



UNIVERSITÀ DEGLI STUDI DI FIRENZE
SCUOLA DI INGEGNERIA - DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Tesi di Laurea Triennale in Ingegneria Informatica

**TECNICHE DI RICONOSCIMENTO E LORO
APPLICAZIONE AL CONTROLLO DI SISTEMI
MOBILI MEDIANTE PIATTAFORME EMBEDDED**

Candidato
Luca Angioloni

Relatore
Prof. Fabrizio Argenti

Anno Accademico 2015-2016

Ringraziamenti

Questo progetto di tesi è stato possibile grazie alla disponibilità del Professor Fabrizio Argenti, relatore del progetto di tesi, a cui va la mia gratitudine per avermi concesso la sua fiducia e per i consigli ricevuti.

Desidero ringraziare i miei genitori e la mia famiglia, per avermi permesso di continuare i miei studi con il loro supporto morale e materiale.

Grazie alla mia ragazza per il sostegno ricevuto e ai miei amici e compagni di studi (Andrei) per il loro aiuto e per le avventure di questi tre anni.

Indice

Introduzione	iv
1 Metodi di riconoscimento visivo	1
1.1 Rilevamento e descrizione di punti di interesse	2
1.1.1 SURF	3
1.1.2 Metodi alternativi	12
1.2 Keypoint Matching	14
1.2.1 Brute Force Matcher	15
1.3 Omografia	15
2 Implementazione mediante sistemi embedded e applicazione al controllo di posizione	17
2.1 Sistemi embedded	17
2.1.1 RaspberryPi	18
2.1.2 Arduino	21
2.2 Costruzione	23
2.3 Riconoscimento visivo	25
2.4 Interfaccia di collegamento con Arduino	27
2.5 Controllo Motori	29
2.6 Controllo Remoto	30

<i>Indice</i>	iii
2.6.1 Mobile App	31
3 Risultati sperimentali	35
3.1 Frame rate e risoluzione	36
3.2 Velocità di risposta	36
3.3 Affidabilità e precisione	37
3.4 Risultati per approcci alternativi	37
3.5 Test eseguiti	39
Conclusioni	40
A Tecnologie utilizzate	44
A.1 Python	44
A.2 OpenCV	45
A.3 TCP Socket	47
A.4 iOS - Swift	48
A.5 JSON	49
A.6 Comunicazione Seriale	51
A.7 Multicast DNS	52
Bibliografia	54

Introduzione

Il progetto di tesi ha come obiettivo la costruzione di un sistema autonomo mobile in grado di riconoscere ed inseguire oggetti arbitrari nello spazio, avvalendosi delle tecnologie embedded attualmente disponibili e di tecniche di riconoscimento visivo all'avanguardia.

Il sistema deve essere di dimensioni contenute, dai bassi consumi e dai costi realizzativi ridotti. Le piccole dimensioni permettono al sistema di essere autonomo ed integrabile in progetti anche più complessi rendendolo allo stesso tempo trasportabile e facilmente utilizzabile. Una lunga autonomia è fondamentale ai fini di un utilizzo intensivo e prolungato, infatti, l'attenzione rivolta ai consumi energetici è di primaria importanza. Infine i costi ridotti rendono il sistema una piattaforma accessibile ad un utenza più ampia rispetto a quella raggiunta dai sistemi disponibili in commercio dai costi elevati.

Il progetto ha comportato lo studio di diverse tecniche di riconoscimento visivo e di piattaforme embedded che grazie a tecnologie differenti hanno potuto operare in accordo.

Dall'analisi di una singola immagine di input, rappresentante un oggetto, il sistema è in grado di estrarre le informazioni necessarie all'elaborazione di immagini continue catturate dalla fotocamera integrata. Con i risultati dell'elaborazione il sistema riesce a stimare la posizione dell'oggetto nello

spazio circostante e ad operare gli attuatori di cui è fornita per inseguirlo e raggiungerlo.

La struttura della tesi è la seguente:

Nel capitolo 1, si introducono e descrivono i metodi di *riconoscimento visivo* utilizzati.

Nel capitolo 2, si descrivono i passi principali necessari allo sviluppo e costruzione del progetto partendo dai metodi di riconoscimento visivo fino all'attuazione di motori. Si introduce anche lo sviluppo dell' App mobile per il controllo remoto.

Nel capitolo 3, si riportano i risultati ottenuti con l'esecuzione di test sperimentali.

Infine nel capitolo conclusivo, si riportano alcune considerazioni e discute brevemente dei possibili sviluppi del progetto e di una direzione di ricerca per ottimizzare il sistema.

In Appendice sono riportate le principali tecnologie utilizzate nella realizzazione del progetto di Tesi.

Capitolo 1

Metodi di riconoscimento visivo

L'uomo utilizza i suoi occhi ed il suo cervello per vedere e percepire visivamente il mondo intorno a lui. La *Computer Vision* (*Visione Computazionale*) è la scienza che si pone come obiettivo di dare capacità simili, se non superiori, a macchine e computer. La visione computazionale, in automatico, si occupa dell'estrazione, analisi e comprensione di informazioni da una singola immagine o una loro sequenza. Comporta lo sviluppo di basi teoriche ed algoritmiche per ottenere una comprensione visuale.[1]

Le applicazioni di questa scienza sono molteplici e spaziano in molti campi, ad esempio la medicina, la biologia, i sistemi di supporto per ipovedenti, i controlli di qualità industriali, la sicurezza e sorveglianza, la realtà aumentata, l'interazione naturale, la robotica e i veicoli autonomi. Si tratta di tecnologie all'avanguardia e di frontiera, su cui il mondo della ricerca, sempre di più, investe gran parte del suo interesse.

Fanno parte di questa scienza i metodi di *riconoscimento visivo* presentati in questo elaborato.

1.1 Rilevamento e descrizione di punti di interesse

Il compito di trovare corrispondenze tra due immagini della stessa scena o oggetto (Riconoscimento visivo), è parte di molte applicazioni di *Computer Vision*. Calibrazioni di videocamere, ricostruzioni 3D e riconoscimento di oggetti, sono solo alcune.

Il concetto di rilevamento o riconoscimento di punti di interesse (*feature detection*), racchiude una serie di metodi per l'estrapolazione di informazioni da una immagine utili a prendere decisioni locali, sull'esistenza o meno di caratteristiche interessanti, grazie alle quali poter svolgere la ricerca di corrispondenze o successive elaborazioni. Eventuali caratteristiche di interesse risulteranno essere 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.[2]

La ricerca di corrispondenze discrete in immagini può essere divisa in tre passi principali. Inizialmente, sono selezionati dei "punti di interesse" (*Keypoints*) in posizioni distintive e particolari dell'immagine, ad esempio angoli, bordi, *blobs*, *T-junctions*. La proprietà più importante di un rilevatore di punti di interesse è la sua ripetibilità, quindi, la capacità di trovare in maniera affidabile gli stessi punti anche in diverse condizioni di vista.

Successivamente, il punto di interesse e le sue vicinanze sono rappresentate da un vettore di caratteristiche (*Features*). Questo "descrittore" deve essere distintivo e, allo stesso tempo, robusto e resistente al rumore, agli errori di rilevamento e alle deformazioni geometriche e fotometriche.

Infine, i vettori di descrizione sono confrontati tra le differenti immagini. Il confronto (*Matching*) è spesso basato su una distanza tra i vettori, ad esempio *Mahanalobis* o la *distanza Euclidea*. Grazie a questo confronto è possibile individuare delle corrispondenze tra i punti di interesse di più immagini che possono essere poi utilizzate nelle varie applicazioni (vedi 1.2).[3]

Queste tecniche di riconoscimento visivo, utilizzate per il tracciamento e rilevamento di oggetti nello spazio, sono alla base di questo progetto di tesi per il quale sono stati analizzati vari approcci e algoritmi di rilevamento e descrizione di punti di interesse (*Feature Extraction: Feature detection and description*).

Con le corrispondenze trovate si stima la posizione dell'oggetto cercato, utilizzando una tecnica di proiezione chiamata *Omografia*.

1.1.1 SURF

Nel 2006 Bay H., Tuytelaars T. e Van Gool L. pubblicarono un articolo, “SURF: Speeded Up Robust Features”, che introduceva un nuovo algoritmo chiamato SURF.[4]

Questo algoritmo permette il rilevamento e la descrizione di punti di interesse in maniera indipendente dalla rotazione e dalla scala. SURF approssima o addirittura ottiene migliori risultati di schemi precedentemente proposti in merito a *ripetibilità*, *distintività* e *robustezza* essendo, però, molto più rapido nel calcolo e nel confronto.

Rilevamento dei punti di interesse

L'approccio per il rilevamento di punti di interesse utilizza una approssimazione semplice della matrice Hessiana¹.

Si predisponde inizialmente l'immagine di input all'elaborazione con l'utilizzo di "*immagini integrali*" (*Integral Images o Summed Area Table*) rese famose da Viola e Jones in [5], che riducono drasticamente i tempi computazionali.[6]

Integral Images

Le *Integral Images* permettono un calcolo più veloce di filtri convoluzionali discreti (*Convolutional Box Filters*)². Per crearne una è necessario costruire la tabella *Summed Area Table* nella quale l'elemento $I_{\Sigma}(\mathbf{x})$ in posizione $\mathbf{x} = (x, y)^T$ rappresenta la somma di tutte le intensità dei pixel nell'immagine di input I all'interno della regione rettangolare formata dall'origine e \mathbf{x} .

$$I_{\Sigma}(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \quad (1.1)$$

Una volta che l'immagine integrale è stata creata, sono necessarie solo tre addizioni per calcolare la somma delle intensità su qualunque area rettangolare. (Vedi figura 1.1)

¹La matrice Hessiana è una matrice quadrata delle derivate parziali di secondo ordine di una funzione a valori scalari o di un campo scalare.

²I *filtri convoluzionali*, anche chiamati *Kernel*, *matrici convoluzionali* o *maschere*, sono piccole matrici utili per smussare, definire, rinforzare e molto altro. Questo è ottenuto effettuando una convoluzione tra il kernel e l'immagine

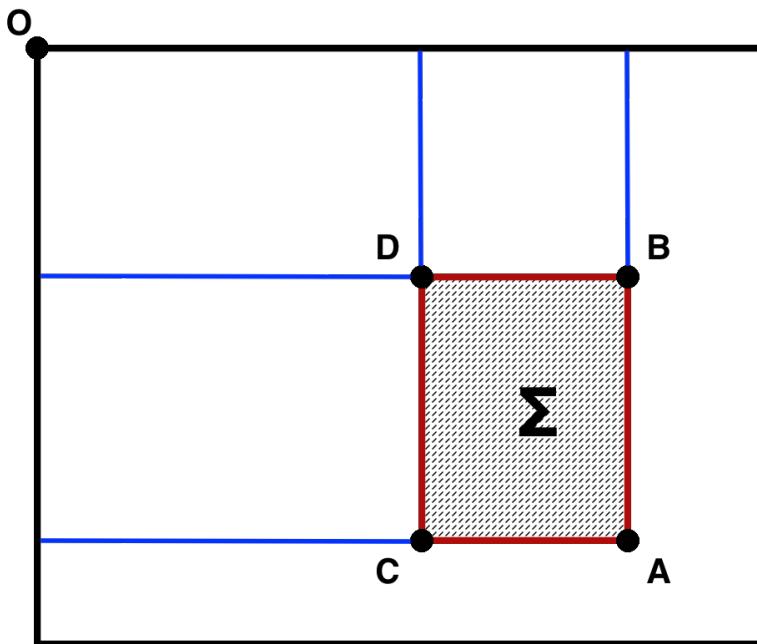


Figura 1.1: $\Sigma = A - B - C + D$ - Usando le *Integral Images*, sono necessarie solo tre addizioni e quattro accessi in memoria per calcolare la somma delle intensità all'interno di una qualunque area rettangolare.

