ROAD MAP PROGETTO ROBOTICA

La prima settimana ho avuto un grosso problema nell'installazione del dvrk, perché all'inizio non funzionava nulla e ho passato una settimana intera a risolvere tutti gli errori che incontravo. Alla fine vedendo una troppa discordanza tra i file ho notato che su github lo stesso file era appena stato aggiornato poche ore prima. Da li ho capito che stavo lavorando con la versione "devel" del codice. Allora ho trovato il problema in dvrk_ros.rosintall https://github.com/jhu-dvrk/sawIntuitiveResearchKit/wiki/CatkinBuild dove alcuni pacchetti erano "master" mentre altri "devel".

Una volta capito il problema ho ripreso in mano tutto da capo e ho scritto una mini guida. Ho creato il file **dvrk_ros.rosintall** con tutti i branch settati alla versione master.

Ci sono due modalità di esecuzione:

1. Poi c'è la possibilità di eseguire il launch-file e quindi eseguire tutti i vari nodi + rviz.
2. Oppure di eseguire semplicemente la console senza rviz.

Prima di proseguire ho cercato di capire il significato di tutti i file interessati nell'esecuzione del PSM,
In primis ho analizzato il console-file **(console-PSM1_KIN_SIMULATED.json).**
Questo file indica come PID il file (**sawControllersPID-PSM.xml)** che è un file contenente le varie specifiche per giunto e anche i vari limiti di giunto.
In Kinematic c'è il file **(psm-large-needle-driver.json)** che contiene oltre alla tabella DH del manipolatore anche altri matrici per gestire la cinematica.

Dopodichè ho analizzato le terne visibili su Rviz e ho cercato di capire come distinguere le 6 terne principali dall'insieme di terne visibili contenente anche frame di "supporto".
Ho analizzato il **drvk_arm_rviz.launch** e in modo ricorsivo ho analizzato tutti i file **urdf.xacro** usati per generare terne e modello del robot in rviz. Ho ordinato l'insieme di link e giunti complessivi e sono riuscito a distinguere le 6 terne principali.
Per fare ciò in modo pratico, usando la tabella DH definita nel file che ho detto prima e partendo dal base_frame, ho ricostruito tutte le terne del PSM1.

Il passo successivo è stato di capire come interfacciarmi con il manipolatore mediante ROS topic.
https://github.com/jhu-dvrk/sawIntuitiveResearchKit/wiki/Kinematics-Simulation
Ci sono due strade per fare ciò:

1. Nella cartella dvrk_python ci sono degli script di esempio per comunicare con il manipolatore mediante funzioni (funzioni che sono state bindate a topic ROS).
   Ho analizzato tutte le varie funzioni, provandole per capire cosa facessero e ho creato un mini script per interagire con il robot.
   (è possibile far muovere il manipolatore su rviz mediante questo modo)

2. Il secondo modo è quello di avviare roscore e console ed inseguito sottoscrivermi/pubblicare direttamente da terminali sui ROS topics senza quindi l'ausilio dello script python.
   Alla pagina *https://github.com/jhu-dvrk/sawIntuitiveResearchKit/wiki/Components-APIs* sono spiegati tutti i vari topic (cosa fanno, se sono public o subscribe)

In secondo luogo ho analizzato il binding tra le funzioni python e i topic ROS per capire a quale topic era asscociata una certa funzione. Tutto ciò per capire meglio i valori di output delle funzioni.

Quello che ho fatto è stato di leggere i valori tramite script python e di leggerli anche sottoscrivendomi ai relativi topic

Ho fatto delle prove tramite un mio script e analizzando i risultati ottenuti (problema con la variazione di x e y anche se di poco)
Ho provato a muovere il primo giunto R di 90° però analizzando i risultati si vede che arriva a muovermi massimo di 70° ed infatti nel file che specifica i limiti di giunto si può verificare questo limite.

In seguito ho fatto altre prove. Anche in 0,0,0 che con la cinematica inversa non ci può arrivare mentre con la cinematica diretta se sposto il giunto 3 ci arriva…
Ciò può essere dovuto al fatto che in realtà lo 0,0,0 è sbagliato perché il manipolatore arriverebbe a valori tendenti allo 0 ma non 0???

Dopo aver trovato il modo di applicare e leggere i valori delle due cinematiche, ho indagato cercando di capire che tecnica avessero usato per il calcolo della cinematica inversa (che utilizza l'algoritmo di newton). Ho cercato alcuni cenni teorici per capire il funzionamento.

Domande sulla cinematica inversa.

CINEMATICA INVERSA (PARTE 2 DEL PROGETTO)

- RCM coincide con l'origine del psm_base_frame
- Dmove = se siamo in 0,0,-0.1135 e faccio una dmove (0,0.05,0) quello che accade è che a partire dal primo punto mi sposto di 5 cm sulla y e ottengo il nuovo punto che sarà **0,0.05,-0.1135**
- Move = se siamo in 0,0,-0.1135 e faccio una move a (0,0.05,0) dovunque sia il manipolatore, esso cercherà di arrivare esattamente nel punto **0,0.05,0**

1.Problema cinematica diretta-inversa (conti non tornano)

Nel primo caso faccio delle stampe in:

- Riposo, EE nell'origine
- Dopo la home che porta EE a -0.1135
- Dopo la dmove del giunto 3 prismatico in modo da portare il manipolatore al punto 0,0,0 con la (cinematica diretta) e ci arriva

