UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

Rilevamento della posizione mediante codici QR

Elaborato di esame di:

Zaghen Nicola, Zampieri Alex

Consegnato il: **7 Giugno 2013**

Sommario

Dato un flusso continuo di immagini provenienti da una webcam, si vuole individuare la presenza di un codice QR e, in caso di rilevamento, stimare la posizione dell'osservatore rispetto al codice QR, cioè stimare la distanza e l'angolo formato dal codice QR con l'osservatore. Unendo queste informazioni con quelle codificate all'interno del codice QR si localizza l'osservatore.

1. Introduzione

Il lavoro si fonda sulla rilevazione di codici QR in un flusso video da webcam. È stato quindi necessario effettuare una localizzazione robusta e veloce dei codici QR e integrare il sistema di rilevamento con un sistema per la gestione del flusso video da webcam. Dal momento che i requisiti di progetto non sono stati in principio fissati, si è deciso di appoggiarsi su di una libreria esistente al fine di modificarla per renderla più aderente per gli scopi del progetto. Una prima fase del lavoro ha pertanto previsto una ricerca individuale al fine di capire da un lato la struttura e le modalità di utilizzo del codice QR e, dall'altro, la scelta della libreria di decodifica più adeguata.

1.1. Il codice QR

Il codice QR è una rappresentazione matriciale di dati con le seguenti caratteristiche:

1. I caratteri possibili, che possono essere numerici (cifre da 0-9), alfanumerici (cifre, lettere maiuscole e minuscole, caratteri speciali), Shift-JIS 8-bit Latin e Kana o lettere Kanji;
2. La rappresentazione dei dati, che prevede un modulo nero per l'uno binario e un modulo bianco per lo zero binario;
3. La dimensione dei simboli rappresentati, che prevede 40 tipologie di QR, ognuna con l'incremento di 4 moduli rispetto alla precedente, a partire da 21 moduli fino ad arrivare a 177;
4. I dati contenuti per simbolo, che nel caso di interesse per il progetto prevedono 7089 caratteri numerici o 4296 caratteri alfanumerici;
5. La percentuale di correzione degli errori applicata, che prevede 4 livelli (L, M, N, Q) da un minimo del 7% ad un massimo del 30%;
6. L'indipendenza dall'orientamento spaziale del QR.

Fig. 1 - Un codice QR versione 1.

Ogni codice QR consiste di moduli quadrati posti in un array regolare di forma quadrata e può essere suddiviso in una regione di codifica (encoding region) e in pattern funzionali (function pattern). Questi ultimi sono chiamati: “position detection patterns”, “separators”, “timing patterns” e “alignment patterns”. I pattern funzionali non possono essere usati per codificare dati. Inoltre il simbolo deve esser circondato su tutti e quattro i lati da un bordo detto “quiet zone”.

Per visualizzare quanto appena descritto è possibile osservare la Figura 2, che mostra un QR di tipo 7.

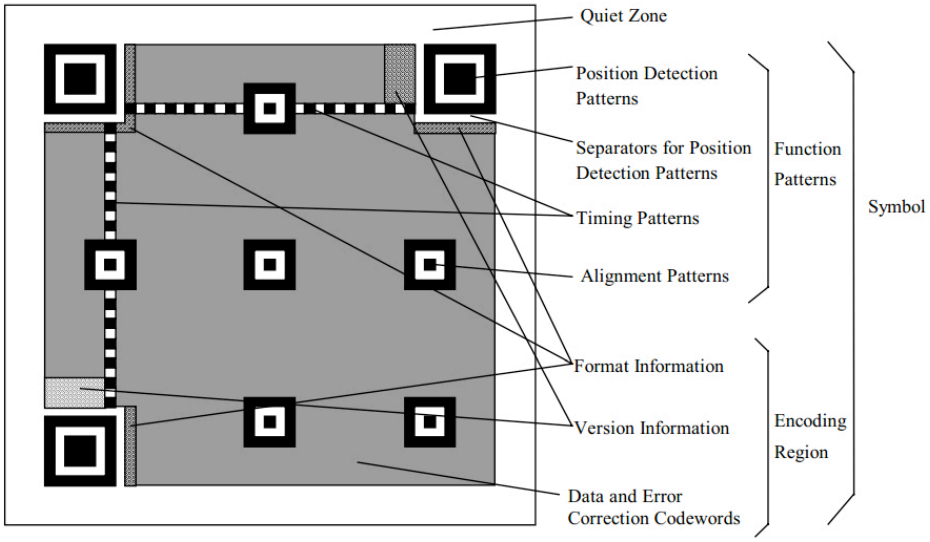


Fig. 2 - I moduli all’interno di un codice QR.

Come già anticipato esistono 40 dimensioni di codice QR, a cui si da il nome di “Versione 1”, “Versione 2”…”Versione 40”. Con riferimento alla struttura quadrata il QR versione 1 misura 21x21 moduli, mentre il QR versione 40 misura 177x177 moduli. Nell’immagine seguente è possibile osservare le differenze che caratterizzano ogni modulo.

