Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Nuovo algoritmo fotogrammetria - Azienda esterna REX SA

Studente/i

Terzi Edoardo

Relatore

Mattei Yari

Correlatore

-

Committente

REX SA

Corso di laurea

Ingegneria informatica (TP)

Modulo

C10051 Progetto di diploma

Anno

2019

Data

19 luglio 2019

Indice

Capitolo 1

Abstract

Questo progetto nasce dalla necessità dell'azienda REX di Mendrisio di creare un sistema per la progettazione di una passerella posta fra i binari di un passo ferroviario. L'azienda in questine si occupa della produzione, della progettazione e della vendita di molteplici articoli tecnici tra i quali le passerelle.

Inizialmente il sistema, costituito da una piattaforma WEB, si occupava di ricevere in input un file DXF realizzato da un geometra dell'azienda. Questo file doveva contenere le rilevazioni del geometra, il calcolo delle griglie che costiutiscono la passerella e il disegno di queste. Infine la piattaforma generava un documento PDF contenente le istruzioni e il numero di pezzi per il montaggio della passerella. Inoltre era possibile gestire i clieniti e gli ordini effettuati mediante la piattaforma.

Attualmente, in seguito a un progetto di semestre svloto da due studenti della SUPSI, la piattaforma è migliorata mantenendo sempre l'aspetto relativo alla gestione degli ordini, dei clienti e del PDF ma è stato alleggerito il lavoro del geometra. Quest'ultimo ora deve inserire nel file DXF solamente le misure effettuate e non deve calcolare a mano le griglie necessarie alla costruzione della passerella.

L'obiettivo di questo progetto di diploma è quello di rendere ancora più semplice ed automatico l'utilizzo dell'intero sistema mediante l'utilizzo di tecniche di processamento dell'immagine. In particolare si vuole realizzare un algoritmo che sia in grado di rilevare le misure necessarie alla costruzione delle griglie che compongono la passerella partendo da una o più foto della scena, quindi del passo ferroviario dove si vuole fare il lavoro di installazione.

Capitolo 2

Progetto

2.1 Descrizione

Rex SA è una società manifatturiera: uno dei molti prodotti commercializzati con il proprio marchio è relativo alla produzione di passaggi di servizio ferroviari in resina per l'attraversamento nelle stazioni ferroviarie e per uscite di sicurezza nelle gallerie di treno o metropolitana, articoli commercializzati con il marchio SWISSCROSS GFK. La sfida del progetto risiede nell'implementazione di un sistema capace di creare e visualizzare in modalità aumentata (sopra una fotografia) un modello virtuale di un passaggio GFK partendo da un'immagine e attivando automaticamente i processi di produzione, consegna, installazione e, in seguito, di manutenzione del prodotto. Grazie alla soluzione proposta, REX sarà in grado rafforzare la propria posizione sul mercato svizzero e modificare il proprio modello di business per iniziare una vendita di servizi digitali a livello internazionale. Con il progetto si vuole incrementare la competitività dell'azienda delegando al cliente la progettazione e la visione virtuale del risultato finale in modalità "self-service" prima ancora di confermare l'ordine tramite la piattaforma cloud, automatizzando quindi ordinazione, produzione e delivery in un vero e proprio processo di Industria 4.0.

2.2 Obiettivi

L'obiettivo principale è quello di sviluppare un algoritmo in grado di sfruttare le metriche conosciute (scarto binari e modelli 3D di binari, traversine e viterie) per ricavare una matrice omografica tramite la quale calcolare il modello completo di una passerella sulla base di una immagine. Allineamento e sovrapposizione dei modelli generati da più immagini verranno realizzati sfruttando appositi marcatori posizionati sul terreno. Sintetizzando l'obiettivo di questo algoritmo è quello di permettere l'estrazione dei punti caratteristici da un immagine (features) necessari al calcolo della forma e delle misure di una passerella GFK.

