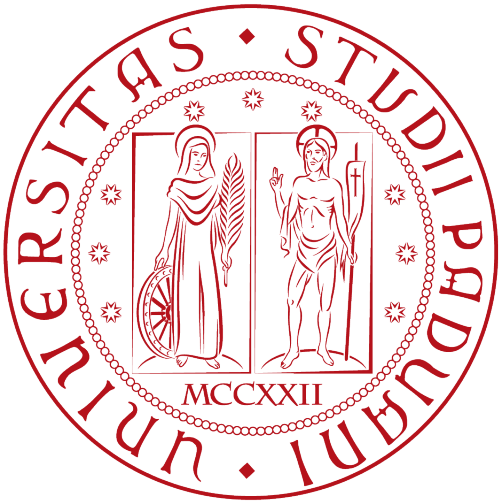
Università degli Studi di Padova
Dipartimento di Matematica
CORSO DI LAUREA IN INFORMATICA



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 gli standard dello sviluppo software.

1.2 Pathflow

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 [1]

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 mappatura 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.

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.

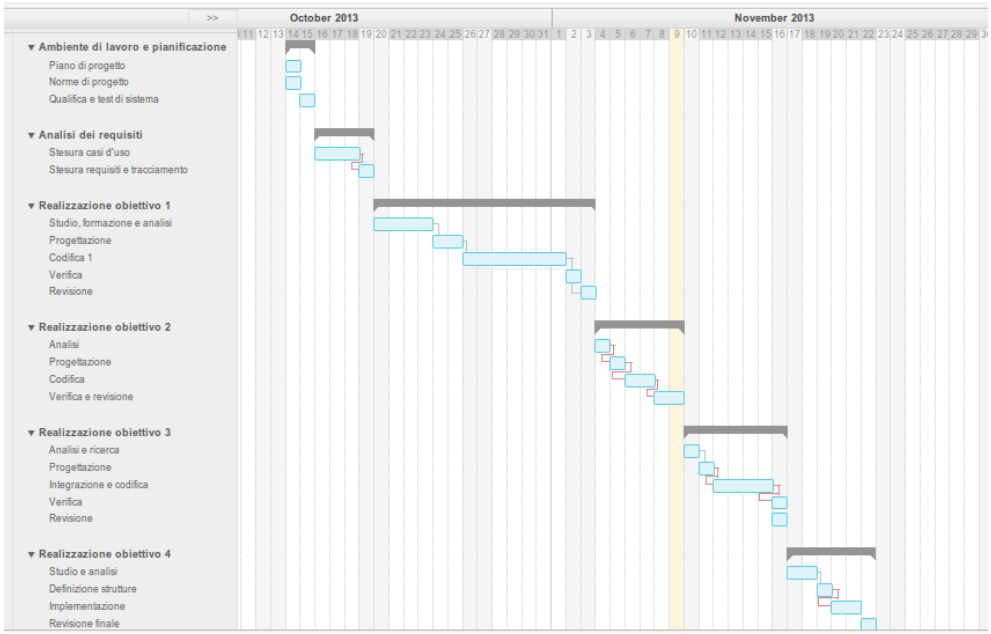


Figure 1: diagramma di Gantt che descrive la pianificazione

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. PRECONDIZIONI: asserzioni che sono valide prima dell'effettiva esecuzione del caso d'uso
3. POSTCONDIZIONI: asserzioni che sono valide dopo l'esecuzione del caso d'uso
4. 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.

4.2 UC0: Scenario Principale

descrizione L'utente può calibrare le telecamere, configurarle, oppure generare le statistiche a partire dai dati di tracking (figura 2)