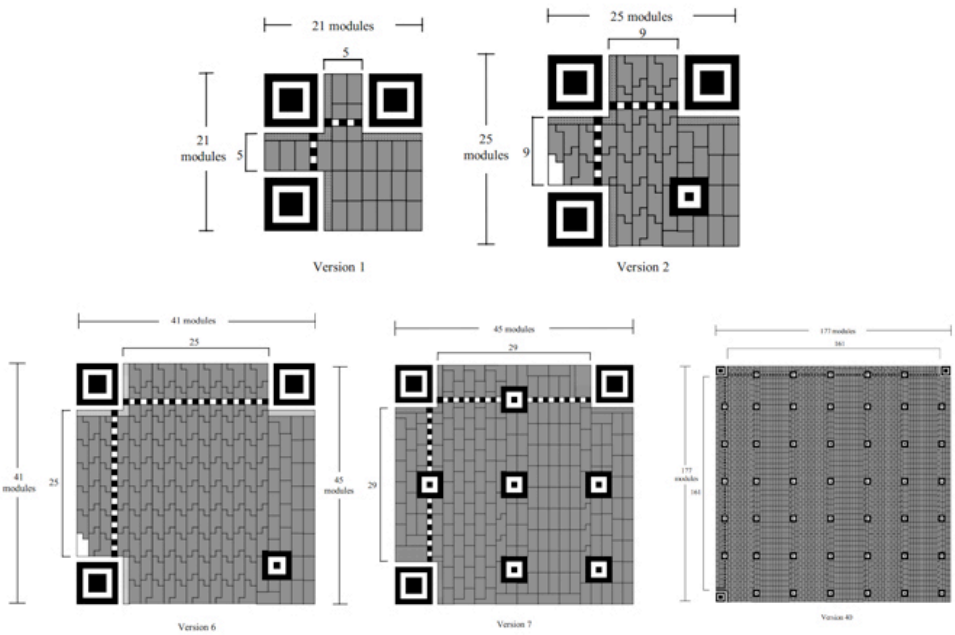


Fig. 3 - Struttura delle principali versioni dei codici QR

Verranno ora brevemente descritti i pattern utilizzati per il riconoscimento:

1. Finder pattern: è formato da tre Position Detection Pattern, posizionati nell'angolo alto sinistro, alto destro e basso sinistro. Come è possibile vedere dalla Figura 3, i quadrati alternati (bianchi e neri) e concentrici di cui è costituito questo pattern variano la loro dimensione secondo un rapporto fisso dipendente dalla versione del QR (1:1:3:1 moduli). È banale osservare che il riconoscimento di questi pattern consenta di ottenere la posizione e l'orientamento del simbolo nel campo visivo, è pertanto un'informazione vitale per il processo di localizzazione oggetto di questo lavoro.
2. Separators: un separatore della dimensione di un modulo è posto tra il Position Detection Pattern e l'Encoding Region. Esso è composto solamente da moduli bianchi.
3. Timing pattern: questi moduli, verticali e orizzontali, consistono in una riga o colonna di moduli alternati bianchi e neri, il cui inizio e terminazione sono individuati da un modulo di colore nero. Essi si trovano tra i Separators per i Position Pattern superiore (orizzontale) e sinistro (verticale) e consentono di controllare la densità dei simboli oltre che fornire dei dati corretti per stabilire le coordinate spaziali del modulo.
4. Alignment pattern: ognuno di questi pattern è costituito da 3 quadrati concentrici e alternati nel colore il cui numero nel QR varia in dipendenza dalla versione. Esso deve essere di norma presente in tutti i QR dalla versione 2.
5. Encoding region: quest'area contiene la rappresentazione in simboli dei dati insieme al codice di correzione degli errori e alle informazioni sulla versione e sul formato.
6. Quiet Zone: Questa regione circonda il simbolo da tutti e 4 i lati, ha una dimensione fissa e deve essere necessariamente dello stesso colore dei moduli bianchi oltre che essere priva di qualsiasi simbolo.

I dati di interesse per il progetto provengono dalle coordinate spaziali dei Finder Pattern, la cui posizione consente di effettuare i calcoli necessari per determinare il centro del QR e la distanza dall'osservatore. Noto questo problema, la ricerca della libreria di decodifica ha previsto il soddisfacimento di questi vincoli:

1. Un buon grado di riconoscimento del codice QR almeno su immagini statiche;
2. L'indipendenza della libreria dalla versione del QR;
3. La disponibilità della libreria in linguaggio C o C++;
4. Una buona leggibilità e semplicità del codice, per gli interventi di modifica da effettuare.

1.2. La scelta della libreria di decodifica

Dopo una prima ricerca, la scelta è ricaduta su queste due librerie:

1. ZXing, una libreria open source, multi-formato 1D/2D, sviluppata in Java, ma con porting in C++;
2. Quirc, una libreria scritta in C sviluppata come progetto indipendente da Daniel Beer, il cui sorgente è disponibile su GitHub.

La libreria Quirc presenta fin da subito un problema per la decodifica di payload numerici (ad esempio fornisce come risultato "1" quando nel codice è contenuto "0"), si decide quindi di integrare la libreria OpenCV per la gestione del flusso da webcam con ZXing.
In fase di test questa si è però rivelata molto sensibile alla rotazione sull'asse verticale del QR, che se non perfettamente allineato verticalmente, fa calare di molto le possibilità di rilevamento del codice QR. Sotto consiglio del docente si procede dunque ad inviare una segnalazione di bug allo sviluppatore di Quirc. A questo fanno seguito alcune correzioni che consentono alla libreria di operare in maniera corretta.

L'integrazione con OpenCV viene quindi trasportata, con le necessarie modifiche, in questa libreria, notando fin da subito prestazioni migliori rispetto a ZXing.

Nel capitolo seguente descriviamo pertanto l'approccio al problema della localizzazione e mostriamo i risultati ottenuti.

1.3. OpenCV

Per la gestione del flusso video dalla webcam "Crystal Eye" integrata nel netbook in dotazione, si è utilizzata la libreria OpenCV nella versione 2.3.1. Le funzioni della libreria sono state utilizzate per effettuare la calibrazione della webcam, attraverso il riconoscimento di un pattern predefinito ed il salvataggio dei parametri, e per acquisire e transcodificare i quadri necessari per l'elaborazione di Quirc.