Quindi, il tempo di calcolo è indipendente dalle dimensioni dell'area. Questo è molto importante poiché, in questo approccio, vengono utilizzati filtri di grandi dimensioni.[6]

Punti di interesse basati sulla matrice Hessiana

Dopo aver calcolato la forma integrale dell'immagine di input si procede alla rilevazione dei punti di interesse. Questa è basata sulla matrice Hessiana per la sua accuratezza e buone performance. Più precisamente, si rilevano strutture a *Blob* in locazioni dove il determinante è massimo. Si basa sul determinante dell'Hessiana anche la scelta del fattore di scala.

Dato un punto $\mathbf{x} = (x, y)$, in un' immagine I , la matrice Hessiana $H(x, \sigma)$

in \mathbf{x} alla scala σ , è definita come:

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (1.2)$$

dove $L_{xx}(x, \sigma)$ è la convoluzione della derivata seconda della Gaussiana $\frac{\delta^2}{\delta x^2}g(\sigma)$ con l'immagine I nel punto \mathbf{x} e, allo stesso modo, $L_{xy}(x, \sigma)$ e $L_{yy}(x, \sigma)$. Le Gaussiane sono ottime per l'analisi della scala spaziale ma, in pratica, vanno discretizzate e ridimensionate per poter sfruttare le immagini integrali. Queste derivate del secondo ordine di Gaussiane approssimate, possono essere calcolate ad un costo computazionale davvero basso usando le *Integral Images* che rendono il tempo di calcolo indipendente dalla dimensione del filtro.

Il determinante approssimato della matrice Hessiana è la risposta a *Blob* dell'immagine I alla posizione \mathbf{x} . Queste risposte sono salvate in una mappa a diverse scale utili a calcolare i massimi locali (vedi "Localizzazione punti di interesse", 1.1.1).[6]

Rappresentazione dello spazio di scala

I punti di interesse devono essere trovati a differenti scale anche perché, spesso, la ricerca di corrispondenze richiede il confronto tra immagini dove sono visti a dimensioni diverse.

La rilevazione tramite Hessiana deve quindi essere eseguita e calcolata a fattori di scala differenti.

Gli spazi di scala sono spesso rappresentati come una piramide di immagini. Queste sono ripetutamente smussate con un filtro gaussiano e *sub-sampled* per ottenere un livello più alto della piramide prima di procedere alla rilevazione di punti di interesse. Dato l'utilizzo di filtri convoluzionali e *Integral Images* e sapendo che il costo di applicazione di un filtro è costante

e non dipende dalle sue dimensioni, anziché scalare e ridurre le dimensioni dell’immagine, che sarebbe più costoso, si aumentano le dimensioni del filtro.

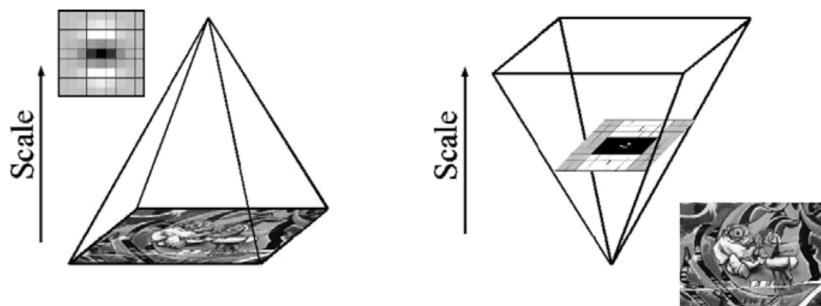


Figura 1.2: Piuttosto che ridurre iterativamente la dimensione dell’immagine (sinistra), l’uso delle *Integral Images* permette di aumentare le dimensioni del filtro a costo costante (destra).

Lo spazio di scala è diviso in ottave (*octaves*) che rappresentano una serie di mappe di risposta ottenute calcolando la convoluzione della stessa immagine di input con filtri di dimensione crescente. Un’ottava, in totale, rappresenta un fattore di scala 2 che porta al raddoppiamento della dimensione del filtro. Ogni ottava è suddivisa in un numero costante di livelli. Per ogni nuova ottava la dimensione del filtro è raddoppiata e, allo stesso tempo, anche gli intervalli tra un livello e l’altro raddoppiano, essendo il numero di intervalli costante. I livelli si fanno più distanti a dimensioni grandi del filtro, dove piccole imprecisioni risultano irrilevanti. In questo modo i costi computazionali sono ridotti e la perdita di accuratezza è accettabile e, comunque, non inferiore al *sub-sampling* degli approcci classici.

Il numero di ottave utilizzato è all’incirca 4 e, in generale, dipende anche dalle dimensioni dell’immagine di input. È dimostrato statisticamente che il numero di punti di interesse rilevati per ottava scende rapidamente con l’aumentare delle dimensioni. Ha quindi senso fermarsi e non effettuare computazioni ulteriori.[6]

Localizzazione dei punti di interesse

Per localizzare i punti di interesse sulle diverse scale si applica una "*non-maximum suppression*"³ sui 3x3x3 vicini. Nello specifico se ne utilizza una variante molto veloce descritta in [7].

I massimi dei determinanti della matrice Hessiana sono poi interpolati nello spazio dell'immagine e di scala. L'interpolazione nello spazio della sala è molto importante in questo caso poiché le distanze tra i primi strati di ogni ottava sono relativamente grandi.[6]

Descrizione dei punti di interesse

Il "descrittore SURF" descrive la distribuzione dell'intensità di contenuto nelle vicinanze dei punti di interesse in maniera simile al gradiente di informazioni estratto dagli approcci classici come *SIFT*⁴ ed i suoi derivati. La differenza sta nell'utilizzo delle risposte a *wavelet di Haar* di prim'ordine⁵ negli assi *x* ed *y*, sfruttando le *Integral Images* e utilizzando solo 64 elementi nei vettori descrittivi (esiste ed è implementata anche una versione con dimensione del descrittore uguale a 128 elementi). Questo riduce i tempi di computazione per la descrizione e per il matching delle features e, allo stesso tempo, ne aumenta la robustezza. Inoltre viene introdotto un nuovo step di indicizzazione basato sul segno della Laplaciana che aumenta la robustezza del descrittore e la velocità di matching (nel caso migliore addirittura di un

³Non-maximum suppression è una tecnica di assottigliamento dei bordi. Spesso utilizzata in fase di rilevamento dei bordi, per renderli ben definiti.

⁴Scale-invariant feature transform (o SIFT) è un algoritmo in visione computazionale per rilevare e descrivere punti di interesse locali in immagini. L'algoritmo è stato pubblicato da David Lowe nel 1999.

⁵Wavelet di Haar è una sequenza di funzioni di "forma quadrata" riscalate che insieme formano una famiglia di wavelet o una base.

fattore 2).

Il primo step consiste nello scegliere e fissare una orientazione in maniera riproducibile da una regione circolare intorno al punto di interesse. Il secondo step consiste nel costruire una regione quadrata allineata all'orientazione scelta per estrarre il descrittore SURF da questa.

Scelta dell'orientamento

Per essere invariante all'orientazione delle immagini, l'algoritmo identifica un orientamento riproducibile per i punti di interesse.

Per questo si calcolano le risposte a wavelet di Haar nelle direzioni x ed y in una regione circolare di raggio $6s$ centrata nel punto di interesse, con s la scala alla quale il punto di interesse è stato rilevato. Lo step di campionamento è dipendente dalla scala ed è scelto uguale ad s e, anche le dimensioni della wavelet di Haar sono dipendenti dalla scala prese con dimensione laterale di $4s$. I filtri utilizzati sono mostrati in figura 1.3. Si sfruttano ancora le *Integral Images*, grazie alle quali sono necessarie solo 6 operazioni per calcolare le risposte nelle direzioni x ed y a qualunque scala.



Figura 1.3: Filtri Wavelet di Haar per calcolare le risposte in x (sinistra) e y (destra). Le parti scure hanno valore -1 e le parti chiare 1.

Una volta che le risposte sono state calcolate e pesate con una Gaussiana (con varianza proporzionale al fattore di scala s), centrata sul punto di interesse, queste vengono rappresentate come punti in uno spazio con la forza di risposta verticale nelle ordinate e quella orizzontale nelle ascisse. L'orienta-

zione dominante è stimata calcolando la somma di tutte le risposte con una finestra di orientamento mobile (*Sliding Orientation Window*) di dimensione $\frac{\pi}{3}$. Le risposte verticali ed orizzontali all'interno della finestra sono sommate. Queste generano il vettore di orientamento locale. Il più lungo di questi vettori tra tutte le finestre definisce l'orientamento del punto di interesse. (vedi in figura 1.4)[6]

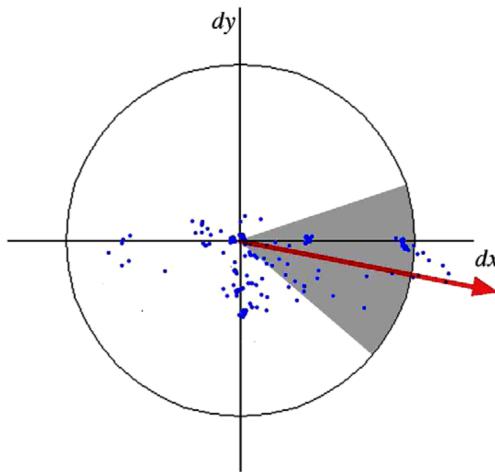


Figura 1.4: Calcolo del vettore di orientamento del punto di interesse.

Descrizione

Si procede con la costruzione di una regione quadrata centrata sul punto di interesse ed orientata secondo l'orientazione calcolata precedentemente. La dimensione di questa regione è di $20s$. La regione è poi divisa in 4×4 ottenendo 16 sotto-regioni regolari. Per ognuna delle sotto-regioni si calcolano le risposte alle wavelet di Haar (filtro di dimensioni $2s$). Anche in questo caso, per aumentare la robustezza, queste risposte vengono pesate utilizzando una Gaussiana centrata nel punto di interesse. Le risposte verticali ed orizzontali sono sommate per ogni sotto-regione e formano un primo set di valori da inserire nel vettore descrittore.

Inoltre, per mantenere le informazioni riguardanti la polarità e variazioni di intensità, si estrae anche la somma dei valori assoluti delle risposte. Quindi, ogni sotto-regione genera un vettore di descrizione di dimensione 4 strutturato in questo modo:

$$\mathbf{v} = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y| \right) \quad (1.3)$$

dove con d_x e d_y si intendono, per semplicità, le risposte alle wavelet di Haar orizzontali e verticali.

Concatenando questo per tutte le sotto-regioni si ottiene il vettore descrittore di lunghezza 64.[6]

Per ottenere una maggiore accuratezza e distinzione è possibile utilizzare un vettore descrittore esteso di dimensione pari a 128. Le somme dei d_x e $|d_x|$ solo calcolate separatamente per $d_y < 0$ e $d_y \geq 0$ e, analogamente, le somme dei d_y e $|d_y|$ sono suddivise secondo il segno di $|d_x|$, raddoppiando così il numero di elementi nel vettore. Questo non aggiunge molta complessità nei calcoli e permette una maggiore efficacia nella fase di matching.[4]

Indicizzazione

Per un'indicizzazione rapida in fase di matching è incluso anche il segno della Laplaciana (traccia della matrice Hessiana) per il punto di interesse in analisi. Tipicamente i punti di interesse sono strutture di tipo *blob*. Il segno della Laplaciana distingue blob chiari su fondo scuro da blob scuri su fondi chiari. Questa informazione è disponibile senza costi computazionali aggiuntivi, essendo calcolata già in fase di rilevamento.