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
0.0

Current position ---- TOPIC: position_cartesian_current :
[[           1,           0,           0;
           0,           1,           0;
           0,           0,           1]
[           0,           0,           0]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.    0.    0.12  0.    0.    0.  ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.      0.      0.0065  0.      0.      0.    ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[-1.19286e-19, 3.34262e-08, 1.84215e-13]]
```

Nel secondo caso (parte 1) faccio delle stampe in:

- Prima della home
- Dopo la home
- Dopo una **move** (cinematica inversa) al punto 0,0,0 e come si può vedere l'EE non ci arriva (la coordinata z non tende a 0 ma ha un suo valore da considerare)

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
0.0

Current position ---- TOPIC: position_cartesian_current :
[[           1,           0,           0;
           0,           1,           0;
           0,           0,           1]
[           0,           0,           0]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.    0.    0.12  0.    0.    0.  ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ -4.52508249e-18  -3.67318842e-06   4.00000000e-02  -9.83519160e-17
   3.67318842e-06   4.02898431e-18]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91206e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91206e-17,          -1]
[ 1.23052e-07, 2.46104e-07,     -0.0335]]
```

Nel secondo caso (parte 2) faccio delle stampe in:

- prima della home
- dopo la Home
- Dopo una **dmove** (cinematica inversa) di 0,0,0.1135 e come si può vedere l'EE non ci arriva di nuovo (la coordinata z non tende a 0 ma ha un suo valore da considerare)

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
0.0

Current position ---- TOPIC: position_cartesian_current :
[[           1,           0,           0;
           0,           1,           0;
           0,           0,           1]
[           0,           0,           0]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.    0.    0.12  0.    0.    0.  ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[  3.20185424e-06   4.58141347e-05   4.00000000e-02   2.35220364e-11
  -4.58141347e-05  -3.20185425e-06]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69856e-11,           1, 9.03369e-17;
           1, 2.69856e-11, 7.34641e-06;
  7.34641e-06, 2.88584e-16,          -1]
[ 2.30314e-07,-9.61386e-07,     -0.0335]]
```

In questo terzo caso invece:

- Prima della home 0,0,0
- Home = -0.1135
- Dmove per la massima estensione = -0.2285
- Move a 0,0,0 ma non ci arriva

```
Current position ---- TOPIC: position_cartesian_current :
[[           1,           0,           0;
           0,           1,           0;
           0,           0,           1]
[          0,           0,           0]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.    0.    0.12  0.    0.    0.  ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.     0.     0.235  0.     0.     0.   ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 8.39327e-07, 8.72754e-07,     -0.2285]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ -2.65697009e-18  -3.67277060e-06   4.00000197e-02  -9.83519160e-17
   3.67318827e-06   2.16258662e-18]

Current position ---- TOPIC: position_cartesian_current :
[[ -2.6984e-11,           1,-9.91117e-17;
           1,  2.6984e-11, 7.34599e-06;
  7.34599e-06, 9.91117e-17,          -1]
[ 1.23052e-07,  2.4609e-07,     -0.0335]]
```

Altro esempio:

Cinematica diretta:

- 0,0,0
- Home = -0.1135
- 40 gradi sul primo giunto rotoidale
- 40 gradi sul secondo giunto rotoidale
- raggiungendo il punto x= +0.0559247 y= -0.0731184 z= -0.0663957

Cinematica inversa:

- CI arriva al punto definito sopra

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[  6.99999150e-01   7.54969599e-01   1.22301165e-01   6.49833972e-06
  -7.54967871e-01  -7.00003603e-01]
```

- Con un altro insieme di valori dei giunti

In questo ultimo caso di esempio accade una cosa molto strana:

- Siamo in 0,0,0
- p.home → 0,0,-0.1135
- p.dmove_joint_one(-0.1135,2) → retraggo il giunto prismatico in modo da finire in 0,0,0
- p.dmove(PyKDL.Vector(0.0,0.0,0.0) → **di logica non dovrebbe spostarsi,** e finisco in 0,0,-0.0335

```
Current position ---- TOPIC: position_cartesian_current :
[[           1,           0,           0;
           0,           1,           0;
           0,           0,           1]
[          0,           0,           0]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.    0.    0.12  0.    0.    0.  ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.      0.      0.0065  0.      0.      0.    ]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[-1.19286e-19, 3.34262e-08, 1.84215e-13]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ -2.49261469e-19  -3.67318826e-06   4.00000000e-02  -9.83519160e-17
   3.67318826e-06  -2.49637105e-19]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91206e-17;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91206e-17,          -1]
[ 1.23052e-07, 2.46104e-07,     -0.0335]]
```

```
[ WARN] [1607096251.913999863]: 1607096251.900802s, #1: PSM1: tool/endoscope needs to be inserted past the cannula to switch to c
artesian space
```

Warning che non accade se evito di fare la p.dmove_joint_one(-0.1135,2) ma comunque sia finisco sempre nello stesso punto: (0, 0, -0.0335)

2.Problema cinematica inversa in forma chiusa

La condizione **sufficiente** affinchè un manipolatore a 6 DOF abbia una soluzione in forma chiusa è:

- O 3 giunti rotazionali consecutivi hanno gli assi che si intersecano in un punto(es. polso sferico)
- Oppure 3 giunti rotazionali consecutivi hanno tutti gli assi paralleli

Questa condizione non è soddisfatta nel manipolatore PSM in quanto negli ultimi 3 giunti rotoidali accade che:

- 2 hanno le terne coincidenti
- 1 (l'ultimo) è spostato dagli altri due di un certo offset

Osservazioni:

1. Vedere se ci sono spunti in https://github.com/LucaArietti/robotics_project_dvrk_psm
2. RCM e psm_base frame coincidono.
3. Assumo come conosciute le coordinate del RCM
4. Tutte le rotazioni dei primi due giunti e i vari roll del gripper avvengono intorno al RCM
5. La posizione del 4 e 5 giunto dipende solamente dai primi 3 giunti.
6. C'è un offset di … dal 4-5 giunto al 6
7. Un altro vincolo da considerare è che pitch(5) e yaw(6) del polso sono ortogonali e mi fanno muovere lo strumento su due piani ortogonali

Passi

1. **L'idea è di trovare il vettore che mi permette di passare dal punto del 6 giunto al punto del 4-5 giunto**
2. Una volta trovare il vettore per tornare indietro, trovare il valore dei primi 3 giunti è facile perché sono una serie di arcos/arcsin/tan di angoli rispetto al base_frame/RCM
3. Fare la prova del 9 per vedere se la cinematica inversa in forma chiusa implementata da me ritorna valori simili a quella già implementata (i valori come 0.0091 e 0.4162 possono essere usati come costanti nel codice)

DOMANDE:

- Ma RCM e base-frame che coincidono sono anche la stessa terna?
- Poi ha fatto tutto con terne e DH di un altro paper ancora. L'RCM mostrato da Arietti non è lo stesso base_frame utilizzato nel dvrk ros.
- Una cosa che non mi quadra è che lui hai considerato L2" + d3 ed è come se L2" arrivasse fino all'origine e dall'origine in poi ci fosse d3.. ma da me non è cosi
- I miei ragionamenti devo farli dalla configurazione Home? Oppure dall'origine?

OSSERVAZIONE PARTICOLARE:

Risultati eseguendo più volte lo script contenente:

- *p.home()*
- *p.move(PyKDL.Vector(0.0 , 0.03, -0.04)*

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ -3.67321106e-06  -7.70616127e-01   5.86674586e-02  -3.76106039e-11
   7.70616127e-01   3.67318486e-06]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69848e-11,           1,-2.22158e-16;
            1, 2.69847e-11, 7.34641e-06;
  7.34641e-06,-2.39177e-17,          -1]
[-4.52634e-18,        0.03,       -0.04]]
```

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ -3.67320455e-06  -6.43497436e-01   7.47000000e-02  -1.57079633e+00
  -3.05432619e+00   6.43504782e-01]

Current position ---- TOPIC: position_cartesian_current :
[[    0.996195,   0.0522913,  -0.0697217;
   -0.0522913,     0.99863,  0.00182639;
    0.0697217,  0.00182639,    0.997565]
[-0.000793084,   0.0300208,  -0.0400277]]
```

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.01677878 -0.64342984  0.0746707  -1.56072716 -3.05432619  0.64350478]

Current position ---- TOPIC: position_cartesian_current :
[[    0.994935,   0.0603115,  -0.0804153;
   -0.0603115,    0.998177,  0.00243114;
    0.0804153,  0.00243114,    0.996758]
[ -0.00012164,   0.0300069,  -0.0400092]]
```

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.01935597 -0.64340748  0.07466099 -1.55918101 -3.05432619  0.64350478]

Current position ---- TOPIC: position_cartesian_current :
[[    0.994726,   0.0615423,  -0.0820564;
   -0.0615423,    0.998101,  0.00253165;
    0.0820564,  0.00253165,    0.996624]
[-1.86671e-05,   0.0300011,  -0.0400015]]
```

Tralasciando il valore dell'EE x,y,z, la cosa particolare è il valore dei giunti che piano a piano si aggiusta fino a convergere:

Il giunto 2 R assume i seguenti valori:

- -0.77061 rad
- -0.64349 rad
- -0.64342 rad
- -0.64340 rad

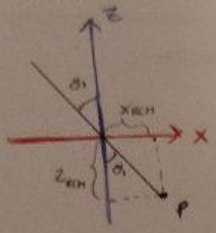
Il giunto 3 P assume i seguenti valori:

- 0.05866 m
- 0.07470 m
- 0.07467 m
- 0.07466 m

PROVA

IK numerica [0.4636435 , 0 , 0.0051221 , 0.0 , -0.4636435]
R R P R R

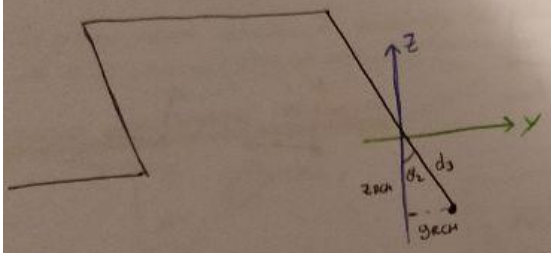
P (+0.02 , 0.0 , -0.04)