1.4. Client/Server

Il programma di acquisizione ed elaborazione è stato poi reso server in modo da poter acquisire le informazioni in remoto tramite protocollo TCP sulla porta 9930. Il server accetta le connessioni in ingresso dal client grazie ad un thread dedicato alla gestione delle connessioni ed invia un pacchetto contenente le caratteristiche dell'ultimo codice QR identificato, complete del messaggio ivi contenuto e dei dati temporali dall'ultima elaborazione. Il client, che conosce la struttura dati definita dal server e inviata sul canale, effettua la stampa a terminale degli stessi.

1.5. GitHub

Per la gestione del progetto è stato creato un repository online sulla piattaforma web GitHub raggiungibile all'indirizzo https://github.com/. Questa piattaforma offre, attraverso una registrazione libera, la possibilità di gestire progetti open source, fornendo opzioni per rendere pubblico il progetto, clonare progetti altrui ed effettuare il controllo di versione mediante "git" del proprio codice. Questo servizio è stato utile per poter importare velocemente le correzioni ai bug riscontrati nella libreria Quirc, presente sulla stessa piattaforma, e per poter effettuare codifica e rimozione di problemi in modo veloce sui diversi calcolatori utilizzati per il progetto.

1.6. Hardware

Il progetto è stato sviluppato per comodità su una macchina Apple MacBook dotato di processore Intel Core 2 Duo 2.4GHz, 2 GB di RAM e webcam iSight 640x480pixel. In seguito il codice è stato portato sul netbook Acer Aspire One, dotato di processore intel Atom 1.60GHz, 512MB di RAM e webcam Crystal Eye 0.3 Mpixel.

1.7. Software

Il MacBook ha installato il sistema operativo OS X nella versione 10.7.

Sul netbook Acer Aspire One si trova installato Linux, con distribuzione Debian e kernel 2.6.32.5-686 (Squeeze).

Il software è stato sviluppato in C++ per la parte di calibrazione della webcam, acquisizione del flusso video e decodifica del QR, mentre il programma client che effettua la connessione al server di decodifica e la stampa dei dati del codice QR identificato dal programma server è stato realizzato in C.

1.8. Precondizioni sull'ambiente

I test sul software sono stati condotti in condizioni di luce diurna o di illuminazione da interni (lampada al neon). La ridotta portata della webcam integrata del netbook Acer Aspire One, che acquisisce immagini alla risoluzione di 320x240 pixel consentono di decodificare correttamente un QR di dimensione 150x150mm ad una distanza di circa 1 metro. Le prestazioni e la qualità della decodifica decadono molto velocemente al diminuire dell'illuminazione.

2. Il problema affrontato

Il lavoro prevede il raggiungimento di due obiettivi: la localizzazione della posizione di un QR nell'immagine e l'estrazione delle informazioni di distanza e rotazione prospettica del QR rispetto all'osservatore.

Per aumentare le possibilità di riconoscimento del codice QR nelle immagini è inoltre necessario calibrare la webcam per correggere le aberrazioni geometriche introdotte dal sistema di lenti della webcam.

2.1. Individuazione del codice QR

L'individuazione del codice QR in un'immagine verrà fatto mediante l'utilizzo di una libreria sviluppata appositamente per il riconoscimento e la decodifica di codici QR. Le libreria fornirà come risultato del processo di individuazione del codice QR la posizione del codice stesso ed il messaggio in esso contenuto.

2.2. Estrazione delle informazioni

Questa parte prevede di estrarre le informazioni sulla posizione del codice QR e la sua rotazione prospettica rispetto all'osservatore. Queste due informazioni non sono indipendenti l'una dall'altra, infatti per calcolare la rotazione prospettica è necessario avere la distanza dall'osservatore del lato sinistro e del lato destro del codice QR.

2.3. Invio delle informazioni sulla rete

Un client può collegarsi al server di localizzazione e richiedere la posizione dell'ultimo codice QR individuato dalla scansione. In questo caso il server deve rispondere alla richiesta inviando al client le informazioni estrapolate dell'ultimo codice QR rilevato.

3. La soluzione adottata

Il problema può essere diviso in modo naturale in tre diverse componenti:

- Calibrazione. Questa fase è necessaria ridurre le aberrazioni geometriche presenti nell'immagine acquisita. Inoltre grazie alla calibrazione è possibile, data la dimensione reale dei codici QR, il rapporto di scala tra pixel e millimetri per la webcam calibrata;
- Server di localizzazione. Questa componente si occupa dell'analisi continua del flusso di immagini provenienti dalla webcam. Ogni volta che viene rilevato un codice QR le informazioni contenute in esso vengono salvate in un'apposita struttura dati: ottenuta la posizione del codice QR nell'immagine, ricavata grazie alla libreria utilizzata, verranno usate le informazioni di calibrazione riguardo al rapporto di scala e la dimensione del codice QR per identificare la distanza del codice QR dall'osservatore e la rotazione prospettica del codice QR.

  Il server ha inoltre la possibilità di accettare connessioni TCP, ad ogni richiesta effettuata dai client collegati verranno inviate le informazioni riguardo all'ultimo codice QR rilevato.
- Client. Questa componente si occupa di instaurare una connessione TCP con il server e di mostrare a video le informazioni ottenute riguardo all'ultimo codice QR rilevato dal server.

3.1. Calibrazione

In questa componente si effettua la calibrazione della webcam per ridurre la distorsione causata dal suo sistema di lenti e per determinare il fattore di scala che permetterà al server di localizzazione di ottenere la distanza dai codici QR rispetto all'osservatore.