In fase di matching si comparano punti di interesse che hanno lo stesso tipo di contrasto (vedi figura 1.5). Questa minima informazione permette di rendere molto più rapido il matching senza ridurre le prestazioni del descrittore.[6]

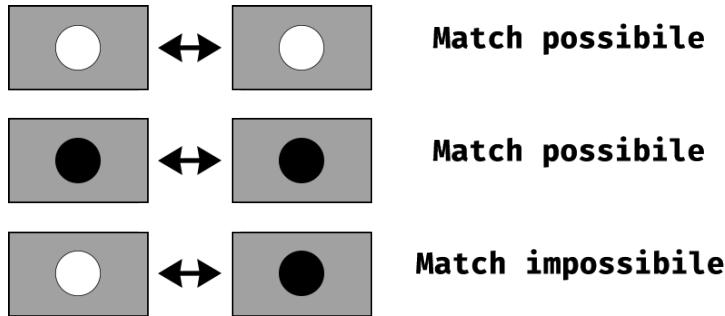


Figura 1.5: Se il contrasto tra due punti di interesse è differente, il candidato non è considerato nella fase di confronto.

1.1.2 Metodi alternativi

I risultati ottenuti con i primi test in fase progettuale non sembravano soddisfacenti e, per questo, si sono cercate nuove soluzioni per ottenere migliori prestazioni. Per fare questo sono stati studiati e testati anche altri algoritmi di estrazione e descrizione di punti di interesse. Una soluzione proposta è stata quella di utilizzare un approccio ibrido, cioè estrarre le features con un algoritmo più leggero e più rapido rispetto a SURF per poi effettuare la descrizione con un secondo algoritmo, poiché, ai fini di un buon riconoscimento, l'importante è trovare match con una certa confidenza e non il numero di questi. Infatti, ai fini applicativi, sarebbero necessari soltanto 4 match corretti e precisi per poter effettuare l'omografia e il tracciamento per riconoscere correttamente l'oggetto nel contesto. Uno degli algoritmi testati è ORB.

ORB

Questo algoritmo è stato ideato da Ethan Rublee, Vincent Rabaud, Kurt Konolige e Gary R. Bradski nei laboratori di OpenCV (vedi A.2) nel 2011 e presentato nell'articolo "ORB: An efficient alternative to SIFT or SURF"[8].

ORB è, in pratica, una fusione tra il rilevatore di punti di interesse *FAST*⁶ ed il descrittore *BRIEF*⁷ con l'aggiunta di numerose modifiche utili ad aumentare le prestazioni. Prima utilizza *FAST* per trovare i punti di interesse, poi applica la misura di angoli di Harris per trovare i migliori N punti tra questi. Utilizza, inoltre, lo spazio di scala a piramide per produrre features multi-scala (si utilizza l'approccio classico in modalità diversa da SURF - vedi parte sinistra figura 1.2).

FAST, però, non calcola e non tiene in considerazione l'orientamento e gli autori hanno dovuto modificare l'algoritmo calcolando il baricentro pesato con l'intensità di una zona d'immagine centrata nel punto rilevato. La direzione del vettore dal punto di interesse al baricentro indica l'orientamento.

Per la descrizione, ORB utilizza il descrittore *BRIEF*, il quale però non ottiene ottimi risultati con la rotazione. ORB quindi provvede a "girare" e guidare *BRIEF* secondo l'orientamento rilevato precedentemente.

L'algoritmo discretizza gli incrementi d'angolo in angoli di $\frac{2\pi}{30}$ (12 gradi) e costruisce una tabella di look-up precalcolata di schemi (*patterns*) *BRIEF*.

Vista la sua velocità, ORB è adatto per sistemi a bassi consumi e basse prestazioni in cui, però, non è necessaria una grande accuratezza e versatilità.[9]

Prestazioni ottenute

L'approccio ibrido, purtroppo, non ha dato i risultati sperati. Si sono ottenuti dei miglioramenti marginali nelle prestazioni computazionali e

⁶Features from accelerated segment test (FAST) è un metodo di rilevamento di angoli e bordi molto leggero e rapido ma non ad alte prestazioni.

⁷BRIEF: Binary Robust Independent Elementary Features usa stringhe binarie come descrittori e permette una valutazione rapida, basata sulla distanza di hamming, piuttosto che sulla norma della distanza.

temporali ma, allo stesso tempo, un certo peggioramento nella qualità del riconoscimento. Sperimentalmente si nota la presenza di falsi positivi nel riconoscimento e una restrizione del dominio di oggetti su cui è applicabile. Per questo si è deciso di abbandonare l'approccio.

1.2 Keypoint Matching

Una volta trovati e descritti i punti di interesse su entrambe le immagini, quella di input e quella su cui vogliamo effettuare il riconoscimento, si procede con una fase di confronto durante la quale si stabilisce se due punti di interesse nelle due immagini corrispondono alla stessa caratteristica (*Feature*) e allo stesso punto dell'oggetto. Questo è importante ed è grazie a questi che si riesce a tracciare l'oggetto. Questa fase è detta di Keypoint Matching (confronto dei punti di interesse) e consiste nell'assegnare uno score (punteggio) alla somiglianza tra due keypoints sfruttando la loro descrizione (molto spesso questo avviene effettuando la distanza euclidea tra i vettori descrittivi). Se lo score è alto (la distanza è quindi piccola) è probabile che i due punti corrispondano, altrimenti no. Vi sono vari algoritmi che permettono di fare questo. Alcuni si avvalgono di modelli probabilistici per decidere su quali punti effettuare un confronto, altri agiscono su ogni coppia possibile di punti. L'approccio probabilistico comporta un miglioramento delle prestazioni, a volte anche notevole, solo se applicato a problemi con un numero di keypoints da analizzare piuttosto elevato (circa un ordine di grandezza più grande di quello necessario e sperimentalmente verificato nel progetto di tesi). La creazione delle strutture ausiliarie e computazioni probabilistiche, per un numero ridotto di keypoints, non è più vantaggiosa (overhead grande). Si è quindi optato per un matching di tutte le possibili coppie di keypoints

detto *Brute Force Matcher*.

1.2.1 Brute Force Matcher

Il Brute Force Matcher è molto semplice. Prende il descrittore di un punto di interesse nel primo set (quello appartenente all'immagine di input) e lo confronta con tutti i descrittori dei punti di interesse del secondo set usando dei calcoli di distanza. Ritorna poi il più vicino. Questo ripetuto per tutti i descrittori del primo set.

Di solito la distanza utilizzata è la *distanza euclidea* detta anche *Norma L-2* che si presta bene per descrittori come SIFT e SURF.

La distanza euclidea tra due vettori x ed y è calcolata come: $d(x, y) := \|x - y\| = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$.

Si tengono solo i migliori match trovati in relazione ad un valore di soglia, oppure, si tengono i migliori tra quelli ottenuti.

1.3 Omografia

In matematica e geometria una omografia è una relazione tra punti di due spazi tali per cui ogni punto di uno spazio corrisponde ad uno ed un solo punto del secondo spazio. Dato un insieme di punti x_i ed un insieme di punti corrispondenti x'_i espressi in coordinate omogenee, si vuole stabilire una trasformazione in grado di trasformare i punti x_i nei punti x'_i . Tale trasformazione riveste grande importanza nella trasformazione di punti da un piano ad un altro, detta omografia bidimensionale, che è quella utilizzata in questo progetto di tesi per proiettare l'immagine di input dell'oggetto sul contesto in cui questo viene cercato, in modo da poterne stimare la posizione, rotazione ed inclinazione.

I punti su cui effettuare la trasformazione, nel caso bidimensionale, devono essere in numero maggiore di 4. Più grande è il numero di punti, migliore e più precisa sarà la trasformazione (considerando corrispondenze con un certo grado di precisione).

Capitolo 2

Implementazione mediante sistemi embedded e applicazione al controllo di posizione

In questo capitolo sono indicati i dettagli della realizzazione del progetto, dalla configurazione hardware alle fasi di sviluppo del software.

2.1 Sistemi embedded

Il sistema, realizzato nell'ottica della leggerezza, delle piccole dimensioni e dei bassi consumi, si avvale di piattaforme *embedded* per effettuare le elaborazioni e il controllo dell'hardware disponibile. In generale, con piattaforma *embedded*, si intende un dispositivo di elaborazione integrato, all'interno di un sistema fisico, capace di controllare le funzioni tramite un apposito programma software, dedicato e progettato per una determinata applicazione, supportato da una piattaforma hardware su misura.

I sistemi embedded sono di dimensioni ridotte e privi di interazione umana che, spesso, si trovano a dover soddisfare vincoli temporali. Impiegano l'hardware e risorse strettamente necessari e sono dispositivi dai bassi consumi e dai costi ridotti.

Più del 98% dei computer attualmente presenti nel mondo sono del tipo "embedded", ovvero, sono parte integrante di sistemi più estesi. Questi componenti hanno lo scopo di gestire le risorse del sistema globale di cui fanno parte, monitorandone e controllandone le funzionalità: il loro ruolo viene espletato tramite l'interazione con appositi dispositivi hardware. Negli ultimi trent'anni, i sistemi di calcolo embedded sono cresciuti in modo esponenziale in settori quali l'automazione industriale, l'avionica, la progettazione di autoveicoli, le telecomunicazioni, l'elettronica di consumo e la robotica. I sistemi embedded stanno trovando sempre maggior spazio in moltissime aree applicative, quali, per esempio, lo sport e la medicina, l'agricoltura, l'industria dei giochi e dei videogiochi, la protezione civile e i sistemi di trasporto intelligenti.[10]

2.1.1 RaspberryPi

Il Raspberry Pi è un piccolo e minimale computer sviluppato nel Regno Unito da un'organizzazione no-profit, chiamata "*The Raspberry Pi Foundation*", con l'intenzione di fornire un computer a basso costo e software gratuito ed aperto per promuovere l'insegnamento delle basi dell'Informatica nelle scuole e nei paesi in via di sviluppo.

Il computer è contenuto in una singola scheda elettronica con le dimensioni di una carta di credito ed include porte e connessioni come:

- HDMI (uscita video)
- USB 2.0 (4 porte)

- Audio analogico
- Alimentazione
- Ethernet (per connettersi ad internet)
- Scheda micro SD (utilizzata come memoria di massa)
- WiFi e Bluetooth nei modelli più recenti

Raspberry Pi si basa interamente su software open-source e dà agli studenti, ai quali è destinato, la possibilità di utilizzare, modificare e sostituire il software secondo i loro bisogni.

Il Raspberry Pi ha debuttato nel Febbraio 2012 quando il gruppo *Raspberry Pi Foundation* ha attivato il progetto per rendere i computer divertenti agli studenti e, allo stesso tempo, creare interesse sul funzionamento dei computer ad un livello base. Diversamente dall'utilizzare un computer con il proprio *case* già assemblato, il Raspberry Pi mostra i componenti fondamentali che di solito si troverebbero sotto la plastica. Sono state rilasciate diverse versioni negli ultimi anni:

- Prima generazione: Raspberry Pi 1 Model B
- Raspberry Pi 1 Model A: un modello più piccolo e semplice
- Raspberry Pi 1 Model B+: modello migliorato rilasciato nel 2014
- Raspberry Pi Zero: un modello di dimensioni inferiori, con ridotto I/O, il venduto per soli 5\$ rilasciato nel 2015
- Raspberry Pi 2 Model B: con prestazioni migliorate e I/O aumentato (Il modello utilizzato in questo progetto) rilasciato sempre nel 2015
- Raspberry Pi 3 Model B che include anche WiFi e Bluetooth integrati rilasciato nel 2016. Al momento questo è l'ultimo modello ed il migliore.