2.3 Compiti

Il compito principale risiede nella creazione quindi di un algoritmo in grado di elaborare un immagine fotografica e ricavarne la relativa matrice omografa. Tramite questa matrice si potrà "raddrizzare" l'immmagine e quindi calcolare le dimensioni reali del passaggio pedonale che dovrà quindi essere visualizzato in modalità "realtà aumentata" sopra la fotografia.

Capitolo 3

Introduzione

3.1 Premessa

3.2 Contesto

3.3 Scopo

Capitolo 4

Pianificazione

4.1 Approccio iniziale

4.2 Svlogimento del lavoro

4.3 Diagramma di Gantt

Capitolo 5

Analisi

5.1 Requisiti

5.2 Architettura

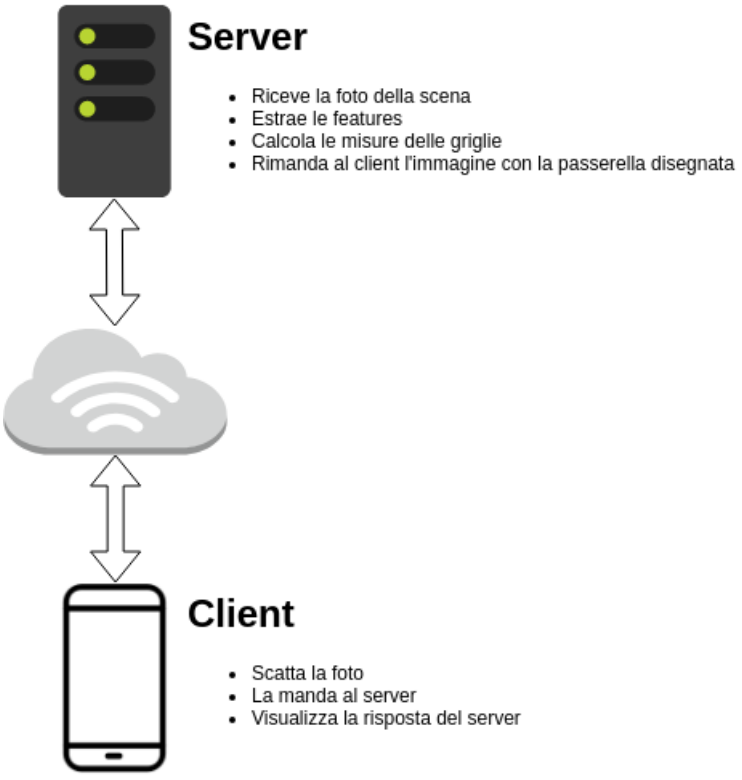

Figura 5.1: Architettura client-server

5.3 Tecnologie utilizzate

5.3.1 Hardware

5.3.1.1 PC per lo sviluppo

Portatile con Windows 10 Pro 64-bit:

Figura 5.2: Dell XPS 9570

- Intel Core i7-8750H CPU 2.20GHz x 12
- 16 GB 2 x 8 GB DDR4 a 2.666 MHz
- PCIe M.2 2280 da 512 GB
- NVIDIA GeForce GTX 1050 Ti con 4 GB di GDDR5

5.3.1.2 Smartphone per l'acquisizione delle immagini

Smartphone Android con fotocamera tripla Leica:

Figura 5.3: Huawei P30

- 40 MP (obiettivo normale, apertura f/1.8)
- 16 MP (obiettivo grandangolare, apertura f/2.2)
- 8 MP (teleobiettivo, apertura f/2.4, OIS)

5.3.2 Software

5.3.2.1 Ambienti di sviluppo, editor

5.3.2.2 Elaborazione dell'immagine

Capitolo 6

Diagrammi

6.1 Casi d'uso

6.2 Sequenza

Capitolo 7

Implementazione

7.1 Linguaggi, librerie, framework

7.1.1 Java

Java è un linguaggio di programmazione ad alto livello, orientato agli oggetti e a tipizzazione statica, che si appoggia sull'omonima piattaforma software, specificamente progettato per essere il più possibile indipendente dalla piattaforma hardware di esecuzione (tramite compilazione in bytecode prima e interpretazione poi da parte di una JVM), sebbene questa caratteristica comporti prestazioni in termini di computazione inferiori a quelle di linguaggi direttamente compilati come C e C++ ovvero dunque perfettamente adattati alla piattaforma hardware. Il back-end dell'intera piattaforma è stato sviluppato mediante l'uso di questo linguaggio e del framework Spring MVC.