Il software di calibrazione è stato preso sorgente dal progetto OpenCV e modificato in modo da aggiungere la porzione di calibrazione necessaria per la stima della distanza dei codice QR.

Il software ha al suo interno un automa a stati finiti per rappresentare gli stati della calibrazione, la transizione da uno stato all'altro è in parte affidata alla pressione di un tasto da parte dell'utente e in parte data dalle informazioni acquisite mediante la webcam. In tutti gli stati dell'automa, l'applicativo visualizza il flusso continuo di immagini provenienti dalla webcam.

Gli stati presenti nell'automa sono i seguenti:

- "Detection", in questo stato l'applicativo offre all'utente la possibilità di iniziare la calibrazione premendo il tasto 'g'. Una volta premuto il tasto 'g' lo stato dell'automa passa a "Capturing";
- "Capturing", in questo stato l'applicativo acquisisce un numero predeterminato di immagini del pattern a scacchiera necessario per la calibrazione. Quando il numero di immagini acquisite è sufficiente, lo stato dell'automa passa a "Calibrated";
- "Calibrated", in questo stato l'applicativo offre all'utente la possibilità di iniziare la parte di calibrazione relativa al codice QR, anche stavolta premendo il tasto 'g'. Una volta premuto il tasto 'g' lo stato dell'automa passa a "QR Calibration";
- "QR Calibration", in questa fase l'applicativo ricerca nell'immagine un codice QR, nel momento in cui è possibile rilevare e decodificare un codice QR nell'immagine il programma immagazzina le informazioni necessarie (dimensione in pixel dei diversi lati) e porta l'automa nella fase successiva, "QR Calibrated";
- "QR Calibrated", in questa fase l'applicativo invita l'utente a premere il tasto "ESC" per arrivare nell'ultimo passo necessario per la calibrazione. Nel momento in cui l'utente preme il tasto "ESC" la finestra di visualizzazione del flusso d'immagini proveniente dalla webcam viene chiusa e l'utente si trova nuovamente nella finestra di terminale.

Al termine della parte di acquisizione di dati dalla webcam viene richiesto all'utente di inserire la distanza dell'osservatore codice QR rilevato e la dimensione in millimetri del codice QR. Utilizzando le immagini del pattern a scacchiera è possibile calcolare la matrice di parametri intrinseci (*intrinsic_matrix*) della webcam ed i coefficienti di distorsione (*distortion_coeffs*), questi due dati saranno poi usati dal server di localizzazione per rimuovere la distorsione nelle immagini acquisite dalla webcam.

Prima della terminazione il programma creerà un file di configurazione il cui nome è quello scelto dall'utente mediante l'apposito parametro inserito dall'utente da riga di comando all'avvio del programma.

3.2. Server di localizzazione

Il server di localizzazione è strutturato secondo un'architettura a thread, è infatti presente un thread adibito all'analisi del flusso continuo di immagini che consente l'estrazione di informazioni dai codici QR rilevati ed un altro thread che consente di inviare le informazioni ricavate dai codici QR ai client che le richiedono. Per la gestione dei thread si è deciso di utilizzare la libreria *pthread* in quanto disponibile per entrambi gli ambienti di test a nostra disposizione.

È evidente che i due thread possono lavorare senza particolari problemi indipendentemente l'uno dall'altro, vi è però il problema della struttura dati condivisa tra i due. L'accesso alla struttura dati che contiene le informazioni riguardo all'ultimo codice QR rilevato è però fatto utilizzando i semafori. Ogni accesso in scrittura da parte del thread di localizzazione è preceduto dall'acquisizione del mutex e al termine della scrittura il mutex viene poi rilasciato. Nel caso del thread di comunicazione il mutex viene acquisito prima dell'invio al client delle informazioni e viene rilasciato nel momento in cui la richiesta viene inviata. In questo modo è stato possibile evitare eventuali race condition che avrebbero potuto portare il server a fornire al client dei dati non validi.

L'acquisizione del mutex è fatta nel seguente modo:

```
while(pthread_mutex_lock(&mutex) != 0);
```

Il rilascio del mutex è stato fatto nel seguente modo:

while(pthread_mutex_unlock(&mutex) != 0);

Nell'eventualità in cui uno dei due thread dovesse incontrare un errore irreversibile e quindi terminare avviserà l'altro thread della propria terminazione, in modo da arrestare l'intero programma.