$\sigma_1 = \text{atan}\left(\frac{x_{RCH}}{z_{RCH}}\right) = \text{atan}\left(\frac{0.02}{-0.04}\right) = \begin{cases} -26.56505118° \\ -0.4636476\,1 \text{ rad} \end{cases}$

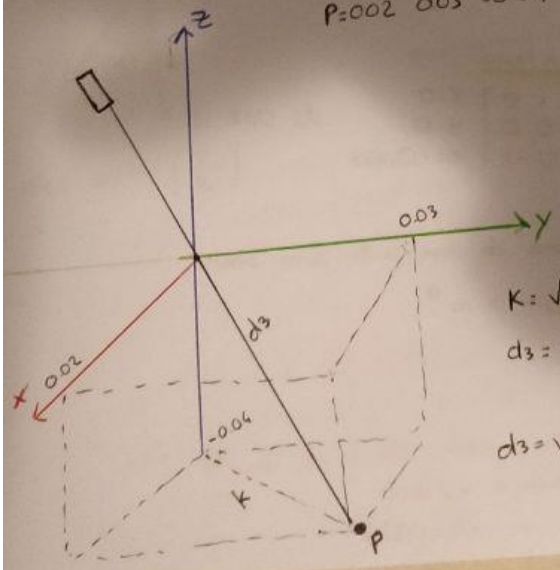
σ_1 va negato
il valore corrisponde

$\sigma_1 = \text{atan2}\,(x_{RCH}, z_{RCH})$



$\sigma_2 = \text{atan}\left(\frac{y}{z}\right) = \text{atan}\left(\frac{0.03}{-0.04}\right) = -0.6435$

P=002 003 -004



$K = \sqrt{002^2 + 003^2} = 0.0360555\ldots$

$d_3 = \sqrt{K^2 + z^2} + 0.0065 = 0.0603516$

$d_3 = \sqrt{x^2 + y^2 + z^2}$ = distanza tra due punti

Provando le formule con un punto generico:

- *p.home()*
- *p.move(PyKDL.Vector(0.02 , 0.03, -0.04)*

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[  4.63644499e-01  -6.99938566e-01   6.21712107e-02   4.29528618e-06
   6.99939342e-01  -4.63641732e-01]

Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-3.62144e-15;
           1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, -3.4232e-15,          -1]
[       0.02,        0.03,       -0.04]]
```

VALORI LUCA ARIETTI:

```
Current joints position (according to API notation):
[ 0.4061888  -0.5350099   0.05884089  0.0000043   0.69993934 -0.46364173]

----------

Joints values calculated are:
theta1:  0.406189    (in rad)
theta2:  0.53501    (in rad)
d3:  0.058841    (in meters)
```

Miei valori:

- theta1 = atan (x/z) o atan2 (x,z) = 0.46364761
- theta2 = atan (y/z) o atan2 (y,z) = -0.643501
- k = sqrt (x^2 + y^2)
- d3 = sqrt(k^2 + z^2) + 0.0065 = 0.0603516

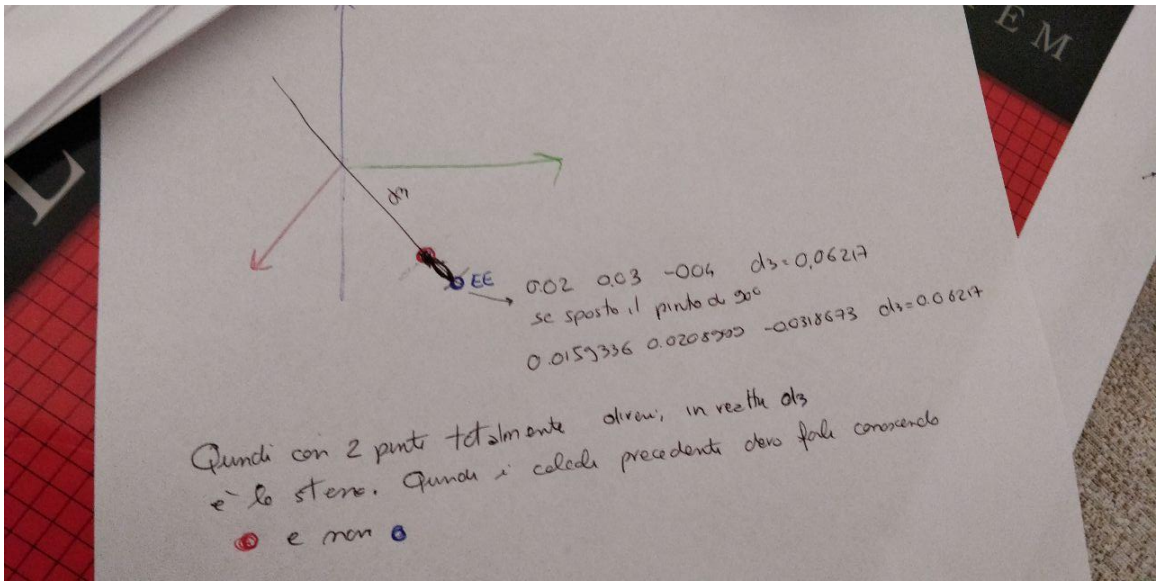
d3 non è corretto calcolarlo così, c'è da tenere conto di un'osservazione definita sotto.
Anche theta1/theta2 fanno riferimento alla posizione x,y,z di EE in questo calcolo. Mentre in realtà dovrebbe far riferimeto ad xyz di 5

Eseguendo una move dei giunti con i valori appena ottenuti, finisco nel punto.