Negli ultimi anni, il Raspberry Pi ha trovato grandi applicazioni anche per la ricerca e nel mondo accademico con decine di progetti mirati e creati ad hoc.

Si presta molto bene come strumento di prototipazione per sistemi embedded, infatti, sono caratteristiche distintive le sue ridotte dimensioni, bassi consumi che non impediscono però di ottenere elevate prestazioni e una grande varietà di sistemi input output.

Gli ultimi modelli rendono possibili applicazioni di visione computazionale. Tra i dispositivi di I/O, per cui il Raspberry Pi è predisposto, troviamo infatti una fotocamera e molto altro.

PiCamera

Tra i dispositivi di I/O disponibili per il Raspberry Pi è presente la *PiCamera*, una videocamera dalle dimensioni e consumi ridotti, che permette al Raspberry Pi di catturare immagini e/o video dell'esterno per essere utilizzate in varie applicazioni e in vari modi. La PiCamera si connette attraverso il connettore dedicato presente su Raspberry Pi con collegamento a pettine. Sono presenti, tra i software installati, anche delle librerie che permettono di controllare tramite codice la PiCamera (sono disponibili APIs per diversi linguaggi, ad esempio Python che è il linguaggio utilizzato in questo progetto - vedi 2.3).

Configurazione

Per questo progetto è stato utilizzata la versione Raspberry Pi 2 Model B che monta un processore quad-core ARM Cortex-A7 a 900MHz e 1 GB di RAM.

Sul Raspberry Pi è stato installato il sistema operativo open-source *Raspbian*¹, fornito direttamente dalla casa produttrice, sul quale è preinstallato Python (vedi A.1) e alcune delle librerie relative all'utilizzo dell'hardware

¹Raspbian è un sistema operativo Linux basato sulla distribuzione Debian.

integrato. In aggiunta sono stati installati i pacchetti Serial e Numpy per Python.

Si è proceduto alla compilazione ed installazione della libreria OpenCV (vedi A.2) con i relativi bindings² per Python.

Inoltre, è stato configurato un server mDNS (vedi A.7), nello specifico avahi-daemon, per ottenere una rete zeroconf in cui non è necessario conoscere relativamente l'indirizzo IP del Raspberry Pi e del dispositivo mobile in cui viene lanciata l'applicazione remota (2.6.1) o del computer con cui viene programmato il sistema.

Raspberry Pi viene alimentato da una batteria USB standard, attualmente in commercio, che fornisce una tensione di 5 Volt e un'intensità massima di 2 Ampere che sono sufficienti ad alimentare il Raspberry, la PiCamera ed il dongle WiFi.

2.1.2 Arduino

Arduino è una piattaforma hardware low-cost programmabile con cui è possibile creare circuiti di ogni tipo per molte applicazioni, soprattutto, in ambito di robotica ed automazione. Si basa su un Microcontrollore³ della ATMEL: l'ATMega168/328.

Nasce a Ivrea nel 2005 da un'idea di un Professore universitario, un Ingegnere Elettronico, Massimo Banzi, che decise di creare una piattaforma per i propri studenti in grado di facilitarli nello studio dell'Interaction Design. Il successo ottenuto spinse l'ingegnere a rendere questa piattaforma Open

²interfaccia APIs per poter utilizzare i metodi della libreria scritta in C++ su Python.

³In elettronica digitale, il microcontrollore o MCU (MicroController Unit), è un dispositivo elettronico integrato su singolo chip nato come evoluzione alternativa al Microprocessore ed utilizzato generalmente in sistemi embedded.

Source (in realtà è Open Hardware - è possibile trovare sul sito ufficiale Arduino, i circuiti, i componenti e, addirittura le istruzioni per realizzarla da soli).

Gli schemi circuitali, essendo Open e quindi visionabili da tutti, possono essere continuamente migliorati dalla comunità e, grazie ad essi, sono state sviluppate un numero incredibile di librerie software che rendono davvero semplice l'interfaccia con periferiche di qualsiasi tipo.

Fu un gruppo di studenti della facoltà di Ingegneria Informatica a scrivere la libreria multi-piattaforma, l'IDE⁴ e le prime API. Grazie a questi studenti, Arduino si programma in modo fluido, semplice e intuitivo. In Internet, addirittura, si possono trovare librerie già scritte in relazione ai vari bisogni.

Il Team Arduino è composto da Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. Il nome Arduino proviene da un bar a Ivrea, dove i fondatori erano soliti incontrarsi. Il bar era intitolato in onore di Arduino d'Ivrea Re d'Italia nel XIII secolo.

La scheda Arduino è in grado di interagire con l'ambiente in cui si trova ricevendo informazioni da una grande varietà di sensori e può comandare luci, LED, motori e altri attuatori.

Il linguaggio di programmazione è basato su Wiring (un ambiente di programmazione Open-Source pensato per una facile applicazione in grado di semplificare la programmazione in C e C++) e sull'interfaccia Processing. I progetti basati su Arduino possono essere indipendenti oppure essere interfacciati con altri software come Processing, MaxMSP, Flash o altri dispositivi hardware, come in questo caso il Raspberry Pi.

Sono disponibili anche delle schede elettroniche dedicate, chiamate *Shield*, che possono essere accoppiate ad Arduino attraverso i pin permettendo l'e-

⁴Integrated Development Environment: ambiente di sviluppo.

stensione delle sue funzionalità e/o l'aggiunta di nuovi tipi di comunicazione I/O.

Motor Shield

Una delle Shields disponibili per Arduino è la *Motor Shield* che permette di controllare motori di tipo *Servo*⁵, passo passo⁶ e normali in corrente continua. Allo stesso tempo consente di utilizzare un' alimentazione esterna con voltaggi anche più alti di quelli fornibili da Arduino per alimentare i motori.

Configurazione

Per questo progetto è stato utilizzato un "Arduino Uno" a cui è stato collegato il Motor Shield. Arduino è stato connesso tramite cavo USB al Raspberry Pi. Al Motor Shiled è collegata l'alimentazione esterna, fornita da 4 pile stilo AA in serie che generano 6 Volt. Il Motor Shield a sua volta alimenta Arduino e i motori. Su Arduino sono state caricate alcune librerie:

- Libreria Adafruit per controllare il Motor Shield: *AFMotor*
- Libreria per l'interpretazione di JSON - formato utilizzato per la comunicazione tra Arduino e Raspberry Pi: *ArduinoJson*.

2.2 Costruzione

I componenti utilizzati sono mostrati in figura 2.1. Il sistema mobile autonomo è costituito da un piano rettangolare orizzontale in plexiglas reso mobile da tre ruote di cui due motrici. I due motori che permettono la

⁵Un Servo è un attuatore di rotazione o lineare che permette un controllo preciso della posizione.

⁶Motore che si muove in step discreti.

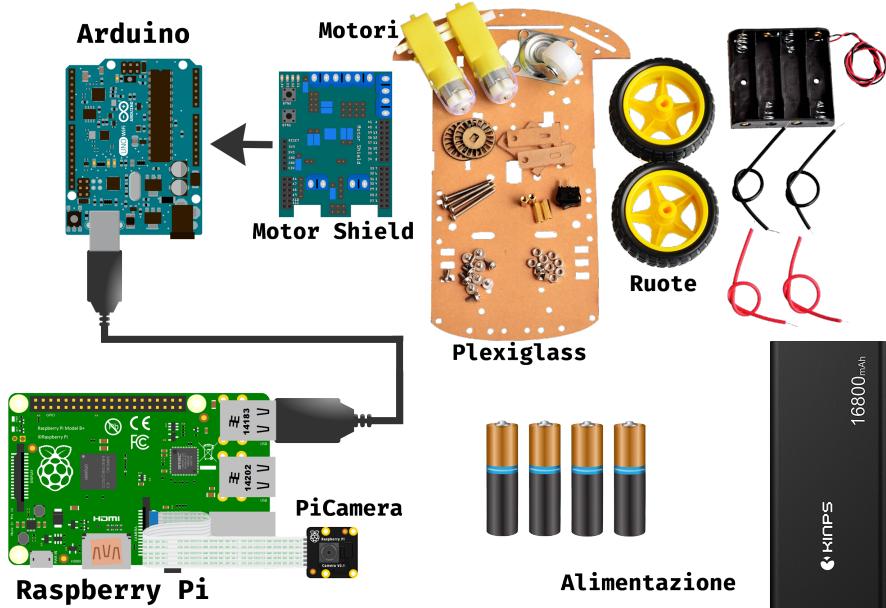


Figura 2.1: Componenti utilizzati

propulsione, uno per ogni ruota motrice, sono collegati sotto al piano. Sulla parte superiore sono assemblati i componenti addetti all'elaborazione e l'alimentazione su due livelli. Sul primo sono stati fissati Arduino, la sua shield per i motori e le batterie per alimentarli. Sul secondo piano, ottenuto con un'altra struttura in plexiglas sagomata, si trova l'alimentazione per il Raspberry Pi e il dispositivo stesso. Questa struttura sopraelevata fornisce anche un supporto ed una protezione per la videocamera che è collegata al Raspberry Pi. Raspberry Pi e Arduino sono collegati tramite un cavo USB. Il sistema è programmabile e controllabile senza bisogno di collegamenti fisici tramite WiFi, essendo RaspberryPi dotato di dongle WiFi⁷.

Per questo progetto è stata sviluppata anche un App mobile che svolge le funzioni di visore remoto e di controllo remoto.

⁷Si utilizza *SSH (Secure Shell)* per controllare il Raspberry, per inviare e ricevere files e per eseguire comandi.

Software

Come linguaggio di programmazione per questo progetto è stato scelto Python (vedi A.1), perchè ottimo in fase di prototipazione. Questo permette di gestire con semplicità tutte le diverse tecnologie impiegate contemporaneamente come comunicazione seriale, computer vision, librerie matematiche e socket TCP/IP. Per le operazioni di riconoscimento visivo è stata utilizzata la libreria OpenCV (vedi A.2), tramite i relativi bindings per Python⁸, che fornisce implementazioni degli algoritmi di estrazione e descrizione di punti d'interesse qui utilizzate. Mette anche a disposizione metodi per effettuare matching (vedi 1.2) e omografie (vedi 1.3). Permette di trattare le immagini come matrici (*Mat* è il nome della classe in C++) che in Python sono matrici del pacchetto Numpy⁹.

Per l'App mobile è stata utilizzata la piattaforma iOS e Swift come linguaggio di programmazione (vedi A.4).

2.3 Riconoscimento visivo

Il primo passo è effettuare il riconoscimento visivo per calcolare la posizione relativa dell'oggetto da raggiungere e seguire.

Il Raspberry Pi è il dispositivo incaricato di eseguire questo compito tramite uno script Python utilizzando la PiCamera per ottenere un flusso di immagini continuo da elaborare. Una volta selezionata l'immagine di input dell'oggetto da identificare, si acquisiscono i frame dalla PiCamera per poter iniziare a cercare l'oggetto.

⁸interfaccia APIs per poter utilizzare i metodi della libreria scritta in C++ su Python

⁹Pacchetto Python per il calcolo scientifico

Dapprima si applicano le tecniche di riconoscimento (viste nel capitolo 1) sull’immagine di input, estraendo i punti d’interesse relativi all’oggetto da riconoscere, ottenendone una descrizione per poterli confrontare con quelli che verranno eventualmente trovati nell’analisi dei singoli frame.