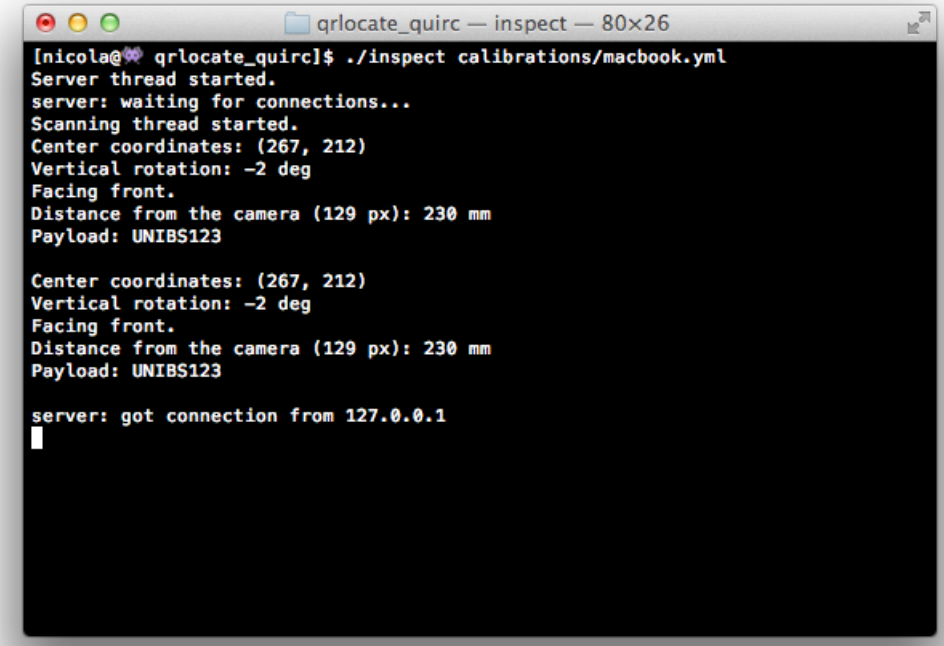


Fig. 4 - Il programma server

3.2.1. Thread di localizzazione

Il thread di localizzazione si occupa di acquisire un flusso continuo d'immagini da webcam, convertirle in un formato comprensibile dalla libreria utilizzata per la localizzazione di codici QR, determinare se è presente un codice QR nell'immagine e, in caso positivo, estrarre le informazioni del codice QR e salvarle nell'apposita struttura dati.

3.2.1.1 Acquisizione e conversione delle immagini

L'acquisizione delle immagini da webcam è fatta sfruttando l'oggetto *VideoCapture* presente nella libreria OpenCV. L'immagine acquisita viene manipolata per rimuovere la distorsione mediante la seguente chiamata di funzione:

undistort(frame, frame_undistort, intrinsic_matrix, distortion_coeffs);

L'immagine risultante, convertita in scala di grigi, viene poi utilizzata nella chiamata alla funzione *cv_to_quirc()* per popolare la struttura dati della libreria di localizzazione di codici QR con i dati dell'immagine da analizzare.

3.2.1.2 Localizzazione del codice QR nell'immagine

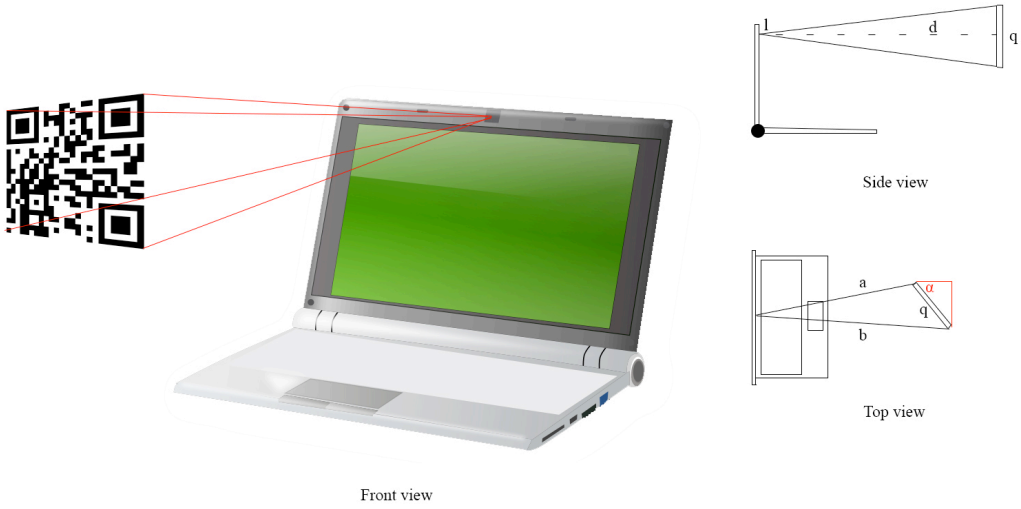
Mediante la chiamata a *quirc_end()* si fa si che la libreria rilevi la presenza di eventuali codici QR nell'immagine. Nel caso in cui vi siano dei codici QR presenti, la chiamata a *quirc_count()* restituirà il numero di codici QR rilevati.

In caso sia presente almeno un codice QR si effettua la chiamata a *quirc_extract()*, una funzione in grado di selezionare l'n-esimo codice QR presente nell'immagine e di popolare la struttura dati *quirc_code* con il contenuto del codice QR. Nella struttura *quirc_code* è presente la posizione del codice QR nell'immagine e grazie alla funzione *quirc_decode()* è possibile ottenere il payload contenuto nella struttura *quirc_code* risultante dall'estrazione.

3.2.1.3 Estrazione delle informazioni

Utilizzando i dati contenuti nella struttura *quirc_code* è possibile determinare in modo molto semplice il centro del codice QR: è necessario mediare la posizione sull'asse X e sull'asse Y di due delle quattro posizioni degli angoli, cioè il punto medio della diagonale del quadrato che racchiude il codice stesso.

Grazie alle informazioni riguardo alla posizione del codice QR nell'immagine è possibile calcolare la lunghezza, in pixel, di ogni suo lato. Inoltre è possibile calcolare la distanza del codice QR rispetto all'osservatore attraverso il rapporto tra la media della lunghezza del lato sinistro e del lato destro e il fattore di scala ottenuto durante la fase di calibrazione.



Si può quindi utilizzare la seguente:

$$d = \frac{fq}{l}$$

ove d è la distanza stimata del codice QR dall'osservatore, f è il fattore di scala ottenuto durante la calibrazione, q è la dimensione in millimetri dei codici QR decisa durante la calibrazione ed l è la lunghezza del lato, in pixel, rispetto al quale si vuole misurare la distanza, nel caso specifico è la media della lunghezza del lato sinistro e del lato destro (e quindi un ipotetico asse verticale posto al centro del codice QR).