```
Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[ 0.46364761 -0.643501    0.0603516   0.          0.          0.        ]

Current position ---- TOPIC: position_cartesian_current :
[[   -0.268333,    0.894427,    0.357768;
     0.799996, 2.20394e-06,    0.600006;
     0.536661,    0.447214,   -0.715538]
[   0.0192666,   0.0323111,  -0.0385329]]
```

Il problema è che

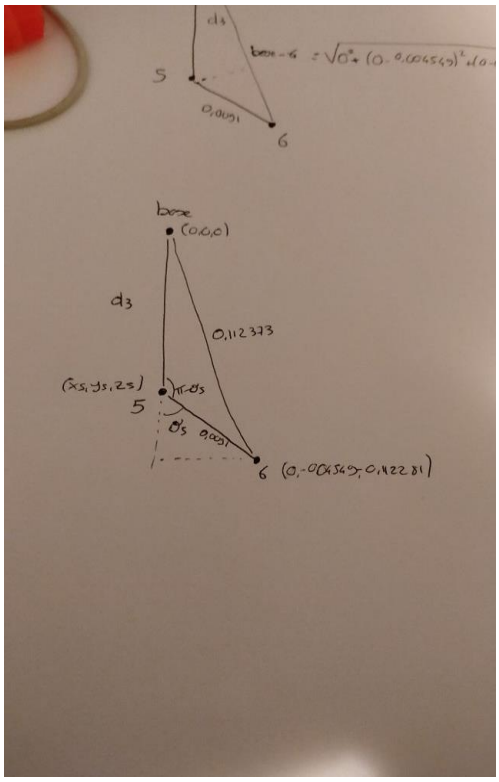


Con due punti diversi xyz dell'EE, otterrei due d3 completamente diversi. Tutto ciò perché dopo d3 c'è un offset di 0.0091m che dipende da come è orientato il giunto 5 R. Quindi prima di calcolare d3 devo risalire al punto in rosso. Mentre l'EE è il punto in blue.

TODO:

-Devo trovare un modo di esprimere il punto del giunto 6, in un punto del giunto 4-5
-Ricordando che la posizione del giunto 4-5 dipende solamente dalla posizione dei primi 3 giunti. (il giunto 5-6 sono ortogonali)

Se conoscessi theta5 potrei stabilire una relazione tra il SR5 e il SR6. Non conoscendo l'angolo non è possibile stabilire questa relazione.
Io conosco solo la distanza tra SR5 e SR6 che è fissata ed è di 0.0091m



Facendo alcune considerazioni comunque mi trovo ad avere d3 e theta5 come incognite.

IDEA: l'idea è di esprimere mediante i vari vincoli, le coordinate del punto del giunto 5 in riferimento al base_frame.
Di conosciuto ho:

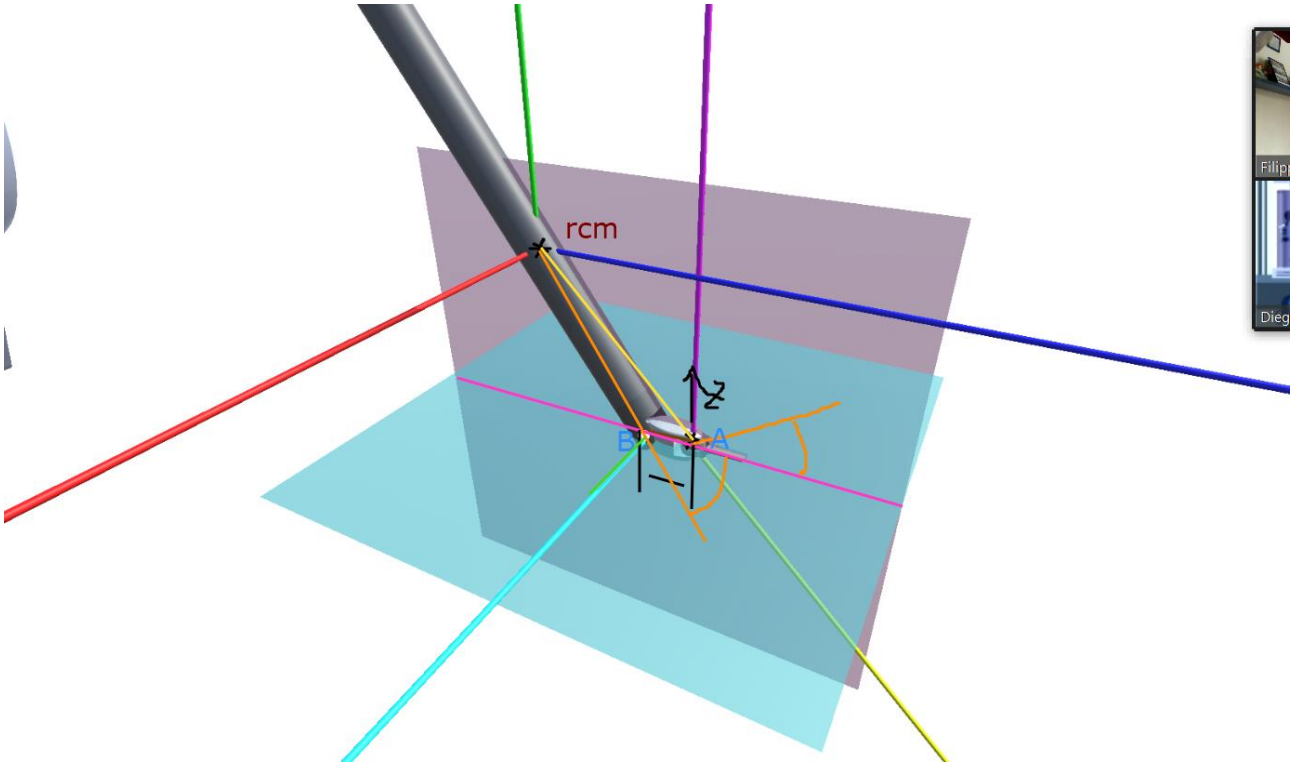
- distanza tra due punti (5-6) = 0.0091
- coordinate x,y,z del giunto 6 (EE)
- theta1 = atan (x5/z5) o atan2 (x,z) = 0.46364761
- theta2 = atan (y5/z5) o asin(y/d3 a cui devo togliere 0.0165 aggiunto per ottenere d3) #da capire
- d3 = sqrt(x5^2 + Y5^2 + Z5^2)+0.0165

Se trovo le coordinate del punto 5 a partire dal base frame sarei in grado di risalire al valore d3 e di conseguenza anche all'angolo theta5. Ma per trovare le coordinate del punto 5 espresso in base_frame dovrei almeno conoscere theta1 e theta2 che però non conosco.

"The rotation of the three end joints (near the end effector) does not affect the spatial position of the surgical instrument, therefore these three joints are neglected in the inverse kinematics calculation." Arietti.

Nel mio caso invece si

TODO-TERZO INCONTRO:



Nel file psm_large_needle_driver.json c'è una matrice, la tooltip-offest che alla fine di tutto fa ruotare il giunto 6 della DH secondo dei valori.
Quindi prima di tutto devo fare la rotazione inversa e sono apposto.

L'idea per procedere è:

- prodotto cross-esterno tra vettore viola (normale z che ottengo dopo l'inversa e sono 3 componenti) e il vettore giallo che è la distanza tra RCM e EE.
  Questo prodotto mi da la normale verde uscente da B e sarebbe la normale del piano grigio.
- L'equazione di un piano si può scrivere avendo un punto del piano e la normale uscente:
  - Piano azzurro = normale viola + A
  - Piano grigio = normale verde e l'RCM
- In seguito devo calcolare l'intersezione tra i due piani che mi da come risultato il segmento in rosa. Con il segmento, il punto EE e l'offset 0.0091 dovrei essere in grado di trovare il punto B (giunto 4-5).
