Andrea Efficace 0300125

Relazione Finale delle Procedure

Robotica Industriale 2020/2021

Questo PDF è una raccolta, commentata, delle procedure assegnate a lezione durante l'anno scolastico indicato sopra. La stesura delle procedure è stata fatta personalmente, confrontando risultati e ragionamenti con alcuni colleghi. In particolare, c'è stata collaborazione con:

Lorenzo Rossi

Alessandro Lomazzo

Gianluca Coccia

Carta d'Identità

COGNOME E NOME DEL PADRE E DELLA MADRE O DI CHI NE FA LE VECI/
FATHER AND MOTHER'S-TUTOR'S NAME

CODICE FISCALE
FISCAL CODE
FFCNDR97E01H501N

ESTREMI ATTO DI NASCITA
00303.1.A66

INDIRIZZO DI RESIDENZA / *RESIDENCE*
VIA DEL MURO LINARI,21/T ROMA (RM)

C<ITACA63937ER9<<<<<<<<<<<<<<<
9705010M3005017ITA<<<<<<<<<<<6
EFFICACE<<ANDREA<<<<<<<<<<<<<<

REPUBBLICA ITALIANA
MINISTERO DELL'INTERNO

CA63937ER

CARTA DI IDENTITÀ / *IDENTITY CARD*

COMUNE DI / *MUNICIPALITY*
ROMA

COGNOME / *SURNAME*
EFFICACE

NOME / *NAME*
ANDREA

LUOGO E DATA DI NASCITA
PLACE AND DATE OF BIRTH
ROMA (RM) 01.05.1997

SESSO / *SEX*: M
STATURA / *HEIGHT*: 178
CITTADINANZA / *NATIONALITY*: ITA

EMISSIONE / *ISSUING*
30.07.2019

SCADENZA / *EXPIRY*
01.05.2030

FIRMA DEL TITOLARE
HOLDER'S SIGNATURE

656029

Procedura 1

Scrivere tre procedure per il calcolo di matrici di rotazione nello spazio di un angolo θ attorno agli assi x, y, z

```
(%i1) Rx(theta):=matrix([1,0,0],
                        [0,cos(theta),-sin(theta)],
                        [0,sin(theta),cos(theta)])
```

(%o1) $\mathrm{Rx}(\vartheta) := \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\vartheta) & -\sin(\vartheta) \\ 0 & \sin(\vartheta) & \cos(\vartheta) \end{pmatrix}$

```
(%i2) Ry(theta):=matrix([cos(theta),0,sin(theta)],
                        [0,1,0],
                        [-sin(theta),0,cos(theta)])
```

(%o2) $\mathrm{Ry}(\vartheta) := \begin{pmatrix} \cos(\vartheta) & 0 & \sin(\vartheta) \\ 0 & 1 & 0 \\ -\sin(\vartheta) & 0 & \cos(\vartheta) \end{pmatrix}$

```
(%i3) Rz(theta):=matrix([cos(theta),-sin(theta),0],
                        [sin(theta),cos(theta),0],
                        [0,0,1])
```

(%o3) $\mathrm{Rz}(\vartheta) := \begin{pmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Procedura 2

Scrivere una procedura per il calcolo di $R_a(\theta)$, dove $a \in \{x, y, z\}$.
Se l'asse cercato non esiste, segnalare l'errore.

```
(%i4) rotationSolver(axis, angle):=block(
       if(axis#x and axis#y and axis#z) then (
        error("Axis not valid, select one from {x,y,z}")),
         if(axis = x) then return(Rx(angle)),
         if(axis = y) then return(Ry(angle)),
         if(axis = z) then return(Rz(angle))
      )$
```

```
(%i5) rotationSolver(x,alpha)
```

(%o5) $\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$

```
(%i6) rotationSolver(y, beta)
```

(%o6) $\begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}$

```
(%i7) rotationSolver(z, gamma)
```

(%o7) $\begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

```
(%i8) rotationSolver(k, zita)
```

```
Axis not valid, select one from {x,y,z}
#0: rotationSolver(axis=k,angle=zita)
 -- an error. To debug this try: debugmode(true);
```

Procedura 3

Dimostrare che $R_x(\alpha)R_y(\beta) \neq R_y(\beta)R_x(\alpha)$ *per valori generici di* α *e* β.
Ritorna vero se il prodotto non commuta.

```
(%i9) commutationTest():=block(
       if(rotationSolver(x,alpha).rotationSolver(y,beta)#rotationSolver(y,
      beta).rotationSolver(x,alpha)) then
         return(true)
         else return(false)
      )$
(%i10) commutationTest()
```

(%o10) **true**

Procedura 4

Utilizzando le procedure precedenti, calcolare $R_z(\gamma)$ $\forall\gamma$, *tale che*

$$R_z(\gamma) = R_x\left(\pm\frac{\pi}{2}\right)R_y(\gamma)R_x\left(\mp\frac{\pi}{2}\right)$$

Verificare poi la correttezza del risultato ottenuto.

```
(%i11) zRotation():=block(
       [xSx,xRx,Ry,Ryn,out1,out2, zeros,Rz],
       zeros:matrix([0,0,0],[0,0,0],[0,0,0]),
       xSx:rotationSolver(x,%pi/2),
       xRx:rotationSolver(x,-%pi/2),
       Ry:rotationSolver(y,gamma),
       Ryn:rotationSolver(y,-gamma),
       out1:xSx.Ry.xRx,
       out2:xRx.Ryn.xSx,
       print(R[z](gamma),"=",xSx,Ry,xRx,"=",out1),
       print(R[z](-gamma),"=",xRx,Ryn,xSx,"=",out2),
       Rz:rotationSolver(z, gamma),
       print(R[z](gamma),"=", Rz),
       if (out1-Rz = zeros and out2-Rz = zeros) then return(true) else
       return(false)
       )$
(%i12) zRotation()
```

$$R_z(\gamma)= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \cos(\gamma) & 0 & \sin(\gamma) \\ 0 & 1 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_z(-\gamma)= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} \cos(\gamma) & 0 & -\sin(\gamma) \\ 0 & 1 & 0 \\ \sin(\gamma) & 0 & \cos(\gamma) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\gamma)= \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(%o12) **true**

Procedura 5

Data una matrice A $n\times n$ scrivere una procedura che controlla se è quadrata e calcola la matrice di rotazione tramite trasformata di Laplace.

$$R_v(\theta)=e^{S(v)\theta}$$

Definisco procedura per creare la matrice antisimmetrica $S(v)$, dato un vettore v.

```
(%i13) S(v):=block(
       [s1,s2,s3,S],
       s1:matrix([0],[v[3,1]],[-v[2,1]]),
       s2:matrix([-v[3,1]],[0],[v[1,1]]),
       s3:matrix([v[2,1]],[-v[1,1]],[0]),
       S:s1,
       S:addcol(S,s2),
       S:addcol(S,s3),
       return(S)
       )$
```

Calcolo S con $v=(1,0,0)^T$. Mi aspetto che la matrice di rotazione sia quindi del tipo:

$$R_x(\theta)=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\vartheta) & -\sin(\vartheta)\\ 0 & \sin(\vartheta) & \cos(\vartheta)\end{pmatrix}$$

```
(%i14) S[v]:S(matrix([1],[0],[0]))
```

(%o14) $\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{pmatrix}$

Definisco procedura per calcolare la dimensione della matrice.

```
(%i15) size(M):=[length(M), length(transpose(M))]$
```

Definisco procedura per il calcolo di e^{At} tramite trasformata di Laplace.

```
(%i16) expLaplace(M):=block(
       [dim, I, sAi, sAiT],
             dim:size(M),
             if(dim[1]#dim[2]) then error("Matrix not Square"),
             I:ident(dim[1]),
             sAi:invert(s*I-M),
             sAiT:sAi,
             for i:1 thru dim[1] do(
              for j:1 thru dim[2] do(
               sAiT[i,j]:ilt(sAi[i,j],s,theta)
              )
             ),
             return(expand(trigreduce(trigsimp(sAiT))))
       )$
```

Eseguo il calcolo della matrice esponenziale.

```
(%i17) expLaplace(S[v])
```

(%o17) $\begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\vartheta) & -\sin(\vartheta)\\ 0 & \sin(\vartheta) & \cos(\vartheta)\end{pmatrix}$

Procedura 6

Scrivere una procedura che utilizzando la forma canonica calcoli la matrice di rotazione attorno all'asse rappresentato dal versore v, di un angolo θ, tale che:

$$S(v) = V \cdot \Lambda \cdot V^{-1}$$
$$R_v(\theta) = V \cdot e^{\Lambda\theta} \cdot V$$

```
(%i18) expVect(M):=block(
       [dim, V, eigVec, Vi, D, expD, res],
             dim:size(M),
             if(dim[1]#dim[2]) then error("Matrix not Square"),
             V:zeromatrix(dim[1],0),
             eigVec:eigenvectors(M),
             for i:1 thru dim[1] do(
              V:addcol(V, transpose(matrix(eigVec[2][i][1])))
             ),
             Vi:invert(V),
             D:Vi.M.V,
             expD:D,
             for i:1 thru dim[1] do(
              expD[i,i]:exp(expD[i,i]*theta)
             ),
             res:V.expD.Vi,
             return(expand(demoivre(res)))
       )$
```

Per il calcolo della matrice esponenziale, uso lo stesso vettore v della *Procedura 5*.

```
(%i19) expVect(S[v])
```

(%o19) $$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\vartheta) & -\sin(\vartheta) \\ 0 & \sin(\vartheta) & \cos(\vartheta) \end{pmatrix}$$

Procedura 7

Scrivere una procedura che calcoli il prodotto vettorial $v \times w$ come prodotto matriciale

$$v \times w = S(v) \cdot w$$

Implemento il prodotto $S(v) \cdot w$

```
(%i20) antiSimProduct(v,w):=block(
       [S,Sw],
        S:S(v),
        Sw:S.w,
        return(Sw)
       )$
```

Testo la procedura con il prodotto vettoriale standard.

```
(%i21) vectProduct(v,w):=block(
       [M,res,det],
        M:matrix([e[1][1],e[2][1],e[3][1]]),
        M:addrow(M,transpose(v),transpose(w)),
        res:zeromatrix(3,1),
        res[1][1]:M[2][2]*M[3][3]-M[2][3]*M[3][2],
        res[2][1]:-(M[2][1]*M[3][3]-M[2][3]*M[3][1]),
        res[3][1]:M[2][1]*M[3][2]-M[2][2]*M[3][1],
        return(res)
       )$
(%i22) v:matrix([v[1]],[v[2]],[v[3]])$
(%i23) w:matrix([w[1]],[w[2]],[w[3]])$
```

```
(%i24) prova:antiSimProduct(v,w)
```

(%o24) $$\begin{pmatrix} v_2\, w_3 - w_2\, v_3 \\ w_1\, v_3 - v_1\, w_3 \\ v_1\, w_2 - w_1\, v_2 \end{pmatrix}$$

```
(%i25) prova:vectProduct(v,w)
```

(%o25) $$\begin{pmatrix} v_2\, w_3 - w_2\, v_3 \\ w_1\, v_3 - v_1\, w_3 \\ v_1\, w_2 - w_1\, v_2 \end{pmatrix}$$

```
(%i26) testProduct(v,w):=block(
       [anti, vect],
                 anti:antiSimProduct(v,w),
                 vect:vectProduct(v,w),
                 if(zeromatrix(3,1) = anti-vect) then true
                 else false
       )$
(%i27) testProduct(v,w)
```

(%o27) **true**

Procedura 8

Scrivere una procedura che implementi la formula di Rodrigues

$$R_v(\theta) = I + S^2(v)(1 - \cos(\theta)) + S(v)\sin(\theta)$$

```
(%i28) rodrigues(v,theta):=block([S,I,S2],
        S:S(v),
        I:ident(size(v)[1]),
        S2:S.S,
        rodr:I+S2*(1-cos(theta))+S*sin(theta),
        return(expand(trigreduce(rodr)))
       )$
```

Definisco vettore v di test.

```
(%i29) v:matrix([1],[0],[0])$
(%i30) rodrigues(v,theta)
```

(%o30) $$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\vartheta) & -\sin(\vartheta) \\ 0 & \sin(\vartheta) & \cos(\vartheta) \end{pmatrix}$$

```
(%i31) rodriguesTest(v,theta):=block(
       [rodr, expLap, dim, S],
              rodr:rodrigues(v,theta),
              S:S(v),
              expLap:expLaplace(S),
              dim:size(v)[1],
              if(zeromatrix(dim,dim) = rodr-expLap) then true
              else false
       )$
(%i32) rodriguesTest(v,theta)
```

(%o32) **true**

Procedura 9

Utilizzando la formula di Rodrigues, calcolare Asse ed Angolo di rotazione.

Definisco procedura per la verifica delle proprietà di rotazione.

```
(%i33) isRot(M):=block(
       [det, Mi, I,dim, MMi, symMMi, symDet],
             dim:size(M),
             if dim[1]#dim[2] then error("Matrix Not Square"),
             I:ident(dim[1]),
             Mi:transpose(M),
             det:trigsimp(trigexpand(trigreduce(determinant(M)))),
             MMi:trigsimp(trigexpand(trigreduce(M.Mi))),
             if((det = 1 or det = -1) and I = MMi) then true
             else false
       )$
```

Definisco la procedura per calcolare la norma di un vettore.

```
(%i34) norm(v):=block(
       [dim, len],
             len:0,
             dim:size(v)[1],
             for i:1 thru dim do(
              len:len+v[i]^2
             ),
             len:sqrt(len),
             return(expand(trigsimp(len)))
       )$
```

Definisco la procedura per determinare l'asse.

```
(%i35) getAxis(R):=block(
       [I, v, ker, dim],
           dim:size(R),
           I:ident(dim[1]),
           if(isRot = false) then
              error("La matrice inserita non è di rotazione.")
           else (
            ker:transpose(adjoint(I-R)),
            zero:zeromatrix(dim[1],1),
            for i:1 thru dim[1] do(
             if(matrix(ker[i])#transpose(zero)) then
               v:ker[i],
               return(trigsimp(trigexpand(trigreduce(v/norm(v)))))
            )
           )
       )$
```

Definisco la procedura per determinare l'angolo.

```
(%i36) getAngle(R):=block(
       [Rplus, Rminus, S1, S2, dim, Rt, v, c, s],
               dim:size(R),
               Rt:transpose(R),
               I:ident(dim[1]),
               Rminus:trigsimp((R-Rt)/2),
               Rplus:trigsimp((R+Rt)/2-I),
               v:transpose(matrix(getAxis(trigsimp(R)))),
               S1:S(v), s:0, c:0,
               if(isRot(R) = false) then
               error("La matrice inserita non è di rotazione."),
               for i:1 thru dim[1] do(
                 for j:1 thru dim[2] do(
                  if(S1[i][j]#0) then
                    s:Rminus[i][j]/S1[i][j],
                    return(s)
                 )
                ),
                S2:S1.S1,
                for i:1 thru dim[1] do(
                 for j:1 thru dim[2] do(
                  if(S2[i][j]#0) then
                    c:1-Rplus[i][j]/S2[i][j],
                    return(c)
                 )
                ),
               return(atan2(expand(s),expand(c)))
       )$
```

Definisco matrice di test.

```
(%i37) R:matrix([0,1,0],[0,0,-1],[-1,0,0])$
```

Definisco procedura per calcolare la lista di parametri per descrivere la rotazione.

```
(%i38) getRotationData(R):=block(
       [v, Theta, param],
           v:getAxis(R),
           Theta:getAngle(R),
           param:[v,Theta],
           return(param)
       )$
(%i39) getRotationData(R)
```

(%o39) $\left[\left[\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right], \frac{2\,\pi}{3}\right]$

Procedura 10

Scrivere una procedura che implementi la parametrizzazione di Cayley:

$$R_v(\theta) = (I + S_v)(I - S_v)^{-1}$$

Dove S è una matrice antisimmetrica.

Definisco procedura per implementare cayley.

```
(%i40) cayley(S):=block(
       [cay, I, dim],
           dim:size(S),
           I:ident(dim[1]),
           cay:(I+S).(invert(I-S)),
           return(cay)
       )$
```

Definisco matrice di tes con $v = (1, 0, 0)^T$

```
(%i41) S:S(matrix([1],[0],[0]))$
(%i42) cayley(S)
```

(%o42) $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$

Procedura 11

Utilizzare le procedure 8 e 9 in cascata: dato asse ed angolo calcolare $R_v(\theta)$ tramite formula di Rodrigues e poi verificare il risultato estraendo asse ed angolo dalla matrice di rotazione ottenuta.

```
(%i43) rodriguesCascade(v, theta):=block(
       [rod, rodInv],
             rod:rodrigues(v,theta),
             print("R[v](theta) = ",rod),
             rodInv:getRotationData(rod),
             print("[asse, angolo] = ",rodInv),
             if(transpose(matrix(rodInv[1])) = v and rodInv[2] = theta) then true
       else false
       )$
```

Definisco vettore ed angolo per testare la procedura.

```
(%i44) v:transpose(1/sqrt(3)*matrix([1,1,-1]))$
(%i45) tht:%pi*2/3$
(%i46) rodriguesCascade(v,tht)
```

R[v](theta) $= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix}$

[asse, angolo] $= \left[\left[\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right], \frac{2\pi}{3}\right]$

(%o46) **true**

Per le procedure successive sono state utilizzate le seguenti funzioni, prese da quelle svolte in precedenza.

Funzioni di utility

1) Determinare le dimensioni di una matrice.

```
(%i1) size(M):=[length(M), length(transpose(M))]$
```

2) Calcolare la matrice antisimmetrica dato un vettore di $\mathbb{R}^3$

```
(%i2) S(v):=block([s1,s2,s3,S],
        s1:matrix([0],[v[3,1]],[-v[2,1]]),
        s2:matrix([-v[3,1]],[0],[v[1,1]]),
        s3:matrix([v[2,1]],[-v[1,1]],[0]),
        S:s1, S:addcol(S,s2), S:addcol(S,s3),
        return(S))$
```

3) Calcolare la matrice di rotazione $R_v(\theta)$ attorno un asse v di un angolo ϑ, in $\mathbb{R}^3$

```
(%i3) rodrigues(v,theta):=block([S,I,S2],
        S:S(v), I:ident(size(v)[1]), S2:S.S,
        rodr:I+S2*(1-cos(theta))+S*sin(theta),
        return(trigsimp(trigexpand(trigreduce(rodr)))))$
```

4) Calcolare la matrice di rotazione $R_x(\theta)$ in $\mathbb{R}^2$

```
(%i4) rot2(theta):=block([rot2],
        rot2:matrix([cos(theta), -sin(theta)],[sin(theta), cos(theta)]),
        return(rot2))$
```

5) Calcolare matrice di rotazione tramite esponenziale di matrice.

```
(%i5) expLaplace(M, theta):=block(
      [dim, I, sAi, sAiT],
            dim:size(M),
            if(dim[1]#dim[2]) then error("Matrix not Square"),
            I:ident(dim[1]),
            sAi:invert(s*I-M),
            sAiT:sAi,
            for i:1 thru dim[1] do(
             for j:1 thru dim[2] do(
              sAiT[i,j]:ilt(sAi[i,j],s,theta)
             )
            ),
            return(expand(trigreduce(trigsimp(sAiT))))
      )$
```

5) Nelle procedure, quando necessario, è stata applicata la ridefinizione della notazione seno e coseno nella forma:

$$\cos(\theta_1 + \cdots + \theta_n) = c_{1\ldots n}$$
$$\sin(\theta_1 + \cdots + \theta_n) = s_{1\ldots n}$$

6) Nelle procedure è stato inserito come argomento la lunghezza dei link dei robot, indicando con L le lunghezze dei link che vengono orientati lungo l'asse Z e con D quelle dei link orientati lungo l'asse X

Procedura 12

Verificare i calcoli della cinematica nel piano.

Calcolare la cinematica di un robot significa determinare le coordinate del punto terminale, detto anche *end-effector*, rispetto alle variabili di giunto del robot. Il numero e il tipo di variabili dipendono dai gradi di libertà del robot stesso.

Robot 1 DOF: *prismatico*

Assumiamo traslazione lungo l'asse e_i

```
(%i6) prism(q1,ei):=block(
      [R, Tr, T01, ex, ey], ex:matrix([1],[0]), ey:matrix([0],[1]),
          if(ei#ex and ei#ey) then
             error("L'asse inserito non è né X né Y"),
          R:ident(2),
          T01:R,
          Tr:q1*ei,
          T01:addcol(T01,Tr),
          T01:addrow(T01,matrix([0,0,1]))
      )$
```

Definisco l'asse su cui fare la traslazione.

```
(%i7) e[x]:matrix([1],[0])$
(%i8) prism(q[1],e[x])
```

(%o8) $$\begin{pmatrix} 1 & 0 & q_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Robot 1 DOF: *rotoidale*

Assumiamo che la coordinata dell'end-effector sia la lunghezza del link del robot L

```
(%i9) roto(q1,L):=block(
      [R, Tr, T01],
          R:rot2(q1),
          R:addcol(R, matrix([0],[0])),
          R:addrow(R,matrix([0,0,1])),
          Tr:prism(L,matrix([1],[0])),
          T01:R.Tr,
          return(T01)
      )$
(%i10) roto(q[1],L)
```

(%o10) $$\begin{pmatrix} \cos(q_1) & -\sin(q_1) & \cos(q_1)\, L \\ \sin(q_1) & \cos(q_1) & \sin(q_1)\, L \\ 0 & 0 & 1 \end{pmatrix}$$

Robot 2 DOF: *rotoidale + rotoidale*

```
(%i11) roto2(q1, q2, L1, L2):=block(
       [T01, T12, T02],
          T01:roto(q1,L1),
          T12:roto(q2,L2),
          T02:T01.T12,
          return(trigreduce(T02))
       )$
(%i12) roto2(q[1],q[2],L[1],L[2])
```

(%o12) $$\begin{pmatrix} \cos(q_2+q_1) & -\sin(q_2+q_1) & L_2\cos(q_2+q_1)+L_1\cos(q_1) \\ \sin(q_2+q_1) & \cos(q_2+q_1) & L_2\sin(q_2+q_1)+L_1\sin(q_1) \\ 0 & 0 & 1 \end{pmatrix}$$

Robot 3 DOF: *rotoidale + rotoidale + rotoidale*

```
(%i13) roto3(q1,q2,q3,L1,L2,L3):=block(
       [T01, T12, T23, T03],
             T01:roto(q1,L1),
             T12:roto(q2,L2),
             T23:roto(q3,L3),
             T03:T01.T12.T23,
             T03:trigreduce(expand(T03)),
             let(cos(q1+q2+q3),c[123]),
             let(sin(q1+q2+q3),s[123]),
             return(letsimp(T03))
       )$
(%i14) roto3(q[1],q[2],q[3],L[1],L[2],L[3])
```

(%o14) $$\begin{pmatrix} c_{123} & -s_{123} & L_2 \cos(q_2 + q_1) + L_1 \cos(q_1) + L_3 c_{123} \\ s_{123} & c_{123} & L_2 \sin(q_2 + q_1) + L_1 \sin(q_1) + L_3 s_{123} \\ 0 & 0 & 1 \end{pmatrix}$$

Robot Cartesiano: *prismatico + prismatico*

Il robot cartesiano monta due giunti prismatici lungo gli assi coordinati x e y.
Nella seguente procedura è implementata la configurazione con asse y come base e asse x come end-effector.

```
(%i15) cartesiano(q1, q2):=block(
       [T01, T12, T02],
             T01:prism(q1,matrix([0],[1])),
             T12:prism(q2,matrix([1],[0])),
             T02:T01.T12,
             return(T02)
       )$
(%i16) cartesiano(q[1],q[2])
```

(%o16) $$\begin{pmatrix} 1 & 0 & q_2 \\ 0 & 1 & q_1 \\ 0 & 0 & 1 \end{pmatrix}$$

Robot 2 DOF: *rotoidale + prismatico*

Il robot di questo tipo visto a lezione vede il giunto prismatico muoversi lungo l'asse y.

```
(%i17) rotoprism(q1,q2,L1):=block(
       [T01, T12, T02],
             T01:roto(q1,L1),
             T12:prism(q2,matrix([0],[1])),
             T02:T01.T12,
             return(T02)
       )$
(%i18) rotoprism(theta[1],q[2],L[1])
```

(%o18) $$\begin{pmatrix} \cos(\vartheta_1) & -\sin(\vartheta_1) & L_1 \cos(\vartheta_1) - q_2 \sin(\vartheta_1) \\ \sin(\vartheta_1) & \cos(\vartheta_1) & L_1 \sin(\vartheta_1) + q_2 \cos(\vartheta_1) \\ 0 & 0 & 1 \end{pmatrix}$$

Robot 2 DOF: *prismatico + rotoidale*

Il robot di questo tipo visto a lezione vede il giunto prismatico muoversi lungo l'asse x

```
(%i19) prismroto(q1,q2,L2):=block(
       [T01, T12, T02],
             T01:prism(q1,matrix([1],[0])),
             T12:roto(q2,L2),
             T02:T01.T12,
             return(T02)
       )$
```

```
(%i20) prismroto(q[1], q[2], L[2])
```

(%o20) $$\begin{pmatrix} \cos(q_2) & -\sin(q_2) & L_2\cos(q_2)+q_1 \\ \sin(q_2) & \cos(q_2) & L_2\sin(q_2) \\ 0 & 0 & 1 \end{pmatrix}$$

Procedura 13

Scrivere una procedura che, dati asse v, angolo ϑ e spostamento d, generi la corrispondente matrice di avvitamento:

$$\mathrm{Av}(v,\theta,d)=\begin{pmatrix} e^{S_v(\theta)} & d\,v \\ 0 & 1 \end{pmatrix}$$

```
(%i21) Av(v, theta, d):=block(
       [exp, S, dv, row, T],
            S:S(v),
            exp:expLaplace(S, theta),
            row:matrix([0,0,0,1]),
            dv:d*v,
            T:addcol(exp, dv),
            T:addrow(T,row),
            return(trigsimp(trigrat(trigreduce(trigexpand(T)))))
       )$
```

Definisco vettore di simbolico.

```
(%i22) assume(v[1]^2+v[2]^2+v[3]^2>0)$
(%i23) v:matrix([v[1]],[v[2]],[v[3]])$
```

Calcolo matrice di avvitamento simbolica. Output soppresso per leggibilità.

```
(%i24) Av(v,theta,d)$
```

Procedura 14

Utilizzando la procedura 13, calcolare

$$\mathrm{Av}(z,\theta,d)$$
$$\mathrm{Av}(x,\alpha,a)$$

```
(%i25) Avz(theta,d):=block(
       [z, Av],
          z:matrix([0],[0],[1]),
          Av:Av(z,theta,d),
          return(Av)
       )$
(%i26) Avz(theta,d)
```

(%o26) $$\begin{pmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
(%i27) Avx(alpha,a):=block(
       [x,Av],
         x:matrix([1],[0],[0]),
         Av:Av(x,alpha,a),
         return(Av)
       )$
(%i28) Avx(alpha,a)
```

(%o28) $$\begin{pmatrix} 1 & 0 & 0 & a \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Procedura 15

Utilizzare la procedura precedente, calcolare

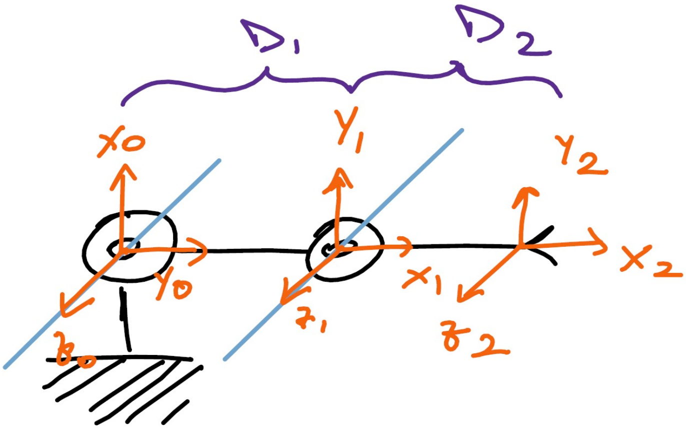
$$Q_{i-1,i} = \mathrm{Av}(z, \theta_i, d_i)\mathrm{Av}(x, \alpha_i, a_i)$$

```
(%i29) Qi1_i(thetai,di,alphai,ai):=block(
       [Avz,Avx],
         Avz:Avz(thetai,di),
         Avx:Avx(alphai,ai),
         Q:Avz.Avx,
         return(Q)
       )$
(%i30) Qi1_i(theta[i],d[i],alpha[i],a[i])
```

(%o30)
$$\begin{pmatrix} \cos(\vartheta_i) & -\cos(\alpha_i)\sin(\vartheta_i) & \sin(\alpha_i)\sin(\vartheta_i) & a_i\cos(\vartheta_i) \\ \sin(\vartheta_i) & \cos(\alpha_i)\cos(\vartheta_i) & -\sin(\alpha_i)\cos(\vartheta_i) & a_i\sin(\vartheta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Robot 2 DOF Planare

Il robot planare a due gradi di libertà vede due giunti di tipo rotoidale a pivot. L'applicazione dell'algoritmo di Denavit-Hartenberg è illustrato in figura.



	θ	d	α	a
1	q_1	0	0	L_1
2	q_2	0	0	L_2

Tabella 1. D - H per Robot 2 DoF Planare

```
(%i9) twoDofPlanar(q1,q2,L1,L2):=block(
      [Q01, Q12, Q02],
            Q01:Q(q1,0,0,L1),
            Q12:Q(q2,0,0,L2),
            Q02:trigreduce(Q01.Q12),
            return(Q02)
      )$
(%i10) DH:twoDofPlanar(q[1],q[2],L[1],L[2])$
(%i11) ridef(DH, q[1],q[2])
```

(%o11)
$$\begin{pmatrix} c_{12} & -s_{12} & 0 & L_2\,c_{12}+L_1\,c_1 \\ s_{12} & c_{12} & 0 & L_2\,s_{12}+L_1\,s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Posizione:

Poiché abbiamo a disposizione solo due gradi di libertà, la cinematica inversa di posizione ci fornisce tutti i parametri necessari per studiare il problema della cinematica inversa del 2DoF Planare.

Dalla cinematica diretta risulta che:

$$d = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} L_2\, c_{12} + L_1\, c_1 \\ L_2\, s_{12} + L_1\, s_1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = R(q_1 + q_2) \begin{pmatrix} L_2 \\ 0 \end{pmatrix} + R(q_1) \begin{pmatrix} L_1 \\ 0 \end{pmatrix}$$

Poiché $R(q_1 + q_2) = R(q_1) R(q_2)$ possiamo raccogliere $R(q_1)$ e sfruttare la proprietà che le rotazioni non modificano la norma:

$$\left\| \begin{pmatrix} x \\ y \end{pmatrix} \right\|_2 = \left\| R(q_2) \begin{pmatrix} L_2 \\ 0 \end{pmatrix} + \begin{pmatrix} L_1 \\ 0 \end{pmatrix} \right\|_2$$

$$x^2 + y^2 = \begin{pmatrix} L_2 & 0 \end{pmatrix} R(q_2)^T R(q_2) \begin{pmatrix} L_2 \\ 0 \end{pmatrix} + \begin{pmatrix} L_1 & 0 \end{pmatrix} \begin{pmatrix} L_1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} L_1 & 0 \end{pmatrix} R(q_2) \begin{pmatrix} L_2 \\ 0 \end{pmatrix}$$

$$x^2 + y^2 = L_2^2 + L_1^2 + 2 L_1 L_2 \cos(q_2)$$

$$\cos(q_2) = \frac{x^2 + y^2 - L_2^2 - L_1^2}{2 L_1 L_2} = c_2$$

Poiché $-1 \leqslant c_2 \leqslant 1$, otteniamo le disequazioni dello spazio operativo del 2DOF:

$$-1 \leqslant \frac{x^2 + y^2 - L_2^2 - L_1^2}{2 L_1 L_2} \leqslant 1$$

$$L_2^2 + L_1^2 - 2 L_1 L_2 \leqslant x^2 + y^2 \leqslant 2 L_1 L_2 + L_2^2 + L_1^2$$

$$(L_1 - L_2)^2 \leqslant x^2 + y^2 \leqslant (L_1 + L_2)^2$$

Queste disequazioni descrivono una corona circolare di circonferenze $r_1 = |L_1 - L_2|$ e $r_2 = L_1 + L_2$, all'interno della quale il robot può operare.
Possiamo dedurre quindi l'espressione per $\sin(q_2)$:

$$\sin(q_2) = s_2 = \pm \sqrt{1 - c_2^2} = \pm \sqrt{1 - \left(\frac{x^2 + y^2 - L_2^2 - L_1^2}{2 L_1 L_2} \right)^2}$$

Possiamo capire che se $\left| \frac{x^2 + y^2 - L_2^2 - L_1^2}{2 L_1 L_2} \right| = 1$ abbiamo una soluzione singolare che coincide con $s_2 = 0$, ovvero con la posizione del secondo link completamente allungata o sovrapposta al primo. Tali posizioni coincidono con i bordi delle circonferenze della corona.
Inoltre, se $L_1 = L_2$ abbiamo infinite soluzioni.
Se $\left| \frac{x^2 + y^2 - L_2^2 - L_1^2}{2 L_1 L_2} \right| \neq 1$, abbiamo due soluzioni per q_2:

$$q_2 = \operatorname{atan2}\left(\pm \sqrt{1 - \left(\frac{x^2 + y^2 - L_2^2 - L_1^2}{2 L_1 L_2} \right)^2}, \frac{x^2 + y^2 - L_2^2 - L_1^2}{2 L_1 L_2} \right)$$

La matrice di rotazione R_2 è nota, pertanto:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{pmatrix} \begin{pmatrix} L_2 c_2 + L_1 \\ s_2 L_2 \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix}$$

Risolvendo in funzione di q_1:

$$\begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \frac{1}{A^2 + B^2} \begin{pmatrix} A & B \\ -B & A \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Poiché $A^2 + B^2 > 0$, deduciamo che

$$q_1 = \operatorname{atan2}(s_1, c_1) = \operatorname{atan2}(-\mathrm{Bx} + \mathrm{Ay}, \mathrm{Ax} + \mathrm{By})$$

$$\text{con } A = L_2 \cos(q_2) + L_1, B = L_2 \sin(q_2)$$

(%i12) posInv(x,y,L1,L2):=block(

```
[q1, s1, c1, q2, s2, c2,Pdes, des, sist, tn1, tn2],
     Pdes:matrix([x,y]).matrix([x],[y]),
     des:x^2+y^2,
     if(des>(L1+L2)^2 or des<(L1-L2)^2) then
       error("Spazio Operativo Violato"),
     c2:(des-L1^2-L2^2)/(2*L1*L2),
     s2:sqrt(1-c2^2),
     q2:[atan2(s2,c2),atan2(-s2,c2)],
     print(q[2],"=",q2),
     tn1:rot2(q2[1]).matrix([L2],[0])+matrix([L1],[0]),
     tn2:rot2(q2[2]).matrix([L2],[0])+matrix([L1],[0]),
     s1:[-tn1[2][1]*x+tn1[1][1]*y,-tn2[2][1]*x+tn2[1][1]*y],
     c1:[tn1[1][1]*x+tn1[2][1]*y,tn2[1][1]*x+tn2[2][1]*y],
     q1:[atan2(s1[1],c1[1]),atan2(s1[2],c1[2])],
     print(q[1],"=",q1),
     return([matrix([q1[1]],[q2[1]]),matrix([q1[2]],[q2[2]])])
)$
```

(%i13) posInv(x,y,L[1],L[2])$

$$q_2 = \left[\operatorname{atan2}\left(\sqrt{1-\frac{(y^2+x^2-L_2^2-L_1^2)^2}{4\,L_1^2\,L_2^2}}, \frac{y^2+x^2-L_2^2-L_1^2}{2\,L_1\,L_2}\right), -\operatorname{atan2}\left(\sqrt{1-\frac{(y^2+x^2-L_2^2-L_1^2)^2}{4\,L_1^2\,L_2^2}}, \frac{y^2+x^2-L_2^2-L_1^2}{2\,L_1\,L_2}\right)\right]$$

$$q_1 = \left[-\operatorname{atan2}\left(L_2\,x\sqrt{1-\frac{(y^2+x^2-L_2^2-L_1^2)^2}{4\,L_1^2\,L_2^2}} - y\left(\frac{y^2+x^2-L_2^2-L_1^2}{2\,L_1}+L_1\right), L_2\,y\sqrt{1-\frac{(y^2+x^2-L_2^2-L_1^2)^2}{4\,L_1^2\,L_2^2}} + x\left(\frac{y^2+x^2-L_2^2-L_1^2}{2\,L_1}+L_1\right)\right), \operatorname{atan2}\left(L_2\,x\sqrt{1-\frac{(y^2+x^2-L_2^2-L_1^2)^2}{4\,L_1^2\,L_2^2}} + y\left(\frac{y^2+x^2-L_2^2-L_1^2}{2\,L_1}+L_1\right), x\left(\frac{y^2+x^2-L_2^2-L_1^2}{2\,L_1}+L_1\right) - L_2\,y\sqrt{1-\frac{(y^2+x^2-L_2^2-L_1^2)^2}{4\,L_1^2\,L_2^2}}\right)\right]$$

(%i14) sol:posInv(1,1,1,1)

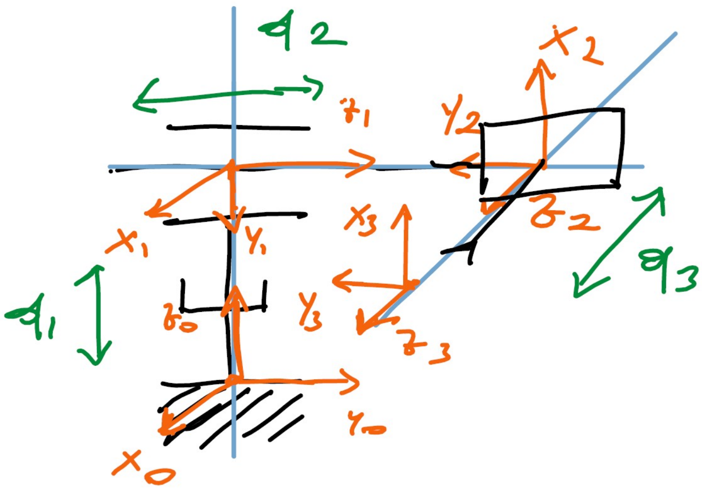
$$q_2 = \left[\frac{\pi}{2}, -\frac{\pi}{2}\right]$$

$$q_1 = \left[0, \frac{\pi}{2}\right]$$

(%o14) $$\left[\begin{pmatrix} 0 \\ \frac{\pi}{2} \end{pmatrix}, \begin{pmatrix} \frac{\pi}{2} \\ -\frac{\pi}{2} \end{pmatrix}\right]$$

Robot Cartesiano

Il robot cartesiano visto a lezione ha un giunto teleferico per il posizionamento verticale e due gradi di libertà, dati da un giunto trapezoidale e un secondo giutno teleferico, che permettono all'end-effector di raggiungere tutti i punti del piano cartesiano individuato dalla lunghezza dei suoi link. L'algoritmo di Denavit-Hartenberg è applicato come in figura.



	θ	d	α	a
1	0	q_1	$-\frac{\pi}{2}$	0
2	$-\frac{\pi}{2}$	q_2	$-\frac{\pi}{2}$	0
3	0	q_3	0	0

Tabella 1. D - H per Robot Cartesiano

```
(%i8) cartesiano(q1,q2,q3):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(0,q1,-%pi/2,0),
            Q12:Q(-%pi/2, q2, -%pi/2, 0),
            Q23:Q(0,q3,0,0),
            Q03:trigreduce(Q01.Q12).Q23,
            return(Q03)
      )$
```

```
(%i9) DH:cartesiano(q[1],q[2],q[3])
```

(%o9) $$\begin{pmatrix} 0 & 0 & 1 & q_3 \\ 0 & -1 & 0 & q_2 \\ 1 & 0 & 0 & q_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Posizione:
Dalla cinematica diretta risulta evidente che:

$$\begin{cases} x = q_3 \\ y = q_2 \\ z = q_1 \end{cases}$$

Tale soluzione è unica e ci dice che lo spazio di lavoro, a meno di fine corsa dei giunti, è tutto $\mathbb{R}^3$.