Sfruttando le informazioni ottenute con la misura di distanza è inoltre possibile stimare la rotazione prospettica del codice QR utilizzando una semplice formula trigonometrica:

$$\alpha = \arcsin\left(\frac{a-b}{q}\right)$$

ove α è l'angolo di rotazione prospettica cercato, a è la distanza del lato sinistro del codice QR dall'osservatore, b è la distanza del lato destro del codice QR dall'osservatore e q è la dimensione in millimetri del codice QR.

3.2.2. Thread di comunicazione

Il thread di comunicazione è un semplice thread che apre un socket TCP sulla porta 9930 e si mette in ascolto, in attesa che dei client si colleghino. Quando un client si collega al server, il server invia al client il contenuto della struttura dati in cui sono contenute le informazioni sull'ultimo codice QR rilevato.

3.3. Client

Il programma client accetta in ingresso come parametro l'indirizzo IP del server di localizzazione a cui deve collegarsi, instaurando una connessione TCP sulla porta 9930, per ricevere le informazioni di localizzazione a lui necessarie. Si occupa poi di visualizzare sulla schermata del terminale le informazioni ricevute.

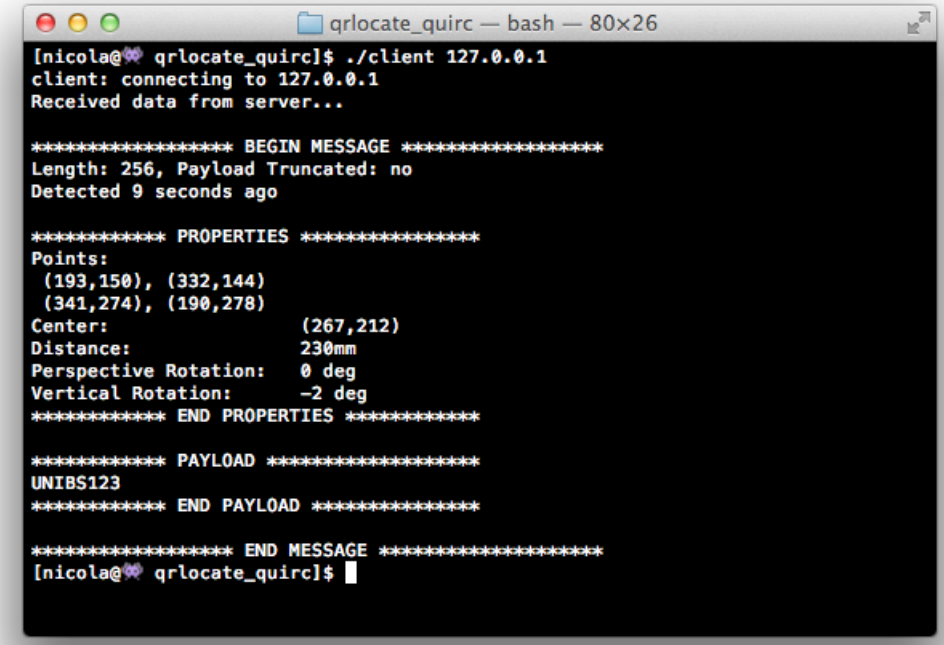


Fig. 5 - Il programma client

3.4. La struttura dati QRInfos

Per poter inviare pacchetti dal client al server si è deciso di utilizzare una struttura dati chiamata *QRInfos*, questa struttura contiene diversi campi di utilità per il client riguardo al codice QR rilevato dal server:

- *message_length*, questo campo è di tipo intero e contiene la lunghezza massima per la stringa *qr_message*, se nessun codice QR è stato rilevato questo campo è impostato al valore `-1`;

- *payload_truncated*, questo campo di tipo intero consente al client di capire se il messaggio *qr_message* è stato troncato perché più lungo della lunghezza massima del messaggio, MAXLENGTH;
- *qr_message[MAXLENGTH+1],* il payload del codice QR, eventualmente troncato per non eccedere la dimensione massima;
- *x0, y0, x1, y1, x2, y2, x3, y3,* le coordinate dei quattro vertici dell'ultimo codice QR rilevato;
- *distance,* la distanza stimata tra l'osservatore e il codice QR rilevato;
- *perspective_rotation*, la rotazione in prospettiva stimata per il codice QR rilevato;
- *vertical_rotation*, la rotazione rispetto all'asse verticale stimata per il codice QR rilevato;
- *timestamp_recognition*, questo campo consente al server di salvare la posizione temporale in cui l'ultimo codice QR è stato rilevato;
- *timestamp_current*, questo campo è utilizzato popolato dal server ad ogni richiesta del client e contiene la posizione temporale attuale del server. Questo è utile per determinare la validità del codice QR in base al tempo che è trascorso tra il rilevamento e la richiesta di questa informazione da parte del client. Utilizzando questo campo aggiuntivo, rispetto ad utilizzare solo *timestamp_recognition*, permette di evitare la necessità di tenere gli orologi di client e server sincronizzati.

3.5. Risultati sperimentali

Si è provveduto ad analizzare le prestazioni e l'accuratezza del metodo proposto utilizzando la webcam del netbook Acer Aspire One in un tipico ambiente con illuminazione al neon. È stata quindi effettuata la calibrazione utilizzando dei codici QR con una dimensione di 180mm.

I risultati sperimentali sono stati ottenuti utilizzando dei codici QR di versione 1 e 2 in quanto nei codici QR versione 1 è possibile memorizzare fino a 25 caratteri alfanumerici mentre con i codici di versione 2 si può arrivare a 47 caratteri alfanumerici, entrambi i valori sono considerati sufficienti per l'applicazione in esame.