- Un altro problema successivamente sarà far tornare i segni.

1- **Rotazione inversa della matrice tooltip-offest che fa ruotare l'end effector, mentre a me interessa la normale (3 componenti di z) della matrice senza la rotazione del tooltip-offset:**

Nelle matrici di rotazione vale che:

$$R^T R = I$$

$$R^T = R^{-1}$$

Una matrice per la sua inversa corrisponde all'identità ed in questo caso se l'inversa è uguale alla trasposta allora anche una matrice per la sua trasposta corrisponde all'identità

Quindi data l'orientazione zEEzEdell'end effector dopo la "home", moltiplico tale orientazione per la trasposta/inversa della matrice tooltip offset:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Orientazione EE * tooltip-offset(Trasposta) = OrientazioneEE(pura)

Tooltip-offset:

$$\begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2- Partiamo con un esempio per capire meglio, eseguo la classica home ed inseguito ruoto di 90° il giunto 5 in modo da influenzare la posa dell'end effector.

In seguito applicando la rotazione inversa del tooltip-offset quello che ottengo è questa matrice:

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Da qui posso ottenere la normale z = 0 0 1

Ora devo fare il prodotto esterno (cross) tra z e il vettore tra RCM e l'EE. Quindi devo calcolare la distanza tra due punti.
Io credo che essendo:

- RCM = (0 ,0 ,0)
- EE = (0, -0.0091, -0.1044)

Modulo = Il prodotto vettoriale ha come modulo |v|*|w|*sin (alfa)

Direzione = Perpendicolare al piano generato da v,w
Verso = E il verso è deciso tramite terna destrorsa tra v,w

Un modo per poter calcolare il prodotto vettore senza l'angolo alpha è mediante l'utilizzo della seguente formula (**numpy.cross** in python):

$$\vec{v} \times \vec{w} = (y_1 z_2 - y_2 z_1)\hat{i} + (x_2 z_1 - x_1 z_2)\hat{j} + (x_1 y_2 - x_2 y_1)\hat{k}$$

Z x distanza(EE-RCM) = (0.0091, 0, 0)

Ora devo calcolare le equazioni dei due piani (https://www.youmath.it/lezioni/algebra-lineare/geometria-dello-spazio/682-equazione-del-piano.html)

1. Piano azzurro = normale viola e A (EE)
2. Piano grigio = normale verde e l'RCM

Equazione piano 1**: z + 0.1044 = 0**
Equazione piano 2: **x = 0**

https://www.geogebra.org/3d?lang=it

Considerando le cose dette e provando comunque a fare i calcoli con l'asse z = 0 0 1
Espressa rispetto a il SR RCM sarebbe z = 0 1 0, quindi rifacendo i vari calcoli:

- Z x distanza(EE-RCM) = (-0.1044, 0, 0)
- Equazione del piano 1: **Y + 0.0091 = 0**
- Equazione del piano 2: **x = 0**

3- Con questi dati adesso l'obiettivo è di calcolare l'intersezione tra i 2 piani che mi da come risultato una retta (**evidenziata in rosa in figura):**

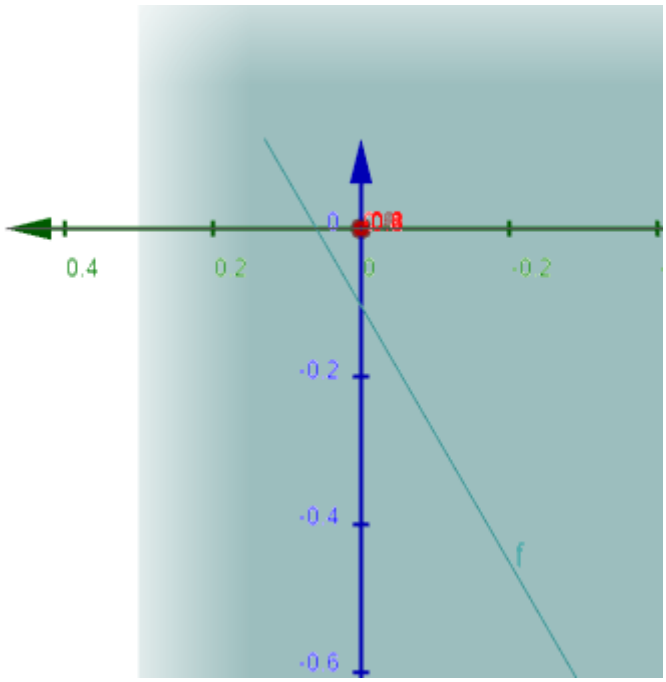
Un modo è mettere a sistema le equazioni dei due piani quindi:

x = 0
y = -0.0091

Quindi fatalità qua è molto facile e l'intersezione tra i due piani è la retta y = -0.0091