Si procede poi con l’analisi del flusso continuo di immagini acquisite applicando le stesse tecniche di riconoscimento visivo. Si estraggono in ogni frame i punti d’interesse e si descrivono con l’algoritmo SURF (visto nella sezione 1.1.1), confrontandoli con quelli trovati nell’immagine di input con l’algoritmo di *matching* (presentato nella sezione 1.2). Per rendere l’algoritmo funzionante in diverse condizioni di luce, visto che *SURF* è invariante a variazioni di scala e di orientamento, ma non a variazioni di intensità, si applica un’equalizzazione dell’immagine (dell’istogramma) prima dell’elaborazione di ogni frame. Tra i possibili *match*¹⁰ rilevati, si scelgono i migliori.

Si ottengono quindi coppie di punti tra l’immagine di input ed il frame attuale che corrispondono con una certa confidenza. Con questi è possibile, se in numero maggiore di 4, effettuare un omografia (vedi 1.3) con la quale si calcola ed ottiene una proiezione dell’oggetto sul frame in analisi. Si effettua un ulteriore controllo sulla qualità della corrispondenza trovata, calcolando la superficie della proiezione dell’oggetto, per verificare che non sia troppo piccola o troppo grande (i calcoli vengono effettuati in relazione alle dimensioni del frame). Dalla proiezione possiamo stimare la posizione dell’oggetto intesa come *displacement*¹¹ relativo al centro ottico dell’immagine (vedi figura 2.2). Questo viene calcolato su entrambi gli assi *x* e *y* ed è normalizzato tra 0 ed 1, dove un displacement pari alla dimensione del frame corrisponde ad 1.

¹⁰Un match rappresenta due punti d’interesse per cui la distanza tra i descrittori è sotto una certa soglia.

¹¹Discostamento dal centro ottico del frame

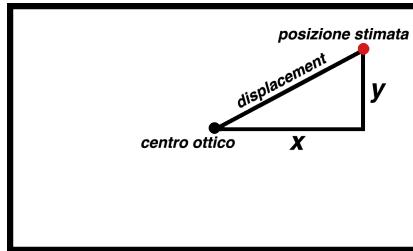


Figura 2.2: Rappresentazione del *displacement*

Si è deciso di considerare l’oggetto di fronte al sistema se il displacement sull’asse *x* ha un valore sotto una certa soglia (nello specifico $\frac{1}{8}$ della dimensione orizzontale del frame), altrimenti se il valore è superiore alla soglia stabilità è necessario che il sistema ruoti per poter raggiungere l’oggetto.

Queste informazioni vengono poi comunicate ad Arduino che funge da attuatore interpretando i dati ed eseguendo le operazioni necessarie.

Se richiesto, ponendo l’apposita flag *Remote* uguale a *True*, lo script procede all’invio del frame appena ottenuto, con sovrappressi i risultati del riconoscimento, tramite connessione TCP verso l’applicazione mobile (descritta nel paragrafo 2.6.1).

2.4 Interfaccia di collegamento con Arduino

Arduino e Raspberry Pi sono connessi tramite un cavo USB e utilizzano quindi la comunicazione seriale (vedi A.6) per comunicare tra di loro.

Raspberry Pi, dopo aver estrappolato le informazioni di posizione dai frame, deve comunicare ad Arduino le istruzioni per poter orientare in maniera corretta il sistema e alimentare i motori per avanzare verso l’oggetto nel caso in cui questo venga rilevato. Le informazioni di posizione, come già visto nella sezione 2.3, sono sotto forma di displacement nei due assi *x* e *y* se l’oggetto è discostato dal centro ottico, mentre se l’oggetto viene rilevato e si

trova nelle vicinanze del centro del frame, si comunica solo il fatto di averlo trovato.

Queste informazioni vengono inviate tramite la connessione seriale utilizzando la libreria Serial per Python su Raspberry Pi e la libreria Serial già disponibile su Arduino per riceverle. Il formato con cui vengono racchiuse è *JSON* (vedi A.5) che garantisce l'integrità dei dati, la facilità di codifica ed interpretazione di quest'ultimi e rende l'interfaccia facilmente estendibile per l'aggiunta o modifica di funzionalità. Gli oggetti JSON inviati, nel caso in cui si debbano comunicare informazioni riguardanti l'orientazione del sistema, sono del tipo:

```
{
    "coordinates": {
        "x": 0.5,
        "y": 0.3
    }
}
```

L'oggetto "*coordinates*" contiene le informazioni riguardanti il displacement sull'asse *x* e sull'asse *y* con valori compresi tra 0 ed 1. Positivi se l'oggetto si trova a destra per l'asse *x* e in basso per l'asse *y* rispetto al centro ottico, negativo viceversa. Nel caso in cui si voglia comunicare di procedere avanti sono invece del tipo:

```
{
    "found": true
}
```

Questo significa che l'oggetto è stato trovato ed è di fronte al sistema. Se il valore di "*found*" è *false* significa che l'oggetto non è più nella visuale e

si comunica di frenare. I vari tipi di informazioni, grazie al formato JSON, possono essere inviate anche insieme. Ad esempio un tipico oggetto inviato è della forma:

```
{
    "coordinates": {
        "x": -0.2,
        "y": 0.1
    },
    "found": true
}
```

in questo modo comunichiamo di cambiare orientazione e di muoversi avanti.

Raspberry Pi e, più precisamente Python, gestiscono nativamente JSON e sono in grado di codificare direttamente le informazioni in questo formato. Su Arduino, invece, è stata installata una libreria chiamata ArduinoJson¹².

2.5 Controllo Motori

Con le informazioni ricevute nella fase precedente, Arduino attua delle misure per eseguire i compiti richiesti. Sono state implementate 3 funzioni principali: una per andare avanti e le altre due per compiere rotazioni discrete a destra e a sinistra. Le funzioni destra e sinistra prendono in ingresso il displacement ricevuto e, conoscendo l'angolo di visualizzazione della camera, il tempo impiegato per compiere una rotazione pari all'angolo di visualizzazione¹³, attivano i motori (uno verso avanti e l'altro nel verso opposto) per un tempo calcolato come: $t_m = T_a \cdot d_x$, dove t_m è il tempo di movimento, T_a è il

¹²Progetto open-source su GitHub: <https://github.com/bblanchon/ArduinoJson>

¹³Porzione di spazio che l'obiettivo ci permette di vedere.

tempo stimato per compiere una rotazione pari all'angolo di visualizzazione e d_x è il modulo del displacement sull'asse x che è un numero compreso tra 0 e 1, quindi, la rotazione effettuata sarà una frazione di un intero angolo di visualizzazione.

Nel caso sia ricevuto il comando:

```
{ "found": true }
```

si utilizza la funzione per andare avanti, che fa avanzare il sistema fino a che non viene ricevuto un nuovo comando o quello di frenata:

```
{ "found": false }
```

dopo il quale viene eseguita la funzione di frenata che pone a zero la velocità dei motori e disattiva l'alimentazione.

Per eseguire queste operazioni si utilizzano le APIs fornite dalla libreria della casa produttrice del Motor Shield (vedi 2.1.2), MotorAF¹⁴. Questa permette di impostare la velocità dei motori con valori tra 0 e 255 e prevede tre modalità di movimento: FORWARD, BACKWARD e BRAKE. Per andare avanti si utilizza per entrambi i motori la modalità FORWARD; per girare a destra e a sinistra invece si pone il motore opposto al senso di rotazione in modalità FORWARD e l'altro in BACKWARD; La modalità BRAKE viene invece usata in caso di frenata.

2.6 Controllo Remoto

Il sistema è stato progettato come un dispositivo *wireless*, ovvero senza necessità di essere collegato ad un computer o ad un alimentazione per poter funzionare. È infatti dotato di batterie per essere alimentato e di adattatori

¹⁴Anche questo è un progetto open-source su GitHub:
<https://github.com/adafruit/Adafruit-Motor-Shield-library>

WiFi per potersi connettere alla rete e per permettere agli sviluppatori e utilizzatori di connettersi ad esso.

Il sistema è infatti interamente programmabile in modalità wireless e i vari script e applicativi possono essere avviati in remoto.

È stata sviluppata anche un App mobile che permette di visualizzare in tempo reale i risultati delle elaborazioni di visione computazionale, in modo da poterne stimare le prestazioni, controllare lo svolgersi delle operazioni e come strumento dimostrativo delle potenzialità del sistema. Sempre tramite la stessa App è possibile controllare i movimenti del sistema, come una specie di radiocomando, così da poterlo posizionare e muovere nello spazio e, contemporaneamente, visualizzare le immagini catturate dalla fotocamera.

2.6.1 Mobile App

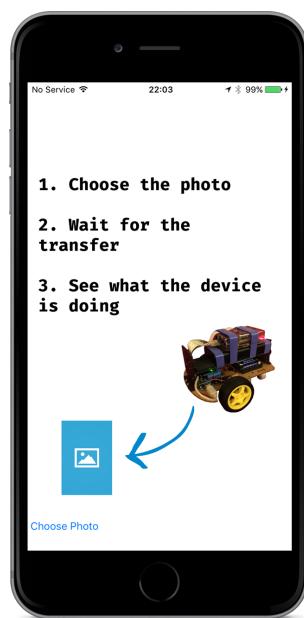


Figura 2.3: Applicazione Mobile

L'applicazione è stata creata utilizzando l'*SDK iOS*¹⁵ fornito da Apple contenuto nell'applicativo *IDE XCode*.¹⁷

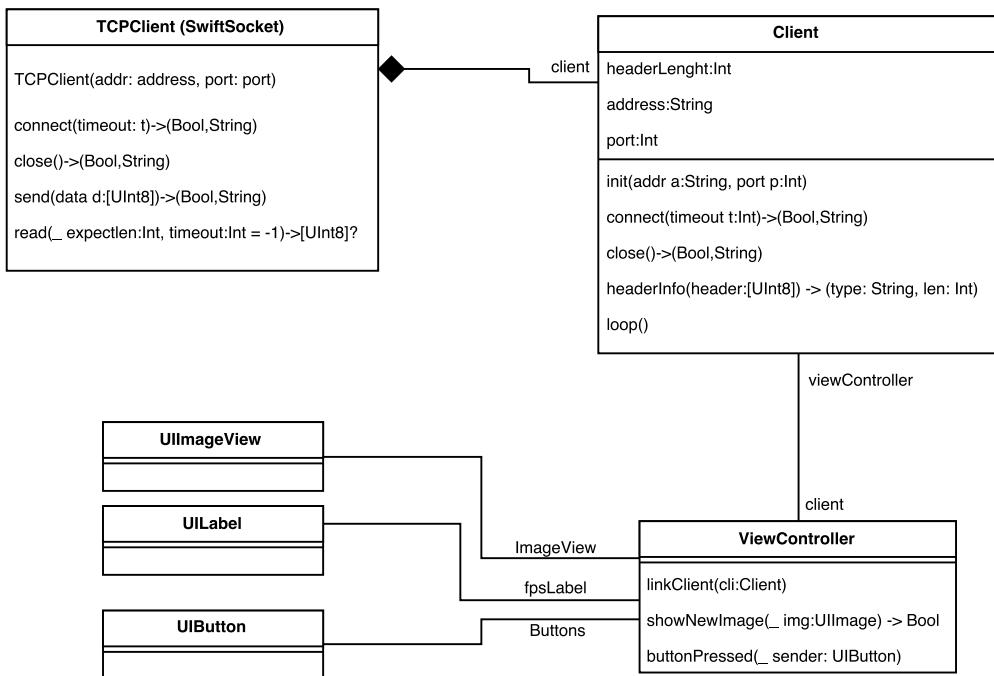


Figura 2.4: Diagramma UML che rappresenta la struttura dell'applicazione

Questa è stata sviluppata seguendo il design pattern MVC¹⁸ in modo da separare i ruoli e le responsabilità. Si presenta anche il diagramma UML in figura 2.4. L'applicazione è di tipo *multi-thread* per rendere l'esperienza utente fluida e responsiva.