7.1.2 Python

È un linguaggio multi-paradigma che ha tra i principali obiettivi dinamicità, semplicità e flessibilità. Supporta il paradigma object oriented, la programmazione strutturata e molte caratteristiche di programmazione funzionale e riflessione. Le caratteristiche più immediatamente riconoscibili di Python sono le variabili non tipizzate e l'uso dell'indentazione per la definizione delle specifiche. Altre caratteristiche distintive sono l'overloading di operatori e funzioni tramite delegation, la presenza di un ricco assortimento di tipi e funzioni di base e librerie standard, sintassi avanzate quali slicing e list comprehension. Abbiamo scelto di utilizzare questo linguaggio in quanto la nota libreria OpenCV, descritta in seguito, è completamente supportata e documentata. Difatti gli script in python si occupano principalmente del processamento delle immagini.

7.1.3 OpenCV

7.2 Feature detection

In computer vision, e nell'elaborazione digitale delle immagini, il concetto di rilevamento di caratteristiche (feature detection) o riconoscimento di caratteristiche racchiude una serie di metodi per l'estrapolazione di informazioni da una immagine e per prendere decisioni locali sull'esistenza o meno di una caratteristica in quel determinato punto. Le caratteristiche risultanti saranno un sottoinsieme del dominio dell'immagine, spesso in forma di punti isolati, curve continue o regioni connesse. Non esiste una definizione universale ed esatta di cosa costituisca una caratteristica dell'immagine (image feature), e la definizione esatta spesso dipende dal problema o dal tipo di applicazione. Le caratteristiche sono usate spesso come punto di partenza da molti algoritmi di computer vision. Una proprietà desiderabile per un rilevatore di caratteristiche è la ripetibilità: la stessa caratteristica dovrebbe essere rilevata in due o più differenti immagini della stessa scena. È un'operazione di elaborazione delle immagini di basso livello, che ed esamina ogni pixel. È la prima operazione che si fa su un'immagine. Se invece fa parte di un algoritmo, allora di solito esamina solamente la regione individuata.

7.2.0.1 Trasformata di Hough

In computer vision esistono innumerevoli tecniche per l'estrazione delle features. Una di queste è la trasformata di Hough, utilizzata per individuare linee all'interno di una immagine. Ideata nella sua forma base agli inizi degli anni '60 da Paul Hough, fu in seguito perfezionata in una forma generalizzata e resa popolare nel 1981 da Dana H. Ballard, docente di informatica dell'università del Texas. La trasformata di Hough necessita di un passo di preprocessing finalizzato all'Edge Detection (es Canny Edge Detection). L'immagine così ottenuta sarà una immagine in bianco e nero dove i pixel bianchi rappresentano gli Edge, ovvero i contorni dei soggetti rappresentati. L'obiettivo della trasformata di Hough è di identificare quali di questi contorni rappresentino una linea retta: se più pixel di Edge sono allineati, anche se non connessi tra loro, viene individuata una linea nell'immagine. Per rappresentare una linea retta servono due parametri: un coefficiente angolare m e una ordinata all'origine c. Considerato un singolo punto, questo appartiene a infinite rette della forma $y = mx + q$, al variare di m e q. Dobbiamo quindi stabilire quali coppie (m, q) siano in grado di individuare al meglio gli allineamenti di pixel. L'idea della trasformata di Hough è di rappresentare le rette nello spazio parametrico $m - q$. In questo modo preso un punto (x, y) l'equazione $y = mx + q$ andrà a rappresentare una sola retta invece di infinite. Nello spazio parametrico vengono quindi rappresentate tante rette quanti sono i pixel di edge. Se queste rette si intersecano, i due punti corrispondenti dell'immagine risulteranno allineati sulla retta con i coefficienti (m, q) corrispondenti. La tecnica sembra quindi funzionare: le