Il problema è che vedendo le mie terne dove quella dell'RCM è differente da quella in figura qua sopra, per essere corretti l'intersezione dovrebbe essere z = -0.1044 (o una cosa del genere) perché nel mio sistema l'asse Z di RCM è al posto della Y in figura e quindi erano giusti i calcoli di prima. Cioè:

Equazione piano 1**: z + 0.1044 = 0**
Equazione piano 2: **x = 0**
Da cui la retta z = -0.1044

4- Tale retta mi permette di capire la profondità Z del SR 4-5. Invece per quanto riguarda la x e la y come si fa?



Facendo la prova con 30° queste sono le equazioni dei due piani risultanti:

- -0.86603 y+0.49999 z+0.0522=0
- x = 0

Ho utilizzato la libreria sympy per l'intersezione tra i due piani che mi ritorna 2 punti 3D che appartengono alla retta di intersezione.

Come procedo dopo l'intersezione dei due piani?

https://math.stackexchange.com/questions/83404/finding-a-point-along-a-line-in-three-dimensions-given-two-points

I need to find a point along a line segment in three-dimensional space, given two points.

For example: Find a point along a line segment between point $a(-2,-2,-2)$ and $b(3,3,3)$ which is at distance 3 from point a, in the direction of point b.

I've been trying to figure out a way to do it with three parametric equations, or with similar triangles, but I can't seem to make sense of it.

Any help is greatly appreciated.

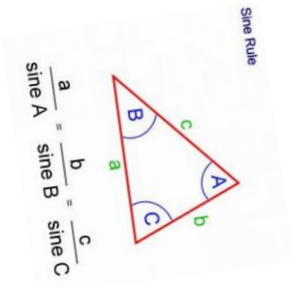
An alternative method even though the question has been answered:

Begin by creating a vector $\overrightarrow{\mathbf{BA}}$ by subtracting $\mathbf{A}$ from $\mathbf{B}$: $\langle 3,3,3\rangle - \langle -2,-2,-2\rangle = \langle 5,5,5\rangle$.
Then, normalize this vector by dividing $\overrightarrow{\mathbf{BA}}$ by its length,
$\frac{\overrightarrow{\mathbf{BA}}}{\|\overrightarrow{\mathbf{BA}}\|} = \frac{\langle 5,5,5\rangle}{\sqrt{5^2+5^2+5^2}} = \langle 0.577, 0.577, 0.577\rangle$. This new unit vector can then be scaled and added to $\mathbf{A}$ to find the point in space at the desired distance. In this case:
$\mathbf{A} + 3\langle 0.577, 0.577, 0.577\rangle = \langle -0.268, -0.268, -0.268\rangle$.

Hope this is useful/correct!

L'idea di procedimento è quella di utilizzare il punto EE e uno dei due punti ottenuti tramite l'intersezione dei piani.
Costruire il vettore direttore della retta passante per quei due punti (non l'equazione della retta). Normalizzare tale vettore e utilizzarlo insieme all'offset = 0.0091 e al punto EE per ottenere il punto del SR4-5.

<u>Come calcolo l'angolo theta5?</u>
https://math.stackexchange.com/questions/59/calculating-an-angle-from-2-points-in-space
https://stackoverflow.com/questions/39497496/how-do-i-retrieve-the-angle-between-two-vectors-3d



Dopo vari aggiustamenti relativi ai segni e provando vari esempi: vedi script.
Sono riuscito ad ottenere i valori dei primi 3 giunti e del quinto.
Nello script c'è anche un caso particolare dove cinemtica diretta/inversa non sono proprio corrette. (da parte del tool)

PROBLEMA:

p.move(PyKDL.Vector(-0.0330523, -0.0987842, -0.121304))