Per la comunicazione tra i due host in gioco si utilizza una connessione TCP (vedi A.3), visto l'utilizzo su rete locale e la necessità di avere uno stream di dati continuo senza limitazioni dovute alle dimensioni di pacchet-

¹⁵Software Development Kit: set di strumenti software di sviluppo che permettono la creazione di app per una certa piattaforma, sia software che hardware.

¹⁶Integrated Development Environment: ambiente di sviluppo.

¹⁷Le versioni utilizzate sono: iOS 10.1.1, Swift 3.1, XCode 8.1

¹⁸Model View Controller (MVC) è un design pattern per implementare interfacce utente.

to e integrità del dato. L'applicazione implementa un client TCP (classe *Client*) che estende per composizione la classe TCPClient fornita dalla libreria open-source SwiftSocket¹⁹. La classe *Client* ha la responsabilità di gestire la connessione, di restare in ascolto in attesa di nuovi dati e di inviarne altri quando richiesto. È anche incaricata di interpretare e decodificare i dati ricevuti dai quali estrapola l'immagine ricevuta. In questa architettura rappresenta il Model.

Le operazioni eseguite dalla classe Client avvengono su un thread, avviato dal metodo `loop()` chiamato alla connessione, parallelo a quello principale su cui vive l'interfaccia grafica. Il Client procede comunicando la ricezione di un frame (decodificato, vedi 2.6.1) al thread principale, nello specifico al controller della grafica (classe `ViewController`), in modo che questo possa far sì che l'immagine, consegnata dal Client, venga mostrata su schermo tramite l'apposito oggetto istanza della classe `UIImageView` (componente della Vista).

Inoltre se l'utente decide di utilizzare i comandi remoti premendo i pulsanti predisposti, il `ViewController` raccoglie il segnale inviato dagli oggetti della vista (`UIButton`) e comunica al Client di inviare il segnale corrispondente tramite la connessione.

L'applicazione ed in generale il sistema sono stati progettati come dispositivi *zeroconf*, cioè dispositivi per cui non è necessaria una configurazione più o meno complessa ad ogni utilizzo o addirittura durante. Per questo è previsto un sistema mDNS (vedi A.7), che rende la comunicazione indipendente dagli indirizzi IP dei dispositivi, evitando quindi di doverli conoscere e configurare. Si è infatti installato su Raspberry Pi un servizio mDNS con il quale si registra il proprio indirizzo IP e si assegna un identificativo all'host

¹⁹Progetto open-source disponibile su GitHub: <https://github.com/swiftsocket/SwiftSocket>

come in un sistema DNS standard. In questo modo l'applicazione mobile e in particolare il client TCP può utilizzare e risolvere l'identificativo per ottenere l'indirizzo IP di Raspberry Pi e non deve conoscere altro (il nome locale utilizzato: *raspberrypi2.local*). Anche la piattaforma su cui viene eseguito l'applicativo mobile deve supportare la tecnologia mDNS, cosa non problematica in questo progetto poiché già presente nei sistemi Apple.

Codifica e Decodifica Immagini

Le immagini che vengono scambiate tra gli applicativi wireless sono codificate per poter essere inviate e decodificate dal sistema che riceve. La codifica scelta, per la sua leggerezza e velocità in fase di compressione e decompressione, fattori critici nelle applicazioni in tempo reale e di rete, è JPEG²⁰.

Le immagini, dopo l'elaborazione sul sistema mobile, sono codificate utilizzando una funzione messa a disposizione dalla libreria OpenCV (vedi A.2) che, data un'immagine in forma matriciale e scelta la codifica, ritorna un array di bytes che rappresenta l'immagine codificata. Questo può essere direttamente inviato tramite il socket TCP utilizzato per comunicare con l'App mobile.

L'App mobile, una volta ricevuto l'array di bytes, utilizza le APIs del *framework AVKit* presente nell'SDK iOS e decodifica l'immagine (nello specifico si utilizza la classe `UIImage`) per poi mostrarla a schermo.

²⁰Il formato di compressione JPEG (Joint Photographic Experts Group), standard ISO/IEC 10918, è stato creato nel 1992 come risultato di un processo nato nel 1986. JPEG è un comune metodo di compressione con perdita per immagini digitali.

Capitolo 3

Risultati sperimentali

I risultati ottenuti attraverso vari test prove pratiche per questo progetto sono ottimi. Il sistema riesce infatti a riconoscere e inseguire una grande varietà di oggetti. Il rilevamento è invariante a variazioni di scala e orientamento ed è robusto in presenza di rumore e variazioni di luce (sono riportati degli esempi nelle figure 3.1 ed 3.2). Il riconoscimento è efficace anche quando parte dell'oggetto è coperta o fuori dall'inquadratura.



Figura 3.1: A sinistra l'immagine di input e a destra il frame analizzato. Sono rappresentati con i colori le corrispondenze tra punti d'interesse. In questo esempio l'oggetto rilevato è di dimensioni inferiori all'immagine di input.

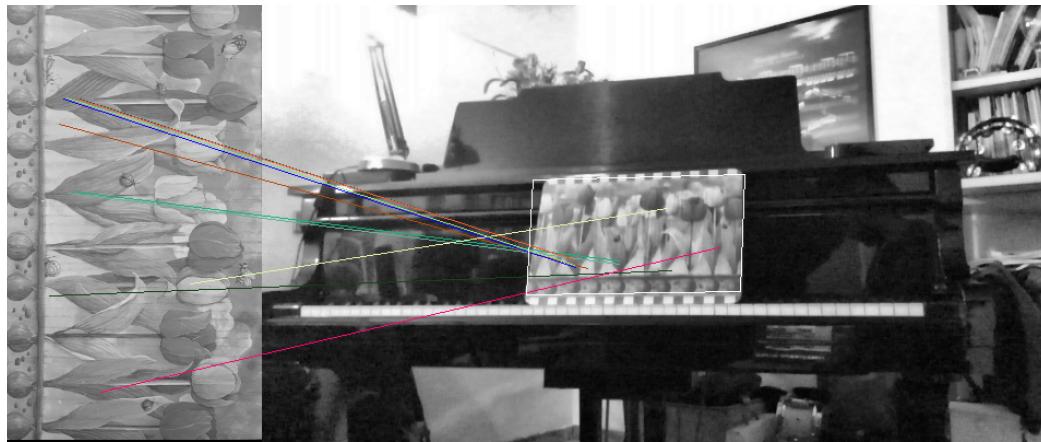


Figura 3.2: In questo esempio l’oggetto rilevato è ruotato e di dimensioni inferiori all’immagine di input

Le immagini riportate sono le stesse che il sistema invia all’App mobile per permettere all’utente di visualizzare i risultati dell’elaborazione in tempo reale.

3.1 Frame rate e risoluzione

Il sistema è in grado di elaborare 2-3 frame al secondo alla risoluzione di 640x480 pixel. Questo è un buon risultato in relazione alle limitate prestazioni del Raspberry Pi che permette comunque un’analisi abbastanza fluida e dettagliata dello spazio di fronte al sistema che si muove in maniera reattiva. L’angolo di visualizzazione della PiCamera è di 54° che permette la visione di una porzione abbastanza ampia di spazio.

3.2 Velocità di risposta

Nel migliore dei casi, se il frame viene acquisito appena l’oggetto entra a far parte dell’inquadratura, il tempo di risposta è solamente il tempo di elab-

borazione del frame che, nelle migliori ipotesi, è di circa un terzo di secondo. Nel caso peggiore (l'oggetto entra nell'inquadratura subito *dopo* l'acquisizione di un frame) il tempo di risposta si aggira intorno ad un secondo. Questo corrisponde al tempo di analisi del frame appena acquisito dove non è presente l'oggetto sommato al tempo di acquisizione ed elaborazione del frame in cui l'oggetto è presente.

Il tempo impiegato per attuare i motori da parte di Arduino è molto breve e non supera i 10 ms.

3.3 Affidabilità e precisione

Nelle immagini di test utilizzate, grazie ai metodi di riconoscimento visivo implementati, si riesce a riconoscere la presenza dell'oggetto più del 90% delle volte in cui è presente.

Nei vari set di 10 immagini di uno stesso oggetto su cui si sono effettuati i test, si è riusciti a rilevare e stimare la posizione di questo 9 volte su 10 e in qualche caso anche nella totalità delle immagini. Inoltre la stima della posizione dell'oggetto è piuttosto precisa e sufficiente per l'applicazione in questo progetto.

3.4 Risultati per approcci alternativi

Sono stati studiati e testati estensivamente anche approcci alternativi come descritto nella sezione 1.1.2. Nello specifico il tempo di elaborazione ed i costi computazionali si sono ridotti, non in maniera sostanziale, ma la precisione e l'affidabilità sono calate in maniera percepibile.

L'approccio ibrido in cui si effettua il rilevamento dei punti di interesse tramite l'algoritmo ORB (vedi 1.1.2) e la descrizione tramite l'algoritmo SURF (vedi 1.1.1) ha portato a questi risultati:

- Framerate alla risoluzione di 640x480 pixel: 3-4 fps. Solo un frame al secondo in più.
- Velocità di risposta nel caso migliore di 0.25 sec (24% più rapido)
- Affidabilità: nelle immagini di test utilizzate si riesce a rilevare l'oggetto solo 6 volte su 10 rispetto alle volte in cui l'algoritmo SURF lo rileva (si rileva l'oggetto il 40% in meno delle volte.).
- Precisione: anche nei casi in cui l'oggetto è rilevato la posizione stimata è imprecisa e a volte errata.

L'utilizzo invece del solo algoritmo ORB per entrambe le fasi di elaborazione ha portato ai seguenti risultati:

- Framerate alla risoluzione di 640x480 pixel: 4 fps costanti.
- Velocità di risposta nel caso migliore di 0.25 sec (24% più rapido)
- Affidabilità: nelle immagini di test utilizzate si riesce a rilevare l'oggetto solo 4 volte su 10 rispetto alle volte in cui l'algoritmo SURF lo rileva (si rileva l'oggetto il 40% in meno delle volte.).
- Precisione: spesso anche nei casi in cui l'oggetto è rilevato la posizione stimata è imprecisa e a volte errata. L'imprecisione è dovuta, molto spesso, al rilevamento di punti d'interesse poco robusti e caratteristici.

I risultati ottenuti non sono stati soddisfacenti e hanno portato all'abbandono di questi metodi.

3.5 Test eseguiti

Sono stati eseguiti test con diversi oggetti di varie categorie. I risultati migliori si ottengono con oggetti che presentano superfici piane non anonime e con caratteristiche particolari. I test sono stati effettuati anche in più ambienti e diverse condizioni di luce e, grazie anche all'equalizzazione dell'istogramma, il sistema riesce a sopportare variazioni di luce e il rumore presente in immagini relativamente buie.

