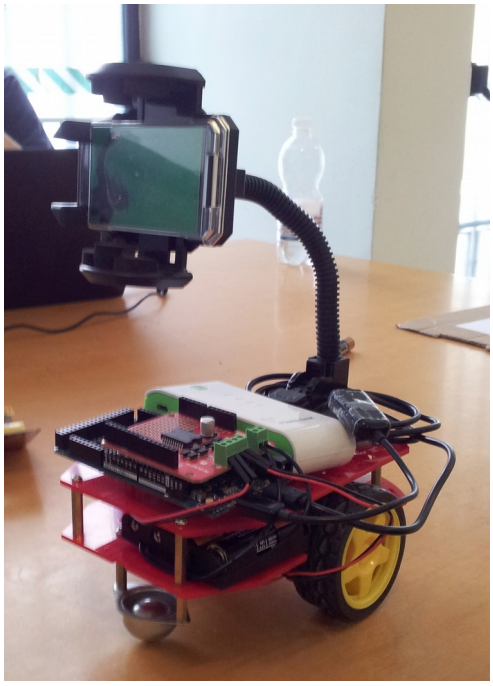
Progetto IA: ArDroid FollowUp

Jacopo Castellini

A. A. 2014/15

Scopo di questo progetto è la realizzazione di un piccolo robot (costruito con la scheda Arduino Due e il modulo Ardumoto per l'utilizzo dei motori) equipaggiato con uno smartphone Android utilizzato come modulo visivo e di elaborazione per il tracking e l'inseguimento di un dato disegno.



Come disegno da far seguire al robot abbiamo scelto il logo dell'UniPG, poiché facile da riconoscere e molto difficile da confondere con altri oggetti presenti nella scena.

Lo sketch Arduino per la comunicazione con Android e la gestione dei motori era stata realizzata precedentemente da un altro studente sempre nell'ambito del progetto di IA. La sola cosa che andava modificata erano dei campi per il riconoscimento del telefono da parte di Arduino.

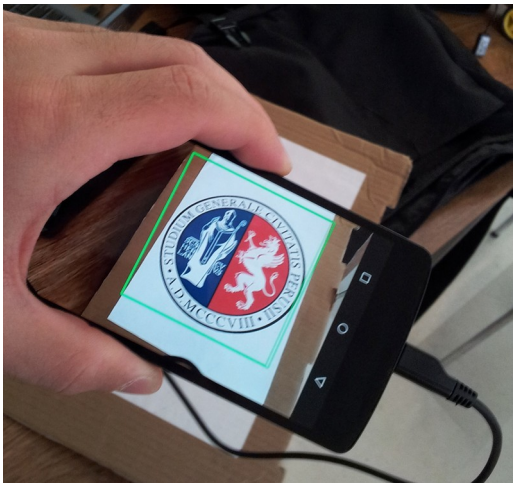
Abbiamo fatto delle assunzioni per il corretto funzionamento del progetto:

1. Non ci sono altri loghi dell'UniPG nell'ambiente in cui opera il robot
2. Il logo da seguire non si trova troppo distante dal robot (questa assunzione è dettata dal basso livello della fotocamera usata)
3. Il logo non fa movimenti o cambi di direzione troppo repentini (anche questo è dettato dal basso livello della fotocamera e dalla limitata capacità computazionale dei dispositivi mobili)
4. Il tempo di utilizzo è breve (questo perché l'intensa attività computazionale unita alla costante apertura a schermo intero surriscalda rapidamente il telefono, e ciò degrada mostruosamente le prestazioni)

Spieghiamo ora come funziona il codice lato Android. Per l'interfacciamento con Arduino abbiamo usato la modalità USB Host offerta da Android collegandoli tramite un cavo OTG, creando quindi un USB Accessory e specificando in un apposito file XML le stesse stringhe inserite prima nello sketch Arduino. Questo consente ai devices di riconoscersi tra loro, sincronizzarsi e comunicare. Vengono poi creati due stream dati per la comunicazione, e ad ogni decisione un thread apposito si occupa di inviare le stringhe che codificano i comandi ad Arduino.

Per quanto riguarda la parte strettamente legata alla Computer Vision abbiamo utilizzato le librerie OpenCV, poiché rappresentano uno standard molto utilizzato e potente. Il primo passo consiste nell'installare il software OpenCV Manager sul telefono, poiché necessario al funzionamento di ogni software che utilizza tale libreria (questo ci ha vincolato ad usarne la versione 2.4.9 nel nostro progetto, poiché il Manager non viene aggiornato da parecchio tempo…). Il layout dell'applicazione consiste semplicemente in una SurfaceView in cui OpenCV disegna ogni volta il frame catturato dalla fotocamera.

Il metodo di riconoscimento del logo si basa sui classificatori a cascata forniti dalla libreria stessa. Il primo passo è stato generare un file XML contenente le regole per il classificatore usando gli appositi tools forniti dalla distribuzione per PC di OpenCV. Abbiamo quindi scaricato un dataset di circa 3000 immagini negative (cioè non contenenti il logo), abbiamo poi generato 1500 campioni positivi (contenenti il logo) dal logo originale tramite il tool createsamples.exe fornito ed infine abbiamo addestrato il classificatore tramite il tool traincascade.exe. Dopo svariate ore di addestramento (abbiamo eseguito 20 stages) abbiamo ottenuto un XML che ben riconosce il nostro logo. Poi, nel codice Java, ad ogni frame catturato dalla fotocamera si lancia un metodo che richiama un oggetto di tipo CascadeClassifier, precedentemente inizializzato con le regole contenute nel file XML generato prima, che si occupa di controllare se la scena contiene o no il logo cercato, ed eventualmente disegna un rettangolo intorno alla posizione trovata. Infine, in base alla dimensione e alla posizione di tale rettangolo viene determinata il movimento da far eseguire al robot.



In verità, avevamo poi considerato anche un altro approccio per il riconoscimento del logo nella scena, basato questa volta sulla Feature Detection grazie all'algoritmo ORB (usato sia come FeatureDetector sia come DescriptorExtractor), che si è rivelata però poco efficiente in termini di velocità, anche se molto buona per il riconoscimento fine a se stesso. Questa all'inizio prende in input l'immagine da cercare e ne estrae dei punti chiave, e poi ricava un descrittore di questi. Ad ogni frame, dopo averne ricavato i punti chiave ed il descrittore, tramite un matching basato su KNN cerca di far corrispondere i punti chiave dell'immagine da cercare ad un sottoinsieme dei punti chiave trovati nella scena. Abbiamo poi scremato questi punti, tenendo in considerazione solo i più “buoni”. Infine, si proiettano tramite un omografia i vertici dell'immagine originale nella scena, in modo da determinare la posizione del logo in essa. Tuttavia l'eccessiva mole di calcoli da eseguire rende la tecnica buona per applicazioni statiche (ricerca di un oggetto in un immagine), ma non per un utilizzo real time.