```
Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
            1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[-0.26602072  0.70276222  0.1684399  -0.00000253 -0.70276196  0.26602236]

Current position ---- TOPIC: position_cartesian_current :
[[ -2.6985e-11,           1,-9.04921e-16;
            1,  2.6985e-11, 7.34641e-06;
  7.34641e-06,-7.06677e-16,          -1]
[  -0.0330523,  -0.0987842,   -0.121304]]
Variabili di giunto:
theta1: -0.266017051233
theta2: 0.702758547491
d3: 0.168439903989
theta5: -0.702765606202761
```

p.move_joint_some(numpy.array([-0.234, 0.696, 0.182, -0.702, -2.387, 1.267]), numpy.array([0, 1, 2, 3, 4, 5]))

```
Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
            1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[-0.234  0.696  0.182 -0.702 -2.387  1.267]

Current position ---- TOPIC: position_cartesian_current :
[[    0.662223,    0.554536,    0.503934;
    0.0122769,   -0.680474,    0.732669;
     0.749206,   -0.479004,   -0.457434]
[  -0.0330523,  -0.0987842,   -0.121304]]
Variabili di giunto:
theta1: -0.299023750796
theta2: 0.632907765974
d3: 0.16924694169
theta5: -0.835817558695956
```

Il tutto è dovuto al fatto che la matrice di orientamento dell'EE cambia nei due casi.
E quindi la normale z che ottengo è diversa.
Più che altro non capisco perché l'orientazione del EE cambi solo nella seconda. Cioè perché dopo aver applicato il punto EE con l'inversa, i giunti hanno un loro valore ma l'orientamento dell'EE non cambia, rimane tale quale a quello della p.home.

In questo caso invece, nonostante uso la cinematica diretta e cambio tutti i giunti, e la matrice di rotazione è differente il risultato è corretto. (apparte theta5 che è discorde ma perché il suo segno dipende da theta4 che devo ancora calcolare)

```
Current position ---- TOPIC: position_cartesian_current :
[[-2.69849e-11,           1,-9.91219e-17;
            1, 2.69849e-11, 7.34641e-06;
  7.34641e-06, 9.91194e-17,          -1]
[ 4.16909e-07, 4.50335e-07,     -0.1135]]

Current Joint position ---- TOPIC: position_joint_current(state_joint_current) :
[0.5  0.5  0.05 0.   0.5  2.  ]

Current position ---- TOPIC: position_cartesian_current :
[[    0.403423,   -0.600744,    0.690186;
     0.540308,    0.765145,    0.350171;
    -0.738455,    0.231646,    0.633264]
[   0.0168307,  -0.0241495,  -0.0308081]]

Le componenti della normale z ottenuta sono:
[[-0.40342349]
 [-0.54030849]
 [ 0.7384553 ]]


Variabili di giunto e relativi errori:
theta1: 0.500003673205  errore di  3.6732051030829993e-06
theta2: 0.499996329172  errore di  3.6708282222241984e-06
d3: 0.0499999999553  errore di  4.466790132218179e-11
theta5: -0.499996314179521  errore di  0.999996314179521
```

Ho notato che tipo in questo esempio, se uso theta5 > 90°
In realtà io per calcolare theta5 dovrei fare il complementare quindi 180°.theta5 però devo riuscire a farlo senza conoscere theta5
Questo per theta4 = 0

Se uso theta4 !=0 le cose si complicano ancora. Non basta nemmeno più fare il complementare. L'unico modo in questo caso è utilizzare -0.0091 anziché 0.0091

ALLA FINE HO SCOPERTO CHE I PROBLEMI SONO DUE:

- **NON SO SE 0.0091 O -0.0091 perché NON SO LA DIREZIONE DA EE A SR5**
- **THETA4 Può INFLUENZARE THETA5 QUINDI PER CAPIRE IL SEGNO DI THETA5 DEVO CONOSCERE THETA4**

Anche il modo usato da altair non è molto corretto (caso con theta5 = -2.0) . A volte non da la posa di SR5 corretta.
Il tutto dipende anche dall'ordine del prodotto vettoriale

Facendo il dotproduct è possibile capire che tipo di angolo ho trovato, se positivo o negativo

- Positivo se $|\mathbf{a}| > 0$, $|\mathbf{b}| > 0$ e l'angolo θ è acuto.
- Nullo se $|\mathbf{a}| = 0$, $|\mathbf{b}| = 0$ oppure θ è retto.
- Negativo se $|\mathbf{a}| > 0$, $|\mathbf{b}| > 0$ e l'angolo θ è ottuso.

Se prodotto scalare è:

- **Positivo allora l'angolo è minore di 90**
- **Se è negativo allora l'angolo è maggiore di 90**
- **Se è nullo allora l'angolo è retto**