precondizione Il sistema è avviato e funzionante

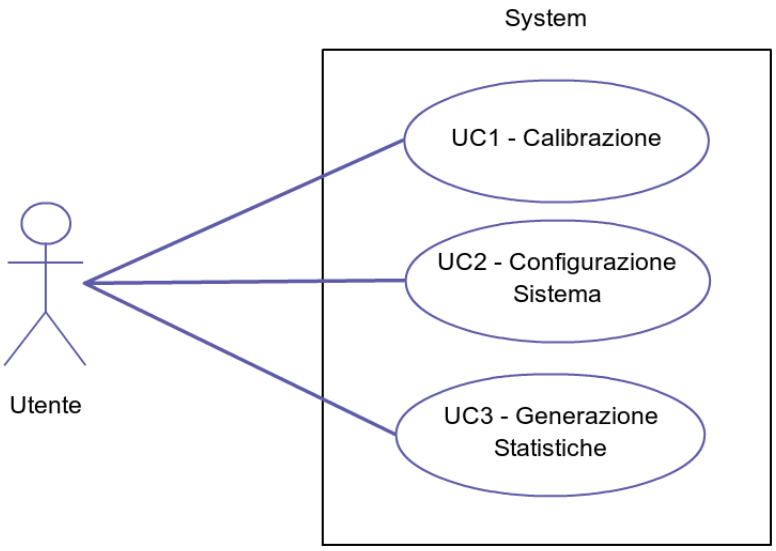


Figure 2: UC0 - Scenario principale

4.3 UC1: Calibrazione telecamere

descrizione L'utente può calibrare le telecamere, o di visualizzare un video *undistorted* che usa i parametri di calibrazione per correggere i frame. (figura 3)

precondizione Il sistema è avviato e funzionante

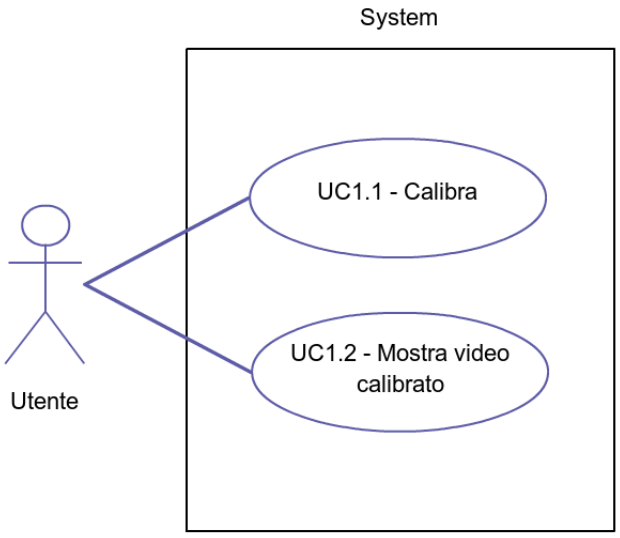


Figure 3: UC1 - Calibrazione telecamere

4.3.1 UC1.1: Calibra

descrizione L'utente seleziona le opzioni per la calibrazione quali: numero di span spot, numero di frame da scartare, indirizzo della camera da calibrare, e poi inizia l'effettiva calibrazione. (figura 4)

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

scenari alternativi L'utente interrompe la procedura, il sistema ritorna in attesa di eventi.
La procedura fallisce, il sistema segnala l'errore e torna in attesa di eventi

UC1.1.1: Seleziona il numero di span spot

descrizione L'utente seleziona il numero di span spot per la calibrazione

postcondizione Il sistema conosce il numero di span spot da utilizzare nella calibrazione

scenari alternativi L'utente interrompe la procedura, il sistema ritorna in attesa di eventi

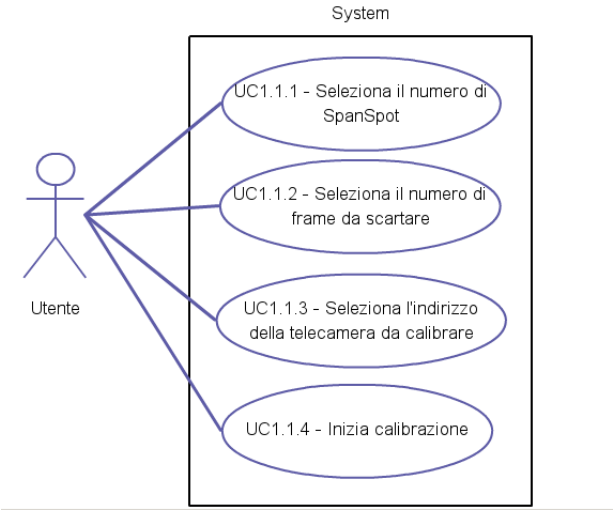


Figure 4: UC1.1 - Calibra

UC1.1.2: Seleziona il numero di frame da scartare

descrizione L'utente seleziona il numero di frame da scartare per la calibrazione

precondizione L'utente ha già selezionato il numero di span spot

postcondizione Il sistema conosce il numero di frame da scartare nella calibrazione

scenari alternativi L'utente interrompe la procedura, il sistema ritorna in attesa di eventi

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 di span spot e il numero di frame da scartare nella calibrazione

postcondizione Il sistema conosce l'indirizzo della telecamera da calibrare

scenari alternativi L'utente interrompe la procedura, il sistema ritorna in attesa di eventi

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 ed extrinsics e li salva nel file system

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

4.4 UC2: Configurazione telecamere

descrizione L'utente può impostare le opzioni di configurazione delle telecamere, aggiungendone di nuove, rimuovendole, modificandole, salvare un frame della telecamera per l'uso futuro, convertire un file .DXF in .PNG o calcolare la *homography matrix* (figura 5)

precondizione Il sistema è avviato e funzionante.