```
(%i10) posInv(x,y,z):=block(
       [q1, q2, q3],
            q1:z,
            q2:y,
            q3:x,
            return([q1,q2,q3])
       )$
(%i11) posInv(x,y,z)
```

(%o11) $[z, y, x]$

Cinematica Inversa di Orientamento.
Poiché la matrice di rotazione del robot cartesiano presenta una singolarità per la terna $\{z, y, x\}$, data dal valore $-s_y = 1$, adottiamo una terna nautica di tipo $\{x, z, y\}$ che non presenta il termine singolare. Tale scelta è puramente arbitraria e basata su conti preliminari.

$$R_{\{x,z,y\}} = R_x(\alpha) \cdot R_z(\gamma) \cdot R_y(\beta) = \begin{pmatrix} c_\beta \, c_\gamma & -s_\gamma & s_\beta \, c_\gamma \\ c_\alpha \, c_\beta \, s_\gamma + s_\alpha \, s_\beta & c_\alpha \, c_\gamma & c_\alpha \, s_\beta \, s_\gamma - s_\alpha \, c_\beta \\ s_\alpha \, c_\beta \, s_\gamma - c_\alpha \, s_\beta & s_\alpha \, c_\gamma & s_\alpha \, s_\beta \, s_\gamma + c_\alpha \, c_\beta \end{pmatrix}$$

Dalla cinematica diretta si vede che la matrice di rotazione è:

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

Da cui:

$$-s_\gamma = 0 \rightarrow c_\gamma = \pm\sqrt{1 - s_\gamma^2} = \pm 1 \rightarrow \gamma = \operatorname{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

$$\begin{cases} c_\alpha c_\gamma = -1 \\ s_\alpha c_\gamma = 0 \end{cases} \rightarrow \begin{cases} \pm c_\alpha = -1 \rightarrow c_\alpha = \mp 1 \\ s_\alpha = 0 \end{cases} \rightarrow \alpha = \operatorname{atan2}(0, \mp 1) = \begin{cases} \pi \\ 0 \end{cases}$$

$$\begin{cases} c_\beta c_\gamma = 0 \\ s_\beta c_\gamma = 1 \end{cases} \rightarrow \begin{cases} c_\beta = 0 \\ \pm s_\beta = 1 \rightarrow s_\beta = \pm 1 \end{cases} \rightarrow \beta = \operatorname{atan2}(\pm 1, 0) = \begin{cases} \frac{\pi}{2} \\ -\frac{\pi}{2} \end{cases}$$

Abbiamo quindi una soluzione per un γ fissato. Riassumendo:

$$\begin{pmatrix} \pi \\ \frac{\pi}{2} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -\frac{\pi}{2} \\ \pi \end{pmatrix}$$

```
(%i12) R(alpha,beta,gamma):=rodrigues(matrix([1],[0],[0]),alpha).
        rodrigues(matrix([0],[0],[1]),gamma).
        rodrigues(matrix([0],[1],[0]),beta)$
```

```
(%i13) orientInv(R):=block(
       [cx, sx, cy, sy, cz, sz, x, y, z],
            sz:R[1][2],
              if(sz=1 or sz=-1) then
                error("Caso Singolare, cambiare terna"),
              cz:sqrt(1-sz^2),
              z:[atan2(sz,cz),atan2(sz,-cz)],
              print(phi[Z],"=",z),
              cy:R[1][1],
              sy:[R[1][3]/cos(z[1]),R[1][3]/cos(z[2])],
              y:[atan2(sy[1],cy),atan2(sy[2],cy)],
              print(phi[Y],"=",y),
              cx:[R[2][2]/cos(z[1]),R[2][2]/cos(z[2])],
              sx:0,
              x:[atan2(sx,cx[1]),atan2(sx,cx[2])],
              print(phi[X],"=",x),
              return([matrix([x[1]],[y[1]],[z[1]]),
                    matrix([x[2]],[y[2]],[z[2]])])
       )$
(%i14) R:submatrix(4,DH,4)
```

(%o14) $$\begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

```
(%i15) orientInv(R)
```

φ_Z= $[0, \pi]$
φ_Y= $\left[\frac{\pi}{2}, -\frac{\pi}{2}\right]$
φ_X= $[\pi, 0]$

(%o15) $$\left[\begin{pmatrix} \pi \\ \frac{\pi}{2} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -\frac{\pi}{2} \\ \pi \end{pmatrix}\right]$$

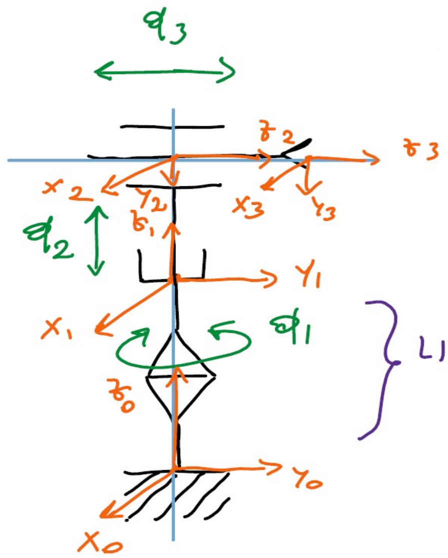
```
(%i16) R(%pi,%pi/2,0)
```

(%o16) $$\begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

```
(%i17)
```

Robot Cilindrico

Il robot cilindrico vede un giunto rotoidale di tipo pivot, un giunto lineare teleferico e uno trapezoidale. Il suo spazio di lavoro è formato dal luogo geometrico dei punti di un cilindro. L'applicazione dell'algoritmo di Denavit-Hartenberg è illustrata in figura.



	θ	d	α	a
1	q_1	L_1	0	0
2	0	q_2	$-\frac{\pi}{2}$	0
3	0	q_3	0	0

Tabella 1. D - H per Robot Cilindrico

```
(%i9) cilindrico(q1,q2,q3,L1):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(q1,L1,0,0),
            Q12:Q(0,q2,-%pi/2,0),
            Q23:Q(0,q3,0,0),
            Q03:Q01.Q12.Q23,
            return(Q03)
      )$
(%i10) DH:cilindrico(q[1],q[2],q[3],L[1])$
(%i19) ridef(DH,2)
```

$$(\%o19)\quad \begin{pmatrix} \cos(q_1) & 0 & -\sin(q_1) & -q_3\sin(q_1) \\ \sin(q_1) & 0 & \cos(q_1) & q_3\cos(q_1) \\ 0 & -1 & 0 & q_2+L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Posizione:
Dalla cinematica diretta capiamo che la posizione dell'end-effector è descritta da:

$$\begin{cases} x = -q_3 \sin(q_1) \\ y = q_3 \cos(q_1) \\ z = q_2 + L_1 \end{cases}$$

Poiché z ed L_1 devono essere noti, abbiamo una possibile soluzione per q_2:

$$q_2 = z - L_1$$

Quadrando e sommando le prime due equazioni otteniamo:

$$x^2 + y^2 = q_3^2 (s_1^2 + c_1^2) = q_3^2$$

Questa equazione, insieme a quella su q_2, ci permette di determinare lo spazio operativo del robot, che corrispnde ad un cilindro con circonferenza ed altezza variabili, determinati nella realtà dai fine corsa dei due giunti prismatici.
Risolvendo per trovare q_3, determinamo le condizioni di singolarità:

$$q_3 = \pm\sqrt{x^2 + y^2}$$

$$\text{se } q_3 \neq 0 \rightarrow x^2 + y^2 \neq 0 \rightarrow x \neq 0 \text{ oppure } y \neq 0 \rightarrow 2 \text{ soluzioni generiche}$$

Se $x^2 + y^2 \neq 0$, la soluzione trovata di q_3 è ammissibile, quindi:

$$\begin{cases} c_1 = \frac{y}{q_3} \\ s_1 = \frac{-x}{q_3} \end{cases} \rightarrow q_1 = \operatorname{atan2}\left(\frac{-x}{q_3}, \frac{y}{q_3}\right)$$

$$\text{se } q_3 = 0 \rightarrow x^2 + y^2 = 0 \rightarrow \text{soluzione singolare}$$

La configurazione assunta dal robot in posizione singolare coincide con l'asse del cilindro, che comporta infinite soluzioni di q_1.
Riassumendo:

$$\begin{pmatrix} \operatorname{atan2}\left(\frac{-x}{q_3}, \frac{y}{q_3}\right) \\ z - L_1 \\ \sqrt{x^2 + y^2} \end{pmatrix}, \begin{pmatrix} \operatorname{atan2}\left(\frac{-x}{q_3}, \frac{y}{q_3}\right) \\ z - L_1 \\ \sqrt{x^2 + y^2} \end{pmatrix}$$

```
(%i12) posInv(x,y,z,L1):=block(
       [q1, q2, q3, sist, des],
            des:trigsimp(x^2+y^2),
            sist:[z=q2+L1],
            q2:map(rhs,solve(sist,q2))[1],
            print(q[2],"=", q2),
            sist:[des=q3^2],
            q3:map(rhs, solve(sist,q3)),
            print(q[3],"=",q3),
            if(des = 0) then (
              error("Spazio Operativo Violato: q[1] ha infinite soluzioni")
            ) else (
              q1:[atan2(-x/q3[1],y/q3[1]), atan2(-x/q3[2],y/q3[2])]
            ),
            print(q[1],"=",q1),
            return([matrix([q1[1]],[q2], [q3[1]]),matrix([q1[2]],[q2], [q3[2]])])

       )$
```

```
(%i13) posInv(5,10,15,L[1])
```

$q_2 = 15 - L_1$
$q_3 = \left[-5^{\frac{3}{2}}, 5^{\frac{3}{2}}\right]$
$q_1 = \left[\pi - \arctan\left(\frac{1}{2}\right), -\arctan\left(\frac{1}{2}\right)\right]$

$$\text{(\%o13)} \quad \left[\begin{pmatrix} \pi - \arctan\left(\frac{1}{2}\right) \\ 15 - L_1 \\ -5^{\frac{3}{2}} \end{pmatrix}, \begin{pmatrix} -\arctan\left(\frac{1}{2}\right) \\ 15 - L_1 \\ 5^{\frac{3}{2}} \end{pmatrix}\right]$$

```
(%i15) test:cilindrico(%pi-atan ((1/2)),15-L[1],-5^((3/2)),L[1])
```

$$\text{(\%o15)} \quad \begin{pmatrix} -\frac{2}{\sqrt{5}} & 0 & -\frac{1}{\sqrt{5}} & 5 \\ \frac{1}{\sqrt{5}} & 0 & -\frac{2}{\sqrt{5}} & 10 \\ 0 & -1 & 0 & 15 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Orientamento:
In questo caso non siamo in condizione singolare per la terna $\{z, y, x\}$, quindi:

$$R_{\{z,y,x\}}(\phi) = R_z(\gamma) R_y(\beta) R_x(\alpha) = \begin{pmatrix} c_\beta c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma & s_\alpha s_\gamma + c_\alpha s_\beta c_\gamma \\ c_\beta s_\gamma & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma \\ -s_\beta & s_\alpha c_\beta & c_\alpha c_\beta \end{pmatrix}$$

Dalla cinematica diretta, la matrice di rotazione è:

$$R = \begin{pmatrix} c_1 & 0 & -s_1 \\ s_1 & 0 & c_1 \\ 0 & -1 & 0 \end{pmatrix}$$

Osserviamo che l'affermazione sulla non singolarità è confermata, poiché $-s_\beta = 0 \neq \pm 1$.
Pertanto:

$$-s_\beta = 0 \rightarrow c_\beta = \pm\sqrt{1 - s_\beta^2} = \pm 1 \rightarrow \beta = \begin{cases} 0 \\ \pi \end{cases}$$

Scrivendo gli altri due sistemi di equazioni troviamo:

$$\begin{cases} c_\beta c_\gamma = c_1 \\ c_\beta s_\gamma = s_1 \end{cases} \rightarrow \begin{cases} \pm c_\gamma = c_1 \rightarrow c_\gamma = \pm c_1 \\ \pm s_\gamma = s_1 \rightarrow s_\gamma = \pm s_1 \end{cases} \rightarrow \gamma = \operatorname{atan2}(\pm s_1, \pm c_1) = \begin{cases} q_1 \\ q_1 + \pi \end{cases}$$

$$\begin{cases} s_\alpha c_\beta = -1 \\ c_\alpha c_\beta = 0 \end{cases} \rightarrow \begin{cases} \pm s_\alpha = -1 \rightarrow s_\alpha = \mp 1 \\ \pm c_\alpha = 0 \rightarrow c_\alpha = 0 \end{cases} \rightarrow \alpha = \operatorname{atan2}(\mp 1, 0) = \begin{cases} -\frac{\pi}{2} \\ \frac{\pi}{2} \end{cases}$$

Riassumendo:

$$\begin{pmatrix} -\frac{\pi}{2} \\ 0 \\ q_1 \end{pmatrix}, \begin{pmatrix} \frac{\pi}{2} \\ \pi \\ q_1 + \pi \end{pmatrix}$$

```
(%i26) R(alpha,beta,gamma):=rodrigues(matrix([0],[0],[1]),gamma).
        rodrigues(matrix([0],[1],[0]),beta).rodrigues(matrix([1],[0],[0]),alpha)$
```

```
(%i21) orientInv(R):=block(
       [cx, sx, cy, sy, cz, sz, x, y, z],
            sy:R[3][1],
            cy:sqrt(1-sy^2),
            y:[atan2(sy,cy),atan2(sy,-cy)],
            print(phi[Y],"=",y),
            sx:[R[3][2]/cos(y[1]),R[3][2]/cos(y[2])],
            cx:R[3][3],
            x:[atan2(sx[1],cx),atan2(sx[2],cx)],
            print(phi[X],"=",x),
            cz:[R[1][1]/cos(y[1]),R[1][1]/cos(y[2])],
            sz:[R[2][1]/cos(y[1]),R[2][1]/cos(y[2])],
            z:[atan2(sz[1],cz[1]),atan2(sz[2],cz[2])],
            print(phi[Z],"=",z),
            return([matrix([x[1]],[y[1]],[z[1]]),
                    matrix([x[2]],[y[2]],[z[2]])])
       )$
(%i22) R:submatrix(4,test,4)
```

(%o22) $$\begin{pmatrix} -\frac{2}{\sqrt{5}} & 0 & -\frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & 0 & -\frac{2}{\sqrt{5}} \\ 0 & -1 & 0 \end{pmatrix}$$

```
(%i23) orientInv(R)
```

$$\varphi_Y = [0, \pi]$$

$$\varphi_X = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

$$\varphi_Z = \left[\pi - \arctan\left(\frac{1}{2}\right), -\arctan\left(\frac{1}{2}\right)\right]$$

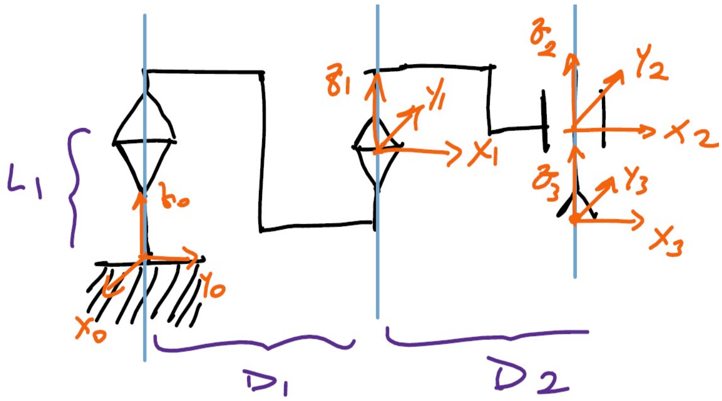
(%o23) $$\left[\begin{pmatrix} -\frac{\pi}{2} \\ 0 \\ \pi - \arctan\left(\frac{1}{2}\right) \end{pmatrix}, \begin{pmatrix} \frac{\pi}{2} \\ \pi \\ -\arctan\left(\frac{1}{2}\right) \end{pmatrix}\right]$$

```
(%i27) R(-(%pi/2), 0, %pi-atan ((1/2)))
```

(%o27) $$\begin{pmatrix} -\frac{2}{\sqrt{5}} & 0 & -\frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & 0 & -\frac{2}{\sqrt{5}} \\ 0 & -1 & 0 \end{pmatrix}$$

Robot Scara

Il robot scara è formato da due giunti rotoidali, e da uno lineare trapezoidale. L'applicazione dell'algoritmo di Denavit-Hartenberg è illustrata in figura.



	θ	d	α	a
1	q_1	L_1	0	D_1
2	q_2	0	0	D_2
3	0	q_3	0	0

Tabella 1. D - H per Robot SCARA

```
(%i9) scara(q1,q2,q3,L1,D1,D2):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(q1,L1,0,D1),
            Q12:Q(q2,0,0,D2),
            Q23:Q(0,q3,0,0),
            Q03:expand(trigreduce(Q01.Q12)).Q23,
            return(Q03)
      )$
(%i10) DH:scara(q[1],q[2],q[3],L[1],D[1],D[2])$
(%i11) ridef(DH,q[1],q[2],q[3])
```

$$(\%o11)\quad \begin{pmatrix} c_{12} & -s_{12} & 0 & D_2\,c_{12}+D_1\,c_1 \\ s_{12} & c_{12} & 0 & D_2\,s_{12}+D_1\,s_1 \\ 0 & 0 & 1 & q_3+L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Posizione:

Dalla cinematica diretta abbiamo che:

$$\begin{cases} x = D_2\,c_{12}+D_1\,c_1 \\ y = D_2\,s_{12}+D_1\,s_1 \\ z = q_3+L_1 \end{cases}$$

Poiché z e L_1 sono noti, possiamo definire una soluzione per q_3:

$$q_3 = z - L_1$$

Nota 1. *E' possibile notare che questa soluzione dipende dalla posizione del SdR inerziale definito durante la procedura di DH. Difatti, posizionando il centro del SdR 0 al centro del primo giunto piuttosto che sulla base, la soluzione diventa* $q_3 = z$.

Prendendo le prime due equazioni e riscrivendole sottoforma di prodotto tra matrici, possiamo raccogliere rispetto a q_1 e scrivere:

$$\begin{pmatrix} x \\ y \end{pmatrix} = R(q_1+q_2)\begin{pmatrix} D_2 \\ 0 \end{pmatrix} + R(q_1)\begin{pmatrix} D_1 \\ 0 \end{pmatrix} = R(q_1)\left\{R(q_2)\begin{pmatrix} D_2 \\ 0 \end{pmatrix} + \begin{pmatrix} D_1 \\ 0 \end{pmatrix}\right\}$$

Nota 2. *E' possibile notare che risolvere questo sistema di equazioni coincide con il problema della cinematica inversa del robot 2DoF-Planare*

Poiché moltiplicare per $R(q_1)$ non altera la norma, abbiamo che:

$$\left\|\begin{pmatrix} x \\ y \end{pmatrix}\right\|_2 = \left\|R(q_2)\begin{pmatrix} D_2 \\ 0 \end{pmatrix} + \begin{pmatrix} D_1 \\ 0 \end{pmatrix}\right\|_2$$

$$x^2+y^2 = \begin{pmatrix} D_2 & 0 \end{pmatrix} R(q_2)^T R(q_2)\begin{pmatrix} D_2 \\ 0 \end{pmatrix} + \begin{pmatrix} D_1 & 0 \end{pmatrix}\begin{pmatrix} D_1 \\ 0 \end{pmatrix} + 2\begin{pmatrix} D_1 & 0 \end{pmatrix} R(q_2)\begin{pmatrix} D_2 \\ 0 \end{pmatrix}$$

$$x^2+y^2 = D_2^2 + D_1^2 + 2D_1D_2\cos(q_2)$$

$$\cos(q_2) = \frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2} = c_2$$

Poiché $-1 \leqslant c_2 \leqslant 1$, otteniamo le disequazioni dello spazio operativo del 2DOF:

$$-1 \leqslant \frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2} \leqslant 1$$

$$D_2^2 + D_1^2 - 2D_1D_2 \leqslant x^2+y^2 \leqslant 2D_1D_2 + D_2^2 + D_1^2$$

$$(D_1-D_2)^2 \leqslant x^2+y^2 \leqslant (D_1+D_2)^2$$

Queste disequazioni descrivono una corona circolare di circonferenze $r_1 = |D_1 - D_2|$ e $r_2 = D_1 + D_2$, all'interno della quale il robot può operare.
Possiamo dedurre quindi l'espressione per $\sin(q_2)$:

$$\sin(q_2) = s_2 = \pm\sqrt{1-c_2^2} = \pm\sqrt{1-\left(\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right)^2}$$

Possiamo capire che se $\left|\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right| = 1$ abbiamo una soluzione singolare che coincide con $s_2 = 0$, ovvero con la posizione del secondo link completamente allungata o sovrapposta al primo. Tali posizioni coincidono con i bordi delle circonferenze della corona.
Inoltre, se $D_1 = D_2$ abbiamo infinite soluzioni.
Se $\left|\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right| \neq 1$, abbiamo due soluzioni per q_2:

$$q_2 = \operatorname{atan2}\left(\pm\sqrt{1-\left(\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right)^2}, \frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right)$$

La matrice di rotazione R_2 è nota, pertanto:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{pmatrix}\begin{pmatrix} L_2c_2 + L_1 \\ s_2L_2 \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix}$$

Risolvendo in funzione di q_1:

$$\begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \frac{1}{A^2+B^2}\begin{pmatrix} A & B \\ -B & A \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

Poiché $A^2+B^2>0$, deduciamo che

$$q_1=\mathrm{atan2}(s_1,c_1)=\mathrm{atan2}(-\mathrm{Bx}+\mathrm{Ay},\mathrm{Ax}+\mathrm{By})$$

$$\text{con } A=D_2\cos(q_2)+D_1, B=D_2\sin(q_2)$$

Riassumendo:

$$\sin(q_1)=\mp D_2x\sqrt{1-\left(\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right)^2}+y\left(\frac{x^2+y^2-D_2^2-D_1^2}{2D_1}+D_1\right)$$

$$\cos(q_1)=D_2x\left(\frac{x^2+y^2-D_2^2-D_1^2}{2D_1}+D_1\right)+y\sqrt{1-\left(\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right)^2}$$

$$q_2=\mathrm{atan2}\left(\pm\sqrt{1-\left(\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right)^2},\frac{x^2+y^2-D_2^2-D_1^2}{2D_1D_2}\right)$$

$$q_3=z-L_1$$

```
(%i12) posInv(x,y,z,L1,D1,D2):=block(
       [q1, s1, c1, q2, s2, c2, q3, quad, sist, a],
            q3:z-L1,
            quad:x^2+y^2,
            if(quad>(D1+D2)^2 or quad<(D1-D2)^2) then
              error("Spazio Operativo Violato"),
            c2:(x^2+y^2-D1^2-D2^2)/(2*D1*D2),
            s2:sqrt(1-c2^2),
            q2:[atan2(s2,c2),atan2(-s2,c2)],
            /*a:ratsubst(a,c2,c2),
            s2:sqrt(1-a^2),
            q2:[atan2(s2,a),atan2(-s2,a)],*/
            A:[(cos(q2[1])*D2)+D1, (cos(q2[2])*D2)+D1],
            B:[sin(q2[1])*D2, sin(q2[2])*D2],
            c1:[A[1]*x+B[1]*y, A[2]*x+B[2]*y],
            s1:[-B[1]*x+A[1]*y, -B[2]*x+A[2]*y],
            q1:[atan2(s1[1],c1[1]),atan2(s1[2],c1[2])],
            return([matrix([q1[1]],[q2[1]],[q3]),matrix([q1[2]],[q2[2]],[q3])])
       )$
(%i13) posInv(x,y,z, L[1], D[1], D[2])
```

(%o13)
$$\left[\begin{pmatrix}-\mathrm{atan2}\left(D_2\,x\,\sqrt{1-\frac{(y^2+x^2-D_2^2-D_1^2)^2}{4\,D_1^2\,D_2^2}}-y\left(\frac{y^2+x^2-D_2^2-D_1^2}{2\,D_1}+D_1\right),\ D_2\,y\,\sqrt{1-\frac{(y^2+x^2-D_2^2-D_1^2)^2}{4\,D_1^2\,D_2^2}}+x\left(\frac{y^2+x^2-D_2^2-D_1^2}{2\,D_1}+D_1\right)\right)\\ \mathrm{atan2}\left(\sqrt{1-\frac{(y^2+x^2-D_2^2-D_1^2)^2}{4\,D_1^2\,D_2^2}},\frac{y^2+x^2-D_2^2-D_1^2}{2\,D_1\,D_2}\right)\\ z-L_1\end{pmatrix},\begin{pmatrix}\mathrm{atan2}\left(D_2\,x\,\sqrt{1-\frac{(y^2+x^2-D_2^2-D_1^2)^2}{4\,D_1^2\,D_2^2}}+y\left(\frac{y^2+x^2-D_2^2-D_1^2}{2\,D_1}+D_1\right),\ x\left(\frac{y^2+x^2-D_2^2-D_1^2}{2\,D_1}+D_1\right)-D_2\,y\,\sqrt{1-\frac{(y^2+x^2-D_2^2-D_1^2)^2}{4\,D_1^2\,D_2^2}}\right)\\ -\mathrm{atan2}\left(\sqrt{1-\frac{(y^2+x^2-D_2^2-D_1^2)^2}{4\,D_1^2\,D_2^2}},\frac{y^2+x^2-D_2^2-D_1^2}{2\,D_1\,D_2}\right)\\ z-L_1\end{pmatrix}\right]$$

```
(%i34) sol:posInv(1,1,1,1,1,1)
```

(%o34) $\left[\begin{pmatrix} 0 \\ \frac{\pi}{2} \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{\pi}{2} \\ -\frac{\pi}{2} \\ 0 \end{pmatrix}\right]$

Verifico il risultato della cinematica inversa

```
(%i35) testPos(sol,L1,D1,D2):=block([i,t],
       for i:1 thru length(sol) do(
       t:scara(sol[i][1][1],sol[i][2][1],sol[i][3][1],L1,D1,D2),
       print("sol[",i,"][1]=",transpose(sol[i]),"->",t)
       )
       )$
(%i38) testPos(sol,1,1,1)
```

sol[1][1]= $\begin{pmatrix} 0 & \frac{\pi}{2} & 0 \end{pmatrix}$-> $\begin{pmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

sol[2][1]= $\begin{pmatrix} \frac{\pi}{2} & -\frac{\pi}{2} & 0 \end{pmatrix}$-> $\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

(%o38) **done**

Cinematica Inversa di Orientamento:
La soluzione della cinematica diretta non è singolare per la terna $\{z, y, x\}$, ovvero:

$$R_{\{z,y,x\}}(\phi) = R_z(\gamma) R_y(\beta) R_x(\alpha) = \begin{pmatrix} c_\beta c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma & s_\alpha s_\gamma + c_\alpha s_\beta c_\gamma \\ c_\beta s_\gamma & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma \\ -s_\beta & s_\alpha c_\beta & c_\alpha c_\beta \end{pmatrix}$$

La matrice di rotazione della cinematica diretta è:

$$R = \begin{pmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Si nota subito che:

$$-s_\beta = 0 \neq \pm 1 \rightarrow c_\beta = \pm\sqrt{1 - s_\beta} = \pm 1 \rightarrow \beta = \operatorname{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

Non siamo in configurazione singolare, quindi tale soluzione è ammissibile. Pertanto:

$$\begin{cases} c_\beta c_\gamma = c_{12} \\ c_\beta s_\gamma = s_{12} \end{cases} \rightarrow \begin{cases} \pm c_\gamma = c_{12} \rightarrow c_\gamma = \pm c_{12} \\ \pm s_\gamma = s_{12} \rightarrow s_\gamma = \pm s_{12} \end{cases} \rightarrow \gamma = \operatorname{atan2}(\pm s_{12}, \pm c_{12}) = \begin{cases} q_1 + q_2 \\ q_1 + q_2 + \pi \end{cases}$$

$$\begin{cases} s_\alpha c_\beta = 0 \\ c_\alpha c_\beta = 1 \end{cases} \rightarrow \begin{cases} \pm s_\alpha = 0 \rightarrow s_\alpha = 0 \\ \pm c_\alpha = 1 \rightarrow c_\alpha = \pm 1 \end{cases} \rightarrow \alpha = \operatorname{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

Riassumendo:

$$\begin{pmatrix} 0 \\ 0 \\ q_1 + q_2 \end{pmatrix}, \begin{pmatrix} \pi \\ \pi \\ q_1 + q_2 + \pi \end{pmatrix}$$

```
(%i37) R(alpha,beta,gamma):=rodrigues(matrix([0],[0],[1]),gamma).
        rodrigues(matrix([0],[1],[0]),beta).rodrigues(matrix([1],[0],[0]),alpha)$
```

(%i17)
```
orientInv(R):=block(
[Rzxy, cx, sx, Phix, cy, sy, Phiy, cz, sz, Phiz],
       sy:-R[3][1],
       if(sy=1 or sy=-1) then error("Terna singolare, vincolo violato."),
       cy:sqrt(1-sy^2),
       Phiy:[atan2(sy,cy), atan2(sy,-cy)],
       print(phi[y], "=", Phiy),
       sx:[R[3][2]/cos(Phiy[1]), R[3][2]/cos(Phiy[2])],
       cx:[R[3][3]/cos(Phiy[1]), R[3][3]/cos(Phiy[2])],
       Phix:[atan2(sx[1],cx[1]),atan2(sx[1],cx[2])],
       print(phi[x], "=", Phix),
       cz:[R[1][1]/cos(Phiy[1]), R[1][1]/cos(Phiy[2])],
       sz:[R[2][1]/cos(Phiy[1]), R[2][1]/cos(Phiy[2])],
       Phiz:[atan2(sz[1],cz[1]),atan2(sz[2],cz[2])],
       print(phi[z], "=", Phiz),
       return([matrix([Phix[1]],[Phiy[1]],[Phiz[1]]),
              matrix([Phix[2]],[Phiy[2]],[Phiz[2]])])
)$
```

(%i40) `T:scara(0, (%pi/2), 0,L[1],D[1],D[2])`

(%o40) $$\begin{pmatrix} 0 & -1 & 0 & D_1 \\ 1 & 0 & 0 & D_2 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(%i42) `orientInv(submatrix(4,T,4))`

$\varphi_y = [0, \pi]$
$\varphi_x = [0, \pi]$
$\varphi_z = \left[\frac{\pi}{2}, -\frac{\pi}{2}\right]$

(%o42) $$\left[\begin{pmatrix} 0 \\ 0 \\ \frac{\pi}{2} \end{pmatrix}, \begin{pmatrix} \pi \\ \pi \\ -\frac{\pi}{2} \end{pmatrix}\right]$$

(%i43) `R(0,0,%pi/2)`

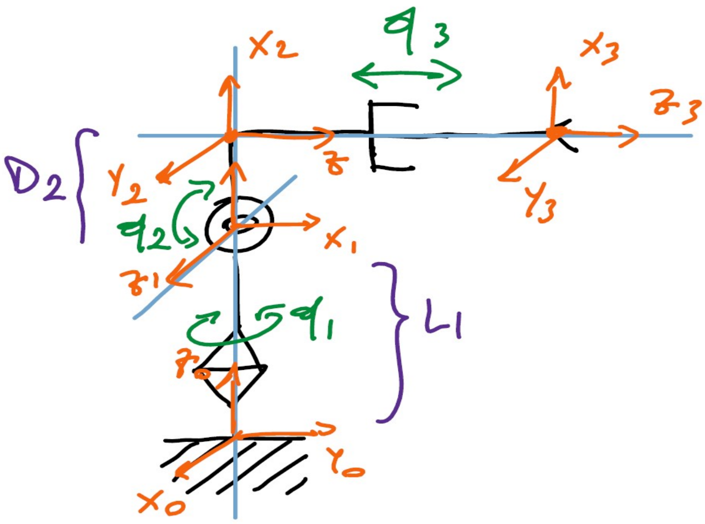
(%o43) $$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(%i44) `R(%pi,%pi,-%pi/2)`

(%o44) $$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Robot Sferico (di primo tipo)

Il robot sferico di primo tipo è formato da due giunti rotoidali di tipo pivot e uno lineare trapezoidale. Il suo spazio di lavoro è formato dal luogo geometrico di punti di una sfera di raggio variabile, dipendente dalla variabile del giunto lineare.
L'algoritmo di Denavit-Hartenberg è applicato come in figura.



	θ	d	α	a
1	q_1	L_1	$\frac{\pi}{2}$	0
2	q_2	0	$\frac{\pi}{2}$	L_2
3	0	q_3	0	0

Tabella 1. D - H per Robot Sferico di Primo Tipo

```
(%i9) sferico(q1,q2,q3,L1,L2):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(q1, L1, %pi/2, 0),
            Q12:Q(q2, 0, %pi/2, L2),
            Q23:Q(0, q3, 0, 0),
            Q03:Q01.Q12.Q23,
            return(Q03)
      )$
(%i10) DH:sferico(q[1],q[2],q[3],L[1],L[2])$
(%i11) ridef(DH,q[1],q[2],q[3])
```

$$(\%o11)\quad \begin{pmatrix} c_1\,c_2 & s_1 & c_1\,s_2 & c_1\,s_2\,q_3+c_1\,L_2\,c_2 \\ s_1\,c_2 & -c_1 & s_1\,s_2 & s_1\,s_2\,q_3+s_1\,L_2\,c_2 \\ s_2 & 0 & -c_2 & -c_2\,q_3+L_2\,s_2+L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Posizione:

Dalla cinematica diretta abbiamo che:

$$\begin{cases} x=c_1\,s_2\,q_3+c_1\,L_2\,c_2=c_1(s_2\,q_3+L_2\,c_2) \\ y=s_1\,s_2\,q_3+s_1\,L_2\,c_2=s_1(s_2\,q_3+L_2\,c_2) \\ z=-c_2\,q_3+L_2\,s_2+L_1 \end{cases}$$

Sommando i quadrati delle prime due equazioni, otteniamo:

$$x^2+y^2=(c_1^2+s_1^2)(s_2\,q_3+L_2\,c_2)^2=(s_2\,q_3+L_2\,c_2)^2$$

Possiamo allora scrivere il seguente sistema:

$$\begin{cases} c_2\, q_3 - L_2\, s_2 = L_1 - z \\ s_2\, q_3 + L_2\, c_2 = \pm\sqrt{x^2+y^2} \end{cases} \rightarrow \begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix}\begin{pmatrix} q_3 \\ L_2 \end{pmatrix} = \begin{pmatrix} L_1 - z \\ \pm\sqrt{x^2+y^2} \end{pmatrix}$$

Osserviamo che:

$$\begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix} = R(q_2)$$

Pertanto il prodotto a primo membro non altera la norma e possiamo uguagliarle:

$$q_3^2 + L_2^2 = (L_1 - z)^2 + x^2 + y^2$$

Questa equazione ci permette di definire lo spazio operativo: una sfera di raggio $r = \sqrt{q_3^2 + L_2^2}$, con centro in $\{0, 0, L_1\}$.

Nota 1. Poiché r dipende da q_3 nella realtà si ha una sfera cava, dovuta ai fine corsa del terzo giunto del robot:

$$r = \begin{cases} L_2 & q_3 = 0 \\ +\infty & q_3 = +\infty \\ \sqrt{q_3^2 + L_2^2} & \text{altrimenti} \end{cases}$$

Abbiamo una soluzione per q_3:

$$q_3 = \pm\sqrt{(L_1 - z)^2 + x^2 + y^2 - L_2^2}$$

La condizione per la singolarità si ha per $q_3 = 0$, ovvero sul bordo interno della sfera cava, definita dall'equazione:

$$x^2 + y^2 + (L_1 - z)^2 = L_2^2$$

Se $q_3 \neq 0$, il suo valore è ora noto, così come quelo del vettore $\begin{pmatrix} L_1 - z \\ \pm\sqrt{x^2+y^2} \end{pmatrix}$.
Per semplicità sia:

$$A_1 = q_3 \qquad A_2 = L_2$$
$$B_1 = L_1 - z \qquad B_2 = \sqrt{x^2+y^2}$$

Allora:

$$\begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix}\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \rightarrow \begin{pmatrix} A_1 & -A_2 \\ A_2 & A_1 \end{pmatrix}\begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$$

Poiché la matrice A è costituita da elementi che si annullano solo in condizione singolare, tale matrice è sempre invertibile, il che ci permette di definire una soluzione per tale sistema:

$$\begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \frac{1}{A_1^2 + A_2^2}\begin{pmatrix} A_1 & A_2 \\ -A_2 & A_1 \end{pmatrix}\begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$$

Poiché $\det(A) > 0$, possiamo semplificarlo nella soluzione di q_2:

$$q_2 = \operatorname{atan2}(-A_2 B_1 + A_1 B_2, A_1 B_1 + A_2 B_2)$$

Supponiamo ora $s_2\, q_3 + L_2\, c_2 \neq 0$. Se $x^2 + y^2 \neq 0$, possiamo trovare una soluzione per q_1:

$$c_1 = \frac{x}{\sqrt{x^2+y^2}}, s_1 = \frac{y}{\sqrt{x^2+y^2}} \rightarrow q_1 = \operatorname{atan2}(y, x) \quad \text{dove } \sqrt{x^2+y^2} > 0$$

Riassumendo, abbiamo due soluzioni per un valore di q_3 fissato:

$$\begin{pmatrix} \operatorname{atan2}(y, x) \\ \operatorname{atan2}\Big(-L_2(L_1 - z) + q_3\sqrt{x^2+y^2}, q_3(L_1 - z) + L_2\sqrt{x^2+y^2}\Big) \\ \pm\sqrt{x^2 + y^2 + (L_1 - z)^2 - L_2^2} \end{pmatrix}$$

(%i43)
```
posInv(x,y,z, L1, L2):=block(
    [q1, q2, q3, pdes, A1, A2, B1, B2],
        pdes:x^2+y^2,
        if(pdes+(L1-z)^2-L2^2<0 or pdes=0) then
          error("Spazio Operativo violato"),
        q3:trigsimp([sqrt(pdes+(L1-z)^2-L2^2),
                    -sqrt(pdes+(L1-z)^2-L2^2)]),
        /*print(q[3],"=",trigsimp(q3)),*/
        A1:[q3[1],q3[2]],A2:L2,
        B1:L1-z, B2:sqrt(pdes),
        c2:[A1[1]*B1+A2*B2,A1[2]*B1+A2*B2],
        s2:[-A2*B1+A1[1]*B2,-A2*B1+A1[2]*B2],
        q2:[atan2(s2[1],c2[1]),atan2(s2[2],c2[2])],
        /*print(q[2],"=",trigsimp(q2)),*/
        c1:x,s1:y,
        q1:atan2(y,x),
        /*print(q[1],"=",trigsimp(q1)),*/
        return([matrix([q1],[q2[1]],[q3[1]]),matrix([q1],[q2[2]],[q3[2]])])
    )$
```

(%i44)
```
posInv(1,1,1,1,1)
```

(%o44)
$$\left[\begin{pmatrix}\frac{\pi}{4}\\ \frac{\pi}{4}\\ 1\end{pmatrix},\begin{pmatrix}\frac{\pi}{4}\\ -\frac{\pi}{4}\\ -1\end{pmatrix}\right]$$

(%i45)
```
T:sferico(%pi/4,%pi/4,1,1,1)
```

(%o45)
$$\begin{pmatrix}\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} & 1\\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} & 1\\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Cinematica Inversa di Orientamento:

La soluzione della cinematica diretta potrebbe presentare una singolarità per la terna $\{z, y, x\}$:

$$R_{\{z,y,x\}}(\phi) = R_z(\gamma)R_y(\beta)R_x(\alpha) = \begin{pmatrix} c_\beta\, c_\gamma & s_\alpha\, s_\beta\, c_\gamma - c_\alpha\, s_\gamma & s_\alpha\, s_\gamma + c_\alpha\, s_\beta\, c_\gamma \\ c_\beta\, s_\gamma & s_\alpha\, s_\beta\, s_\gamma + c_\alpha\, c_\gamma & c_\alpha\, s_\beta\, s_\gamma - s_\alpha\, c_\gamma \\ -s_\beta & s_\alpha\, c_\beta & c_\alpha\, c_\beta \end{pmatrix}$$

La matrice di rotazione della cinematica diretta è:

$$R = \begin{pmatrix} c_1\, c_2 & s_1 & c_1\, s_2 \\ s_1\, c_2 & -c_1 & s_1\, s_2 \\ s_2 & 0 & -c_2 \end{pmatrix}$$

Se $-s_\beta = s_2 = \pm 1$ siamo in un caso singolare, ovvero $\beta = q_2 = \pm\frac{\pi}{2}$.
Supponiamo quindi $q_2 \neq \pm\frac{\pi}{2}$.

$$s_\beta = -s_2 \rightarrow c_\beta = \pm\sqrt{1 - s_\beta^2} = \pm\sqrt{1 - s_2^2} = \pm c_2 \rightarrow \beta = \mathrm{atan2}(-s_2, \pm c2) = \begin{cases} -q_2 \\ q_2 + \pi \end{cases}$$

Allora:

$$\begin{cases} c_\beta\, c_\gamma = c_1 c_2 \\ c_\beta\, s_\gamma = s_1 c_2 \end{cases} \rightarrow \begin{cases} \pm c_2\, c_\gamma = c_1 c_2 \rightarrow c_\gamma = \pm c_1 \\ \pm c_2 s_\gamma = s_1 c_2 \rightarrow s_\gamma = \pm s_1 \end{cases} \rightarrow \gamma = \mathrm{atan2}(\pm s_1, \pm c_1) = \begin{cases} q_1 \\ q_1 + \pi \end{cases}$$

$$\begin{cases} s_\alpha\, c_\beta = 0 \\ c_\alpha\, c_\beta = -c_2 \end{cases} \rightarrow \begin{cases} s_\alpha = 0 \\ \pm c_2 c_\alpha = -c_2 \rightarrow c_\alpha = \mp 1 \end{cases} \rightarrow \alpha = \mathrm{atan2}(0, \mp 1) = \begin{cases} \pi \\ 0 \end{cases}$$

In condizione di non singolarità abbiamo quindi due soluzioni generali

$$\begin{pmatrix} \pi \\ -q_2 \\ q_1 \end{pmatrix}, \begin{pmatrix} 0 \\ q_2 + \pi \\ q_1 + \pi \end{pmatrix}$$

```
(%i47) R(alpha,beta,gamma):=rodrigues(matrix([0],[0],[1]),gamma).
        rodrigues(matrix([0],[1],[0]),beta).rodrigues(matrix([1],[0],[0]),alpha)$
(%i48) orientInv(R):=block(
       [cx, sx, Phix, cy, sy, Phiy, cz, sz, Phiz],
              sy:-R[3][1],
              if(sy=1 or sy=-1) then
                error("Condizione Singolare"),
              cy:sqrt(1-sy^2),
              Phiy:[atan2(sy,cy),atan2(sy,-cy)],
              /*print(phi[y], "=", Phiy),*/
              sx:[R[3][2]/cos(Phiy[1]), R[3][2]/cos(Phiy[2])],
              cx:[R[3][3]/cos(Phiy[1]), R[3][3]/cos(Phiy[2])],
              Phix:[atan2(sx[1],cx[1]),atan2(sx[2],cx[2])],
              /*print(phi[x], "=", Phix),*/
              cz:[R[1][1]/cos(Phiy[1]),R[1][1]/cos(Phiy[2])],
              sz:[R[2][1]/cos(Phiy[1]),R[2][1]/cos(Phiy[2])],
              Phiz:[atan2(sz[1],cz[1]),atan2(sz[2],cz[2])],
              /*print(phi[z], "=", Phiz),*/
              return([matrix([Phix[1]],[Phiy[1]],[Phiz[1]]),
                      matrix([Phix[2]],[Phiy[2]],[Phiz[2]])])
       )$
(%i49) R:submatrix(4,T,4)
```

(%o49)
$$\begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

```
(%i50) orientInv(R)
```

(%o50)
$$\left[\begin{pmatrix} \pi \\ -\frac{\pi}{4} \\ \frac{\pi}{4} \end{pmatrix}, \begin{pmatrix} 0 \\ -\frac{3\pi}{4} \\ -\frac{3\pi}{4} \end{pmatrix} \right]$$

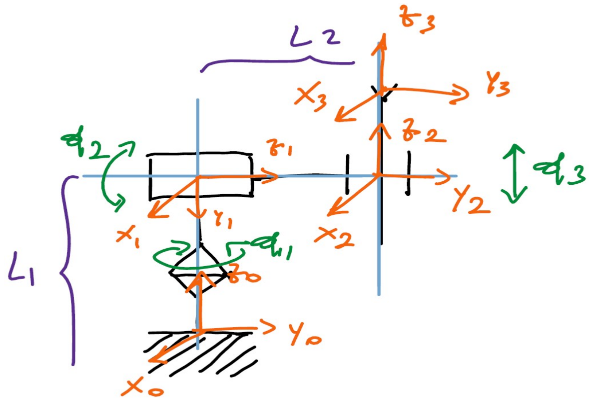
Verifica

```
(%i53) [R(%pi,-%pi/4,%pi/4),R(0,-3*%pi/4,-3*%pi/4)]
```

(%o53)
$$\left[\begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} \right]$$

Robot Sferico - Stanford (di secondo tipo)

Il robot sferico di Standford è formato da un giunto rotoidale pivot, uno rotoidale a cerniera e uno lineare trapezoidale. L'applicazione dell'algoritmo di Denavit-Hartenberg è illustrato in figura.



	θ	d	α	a
1	q_1	L_1	$-\frac{\pi}{2}$	0
2	q_2	L_2	$\frac{\pi}{2}$	0
3	0	q_3	0	0
$3'$	$-\frac{\pi}{2}$	q_3	0	0

Tabella 1. D - H per il Robot Sferico di Stanford a 3 DoF.

Procedura con le prime tre righe della tabella.

