Progetto di robotica

Inversione cinematica
e
visione robotica

Marco Loddo
Maria Chiara Cecconi

Obbiettivo

- Creare un software di controllo per il braccio robotico UR5

- Il software deve permettere di riconoscere e raggiungere oggetti marker predefiniti

UR5:

- 6 gradi di libertà
- Peso massimo sollevato 5 kg
- Ampiezza massima raggiungibile 850mm

Tecnologie usate per lo sviluppo

- **Gazebo**
- **ROS**
- **Python**
  Librerie python d'interesse:
  - Supporto a ROS (*roslib, rospy,* …)
  - Analisi visiva (*OpenCV, Camera.py,* ...)

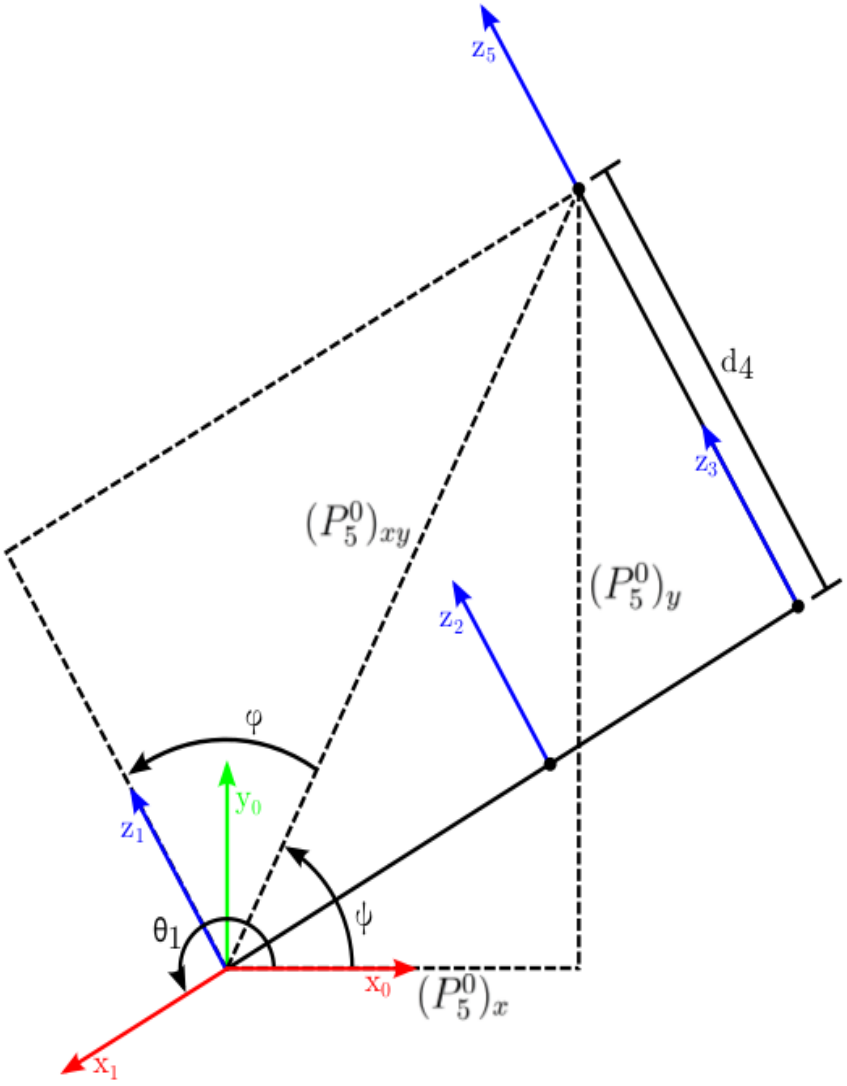
Metodi usati – Inverse Kinematics

- Raggiungimento della posizione obbiettivo
- Algoritmo analitico per il calcolo della IK
- Approccio:
  - geometria del braccio
  - calcolo a partire dal primo giunto (shoulder)
  - ricaviamo giunto per giunto tutte le possibili soluzioni
  - si seleziona la “miglior” soluzione

IK - Geometria UR5

THETA 1 – SHOULDER UP/DOWN:

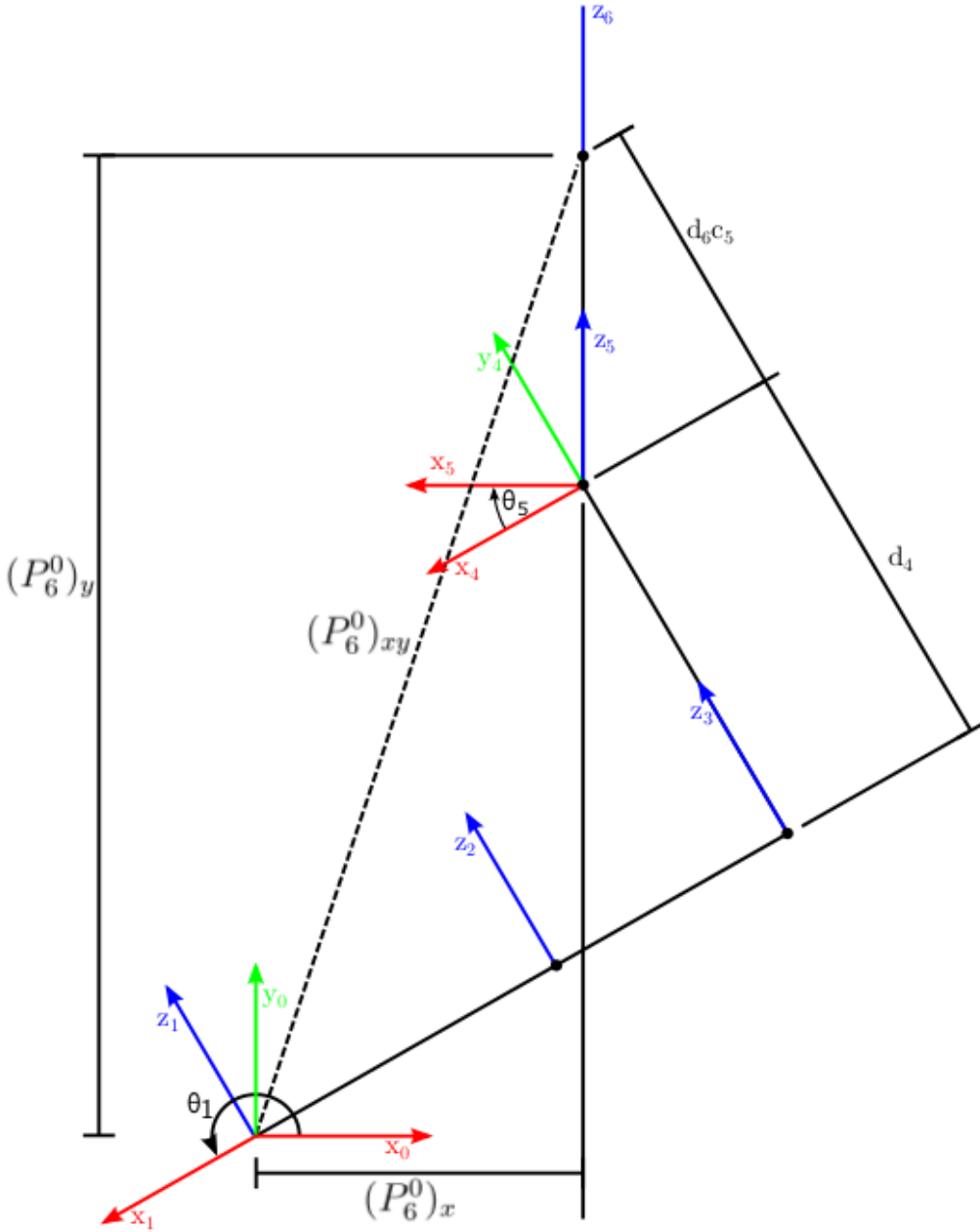
- Considero vettore posizione tra base e frame 5
- P_{05}: traslare di d6 in direzione negativa *z* del frame 6
- Si vede dalla figura che
  $\theta 1 = \psi + \varphi + \pi/2$
- dove
  $\psi = atan2((P_{05})_y , (P_{05})_x)$
  $\varphi = \pm arccos(d4 / ||(P_{05})||_{xy})$