Conclusioni

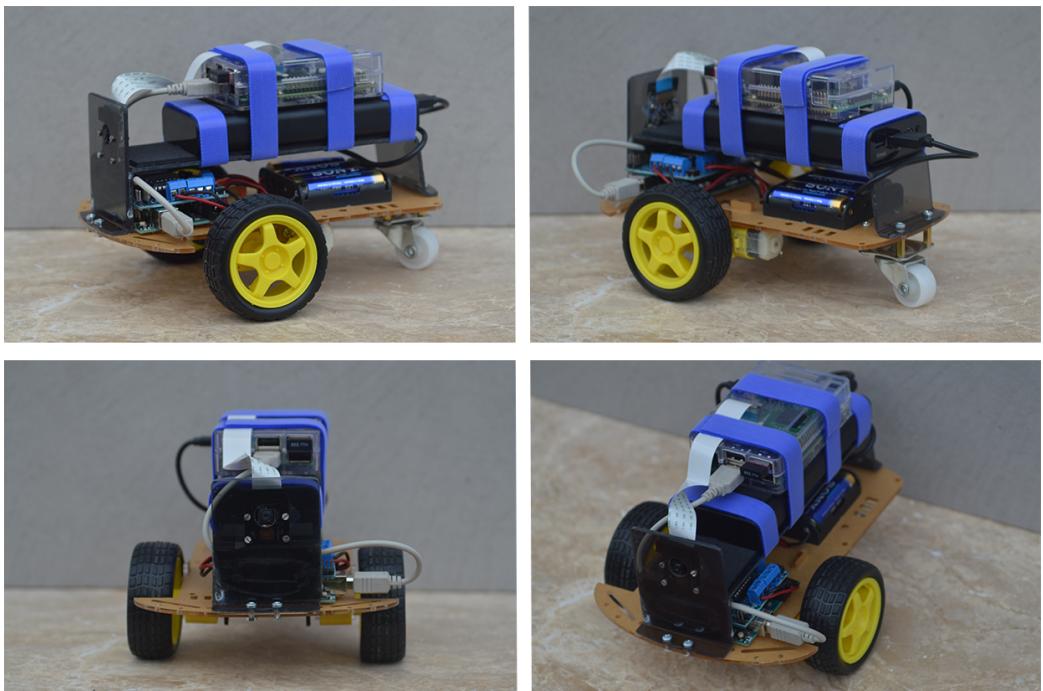


Figura 3.3: Immagini del sistema da più angolazioni.

Il sistema realizzato è solido, stabile e rappresenta una buona base per studi ed applicazioni future. Questo progetto ha coinvolto più discipline: elettronica, meccanica ed informatica che hanno permesso, attraverso la loro interazione, di rendere funzionale il sistema. Diverse tecnologie sono state utilizzate e integrate tra di loro al fine di creare una piattaforma di sviluppo per sistemi embedded, versatile e pratica per ogni sorta di utilizzo.

Grande attenzione è stata rivolta ai calcoli relativi al consumo energetico e alle ottimizzazioni che permettono l'utilizzo di piattaforme hardware ridotte.

La scelta dei materiali e delle attrezzature impiegate consente di realizzare il sistema con costi relativamente bassi e accessibili rispetto a quanto già presente sul mercato.

Realizzazione su sistemi embedded moderni

I sistemi attualmente disponibili permettono di ottenere buoni risultati con costi contenuti. In passato un progetto di queste dimensioni e complessità sarebbe stato possibile esclusivamente utilizzando prodotti professionali e specifici con costi decisamente elevati e, probabilmente, non con la stessa modalità di progettazione e realizzazione.

La velocità con cui la tecnologia si sviluppa cresce esponenzialmente, apre nuove frontiere e infinite possibilità ad oggi inimmaginabili. Nel futuro il costo della tecnologia sarà minore e renderà possibile una diffusione sempre più capillare di questi strumenti anche tra gli utenti comuni. I risultati ottenuti con i sistemi odierni potranno nel breve futuro migliorare, creando al contempo le basi per lo sviluppo di ulteriori applicazioni più complesse e funzionali.

Sviluppi Futuri

Il progetto, così come è stato strutturato, si presta come base solida per miglioramenti ed ottimizzazioni dell'applicazione attuale e per sviluppi futuri, come nuove applicazioni e progetti di ricerca.

Tecniche di riconoscimento visivo migliorate

Per migliorare le prestazioni, soprattutto nella fase di riconoscimento visivo, una direzione utile di ricerca potrebbe essere quella di studiare e testare nuovi algoritmi di estrazione di caratteristiche, che non sono stati trattati in questo elaborato, progettando eventualmente un nuovo approccio, anche ibrido, per rendere meno costose determinate operazioni senza però perdere accuratezza e precisione nel riconoscimento.

Si può anche pensare di modificare i metodi studiati, come SURF (vedi 1.1.1), in modo da concentrare l'attenzione su parti dell'algoritmo specificamente importanti per questa applicazione e ridurre i costi di sezioni dell'algoritmo di importanza solo marginale.

Sarebbe inoltre ragionevole provare ad effettuare del pre-processing dell'immagine con altri metodi di visione computazionale, cercando o di rendere l'immagine di qualità migliore e più adatta alle elaborazioni future, o di effettuare un riconoscimento sommario, basato anche su altre tecniche di riconoscimento, per ottenere una prima approssimazione della posizione dell'oggetto da cercare in modo da ridurre la regione di immagine su cui effettuare la ricerca con gli algoritmi visti nel capitolo 1.

Utilizzo GPU e computazione parallelizzata

Sarebbe importante riuscire a sfruttare completamente le piattaforme in uso come il Raspberry Pi. Al momento gli algoritmi sono eseguiti solamente su CPU e con un utilizzo non troppo estensivo di parallelizzazione.

In futuro potrebbe essere studiato un sistema per permettere alla libreria utilizzata (OpenCV, vedi A.2) di sfruttare la GPU, vista anche la sua predisposizione (è già possibile e disponibile in fase di compilazione aggiungere il

supporto a GPU per chipset Nvidia¹ - *CUDA drivers*).

Inoltre potrebbero essere implementate estensivamente tecniche di computazione parallelizzata in modo da sfruttare a pieno la CPU del Raspberry Pi, essendo molte parti degli algoritmi di riconoscimento adatte a questo approccio.

App multipiattaforma

In modo da consentire l'utilizzo a tutti, a prescindere dal dispositivo utilizzato, l'applicazione per il controllo remoto potrebbe essere resa multipiattaforma: sia per dispositivi mobili che desktop.

Nuove tecnologie

In futuro potranno essere integrate nuove tecnologie in sostituzione e aggiunta alle presenti per migliorare le prestazioni ed i risultati ottenuti, magari riducendo i costi e le dimensioni.

¹Casa produttrice di GPU

Appendice A

Tecnologie utilizzate

A.1 Python

Python è un linguaggio di programmazione ad alto livello, orientato agli oggetti, adatto tra gli altri usi per sviluppare applicazioni distribuite, scripting, computazione numerica e system testing.

Fu ideato da Guido van Rossum all'inizio degli anni novanta. Il nome fu scelto per via della passione di van Rossum per i Monty Python e per la loro serie televisiva Monty Python's Flying Circus.

Python è un linguaggio multi-paradigma che fa della dinamicità, semplicità e flessibilità i suoi principali obiettivi. 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 dei blocchi. 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.

Il controllo dei tipi è comunque forte (strong typing) e viene eseguito a runtime (dynamic typing). In altre parole una variabile è un contenitore al quale è associata un’etichetta (il nome) che può essere abbinata a diversi contenitori anche di tipo diverso durante il suo tempo di vita. Usa un garbage collector per la liberazione automatica della memoria.

I suoi progettisti hanno scelto una sintassi più essenziale e uniforme, con l’obiettivo di aumentare la leggibilità del codice.

È spesso classificato come linguaggio di scripting, ma pur essendo utile per scrivere script di sistema (in alternativa ad esempio a bash), la grande quantità di librerie disponibili e la facilità con cui questo linguaggio permette di scrivere software modulare, favoriscono anche lo sviluppo di applicazioni molto complesse.[11]

A.2 OpenCV

OpenCV (Open Source Computer Vision) è una libreria di funzioni di programmazione il cui scopo principale è la visione computazionale per applicazioni real-time.[12]

Originariamente sviluppata dal centro di ricerca *Intel* di Nižnij Novgorod (Russia), è stata successivamente supportata da *Willow Garage* ed è adesso mantenuta da *Itseez*.[13]

La libreria è multipliattaforma e gratuita sia per utilizzi accademici che commerciali, rilasciata sotto la licenza open-source BSD. Ha interfacce C++, C, Python (quella utilizzata nel progetto di Tesi) e Java. Supporta Windows, Linux, MacOS, iOS e Android.

OpenCV è stato progettato per l’efficienza computazionale e con un grande interesse per le applicazioni real-time. Scritto in C/C++ ottimizzato la li-

breria si avvale di elaborazione multi-core. Con il supporto a OpenCL, può sfruttare l'accelerazione hardware di strutture computazionali eterogenee. OpenCV, adottato in tutto il mondo, ha più di 47 mila persone nella sua comunità di utenti ed un numero di downloads stimati che supera i 9 milioni. Gli utilizzi possibili spaziano dalle arti interattive, guida automatica e supporti per ipovedenti.[14]

Storia

Lanciato ufficialmente nel 1999, il progetto OpenCV era alla nascita un'iniziativa della Ricerca Intel per applicazioni CPU-intensive, parte di una serie di progetti che includevano il tracciamento real-time e schermi 3D. I principali contributori del progetto includevano alcuni esperti di ottimizzazione di *Intel Russia* ed il team della *Performance Library* di Intel.

Agli esordi di OpenCV, gli obiettivi del progetto erano descritti come[15]:

- Avanzare la ricerca fornendo non solo codice *Open* ma anche ottimizzato, per infrastrutture di visione computazionale.
- Disseminare la conoscenza della visione fornendo un'infrastruttura comune su cui gli sviluppatori possano costruire sopra, in modo che il codice sia più leggibile e trasportabile.
- Avanzare applicazioni commerciali basate sulla visione computazionale creando codice trasportabile e dalle performance ottimizzate, disponibile gratuitamente.

La prima versione *alpha* di OpenCV è stata rilasciata al pubblico alla "Conferenza IEEE sulla Visione Computazionale e Riconoscimento di Pattern" nel 2000. La prima versione - 1.0 - è stata rilasciata nel 2006. Nel 2008, OpenCV ha ottenuto il supporto aziendale di *Willow Garage* e è stata di nuovo in una fase attiva di sviluppo. Una "pre-release" della versione 1.1 è

stata rilasciata nell’ottobre del 2008.

La seconda release ufficiale è stata nell’ottobre del 2009. OpenCV 2 include modifiche importanti all’interfaccia C++ con l’obiettivo di nuove funzioni, più semplici e design patterns type-safe che risultano in migliori implementazioni di quelle esistenti in termini di performance.

Le release ufficiali attualmente vengono rilasciate ogni sei mesi. Nell’Agosto del 2012 il supporto di OpenCV è stato consegnato ad una organizzazione non-profit chiamata OpenCV.org che mantiene il sito web per utenti e sviluppatori.[16]

A.3 TCP Socket

I Sockets sono la tecnologia fondamentale per programmare software che sia in grado di comunicare in reti TCP/IP. Un socket indica un’astrazione software progettata per poter utilizzare delle API standard e condivise per la trasmissione e la ricezione di dati attraverso una rete, rappresenta un punto di accesso ad una singola connessione tra due applicazioni di rete e permette una comunicazione bidirezionale per scambiare dati con un altro socket.

Più precisamente un Socket è uno strumento che un programma locale può passare all’interfaccia di rete (*Network APIs*) per usare la connessione associata. Una socket API è di solito fornita dal sistema operativo e permette ai programmi di controllare ed utilizzare i socket di rete.

I socket sono largamente diffusi ed utilizzati fin dai primi anni ottanta. Furono introdotti dapprima in una distribuzione BSD (*Berkley Software Distribution*) nel 1983 e poi adottati da tutti gli altri sistemi operativi, per questo motivo solitamente le funzioni di programmazione dei socket vengono chiamate Berkeley socket API.[17]

Le connessioni socket normalmente vengono utilizzate tra due computer differenti su una rete locale o su internet, ma possono anche essere utilizzate per comunicazioni *inter-processo* su un singolo computer.[18][19]

I socket di rete vengono identificati in maniera univoca tramite un *ID* e ad ognuno di essi è associato un *Socket Address* (con un'operazione detta di *Binding*) che è composto da un indirizzo IP e da una porta. Basandosi su questo indirizzo i socket di rete consegnano i pacchetti di dati scambiati alla corretta applicazione, processo o thread.