Si è verificato sperimentalmente che utilizzando dei codici QR versione 1 è possibile ottenere una decodifica corretta fino alla distanza di 1700mm. Utilizzando invece dei codici QR versione 2 è stato possibile ottenere una decodifica corretta fino a 1500mm a causa del contenuto informativo più denso. Il valore massimo a cui è possibile effettuare una decodifica corretta è legato alla risoluzione della webcam utilizzata e quindi per aumentare la distanza a cui è possibile decodificare correttamente i codici QR è necessario utilizzare una webcam a risoluzione più elevata.

Le prestazioni in termini di riconoscimenti al secondo sono risultate simili in entrambi i casi: a distanza massima le prestazioni di riconoscimento sono di 1 ogni 2 secondi mentre a distanze inferiori i riconoscimenti aumentano fino a 5 al secondo. Il numero di riconoscimenti al secondo è legato alla configurazione hardware del calcolatore in uso, pertanto un calcolatore con delle specifiche tecniche migliori sarà in grado di effettuare un numero di riconoscimenti al secondo maggiore.

A distanze nell'intorno del massimo ottenibile per la configurazione in uso (webcam e hardware) è possibile che la libreria Quirc rilevi correttamente la posizione del codice QR ma non sia in grado di decodificarne in modo corretto il payload.

4. Modalità operative

In questo paragrafo si analizzano le modalità di installazione ed esecuzione del software per la distribuzione Linux Debian.

4.1. Componenti necessari

Per l'installazione e la compilazione del programma è necessario digitare i seguenti comandi:

sudo apt-get install build-essential

sudo apt-get install git

sudo apt-get install pkg-config

sudo apt-get install libcv-dev

sudo apt-get install libcvaux-dev

Durante l'utilizzo del programma di calibrazione, descritto più avanti, è possibile il verificarsi di un bug presente nella versione compilata per Debian di OpenCV: qualora questo accada si rimanda alla sezione avvertenze.

4.2. Modalità di installazione

È ora necessario effettuare la clonazione dei sorgenti del progetto dal repository GitHub attraverso il seguente comando:

git clone https://github.com/nzaghen/qrlocate.git

Questo commando crea la cartella qrlocate nella quale sono contenuti i file sorgente ed il Makefile necessario per la compilazione.

Per compilare il programma è sufficiente digitare i seguenti comandi:

cd qrlocate

make

Se la compilazione è completata senza errori, nella cartella corrente sono stati creati gli eseguibili dei programmi **calibrate, inspect** e **client.** Il programma **calibrate** consente di calibrare la webcam per compensare le distorsioni introdotte dalla stessa; **inspect** costituisce il server che acquisisce il flusso di immagini dalla webcam e invia le informazioni decodificate, se presenti, al client connesso; infine **client** consente di aprire una connessione al server sulla porta *9930*, di ricevere le informazioni decodificate dal server e di visualizzare tali informazioni sulla shell di sistema.

Per rendere operativo il programma è necessario avere a disposizione il seguente materiale:

1. La stampa cartacea del pattern a scacchiera, inclusa nel repository del progetto;
2. La stampa cartacea di un codice QR, con una dimensione non inferiore ai 150mm di lato (per lato si indica la distanza tra i due vertici superiori esterni dei Position Pattern superiori del QR, cioè i due vertici esterni del lato superiore). Questa dimensione deve essere nota al momento della calibrazione della webcam. Prima di procedere è inoltre necessario stabilire e annotare la distanza dalla webcam alla quale si intende far rilevare il codice QR al programma di calibrazione.

Una volta procurato il materiale necessario, è mandatorio seguire i seguenti passi:

1. Effettuare la calibrazione della webcam: questa consente di generare i parametri di compensazione della distorsione e i dati necessari per il calcolo della distanza dell'osservatore dal codice QR nel programma **inspect**.
2. Avviare il programma **inspect:** avendo cura di fornire come parametro di ingresso il file *nome_file_parametri.yml* creato in fase di calibrazione. Questo crea il server che si mette in ascolto delle connessioni sulla porta 9930.
3. Avviare il programma **client**: fornendo come parametro di ingresso l'indirizzo IP del calcolatore sul quale è in esecuzione il programma server.

Di seguito viene descritta taratura della webcam, analizzando le modalità con cui questa operazione possa essere svolta e motivando i risultati ottenuti.

4.3. Modalità di taratura

È stato già descritto il materiale necessario per effettuare l'operazione di taratura, che si riporta per completezza:

1. Pattern a scacchiera, nel nostro esempio di 9 e 6 quadrati rispettivamente per il lato lungo e corto (larghezza e altezza)
2. Codice QR di dimensione nota (decidere e annotare la distanza di acquisizione)

Per avviare il programma di taratura deve essere digitato il seguente comando:

./calibration calibration -w 9 -h 6 -o parameters.yml -op -oe

Il nome del file dei parametri può essere deciso dall'utente in fase di esecuzione, rispettando l'estensione del file "yml". In caso non venga fornita alcun nome per il salvataggio dei parametri, il corrispondente file è creato con il nome di "out_camera_data.yml". Questo file costituisce un parametro necessario per il funzionamento del programma **inspect.**

4.3.1. Checkerboard pattern e riconoscimento del QR

Una volta avviato, il programma mostra una finestra nella quale è possibile vedere il video acquisito dalla webcam in proprio possesso. Per avviare la taratura l'utente è invitato a premere il tasto 'g' sulla tastiera: in questo modo il programma inizia la ricerca del pattern nell'immagine ed effettua 15 diapositive nel momento in cui il pattern viene riconosciuto.