IK - Geometria UR5

<u>THETA 5 – WRIST UP/DOWN:</u>

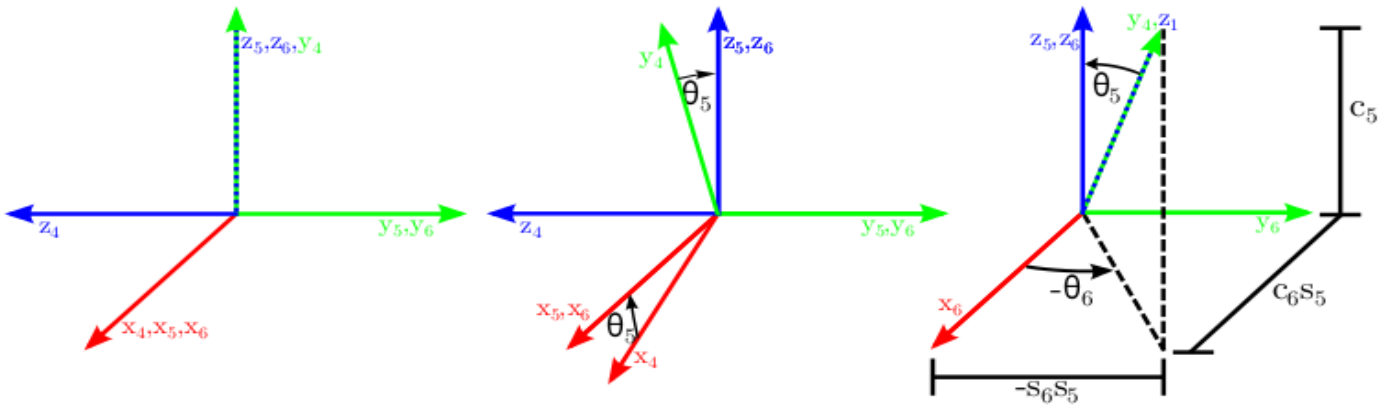
- Conoscendo θ1 risolviamo per θ5
- Consideriamo ora la posizione del frame 6 rispetto a frame 1
  $(P_{61})_z = d6*\cos(\theta5) + d4$
- dove
  $(P_{61})_z = (P_{60})_x*\sin(\theta1) - (P_{60})_y*\cos(\theta1)$
- Risolviamo per θ5
  $\theta5 = \pm \arccos(((P_{61})_z - d4) / d6)$
- θ5 perciò è ben definito se l'argomento è ≤ 1.



IK - Geometria UR5

THETA 6

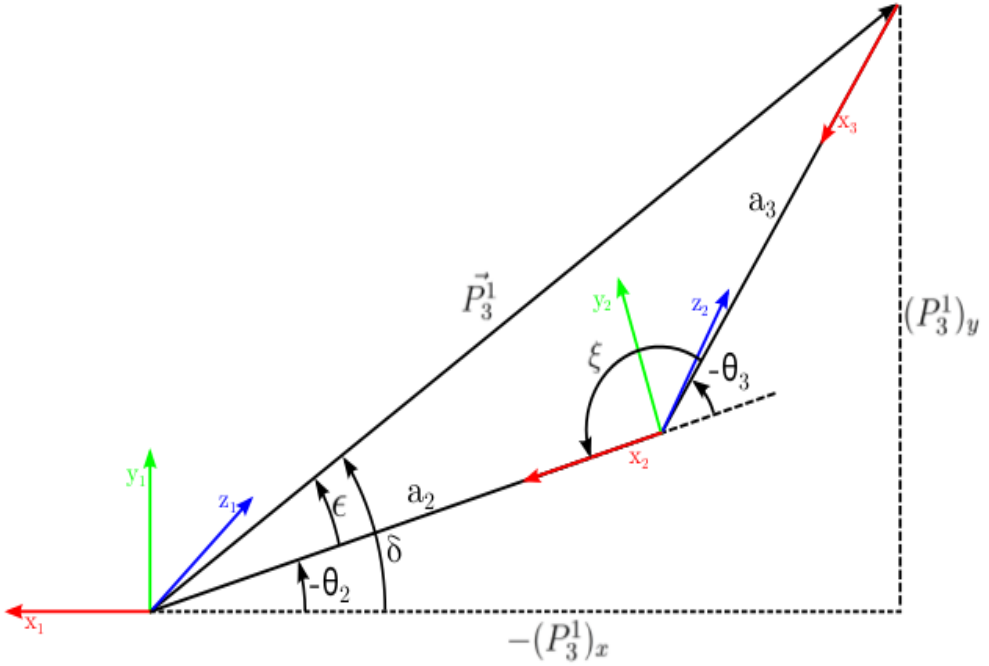
- Adesso troviamo la trasformazione dal frame 6 al frame 1
  $T_{61} = ((T_{01})^{-1}\ T_{06})^{-1}$
- Ricordando la struttura delle trasformazioni omogenee, avremo nella terza colonna
  $-\sin(\theta 6)\ \sin(\theta 5) = z_y$
  $\cos(\theta 6)\ \sin(\theta 5) = z_x$
- Risolviamo per θ6
  $\theta 6 = \text{atan2}(-z_y/\sin(\theta 5), z_x/\sin(\theta 5))$
- Non è ben definito se $\sin(\theta 5) = 0$ oppure $z_x, z_y = 0$