rette corrispondenti ad un alto numero di intersezioni nello spazio parametrico sono le linee che volevamo ottenere.

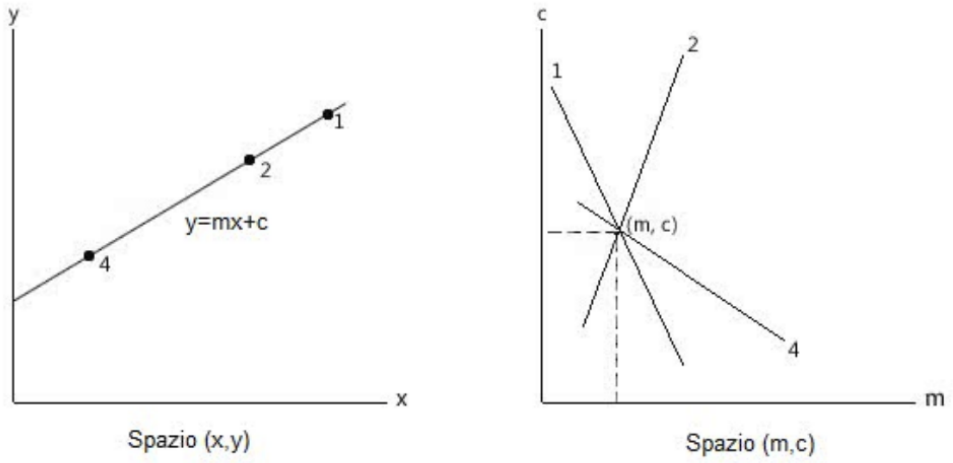

Figura 7.1: I parametri m e q vengono mappati nello spazio parametrico e la linea è individuata dall'intersezione delle rette.

La trasformata però in questa forma presenta dei problemi che non la rendono applicabile su un calcolatore. Infatti per linee verticali il coefficiente angolare m tende all'infinito, contrastando le limitazioni di memoria di ogni computer. Si ricorre quindi ad una versione detta forma normale dove le rette vengono rappresentate mediante la forma $xcos\theta + ysin\theta = r$. I parametri θ e r sono rispettivamente l'angolo e la distanza della retta dall'origine. Quantizzando θ e utilizzando come spazio parametrico $\theta - r$ si risolve quindi il problema della finitezza della memoria. All'interno della libreria OpenCV la traformata di Hough si ottiene tramite la funzione:

```
def HoughLinesP(image, rho, theta, threshold, lines=None,
                minLineLength=None, maxLineGap=None):
```

- **image:** puntatore all'immagine degli edge ottenuta tramite un algoritmo di Edge Detection.
- **rho:** spessore in pixel delle linee rappresentate nello spazio parametrico. Viene utilizzato per accorpare tra loro intersezioni vicine.
- **threshold:** soglia sul numero di intersezioni che si devono ottenere affinchè una linea sia identificata come tale.
- **lines:** struttura dove la funzione scrive le linee, tipicamente sotto forma di CvMemoryStorage.

- **minLineLength:** rappresenta la lunghezza minima di una linea.
- **maxLineGap:** rappresenta a distanza tra punti sulla stessa retta, per spezzarla in due linee differenti.