```
(%i9) stanford(q1,q2,q3,L1,L2):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(q1, L1, -%pi/2, 0),
            Q12:Q(q2, L2, %pi/2, 0),
            Q23:Q(0, q3, 0, 0),
            Q03:Q01.Q12.Q23,
            return(Q03)
      )$
```

Procedura con la riga $3'$, utile per il problema di cinematica inversa di orientamento

```
(%i10) stanford1(q1,q2,q3,L1,L2):=block(
       [Q01, Q12, Q23, Q03],
             Q01:Q(q1, L1, -%pi/2, 0),
             Q12:Q(q2, L2, %pi/2, 0),
             Q23:Q(-%pi/2, q3, 0, 0),
             Q03:Q01.Q12.Q23,
             return(Q03)
       )$
(%i11) DH:stanford(q[1],q[2],q[3],L[1],L[2])$
(%i12) DH1:stanford1(q[1],q[2],q[3],L[1],L[2])$
(%i13) [ridef(DH,q[1],q[2],q[3]),ridef(DH1,q[1],q[2],q[3])]
```

(%o13)
$$\left[\begin{pmatrix} c_1\,c_2 & -s_1 & c_1\,s_2 & c_1\,s_2\,q_3 - s_1\,L_2 \\ s_1\,c_2 & c_1 & s_1\,s_2 & s_1\,s_2\,q_3 + c_1\,L_2 \\ -s_2 & 0 & c_2 & c_2\,q_3 + L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} s_1 & c_1\,c_2 & c_1\,s_2 & c_1\,s_2\,q_3 - s_1\,L_2 \\ -c_1 & s_1\,c_2 & s_1\,s_2 & s_1\,s_2\,q_3 + c_1\,L_2 \\ 0 & -s_2 & c_2 & c_2\,q_3 + L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right]$$

Cinematica Inversa di Posizione:
Dalla cinematica diretta, la posizione dell'end-effector è:

$$\begin{cases} x = c_1\, s_2\, q_3 - s_1\, L_2 \\ y = s_1\, s_2\, q_3 + c_1\, L_2 \\ z = c_2\, q_3 + L_1 \end{cases}$$

Dalle prime due equazioni, possiamo scrivere il seguente sistema:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{pmatrix} \begin{pmatrix} q_3 s_2 \\ L_2 \end{pmatrix}$$

Osserviamo che:

$$\begin{pmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{pmatrix} = R(q_1)$$

Moltiplicare per $R(q_1)$ non altera la norma, pertanto possiamo definire un nuovo sistema con il quadrato della terza equazione:

$$\begin{cases} x^2 + y^2 = q_3^2 s_2^2 + L_2^2 \\ z = q_3 c_2 + L_1 \end{cases} \rightarrow \begin{cases} x^2 + y^2 = q_3^2 s_2^2 + L_2^2 \\ (z - L_1)^2 = q_3^2 c_2^2 \end{cases}$$

Da cui:

$$x^2 + y^2 + (z - L_1)^2 = q_3^2 s_2^2 + q_3^2 c_2^2 + L_2^2 = q_3^2 (s_2^2 + c_2^2) + L_2^2 = q_3^2 + L_2^2$$

Abbiamo quindi l'equzione dello spazio operativo del robot:

$$x^2 + y^2 + (z - L_1)^2 = q_3^2 + L_2^2$$

Che corrisponde ad una sfera cava di raggio dipendente da q_3 pari a $r = \sqrt{q_3^2 + L_2^2}$ e centrata in $\{0, 0, L_1\}$. Valgono le stesse considerazioni del robot sferico di tipo 1.
La soluzione per q_3 è quindi:

$$q_3 = \pm \sqrt{x^2 + y^2 + (z - L_1)^2 - L_2^2}$$

Se $q_3 = 0$ siamo nel caso singolare, sul bordo della sfera interna, definita dall'equazione:

$$x^2 + y^2 + (z - L_1)^2 = L_2^2$$

Se $q_3 \neq 0$ il suo valore è noto e quindi:

$$c_2 = \frac{z - L_1}{\pm q_3}$$

$$s_2 = \pm \sqrt{\frac{x^2 + y^2 - L_2^2}{{q_3}^2}}$$

$$q_2 = \mathrm{atan2}\left(\pm \sqrt{\frac{x^2 + y^2 - L_2^2}{{q_3}^2}}, \pm \frac{z - L_1}{q_3} \right)$$

Nota 1. Osserviamo che deve valere $x^2 + y^2 - L_2^2 \geqslant 0$, ovvero $x^2 + y^2 \geqslant L_2^2$.
In particolare se $x^2 + y^2 = L_2^2$, abbiamo un cilindro cavo che limita lo spazio operativo del robot.

Per q_2 noto, possiamo esprimere le prime due equazioni sottoforma di prodotto di matrici:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} q_3 s_2 & -L_2 \\ L_2 & q_3 s_2 \end{pmatrix} \begin{pmatrix} c_1 \\ s_1 \end{pmatrix}$$

Poiché $\left(\begin{smallmatrix} q_3 s_2 & -L_2 \\ L_2 & q_3 s_2 \end{smallmatrix} \right) \neq 0$ sempre in condizione di non singolarità:

$$\begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \frac{1}{q_3^2 s_2^2 + L_2^2} \begin{pmatrix} q_3 s_2 & L_2 \\ -L_2 & q_3 s_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Osservando che $q_3^2 s_2^2 + L_2^2 > 0$, allora:

$$c_1 = q_3 s_2 x + L_2 y,\, s_1 = -L_2 x + q_3 s_2 y \rightarrow q_1 = \operatorname{atan2}\left(-L_2 x + q_3 s_2 y,\, q_3 s_2 x + L_2 y\right)$$

Riassumendo:

$$\begin{pmatrix} \operatorname{atan2}\left(-L_2 x + q_3 s_2 y,\, q_3 s_2 x + L_2 y\right) \\ \operatorname{atan2}\left(\pm\sqrt{\frac{x^2+y^2-L_2^2}{q_3{}^2}},\, \pm\frac{z-L_1}{q_3}\right) \\ \pm\sqrt{x^2+y^2+(z-L_1)^2-L_2^2} \end{pmatrix}$$

```
(%i14) posInv(x,y,z, L1, L2):=block(
       [q1, c2, s2, q2, q3, pdes, det],
          pdes:trigsimp(x^2+y^2),
          if(pdes+(z-L1)^2-L2^2<0 or pdes=0 or pdes<L2^2) then
             error("Spazio Operativo violato"),
          q3:sqrt(pdes+(z-L1)^2-L2^2),
          q3:[q3,-q3],
          /*print(q[3],"=",q3),*/
          c2:[(z-L1)/q3[1], (z-L1)/q3[2]],
          s2:sqrt((pdes-L2^2)/q3[1]^2),
          s2:[s2,-s2],
          q2:[atan2(s2[1],c2[1]),atan2(s2[2],c2[2])],
          /*print(q[2],"=",q2),*/
          c1:[q3[1]*s2[1]*x+L2*y,q3[2]*s2[2]*x+L2*y],
          s1:[-L2*x+q3[1]*s2[1]*y,-L2*x+q3[2]*s2[2]*y],
          q1:[atan2(s1[1],c1[1]),atan2(s1[2],c1[2])],
          /*print(q[1],"=",q1),*/
          return([matrix([q1[1]],[q2[1]],[q3[1]]),
                 matrix([q1[2]],[q2[2]],[q3[2]])])
       )$
(%i15) posInv(10,5,5,10,5)
```

(%o15) $$\left[\begin{pmatrix} 0 \\ \pi - \arctan(2) \\ 5^{\frac{3}{2}} \end{pmatrix}, \begin{pmatrix} 0 \\ -\arctan(2) \\ -5^{\frac{3}{2}} \end{pmatrix}\right]$$

```
(%i16) T:stanford1(0,%pi-atan(2),sqrt(5)^3,10,5)
```

(%o16) $$\begin{pmatrix} 0 & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 10 \\ -1 & 0 & 0 & 5 \\ 0 & -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica inversa di Orientamento.

Dalla cinematica diretta osserviamo che la notazione $\{z, y, x\}$ è sempre non singolare nel caso in cui si usa la soluzione con la riga $3'$. Pertanto usiamo questa terna:

$$R_{\{z,y,x\}}(\phi) = R_z(\gamma) R_y(\beta) R_x(\alpha) = \begin{pmatrix} c_\beta\, c_\gamma & s_\alpha\, s_\beta\, c_\gamma - c_\alpha\, s_\gamma & s_\alpha\, s_\gamma + c_\alpha\, s_\beta\, c_\gamma \\ c_\beta\, s_\gamma & s_\alpha\, s_\beta\, s_\gamma + c_\alpha\, c_\gamma & c_\alpha\, s_\beta\, s_\gamma - s_\alpha\, c_\gamma \\ -s_\beta & s_\alpha\, c_\beta & c_\alpha\, c_\beta \end{pmatrix}$$

Dalla cinematica diretta vediamo che la matrie di rotazione è:

$$R = \begin{pmatrix} s_1 & c_1\, c_2 & c_1\, s_2 \\ -c_1 & s_1\, c_2 & s_1\, s_2 \\ 0 & -s_2 & c_2 \end{pmatrix}$$

Allora:

$$-s_\beta = 0 \neq \pm 1 \rightarrow c_\beta = \pm\sqrt{1-s_\beta^2} = \pm 1 \rightarrow \beta = \mathrm{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

Da cui:

$$\begin{cases} c_\beta\, c_\gamma = s_1 \\ c_\beta\, s_\gamma = -c_1 \end{cases} \rightarrow \begin{cases} \pm c_\gamma = s_1 \rightarrow c_\gamma = \pm s_1 \\ \pm s_\gamma = -c_1 \rightarrow s_\gamma = \mp c_1 \end{cases}$$

$$\gamma = \mathrm{atan2}(\mp c_1, \pm s_1) = \mathrm{atan2}\left(\mp\sin\left(q_1 + \frac{\pi}{2}\right), \mp\cos\left(q_1 + \frac{\pi}{2}\right)\right) = \begin{cases} q_1 + \frac{3\pi}{2} \\ q1 + \frac{\pi}{2} \end{cases}$$

$$\begin{cases} s_\alpha\, c_\beta = -s_2 \\ c_\alpha\, c_\beta = c_2 \end{cases} \rightarrow \begin{cases} \pm s_\alpha = -s_2 \rightarrow s_\alpha = \mp s_2 \\ \pm c_\alpha = c_2 \rightarrow c_\alpha = \pm c_2 \end{cases} \rightarrow \alpha = \mathrm{atan2}(\mp s_2, \pm c_2) = \begin{cases} -q_2 \\ \pi - q_2 \end{cases}$$

Riassumendo:

$$\begin{pmatrix} -q_2 \\ 0 \\ q_1 + \frac{3\pi}{2} \end{pmatrix}, \begin{pmatrix} \pi - q_2 \\ \pi \\ q_1 + \frac{\pi}{2} \end{pmatrix}$$

```
(%i17) R(alpha,beta,gamma):=rodrigues(matrix([0],[0],[1]),gamma).
        rodrigues(matrix([0],[1],[0]),beta).rodrigues(matrix([1],[0],[0]),alpha)$
(%i18) orientInv(R):=block(
       [cx, sx, Phix, cy, sy, Phiy, cz, sz, Phiz],
            sy:R[3][1],
            cy:sqrt(1-sy^2),
            Phiy:[atan2(sy,cy), atan2(sy, -cy)],
            /*print(phi[y], "=", Phiy),*/
            sx:[R[3][2]/cos(Phiy[1]),R[3][2]/cos(Phiy[2])],
            cx:[R[3][3]/cos(Phiy[1]),R[3][3]/cos(Phiy[2])],
            Phix:[atan2(sx[1],cx[1]),atan2(sx[2],cx[2])],
            /*print(phi[x], "=", Phix),*/
            sz:[R[2][1]/cos(Phiy[1]),R[2][1]/cos(Phiy[2])],
            cz:[R[1][1]/cos(Phiy[1]),R[1][1]/cos(Phiy[2])],
            Phiz:[atan2(sz[1],cz[1]),atan2(sz[2],cz[2])],
            /*print(phi[z], "=", Phiz),*/
            return([matrix([Phix[1]],[Phiy[1]],[Phiz[1]]),
                    matrix([Phix[2]],[Phiy[2]],[Phiz[2]])])
       )$
(%i19) R:submatrix(4,T,4)
```

(%o19) $$\begin{pmatrix} 0 & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ -1 & 0 & 0 \\ 0 & -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{pmatrix}$$

```
(%i20) orientInv(R)
```

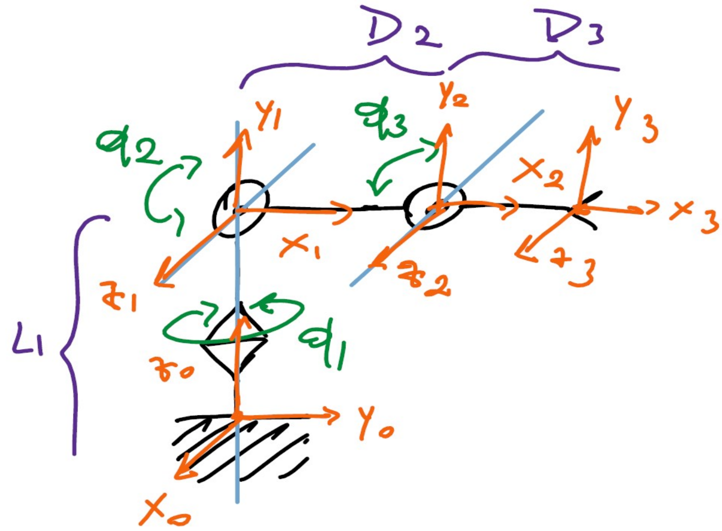
(%o20) $$\left[\begin{pmatrix} \arctan(2) - \pi \\ 0 \\ -\frac{\pi}{2} \end{pmatrix}, \begin{pmatrix} \arctan(2) \\ \pi \\ \frac{\pi}{2} \end{pmatrix}\right]$$

```
(%i21) [R(atan(2)-%pi,0,-%pi/2),R(atan(2),%pi,%pi/2)]
```

(%o21) $$\left[\begin{pmatrix} 0 & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ -1 & 0 & 0 \\ 0 & -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{pmatrix}, \begin{pmatrix} 0 & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ -1 & 0 & 0 \\ 0 & -\frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{pmatrix}\right]$$

Robot Antropomorfo

Il robot antropomorfo è formato da tre giunti di tipo rotoidali. Il vantaggio del robot antropomorfo è quello di riprodurre movimenti simili a quelli di un braccio umano. L'applicazione dell'algoritmo di Denavit-Hartenberg è illustrata in figura.



	θ	d	α	a
1	q_1	L_1	$\frac{\pi}{2}$	0
2	q_2	0	0	D_2
3	q_3	0	0	D_3

Tabella 1. D - H per Robot Antropomorfo

```
(%i9) antropomorfo(q1,q2,q3,L1,D2,D3):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(q1,L1,%pi/2,0),
            Q12:Q(q2,0,0,D2),
            Q23:Q(q3,0,0,D3),
            Q03:Q01.trigreduce(Q12.Q23),
            return(Q03)
      )$
(%i10) DH:antropomorfo(q[1],q[2],q[3],L[1],D[2],D[3])$
(%i11) ridef(DH, q[1],q[2],q[3])
```

$$(\%o11) \quad \begin{pmatrix} c_1\,c_{23} & -c_1\,s_{23} & s_1 & c_1\,D_3\,c_{23}+c_1\,D_2\,c_2 \\ s_1\,c_{23} & -s_1\,s_{23} & -c_1 & s_1\,D_3\,c_{23}+s_1\,D_2\,c_2 \\ s_{23} & c_{23} & 0 & D_3\,s_{23}+D_2\,s_2+L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Posizione:

Date tre coordinate $\{x, y, z\}$, determinare le variabili di giunto necessarie a raggiungere quella posizione.

Dalla cinematica diretta abbiamo che:

$$\begin{cases} x = c_1\, D_3\, c_{23} + c_1\, D_2\, c_2 \\ y = s_1\, D_3\, c_{23} + s_1\, D_2\, c_2 \\ z = D_3\, s_{23} + D_2\, s_2 + L_1 \end{cases}$$

Dalle prime due equazioni:

$$\begin{cases} x = c_1(D_3\, c_{23} + D_2\, c_2) \\ y = s_1(D_3\, c_{23} + D_2\, c_2) \end{cases}$$

Sommando i quadrati, otteniamo:

$$x^2 + y^2 = (c_1^2 + s_1^2)(D_3\, c_{23} + D_2\, c_2)^2 = (D_3\, c_{23} + D_2\, c_2)^2$$

$$D_3\, c_{23} + D_2\, c_2 = \pm\sqrt{x^2 + y^2}$$

Mettendo insieme alla terza equazione:

$$\begin{cases} D_3\, c_{23} + D_2\, c_2 = \pm\sqrt{x^2 + y^2} \\ D_3\, s_{23} + D_2\, s_2 = z - L_1 \end{cases}$$

Possiamo riscrivere sottoforma di prodotto matriciale:

$$\begin{pmatrix} c_{23} & -s_{23} \\ s_{23} & c_{23} \end{pmatrix}\begin{pmatrix} D_3 \\ 0 \end{pmatrix} + \begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix}\begin{pmatrix} D_2 \\ 0 \end{pmatrix} = \begin{pmatrix} \pm\sqrt{x^2 + y^2} \\ z - L_1 \end{pmatrix}$$

Scomponendo la matrice R_{23}, possiamo raccogliere in q_2 ed ottenere:

$$R(q_2)\left\{R(q_3)\begin{pmatrix} D_3 \\ 0 \end{pmatrix} + \begin{pmatrix} D_2 \\ 0 \end{pmatrix}\right\} = \begin{pmatrix} \pm\sqrt{x^2 + y^2} \\ z - L_1 \end{pmatrix}$$

Poiché $R(q_2)$ è di rotazione, il prodotto per questa non altera la norma, pertanto:

$$\left(\begin{pmatrix} D_3 & 0 \end{pmatrix} R(q_3)^T + \begin{pmatrix} D_2 & 0 \end{pmatrix}\right)\left(R(q_3)\begin{pmatrix} D_3 \\ 0 \end{pmatrix} + \begin{pmatrix} D_2 \\ 0 \end{pmatrix}\right)$$

$$= \begin{pmatrix} D_3 & 0 \end{pmatrix} R(q_3)^T R(q_3)\begin{pmatrix} D_3 \\ 0 \end{pmatrix} + \begin{pmatrix} D_2 & 0 \end{pmatrix}\begin{pmatrix} D_2 \\ 0 \end{pmatrix} + 2\begin{pmatrix} D_2 & 0 \end{pmatrix} R(q_3)\begin{pmatrix} D_3 \\ 0 \end{pmatrix} =$$

$$= D_3^2 + D_2^2 + 2D_2 D_3 c_3 = x^2 + y^2 + (z - L_1)^2$$

Da cui:

$$c_3 = \frac{x^2 + y^2 + (z - L_1)^2 - D_3^2 - D_2^2}{2D_2 D_3}$$

Poiché $-1 \leqslant c_3 \leqslant 1$, abbiamo la condizione:

$$-1 \leqslant \frac{x^2 + y^2 + (z - L_1)^2 - D_3^2 - D_2^2}{2D_2 D_3} \leqslant 1$$

$$D_3^2 + D_2^2 - 2D_2 D_3 \leqslant x^2 + y^2 + (z - L_1)^2 \leqslant D_3^2 + D_2^2 + 2D_2 D_3$$

$$(D_3 - D_2)^2 \leqslant x^2 + y^2 + (z - L_1)^2 \leqslant (D_3 + D_2)^2$$

Lo spazio operativo del robot è quindi una sfera cava centrata in $\{0, 0, L_1\}$, di raggio

$$|D_3 - D_2| \leqslant r \leqslant D_3 + D_2$$

In condizione singolare, il robot si trova vicino ai bordi delle sfere che definiscono lo spazio operativo.
Se siamo in condizione di non singolarità:

$$s_3 = \pm\sqrt{1 - c_3^2} \rightarrow q_3 = \text{atan2}(\pm s_3, c_3)$$

Se ora q_3 è noto, anche $R(q_3)$ è nota. Riscriviamo il prodotto matriciale precedente:

$$\begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix}\begin{pmatrix} c_3 D_3 + D_2 \\ s_3 D_3 \end{pmatrix} = \begin{pmatrix} \pm\sqrt{x^2 + y^2} \\ z - L_1 \end{pmatrix}$$

$$\begin{pmatrix} c_3 D_3 + D_2 & -s_3 D_3 \\ s_3 D_3 & c_3 D_3 + D_2 \end{pmatrix}\begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \begin{pmatrix} \pm\sqrt{x^2 + y^2} \\ z - L_1 \end{pmatrix}$$

Definiamo anche le seguenti sostituzioni:

$$A_1 = c_3 D_3 + D_2 \qquad A_2 = s_3 D_3$$
$$B_1 = \sqrt{x^2 + y^2} \qquad B_2 = z - L_1$$

Quindi:

$$\begin{pmatrix} A_1 & -A_2 \\ A_2 & A_1 \end{pmatrix}\begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \begin{pmatrix} \pm B_1 \\ B_2 \end{pmatrix}$$

In condizioni di non singolarità, la matrice A ha sempre determinante diverso da zero (in particolare è sempre maggiore di zero), pertanto è invertibile e il sistema ha soluzione in quanto A è invertibile:

$$\begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \frac{1}{A_1^2 + A_2^2}\begin{pmatrix} A_1 & A_2 \\ -A_2 & A_1 \end{pmatrix}\begin{pmatrix} \pm B_1 \\ B_2 \end{pmatrix}$$

Allora possiamo dare una soluzione per q_2:

$$q_2 = \text{atan2}(\pm A_2 B_1 + A_1 B_2, \pm A_1 B_1 + A_2 B_2)$$

Riprendendo le prime due equazioni, abbiamo:

$$\begin{cases} x = c_1(D_3\, c_{23} + D_2\, c_2) \\ y = s_1(D_3\, c_{23} + D_2\, c_2) \end{cases}$$

Se ora sia q_2 che q_3 sono note, $(D_3\, c_{23} + D_2\, c_2)$ è noto. Pertanto:

$$c_1 = \frac{x}{(D_3\, c_{23} + D_2\, c_2)}$$

$$s_1 = \frac{y}{(D_3\, c_{23} + D_2\, c_2)}$$

$$q_1 = \text{atan2}(s_1, c_1)$$

Riassumendo, notiamo che abbiamo due coppie di soluzioni per un q_3 fissato, dovute alla variazione di segno in q_2.

$$\begin{pmatrix} \mathrm{atan2}\Big(\frac{y}{(D_3\,c_{23}+D_2\,c_2)},\frac{x}{(D_3\,c_{23}+D_2\,c_2)}\Big) \\ \mathrm{atan2}\Big(\pm s_3 D_3\sqrt{x^2+y^2}+(c_3D_3+D_2)(z-L_1),\pm(c_3D_3+D_2)\sqrt{x^2+y^2}+s_3D_3(z-L_1)\Big) \\ \mathrm{atan2}\left(\sqrt{1-\left(\frac{x^2+y^2+(z-L_1)^2-D_3^2-D_2^2}{2D_2D_3}\right)^2},\frac{x^2+y^2+(z-L_1)^2-D_3^2-D_2^2}{2D_2D_3}\right) \end{pmatrix}$$

$$\begin{pmatrix} \mathrm{atan2}\Big(\frac{y}{(D_3\,c_{23}+D_2\,c_2)},\frac{x}{(D_3\,c_{23}+D_2\,c_2)}\Big) \\ \mathrm{atan2}\Big(\pm s_3 D_3\sqrt{x^2+y^2}+(c_3D_3+D_2)(z-L_1),\pm(c_3D_3+D_2)\sqrt{x^2+y^2}+s_3D_3(z-L_1)\Big) \\ \mathrm{atan2}\left(-\sqrt{1-\left(\frac{x^2+y^2+(z-L_1)^2-D_3^2-D_2^2}{2D_2D_3}\right)^2},\frac{x^2+y^2+(z-L_1)^2-D_3^2-D_2^2}{2D_2D_3}\right) \end{pmatrix}$$

```
(%i12) posInv(x,y,z,L1,D2,D3):=block(
       [c1, s1, c2, s2, c3, s3, pdes, cond, q1, q2, q3, A,B,C,D, tn],
           pdes:x^2+y^2,
           cond:pdes+(z-L1)^2,
           if(cond>(D2+D3)^2 or cond<(D2-D3)^2 or (x=0 and y=0)) then
             error("Spazio Operativo Violato"),
           c3:(cond-D3^2-D2^2)/(2*D2*D3),
           s3:sqrt(1-c3^2),
           q3:[atan2(s3,c3),atan2(-s3,c3)],
           A:[sqrt(pdes),-sqrt(pdes)],B:z-L1,
           C:[cos(q3[1])*D3+D2,cos(q3[2])*D3+D2],
           D:[sin(q3[1])*D3,sin(q3[2])*D3], det:[C[1]^2+D[1]^2,
       C[2]^2+D[2]^2],
           c2:[(C[1]*A[1]+D[1]*B)/det[1],(C[1]*A[2]+D[1]*B)/det[1],
               (C[2]*A[1]+D[2]*B)/det[2],(C[2]*A[2]+D[2]*B)/det[2]],
           s2:[(-D[1]*A[1]+C[1]*B)/det[1],(-D[1]*A[2]+C[1]*B)/det[1],
               (-D[2]*A[1]+C[2]*B)/det[2],(-D[2]*A[2]+C[2]*B)/det[2]],
           q2:[atan2(s2[1],c2[1]),atan2(s2[2],c2[2]),
               atan2(s2[3],c2[3]),atan2(s2[4],c2[4])],
           tn:[D3*cos(q3[1]+q2[1])+D2*cos(q2[1]),
                D3*cos(q3[1]+q2[2])+D2*cos(q2[2]),
                D3*cos(q3[2]+q2[3])+D2*cos(q2[3]),
                D3*cos(q3[2]+q2[4])+D2*cos(q2[4])],
           c1:[x/tn[1],x/tn[2],x/tn[3],x/tn[4]],
           s1:[y/tn[1],y/tn[2],y/tn[3],y/tn[4]],
           q1:[atan2(s1[1],c1[1]),atan2(s1[2],c1[2]),
               atan2(s1[3],c1[3]),atan2(s1[4],c1[4])],
           M:append([matrix([q1[1]],[q2[1]],[q3[1]]),
              matrix([q1[2]],[q2[2]],[q3[1]])],
              [matrix([q1[3]],[q2[3]],[q3[2]]),
              matrix([q1[4]],[q2[4]],[q3[2]])]),
           return(M)
       )$
```

Output soppresso per leggibilità.

```
(%i13) posInv(x,y,z,L[1],D[2],D[3])$
(%i14) sol:posInv(1,1,1,1,1,1)
```

(%o14) $$\left[\begin{pmatrix}\frac{\pi}{4}\\ -\frac{\pi}{4}\\ \frac{\pi}{2}\end{pmatrix},\begin{pmatrix}-\frac{3\pi}{4}\\ \frac{3\pi}{4}\\ \frac{\pi}{2}\end{pmatrix},\begin{pmatrix}\frac{\pi}{4}\\ \frac{\pi}{4}\\ -\frac{\pi}{2}\end{pmatrix},\begin{pmatrix}-\frac{3\pi}{4}\\ -\frac{3\pi}{4}\\ -\frac{\pi}{2}\end{pmatrix}\right]$$

Ottengo 4 soluzioni, raggruppate a due a due per un q_3 fissato.

```
(%i15) testPos(sol,L1,D2,D3):=block([t],
       for i:1 thru length(sol) do(
         t:antropomorfo(sol[i][1][1],sol[i][2][1],sol[i][3][1],L1,D2,D3),
         print("sol",[i],"= ",transpose(sol[i]),"->",t)
       )
       )$
(%i16) testPos(sol,1,1,1)
```

$$\text{sol }[1]=\begin{pmatrix}\frac{\pi}{4} & -\frac{\pi}{4} & \frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}\frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} & 1\\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{\sqrt{2}} & 1\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$\text{sol }[2]=\begin{pmatrix}-\frac{3\pi}{4} & \frac{3\pi}{4} & \frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}\frac{1}{2} & -\frac{1}{2} & -\frac{1}{\sqrt{2}} & 1\\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} & 1\\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$\text{sol }[3]=\begin{pmatrix}\frac{\pi}{4} & \frac{\pi}{4} & -\frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}\frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 1\\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{\sqrt{2}} & 1\\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$\text{sol }[4]=\begin{pmatrix}-\frac{3\pi}{4} & -\frac{3\pi}{4} & -\frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}\frac{1}{2} & \frac{1}{2} & -\frac{1}{\sqrt{2}} & 1\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} & 1\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

(%o16) **done**

Cinematica inversa di orientamento:
La soluzione della cinematica diretta potrebbe presentare una singolarità per la terna $\{z,y,x\}$:

$$R_{\{z,y,x\}}(\phi)=R_z(\gamma)R_y(\beta)R_x(\alpha)=\begin{pmatrix}c_\beta c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma & s_\alpha s_\gamma + c_\alpha s_\beta c_\gamma\\ c_\beta s_\gamma & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma\\ -s_\beta & s_\alpha c_\beta & c_\alpha c_\beta\end{pmatrix}$$

Dalla cinematica diretta, la matrice di rotazione vale:

$$R=\begin{pmatrix} c_1\,c_{23} & -c_1\,s_{23} & s_1\\ s_1\,c_{23} & -s_1\,s_{23} & -c_1\\ s_{23} & c_{23} & 0 \end{pmatrix}$$

La terna $\{z, y, x\}$ non è singolare se il termine $s_{23} = \sin(q_2 + q_3) \neq \pm 1$.
Supponiamo di non essere in condizioni di singolarità.

$$-s_\beta = s_{23} \rightarrow s_\beta = -s_{23} \rightarrow c_\beta = \pm\sqrt{1 - s_\beta^2} = \pm\sqrt{1 - s_{23}^2} = \pm c_{23}$$

$$\beta = \operatorname{atan2}(-s_{23}, \pm c_{23}) = \begin{cases} -(q_2 + q_3)\\ \pi + q_2 + q_3 \end{cases}$$

$$\begin{cases} c_\beta\,c_\gamma = c_1\,c_{23}\\ c_\beta\,s_\gamma = s_1\,c_{23} \end{cases} \rightarrow \begin{cases} \pm c_{23}\,c_\gamma = c_1\,c_{23} \rightarrow c_\gamma = \pm c_1\\ \pm c_{23} s_\gamma = s_1\,c_{23} \rightarrow s_\gamma = \pm s_1 \end{cases} \rightarrow \gamma = \operatorname{atan2}(\pm s_1, \pm c_1) = \begin{cases} q_1\\ q1 + \pi \end{cases}$$

$$\begin{cases} s_\alpha\,c_\beta = c_{23}\\ c_\alpha\,c_\beta = 0 \end{cases} \rightarrow \begin{cases} \pm c_{23} s_a = c_{23} \rightarrow s_a = \pm 1\\ \pm c_{23} c_\alpha = 0 \rightarrow c_\alpha = 0 \end{cases} \rightarrow \alpha = \operatorname{atan2}(\pm 1, 0) = \begin{cases} \frac{\pi}{2}\\ -\frac{\pi}{2} \end{cases}$$

Riassumendo:

$$\begin{pmatrix} \frac{\pi}{2}\\ -(q_2 + q_3)\\ q_1 \end{pmatrix}, \begin{pmatrix} -\frac{\pi}{2}\\ \pi + q_2 + q_3\\ q1 + \pi \end{pmatrix}$$

```
(%i17) R(alpha,beta,gamma):=block([R],
       R:rodrigues(matrix([0],[0],[1]),gamma).
       rodrigues(matrix([0],[1],[0]),beta).rodrigues(matrix([1],[0],[0]),
       alpha),return(R)
       )$
(%i18) orientInv(R):=block(
       [cx, sx, cy, sy, cz, sz, Phix, Phiy, Phiz],
            sy:-R[3][1],
            if(sy=1 or sy=-1) then
              error("La configurazione data è singolare"),
            cy:trigsimp(sqrt(1-sy^2)),
            Phiy:[atan2(sy,cy),atan2(sy,-cy)],
            /*print(phi[y],"=",Phiy),*/
            sx:[R[3][2]/cy,-R[3][2]/cy],
            cx:R[3][3],
            Phix:[atan2(sx[1],cx),atan2(sx[2],cx)],
            /*print(phi[x],"=",Phix),*/
            cz:[R[1][1]/cy,-R[1][1]/cy],
            sz:[R[2][1]/cy,-R[2][1]/cy],
            Phiz:[atan2(sz[1],cz[1]),atan2(sz[2],cz[2])],
            /*print(phi[z],"=",Phiz),*/
            return([matrix([Phix[1]],[Phiy[1]],[Phiz[1]]),
                    matrix([Phix[2]],[Phiy[2]],[Phiz[2]])])
       )$
(%i19) R:submatrix(4,antropomorfo((%pi/4), -(%pi/4), (%pi/2),1,1,1),4)
```

$$\text{(\%o19)} \quad \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}$$

(%i20) `orientInv(R)`

$$\text{(\%o20)} \quad \left[\begin{pmatrix} \frac{\pi}{2} \\ -\frac{\pi}{4} \\ \frac{\pi}{4} \end{pmatrix}, \begin{pmatrix} -\frac{\pi}{2} \\ -\frac{3\,\pi}{4} \\ -\frac{3\,\pi}{4} \end{pmatrix}\right]$$

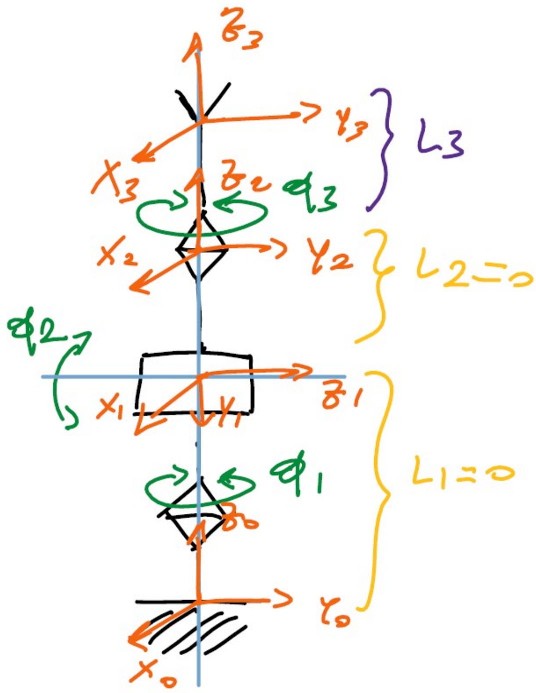
(%i21) `[R(%pi/2,-%pi/4,%pi/4),R(-%pi/2,-3*%pi/4,-3*%pi/4)]`

$$\text{(\%o21)} \quad \left[\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}\right]$$

(%i22)

Polso Sferico

E' l'effettivo attuatore in grado di orientare l'end-effector. E' formato da tre giunti di tipo rotoidale, il primo e l'ultimo di tipo pivot, il secondo a cerniera.
L'algoritmo di Denavit-Hartenberg è applicato come in figura.



	θ	d	α	a
1	q_1	0	$-\frac{\pi}{2}$	0
2	q_2	0	$\frac{\pi}{2}$	0
3	q_3	L_3	0	0

Tabella 1. D - H per il Polso Sferico

Cinematica Diretta:

```
(%i9) polso(q1,q2,q3,L3):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(q1,0,-%pi/2,0),
            Q12:Q(q2,0,%pi/2,0),
            Q23:Q(q3,L3,0,0),
            Q03:Q01.Q12.Q23,
            return(Q03)
      )$
(%i10) polso:polso(q[1], q[2], q[3], L[3])$
(%i11) ridef(polso,q[1],q[2],q[3])
```

$$(\%o11)\quad \begin{pmatrix} c_1\,c_2\,c_3-s_1\,s_3 & -c_1\,c_2\,s_3-s_1\,c_3 & c_1\,s_2 & c_1\,s_2\,L_3 \\ c_1\,s_3+s_1\,c_2\,c_3 & c_1\,c_3-s_1\,c_2\,s_3 & s_1\,s_2 & s_1\,s_2\,L_3 \\ -s_2\,c_3 & s_2\,s_3 & c_2 & c_2\,L_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica Inversa di Orientamento:
In quanto parte terminale di un robot, calcoliamo solo l'orientamento del polso.
Dalla cinematica diretta si può vedere che la matrice di rotazione ha la stessa struttura di una matrice di rotazione costruita sulla terna $\{z, y, z\}$.

$$R_{\{z,y,z\}}(\phi) = R_z(\gamma) R_y(\beta) R_z(\alpha) = \begin{pmatrix} c_\alpha\, c_\beta\, c_\gamma - s_\alpha\, s_\gamma & -c_\alpha\, c_\beta\, s_\gamma - s_\alpha\, c_\gamma & c_\alpha\, s_\beta \\ c_\alpha\, s_\gamma + s_\alpha\, c_\beta\, c_\gamma & c_\alpha\, c_\gamma - s_\alpha\, c_\beta\, s_\gamma & s_\alpha\, s_\beta \\ -s_\beta\, c_\gamma & s_\beta\, s_\gamma & c_\beta \end{pmatrix}$$

tale matrice è non singolare per $\cos(\beta) \neq \pm 1$.
Supposto di non essere in condizioni di singolarità:

$$c_\beta = c_2 \longrightarrow s_2 = \pm\sqrt{1 - c_2^2} \longrightarrow \beta = \text{atan2}(\pm s2, c2)$$

Da cui:

$$\begin{cases} c_\alpha\, s_\beta = c_1 s_2 \\ s_\alpha\, s_\beta = s_1 s_2 \end{cases} \longrightarrow \begin{cases} \pm c_\alpha\, s_2 = c_1 s_2 \longrightarrow c_\alpha = \pm c_1 \\ \pm s_\alpha\, s_2 = s_1 s_2 \longrightarrow s_\alpha = \pm s_1 \end{cases} \longrightarrow \alpha = \text{atan2}\left(\pm s_1, \pm c_1\right)$$

$$\begin{cases} -s_\beta\, c_\gamma = -s_2 s_3 \\ s_\beta\, s_\gamma = s_2 s_3 \end{cases} \longrightarrow \begin{cases} \mp s_2\, c_\gamma = -s_2 c_3 \longrightarrow c_\gamma = \pm c_3 \\ \pm s_2\, s_\gamma = s_2 s_3 \longrightarrow s_\gamma = \pm s_3 \end{cases} \longrightarrow \gamma = \text{atan2}(\pm s_3, \pm c_3)$$

```
(%i12) R(alpha,beta,gamma):=block([R],
       R:rodrigues(matrix([0],[0],[1]),alpha).rodrigues(matrix([0],[1],[0]),
       beta).rodrigues(matrix([0],[0],[1]),gamma),return(R)
       )$
(%i13) orientInv(alpha,beta,gamma):=block(
       [q1, q2, q3, ca, sa, cb, sb, cg, sg, sol, R],
           R:R(alpha,beta,gamma),
           cb:R[3][3], sb:sqrt(1-cb^2),
           q2:[atan2(sb,cb), atan2(-sb,cb)],
           cg:-R[3][1]/sb, sg:R[3][2]/sb,
           q3:[atan2(sg,cg),atan2(-sg,-cg)],
           ca:R[1][3]/sb, sa:R[2][3]/sb,
           q1:[atan2(sa,ca),atan2(-sa,-ca)],
           sol:[matrix([q1[1]],[q2[1]],[q3[1]]),
           matrix([q1[2]],[q2[2]],[q3[2]])],
           return(sol)
       )$
(%i14) orientInv(alpha,beta,gamma)
```

$$\text{(\%o14)}\quad \left[\begin{pmatrix} \text{atan2}\left(\frac{\sin(\alpha)\sin(\beta)}{\sqrt{1-\cos(\beta)^2}}, \frac{\cos(\alpha)\sin(\beta)}{\sqrt{1-\cos(\beta)^2}}\right) \\ \text{atan2}\left(\sqrt{1-\cos(\beta)^2}, \cos(\beta)\right) \\ \text{atan2}\left(\frac{\sin(\beta)\sin(\gamma)}{\sqrt{1-\cos(\beta)^2}}, \frac{\sin(\beta)\cos(\gamma)}{\sqrt{1-\cos(\beta)^2}}\right) \end{pmatrix}, \begin{pmatrix} -\text{atan2}\left(\frac{\sin(\alpha)\sin(\beta)}{\sqrt{1-\cos(\beta)^2}}, -\frac{\cos(\alpha)\sin(\beta)}{\sqrt{1-\cos(\beta)^2}}\right) \\ -\text{atan2}\left(\sqrt{1-\cos(\beta)^2}, \cos(\beta)\right) \\ -\text{atan2}\left(\frac{\sin(\beta)\sin(\gamma)}{\sqrt{1-\cos(\beta)^2}}, -\frac{\sin(\beta)\cos(\gamma)}{\sqrt{1-\cos(\beta)^2}}\right) \end{pmatrix}\right]$$

```
(%i15) sol:orientInv(0, %pi/2, %pi/4)
```

$$\text{(\%o15)}\quad \left[\begin{pmatrix} 0 \\ \frac{\pi}{2} \\ \frac{\pi}{4} \end{pmatrix}, \begin{pmatrix} \pi \\ -\frac{\pi}{2} \\ -\frac{3\pi}{4} \end{pmatrix}\right]$$

```
(%i22) testOrient(sol,L3):=block(
       for i:1 thru length(sol) do(
       t:polso(sol[i][1][1],sol[i][2][1],sol[i][3][1],L3),
              print("sol[",i,"]=",transpose(sol[i]),"->",t)
       )
       )$
(%i23) testOrient(sol,L3)
```

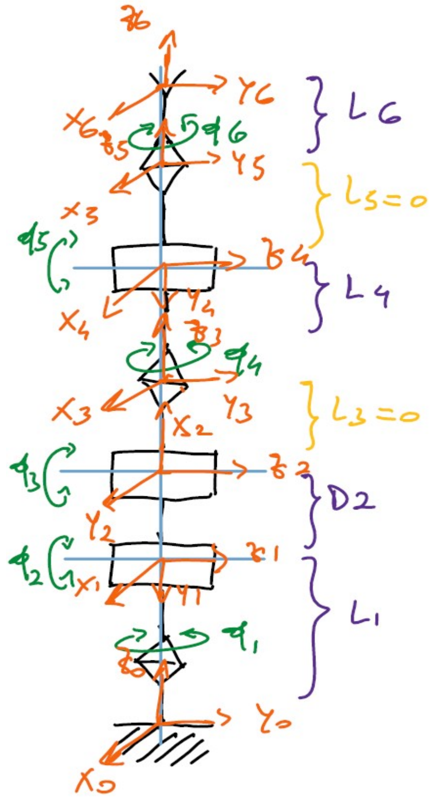
$$\text{sol[1]=} \begin{pmatrix} 0 & \frac{\pi}{2} & \frac{\pi}{4} \end{pmatrix} \text{->} \begin{pmatrix} 0 & 0 & 1 & L3 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{sol[2]=} \begin{pmatrix} \pi & -\frac{\pi}{2} & -\frac{3\pi}{4} \end{pmatrix} \text{->} \begin{pmatrix} 0 & 0 & 1 & L3 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(%o23) **done**

Robot PUMA

Il robot PUMA ha 6 DOF di tipo rotoidale. I primi tre determinano la posizione dell'end-effector, gli ultimi tre invece l'orientamento dello stesso, dato dal polso sferico.
L'algoritmo di Denavit-Hartenberg è applicato come in figura.



	θ	d	α	a
1	q_1	L_1	$-\frac{\pi}{2}$	0
2	q_2	0	0	D_2
3	q_3	0	$\frac{\pi}{2}$	0
4	q_4	L_4	$-\frac{\pi}{2}$	0
5	q_5	0	$\frac{\pi}{2}$	0
6	q_6	L_6	0	0

Tabella 1. D - H per robot P.U.M.A.

Cinematica diretta dei primi tre gradi di libertà.