Per effettuare al meglio questa operazione il programma attende un tempo fisso di circa 2 secondi tra l'acquisizione di due diapositive successive: questo consente di non catturare quadri identici e generare quindi informazione inutile ai fini della calibrazione.

L'utente che effettua la calibrazione deve fare in modo di muovere il pattern in tutta l'area coperta dalla webcam, avendo cura di non piegare il foglio sul quale è stampato: per questo è utile usare un foglio di cartoncino rigido che non distorca la scacchiera.

Una volta completata l'acquisizione delle immagini, il programma suggerisce all'utente di premere nuovamente il tasto 'g' sulla tastiera per effettuare il riconoscimento del codice QR. Questa operazione deve essere condotta nel rispetto di alcune condizioni, dal momento che il programma salva i parametri del primo QR identificato:

1. Utilizzare un codice QR di cui sia nota la dimensione in millimetri, secondo le regole già esposte per calcolarla.
2. Posizionarsi prima della pressione del tasto "g" alla distanza (nota) a cui si desidera effettuare l'identificazione.

Al termine di questo riconoscimento l'utente è avvisato con un messaggio a video della corretta terminazione della calibrazione e può pertanto chiudere (premendo 'esc' sulla tastiera) la finestra di acquisizione. Sulla finestra di terminale dalla quale è stato avviato il programma vengono quindi richiesti in input dall'utente i seguenti dati:

1. Dimensione in millimetri del codice QR
2. Distanza di acquisizione, anch'essa in millimetri, del QR dalla webcam

Al termine dell'inserimento di questi dati, la calibrazione è conclusa.

4.3.2. Scopo della taratura

Come già descritto nel paragrafo sulla soluzione adottata, la calibrazione ha il duplice scopo di creare la matrice di distorsione per la webcam in uso e di trovare il fattore di riduzione necessario per calcolare la distanza da un codice QR di dimensione nota.

Col primo risultato ci si aspetta di ridurre le aberrazioni geometriche introdotte dal sistema di lenti della webcam e pertanto di fornire quadri privi di distorsioni e più aderenti alla realtà alla libreria per la decodifica del codice QR.

Con la seconda operazione, nota al momento dell'acquisizione la dimensione del codice QR cercato, è possibile stimare non solo la distanza con buona approssimazione dall'osservatore, ma anche il grado di rotazione prospettica. Queste informazioni, se combinate con l'informazione contenuta nel codice QR, possono consentire ad un osservatore di localizzarsi nello spazio.

4.4. Avvertenze

Durante il normale utilizzo del programma di calibrazione, per la distribuzione Linux Debian, siamo incorsi nel bug documentato al link https://bugs.launchpad.net/ubuntu/+source/opencv/+bug/684302 che porta in programma a terminare a causa di un Segmentation Fault.

Per correggere questo bug è stato usato un workaround che prevede di cancellare i cambiamenti contenuti in una patch applicata ad OpenCV dagli sviluppatori di Debian e ricompilare la libreria OpenCV. I comandi da eseguire sono i seguenti:

sudo apt-get install devscripts fakeroot quilt cmak librawl1394-dev dibdc1394-22-dev liblapack-dev swig texlive-fonts-extra texlive-latex-extra texlive-latex-recommended texlive-latex-xcolor texlive-fonts-recommended

apt-get source opencv

cd opencv-2.1.0

cat /dev/null > debian/patches/fix_3rdparty_build.patch

debuild

cd ..

*sudo dpkg -i *.deb*

Al termine di questa operazione è stato possibile evitare il bug che causava l'interruzione inaspettata del programma di calibrazione.

5. Conclusioni e sviluppi futuri

È stato quindi realizzato un sistema software costituito da un programma server che effettua il rilevamento di codici QR considerato come ingresso il flusso di immagini proveniente dalla webcam, e un programma client che, previo collegamento al server, riceve i dati contenuti nel codice QR, insieme ai riferimenti spaziali sulla distanza dall'osservatore, al grado di rotazione prospettica e al tempo dall'ultimo rilevamento. Il software è corredato da un programma che consente di effettuare la calibrazione della webcam in uso al fine di generare i parametri per la correzione della distorsione e per il calcolo della distanza dell'osservatore dal codice QR da decodificare.
La verifica del funzionamento del software sviluppato è stata effettuata con l'hardware a disposizione nel netbook Acer Aspire One fornito in dotazione per il progetto, è necessario quindi portare il programma server nell'ambiente reale a cui è destinato, ovvero il laboratorio di robotica avanzata.

Questo lavoro potrebbe essere in futuro esteso per consentire la localizzazione di un osservatore mobile, come ad esempio un robot, utilizzando diversi codici QR distribuiti nello spazio. L'approccio possibile è utilizzare il payload contenuto nel codice QR come campo chiave all'interno di una base di dati ed utilizzare il client per effettuare interrogazioni a questa base di dati una volta ottenuti i dati del codice QR dal server.

Bibliografia

[1] OpenCV tutorial per la calibrazione della webcam:
http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

[2] Daniel Beer, Libreria Quirc per la decodifica di QR, repository GitHub:
https://github.com/dlbeer/quirc

[3] Zxing, libreria per la decodifica di codici a barre 1D e 2D con client per Android e Java:
http://code.google.com/p/zxing/

[4] DENSO WAVE INCORPORATED, QR code:
http://www.qrcode.com/en/

[5] ISO/IEC 18004, il codice QR:
http://raidenii.net/files/datasheets/misc/qr_code.pdf

Indice