7.2.0.2 Binari

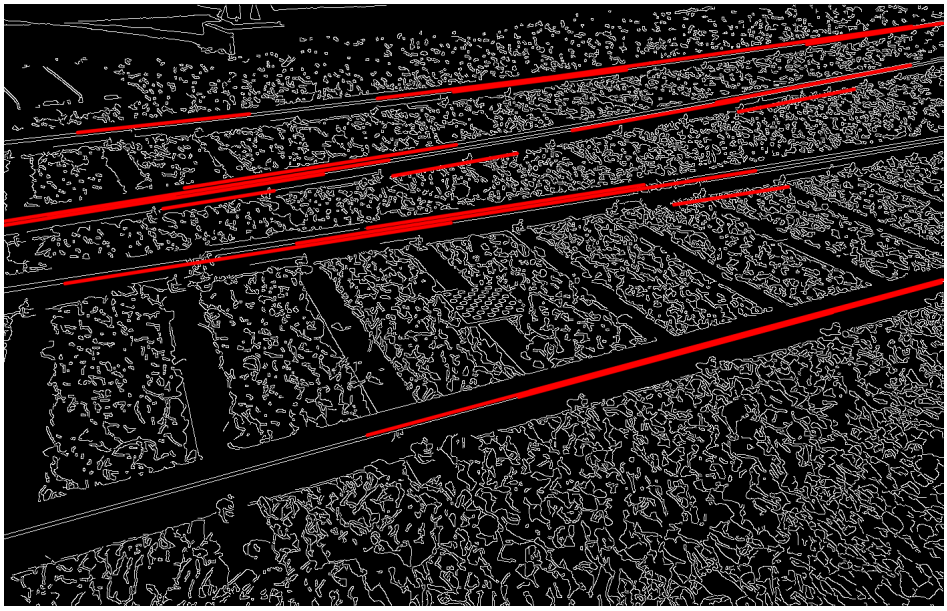

Figura 7.2: Ricerca dei binari mediante la trasformata di Hough

Come si evince dall'immagine sopra, mediante la trasformata di Hough è possibile identificare i binari all'interno di una foto. Al fine di identificare solo i binari e non altre linee rette è necessario settare correttamente i parametri, in particolar modo il valore di threshold e la lunghezza minima delle rette. Per torvare il valore ottimale di threshold abbiamo usato la soglia OTSU calcolata sfruttanto la varianza tra classi di pixel (l'algoritmo che calcola questa soglia è implementato nativamente nella libreria OpenCV). Invece per determinare la lunghezza minima delle linee abbiamo effettuato un calcolo assumendo che un binario sia lungo quanto la larghezza o lunghezza della foto. Tuttavia, si può notare come un binario può essere identificato da più rette. Inoltre, nonostante il settaggio corretto dei parametri è comunque possibile che vengano rilevate linee che non centrano nulla con i binari. L'approccio scelto per risolvere questi due problemi è di tipo matematico. Per identificare un binario con una sola abbiamo deciso di raggupparle sfruttando l'area sottesa alla retta trovata, ovvero l'integrale, assumendo che due rette che rappresentano lo stesso binario hanno un'area circa uguale. Infine per identificare solo i binari e non altre linee rette abbiamo rimosso tutte le rette che si intersecano all'interno dello spazio definito dalla foto e per maggiore sicurezza l'immagine è anche stata sfocata mediante un filtro gaussiano al fine di far rimanere in evidenza solo le linee dei binari.