```
(%i9) Q03(q1,q2,q3,L1,D2):=block([Q01, Q12, Q23, Q03],
            Q01:Q(q1,L1,-%pi/2,0),Q12:Q(q2,0,0,D2),Q23:Q(q3,0,%pi/2,0),
            Q03:Q01.trigreduce(Q12.Q23),return(Q03))$
(%i10) Q03:Q03(q[1],q[2],q[3],L[1],D[2])$
```

```
(%i11) ridef(Q03,q[1],q[2],q[3],0,0,0)
```

$$\text{(\%o11)} \begin{pmatrix} c_1\,c_{23} & -s_1 & c_1\,s_{23} & c_1\,D_2\,c_2 \\ s_1\,c_{23} & c_1 & s_1\,s_{23} & s_1\,D_2\,c_2 \\ -s_{23} & 0 & c_{23} & L_1-D_2\,s_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Cinematica diretta del polso sferico.

```
(%i12) Q36(q4,q5,q6,L4,L6):=block([Q34, Q45, Q56, Q36],
          Q34:Q(q4,L4,-%pi/2,0),Q45:Q(q5,0,%pi/2,0),Q56:Q(q6,L6,0,0),
          Q36:Q34.Q45.Q56,return(Q36))$
(%i13) Q36:Q36(q[4],q[5],q[6],L[4],L[6])$
(%i14) ridef(Q36,0,0,0,q[4],q[5],q[6])
```

$$\text{(\%o14)} \begin{pmatrix} c_4\,c_5\,c_6-s_4\,s_6 & -c_4\,c_5\,s_6-s_4\,c_6 & c_4\,s_5 & c_4\,s_5\,L_6 \\ c_4\,s_6+s_4\,c_5\,c_6 & c_4\,c_6-s_4\,c_5\,s_6 & s_4\,s_5 & s_4\,s_5\,L_6 \\ -s_5\,c_6 & s_5\,s_6 & c_5 & c_5\,L_6+L_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Calcolo la cinematica diretta completa:

```
(%i15) puma(q1,q2,q3,q4,q5,q6,L1,D2,L4,L6):=block([Q03, Q36, Q06],
          Q03:Q03(q1,q2,q3,L1,D2),
          Q36:Q36(q4,q5,q6,L4,L6),
          Q06:Q03.Q36,
          return(Q06)
       )$
(%i16) Q06:puma(q[1],q[2],q[3],q[4],q[5],q[6],L[1],D[2],L[4],L[6])$
(%i17) V:ridef(Q06,q[1],q[2],q[3],q[4],q[5],q[6])
```

(%o17) $(-c_1\,s_5\,c_6\,s_{23}-c_1\,s_4\,s_6\,c_{23}+c_1\,c_4\,c_5\,c_6\,c_{23}-s_1\,c_4\,s_6-s_1\,s_4\,c_5\,c_6, c_1\,s_5\,s_6\,s_{23}-c_1\,c_4\,c_5\,s_6\,c_{23}-c_1\,s_4\,c_6\,c_{23}+s_1\,s_4\,c_5\,s_6-s_1\,c_4\,c_6, c_1\,c_5\,s_{23}+c_1\,c_4\,s_5\,c_{23}-s_1\,s_4\,s_5, c_1\,c_5\,L_6\,s_{23}+c_1\,L_4\,s_{23}+c_1\,c_4\,s_5\,L_6\,c_{23}-s_1\,s_4\,s_5\,L_6+c_1\,D_2\,c_2; -s_1\,s_5\,c_6\,s_{23}-s_1\,s_4\,s_6\,c_{23}+s_1\,c_4\,c_5\,c_6\,c_{23}+c_1\,c_4\,s_6+c_1\,s_4\,c_5\,c_6, s_1\,s_5\,s_6\,s_{23}-s_1\,c_4\,c_5\,s_6\,c_{23}-s_1\,s_4\,c_6\,c_{23}-c_1\,s_4\,c_5\,s_6+c_1\,c_4\,c_6, s_1\,c_5\,s_{23}+s_1\,c_4\,s_5\,c_{23}+c_1\,s_4\,s_5, s_1\,c_5\,L_6\,s_{23}+s_1\,L_4\,s_{23}+s_1\,c_4\,s_5\,L_6\,c_{23}+c_1\,s_4\,s_5\,L_6+s_1\,D_2\,c_2; s_4\,s_6\,s_{23}-c_4\,c_5\,c_6\,s_{23}-s_5\,c_6\,c_{23}, c_4\,c_5\,s_6\,s_{23}+s_4\,c_6\,s_{23}+s_5\,s_6\,c_{23}, c_5\,c_{23}-c_4\,s_5\,s_{23}, -c_4\,s_5\,L_6\,s_{23}+c_5\,L_6\,c_{23}+L_4\,c_{23}-D_2\,s_2+L_1; 0, 0, 0, 1)$

Cinematica Inversa Completa:

Per i robot a 6 gradi di libertà, il problema della cinematica inversa completa è complesso, ma si può semplificare verificando la *condizione di disaccoppiamento polso-struttura portante*. Se la condizione è verificata è possibile calcolare separatamente il problema della cinematica inversa per i tre DOF rotoidali iniziali e poi quelli del polso.

Deve valere quindi: $d_{36}=R_{36}d_1+d_0$ dove d_0 e d_1 sono vettori costanti.
Ossia:

$$\begin{pmatrix} c_4\,s_5\,L_6 \\ s_4\,s_5\,L_6 \\ c_5\,L_6+L_4 \end{pmatrix} = \begin{pmatrix} c_4\,c_5\,c_6-s_4\,s_6 & -c_4\,c_5\,s_6-s_4\,c_6 & c_4\,s_5 \\ c_4\,s_6+s_4\,c_5\,c_6 & c_4\,c_6-s_4\,c_5\,s_6 & s_4\,s_5 \\ -s_5\,c_6 & s_5\,s_6 & c_5 \end{pmatrix} d_1+d_0$$

Da una semplice ispezione visiva, si può notare che

$$d_0=\begin{pmatrix} 0 \\ 0 \\ L_4 \end{pmatrix}, d_1=\begin{pmatrix} 0 \\ 0 \\ L_6 \end{pmatrix}$$

Pertanto si ha che:

$$\begin{pmatrix} c_4 s_5 L_6 \\ s_4 s_5 L6 \\ c_4 L_6 + L_4 \end{pmatrix} = R_{3,6} \cdot \begin{pmatrix} 0 \\ 0 \\ L_6 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ L_4 \end{pmatrix}$$

Come terna per le matrici di rotazione, useremo una terna newtoniana del tipo $\{z, y, z\}$:

$$R_{\{z,y,z\}}(\phi) = R_z(\gamma) R_y(\beta) R_z(\alpha) = \begin{pmatrix} c_\alpha\, c_\beta\, c_\gamma - s_\alpha\, s_\gamma & -c_\alpha\, c_\beta\, s_\gamma - s_\alpha\, c_\gamma & c_\alpha\, s_\beta \\ c_\alpha\, s_\gamma + s_\alpha\, c_\beta\, c_\gamma & c_\alpha\, c_\gamma - s_\alpha\, c_\beta\, s_\gamma & s_\alpha\, s_\beta \\ -s_\beta\, c_\gamma & s_\beta\, s_\gamma & c_\beta \end{pmatrix}$$

Siano R e P matrice di orientamento e vettore di posizione desiderati, rispettivamente.
Sappiamo che vale:

$$\begin{cases} R = R_{03} R_{36} \\ P = R_{03} d_{36} + d_{03} \end{cases}$$

Poiché vale la proprietà di disaccoppiamento: $d_{36} = R_{36} d_1 + d_0$
Sostituendo:

$$P = R_{03} R_{36} d_1 + R_{03} d_0 + d_{03} = R d_1 + R_{03} d_0 + d_{03}$$

Definiamo ora le coordinate del "polso" del robot come:

$$\hat{P} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = P - R d_1 = R_{03} d_0 + d_{03}$$

Poiché le coordinate $\{x, y, z\}$ sono note, come l'ultima colonna di R, conosiamo il valore di $\hat{P}$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = P - R d_1 = \begin{pmatrix} x \\ y \\ z \end{pmatrix} - R \begin{pmatrix} 0 \\ 0 \\ L_6 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} c_\alpha\, s_\beta \\ s_\alpha\, s_\beta \\ c_\beta \end{pmatrix} L_6$$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} x - L_6 c_\alpha\, s_\beta \\ y - L_6 s_\alpha\, s_\beta \\ z - L_6 c_\beta \end{pmatrix}$$

Questo significa che possiamo risolvere il sistema descritto dall'equazione:

$$\hat{P} = R_{03} d_0 + d_{03}$$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} L_4 c_1\, s_{23} + c_1\, D_2\, c_2 \\ L_4 s_1\, s_{23} + s_1\, D_2\, c_2 \\ L_4\, c_{23} - D_2\, s_2 + L_1 \end{pmatrix}$$

Per semplicità, verrà mantenuta una forma simbolica per le coordinate di $\hat{P}$.
Dalle prime due equazioni, possiamo raccogliere in q_1 e trovare un termine comune:

$$\begin{cases} \hat{x} = c_1 (L_4\, s_{23} + D_2\, c_2) = c_1 k \\ \hat{y} = s_1 (L_4\, s_{23} + D_2\, c_2) = s_1 k \end{cases}, k = L_4\, s_{23} + D_2\, c_2$$

Calcolando il quadrato e la loro somma si ha:

$$\hat{x}^2 + \hat{y}^2 = k^2 (c_1^2 + s_1^2) = k^2 = (L_4\, s_{23} + D_2\, c_2)^2$$

Riscriviamo ora la terza nella seguente forma:

$$\hat{z} - L_1 = L_4\, c_{23} - D_2\, s_2$$

Mettendo a sistema:

$$\begin{cases} \hat{z} - L_1 = L_4\, c_{23} - D_2\, s_2 \\ \pm\sqrt{\hat{x}^2 + \hat{y}^2} = L_4\, s_{23} + D_2\, c_2 \end{cases}$$

In forma matriciale possiamo individuare due matrici di rotazione $R(q_2 + q_3)$ e $R(q_2)$:

$$\begin{pmatrix} \hat{z} - L_1 \\ \pm\sqrt{\hat{x}^2 + \hat{y}^2} \end{pmatrix} = R_{23}\begin{pmatrix} L_4 \\ 0 \end{pmatrix} + R_2\begin{pmatrix} 0 \\ D_2 \end{pmatrix} = R_2\left\{R_3\begin{pmatrix} L_4 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ D_2 \end{pmatrix}\right\}$$

Poiché R_2 è matrice di rotazione, questa è isometrica. Pertanto possiamo eguagliare le norme:

$$\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 = D_2^2 + L_4^2 + 2D_2L_4s_3$$

Da cui:

$$s_3 = \frac{\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - D_2^2 - L_4^2}{2D_2L_4}$$

$$c_3 = \pm\sqrt{1 - s_3^2}$$

$$q_3 = \operatorname{atan2}(s_3, \pm c_3)$$

Poiché $-1 \leqslant s_3 \leqslant 1$:

$$-1 \leqslant \frac{\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - D_2^2 - L_4^2}{2D_2L_4} \leqslant 1$$

$$D_2^2 + L_4^2 - 2D_2L_4 \leqslant \hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 \leqslant 2D_2L_4 + D_2^2 + L_4^2$$

$$(D_2 - L_4)^2 \leqslant \hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 \leqslant (D_2 + L_4)^2$$

Questa disequazione definisce lo spazio operativo del robot, identificando una sfera cava di centro in $\{0, 0, L_1\}$ e con raggio $|D_2 - L_4| \leqslant r \leqslant D_2 + L_4$
In condizioni di singolarità, abbiamo una soluzione degenere per q_3:

$$q_3 = \begin{cases} \operatorname{atan2}(1, 0) = \frac{\pi}{2} & \text{se } s_3 = 1 \\ \operatorname{atan2}(-1, 0) = -\frac{\pi}{2} & \text{se } s_3 = -1 \end{cases}$$

Se q_3 è noto, il sistema di equazioni precedenti può essere riscritto come:

$$\begin{pmatrix} \hat{z} - L_1 \\ \pm\sqrt{\hat{x}^2 + \hat{y}^2} \end{pmatrix} = \begin{pmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{pmatrix}\begin{pmatrix} c_3L_4 \\ s_3L_4 + D_2 \end{pmatrix}$$

Siano ora:

$$A_1 = \hat{z} - L_1 \qquad A_2 = \sqrt{\hat{x}^2 + \hat{y}^2}$$

$$B_1 = c_3L_4 \qquad B_2 = s_3L_4 + D_2$$

$$\begin{pmatrix} A_1 \\ \pm A_2 \end{pmatrix} = \begin{pmatrix} B_1 & -B_2 \\ B_2 & B_1 \end{pmatrix}\begin{pmatrix} c_2 \\ s_2 \end{pmatrix}$$

Poiché la matrice B ha sempre determinante diverso da zero, è invertibile. Quindi:

$$\begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \frac{1}{B_1^2 + B_2^2}\begin{pmatrix} B_1 & B_2 \\ -B_2 & B_1 \end{pmatrix}\begin{pmatrix} A_1 \\ \pm A_2 \end{pmatrix}$$

Poiché $\det(B) > 0$, possiamo scrivere:

$$q_2 = \operatorname{atan2}(s_2, c_2) = \operatorname{atan2}(-B_2A_1 \pm B_1A_2, B_1A_1 \pm B_2A_2)$$

Fissato un valore di q_3 si hanno quindi due soluzioni di q_2.

Noti ora q_2 e q_3, conosciamo il termine k. Pertanto, ricordando che:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} c_1 \\ s_1 \end{pmatrix} k \longrightarrow c_1 = \frac{\hat{x}}{k} \quad s_1 = \frac{\hat{y}}{k}$$

$$q_1 = \mathrm{atan2}(s_1, c_1) = \mathrm{atan2}\left(\frac{\hat{y}}{k}, \frac{\hat{x}}{k}\right) \quad \text{dove}\, k = L_4\, s_{23} + D_2\, c_2$$

Riassumendo:

$$\begin{pmatrix} \mathrm{atan2}\left(\frac{\hat{y}}{L_4\, s_{23} + D_2\, c_2}, \frac{\hat{x}}{L_4\, s_{23} + D_2\, c_2}\right) \\ \mathrm{atan2}\left(-(s_3 L_4 + D_2)(z - L_1) \pm c_3 L_4 \sqrt{\hat{x}^2 + \hat{y}^2}, c_3 L_4 (z - L_1) \pm (s_3 L_4 + D_2) \sqrt{\hat{x}^2 + \hat{y}^2}\right) \\ \mathrm{atan2}\left(\frac{\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - D_2^2 - L_4^2}{2 D_2 L_4}, \sqrt{1 - \left(\frac{\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - D_2^2 - L_4^2}{2 D_2 L_4}\right)^2}\right) \end{pmatrix}$$

$$\begin{pmatrix} \mathrm{atan2}\left(\frac{\hat{y}}{L_4\, s_{23} + D_2\, c_2}, \frac{\hat{x}}{L_4\, s_{23} + D_2\, c_2}\right) \\ \mathrm{atan2}\left(-(s_3 L_4 + D_2)(z - L_1) \pm c_3 L_4 \sqrt{\hat{x}^2 + \hat{y}^2}, c_3 L_4 (z - L_1) \pm (s_3 L_4 + D_2) \sqrt{\hat{x}^2 + \hat{y}^2}\right) \\ \mathrm{atan2}\left(\frac{\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - D_2^2 - L_4^2}{2 D_2 L_4}, -\sqrt{1 - \left(\frac{\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - D_2^2 - L_4^2}{2 D_2 L_4}\right)^2}\right) \end{pmatrix}$$

In condizione non singolare abbiamo quindi due coppie di soluzioni, per un totale di quattro.

Risolviamo ora le equazioni per gli ultimi tre gradi di libertà.
Ricordando che:

$$R = R_{03} R_{36}$$

E che R_{03} è ora nota, per le proprietà delle matrici di rotazione possiamo scrivere:

$$R_{36} = R_{03}^T R = \begin{pmatrix} \hat{r}_{11} & \hat{r}_{12} & \hat{r}_{13} \\ \hat{r}_{21} & r_{22} & \hat{r}_{23} \\ \hat{r}_{31} & \hat{r}_{32} & \hat{r}_{33} \end{pmatrix} = \begin{pmatrix} c_4\, c_5\, c_6 - s_4\, s_6 & -c_4\, c_5\, s_6 - s_4\, c_6 & c_4\, s_5 \\ c_4\, s_6 + s_4\, c_5\, c_6 & c_4\, c_6 - s_4\, c_5\, s_6 & s_4\, s_5 \\ -s_5\, c_6 & s_5\, s_6 & c_5 \end{pmatrix}$$

Ne deduciamo che:

$$c_5 = \hat{r}_{33} \longrightarrow s_5 = \pm\sqrt{1 - c_5^2} = \pm\sqrt{1 - \hat{r}_{33}^2} \longrightarrow q_5 = \mathrm{atan2}\left(\pm\sqrt{1 - \hat{r}_{33}^2}, \hat{r}_{33}\right)$$

$$\begin{cases} c_4\, s_5 = \hat{r}_{13} \\ s_4\, s_5 = \hat{r}_{23} \end{cases} \rightarrow \begin{cases} \pm c_4\, s_5 = \hat{r}_{13} \longrightarrow c_4 = \frac{\hat{r}_{13}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \\ \pm s_4\, s_5 = \hat{r}_{23} \longrightarrow s_4 = \frac{\hat{r}_{23}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \end{cases} \longrightarrow q_4 = \mathrm{atan2}\left(\pm\frac{\hat{r}_{23}}{\sqrt{1 - \hat{r}_{33}^2}}, \pm\frac{\hat{r}_{13}}{\sqrt{1 - \hat{r}_{33}^2}}\right)$$

$$\begin{cases} -s_5\, c_6 = \hat{r}_{31} \\ s_5\, s_6 = \hat{r}_{32} \end{cases} \longrightarrow \begin{cases} \mp s_5\, c_6 = \hat{r}_{31} \longrightarrow c_6 = \frac{\hat{r}_{31}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \\ \pm s_5\, s_6 = \hat{r}_{32} \longrightarrow s_6 = \frac{\hat{r}_{32}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \end{cases} \longrightarrow q_6 = \mathrm{atan2}\left(\pm\frac{\hat{r}_{32}}{\sqrt{1 - \hat{r}_{33}^2}}, \mp\frac{\hat{r}_{31}}{\sqrt{1 - \hat{r}_{33}^2}}\right)$$

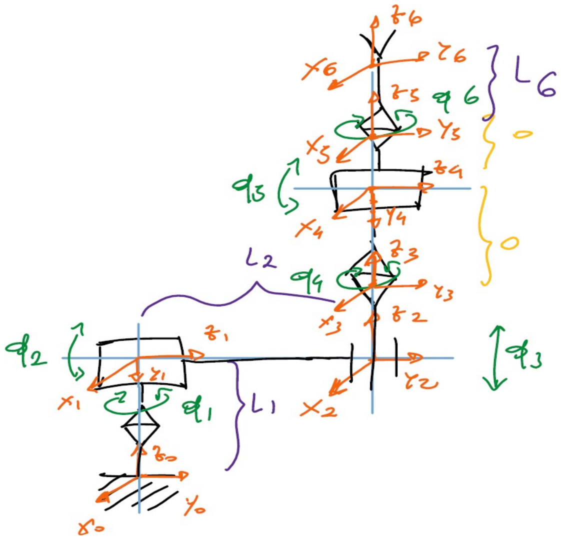
```
(%i18) R(alpha,beta,gamma):=block([R],
       R:rodrigues(matrix([0],[0],[1]),gamma).rodrigues(matrix([0],[1],[0]),
       beta).rodrigues(matrix([0],[0],[1]),alpha),return(R)
       )$
(%i19) strutturaInv(x,y,z,R,L1,D2,L4,L6):=block(
       [P, xh, yh, zh, Ph, R3, q1, q2, q3, s1, c1, s2, c2, c3, s3, A1, A2,
        B1, B2, sol03],
        P:matrix([x],[y],[z]), /*R:R(alpha,beta,gamma),*/
           /*print(R),*/
           Ph:P-R.matrix([0],[0],[L6]),
           /*print(Ph),*/
           xh:Ph[1][1], yh:Ph[2][1], zh:Ph[3][1],
           s3:trigsimp((xh^2+yh^2+(zh-L1)^2-D2^2-L4^2)/(2*D2*L4)),
           if (s3>1 or s3<-1) then error("Spazio operativo violato"),
           c3:sqrt(1-s3^2), q3:[atan2(s3,c3), atan2(s3, -c3)],
           A1:zh-L1, A2:sqrt(xh^2+yh^2),
           B1:[c3*L4,-c3*L4], B2:D2+s3*L4,
           s2:[-B2*A1+B1[1]*A2,-B2*A1-B1[1]*A2,
               -B2*A1+B1[2]*A2,-B2*A1-B1[2]*A1],
           c2:[B1[1]*A1+B2*A2,B1[1]*A1-B2*A2,
               B1[2]*A1+B2*A2,B1[2]*A1-B2*A2],
           q2:[atan2(s2[1],c2[1]),atan2(s2[2],c2[2]),
               atan2(s2[3],c2[3]),atan2(s2[4],c2[4])],
           k:[sin(q3[1]+q2[1])*L4+D2*cos(q2[1]),
              sin(q3[1]+q2[2])*L4+D2*cos(q2[2]),
              sin(q3[2]+q2[3])*L4+D2*cos(q2[3]),
              sin(q3[2]+q2[4])*L4+D2*cos(q2[4])],
           q1:[], sol03:[],
           for i:1 thru 4 do(
             c1:xh/k[i], s1:yh/k[i],
             q1:append(q1,[atan2(s1,c1)])
           ),
           sol03:ratsimp([[q1[1],q2[1],q3[1]],[q1[2],q2[2],q3[1]],
                 [q1[3],q2[3],q3[2]],[q1[4],q2[4],q3[2]]]),
           return(sol03)
       )$
```

```
(%i20) polsoInv(sol03,R,L1,D2,L4,L6):=block(
       [R36, R03, R03t, c4, s4, c5, s5, c6, s6, sol, sol36, sol06, i, valid, t],
            R03:[submatrix(4,Q03(sol03[1][1],sol03[1][2],sol03[1][3],L1,D2),4),
            submatrix(4,Q03(sol03[2][1],sol03[2][2],sol03[2][3],L1,D2),4),
            submatrix(4,Q03(sol03[3][1],sol03[3][2],sol03[3][3],L1,D2),4),
            submatrix(4,Q03(sol03[4][1],sol03[4][2],sol03[4][3],L1,D2),4)],
            R03t:[0,0,0,0],Rh:[0,0,0,0],
            for i:1 thru 4 do(
             R03t[i]:transpose(R03[i]),
             Rh[i]:R03t[i].R
            ),
            sol:[], sol36:[], valid:[], t:1,
            for i:1 thru length(Rh) do block(
            c5:Rh[i][3][3], s5:sqrt(1-c5^2), q5:[atan2(s5,c5), atan2(-s5,c5)],
            if(s5=0) then(
              /*print("Trovata configurazione singolare per i=",i),*/
              return()),
            c4:[Rh[i][1][3]/sin(q5[1]), Rh[i][1][3]/sin(q5[2])],
            s4:[Rh[i][2][3]/sin(q5[1]), Rh[i][2][3]/sin(q5[2])],
            q4:[atan2(s4[1],c4[1]), atan2(s4[2],c4[2])],
            c6:[-Rh[i][3][1]/sin(q5[1]),-Rh[i][3][1]/sin(q5[1])],
            s6:[Rh[i][3][2]/sin(q5[1]),Rh[i][3][2]/sin(q5[1])],
            q6:[atan2(s6[1],c6[1]),atan2(s6[2],c6[2])],
            sol36:append(sol36,[[matrix([q4[1]],[q5[1]],[q6[1]]),
                                 matrix([q4[2]],[q5[2]],[q6[2]])]]),
            valid:append(valid,[i])),
          return([sol36,valid])

       )$
(%i21) cinInv(x,y,z,R,L1,D2,L4,L6):=block(
       [sol03,sol36,sol06:[],i,t,valid],
       sol03:strutturaInv(x,y,z,R,L1,D2,L4,L6),
       t:polsoInv(sol03,R,L1,D2,L4,L6),
       sol36:t[1],
       valid:t[2],
       t:0,
       for i:1 thru length(valid) do(
         t:valid[i],
         sol:[[append(matrix(
          [sol03[t][1]],[sol03[t][2]],[sol03[t][3]]),sol36[i][1]),
              append(matrix(
          [sol03[t][1]],[sol03[t][2]],[sol03[t][3]]),sol36[i][2])]],
         sol06:append(sol06,sol)
       ),
       return(sol06)
       )$
```

Robot Stanford Completo

Il robot sferico di Standford è formato da un giunto rotoidale pivot, uno rotoidale a cerniera e uno lineare trapezoidale. Nella sua versione completa a 6 gradi di libertà, un polso sferico è collegato al giunto numero 3.
L'algoritmo di Denavit-Hartenberg è applicato come in figura.



	θ	d	α	a
1	q_1	L_1	$-\frac{\pi}{2}$	0
2	q_2	L_2	$\frac{\pi}{2}$	0
3	0	q_3	0	0
$3'$	$-\frac{\pi}{2}$	q_3	0	0
4	q_4	0	$-\frac{\pi}{2}$	0
5	q_5	0	$\frac{\pi}{2}$	0
6	q_6	L_6	0	0

Per semplificare il calcolo della cinematica inversa separo la matrice della cinematica completa nei dividendo i primi tre gradi di libertà dai secondi tre. Inoltre, verrà usata la riga $3'$ per non rendere mai singolare la terna $\{z, x, y\}$.

```
(%i9) Q03(q1,q2,q3,L1,L2):=block([Q01, Q12, Q23, Q03],
          Q01:Q(q1,L1,-%pi/2,0),
          Q12:Q(q2,L2,%pi/2,0),
          Q23:Q(-%pi/2,q3,0,0),
          Q03:Q01.Q12.Q23,
          return(Q03))$
(%i10) ridef(Q03:Q03(q[1],q[2],q[3],L[1],L[2]),q[1],q[2],q[3],0,0,0)
```

$$(\%o10) \quad \begin{pmatrix} s_1 & c_1\,c_2 & c_1\,s_2 & c_1\,s_2\,q_3 - s_1\,L_2 \\ -c_1 & s_1\,c_2 & s_1\,s_2 & s_1\,s_2\,q_3 + c_1\,L_2 \\ 0 & -s_2 & c_2 & c_2\,q_3 + L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
(%i11) Q36(q4,q5,q6,L6):=block([Q34, Q45, Q56, Q36],
          Q34:Q(q4,0,-%pi/2,0),Q45:Q(q5,0,%pi/2,0),Q56:Q(q6,L6,0,0),
          Q36:Q34.Q45.Q56,return(Q36))$
(%i12) ridef(Q36:Q36(q[4],q[5],q[6],L[6]),0,0,0,q[4],q[5],q[6])
```

$$(\%o12) \quad \begin{pmatrix} c_4\,c_5\,c_6 - s_4\,s_6 & -c_4\,c_5\,s_6 - s_4\,c_6 & c_4\,s_5 & c_4\,s_5\,L_6 \\ c_4\,s_6 + s_4\,c_5\,c_6 & c_4\,c_6 - s_4\,c_5\,s_6 & s_4\,s_5 & s_4\,s_5\,L_6 \\ -s_5\,c_6 & s_5\,s_6 & c_5 & c_5\,L_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Calcolo finalmente la cinematica diretta per lo stanford a 6DOF.

```
(%i13) stanford6dof(q1,q2,q3,q4,q5,q6,L1,L2,L6):=block(
       [Q03, Q36, Q06],
          Q03:Q03(q1,q2,q3,L1,L2),
          Q36:Q36(q4,q5,q6,L6),
          Q06:Q03.Q36,
          return(Q06)
       )$
```

```
(%i22) DH:stanford6dof(q[1],q[2],q[3],q[4],q[5],q[6],L[1],L[2],L[6])$
(%i23) ridef(DH,q[1],q[2],q[3],q[4],q[5],q[6])
```

(%o23) $(-s_1\,s_4\,s_6+c_1\,c_2\,c_4\,s_6-c_1\,s_2\,s_5\,c_6+c_1\,c_2\,s_4\,c_5\,c_6+s_1\,c_4\,c_5\,c_6, c_1\,s_2\,s_5\,s_6-c_1\,c_2\,s_4\,c_5\,s_6-s_1\,c_4\,c_5\,s_6-s_1\,s_4\,c_6+c_1\,c_2\,c_4\,c_6, c_1\,c_2\,s_4\,s_5+s_1\,c_4\,s_5+c_1\,s_2\,c_5, c_1\,c_2\,s_4\,s_5\,L_6+s_1\,c_4\,s_5\,L_6+c_1\,s_2\,c_5\,L_6+c_1\,s_2\,q_3-s_1\,L_2; c_1\,s_4\,s_6+s_1\,c_2\,c_4\,s_6-s_1\,s_2\,s_5\,c_6+s_1\,c_2\,s_4\,c_5\,c_6-c_1\,c_4\,c_5\,c_6, s_1\,s_2\,s_5\,s_6-s_1\,c_2\,s_4\,c_5\,s_6+c_1\,c_4\,c_5\,s_6+c_1\,s_4\,c_6+s_1\,c_2\,c_4\,c_6, s_1\,c_2\,s_4\,s_5-c_1\,c_4\,s_5+s_1\,s_2\,c_5, s_1\,c_2\,s_4\,s_5\,L_6-c_1\,c_4\,s_5\,L_6+s_1\,s_2\,c_5\,L_6+s_1\,s_2\,q_3+c_1\,L_2; -s_2\,c_4\,s_6-c_2\,s_5\,c_6-s_2\,s_4\,c_5\,c_6, c_2\,s_5\,s_6+s_2\,s_4\,c_5\,s_6-s_2\,c_4\,c_6, c_2\,c_5-s_2\,s_4\,s_5, -s_2\,s_4\,s_5\,L_6+c_2\,c_5\,L_6+c_2\,q_3+L_1; 0,0,0,1)$

Cinematica Inversa Completa:
Per i robot a 6 gradi di libertà, il problema della cinematica inversa completa è complesso, ma si può semplificare verificando la *condizione di disaccoppiamento polso-struttura portante*. Se la condizione è verificata è possibile calcolare separatamente il problema della cinematica inversa per i tre DOF rotoidali iniziali e poi quelli del polso.

Deve valere quindi: $d_{36}=R_{36}d_1+d_0$ dove d_0 e d_1 sono vettori costant. Per semplicità verifico se esiste una soluzione per il sistema dato dall'equazione $d_{36}=R_{36}d_1$. Se la soluzione esiste, allora il vettore costante $d1$ soddisfa la condizione di disaccoppiamento. Se non c'è soluzione, il modo più facile per verificare la condizione tramite il vettore costante d_0 è controllare manualmente.

```
(%i15) disaccoppiamento(Q36):=block(
       [d0, d1, R36, d36, sist, tdx],
            d0:matrix([a0],[b0],[c0]),d1:matrix([a1],[b1],[c1]),
            R36:ident(3),
            for i:1 thru 3 do(
               for j:1 thru 3 do(
                 R36[i][j]:Q36[i][j]
               )
            ),
            d36:matrix([Q36[1][4]],[Q36[2][4]],[Q36[3][4]]),
            tdx:R36.d1,
            sist:[d36[1][1]=tdx[1][1],
                  d36[2][1]=tdx[2][1],
                  d36[3][1]=tdx[3][1]],
            d1:map(rhs, solve(sist, [a1,b1,c1])[1]),
            d1:matrix([d1[1]],[d1[2]],[d1[3]]),

            if(d1#zeromatrix(3,1)) then (
              d0:zeromatrix(3,1),
              print(d[0],"=",d0, d[1],"=",d1),
              print("La struttura portante è disaccoppiata dal polso"),
              return(true)
            )else(
              print("La struttura portante non è disaccoppiata dal polso"),
              return(false)
            )
       )$
(%i16) disaccoppiamento(Q36)
```

$d_0=\begin{pmatrix}0\\0\\0\end{pmatrix} d_1=\begin{pmatrix}0\\0\\L_6\end{pmatrix}$

La struttura portante è disaccoppiata dal polso

(%o16) **true**

Abbiamo verificato che la condizione di disaccoppiamento è valida e possiamo quindi procedere nel seguente modo.
Scegliamo $\{z,y,z\}$ come terna per la matrice di orientamento:

$$R_{\{z,y,z\}}(\phi) = R_z(\gamma)R_y(\beta)R_z(\alpha) = \begin{pmatrix} c_\alpha\, c_\beta\, c_\gamma - s_\alpha\, s_\gamma & -c_\alpha\, c_\beta\, s_\gamma - s_\alpha\, c_\gamma & c_\alpha\, s_\beta \\ c_\alpha\, s_\gamma + s_\alpha\, c_\beta\, c_\gamma & c_\alpha\, c_\gamma - s_\alpha\, c_\beta\, s_\gamma & s_\alpha\, s_\beta \\ -s_\beta\, c_\gamma & s_\beta\, s_\gamma & c_\beta \end{pmatrix}$$

Abbiamo verificato che vale: $d_{36} = R_{36}d_1 + d_0$ dove d_0 e d_1 sono vettori costanti.
In particolare:

$$d_0 = 0 \qquad d_1 = \begin{pmatrix} 0 & 0 & L_6 \end{pmatrix}^T$$

Pertanto si ha che:

$$\begin{pmatrix} c_4\, s_5\, L_6 \\ s_4\, s_5\, L_6 \\ c_5\, L_6 \end{pmatrix} = R_{3,6} \cdot \begin{pmatrix} 0 \\ 0 \\ L_6 \end{pmatrix}$$

Siano R e P matrice di orientamento e vettore di posizione desiderati, rispettivamente.
Sappiamo che vale:

$$\begin{cases} R = R_{03}R_{36} \\ P = R_{03}d_{36} + d_{03} \end{cases}$$

Dalla proprietà di disaccoppiamento:

$$P = R_{03}R_{36}d_1 + R_{03}d_0 + d_{03} = R\,d_1 + R_{03}d_0 + d_{03} = R\,d_1 + d_{03}$$

Definiamo ora le coordinate del "polso" del robot come:

$$\hat{P} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = P - R\,d_1 = d_{03}$$

Poiché le coordinate $\{x,y,z\}$ sono note, come l'ultima colonna di R, conosiamo il valore di $\hat{P}$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = P - R\,d_1 = \begin{pmatrix} x \\ y \\ z \end{pmatrix} - R\begin{pmatrix} 0 \\ 0 \\ L_6 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} c_\alpha\, s_\beta \\ s_\alpha\, s_\beta \\ c_\beta \end{pmatrix} L_6$$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} x - L_6 c_\alpha\, s_\beta \\ y - L_6 s_\alpha\, s_\beta \\ z - L_6 c_\beta \end{pmatrix}$$

Questo significa che possiamo risolvere il sistema descritto dall'equazione:

$$\hat{P} = d_{03}$$

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} c_1\, s_2\, q_3 - s_1\, L_2 \\ s_1\, s_2\, q_3 + c_1\, L_2 \\ c_2\, q_3 + L_1 \end{pmatrix}$$

Per semplicità, verrà mantenuta una forma simbolica per le coordinate di $\hat{P}$.
Considerando le prime due equazioni in forma matriciale, possiamo individuare $R(q_1)$:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{pmatrix} \begin{pmatrix} s_2 q_3 \\ L_2 \end{pmatrix}$$

Poiché il prodotto per $R(q_1)$ non altera la norma in quanto è una matrice isometrica:

$$\hat{x}^2 + \hat{y}^2 = s_2^2 q_3^2 + L_2^2$$

Sommandola al quadrato della terza equazione, troviamo:

$$\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - L_2^2 = q_3^2(c_2^2 + s_2^2) = q_3^2$$

Abbiamo quindi una soluzione per q_3:

$$q_3 = \pm\sqrt{\hat{x}^2 + \hat{y}^2 + (\hat{z} - L_1)^2 - L_2^2}$$

Sempre dalla terza equazione possiamo trovare una soluzione per q_2:

$$c_2 = \pm\frac{\hat{z} - L_1}{q_3} \longrightarrow s_2 = \pm\sqrt{1 - c_2^2} = \pm\sqrt{\frac{\hat{x}^2 + \hat{y}^2 - L_2^2}{{q_3}^2}} \longrightarrow q_2 = \operatorname{atan2}(\pm s_2, \pm c_2)$$

Per q_2 noto, possiamo esprimere il prodotto matriciale precedente come:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} s_2 q_3 & -L_2 \\ L_2 & s_2 q_3 \end{pmatrix} \begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \begin{pmatrix} A_1 & -A_2 \\ A_2 & A_1 \end{pmatrix} \begin{pmatrix} c_1 \\ s_1 \end{pmatrix}$$