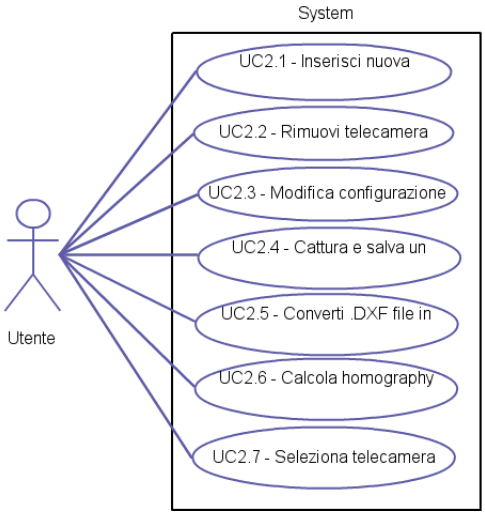


Figure 5: UC2 - Configurazione telecamere

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

4.4.2 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

4.4.3 UC2.3: Modifica configurazione

descrizione L'utente modifica le opzioni di configurazione della telecamera selezionata, quali: file extrinsics, intrinsics, valore dell'altezza del frame, valore della larghezza del frame, percorso del file contentente la *homography matrix* (figura 6).

precondizione L'utente ha selezionato una telecamera tra quelle esistenti

scenari alternativi L'utente interrompe la modifica, il sistema ritorna in attesa di eventi

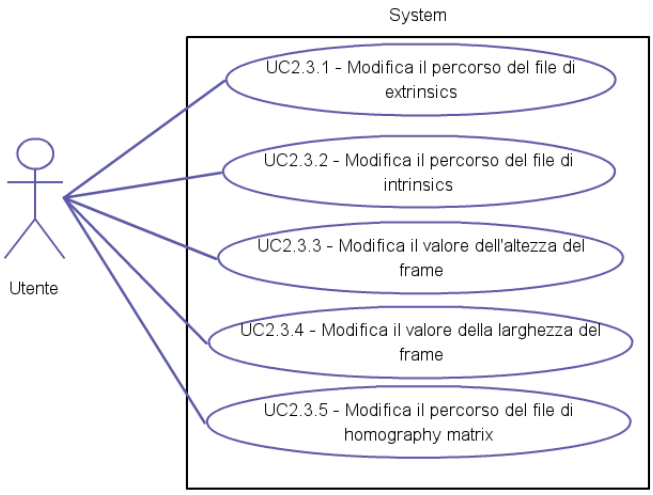


Figure 6: UC2.3 - Modifica telecamere

UC2.3.1: Modifica il percorso del file di extrinsics

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

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

scenari alternativi L'utente interrompe la selezione, il sistema ritorna in attesa di eventi

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 L'utente interrompe la selezione, il sistema ritorna in attesa di eventi

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

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

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 L'utente interrompe la selezione, il sistema ritorna in attesa di eventi

4.5 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)

precondizione Il sistema è avviato e funzionante.

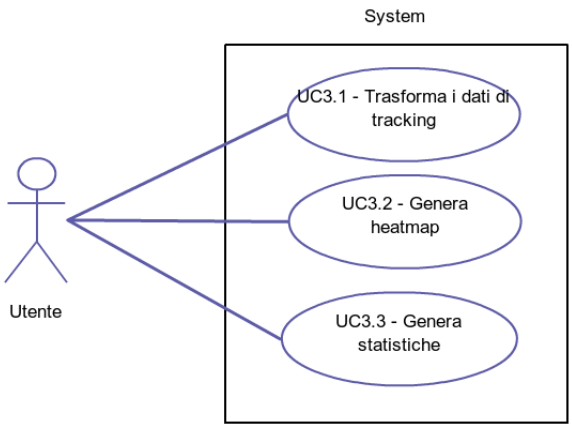


Figure 7: UC2 - Generazione statistiche

4.5.1 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.

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

4.6 Requisiti

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