A.4 iOS - Swift

iOS

iOS (una volta chiamato iPhone OS) è un sistema operativo mobile creato e sviluppato da Apple Inc. esclusivamente per il suo hardware.

Originariamente svelato nel 2007, durante il WWDC, insieme all'*iPhone* ha successivamente esteso il supporto ad altri dispositivi. È ad oggi il sistema operativo che viene eseguito da molti dei dispositivi Apple, come *iPhone*, *iPad*, *iPod touch*, e *Apple Watch*.

Nativamente è presente uno store (*App Store*) tramite il quale è possibile scaricare ed installare una vasta gamma di applicazioni che vengono mantenute, aggiornate e messe in sicurezza in maniera automatica.

Parallelamente è stato sviluppato un kit di sviluppo nativo iOS SDK (*SDK - Software Developement Kit*) grazie al quale gli sviluppatori software possono creare nuove applicazioni e caricarle sullo store.

Il linguaggio nativo di questo ambiente di sviluppo è *Swift*, che insieme ai Framework forniti diventa uno strumento di sviluppo completo per queste

piattaforme. È però possibile utilizzare molti altri linguaggi che vengono supportati nativamente, come ad esempio C, C++ ed Objective-C.

Swift

Swift è un linguaggio di programmazione ad alte performance, *general-purpose* e compilato, sviluppato da Apple Inc. È stato progettato per lavorare con i Framework *Cocoa* e *Cocoa Touch* di Apple e per supportare codice C e Objective-C.

Swift ha una sintassi moderna e pulita, con l'obiettivo di essere più resiliente agli errori ed essere più sicuro.

Anche se ispirato da Objective-C e molti altri linguaggi, non è un linguaggio derivato da C. È un nuovo e completo linguaggio indipendente da altri, con funzionalità come controllo di flusso, strutture dati e funzioni con costrutti di alto livello come oggetti, protocolli, *closures*, e *generics*.

Swift è un linguaggio *Open-Source* dalla sua versione 2.0 ed è quindi sviluppato sia all'interno di Apple che da un grande numero di sviluppatori autonomi.

Attualmente è stata rilasciata la versione 3.0 e si continua lo sviluppo per future versioni. È uno dei linguaggi con più alti tassi di crescita nell'utilizzo ed è comunque uno dei più diffusi ed utilizzati, soprattutto nel settore mobile.

A.5 JSON

JSON (*JavaScript Object Notation*) è un popolare formato, semplice e leggero, utilizzato per lo scambio di oggetti di dati tra applicativi anche di tipo differente. È un formato standard aperto che usa testo, semplice da leggere e scrivere per gli umani e facile da generare ed interpretare per i com-

puter.[20]

Si basa su un sottoinsieme del linguaggio di programmazione *JavaScript*, definito nelle specifiche ECMA-262, 3a edizione, del dicembre 1999.[21]

JSON è un formato di testo completamente indipendente dal linguaggio ma che usa convenzioni già familiari ai programmatore dei linguaggi derivati dal C, tra cui C, C++, C#, Java, JavaScript, Perl, Python e molti altri. Queste caratteristiche rendono JSON un linguaggio ideale per lo scambio di dati.[22] È infatti molto popolare in applicazioni Web, essendo una valida alternativa al formato XML-XSLT, e ben integrato nei sistemi AJAX. JSON deve essere servito con il content type impostato su application/json come definito in [23]

Sintassi

JSON prende origine dalla sintassi degli oggetti letterali in JavaScript.

Un oggetto letterale può essere definito così:

```
var JSON = {  
    proprietA1: 'Valore',  
    proprietA2: 'Valore',  
    proprietAN: 'Valore'  
}
```

Si tratta di coppie di proprietà/valori separate dalla virgola ad eccezione dell'ultima. L'intero oggetto viene racchiuso tra parentesi graffe. A differenza di questa notazione JavaScript, che può contenere anche funzioni e valori complessi, JSON ammette solo valori semplici ed atomici, tra cui:

- stringhe
- numeri

- booleani : true, false
- null
- array
- oggetti letterali

JSON inizia con una coppia di parentesi graffe che racchiudono il corpo dell'intera struttura. Seguono poi le coppie di proprietà e valori che, come in questo esempio, possono contenere i valori atomici elencati sopra. A differenza degli oggetti letterali, JSON richiede che i nomi delle proprietà e i valori stringa siano racchiusi tra doppie virgolette. Riportiamo di seguito un esempio di oggetto JSON:

```
{  
    "coordinates": {  
        "x": 0.5,  
        "y": 0.3  
    },  
    "found": true  
}
```

A.6 Comunicazione Seriale

La comunicazione seriale è utilizzata per la comunicazione tra una scheda Arduino ed un computer o altri dispositivi.

La comunicazione seriale sui pin TX/RX usa i livelli logici TTL (5V o 3.3V a seconda).

Tutte le schede Arduino posseggono almeno una porta seriale (anche conosciuta come UART o USART). Comunica con i pin digitali 0 (RX) e 1 (TX) e anche con un computer via USB. Per la comunicazione i due terminali de-

vono essere in accordo sulla frequenza di trasmissione dati (*Baud Rate*) con la quale impostano le velocità di campionamento e di scrittura del segnale.

A.7 Multicast DNS

Con il ridursi delle dimensioni dei dispositivi network che diventano sempre più trasportabili e sempre più presenti, l'abilità di operare con infrastrutture meno configurate diventa sempre più importante.

Multicast DNS (mDNS) fornisce l'abilità di effettuare operazioni simili a quelle di un DNS su una rete locale nell'assenza di un convenzionale server Unicast DNS. Inoltre Multicast DNS designa una porzione dello spazio dei nomi per essere libera per l'utilizzo locale, senza bisogno di configurare delegazioni o server DNS convenzionali per rispondere per quei nomi.

I benefici primari dei nomi Multicast DNS sono che non richiedono configurazione o amministrazione, funzionano anche quando nessuna infrastruttura è presente, e funzionano durante un possibile fallimento delle infrastrutture DNS.[24]

Il Multicast Domain Name System (mDNS) risolve gli *hostnames* in indirizzi IP. È un servizio *zero-configuration* (che non richiede configurazione) usando essenzialmente le stesse interfacce di programmazione, formati pacchetti e semantica di un normale unicast Domain Name System (DNS).

La porzione di spazio dei nomi che viene riservata a questo servizio è quella terminante per *.local* .

Il servizio si avvale di una fase di *Discovery* (DNS-SD) durante la quale si invia un pacchetto DNS attraverso UDP ad un certo indirizzo multicast. Tutti gli host in grado di utilizzare mDNS ascoltano su questo indirizzo. Utilizzando UDP si ottiene un basso *overhead*, inoltre i client sono progettati

per ridurre al minimo il traffico sulla rete, utilizzando estensivamente una memoria di *cache*. Il servizio di Discovery è un processo in due fasi:

- Trovare i nomi di tutti gli host che forniscono un determinato servizio (ad esempio stampare). Questo non fornirà ancora gli indirizzi IP ma solo i nomi mDNS (che termineranno con *.local*).
- Risolvere i nomi *.local* degli host con mDNS, chiedendo tramite multi-cast chi è che possiede il nome da risolvere. L'host corrispondente vedrà il pacchetto di richiesta, e risponderà in Broadcast con il suo indirizzo IP, numero di porta ed altre informazioni.

Breve storia dello sviluppo

mDNS è un progetto che è stato ideato nel Luglio del 1997 come successore di "AppleTalk Name Binding Protocol" (NBP) e che ha visto una sua prima implementazione e distribuzione nel 2001 con un aggiornamento di Mac OS 9. Il progetto è stato poi reso open-source nel 2002 ed è tutt'ora sviluppato ed aggiornato.[24]

Bibliografia

- [1] BMVA. *What is computer vision?* URL: <http://www.bmva.org/visionoverview> (cit. a p. 1).
- [2] Wikipedia. *Feature Detection (Computer Vision)*. 2016. URL: [https://en.wikipedia.org/wiki/Feature_detection_\(computer_vision\)](https://en.wikipedia.org/wiki/Feature_detection_(computer_vision)) (cit. a p. 2).
- [3] Mark S. Nixon; Alberto S. Aguado. *Feature Extraction and Image Processing*. A cura di Academic Press. Second edition. Elsevier, 2008. (Cit. a p. 3).
- [4] OpenCV Documentation. *Introduction to SURF (Speeded-Up Robust Features)*. 2016. URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html (cit. alle pp. 3, 11).
- [5] P.A. Viola; M.J. Jones. “Rapid object detection using a boosted cascade of simple features”. *CVPR*, pp. 511–518, 2001. (Cit. a p. 4).
- [6] Herbert Bay; Tinne Tuytelaars; e Luc Van Gool. “SURF: Speeded Up Robust Features”. *Computer Vision and Image Understanding*, 2006. (Cit. alle pp. 4–8, 10, 11).
- [7] A. Neubeck; L. Van Gool. “Efficient non-maximum suppression”. *ICPR*, 2006. (Cit. a p. 8).

- [8] Ethan Rublee; Vincent Rabaud; Kurt Konolige; Gary R. Bradski. “ORB: An efficient alternative to SIFT or SURF”. 2011. (Cit. a p. 12).
- [9] OpenCV. *ORB (Oriented FAST and Rotated BRIEF)*. 2015. URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html (cit. a p. 13).
- [10] Universita di Pisa. *Laurea Magistrale in Sistemi Embedded*. URL: <http://ce.iet.unipi.it/index.php/it/mecs> (cit. a p. 18).
- [11] Wikipedia. *Python*. 2016. URL: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)) (cit. a p. 45).
- [12] Pulli Kari; Baksheey Anatoly; Kornyakov Kirill; Eruhimov Victor. *Real-time Computer Vision with OpenCV*. 2012. URL: <http://dl.acm.org/citation.cfm?id=2206309> (cit. a p. 45).
- [13] Itseez. *Itseez*. 2016. URL: <http://itseez.com/> (cit. a p. 45).
- [14] Itseez. *OpenCV website*. 2016. URL: <http://opencv.org> (cit. a p. 46).
- [15] Adrian Bradski Gary; Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. A cura di Inc. O'Reilly Media. 2008. P. 6. (Cit. a p. 46).
- [16] Wikipedia. *OpenCV*. 2016. URL: <https://en.wikipedia.org/wiki/OpenCV> (cit. a p. 47).
- [17] Wikipedia. *Network socket*. 2016. URL: https://en.wikipedia.org/wiki/Network_socket (cit. a p. 47).
- [18] Bradley Mitchell. *Socket Programming for Computer Networks*. 2016. URL: <https://www.lifewire.com/definition-of-socket-817934> (cit. a p. 48).

- [19] RT; EJP; Galwegian. *What is the difference between a port and a socket?* 2012. URL: <http://stackoverflow.com/questions/152457/what-is-the-difference-between-a-port-and-a-socket> (cit. a p. 48).
- [20] Wikipedia. *JSON*. 2016. URL: <https://en.wikipedia.org/wiki/JSON> (cit. a p. 50).
- [21] *ECMA-262, 3a edizione, del dicembre 1999.* 1999. (Cit. a p. 50).
- [22] JSON Organization. *JSON*. 2000. URL: <http://www.json.org> (cit. a p. 50).
- [23] D. Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*. Giu. 2006. (Cit. a p. 50).
- [24] S. Cheshire; M. Krochmal; Apple Inc. *Multicast DNS*. Internet Engineering Task Force (IETF), feb. 2013. (Cit. alle pp. 52, 53).