La matrice A ha determinante diverso da zero e in particolare maggiore di zero. Questo significa che tale sistema ha una soluzione e possiamo trovare una soluzione per q_1

$$\begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \frac{1}{A_1^2 + A_2^2} \begin{pmatrix} A_1 & A_2 \\ -A_2 & A_1 \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}$$

$$q_1 = \operatorname{atan2}(-A_2\hat{x} + A_1\hat{y}, A_1\hat{x} + A_2\hat{y})$$

Nota 1. Per quanto riguarda lo spazio operativo, valgono le stesse considerazioni dello Stanford a 3DoF

Risolviamo ora le equazioni per gli ultimi tre gradi di libertà.
Ricordando che:

$$R = R_{03} R_{36}$$

E che R_{03} è ora nota, per le proprietà delle matrici di rotazione possiamo scrivere:

$$R_{36} = R_{03}^T R = \begin{pmatrix} \hat{r}_{11} & \hat{r}_{12} & \hat{r}_{13} \\ \hat{r}_{21} & r_{22} & \hat{r}_{23} \\ \hat{r}_{31} & \hat{r}_{32} & \hat{r}_{33} \end{pmatrix} = \begin{pmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & c_4 s_5 \\ c_4 s_6 + s_4 c_5 c_6 & c_4 c_6 - s_4 c_5 s_6 & s_4 s_5 \\ -s_5 c_6 & s_5 s_6 & c_5 \end{pmatrix}$$

Ne deduciamo che:

$$c_5 = \hat{r}_{33} \longrightarrow s_5 = \pm\sqrt{1 - c_5^2} = \pm\sqrt{1 - \hat{r}_{33}^2} \longrightarrow q_5 = \operatorname{atan2}\Big(\pm\sqrt{1 - \hat{r}_{33}^2}, \hat{r}_{33}\Big)$$

$$\begin{cases} c_4 s_5 = \hat{r}_{13} \\ s_4 s_5 = \hat{r}_{23} \end{cases} \rightarrow \begin{cases} \pm c_4 s_5 = \hat{r}_{13} \longrightarrow c_4 = \frac{\hat{r}_{13}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \\ \pm s_4 s_5 = \hat{r}_{23} \longrightarrow s_4 = \frac{\hat{r}_{23}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \end{cases} \longrightarrow q_4 = \operatorname{atan2}\left(\pm\frac{\hat{r}_{23}}{\sqrt{1 - \hat{r}_{33}^2}}, \pm\frac{\hat{r}_{13}}{\sqrt{1 - \hat{r}_{33}^2}}\right)$$

$$\begin{cases} -s_5 c_6 = \hat{r}_{31} \\ s_5 s_6 = \hat{r}_{32} \end{cases} \longrightarrow \begin{cases} \mp s_5 c_6 = \hat{r}_{31} \longrightarrow c_6 = \frac{\hat{r}_{31}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \\ \pm s_5 s_6 = \hat{r}_{32} \longrightarrow s_6 = \frac{\hat{r}_{32}}{\pm\sqrt{1 - \hat{r}_{33}^2}} \end{cases} \longrightarrow q_6 = \operatorname{atan2}\left(\pm\frac{\hat{r}_{32}}{\sqrt{1 - \hat{r}_{33}^2}}, \mp\frac{\hat{r}_{31}}{\sqrt{1 - \hat{r}_{33}^2}}\right)$$

```
(%i17) R(alpha,beta,gamma):=block([R],
       R:rodrigues(matrix([0],[0],[1]),alpha).rodrigues(matrix([0],[1],[0]),
       beta).rodrigues(matrix([0],[0],[1]),gamma),return(R)
       )$
(%i18) cinInv(x,y,z,alpha,beta,gamma,L1,L2,L6):=block(
       [P, xh, yh, zh, Ph, R, q1, q2, q3, q4, q5, q6, s1, c1, s2, c2, c4, s4, c5,
       s5, c6, s6, det, sol],
            P:matrix([x],[y],[z]),
            R:R(alpha,beta,gamma),
            Ph:P-R.matrix([0],[0],[1])*L6,
            print("Orientamento Desiderato R[z,y,z]= ",R),
            print("Posizione Desiderata dell'End-Effector P= ",P),
            xh:Ph[1][1],yh:Ph[2][1],zh:Ph[3][1],
            q3:sqrt(yh^2+xh^2+(zh-L1)^2-L2^2),
            q3:[q3, -q3], /*print(q[3],"= ",q3),*/
            c2:[(zh-L1)/q3[1],(zh-L1)/q3[2]],
            s2:[sqrt(xh^2+yh^2-L2^2)/q3[1],sqrt(xh^2+yh^2-L2^2)/q3[2]],
            q2:[atan2(s2[1],c2[1]),atan2(s2[2],c2[2])],
            /*print(cos(q[2]),"= ",c2),print(sin(q[2]),"= " ,s2),
            print(q[2],"=",q2),*/
            det:xh^2+yh^2,
            c1:[(q3[1]*s2[1]*xh+yh*L2)/det,(q3[2]*s2[2]*xh+yh*L2)/det],
            s1:[(q3[1]*yh*s2[1]-xh*L2)/det,(q3[2]*yh*s2[2]-xh*L2)/det],
            q1:[atan2(s1[1],c1[1]),atan2(s1[2],c1[2])],
            /*print(cos(q[1]),"= ",c1),print(sin(q[1]),"= " ,s1),
            print(q[1],"=",q1),*/
            R03t:[transpose(submatrix(4,Q03(q1[1],q2[1],q3[1],L1,L2),4)),
                  transpose(submatrix(4,Q03(q1[2],q2[2],q3[2],L1,L2),4))],
            Rh:[R03t[1].R,R03t[2].R], sol:[],
            for i:1 thru 2 do(
              c5:Rh[i][3][3],
              s5:sqrt(1-c5^2),s5:[s5,-s5],
              q5:[atan2(s5[1],c5),atan2(s5[2],c5)],
              c6:[-Rh[i][3][1]/s5[1],-Rh[i][3][1]/s5[2]],
              s6:[Rh[i][3][2]/s5[1],Rh[i][3][2]/s5[2]],
              q6:[atan2(s6[1],c6[1]),atan2(s6[2],c6[2])],
              c4:[Rh[i][1][3]/s5[1],Rh[i][1][3]/s5[2]],
              s4:[Rh[i][2][3]/s5[1],Rh[i][2][3]/s5[2]],
              q4:[atan2(s4[1],c4[1]),atan2(s4[2],c4[2])],
              sol:append(sol,
                  [matrix([q1[i]],[q2[i]],[q3[i]],[q4[1]],[q5[1]],[q6[1]]),
                  matrix([q1[i]],[q2[i]],[q3[i]],[q4[2]],[q5[2]],[q6[2]])])
            ),
            return(sol)
       )$
(%i62)
(%i19) sol:cinInv(3,2,1,%pi/2,%pi/2,%pi,1,1,1)
```

Orientamento Desiderato R[z,y,z]= $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$

Posizione Desiderata dell'End-Effector P= $\begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$

(%o19) $$\left[\begin{pmatrix}0\\ \frac{\pi}{2}\\ 3\\ \pi\\ \frac{\pi}{2}\\ \frac{\pi}{2}\end{pmatrix},\begin{pmatrix}0\\ \frac{\pi}{2}\\ 3\\ 0\\ -\frac{\pi}{2}\\ -\frac{\pi}{2}\end{pmatrix},\begin{pmatrix}0\\ -\frac{\pi}{2}\\ -3\\ \pi\\ \frac{\pi}{2}\\ -\frac{\pi}{2}\end{pmatrix},\begin{pmatrix}0\\ -\frac{\pi}{2}\\ -3\\ 0\\ -\frac{\pi}{2}\\ \frac{\pi}{2}\end{pmatrix}\right]$$

```
(%i20) test(sol,L1,L2,L6):=block([t],
       for i:1 thru length(sol) do(
         t:stanford6dof(sol[i][1][1],sol[i][2][1],sol[i][3][1],
                        sol[i][4][1],sol[i][5][1],sol[i][6][1],L1,L2,L6),
         print("sol",[i],"= ",transpose(sol[i]),"->",t)
       )
       )$
(%i21) test(sol,1,1,1)
```

$$\text{sol }[1]=\begin{pmatrix}0 & \frac{\pi}{2} & 3 & \pi & \frac{\pi}{2} & \frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}0&1&0&3\\0&0&1&2\\1&0&0&1\\0&0&0&1\end{pmatrix}$$

$$\text{sol }[2]=\begin{pmatrix}0 & \frac{\pi}{2} & 3 & 0 & -\frac{\pi}{2} & -\frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}0&1&0&3\\0&0&1&2\\1&0&0&1\\0&0&0&1\end{pmatrix}$$

$$\text{sol }[3]=\begin{pmatrix}0 & -\frac{\pi}{2} & -3 & \pi & \frac{\pi}{2} & -\frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}0&1&0&3\\0&0&1&2\\1&0&0&1\\0&0&0&1\end{pmatrix}$$

$$\text{sol }[4]=\begin{pmatrix}0 & -\frac{\pi}{2} & -3 & 0 & -\frac{\pi}{2} & \frac{\pi}{2}\end{pmatrix}\text{->}\begin{pmatrix}0&1&0&3\\0&0&1&2\\1&0&0&1\\0&0&0&1\end{pmatrix}$$

(%o21) **done**

(%i22)

Problema dell'Orientamento Inverso per Strutture Portanti

con Terne di Eulero

1. Robot Cartesiano

La matrice di rotazione nella cinematica diretta è:

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

Scegliamo la terna $\{z, y, z\}$:

$$R_{\{z,y,z\}}(\phi) = R_z(\gamma) R_y(\beta) R_z(\alpha) = \begin{pmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta \\ c_\alpha s_\gamma + s_\alpha c_\beta c_\gamma & c_\alpha c_\gamma - s_\alpha c_\beta s_\gamma & s_\alpha s_\beta \\ -s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \end{pmatrix}$$

Vediamo che la scelta di tale terna non è mai singolare:

$$c_\beta = 0 \neq \pm 1 \longrightarrow s_\beta = \pm\sqrt{1 - c_\beta^2} = \pm 1 \longrightarrow \beta = \operatorname{atan2}(\pm 1, 0) = \begin{cases} \frac{\pi}{2} \\ -\frac{\pi}{2} \end{cases}$$

Da cui:

$$\begin{cases} c_\alpha s_\beta = 1 \\ s_\alpha s_\beta = 0 \end{cases} \longrightarrow \begin{cases} \pm c_\alpha = 1 \longrightarrow c_\alpha = \pm 1 \\ \pm s_\alpha = 0 \longrightarrow s_\alpha = 0 \end{cases} \longrightarrow \alpha = \operatorname{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

$$\begin{cases} -s_\beta c_\gamma = 1 \\ s_\beta s_\gamma = 0 \end{cases} \longrightarrow \begin{cases} \mp c_\gamma = 1 \longrightarrow c_\gamma = \mp 1 \\ \pm s_\gamma = 0 \longrightarrow s_\gamma = 0 \end{cases} \longrightarrow \gamma = \operatorname{atan2}(0, \mp 1) = \begin{cases} \pi \\ 0 \end{cases}$$

Riassumendo:

$$\begin{pmatrix} 0 \\ \frac{\pi}{2} \\ \pi \end{pmatrix}, \begin{pmatrix} \pi \\ -\frac{\pi}{2} \\ 0 \end{pmatrix}$$

2. Robot Cilindrico

La matrice di rotazione nella cinematica diretta è:

$$R = \begin{pmatrix} c_1 & 0 & -s_1 \\ s_1 & 0 & c_1 \\ 0 & -1 & 0 \end{pmatrix}$$

Scegliamo la terna $\{y, x, y\}$

$$R_{\{y,x,y\}}(\phi) = R_y(\gamma) R_x(\beta) R_y(\alpha) = \begin{pmatrix} c_\alpha c_\gamma - s_\alpha c_\beta s_\gamma & s_\beta s_\gamma & c_\alpha c_\beta s_\gamma + s_\alpha c_\gamma \\ s_\alpha s_\beta & c_\beta & -c_\alpha s_\beta \\ -c_\alpha s_\gamma - s_\alpha c_\beta c_\gamma & s_\beta c_\gamma & c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma \end{pmatrix}$$

Notiamo che la terna scelta non è mai singolare. Difatti:

$$c_\beta = 0 \neq \pm 1 \longrightarrow s_\beta = \pm\sqrt{1 - c_\beta^2} = \pm 1 \longrightarrow \beta = \text{atan2}(\pm 1, 0) = \begin{cases} \frac{\pi}{2} \\ -\frac{\pi}{2} \end{cases}$$

Da cui:

$$\begin{cases} s_\gamma s_\beta = 0 \\ c_\gamma s_\beta = -1 \end{cases} \longrightarrow \begin{cases} \pm s_\gamma = 0 \longrightarrow s_\gamma = 0 \\ \pm c_\gamma = -1 \longrightarrow c_\gamma = \mp 1 \end{cases} \longrightarrow \gamma = \text{atan2}(0, \mp 1) = \begin{cases} \pi \\ 0 \end{cases}$$

$$\begin{cases} s_\beta s_\alpha = s_1 \\ -s_\beta c_\alpha = c_1 \end{cases} \longrightarrow \begin{cases} \pm s_\alpha = s_1 \longrightarrow s_\alpha = \pm s_1 \\ \mp c_\alpha = c_1 \longrightarrow c_\alpha = \mp c_1 \end{cases} \longrightarrow \alpha = \text{atan2}(\pm s_1, \mp c_1) = \begin{cases} \pi - q_1 \\ -q_1 \end{cases}$$

Riassumendo:

$$\begin{pmatrix} \pi - q_1 \\ \frac{\pi}{2} \\ \pi \end{pmatrix}, \begin{pmatrix} -q_1 \\ -\frac{\pi}{2} \\ 0 \end{pmatrix}$$

3. Robot SCARA

Dalla cinematica diretta abbiamo:

$$R = \begin{pmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Scegliamo la terna $\{x, z, x\}$

$$R_{\{x,z,x\}}(\phi) = R_x(\gamma) R_z(\beta) R_x(\alpha) = \begin{pmatrix} c_\beta & -s_\beta\, c_\gamma & s_\beta\, s_\gamma \\ c_\alpha\, s_\beta & c_\alpha\, c_\beta\, c_\gamma - s_\alpha\, s_\gamma & -c_\alpha\, c_\beta\, s_\gamma - s_\alpha\, c_\gamma \\ s_\alpha\, s_\beta & c_\alpha\, s_\gamma + s_\alpha\, c_\beta\, c_\gamma & c_\alpha\, c_\gamma - s_\alpha\, c_\beta\, s_\gamma \end{pmatrix}$$

Notiamo che potrebbe esserci una singolarità:

$$c_\beta = c_{12} \longrightarrow s_\beta = \pm\sqrt{1 - c_\beta^2} = \pm\sqrt{1 - c_{12}^2} = \pm s_{12} \longrightarrow \beta = \mathrm{atan2}(\pm s_{12}, c_{12}) = \begin{cases} \pi - (q_1 + q_2) \\ \pi + q_1 + q_2 \end{cases}$$

Se $c_{12} \neq \pm 1$ non siamo in condizioni singolare, pertanto la soluzione trovata è valida.

Allora:

$$\begin{cases} c_\alpha\, s_\beta = s_{12} \\ s_\alpha\, s_\beta = 0 \end{cases} \longrightarrow \begin{cases} \pm s_{12} c_\alpha = s_{12} \longrightarrow c_\alpha = \pm 1 \\ \pm s_\alpha = 0 \longrightarrow s_\alpha = 0 \end{cases} \longrightarrow \alpha = \mathrm{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

$$\begin{cases} -s_\beta\, c_\gamma = -s_{12} \\ s_\beta\, s_\gamma = 0 \end{cases} \longrightarrow \begin{cases} \mp s_{12} c_\gamma = -s_{12} \longrightarrow c_\gamma = \pm 1 \\ \pm s_{12}\, s_\gamma = 0 \longrightarrow s_\gamma = 0 \end{cases} \longrightarrow \gamma = \mathrm{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

Riassumendo:

$$\begin{pmatrix} 0 \\ \pi - (q_1 + q_2) \\ \pi \end{pmatrix}, \begin{pmatrix} 0 \\ \pi + q_1 + q_2 \\ \pi \end{pmatrix}$$

4. Robot Sferico di Primo Tipo

Dalla cinematica diretta abbiamo

$$R = \begin{pmatrix} c_1c_2 & s_1 & c_1s_2 \\ s_1c_2 & -c_1 & s_1s_2 \\ s_2 & 0 & -c_2 \end{pmatrix}$$

Scegliamo la terna $\{z, y, z\}$

$$R_{\{z,y,z\}}(\phi) = R_z(\gamma)R_y(\beta)R_z(\alpha) = \begin{pmatrix} c_\alpha\, c_\beta\, c_\gamma - s_\alpha\, s_\gamma & -c_\alpha\, c_\beta\, s_\gamma - s_\alpha\, c_\gamma & c_\alpha\, s_\beta \\ c_\alpha\, s_\gamma + s_\alpha\, c_\beta\, c_\gamma & c_\alpha\, c_\gamma - s_\alpha\, c_\beta\, s_\gamma & s_\alpha\, s_\beta \\ -s_\beta\, c_\gamma & s_\beta\, s_\gamma & c_\beta \end{pmatrix}$$

Questa terna potrebbe presentare una singolarità. Infatti:

$$c_\beta = -c_2 \longrightarrow s_\beta = \pm\sqrt{1 - c_\beta^2} = \pm s_2 \longrightarrow \beta = \mathrm{atan2}(\pm s_2, -c_2) = \begin{cases} \pi - q_2 \\ \pi + q_2 \end{cases}$$

Se $c_\beta \neq \pm 1$ non siamo in condizioni singolari, pertanto la soluzione trovata è valida. Quindi:

$$\begin{cases} c_\alpha\, s_\beta = c_1s_2 \\ s_\alpha\, s_\beta = s_1s_2 \end{cases} \longrightarrow \begin{cases} \pm s_2c_\alpha = c_1s_2 \longrightarrow c_\alpha = \pm c_1 \\ \pm s_2s_\alpha = s_1s_2 \longrightarrow s_\alpha = \pm s_1 \end{cases} \longrightarrow \alpha = \mathrm{atan2}(\pm s_1, \pm c_1) = \begin{cases} q_1 \\ -q_1 \end{cases}$$

$$\begin{cases} -s_\beta\, c_\gamma = s_2 \\ s_\beta\, s_\gamma = 0 \end{cases} \longrightarrow \begin{cases} \mp s_2c_\gamma = s_2 \longrightarrow c_\gamma = \pm 1 \\ \pm s_2\, s_\gamma = 0 \longrightarrow s_\gamma = 0 \end{cases} \longrightarrow \gamma = \mathrm{atan2}(0, \pm 1) = \begin{cases} 0 \\ \pi \end{cases}$$

Riassumendo:

$$\begin{pmatrix} q_1 \\ \pi - q_2 \\ 0 \end{pmatrix}, \begin{pmatrix} -q_1 \\ \pi + q_2 \\ \pi \end{pmatrix}$$

5. Robot Sferico di II Tipo - Stanford

Dalla cinematica diretta abbiamo:

$$R = \begin{pmatrix} s_1 & c_1 c_2 & c_1 s_2 \\ -c_1 & s_1 c_2 & s_1 s_2 \\ 0 & -s_2 & c_2 \end{pmatrix}$$

Scegliamo la terna $\{z, y, z\}$

$$R_{\{z,y,z\}}(\phi) = R_z(\gamma) R_y(\beta) R_z(\alpha) = \begin{pmatrix} c_\alpha\, c_\beta\, c_\gamma - s_\alpha\, s_\gamma & -c_\alpha\, c_\beta\, s_\gamma - s_\alpha\, c_\gamma & c_\alpha\, s_\beta \\ c_\alpha\, s_\gamma + s_\alpha\, c_\beta\, c_\gamma & c_\alpha\, c_\gamma - s_\alpha\, c_\beta\, s_\gamma & s_\alpha\, s_\beta \\ -s_\beta\, c_\gamma & s_\beta\, s_\gamma & c_\beta \end{pmatrix}$$

La terna scelta potrebbe presentare una singolarità. Infatti:

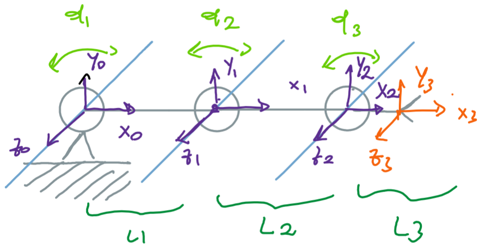
$$c_\beta = c_2 \longrightarrow s_\beta = \pm\sqrt{1 - c_\beta^2} = \pm s_2 \longrightarrow \beta = \operatorname{atan2}(\pm s_2, c_2) = \begin{cases} q_2 \\ \pi - q_2 \end{cases}$$

Se $c_\beta \neq \pm 1$ non siamo in condizioni singolari, pertanto la soluzione trovata è valida. Allora:

$$\begin{cases} c_\alpha\, s_\beta = c_1 s_2 \\ s_\alpha\, s_\beta = s_1 s_2 \end{cases} \longrightarrow \begin{cases} \pm s_2 c_\alpha = c_1 s_2 \longrightarrow c_\alpha = \pm c_1 \\ \pm s_2 s_\alpha = s_1 s_2 \longrightarrow s_\alpha = \pm s_1 \end{cases} \longrightarrow \alpha = \operatorname{atan2}(\pm s_1, \pm c_1) = \begin{cases} q_1 \\ -q_1 \end{cases}$$

$$\begin{cases} -s_\beta\, c_\gamma = 0 \\ s_\beta\, s_\gamma = -s_2 \end{cases} \longrightarrow \begin{cases} \mp s_2 c_\gamma = 0 \longrightarrow c_\gamma = 0 \\ \pm s_2\, s_\gamma = -s_2 \longrightarrow s_\gamma = \mp 1 \end{cases} \longrightarrow \gamma = \operatorname{atan2}(\mp 1, 0) = \begin{cases} -\frac{\pi}{2} \\ \frac{\pi}{2} \end{cases}$$

Robot 3 DoF Planare



	θ	d	α	a
1	q_1	0	0	L_1
2	q_2	0	0	L_2
3	q_3	0	0	L_3

Tabella 1. D - H per robot 3 DOF

Cinematica Diretta

```
(%i9) threeDofPlanar(q1,q2,q3,L1,L2,L3):=block(
      [Q01, Q12, Q23, Q03],
            Q01:Q(q1,0,0,L1),
            Q12:Q(q2,0,0,L2),
            Q23:Q(q3,0,0,L3),
            Q03:trigreduce(trigreduce(Q01.Q12).Q23),
            return(Q03)
      )$
(%i10) DH:threeDofPlanar(q[1],q[2],q[3],L[1],L[2],L[3])
```

(%o10) $(\cos(q_3+q_2+q_1), -\sin(q_3+q_2+q_1), 0, L_3\cos(q_3+q_2+q_1)+L_2\cos(q_2+q_1)+ L_1\cos(q_1); \sin(q_3+q_2+q_1), \cos(q_3+q_2+q_1), 0, L_3\sin(q_3+q_2+q_1)+L_2\sin(q_2+q_1)+L_1\sin(q_1); 0,0,1,0; 0,0,0,1)$

```
(%i11) ridef(DH,q[1],q[2],q[3])
```

(%o11) $$\begin{pmatrix} c_{123} & -s_{123} & 0 & L_3\,c_{123}+L_2\,c_{12}+L_1\,c_1 \\ s_{123} & c_{123} & 0 & L_3\,s_{123}+L_2\,s_{12}+L_1\,s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(%i12)

Cinematica Inversa

Dalla cinematica diretta vediamo che la posizione finale dell'end-effector è:

$$P=\begin{pmatrix} x \\ y \\ z \end{pmatrix}=\begin{pmatrix} L_3\,c_{123}+L_2\,c_{12}+L_1\,c_1 \\ L_3\,s_{123}+L_2\,s_{12}+L_1\,s_1 \\ 0 \end{pmatrix}$$

L'orientamento è determinato da una matrice di rotazione complessiva del tipo:

$$\mathrm{R}(\phi)=\mathrm{R}(q_3+q_2+q_1)=\begin{pmatrix} c_{123} & -s_{123} & 0 \\ s_{123} & c_{123} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Possiamo allora imporre il sistema seguente:

$$\begin{cases} x=L_3\,c_{123}+L_2\,c_{12}+L_1\,c_1 \\ y=L_3\,s_{123}+L_2\,s_{12}+L_1\,s_1 \\ z=0 \\ \phi=q_1+q_2+q_3 \end{cases} \rightarrow \begin{cases} \begin{pmatrix} x \\ y \end{pmatrix}=R_{123}\begin{bmatrix} L_3 \\ 0 \end{bmatrix}+R_{12}\begin{bmatrix} L_2 \\ 0 \end{bmatrix}+R_1\begin{bmatrix} L_1 \\ 0 \end{bmatrix} \\ \phi=q_1+q_2+q_3 \end{cases}$$

Dove con R_{123}, R_{12} e R_1 indichiamo le matrici di rotazione. Poiché conosciamo ϕ, conosciamo anche R_{123}, ovvero: $R(\phi) = R_{123}$.

Indicando allora con $\hat{P} = \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - R_{123}\begin{bmatrix} L_3 \\ 0 \end{bmatrix}$ la posizione del "polso" del robot, trasformiamo il problema della cinematica inversa del 3DOF in quello del 2DOF, pertanto abbiamo:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = R_{12}\begin{bmatrix} L_2 \\ 0 \end{bmatrix} + R_1\begin{bmatrix} L_1 \\ 0 \end{bmatrix} = R_1\left(R_2\begin{bmatrix} L_2 \\ 0 \end{bmatrix} + \begin{bmatrix} L_1 \\ 0 \end{bmatrix}\right)\mathrm{d}$$

Poiché R_1 è una matrice di rotazione è anche isometrica, pertanto mantiene la norma anche dopo il prodotto. Uguagliando le norme troviamo quindi che:

$$\hat{x} + \hat{y} = L_1^2 + L_2^2 + 2L_1L_2c_2 \qquad \rightarrow \qquad c_2 = \frac{\hat{x} + \hat{y} - L_1^2 - L_2^2}{2L_1L_2} \qquad \rightarrow \qquad s_2 = \pm\sqrt{1 - c_2^2}$$

$$q_2 = \mathrm{atan2}(\pm s_2, c_2)$$

Poiché $\cos(q_2) \in [-1, 1]$, possiamo individuare lo spazio operativo del robot, che coincide con una corona circolare di centro $(\hat{x}, \hat{y}, 0)$ e raggi $r_1 = |L_1 - L_2|$, $r_2 = L_1 + L_2$.

Ora che R_2 è nota, possiamo riscrivere l'equazione precedente come:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 \\ s1 & c_1 \end{pmatrix}\begin{pmatrix} c_2L_2 + L_1 \\ s_2L_2 \end{pmatrix} = \begin{pmatrix} c_2L_2 + L_1 & -s_2L_2 \\ s_2L_2 & c_2L_2 + L_1 \end{pmatrix}\begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}\begin{pmatrix} c_1 \\ s_1 \end{pmatrix}$$

Dove $A = c_2L_2 + L_1$ e $B = s_2L_2$.

Invertendo la matrice formata dai coefficienti A e B, possiamo risolvere per trovare $q1$:

$$\begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \frac{1}{A^2 + B^2}\begin{pmatrix} A & B \\ -B & A \end{pmatrix}\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix}$$

$$c_1 = \frac{(c_2L_2 + L_1)\hat{x} + (s_2L_2)\hat{y}}{A^2 + B^2}$$

$$s_1 = \frac{-(s_2L_2)\hat{x} + (c_2L_2 + L_1)\hat{y}}{A^2 + B^2}$$

$$q_1 = \mathrm{atan2}(s1, c1)$$

Poiché il determinante della matrice è sempre diverso da zero e positivo, possiamo semplificarlo.

Per determinare q_3, conoscendo ϕ, si ha:

$$q_3 = \phi - q_1 - q_2$$

Osserviamo che, fissata una soluzione di q_2 otteniamo una sola soluzione di q_1 e q_3.

Riassumendo:

$$\begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}; \begin{pmatrix} q_1 \\ -q_2 \\ q_3 \end{pmatrix}$$

Robot 3-DoF Planare in Processing

e Algoritmi Numerici per la sua Cinematica Inversa

Nelle pagine successive verrà listato il codice processing per la realizzazione del robot 3 Dof Planare e della sua cinematica diretta e inversa.

E' possibile interagire con il robot nei seguenti modi:

1. Click del mouse all'interno dello spazio operativo del robot (cerchio bianco), per spostare tramite cinematica inversa il robot.

2. Aumentare il valore delle variabili di giunto con i tasti 1, 2, 3.

3. Decrementare il valore delle variabili di giunto con i tasti 7, 8, 9.

4. Cambiare il gomito scelto dalla soluzione della cinematica inversa con il tasto "G"

5. Aumentare il valore dell'angolo che stabilisce l'orientamento desiderato nella cinematica inversa tramite il tasto "F".

6. Decrementare il valore dell'angolo che stabilisce l'orientamento desiderato nella cinematica inversa tramite il tasto "V".

7. Riportare il valore di tutte le variabili del robot ad un valore di default (zero), tramite il tasto "ENTER".

Osservazioni sull'Algoritmo di Newton

Il funzionamento dell'algoritmo si basa sulla riduzione dell'errore di stima con una dinamica d'errore a decrescita esponenziale.

La funzione di stima è soddisfatta dallo Jacobiano delle velocità del robot.

La versione implementata qui è ovviamente una sua versione discretizzata.

Vantaggi:

1. L'algoritmo è molto buono da un punto di vista numerico, con il vantaggio di avere una convergenza esponenziale e veloce.

2. E' inoltre poco affetto da disturbi come rumori ed incertezze.

Svantaggi:

1. Nella teoria funziona se non ci troviamo sopra una singolarità, ma nella pratica l'algoritmo ha problemi man mano che ci si avvicina ad una possibile posizione singolare

   Questo è dovuto al fatto che se il $\det(J) \cong 0$, J^{-1} diventa moolto grande e l'algoritmo è mal condizionato

2. Nella pratica risolvere le equazioni della stima delle variabili di giunto può essere molto complicato in quanto vi sono n soluzioni per n DOF, con termini trigonometrici.

Nota 1. Poiché risolvere le equazioni di stima è complesso, viene confusa la derivata con il rapporto incrementale.

Nota 2. Inizializzando in ambiente virtuale le variabili a zero, si pone l'algoritmo già in una situazione di singolarità, ed è pertanto necessario correggere manualmente il valore di q2 ad un valore prossimo a zero.

Nota 3. Il robot in singolarità tende a "sbracciare", muovendosi a scatti nel tentativo di raggiungere una soluzione stabile.

Nota 4. **(extra)** ***CAMBIO DI GOMITO DELL'ALGORITMO DI NEWTON***

Per avere un controllo sulla scelta del gomito selezionato dall'algoritmo, bisogna agire sul valore della soluzione del secondo giunto del robot. Inoltre, è importante mantenere il robot al di fuori dalle singolarità e muoverlo forzatamente verso la seconda soluzione possibile della cinematica inversa. Questo viene fatto sommando ad esso un angolo pari a PI, con un oppotuno segno che stabilisce quale delle due soluzioni è quella desiderata.

Osservazioni sull'algoritmo del Gradiente

Il funzionamento dell'algoritmo si basa sul concetto di "discesa del gradiente", nel senso che si vuole ripercorrere l'andamento della funzione gradiente in cerca dei suoi minimi locali, scegliendo di volta in volta la direzione verso la quale si percorre tale funzione per ottenere il minimo assoluto della funzione che descrive l'andamento del parametro da stimare.

La funzione di stima è soddisfatta dal trasposto dello Jacobiano delle velocità del robot.

Vantaggi:

1. A differenza dell'algoritmo di Newton, non soffre di problemi di mal condizionamento in prossimità delle singolarità.
2. Non essendoci delle divisioni all'interno della funzione di stima, ci sono meno problematiche dovute a divisioni per zero.

Svantaggi:

1. La stima è di tipo asintotico e non esponenziale, in quanto gli autovalori di $J \cdot J^T$ dipendono dalla funzione di stima, quindi la stima è di tipo asintotico e non esponenziale, in quanto gli autovalori di $J \cdot J^T$ dipendono dalla funzione di stima, quindi non possiamo essere certi sulla velocità co cui l'errore di stima tende a zero.
2. I parametri di aggiornamento della stima delle variabili di giunto influenzano pesantemente l'andamento della discesa. Se non calibrati accuratamente, l'algoritmo converge verso uno dei minimi locali della funzione, arrestando la stima.

Nota 5. Per avere stabilità asintotica, dovremmo avere che tutti gli autovalori di $J \cdot J^T$ siano a parte reale strettamente positiva. E' dimostrabile che ciò è vero se solo se il determinante di J è diverso da zero. In quel caso, la funzione di stima è definita negativa e quindi non si è mai in condizione singolare.

Se il $\det(J) = 0$, il nucleo di J contiene altri vettori oltre a quello nullo, ma che non possono quindi stare nel nucleo di J^T. Questo implica che in singolarità, potrebbero esserci errori di stima che appartengono al $\ker(J^T)$ e che danno luogo a stima nulla, provocando l'arresto dell'algoritmo prima di aver raggiunto un minimo assoluto.

Per evitare questo problema, è necessario spostare dalla singolarità il robot tramite assegnazione manuale delle variabili di giunto, dopo la quale l'algoritmo torna a stimare correttamente.

Nota 6. Riguardo ai parametri di aggiornamento della stima, relativamente all'esempio implemenntato da me, è stato trovato che per i primi due gradi di libertà non si ha una convergenza al minimo assoluto al di sotto di 10^{-5}, mentre per il parametro di stima dell'orientamento è necessario un valore dell'ordine di 10^{-10}.

Se andiamo verso ordini più bassi, la discesa diventa troppo lenta per essere accettabile.