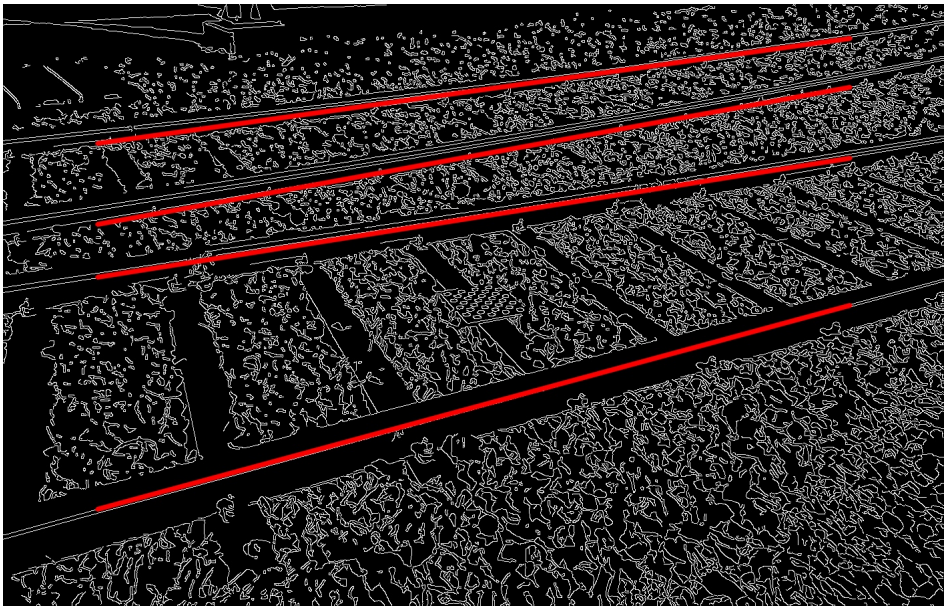

Figura 7.3: Risultato filtraggi matematici

Il risultato ottenuto sembra promettere bene tuttavia, sfruttando questa tecnica si rischia di non fare distinzione tra il lato interno o esterno del binario. Inoltre non è possibile determinare se la linea identifica il binari sullo stesso piano. Come soluzione abbiamo deciso di sfruttare i bulloni delle traversine messe in risalto con del nastro rosso. Posizionandone anche solo quattro è possibile identificare correttamente i binari.

7.2.0.3 Punti

Figura 7.4: Rilevamento delle palline da tennis e dei nastri rossi

Per il rilevamento dei punti abbiamo sfruttato due diversi approci a seconda del tipo di punto da rilevare. Nonostante questo il punto di partenza delle tecniche è il medesimo: il colore. Abbiamo effettuato un filtraggio sul colore in modo da avere il resto dell'immagine nera e solo i punti di nostro interesse in risalto.

Figura 7.5: Filtraggio sul colore del nastro rosso

Nell'immagine sopra sono rimasti solo i nastri rossi, in questo modo ci è stato possibile ricercare i contorni delle regioni di colore rosso e identificare così i 4 bulloni. Nel caso delle palline da tennis abbiamo risfruttato l'algoritmo di Hough mediante la seguente funzione:

```
def HoughCircles(image, method, dp, minDist, circles=None,
                 param1=None, param2=None, minRadius=None,
                 maxRadius=None):
```

- **image:**
- **method:**
- **dp:**
- **minDist:**
- **circles:**
- **param1:**
- **param2:**
- **minRadius:**
- **maxRadius:**

7.3 Calibratore

7.4 Matrice Omografica

7.5 Interfacciamento sistema pre esistente

7.6 Gestione errori

7.7 Testing

Capitolo 8

Conclusioni

8.1 Problematiche

8.2 Sviluppi futuri

8.3 Considerazioni finali

Bibliografia

[1] C. Yuen and A. Oudalov. The feasibility and profitability of ancillary services provision from multi-microgrids. In *Power Tech, 2007 IEEE Lausanne*, pages 598 –603, july 2007.

[2] H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 8(1):18 –28, january-february 2010.

[3] L.H. Tsoukalas and R. Gao. From smart grids to an energy internet: Assumptions, architectures and requirements. In *Electric Utility Deregulation and Restructuring and Power Technologies, 2008. DRPT 2008. Third International Conference on*, pages 94 –98, april 2008.

[4] F. Blaabjerg, R. Teodorescu, M. Liserre, and A.V. Timbus. Overview of control and grid synchronization for distributed power generation systems. *Industrial Electronics, IEEE Transactions on*, 53(5):1398 –1409, oct. 2006.