IK - Geometria UR5

THETA 2, 3 – ELBOW UP/DOWN:

- Consideriamo I giunti rimanenti come un manipolatore planare 2R
- Troviamo la posizione del frame 3 rispetto all' 1: P_{13}



- Ci consentirà di ricavare $\theta 3$

  $\theta 3 = \pm arccos((||P_{31}||^2 - a_2^2 - a_3^2)/2*a_2*a_3)$

- Da esso troviamo $\theta 2$

  $\theta 2 = -atan2((P_{31})_y , -(P_{31})_x) + arcsin(a_3*sin(\theta 3)/||P_{31}||)$

- Infine $\theta 4$

  $T_{34} = T_{31}*T_{14} =$

  $= (T_{12}*T_{23})^{-1}*T_{14}$

  $\theta 4 = atan2(x_y , x_x)$

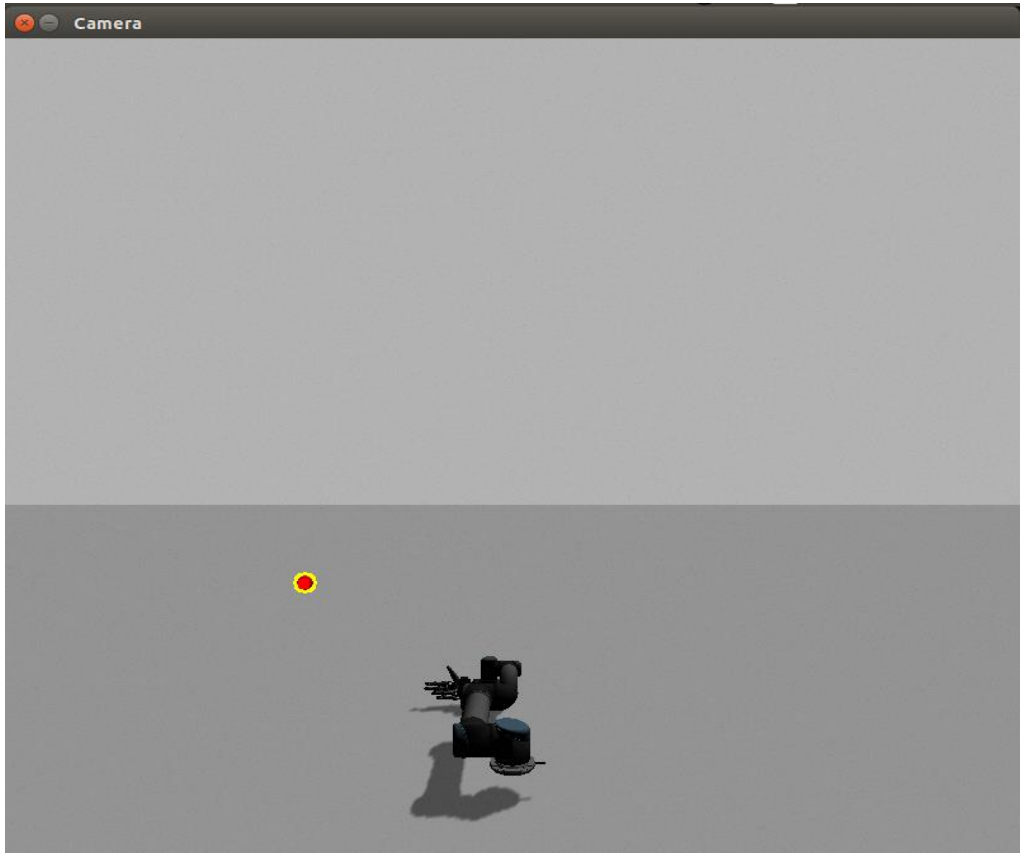
Metodi usati – modifiche al modello

Si è modificato il modello *UR5.sdf* come segue:

- Inserita una *box* dietro al braccio, in questa viene posto il modello di un sensore, in particolare una *telecamera*. Questa servirà a vedere il *marker* da raggiungere (ball rossa)

- Inserito un *end effector*, in particolare una *mano*. Il modello originario è il braccio *Hollie*, dal quale si è isolata la mano ***Schunk hand*** per poi inserirla nel modello dell'UR5.

Metodi usati – Visione



- Presa un'immagine, si filtra pixel per pixel con una mask (i limiti di valore da 0 a 255 del colore desiderato)
- Una volta ottenuta la maschera, si trova il bordo più piccolo che la contiene, con annesse coordinate e dimensioni.
- Prese queste dimensioni, viene effettuato un calcolo di trasformazione di coordinate da schermo a spazio 3D
- Ottenute le coordinate, vengono convertite poi in coordinate polari per poter essere ammissibili nello spazio del robot*
- Viene effettuato uno shift di coordinate da coordinate della camera a coordinate del robot

*la conversione è soggetta a un errore piuttosto trascurabile dati i vari difetti di conversione per l'errore di macchina.

Processo esecutivo

Il processo viene eseguito come segue:

- Si ha una **pallina** rossa, che sia visibile dalla telecamera
- Il codice acquisisce le **immagini** e identifica il **marker**
- Vengono calcolate le **coordinate** nel mondo dell'oggetto riconosciuto

- Le coordinate vengono passate al metodo di calcolo della **IK**
- Calcolate le varie soluzioni dei giunti della IK, si sceglie quella **ottimale**

- Si fanno ruotare i **giunti** del robot secondo la soluzione

- A tal punto la mano sarà vicina alla pallina, abbastanza da poterla afferrare
- Si fa eseguire alla mano il movimento di "**grasp**"

Risultati

- Test eseguiti sulla posizione calcolata dall'immagine della pallina
- Per ognuno dei 3 test si sono presi 3 risultati

TEST 1	TEST 2	TEST 3
0,2027 – 0,207226	-0,6001 – -0,60495	0,4157 – 0,410265
0,1435 – 0,144019	0,4103 – 0,415624	-0,0989 – -0,09859
0,5012 – 0,656701	0,3021 – 0,474741	0,6257 – 0,717697
T.E.: 0,014040 sec	**T.E.:** 0,015161 sec	**T.E.:** 0,017479 sec

- I 3 risultati erano uguali in ogni caso a quelli sopra mostrati
- Si rileva il maggior scostamento nell'asse Z

Risultati

- Test eseguiti sulla IK
- Il risultato della IK è stato rielaborato con la FK per cercare di ottenere il punto XYZ di partenza

TEST 1	**TEST 2**	**TEST 3**
0,2027 – 0,2025	-0,6001 – -0,6007	0,4157 – 0,4102
0,1435 – 0,1437	0,4103 – 0,4129	-0,0989 – -0,09836
0,5012 – 0,656701	0,3021 – 0,3024	0,6257 – 0,6266
T.E.: 0,0151610 sec	**T.E.:** 0,0174798 sec	**T.E.:** 0,026179 sec
Tot: 2,04254 sec	**T.E.:** 2,0196274 sec	**T.E.:** 2,056328 sec

Conclusioni

- La Cinematica inversa con approccio algebrico è veloce, ma complessa da studiare e implementare
- La IK singolarmente si dimostra precisa, sempre che il goal sia nel workspace
- La visione combinata alla IK si presenta precisa almeno nell'ordine delle prime due cifre decimali, fatta eccezione per Z (comunque il risultato è apprezzabile all'atto pratico)
- Il braccio sarà quindi in grado di raggiungere la pallina in maniera adeguata
- Una volta effettuato il grasp, possiamo usare la FK, o la IK nuovamente, per dire al braccio la nuova posizione in cui spostare la pallina