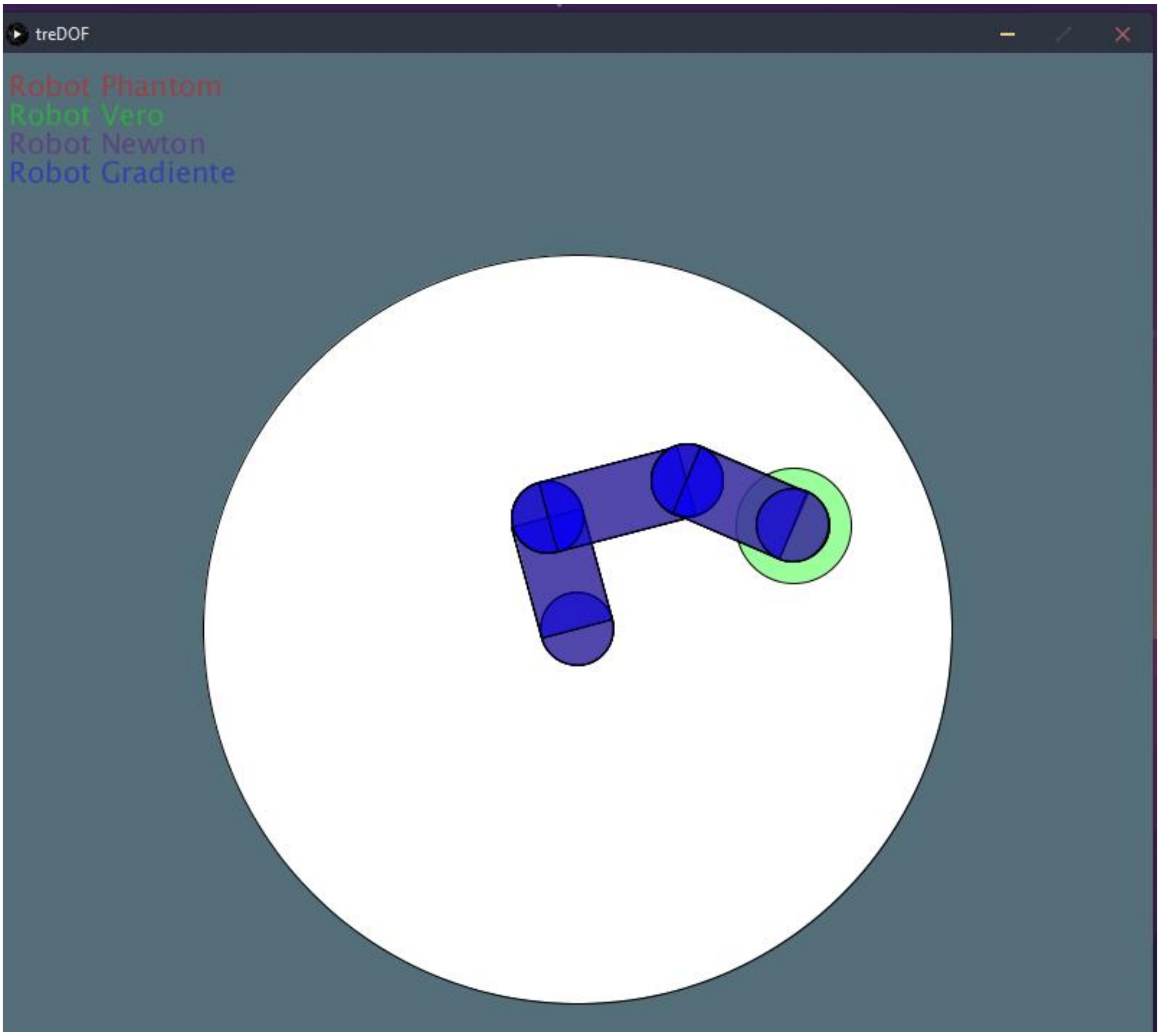


Figura 1 Gomito Alto

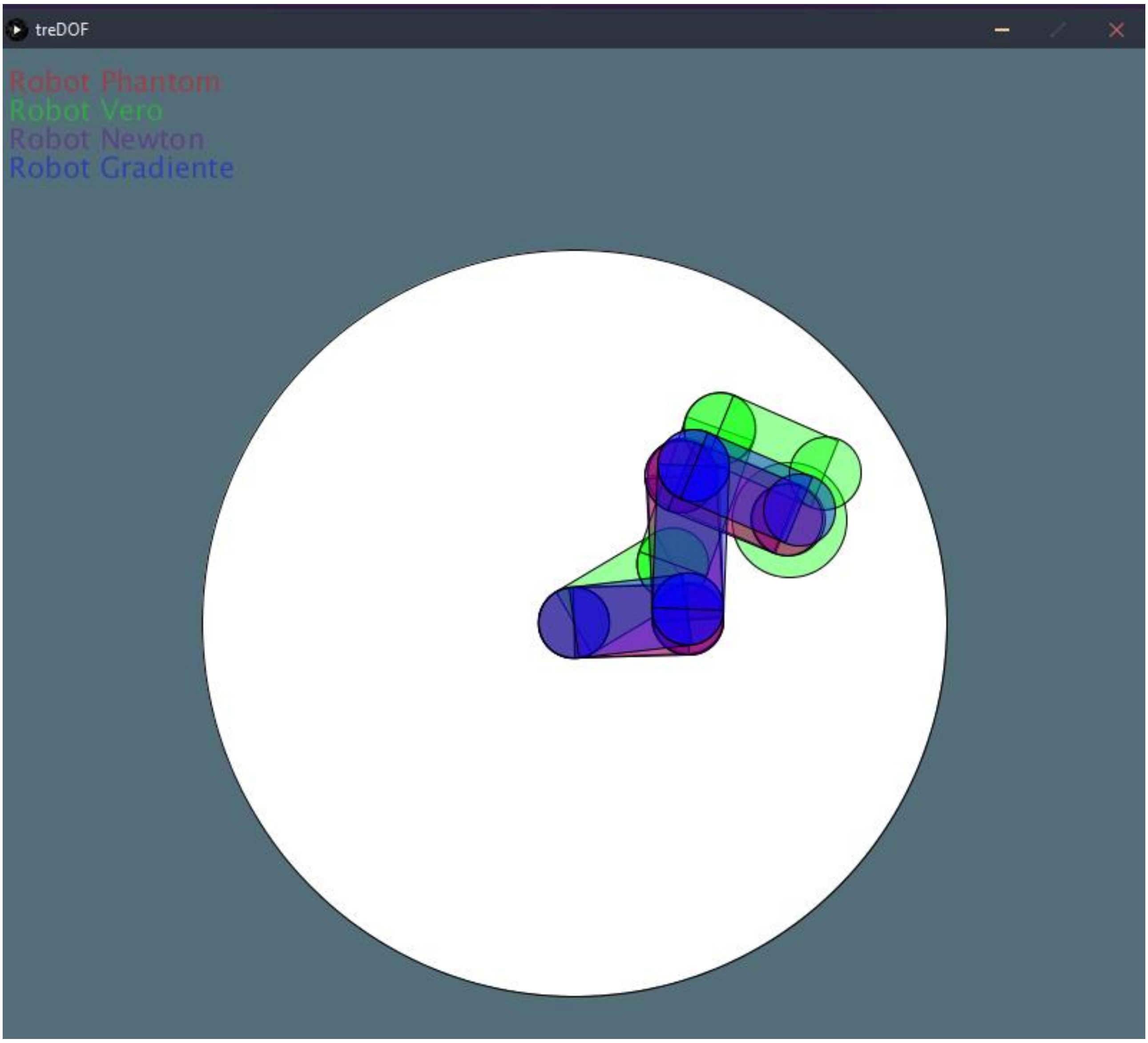


Figura 2 Cambio Gomito

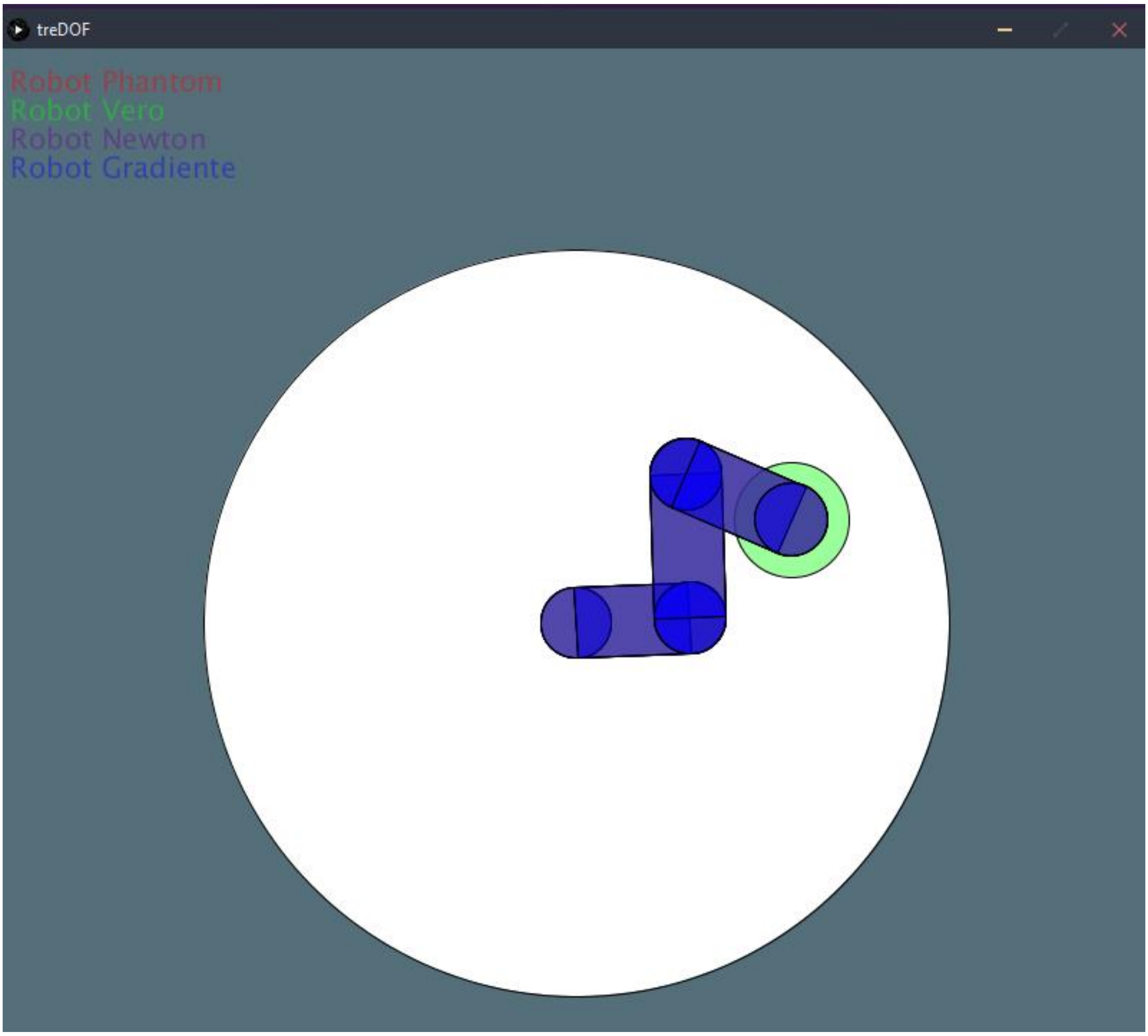


Figura 3 Gomito Basso

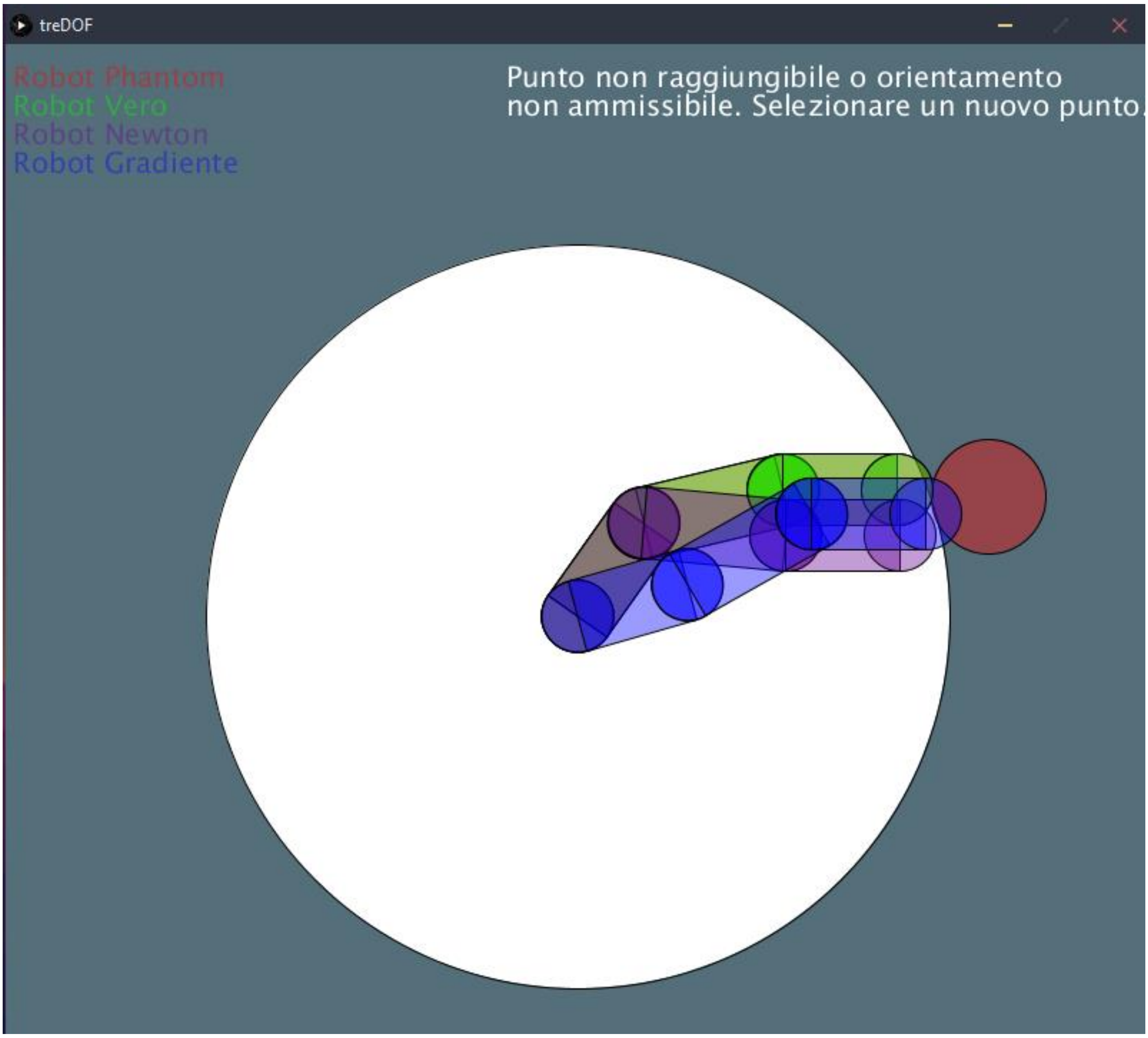


Figura 4 Punto Non Raggiungibile

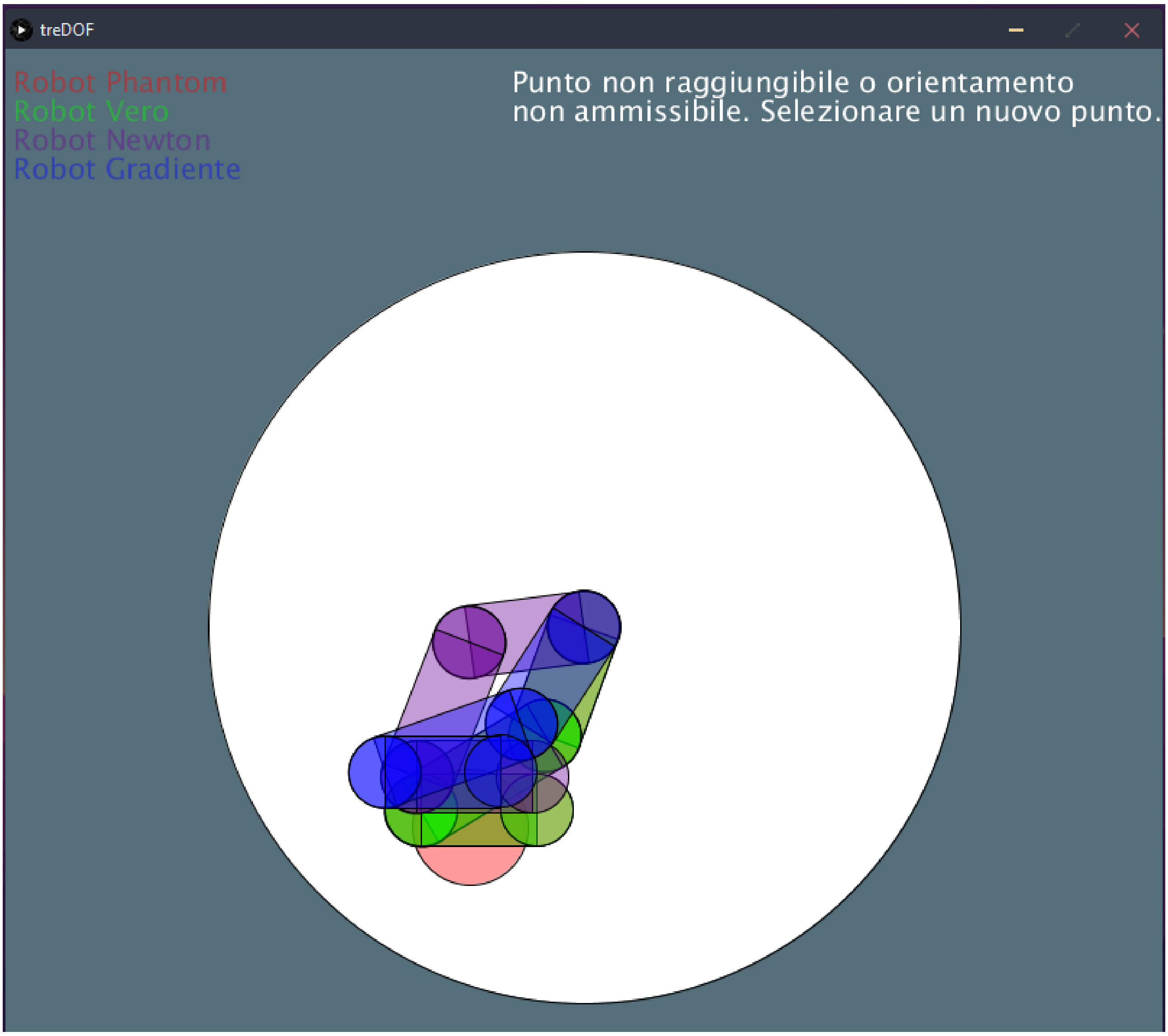


Figura 5 Punto non raggiungibile con orientamento dato

```
// definisco dimensioni link
int L1 = 80;
int L2 = 100;
int L3 = 80;
int larg = 50;
int ray = 50;

// variabili del robot phantom
color rgb = color(255,0,0,100);
float q1 = 0.0;
float q2 = 0.0;
float q3 = 0.0;

// le variabili del robot effettivo.
color rgbV = color(0,255,0,100);
float q1v = 0.0;
float q2v = 0.0;
float q3v = 0.0;

// guadagno della legge di controllo proporzionale
float K = 0.05;

// posizione desiderata per cinematica inversa
float x = 0.0;
float y = L1 + L2 + L3;

// controllo del gomito
float g = 1.0; // gomito

// posizione del polso del tre dof
float xh = 0.0;
float yh  = 0.0;
color target_color = color(255,0);

// orientamento desiderato
float phi = 0.0;

void setup() {
   size(800, 800);
   background(#546e7a);
}

void draw() {
   background(#546e7a);
   pushMatrix(); // salvo s.d.r.
   translate(width / 2, height / 2);   // traslo
   opSpace(L1, L2, L3); // disegno limite dello spazio operativo
   target(ray);
   treDoF(ray, L1, L2, L3, larg, q1, q2, q3, rgb); // disegno robot phantom.
   popMatrix(); // elimino s.d.r.
   controllo(K);   // inserisco legge di controllo
   pushMatrix(); // risalvo s.d.r.
   translate(width / 2 , height / 2);   // traslo
   treDoF(ray, L1, L2, L3, larg, q1v, q2v, q3v, rgbV); // disegno robot vero.
   popMatrix(); // elimino s.d.r.

   newton();
   pushMatrix(); // risalvo s.d.r.
   translate(width / 2 , height / 2);   // traslo
   treDoF(ray, L1, L2, L3, larg, q_1 - PI / 2, q_2, q_3, rgbN); // disegno robot vero.
   popMatrix(); // elimino s.d.r.

   gradient();
   // adam();
   pushMatrix(); // risalvo s.d.r.
   translate(width / 2 , height / 2);   // traslo
```

```
    treDoF(ray, L1, L2, L3, larg, q_1g - PI / 2, q_2g, q_3g, rgbG); // disegno robot vero.
    popMatrix(); // elimino s.d.r.

    textSize(20);
    textLeading(20);
    fill(rgb);
    text("Robot Phantom", 5, 30);
    fill(rgbV);
    text("Robot Vero", 5, 50);
    fill(rgbN);
    text("Robot Newton", 5, 70);
    fill(rgbG);
    text("Robot Gradiente", 5, 90);
    fill(255);
    if (target_color == rgb)
        text("Punto non raggiungibile o orientamento \nnon ammissibile. Selezionare un nuovo punto.", width / 2 - 50, 30);
    fill(0);
    text("Guadagno Q1 = " + kg1 + " Q2 gradiente = " + kg2, 0, height - 50);
    text("Guadagno Q3 gradiente = " + kPhi, 0, height - 30);
    pushMatrix();
    translate(width / 2, height / 2);   // traslo
    popMatrix();
}

// definisco legge di controllo
void controllo(float K) {
    q1v = q1v + K * (q1 - q1v);
    q2v = q2v + K * (q2 - q2v);
    q3v = q3v + K * (q3 - q3v);
}

void treDoF(int ray, int L1, int L2, int L3, int larg, float q1, float q2, float q3, color rgb) {
    rotate(q1);
    link(ray,L1,larg,rgb);
    translate(0 ,L1);
    rotate(q2);
    link(ray,L2,larg,rgb);
    translate(0, L2);
    rotate(q3);
    link(ray,L3,larg,rgb);
}

void link(int R, int lung, int larg, color rgb) {
    fill(rgb);
    circle(0,0,R);
    rect( - R / 2, 0, larg, lung);
    circle(0, lung, R);
}

void keyPressed() {
    // istruzione di home
    if (keyCode == ENTER) {
        q1 = 0.0;
        q2 = 0.0;
        q3 = 0.0;
        q_1 = 0.0;
        q_2 = 0.0;
        q_3 = 0.0;
        phi = 0.0;
        phig = 0.0;
        y = L1 + L2 + L3;
        x = 0.0;
        yh = 0.0;
        xh = 0.0;
        target(ray);
    }
    // Controllo sulle variabili di giunto.
```

```
    if (keyCode == '1') q1 = q1 + 0.1;
    if (keyCode == '2') q2 = q2 + 0.1;
    if (keyCode == '3') q3 = q3 + 0.1;

    if (keyCode == '9') q1 = q1 - 0.1;
    if (keyCode == '8') q2 = q2 - 0.1;
    if (keyCode == '7') q3 = q3 - 0.1;

    // cambio il gomito
    if (keyCode == 'G') {
        g = - g;
        cinInv();
        q_2 = q_2 + g * PI;
        q_2g = q_2g + g * PI;
    }

    // cabio angolo d'orientamento desiderato
    if (keyCode == 'F') {
        phi = phi + 0.1;
        phig += 0.1;
        cinInv();
    } else if (keyCode == 'V') {
        phi = phi - 0.1;
        phig -= 0.1;
        cinInv();
    }
}

void opSpace(int r1, int r2, int r3) {
    fill(255);
    circle(0,0, 2 * (r1 + r2 + r3));  // devo moltiplicare per due perché per circle() il terzo parametro è il raggio.
}

void target(int R) {
    pushMatrix();
    translate(x, y);
    fill(target_color);
    circle(0,0, R + 30);
    popMatrix();
}

void mousePressed() {
    x = mouseX - width / 2;
    y = mouseY - height / 2;

    cinInv();

    if (abs(s_2g) < 0.01) {
        println("q2 singolare");
    }
    if (abs(s_3g) < 0.01) {
        println("q3 singolare");
    }

}

// Variabili per la cinematica inversa
float A = 0.0;
float c2 = 0.0;
float s2 = 0.0;
float b1 = 0.0;
float b2 = 0.0;
float c1 = 0.0;
float s1 = 0.0;
float c3 = 0.0;
float s3 = 0.0;

```

```
void cinInv() {
   xh = x - cos(phi) * L3;
   yh = y - sin(phi) * L3;
   // inserisco controllo per determinare se sto violando lo spazio operativo
   if (xh * xh + yh * yh <= pow(L1 + L2,2) &&
      xh * xh + yh * yh >= pow(L1 - L2,2)) {
      target_color = rgbV;
   } else {
      target_color = rgb;
   }
   A = (xh * xh + yh * yh - L1 * L1 - L2 * L2) / (2 * L1 * L2);
   c2 = A;
   s2 = g * sqrt(abs(1 - A * A));
   b1 = L1 + c2 * L2;
   b2 = L2 * s2;
   c1 = b1 * xh + b2 * yh;
   s1 = - b2 * xh + b1 * yh;
   q1 = atan2(s1, c1) - PI / 2;
   q2 = atan2(s2, c2);
   q3 = phi - q1 - q2 - PI / 2;
}

// Creo un robot phantom la cui cinematica inversa è determinata dall'algoritmo di Newton.
float q_1 = PI / 2;   // parto da PI/2 per portare il robot verso il basso come condizione iniziale
float q_2 = 0.0;
float q_3 = 0.0;
color rgbN = color(104,2,151,100);
float s_123 = 0.0;
float c_123 = 0.0;
float s_12 = 0.0;
float c_12 = 0.0;
float s_23 = 0.0;
float c_23 = 0.0;
float c_1 = 0.0;
float s_1 = 0.0;
float c_2 = 0.0;
float s_2 = 0.0;
float c_3 = 0.0;
float s_3 = 0.0;
float Q1 = 0.0;
float Q2 = 0.0;
float Q3 = 0.0;

void newton() {
   s_123 = sin(q_1 + q_2 + q_3);
   c_123 = cos(q_1 + q_2 + q_3);
   s_12 = sin(q_1 + q_2);
   c_12 = cos(q_1 + q_2);
   s_23 = sin(q_3 + q_2);
   c_23 = cos(q_3 + q_2);
   c_1 = cos(q_1);
   s_1 = sin(q_1);
   c_2 = cos(q_2);
   s_2 = sin(q_2);
   c_3 = cos(q_3);
   s_3 = sin(q_3);

   // Poiché sin(q2)=0 all'inizio, siamo già in configurazione singolare pertanto l'algoritmo non funzionerebbe
   // Devo effettuare una verifica e, in caso positivo, effettuare una saturazione ad un valore di riferimento noto.
   if (abs(s_2) < 0.01) s_2 = 0.01; // impedisco all'algoritmo di andare in singolarità.

   Q1 = (s_12 * (y - s_123 * L3 - s_12 * L2 - s_1 * L1)) / (s_2 * L1) +
        (c_12 * (x - c_123 * L3 - c_12 * L2 - c_1 * L1)) / (s_2 * L1) +
        (s_3 * L3 * (phi - q_3 - q_2 - q_1)) / (s_2 * L1);

   Q2 = ((- s_12 * L2 - s_1 * L1) * (y - s_123 * L3 - s_12 * L2 - s_1 * L1)) / (s_2 * L1 * L2) +
        ((- c_12 * L2 - c_1 * L1) * (x - c_123 * L3 - c_12 * L2 - c_1 * L1)) / (s_2 * L1 * L2) +
```

```
        ((- s_3 * L2 * L3 - s_23 * L1 * L3) * (phi - q_3 - q_2 - q_1)) / (s_2 * L1 * L2);

    Q3 = (s_1 * (y - s_123 * L3 - s_12 * L2 - s_1 * L1)) / (s_2 * L2) +
        (c_1 * (x - c_123 * L3 - c_12 * L2 - c_1 * L1)) / (s_2 * L2) +
        ((s_23 * L1 * L3 + s_2 * L1 * L2) * (phi - q_3 - q_2 - q_1)) / (s_2 * L1 * L2);
    // lambda/2
    float kk = 0.2;
    // implemento Newton tempo discreto
    q_1 = q_1 + kk * Q1;
    q_2 = q_2 + kk * Q2;
    q_3 = q_3 + kk * Q3;
}

float q_1g = PI / 2;
float q_2g = 0.0;
float q_3g = 0.0;
color rgbG = color(0,0,255,100);
float s_123g = 0.0;
float c_123g = 0.0;
float s_12g = 0.0;
float c_12g = 0.0;
float s_23g = 0.0;
float c_23g = 0.0;
float c_1g = 0.0;
float s_1g = 0.0;
float c_2g = 0.0;
float s_2g = 0.0;
float c_3g = 0.0;
float s_3g = 0.0;
float Q1g = 0.0;
float Q2g = 0.0;
float Q3g = 0.0;
float phig = 0.0;
// Limite di convergenza 10^5
float kg1 = 0.00001;
float kg2 = 0.00001;
float kPhi = 0.0000000001;

void gradient() {
    s_123g = sin(q_1g + q_2g + q_3g);
    c_123g = cos(q_1g + q_2g + q_3g);
    s_12g = sin(q_1g + q_2g);
    c_12g = cos(q_1g + q_2g);
    s_23g = sin(q_3g + q_2g);
    c_23g = cos(q_3g + q_2g);
    c_1g = cos(q_1g);
    s_1g = sin(q_1g);
    c_2g = cos(q_2g);
    s_2g = sin(q_2g);
    c_3g = cos(q_3g);
    s_3g = sin(q_3g);

    if (abs(s_2g) < 0.01) s_2g = 0.5;
    if (abs(s_3g) < 0.01) s_3g = 0.5;

    Q1g = (c_123g * L3 + c_12g * L2 + c_1g * L1) * (y - s_123g * L3 - s_12g * L2 - s_1g * L1) +
          (- s_123g * L3 - s_12g * L2 - s_1g * L1) * (x - c_123g * L3 - c_12g * L2 - c_1g * L1) +
          phig - q_3g - q_2g - q_1g;
    Q2g = (c_123g * L3 + c_12g * L2) * (y - s_123g * L3 - s_12g * L2 - s_1g * L1) +
          (- s_123g * L3 - s_12g * L2) * (x - c_123g * L3 - c_12g * L2 - c_1g * L1) +
          phig - q_3g - q_2g - q_1g;
    Q3g = c_123g * L3 * (y - s_123g * L3 - s_12g * L2 - s_1g * L1) -
          s_123g * L3 * (x - c_123g * L3 - c_12g * L2 - c_1g * L1) + phig - q_3g - q_2g - q_1g;

    phig = phig + kPhi * Q3g;

    q_1g = q_1g + kg1 * Q1g;
```

```
    q_2g = q_2g + kg2 * Q2g;
    q_3g = phig - q_1g - q_2g;

    println("q1G = " + q_1g + "q2G = " + q_2g + " q3G = " + q_3g);
}
```

Strutture Portanti in Processing

Nelle pagine successive verrà listato il codice processing per la realizzazione dei diversi robot visti nel corso, in ambiente virtuale 3D.

E' possibile interagire con l'ambiente creato nei seguenti modi:

1. Ciclare tra le diverse strutture implementate nel sistema con il tasto "S".
2. Ruotare la telecamera con il tasto sinistro del mouse
3. Aumentare il valore delle variabili di giunto attravero i tasti "1-2-3" per i primi 3 gradi di libertà, "7-8-9" per gli ultimi 3.
4. Diminuire il valore delle variabili di giunto attravero i tasti "Q,W,E" per i primi 3 gradi di libertà, "Y,U,I" per gli ultimi 3.

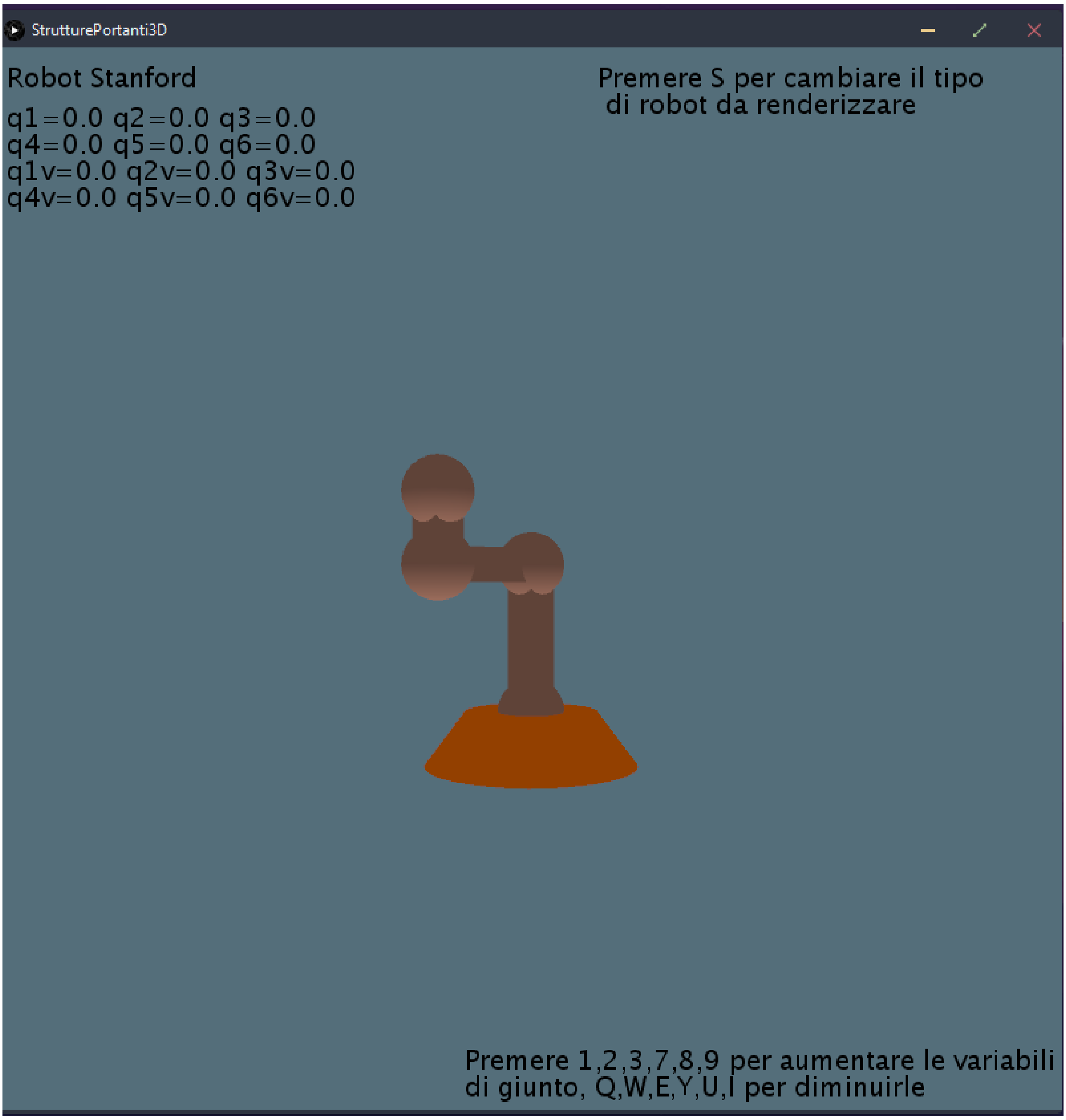


Figura 1 Robot Stanford a 3 DoF

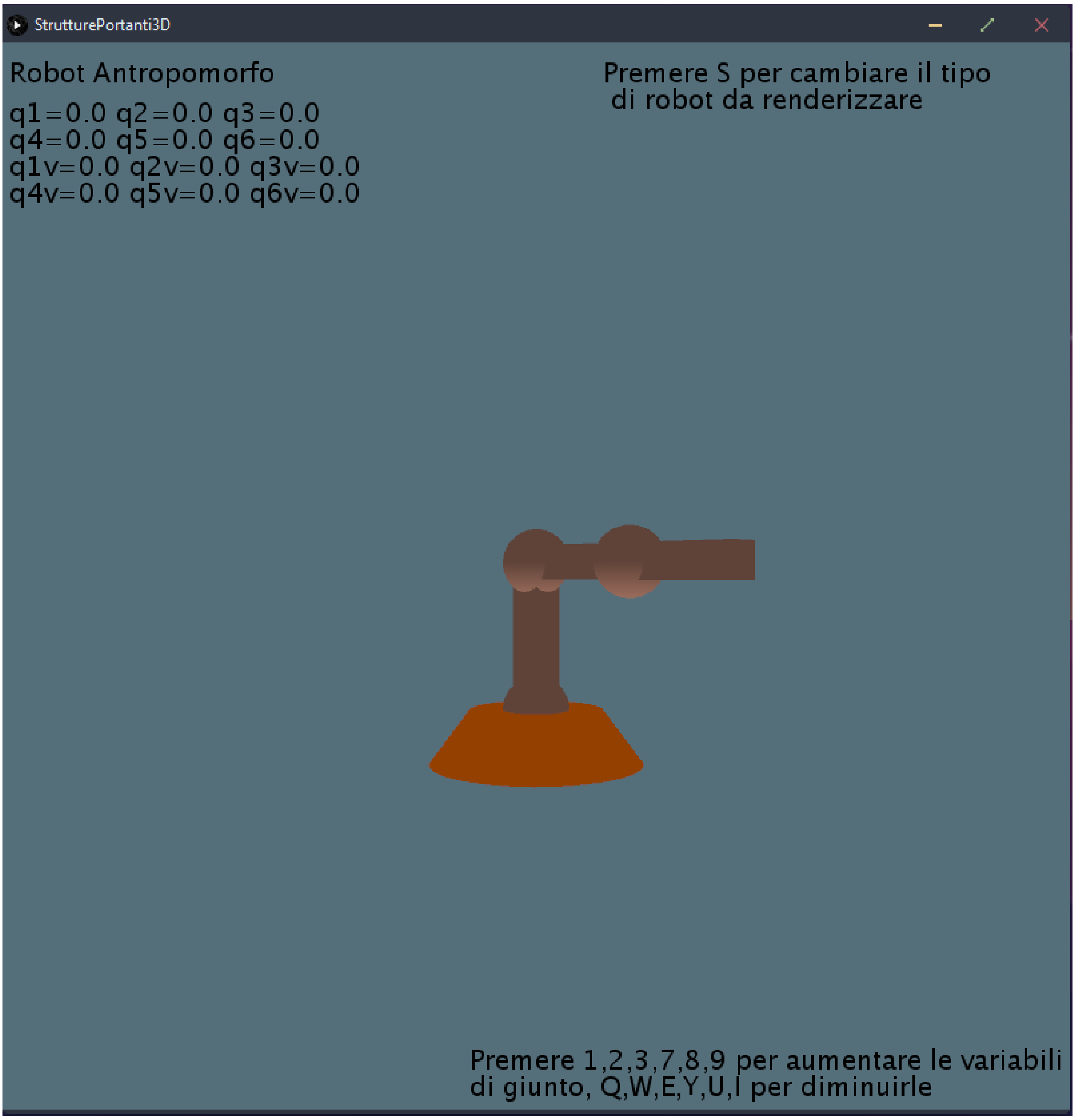


Figura 2 Robot Antropomorfo

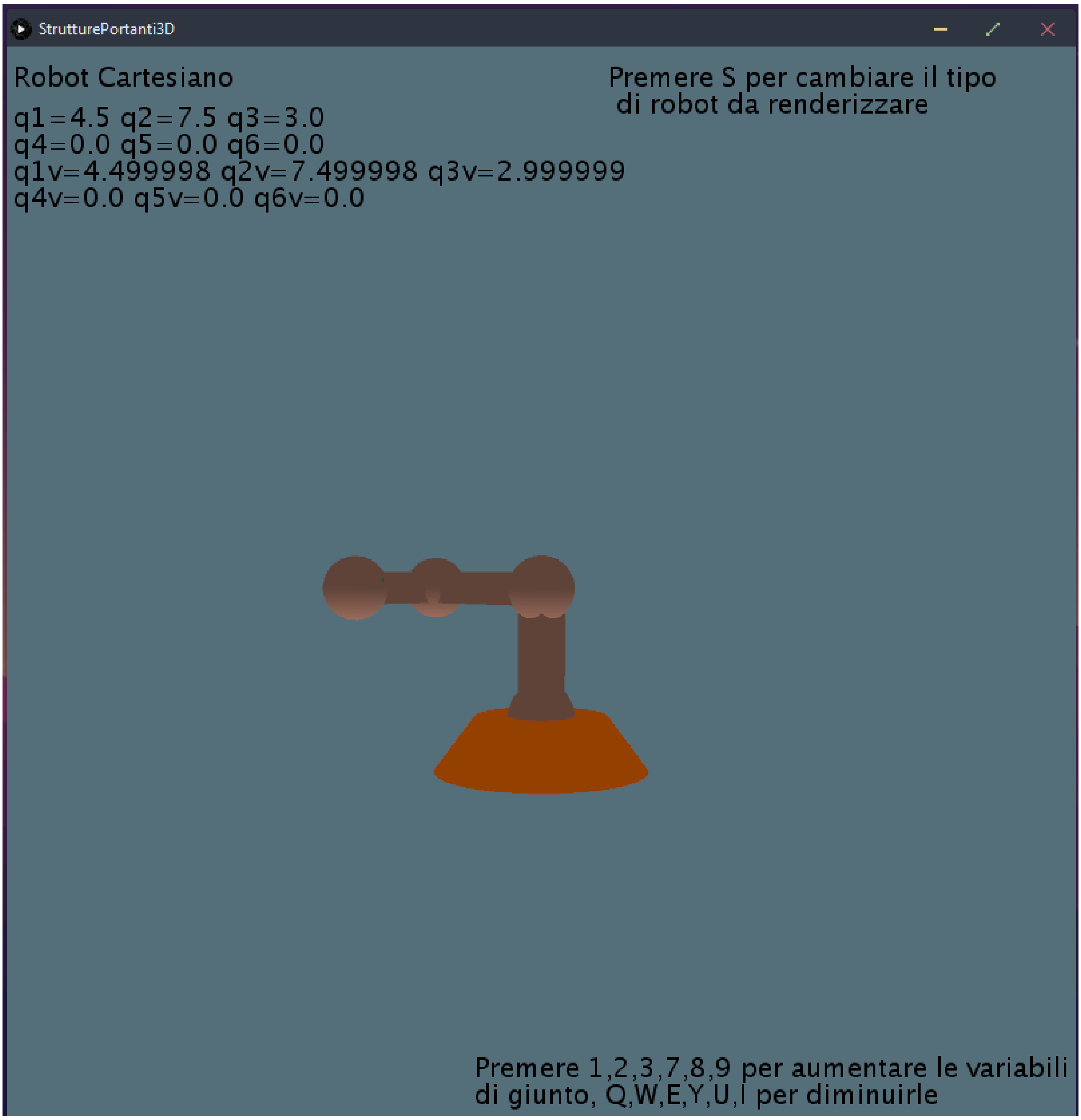


Figura 3 Robot Cartesiano

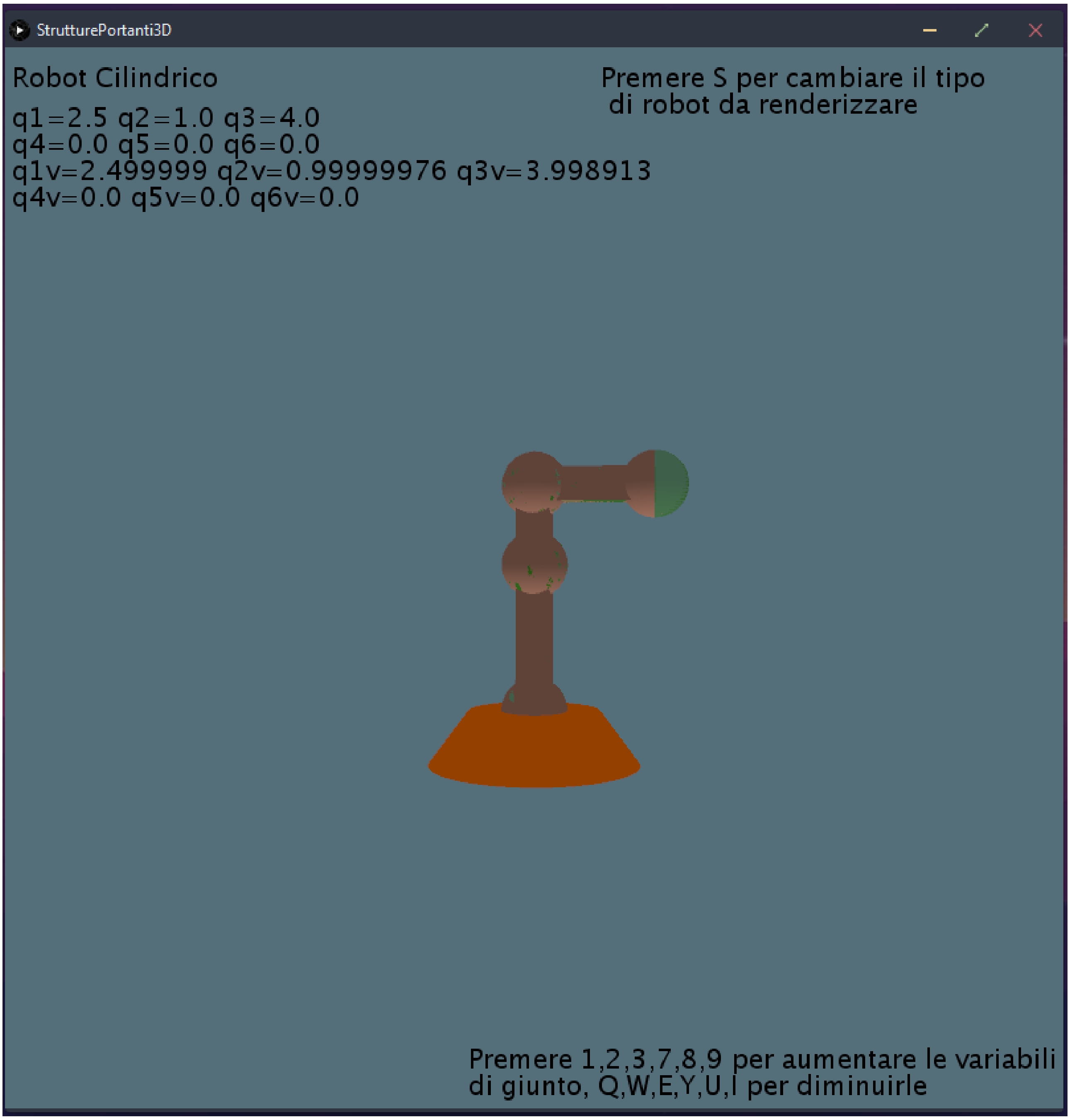


Figura 4 Robot Cilindrico

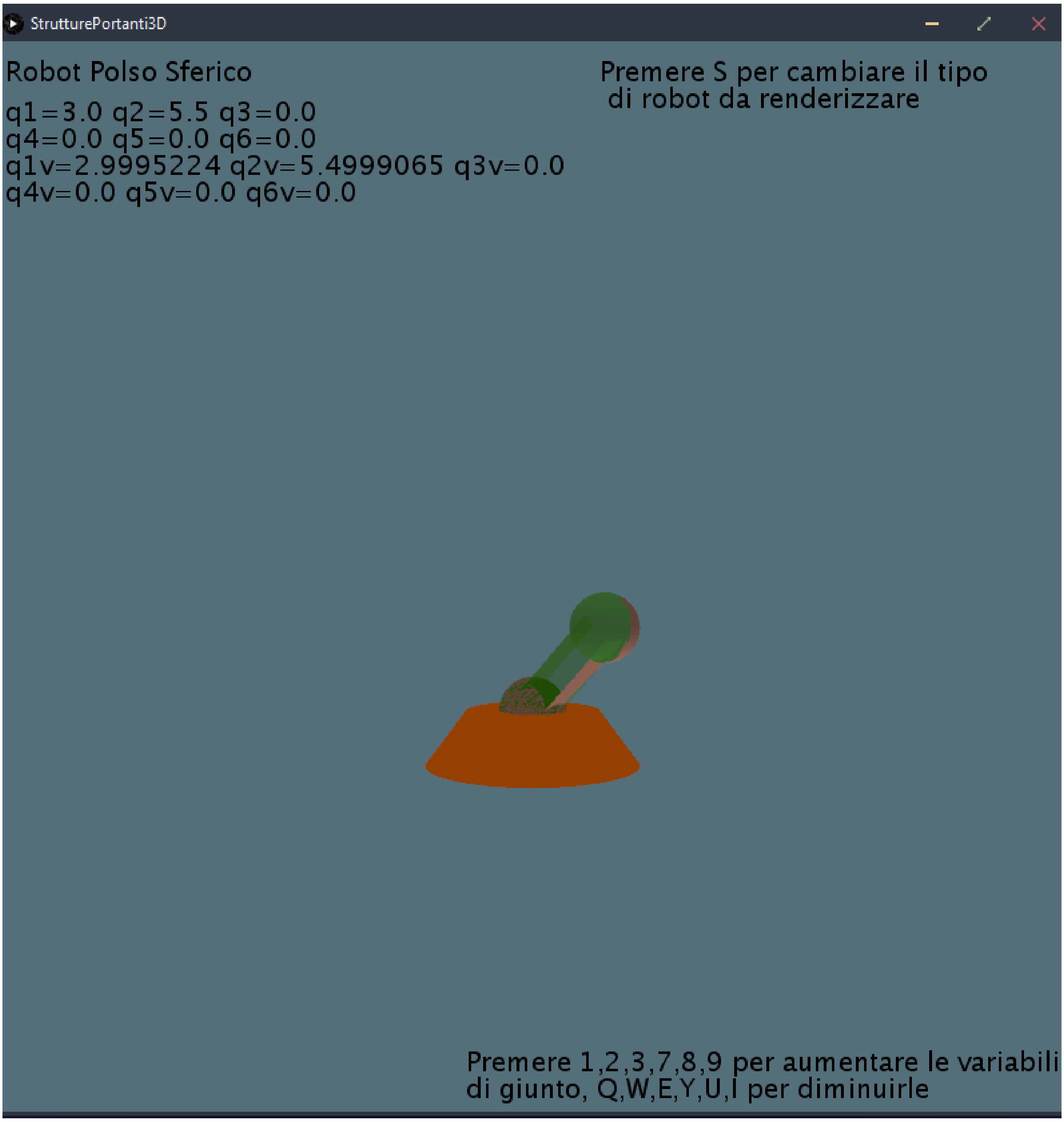


Figura 5 Polso Sferico

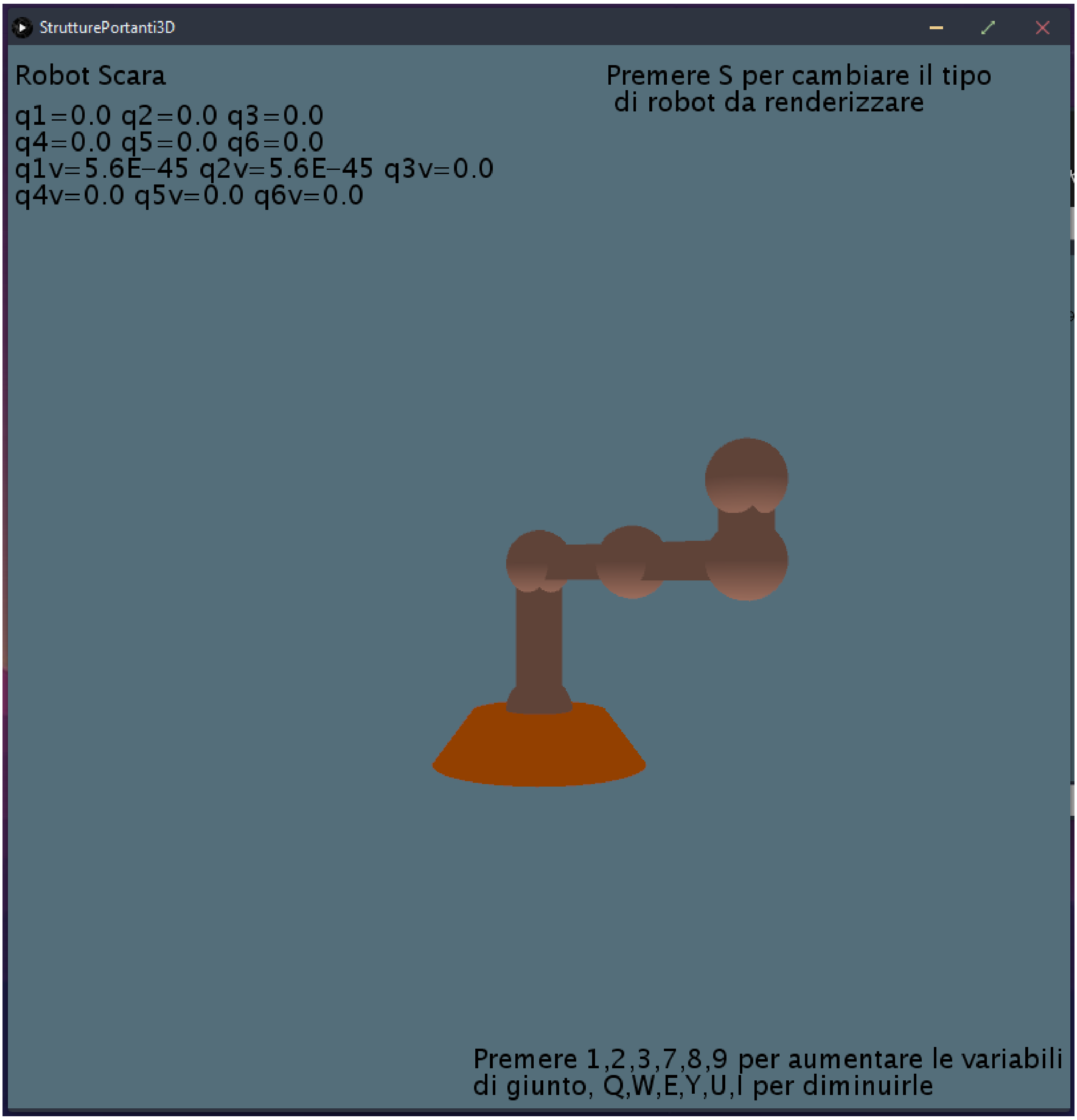


Figura 6 Robot SCARA

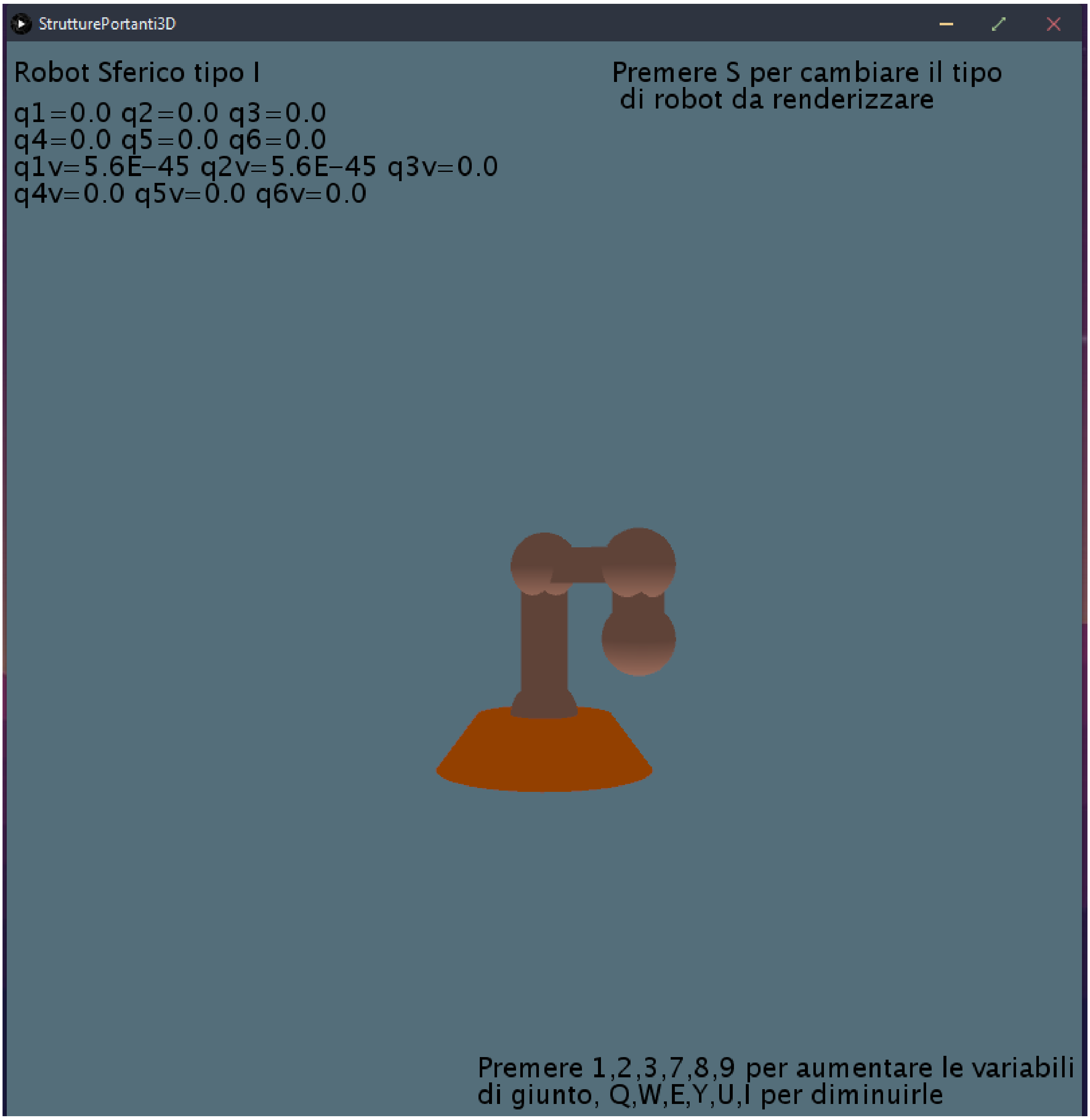


Figura 7 Robot Sferico Tipo 1

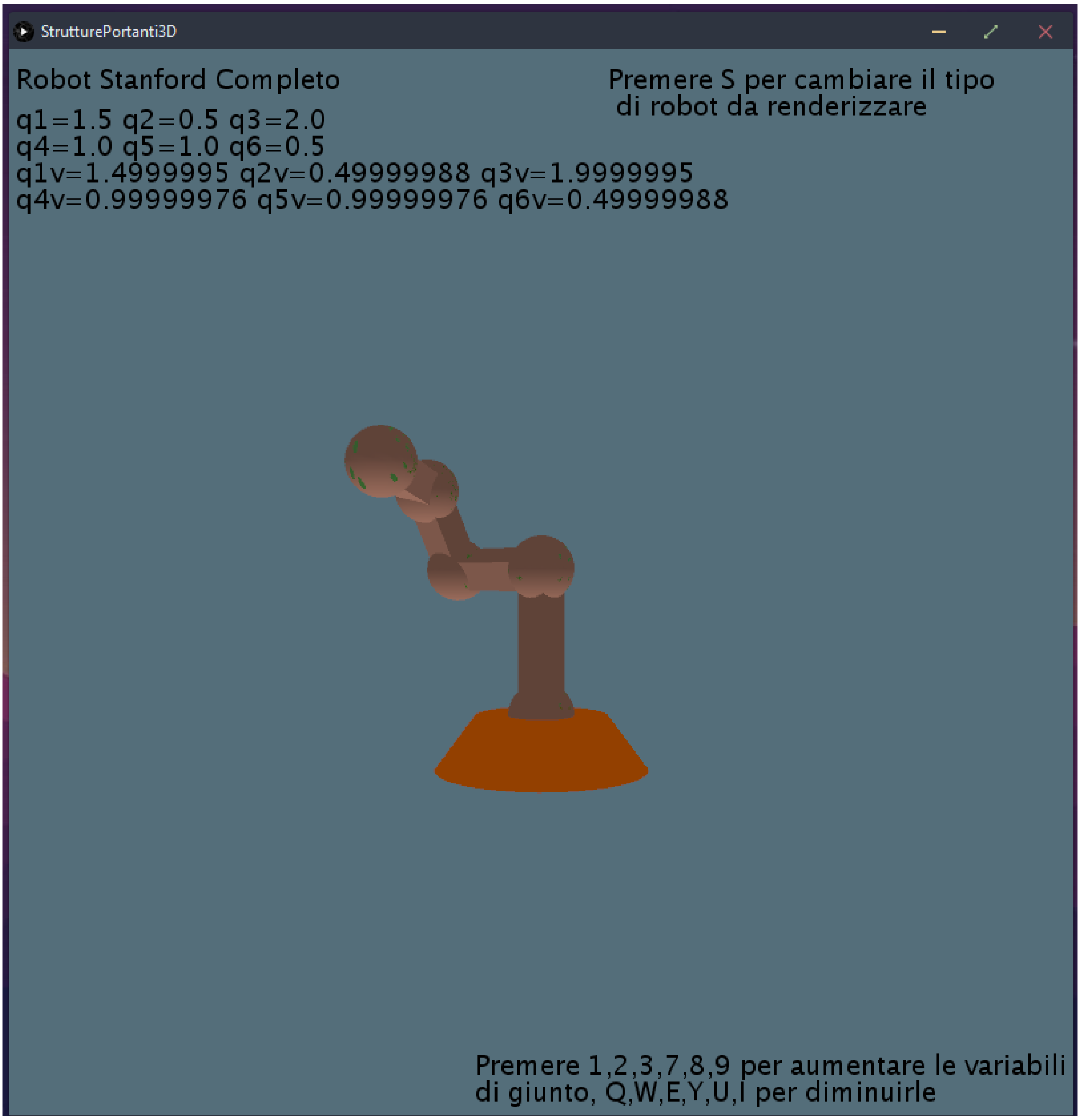


Figura 8 Robot Stanford Completo

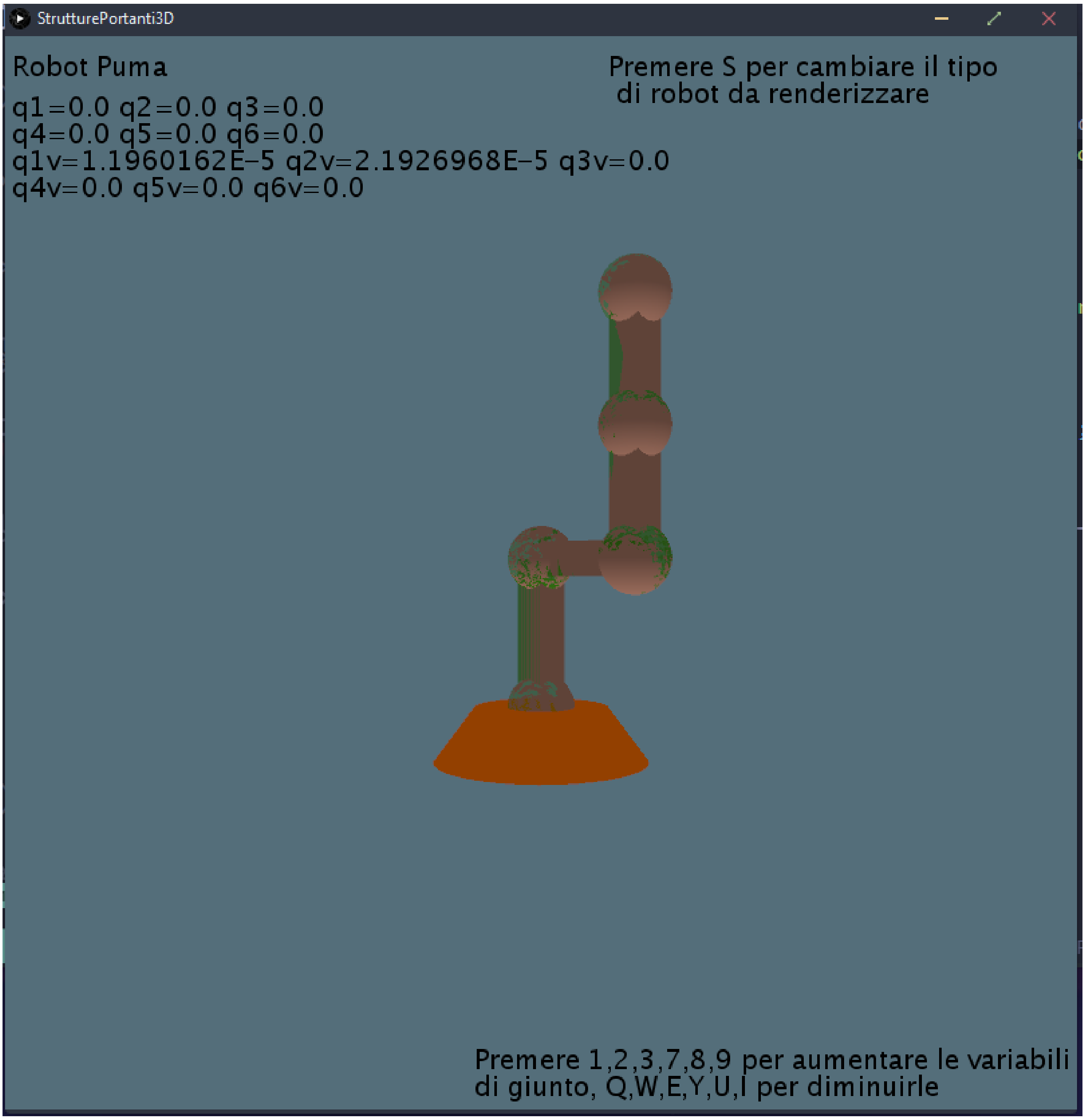


Figura 9 Robot PUMA

```
/*
Questo programma processing definisce un ambiente 3D che riceve in ingresso la tabella di
Denavit-Hartenberg e restituisce un robot in tre dimensioni.
*/
// Definisco Variabili di Giunto fantasma.
float q1 = 0.0;
float q2 = 0.0;
float q3 = 0.0;
float q4 = 0.0;
float q5 = 0.0;
float q6 = 0.0;
// Definisco Variabili di Giunto per il robot effettivo.
float q1v = 0.0;
float q2v = 0.0;
float q3v = 0.0;
float q4v = 0.0;
float q5v = 0.0;
float q6v = 0.0;
// Definisco Variabili di visuale, per orientare la prospettiva del disegno 3D
float alphaZ = 0.0;
float alphaY = 0.0;
// Aggiungo angolo di partenza.
float alphaZp = 0.0;
float alphaYp = 0.0;
// Definisco costante di proporzionalitĂ  per il controllore
float K = 0.1;

// Definisco variabile per cambiare il robot selezionato.
int s = 0;
// Definisco lunghezze dei link
float L1 = 110;
float D1 = 90;
float D2 = 90;
float L4 = 90;
float D6 = 90;
float L2 = 90;
float D3 = 90;
float L3 = 90;
float L6 = 90;

void setup() {
 size(800,800,P3D);
 background(#546e7a);
}

// definisco legge di controllo
void control(float K) {
 q1v = q1v - K * (q1v - q1);
 q2v = q2v - K * (q2v - q2);
 q3v = q3v - K * (q3v - q3);
 q4v = q4v - K * (q4v - q4);
 q5v = q5v - K * (q5v - q5);
 q6v = q6v - K * (q6v - q6);
}
/**
Assemblo il robot formato da diversi link
*/
void robot(int s, float a1, float a2, float a3, float a4, float a5, float a6) {
 switch(s) {
  case 0 :    // ROBOT STANFORD
  link(a1, L1, - PI / 2, 0);
  link(a2, L2, PI / 2, 0);
  link(- PI / 2, 10 * a3 + 50, 0, 0);
  break;
  case 1 :    // ROBOT ANTROPOMORFO
  link(a1, L1, PI / 2, 0);
```

```
  link(a2, 0, 0, D2);
  link(a3, 0, 0, D3);
  break;

  case 2 :     // ROBOT CARTESIANO
  link(0, 10 * a1 + 50, - PI / 2, 0);
  link(- PI / 2, 10 * a2 + 50, - PI / 2, 0);
  link(0, 10 * a3 + 50, 0, 0);
  break;
  case 3 :    // ROBOT CILINDRICO
  link(a1, L1, 0, 0);
  link(0, 10 * a2 + 50, - PI / 2, 0);
  link(0, 10 * a3 + 50, 0, 0);
  break;
  case 4 :    // POLSO SFERICO
  link(a1, 0, - PI / 2, 0);
  link(a2, 0, PI / 2, 0);
  link(a3, L3, 0, 0);
  break;
  case 5 :     // ROBOT PUMA
  link(a1, L1, - PI / 2, 0);
  link(a2, 0, 0, D2);
  link(a3, 0,  PI / 2, 0);
  link(a4, L4, - PI / 2, 0);
  link(a5, 0, PI / 2, 0);
  link(a6, L6, 0, 0);
  break;
  case 6 :    // ROBOT SCARA
  link(a1, L1, 0, D1);
  link(a2, 0, 0, D2);
  link(0, 10 * a3 + 50, 0, 0);
  break;
  case 7 :    // ROBOT SFERICO TIPO 1
  link(a1, L1, - PI / 2, 0);
  link(a2, 0, - PI / 2, L2);
  link(0, 10 * a3 + 50, 0, 0);
  break;
  case 8 :    // ROBOT STANFORD COMPLETO
  link(a1, L1, - PI / 2, 0);
  link(a2, L2, PI / 2, 0);
  link( - PI / 2, 10 * a3 + 50, 0, 0);
  link(a4, 0, - PI / 2, 0);
  link(a5, 0, PI / 2, 0);
  link(a6, L6, 0, 0);
  break;
  default :
  break;
 }
}
/**
Costruisce il link del robot corrispondente alla riga i-esima della tabella di DH.
param: theta, angolo di rotazione su Z.
param: d, traslazione lungo Z.
param: alpha, angolo di rotazione su X.
param: a, traslazione lungo X.
*/
void link(float theta, float d, float alpha, float a) {
 rotateZ(theta);
 sphere(25);
 // Traslo il disegno per avre una buona giunzione tra giunto rotoidale e braccio
 translate(0.0,0.0,d / 2);
 box(25, 25, d);
 // Il centro del S.d.R. Ă¨ al centro della box. Devo traslare fino alla sua fine.
 translate(0.0,0.0,d / 2);
 sphere(25);
 rotateX(alpha);
 translate(a / 2, 0.0, 0.0);
```

```
 box(a, 25, 25);
 translate(a / 2, 0.0, 0.0);
}
/**
Costruisco la base del robot
Preso da: https://forum.processing.org/one/topic/draw-a-cone-cylinder-in-p3d.html
*/
void base(float bottom, float top, float h, int sides) {
 pushMatrix();

 translate(0,0, - h / 2);
 rotateX(PI / 2);
 float angle;
 float[] x = new float[sides + 1];
 float[] z = new float[sides + 1];

 float[] x2 = new float[sides + 1];
 float[] z2 = new float[sides + 1];

 //get the x and z position on a circle for all the sides
 for (int i = 0; i < x.length; i++) {
  angle = TWO_PI / (sides) * i;
  x[i] = sin(angle) * bottom;
  z[i] = cos(angle) * bottom;
 }

 for (int i = 0; i < x.length; i++) {
  angle = TWO_PI / (sides) * i;
  x2[i] = sin(angle) * top;
  z2[i] = cos(angle) * top;
 }

 //draw the bottom of the cylinder
 beginShape(TRIANGLE_FAN);

 vertex(0,   - h / 2,    0);

 for (int i = 0; i < x.length; i++) {
  vertex(x[i], - h / 2, z[i]);
 }

 endShape();

 //draw the center of the cylinder
 beginShape(QUAD_STRIP);

 for (int i = 0; i < x.length; i++) {
  vertex(x[i], - h / 2, z[i]);
  vertex(x2[i], h / 2, z2[i]);
 }

 endShape();

 //draw the top of the cylinder
 beginShape(TRIANGLE_FAN);

 vertex(0,   h / 2,    0);

 for (int i = 0; i < x.length; i++) {
  vertex(x2[i], h / 2, z2[i]);
 }

 endShape();

 popMatrix();
}
void draw() {
```

```
 background(#546e7a);
 // Aggiungo legge di controllo
 control(K);
 // Aggiungo HUD
 hud();
 //  traslo al centro della schermata
 translate(width / 2, height / 2 + 100, 0);

 // aggiungo una rotazione per favorire la vista
 rotateX(PI / 2);
 rotateZ(PI / 4);
 rotateY(alphaY);
 rotateZ(alphaZ);
 // Aggiungo degli effetti di luce direzionale
 directionalLight(126, 126, 126, 0, 0, 0.7);
 // Aggiungo degli effetti di luce ambientale
 ambientLight(200, 200, 200);
 noStroke();
 // Disegno base
 fill(#bc5100);
 base(80, 50, 40, 50);

 // Disegno il robot fantasma
 fill(#255d00,100);
 pushMatrix();
 robot(s, q1, q2, q3, q4, q5, q6);
 popMatrix();
 pushMatrix();
 fill(#795548);
 robot(s, q1v, q2v, q3v, q4v, q5v, q6v);
 popMatrix();
}

int k = 5;
// intercetto azioni del mouse
void mousePressed() {    // eseguo alla pressione

 alphaYp = alphaY + mouseY * PI / (180*k);
 alphaZp = alphaZ + mouseX * PI / (180*k);
}
void mouseDragged() {   // eseguo al trascinamento
 alphaY = alphaYp + mouseY * PI / (180*k);
 alphaZ = alphaZp + mouseX * PI / (180*k);
}
// intercetto azioni da tastiera
void keyPressed() {
 // Controllo se voglio aumentare o diminuire le variabili di giunto
 if (keyCode == '1') {
  q1 += 0.5;
 } else if (keyCode == 'Q') {
  q1 -= 0.5;
 }
 if (keyCode == '2') {
  q2 += 0.5;
 } else if (keyCode == 'W') {
  q2 -= 0.5;
 }
 if (keyCode == '3') {
  q3 += 0.5;
 } else if (keyCode == 'E') {
  q3 -= 0.5;
 }
 if (keyCode == '7') {
  q4 += 0.5;
 } else if (keyCode == 'Y') {
  q4 -= 0.5;
 }
```

```
 if (keyCode == '8') {
  q5 += 0.5;
 } else if (keyCode == 'U') {
  q5 -= 0.5;
 }
 if (keyCode == '9') {
  q6 += 0.5;
 } else if (keyCode == 'I') {
  q6 -= 0.5;
 }

 if (keyCode == ENTER) {
  q1 = 0;
  q2 = 0;
  q3 = 0;
  q4 = 0;
  q5 = 0;
  q6 = 0;
 }

 if (keyCode == 'S') {
  if (s < 8) {
   s +=1;
  } else {
   s = 0;
  }
  println(s);
  q1 = 0;
  q2 = 0;
  q3 = 0;
  q4 = 0;
  q5 = 0;
  q6 = 0;
 }
}

// Creo interfaccia utente
void hud() {
 String joints = "q1=" + q1 + "\t q2=" + q2 + "\t q3=" + q3 + "\nq4=" + q4 + "\t q5=" + q5 + "\t q6=" + q6;
 String jointsv = "q1v=" + q1v + "\t q2v=" + q2v + "\t q3v=" + q3v + "\nq4v=" + q4v + "\t q5v=" + q5v + "\t q6v=" + q6v;
 String[] robotName = {"Stanford","Antropomorfo","Cartesiano","Cilindrico","Polso Sferico","Puma",
              "Scara","Sferico tipo I","Stanford Completo"};
 textSize(20);
 textLeading(20);
 fill(0);
 text("Premere 1,2,3,7,8,9 per aumentare le variabili\ndi giunto, Q,W,E,Y,U,I per diminuirle",width / 2-50 ,height - 30);
 text("Robot " + robotName[s], 5, 30);
 text(joints,5,60);
 text(jointsv,5,100);
 text("Premere S per cambiare il tipo\n di robot da renderizzare",width/2+50,30);
}
```

Singolarità Cinematiche (di forza e velocità) delle Strutture Portanti

Le singolarità cinematiche di posizione sono state già individuate durante i calcoli della cinematica inversa di ogni robot, pertanto per semplicità verranno omesse da questa parte del rapporto.
Per il calcolo delle singolarità di velocità e forza, abbiamo bisogno di calcolare lo Jacobiano di velocità, che corrisponde alla matrice delle derivate parziali dell'end-effector di ogni robot.
Pertanto, sia:

$$P(h(q)) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Allora, lo Jacobiano J sarà:

$$J = \frac{\partial h}{\partial q} = \begin{pmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} \end{pmatrix}$$

Il determinante di J ci fornirà le equazioni per individuare le singolarità di velocità del robot.
Per le singolarità di forza, dovremo calcolare invece $\det(J^T)$.
Avremo quindi che:

$$\ker(J) \longrightarrow \text{Spazio di tutte le velocità che posso dare ai giunti, che danno luogo } a \text{ velocità nulla in punta.}$$

$$\mathrm{Im}(J) \longrightarrow \text{Spazio di tutte le velocità che posso raggiungere in punta}$$

```
(%i1) J(p):=block(
      [J, jq1, jq2, jq3],
          jq1:diff(p,q[1]), jq2:diff(p,q[2]), jq3:diff(p,q[3]),
          J:addcol(jq1,jq2,jq3), return(J)
      )$
(%i2) ridef(M):=block(
      let(sin(q[1]), s[1]), let(cos(q[1]), c[1]),
      let(sin(q[2]), s[2]), let(cos(q[2]), c[2]),
      let(sin(q[2]+q[1]), s[12]), let(cos(q[2]+q[1]), c[12]),
      let(sin(q[2]+q[3]), s[23]), let(cos(q[2]+q[3]), c[23]),
      return(letsimp(M))
      )$
```

1. Robot 2 DoF Planare

```
(%i3) p2dof:matrix([L[1]*cos(q[1])+L[2]*cos(q[1]+q[2])],
                   [L[1]*sin(q[1])+L[2]*sin(q[1]+q[2])])
```

(%o3) $\begin{pmatrix} L_2 \cos(q_2+q_1) + L_1 \cos(q_1) \\ L_2 \sin(q_2+q_1) + L_1 \sin(q_1) \end{pmatrix}$

```
(%i4) J2dof:submatrix(J(p2dof),3)
```

(%o4) $\begin{pmatrix} -L_2 \sin(q_2+q_1) - L_1 \sin(q_1) & -L_2 \sin(q_2+q_1) \\ L_2 \cos(q_2+q_1) + L_1 \cos(q_1) & L_2 \cos(q_2+q_1) \end{pmatrix}$

```
(%i5) dJ2dof:trigreduce(expand(determinant(J2dof)))
```

(%o5) $L_1 L_2 \sin(q_2)$

Singolarità di Velocità

Vediamo quindi che

$$\det(J) = L_1 L_2 \sin(q_2) = 0 \Longleftrightarrow q_2 = \begin{cases} 0 \\ \pi \end{cases}$$

Che corrispondono ad una posizione completamente allungata o sovrapposta del secondo link.
Vediamo ora le velocità che non possiamo raggiungere, cercando il nucleo di J quando siamo in singolarità.

```
(%i6) Jq2:factor(subst(q[2]=0,J2dof))
```

$$\text{(\%o6)} \begin{pmatrix} -(L_2+L_1)\sin(q_1) & -L_2\sin(q_1) \\ (L_2+L_1)\cos(q_1) & L_2\cos(q_1) \end{pmatrix}$$

```
(%i7) nullspace(Jq2)
```

Proviso: notequal$(-L_2\sin(q_1), 0)$

$$\text{(\%o7)}\ \operatorname{span}\left(\begin{pmatrix} -L_2\sin(q_1) \\ (L_2+L_1)\sin(q_1) \end{pmatrix}\right)$$

Abbiamo quindi che:

$$\ker(J_{q_2=0}) = \operatorname{span}\begin{pmatrix} -L_2 \\ L_2+L_1 \end{pmatrix}$$

Per quanto riguarda le velocità che l'end-effector non può ragiungere, per il teorema di nullità più rango sappiamo che l'immagine avrà dimensione 1. Pertanto:

```
(%i8) columnspace(subst(q[1]=0,Jq2))
```

Proviso: notequal$(L_2+L_1, 0)$

$$\text{(\%o8)}\ \operatorname{span}\left(\begin{pmatrix} 0 \\ L_2+L_1 \end{pmatrix}\right)$$

E possiamo quindi vedere che

$$\operatorname{Im}(J_{q_2=0}) = \operatorname{span}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

In conclusione, in punta ci troviamo ad avere velocità nulle lungo l'asse x ed arbitrarie lungo y.

Singolarità di Forza

Le singolarità di forza sono date dal trasposto dello jacobiano.

```
(%i9) Jt2dof:-transpose(J2dof)
```

$$\text{(\%o9)} \begin{pmatrix} L_2\sin(q_2+q_1)+L_1\sin(q_1) & -L_2\cos(q_2+q_1)-L_1\cos(q_1) \\ L_2\sin(q_2+q_1) & -L_2\cos(q_2+q_1) \end{pmatrix}$$

```
(%i10) dJt2dof:trigreduce(expand(determinant(Jt2dof)))
```

$$\text{(\%o10)}\ L_1 L_2 \sin(q_2)$$

Vediamo quindi che

$$\det(J^T) = L_1 L_2 \sin(q_2) = 0 \Longleftrightarrow q_2 = \begin{cases} 0 \\ \pi \end{cases}$$

```
(%i11) Jtq2:factor(subst(q[2]=0,Jt2dof))
```

$$\text{(\%o11)} \begin{pmatrix} (L_2+L_1)\sin(q_1) & -(L_2+L_1)\cos(q_1) \\ L_2\sin(q_1) & -L_2\cos(q_1) \end{pmatrix}$$

```
(%i12) nullspace(Jtq2)
```

Proviso: notequal$((L_2+L_1)\sin(q_1), 0)$

$$\text{(\%o12)}\ \operatorname{span}\left(\begin{pmatrix} (L_2+L_1)\cos(q_1) \\ (L_2+L_1)\sin(q_1) \end{pmatrix}\right)$$

Di conseguenza:

$$\ker(J^T_{q_2=0}) = \operatorname{span}\begin{pmatrix} \cos(q_1) \\ \sin(q_1) \end{pmatrix}$$

Abbiamo quindi individuato il sottospazio delle forze che possiamo applicare sui giunti che danno forza nulla in punta.

Per le forze che non possiamo sprigionare in punta dobbiamo calcolare l'immagine, che deve avere dimensione pari ad uno.
Supponendo $q_1 \neq 0$:

```
(%i13) columnspace(subst(q[1]=%pi/2,Jtq2))
```

Proviso: notequal$(L_2, 0)$

(%o13) $\text{span}\left(\begin{pmatrix} L_2+L_1 \\ L_2 \end{pmatrix}\right)$

Di conseguenza

$$\text{Im}(J^T_{q_2=0}) = \begin{pmatrix} -(L_2+L_1) \\ -L_2 \end{pmatrix}$$

Capiamo quindi che le forze sui giunti dipendono dalle lunghezze dei link.

---

2. Robot Cilindrico

```
(%i14) pCilindrico:matrix([-q[3]*sin(q[1])],[q[3]*cos(q[1])],[q[2]+L[1]])
```

(%o14) $\begin{pmatrix} -q_3 \sin(q_1) \\ q_3 \cos(q_1) \\ q_2 + L_1 \end{pmatrix}$

```
(%i15) JCil:J(pCilindrico)
```

(%o15) $\begin{pmatrix} -q_3 \cos(q_1) & 0 & -\sin(q_1) \\ -q_3 \sin(q_1) & 0 & \cos(q_1) \\ 0 & 1 & 0 \end{pmatrix}$

```
(%i16) dJCil:trigsimp(determinant(JCil))
```

(%o16) q_3

Singolarità di Velocità
Vediamo quindi che

$$\det(J) = 0 \Longleftrightarrow q_3 = 0$$

Ovvero, la condizione singolare è quella per la quale la punta del robot si trova sull'asse z.

```
(%i17) Jq3cil:subst(q[3]=0,JCil)
```

(%o17) $\begin{pmatrix} 0 & 0 & -\sin(q_1) \\ 0 & 0 & \cos(q_1) \\ 0 & 1 & 0 \end{pmatrix}$

Si vede subito quindi che il nucleo è identificato dal vettore e_x

$$\ker(J_{q_3=0}) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Per il teorema di nullità più rango, l'immagine deve avere dimensione 2.

```
(%i18) columnspace(Jq3cil)
```

Proviso: notequal$(\cos(q_1), 0)$

(%o18) $\text{span}\left(\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -\sin(q_1) \\ \cos(q_1) \\ 0 \end{pmatrix}\right)$

Singolarità di Forza

(%i19) `JtCil:-transpose(JCil)`

$$(\%o19)\quad \begin{pmatrix} q_3\cos(q_1) & q_3\sin(q_1) & 0 \\ 0 & 0 & -1 \\ \sin(q_1) & -\cos(q_1) & 0 \end{pmatrix}$$

(%i20) `dJtCil:trigsimp(determinant(JtCil))`

$$(\%o20)\quad -q_3$$

Abbiamo singolarità per $q_3 = 0$

(%i21) `subst(q[3]=0,JtCil)`

$$(\%o21)\quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ \sin(q_1) & -\cos(q_1) & 0 \end{pmatrix}$$

(%i22) `nullspace(subst(q[3]=0,JtCil))`

Proviso: notequal$(\cos(q_1), 0)$

$$(\%o22)\quad \mathrm{span}\left(\begin{pmatrix} \cos(q_1) \\ \sin(q_1) \\ 0 \end{pmatrix}\right)$$

(%i23) `columnspace(subst(q[3]=0,JtCil))`

Proviso: notequal$(\sin(q_1), 0) \wedge$ notequal$(-\sin(q_1), 0)$

$$(\%o23)\quad \mathrm{span}\left(\begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \sin(q_1) \end{pmatrix}\right)$$

---

3. Robot Cartesiano

(%i24) `pCart:matrix([q[3]],[q[2]],[q[1]])`

$$(\%o24)\quad \begin{pmatrix} q_3 \\ q_2 \\ q_1 \end{pmatrix}$$

(%i25) `Jcart:J(pCart)`

$$(\%o25)\quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

(%i26) `dJcart:determinant(Jcart)`

$$(\%o26)\quad -1$$

Singolarità di Velocità

Vediamo quindi che il determinante non si annulla mai per ogni valore delle variabili di giunto. Questo significa che il nucleo di J è formato dal solo vettore nullo.
Pertanto non abbiamo singolarità di velocità, il che ci porta a concludere che:

1. Non ci sono velocità che possiamo dare ai giunti che portano a velocità nulle in punta.
2. Se il nucleo è uguale al solo vettore nullo, significa che l'immagine ha rango pieno. Pertanto possiamo raggiungere qualsiasi velocità in punta.

Singolarità di Forza

```
(%i27) Jtcart:-transpose(Jcart)
```

(%o27) $\begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$

```
(%i28) dJtcart:determinant(Jtcart)
```

(%o28) 1

Tale determinante non si annulla mai e possiamo quindi trarre le medesime conclusioni per le forze che abbiamo tratto sulle velocità.

---

4. Robot Scara

```
(%i29) pscara:matrix([D[2]*cos(q[1]+q[2])+D[1]*cos(q[1])],
                    [D[2]*sin(q[1]+q[2])+D[1]*sin(q[1])],
                    [q[3]+L[1]])
```

(%o29) $\begin{pmatrix} D_2\cos(q_2+q_1)+D_1\cos(q_1) \\ D_2\sin(q_2+q_1)+D_1\sin(q_1) \\ q_3+L_1 \end{pmatrix}$

```
(%i30) Jscara:J(pscara)
```

(%o30) $\begin{pmatrix} -D_2\sin(q_2+q_1)-D_1\sin(q_1) & -D_2\sin(q_2+q_1) & 0 \\ D_2\cos(q_2+q_1)+D_1\cos(q_1) & D_2\cos(q_2+q_1) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

```
(%i31) dJscara:trigsimp(trigexpand(determinant(Jscara)))
```

(%o31) $D_1\, D_2 \sin(q_2)$

Singolarità di Velocità

Vediamo che:

$$\det(J)=0 \Longleftrightarrow q_2=\begin{cases} 0 \\ \pi \end{cases}$$

Che corrisponde ad avere il secondo link sovrapposto al primo.

```
(%i32) Jq2scara:subst(q[2]=0,Jscara)
```

(%o32) $\begin{pmatrix} -D_2\sin(q_1)-D_1\sin(q_1) & -D_2\sin(q_1) & 0 \\ D_2\cos(q_1)+D_1\cos(q_1) & D_2\cos(q_1) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

```
(%i33) nullspace(Jq2scara)
```

Proviso: notequal$(-D_2\sin(q_1),0) \wedge$ notequal$(-D_2\sin(q_1),0)$

(%o33) $\mathrm{span}\left(\begin{pmatrix} -D_2\sin(q_1) \\ (D_2+D_1)\sin(q_1) \\ 0 \end{pmatrix}\right)$

Capiamo quindi che per $q_1 \neq \begin{cases} 0 \\ \pi \end{cases}$ il nucleo ha la seguente espressione:

$$\ker(J_{q_2=0})=\mathrm{span}\begin{pmatrix} -D_2 \\ D_2+D_1 \\ 0 \end{pmatrix}$$

Questo significa che le velocità nulle in punta dipendono dai primi due giunti e dalla lunghezza dei loro link.

```
(%i34) factor(columnspace(Jq2scara))
```

Proviso: notequal$((-D_2-D_1)\sin(q_1),0) \wedge$ notequal$((-D_2-D_1)\sin(q_1),0)$

(%o34) $\text{span}\left(\begin{pmatrix}0\\0\\1\end{pmatrix},\begin{pmatrix}-(D_2+D_1)\sin(q_1)\\(D_2+D_1)\cos(q_1)\\0\end{pmatrix}\right)$

Vediamo che per $q_1=0$, l'immagine diventa:

$$\text{Im}(J_{q_2=0})=\text{span}\left(\begin{pmatrix}0\\0\\1\end{pmatrix},\begin{pmatrix}0\\D_2+D_1\\0\end{pmatrix}\right)$$

Singolarità di Forza

```
(%i35) Jtscara:-transpose(Jscara)
```

(%o35) $\begin{pmatrix}D_2\sin(q_2+q_1)+D_1\sin(q_1) & -D_2\cos(q_2+q_1)-D_1\cos(q_1) & 0\\ D_2\sin(q_2+q_1) & -D_2\cos(q_2+q_1) & 0\\ 0 & 0 & -1\end{pmatrix}$

```
(%i36) dJtscara:trigreduce(expand(determinant(Jtscara)))
```

(%o36) $-D_1\,D_2\sin(q_2)$

Vediamo che il determinante si annulla per

$$\det(J)=0\Longleftrightarrow q_2=\begin{cases}0\\ \pi\end{cases}$$

```
(%i37) Jtq2scara:factor(subst(q[2]=0,Jtscara))
```

(%o37) $\begin{pmatrix}(D_2+D_1)\sin(q_1) & -(D_2+D_1)\cos(q_1) & 0\\ D_2\sin(q_1) & -D_2\cos(q_1) & 0\\ 0 & 0 & -1\end{pmatrix}$

```
(%i38) nullspace(Jtq2scara)
```

Proviso: notequal$((D_2+D_1)\sin(q_1),0)\wedge$notequal$((-D_2-D_1)\sin(q_1),0)$

(%o38) $\text{span}\left(\begin{pmatrix}(-D_2-D_1)\cos(q_1)\\(-D_2-D_1)\sin(q_1)\\0\end{pmatrix}\right)$

Quindi per $q_1=0$ otteniamo che

$$\ker(J_{q_1=0})=\text{span}\begin{pmatrix}-(D_2+D_1)\\0\\0\end{pmatrix}$$

```
(%i39) columnspace(Jtq2scara)
```

Proviso: notequal$(D_2\sin(q_1),0)\wedge$notequal$(-D_2\sin(q_1),0)$

(%o39) $\text{span}\left(\begin{pmatrix}0\\0\\-1\end{pmatrix},\begin{pmatrix}(D_2+D_1)\sin(q_1)\\D_2\sin(q_1)\\0\end{pmatrix}\right)$

---

5. Robot Sferico di Primo Tipo

```
(%i40) psferico:matrix([q[3]*cos(q[1])*sin(q[2])+L[2]*cos(q[1])*cos(q[2])],
                       [q[3]*sin(q[1])*sin(q[2])+L[2]*sin(q[1])*cos(q[2])],
                       [L[2]*sin(q[2])-q[3]*cos(q[2])+L[1]])
```

(%o40) $\begin{pmatrix}q_3\cos(q_1)\sin(q_2)+L_2\cos(q_1)\cos(q_2)\\ q_3\sin(q_1)\sin(q_2)+L_2\sin(q_1)\cos(q_2)\\ L_2\sin(q_2)-q_3\cos(q_2)+L_1\end{pmatrix}$

```
(%i41) Jsferico:J(psferico)$
```

(%i42) ridef(Jsferico)

(%o42) $$\begin{pmatrix} -s_1\,s_2\,q_3 - s_1\,L_2\,c_2 & c_1\,c_2\,q_3 - c_1\,L_2\,s_2 & c_1\,s_2 \\ c_1\,s_2\,q_3 + c_1\,L_2\,c_2 & s_1\,c_2\,q_3 - s_1\,L_2\,s_2 & s_1\,s_2 \\ 0 & s_2\,q_3 + L_2\,c_2 & -c_2 \end{pmatrix}$$

(%i43) dJsferico:factor(trigsimp(trigreduce((determinant(Jsferico)))))

(%o43) $q_3\,(q_3 \sin(q_2) + L_2 \cos(q_2))$

Vediamo che il determinante is annulla per due condizioni:

$$q_3 = 0$$

$$q_3 \sin(q_2) + L_2 \cos(q_2) = 0$$

Consideriamo la prima condizione.

Singolarità di Velocità

(%i44) Jq3sferico:subst(q[3]=0,Jsferico)

(%o44) $$\begin{pmatrix} -L_2 \sin(q_1) \cos(q_2) & -L_2 \cos(q_1) \sin(q_2) & \cos(q_1) \sin(q_2) \\ L_2 \cos(q_1) \cos(q_2) & -L_2 \sin(q_1) \sin(q_2) & \sin(q_1) \sin(q_2) \\ 0 & L_2 \cos(q_2) & -\cos(q_2) \end{pmatrix}$$

(%i45) trigsimp(nullspace(Jq3sferico))

Proviso: notequal($\cos(q_1) \sin(q_2), 0$) $\wedge$ notequal($(L_2 \sin(q_1)^2 + L_2 \cos(q_1)^2) \cos(q_2) \sin(q_2), 0$)

(%o45) $$\mathrm{span}\left(\begin{pmatrix} 0 \\ L_2 \cos(q_2) \sin(q_2) \\ L_2^2 \cos(q_2) \sin(q_2) \end{pmatrix}\right)$$

Vediamo che per $q_2 \neq \{0, \pi, \pm\frac{\pi}{2}\}$

$$\ker(J_{q_3=0}) = \mathrm{span}\begin{pmatrix} 0 \\ 1 \\ L_2 \end{pmatrix}$$

(%i46) columnspace(Jq3sferico)

Proviso: notequal($-L_2 \sin(q_1) \cos(q_2), 0$) $\wedge$ notequal($-L_2^2 \sin(q_1) \cos(q_2)^2, 0$)

(%o46) $$\mathrm{span}\left(\begin{pmatrix} -L_2 \sin(q_1) \cos(q_2) \\ L_2 \cos(q_1) \cos(q_2) \\ 0 \end{pmatrix}, \begin{pmatrix} -L_2 \cos(q_1) \sin(q_2) \\ -L_2 \sin(q_1) \sin(q_2) \\ L_2 \cos(q_2) \end{pmatrix}\right)$$

Consideriamo ora la seconda condizione.

$$q_3 \sin(q_2) + L_2 \cos(q_2) = 0 \longrightarrow s_2 = \frac{-L_2 c_2}{q_3}$$

$$\begin{cases} s_2 = \frac{-L_2 c_2}{q_3} \\ c_2^2 + s_2^2 = 1 \end{cases} \longrightarrow \left(\frac{L_2 c_2}{q_3}\right)^2 + c_2^2 = 1 \longrightarrow c_2 = \pm\frac{q_3}{\sqrt{L_2^2 + q_3^2}} \longrightarrow s_2 = \mp\frac{L_2}{\sqrt{L_2^2 + q_3^2}}$$

$$q_2 = \mathrm{atan2}(\mp L_2, \pm q_3) \qquad \text{con } \sqrt{L_2^2 + q_3^2} > 0$$

(%i47) Jq2sferico:subst(q[3]*sin (q[2])+L[2]*cos (q[2])=0,factor(Jsferico))$

(%i48) ridef(Jq2sferico)

(%o48) $$\begin{pmatrix} 0 & c_1\,c_2\,q_3 - c_1\,L_2\,s_2 & c_1\,s_2 \\ 0 & s_1\,c_2\,q_3 - s_1\,L_2\,s_2 & s_1\,s_2 \\ 0 & 0 & -c_2 \end{pmatrix}$$

(%i49) factor(nullspace(Jq2sferico))

Proviso: notequal($\cos(q_1) \sin(q_2), 0$) $\wedge$ notequal($q_3 \cos(q_1) \cos(q_2)^2 - L_2 \cos(q_1) \cos(q_2) \sin(q_2), 0$)

(%o49) $\operatorname{span}\left(\begin{pmatrix} -\cos(q_1)\cos(q_2)(L_2\sin(q_2)-q_3\cos(q_2)) \\ 0 \\ 0 \end{pmatrix}\right)$

Per $q_1=0,\ q_2=0$, otteniamo che

$$\ker(J_{q_3 s_2+L_2 c_2=0})=\operatorname{span}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

```
(%i50) columnspace(Jq2sferico)
```

Proviso: notequal$(q_3\cos(q_1)\cos(q_2)-L_2\cos(q_1)\sin(q_2),0)\wedge$
notequal$(L_2\cos(q_1)\cos(q_2)\sin(q_2)-q_3\cos(q_1)\cos(q_2)^2,0)$

(%o50) $\operatorname{span}\left(\begin{pmatrix} \cos(q_1)\sin(q_2) \\ \sin(q_1)\sin(q_2) \\ -\cos(q_2) \end{pmatrix},\begin{pmatrix} -\cos(q_1)(L_2\sin(q_2)-q_3\cos(q_2)) \\ -\sin(q_1)(L_2\sin(q_2)-q_3\cos(q_2)) \\ 0 \end{pmatrix}\right)$

Singolarità di Forza

```
(%i51) Jtsferico:-transpose(Jsferico)$
(%i52) factor(ridef(Jtsferico))
```

(%o52) $\begin{pmatrix} s_1(s_2 q_3+L_2 c_2) & -c_1(s_2 q_3+L_2 c_2) & 0 \\ -c_1(c_2 q_3-L_2 s_2) & -s_1(c_2 q_3-L_2 s_2) & -(s_2 q_3+L_2 c_2) \\ -c_1 s_2 & -s_1 s_2 & c_2 \end{pmatrix}$

```
(%i53) dJtsferico:factor(trigsimp(trigreduce((determinant(Jtsferico)))))
```

(%o53) $-q_3(q_3\sin(q_2)+L_2\cos(q_2))$

Valgono le stesse considerazioni fatte per le cinematiche di velocità per l'annullamento del determinante. Pertanto:

```
(%i54) Jtq3sferico:subst(q[3]=0,Jtsferico)$
(%i55) ridef(Jtq3sferico)
```

(%o55) $\begin{pmatrix} s_1 L_2 c_2 & -c_1 L_2 c_2 & 0 \\ c_1 L_2 s_2 & s_1 L_2 s_2 & -L_2 c_2 \\ -c_1 s_2 & -s_1 s_2 & c_2 \end{pmatrix}$

```
(%i56) factor(trigsimp(nullspace(Jtq3sferico)))
```

Proviso: notequal$(L_2\sin(q_1)\cos(q_2),0)\wedge$notequal$(-L_2^2\sin(q_1)\cos(q_2)^2,0)$

(%o56) $\operatorname{span}\left(\begin{pmatrix} -L_2^2\cos(q_1)\cos(q_2)^2 \\ -L_2^2\sin(q_1)\cos(q_2)^2 \\ -L_2^2\cos(q_2)\sin(q_2) \end{pmatrix}\right)$

Possiamo quindi concludere che per $q_2\neq\pm\frac{\pi}{2}$

$$\ker(J^T_{q_3=0})=\operatorname{span}\begin{pmatrix} c_1 c_2 \\ s_1 c_2 \\ s_2 \end{pmatrix}$$

```
(%i57) columnspace(Jtq3sferico)
```

Proviso: notequal$(-\cos(q_1)\sin(q_2),0)\wedge$notequal$((L_2\sin(q_1)^2+L_2\cos(q_1)^2)\cos(q_2)\sin(q_2),0)$

(%o57) $\operatorname{span}\left(\begin{pmatrix} -L_2\cos(q_1)\cos(q_2) \\ L_2\sin(q_1)\sin(q_2) \\ -\sin(q_1)\sin(q_2) \end{pmatrix},\begin{pmatrix} L_2\sin(q_1)\cos(q_2) \\ L_2\cos(q_1)\sin(q_2) \\ -\cos(q_1)\sin(q_2) \end{pmatrix}\right)$

Consideriamo la seconda condizione.

```
(%i58) Jtq2sferico:subst(q[3]*sin (q[2])+L[2]*cos (q[2])=0,factor(Jtsferico))$
```

```
(%i59) factor(ridef(Jtq2sferico))
```

$$(\%o59)\quad \begin{pmatrix} 0 & 0 & 0 \\ -c_1\,(c_2\,q_3 - L_2\,s_2) & -s_1\,(c_2\,q_3 - L_2\,s_2) & 0 \\ -c_1\,s_2 & -s_1\,s_2 & c_2 \end{pmatrix}$$

```
(%i60) factor(nullspace(Jtq2sferico))
```

Proviso: notequal($L_2\cos(q_1)\sin(q_2) - q_3\cos(q_1)\cos(q_2), 0$) $\wedge$ notequal($L_2\cos(q_1)\cos(q_2)\sin(q_2) - q_3\cos(q_1)\cos(q_2)^2, 0$)

$$(\%o60)\quad \mathrm{span}\left(\begin{pmatrix} -\sin(q_1)\cos(q_2)\,(L_2\sin(q_2) - q_3\cos(q_2)) \\ \cos(q_1)\cos(q_2)\,(L_2\sin(q_2) - q_3\cos(q_2)) \\ 0 \end{pmatrix}\right)$$

Vediamo che per $q_2 = 0$ si ha

$$\ker(J_{q_3 s_2 + L_2 c_2 = 0}) = \mathrm{span}\begin{pmatrix} s_1 \\ -c_1 \\ 0 \end{pmatrix}$$

```
(%i61) columnspace(Jtq2sferico)
```

Proviso: notequal($-\cos(q_1)\sin(q_2), 0$) $\wedge$ notequal($q_3\cos(q_1)\cos(q_2)^2 - L_2\cos(q_1)\cos(q_2)\sin(q_2), 0$)

$$(\%o61)\quad \mathrm{span}\left(\begin{pmatrix} 0 \\ 0 \\ \cos(q_2) \end{pmatrix}, \begin{pmatrix} 0 \\ \cos(q_1)\,(L_2\sin(q_2) - q_3\cos(q_2)) \\ -\cos(q_1)\sin(q_2) \end{pmatrix}\right)$$

---

6. Robot Sferico di Tipo 2 (Stanford)

```
(%i62) p:matrix([q[3]*cos(q[1])*sin(q[2])-L[2]*sin(q[1])],
                [q[3]*sin(q[1])*sin(q[2])+L[2]*cos(q[1])],
                [q[3]]*cos(q[2])+L[1])
```

$$(\%o62)\quad \begin{pmatrix} q_3\cos(q_1)\sin(q_2) - L_2\sin(q_1) \\ q_3\sin(q_1)\sin(q_2) + L_2\cos(q_1) \\ q_3\cos(q_2) + L_1 \end{pmatrix}$$

```
(%i63) Jstanford:J(p)
```

$$(\%o63)\quad \begin{pmatrix} -q_3\sin(q_1)\sin(q_2) - L_2\cos(q_1) & q_3\cos(q_1)\cos(q_2) & \cos(q_1)\sin(q_2) \\ q_3\cos(q_1)\sin(q_2) - L_2\sin(q_1) & q_3\sin(q_1)\cos(q_2) & \sin(q_1)\sin(q_2) \\ 0 & -q_3\sin(q_2) & \cos(q_2) \end{pmatrix}$$

```
(%i64) dJstanford:trigsimp(determinant(Jstanford))
```

$$(\%o64)\quad -q_3^2\sin(q_2)$$

Vediamo che il determinante si annulla per

$$q_3 = 0,\; q_2 = \begin{cases} 0 \\ \pi \end{cases}$$

```
(%i65) Jq3stanford:subst(q[3]=0,Jstanford)
```

$$(\%o65)\quad \begin{pmatrix} -L_2\cos(q_1) & 0 & \cos(q_1)\sin(q_2) \\ -L_2\sin(q_1) & 0 & \sin(q_1)\sin(q_2) \\ 0 & 0 & \cos(q_2) \end{pmatrix}$$

Singolarità di Velocità

```
(%i66) nullspace(Jq3stanford)
```

Proviso: notequal$(-L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1)\cos(q_2),0)$

(%o66) $\mathrm{span}\left(\begin{pmatrix} 0 \\ -L_2\cos(q_1)\cos(q_2) \\ 0 \end{pmatrix}\right)$

Si vede quindi che

$$\ker(J_{q_3=0}) = \mathrm{span}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

(%i67) `columnspace(Jq3stanford)`

Proviso: notequal$(-L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1)\cos(q_2),0)$

(%o67) $\mathrm{span}\left(\begin{pmatrix} -L_2\cos(q_1) \\ -L_2\sin(q_1) \\ 0 \end{pmatrix}, \begin{pmatrix} \cos(q_1)\sin(q_2) \\ \sin(q_1)\sin(q_2) \\ \cos(q_2) \end{pmatrix}\right)$

(%i68) `Jq2stanford:subst(q[2]=0,Jstanford)`

(%o68) $\begin{pmatrix} -L_2\cos(q_1) & q_3\cos(q_1) & 0 \\ -L_2\sin(q_1) & q_3\sin(q_1) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

(%i69) `nullspace(Jq2stanford)`

Proviso: notequal$(-L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1),0)$

(%o69) $\mathrm{span}\left(\begin{pmatrix} -q_3\cos(q_1) \\ -L_2\cos(q_1) \\ 0 \end{pmatrix}\right)$

(%i70) `columnspace(Jq2stanford)`

Proviso: notequal$(-L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1),0)$

(%o70) $\mathrm{span}\left(\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -L_2\cos(q_1) \\ -L_2\sin(q_1) \\ 0 \end{pmatrix}\right)$

Singolarità di Forza

(%i71) `Jtstanford:-transpose(Jstanford)`

(%o71) $\begin{pmatrix} q_3\sin(q_1)\sin(q_2)+L_2\cos(q_1) & L_2\sin(q_1)-q_3\cos(q_1)\sin(q_2) & 0 \\ -q_3\cos(q_1)\cos(q_2) & -q_3\sin(q_1)\cos(q_2) & q_3\sin(q_2) \\ -\cos(q_1)\sin(q_2) & -\sin(q_1)\sin(q_2) & -\cos(q_2) \end{pmatrix}$

(%i72) `dJstanford:trigsimp(determinant(Jtstanford))`

(%o72) $q_3^2\sin(q_2)$

Valgono le stesse considerazioni sul determinante. Quindi:

(%i73) `Jtq3stanford:subst(q[3]=0,Jtstanford)`

(%o73) $\begin{pmatrix} L_2\cos(q_1) & L_2\sin(q_1) & 0 \\ 0 & 0 & 0 \\ -\cos(q_1)\sin(q_2) & -\sin(q_1)\sin(q_2) & -\cos(q_2) \end{pmatrix}$

(%i74) `nullspace(Jtq3stanford)`

Proviso: notequal$(L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1)\cos(q_2),0)$

(%o74) $\mathrm{span}\left(\begin{pmatrix} L_2\sin(q_1)\cos(q_2) \\ -L_2\cos(q_1)\cos(q_2) \\ 0 \end{pmatrix}\right)$

```
(%i75) columnspace(Jtq3stanford)
```

Proviso: notequal$(L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1)\cos(q_2),0)$

(%o75) $$\operatorname{span}\left(\begin{pmatrix}0\\0\\-\cos(q_2)\end{pmatrix},\begin{pmatrix}L_2\cos(q_1)\\0\\-\cos(q_1)\sin(q_2)\end{pmatrix}\right)$$

```
(%i76) Jtq2stanford:subst(q[2]=0,Jtstanford)
```

(%o76) $$\begin{pmatrix}L_2\cos(q_1) & L_2\sin(q_1) & 0\\ -q_3\cos(q_1) & -q_3\sin(q_1) & 0\\ 0 & 0 & -1\end{pmatrix}$$

```
(%i77) nullspace(Jtq2stanford)
```

Proviso: notequal$(L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1),0)$

(%o77) $$\operatorname{span}\left(\begin{pmatrix}L_2\sin(q_1)\\-L_2\cos(q_1)\\0\end{pmatrix}\right)$$

```
(%i78) columnspace(Jtq2stanford)
```

Proviso: notequal$(L_2\cos(q_1),0) \wedge$ notequal$(-L_2\cos(q_1),0)$

(%o78) $$\operatorname{span}\left(\begin{pmatrix}0\\0\\-1\end{pmatrix},\begin{pmatrix}L_2\cos(q_1)\\-q_3\cos(q_1)\\0\end{pmatrix}\right)$$

---

7. Robot Antropomorfo

```
(%i79) pantro:matrix([D[3]*cos(q[1])*cos(q[2]+q[3])+D[2]*cos(q[1])*cos(q[2])],
                     [D[3]*sin(q[1])*cos(q[2]+q[3])+D[2]*sin(q[1])*cos(q[2])],
                     [D[3]*sin(q[2]+q[3])+D[2]*sin(q[2])+L[1]])
```

(%o79) $$\begin{pmatrix}D_3\cos(q_1)\cos(q_3+q_2)+D_2\cos(q_1)\cos(q_2)\\ D_3\sin(q_1)\cos(q_3+q_2)+D_2\sin(q_1)\cos(q_2)\\ D_3\sin(q_3+q_2)+D_2\sin(q_2)+L_1\end{pmatrix}$$

```
(%i80) Jantro:J(pantro)$
(%i81) factor(ridef(Jantro))
```

(%o81) $$\begin{pmatrix}-s_1(D_3c_{23}+D_2c_2) & -c_1(D_3s_{23}+D_2s_2) & -c_1D_3s_{23}\\ c_1(D_3c_{23}+D_2c_2) & -s_1(D_3s_{23}+D_2s_2) & -s_1D_3s_{23}\\ 0 & D_3c_{23}+D_2c_2 & D_3c_{23}\end{pmatrix}$$

```
(%i82) dJantro:factor(trigsimp(factor(determinant(Jantro))))
```

(%o82) $$-D_2D_3(D_3\cos(q_3+q_2)+D_2\cos(q_2))(\cos(q_2)\sin(q_3+q_2)-\sin(q_2)\cos(q_3+q_2))$$

```
(%i83) factor(ridef(dJantro))
```

(%o83) $$-D_2D_3(D_3c_{23}+D_2c_2)(c_2s_{23}-s_2c_{23})$$

vediamo che il determinante si annulla se solo se una tra le seguenti due equazioni è zero:

$$D_3c_{23}+D_2c_2=0$$

$$c_2s_{23}-s_2c_{23}=0$$

Si può vedere che la prima equazione si azzera per

$$\begin{cases}q_3=0\\ q_2=\pm\frac{\pi}{2}\end{cases}\qquad \text{se } D_2\neq D_3$$

$$q_2=q_3=\pi \qquad \text{se } D_2=D_3$$

La seconda equazione invece si azzera per

$$q_3 = \begin{cases} 0 \\ \pi \end{cases}$$

Singolarità di Velocità
Prima Condizione:

$$\begin{cases} q_3 = 0 \\ q_2 = \pm\frac{\pi}{2} \end{cases} \quad \text{se } D_2 \neq D_3$$

```
(%i84) Jq3q2antro:subst([q[3]=0,q[2]=%pi/2],Jantro)$
(%i85) factor(ridef(Jq3q2antro))
```

$$\text{(\%o85)} \begin{pmatrix} 0 & -c_1(D_3+D_2) & -c_1 D_3 \\ 0 & -s_1(D_3+D_2) & -s_1 D_3 \\ 0 & 0 & 0 \end{pmatrix}$$

```
(%i86) trigsimp(nullspace(Jq3q2antro))
```

Proviso: notequal$(-D_3\cos(q_1), 0)$

$$\text{(\%o86)} \ \operatorname{span}\left(\begin{pmatrix} 0 \\ -D_3\cos(q_1) \\ (D_3+D_2)\cos(q_1) \end{pmatrix}, \begin{pmatrix} -D_3\cos(q_1) \\ 0 \\ 0 \end{pmatrix}\right)$$

```
(%i87) factor(columnspace(Jq3q2antro))
```

Proviso: notequal$((-D_3-D_2)\cos(q_1), 0)$

$$\text{(\%o87)} \ \operatorname{span}\left(\begin{pmatrix} -(D_3+D_2)\cos(q_1) \\ -(D_3+D_2)\sin(q_1) \\ 0 \end{pmatrix}\right)$$

Seconda Condizione:

$$q_2 = q_3 = \pi \quad \text{se } D_2 = D_3$$

```
(%i88) Jq3q2antro1:factor(subst([q[3]=%pi,q[2]=%pi,D[2]=D[3]],Jantro))
```

$$\text{(\%o88)} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & D_3 \end{pmatrix}$$

```
(%i89) nullspace(Jq3q2antro1)
```

Proviso: notequal$(D_3, 0)$

$$\text{(\%o89)} \ \operatorname{span}\left(\begin{pmatrix} 0 \\ D_3 \\ 0 \end{pmatrix}, \begin{pmatrix} D_3 \\ 0 \\ 0 \end{pmatrix}\right)$$

```
(%i90) columnspace(Jq3q2antro1)
```

Proviso: notequal$(D_3, 0)$

$$\text{(\%o90)} \ \operatorname{span}\left(\begin{pmatrix} 0 \\ 0 \\ D_3 \end{pmatrix}\right)$$

Terza Condizione

$$q_3 = \begin{cases} 0 \\ \pi \end{cases}$$

```
(%i91) Jq3antro:subst(q[3]=0,Jantro)$
```

(%i92) `factor(ridef(Jq3antro))`

$$(\%o92)\quad \begin{pmatrix} -s_1\,c_2\,(D_3+D_2) & -c_1\,s_2\,(D_3+D_2) & -c_1\,s_2\,D_3 \\ c_1\,c_2\,(D_3+D_2) & -s_1\,s_2\,(D_3+D_2) & -s_1\,s_2\,D_3 \\ 0 & c_2\,(D_3+D_2) & c_2\,D_3 \end{pmatrix}$$

(%i93) `factor(trigsimp(factor(nullspace(Jq3antro))))`

Proviso: notequal$(-D_3\cos(q_1)\sin(q_2), 0) \wedge$ notequal$(((-D_3^2 - D_2\,D_3)\sin(q_1)^2 + (-D_3^2 - D_2\,D_3)\cos(q_1)^2)\cos(q_2)\sin(q_2), 0)$

$$(\%o93)\quad \text{span}\left(\begin{pmatrix} 0 \\ -D_3\,(D_3+D_2)\cos(q_2)\sin(q_2) \\ (D_3^2+2\,D_2\,D_3+D_2^2)\cos(q_2)\sin(q_2) \end{pmatrix}\right)$$

(%i94) `factor(columnspace(Jq3antro))`

Proviso: notequal$((-D_3 - D_2)\sin(q_1)\cos(q_2), 0) \wedge$ notequal$((-D_3^2 - 2\,D_2\,D_3 - D_2^2)\sin(q_1)\cos(q_2)^2, 0)$

$$(\%o94)\quad \text{span}\left(\begin{pmatrix} -(D_3+D_2)\sin(q_1)\cos(q_2) \\ (D_3+D_2)\cos(q_1)\cos(q_2) \\ 0 \end{pmatrix}, \begin{pmatrix} -(D_3+D_2)\cos(q_1)\sin(q_2) \\ -(D_3+D_2)\sin(q_1)\sin(q_2) \\ (D_3+D_2)\cos(q_2) \end{pmatrix}\right)$$

Singolarità di Forza

(%i95) `Jtantro:-transpose(Jantro)$`

(%i96) `factor(ridef(Jtantro))`

$$(\%o96)\quad \begin{pmatrix} s_1\,(D_3\,c_{23}+D_2\,c_2) & -c_1\,(D_3\,c_{23}+D_2\,c_2) & 0 \\ c_1\,(D_3\,s_{23}+D_2\,s_2) & s_1\,(D_3\,s_{23}+D_2\,s_2) & -(D_3\,c_{23}+D_2\,c_2) \\ c_1\,D_3\,s_{23} & s_1\,D_3\,s_{23} & -D_3\,c_{23} \end{pmatrix}$$

(%i97) `dJtantro:factor(trigsimp(factor(determinant(Jtantro))))$`

(%i98) `factor(ridef(dJtantro))`

$$(\%o98)\quad D_2\,D_3\,(D_3\,c_{23}+D_2\,c_2)\,(c_2\,s_{23}-s_2\,c_{23})$$

Valgono le stesse considerazioni della cinematica di velocità, quindi:
Prima Condizione:

$$\begin{cases} q_3 = 0 \\ q_2 = \pm\frac{\pi}{2} \end{cases} \qquad \text{se } D_2 \neq D_3$$

(%i99) `Jtq3q2antro:subst([q[3]=0,q[2]=%pi/2],Jtantro)$`

(%i100) `factor(ridef(Jtq3q2antro))`

$$(\%o100)\quad \begin{pmatrix} 0 & 0 & 0 \\ c_1\,(D_3+D_2) & s_1\,(D_3+D_2) & 0 \\ c_1\,D_3 & s_1\,D_3 & 0 \end{pmatrix}$$

(%i101) `trigsimp(nullspace(Jtq3q2antro))`

Proviso: notequal$((D_3+D_2)\cos(q_1), 0)$

$$(\%o101)\quad \text{span}\left(\begin{pmatrix} 0 \\ 0 \\ (-D_3-D_2)\sin(q_1) \end{pmatrix}, \begin{pmatrix} (-D_3-D_2)\sin(q_1) \\ (D_3+D_2)\cos(q_1) \\ 0 \end{pmatrix}\right)$$

(%i102) `factor(columnspace(Jtq3q2antro))`

Proviso: notequal$(D_3\cos(q_1), 0)$

(%o102) $\operatorname{span}\left(\begin{pmatrix} 0 \\ (D_3+D_2)\cos(q_1) \\ D_3\cos(q_1) \end{pmatrix}\right)$

Seconda Condizione:

$$q_2 = q_3 = \pi \qquad \text{se } D_2 = D_3$$

(%i103) `Jtq3q2antro1:factor(subst([q[3]=%pi,q[2]=%pi,D[2]=D[3]],Jtantro))`

(%o103) $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -D_3 \end{pmatrix}$

(%i104) `nullspace(Jtq3q2antro1)`

Proviso: notequal$(-D_3, 0)$

(%o104) $\operatorname{span}\left(\begin{pmatrix} 0 \\ -D_3 \\ 0 \end{pmatrix}, \begin{pmatrix} -D_3 \\ 0 \\ 0 \end{pmatrix}\right)$

(%i105) `columnspace(Jtq3q2antro1)`

Proviso: notequal$(-D_3, 0)$

(%o105) $\operatorname{span}\left(\begin{pmatrix} 0 \\ 0 \\ -D_3 \end{pmatrix}\right)$

Terza Condizione

$$q_3 = \begin{cases} 0 \\ \pi \end{cases}$$

(%i106) `Jtq3antro:subst(q[3]=0,Jtantro)$`

(%i107) `factor(ridef(Jtq3antro))`

(%o107) $\begin{pmatrix} s_1\,c_2\,(D_3+D_2) & -c_1\,c_2\,(D_3+D_2) & 0 \\ c_1\,s_2\,(D_3+D_2) & s_1\,s_2\,(D_3+D_2) & -c_2\,(D_3+D_2) \\ c_1\,s_2\,D_3 & s_1\,s_2\,D_3 & -c_2\,D_3 \end{pmatrix}$

(%i108) `factor(trigsimp(factor(nullspace(Jtq3antro))))`

Proviso: notequal$((D_3+D_2)\sin(q_1)\cos(q_2), 0) \wedge$ notequal$((-D_3^2 - 2\,D_2\,D_3 - D_2^2)\sin(q_1)\cos(q_2)^2,$ $0)$

(%o108) $\operatorname{span}\left(\begin{pmatrix} -(D_3^2+2\,D_2\,D_3+D_2^2)\cos(q_1)\cos(q_2)^2 \\ -(D_3^2+2\,D_2\,D_3+D_2^2)\sin(q_1)\cos(q_2)^2 \\ -(D_3^2+2\,D_2\,D_3+D_2^2)\cos(q_2)\sin(q_2) \end{pmatrix}\right)$

(%i109) `factor(columnspace(Jtq3antro))`

Proviso: notequal$(D_3\cos(q_1)\sin(q_2), 0) \wedge$ notequal$(((-D_3^2 - D_2\,D_3)\sin(q_1)^2 + (-D_3^2 -$ $D_2\,D_3)\cos(q_1)^2)\cos(q_2)\sin(q_2), 0)$

(%o109) $\operatorname{span}\left(\begin{pmatrix} -(D_3+D_2)\cos(q_1)\cos(q_2) \\ (D_3+D_2)\sin(q_1)\sin(q_2) \\ D_3\sin(q_1)\sin(q_2) \end{pmatrix}, \begin{pmatrix} (D_3+D_2)\sin(q_1)\cos(q_2) \\ (D_3+D_2)\cos(q_1)\sin(q_2) \\ D_3\cos(q_1)\sin(q_2) \end{pmatrix}\right)$

Energia Cinetica e Potenziale di Strutture Portanti
Funzioni di utility

1) Determinare le dimensioni di una matrice.

```
(%i1) size(M):=[length(M), length(transpose(M))]$
```

2) Calcolare la matrice antisimmetrica dato un vettore di $\mathbb{R}^3$

```
(%i2) S(v):=block([s1,s2,s3,S],
        s1:matrix([0],[v[3,1]],[-v[2,1]]),
        s2:matrix([-v[3,1]],[0],[v[1,1]]),
        s3:matrix([v[2,1]],[-v[1,1]],[0]),
        S:s1, S:addcol(S,s2), S:addcol(S,s3),
        return(S))$
```

3) Calcolare la matrice di rotazione $R_v(\theta)$ attorno un asse v di un angolo ϑ, in $\mathbb{R}^3$

```
(%i3) rodrigues(v,theta):=block([S,I,S2],
        S:S(v), I:ident(size(v)[1]), S2:S.S,
        rodr:I+S2*(1-cos(theta))+S*sin(theta),
        return(trigsimp(trigexpand(trigreduce(rodr)))))$
```

4) Calcolare la matrice di rotazione $R_x(\theta)$ in $\mathbb{R}^2$

```
(%i4) rot2(theta):=block([rot2],
        rot2:matrix([cos(theta), -sin(theta)],[sin(theta), cos(theta)]),
        return(rot2))$
```

5) Calcolare matrice di rotazione tramite esponenziale di matrice.

```
(%i5) expLaplace(M, theta):=block(
      [dim, I, sAi, sAiT],
            dim:size(M),
            if(dim[1]#dim[2]) then error("Matrix not Square"),
            I:ident(dim[1]),
            sAi:invert(s*I-M),
            sAiT:sAi,
            for i:1 thru dim[1] do(
             for j:1 thru dim[2] do(
              sAiT[i,j]:ilt(sAi[i,j],s,theta)
             )
            ),
            return(expand(trigreduce(trigsimp(sAiT))))
      )$
```

6) Matrice di Avvitamento dati asse v, angolo ϑ e spostamento d: $\mathrm{Av}(v,\theta,d)=\begin{pmatrix} e^{\boldsymbol{S}_v(\theta)} & d\,v \\ 0 & 1 \end{pmatrix}$

```
(%i6) Av(v, theta, d):=block(
      [exp, S, dv, row, T],
            S:S(v),
            exp:expLaplace(S, theta),
            row:matrix([0,0,0,1]),
            dv:d*v,
            T:addcol(exp, dv),
            T:addrow(T,row),
            return(trigsimp(trigrat(trigreduce(trigexpand(T)))))
      )$
```

7) Matrice della trasformazione completa: $Q_{i-1,i}=\mathrm{Av}(z,\theta_i,d_i)\mathrm{Av}(x,\alpha_i,a_i)$

```
(%i7) Q(thetai,di,alphai,ai):=block(
      [Avz, Avx, x, z, Q], z:matrix([0],[0],[1]), x:matrix([1],[0],[0]),
            Avz:Av(z,thetai,di),
            Avx:Av(x,alphai,ai),
            Q:Avz.Avx,
            return(Q)
      )$
```

8) Nelle procedure è stato inserito come argomento la lunghezza dei link dei robot, indicando con L le lunghezze dei link che vengono orientati lungo l'asse Z e con D quelle dei link orientati lungo l'asse X

Determino la posizione del baricentro di un link data la corrispondente riga di Denavit-Hartenberg.

```
(%i8) baricentro(row):=block(
      [bc],
          bc:addrow(addcol(ident(3),transpose(row)),[0,0,0,1]),
          return(bc)
      )$
```

Creo una matrice di inerzia in forma simbolica.

```
(%i9) inerzia(j):=block(
      [II],
      II:matrix([I[xx[i]],I[xy[i]],I[xz[i]]],
                [I[xy[i]],I[yy[i]],I[yz[i]]],
                [I[xz[i]],I[yz[i]],I[zz[i]]]),
      II:subst(j, i, II),
      return(II)
      )$
```

Estendo la funzione diff() per la derivata di una funzione composta (secondo specifiche del manuale)

```
(%i10) derivata(f, n):=block(
       [df],
       df:0,
       for i:1 thru n do(
         df: df+diff(f,q[i])*v[i]
       ),
       return(df)
       )$
```

Definisco procedura per estrarre la forma quadratica di una funzione.

```
(%i11) formaQuad(func, n) := block(
           [B],
           B: zeromatrix(n,n),
           /*Derivo due volte: ottengo i termini quadratici nelle velocità*/
           for k:1 thru n do(
               B[k, k]: diff(func, v[k], 2)),
           /*Derivo i termini misti:*/
           /*ottengo i termini sopra e sotto la diagonale principale.*/
           for i:1 thru n do(
               for j:i+1 thru n do(
                   B[i, j]: 1/2*diff( diff( func, v[j] ), v[i] ),
                   B[j, i]: B[i, j]
               )
           ),
           return (B)
       )$
```

Definisco procedura per il calcolo dell'energia cinetica di un link, data la corrispondente matrice di DH traslata nel baricentro.

```
(%i12) Tlink(Qh, dof, M):=block(
       [Tr, Tt, w, wt, II, R, Rt, Rd, Sw, result],
            R:submatrix(4,Qh,4),
            d:submatrix(4,Qh,1,2,3),
            Rd:derivata(R, dof),
            dd:derivata(d, dof),
            Sw:trigsimp(Rd.transpose(R)),
            w:matrix([Sw[3][2]],[Sw[1][3]],[Sw[2][1]]),
            wt:transpose(w),
            dd:trigsimp(trigreduce(trigexpand(dd))),
            ddt:transpose(dd),
            II:inerzia(dof),
            Tt:1/2*M*trigsimp(trigreduce(trigexpand(ddt.dd))),
            Tr:1/2*trigsimp(trigreduce(trigexpand(wt.R.II.transpose(R).w))),
            result:[Tr, Tt, II],
            return(result)
       )$
```

Definisco procedura per il calcolo dell'energia potenziale di un link, data la corrispondente matrice di DH traslata nel baricentro.

```
(%i13) Ulink(Qh, M):=block(
       [g, U, d],
           /*Approssimo accelerazione gravitazionale a 10 m/s^2*/
            g:[0,0,10],
            d:submatrix(4,Qh,1,2,3),
            U:-M*g.d,
            return(U)
       )$
```

Definisco funzione per la stampa delle quantità calcolate. E' commentata nella prossima funzione.

```
(%i14) stampa(T,U,dof):=block(
       /*T[i]:[EkR, EkT, Mat Inerzia], T[4]:Ek totale*/
       let(cos(q[1]),c[1]),let(cos(q[2]),c[2]),let(cos(q[3]),c[3]),
            let(sin(q[1]),s[1]),let(sin(q[2]),s[2]),let(sin(q[3]),s[3]),
            let(cos(q[1]+q[2]),c[12]),let(sin(q[1]+q[2]),s[12]),
            let(cos(q[3]+q[2]),c[23]),let(sin(q[3]+q[2]),s[23]),
       if(dof = 2) then(
            print("Energia Cinetica di Rotazione Link 1: ", letsimp(T[1][1])),
            print("Energia Cinetica di Traslazione Link 1: ", letsimp(T[1][2])),
            print("Matrice di Inerzia Link 1: ", T[1][3]),
            print("Energia Potenziale Link 1: ", letsimp(U[1])),
            print("Energia Cinetica di Rotazione Link 2: ", letsimp(T[2][1])),
            print("Energia Cinetica di Traslazione Link 2: ", letsimp(T[2][2])),
            print("Matrice di Inerzia Link 2: ", T[2][3]),
            print("Energia Potenziale Link 2: ", letsimp(U[2])),
            print("Energia Cinetica complessiva: ",
                   1/2,matrix([v[1],v[2]]),letsimp(T[4]),
                   transpose(matrix([v[1],v[2]]))),
            print("Energia Potenziale Complessiva: ", letsimp(U[1]+U[2]))
       ),
       if (dof = 3) then(
            print("Energia Cinetica di Rotazione Link 1: ", letsimp(T[1][1])),
            print("Energia Cinetica di Traslazione Link 1: ", letsimp(T[1][2])),
            print("Matrice di Inerzia Link 1: ", T[1][3]),
            print("Energia Potenziale Link 1: ", letsimp(U[1])),
            print("Energia Cinetica di Rotazione Link 2: ", letsimp(T[2][1])),
            print("Energia Cinetica di Traslazione Link 2: ", letsimp(T[2][2])),
            print("Matrice di Inerzia Link 2: ", T[2][3]),
            print("Energia Potenziale Link 2: ", letsimp(U[2])),
            print("Energia Cinetica di Rotazione Link 3: ", letsimp(T[3][1])),
            print("Energia Cinetica di Traslazione Link 3: ", letsimp(T[3][2])),
            print("Energia Potenziale Link 3: ", letsimp(U[3])),
            print("Matrice di Inerzia Link 3: ", T[3][3]),
            print("Energia Cinetica complessiva: ",
                   1/2,matrix([v[1],v[2],v[3]]),letsimp(T[4]),
                   transpose(matrix([v[1],v[2],v[3]]))),
            print("Energia Potenziale Complessiva: ", letsimp(U[1]+U[2]+U[3]))
       )$
```

Calcolo l'energia di un robot, fornendo la Massa dei link in forma di lista ($[M_1, M_2, M_3]$) e la relativa tabella di Denavit-Hartenberg.

```
(%i15) energia(DH, M, Trsz):=block(
       [T, U, E1, E2, E3, Q, i, row, rowList, dof, M1, M2, M3,
        Q01, Q12, Q23, Q03, Q01h, Q12h, Q23h, Q03h, g, U1, U2, U3],
        let(cos(q[1]),c[1]),let(cos(q[2]),c[2]),let(cos(q[3]),c[3]),
            let(sin(q[1]),s[1]),let(sin(q[2]),s[2]),let(sin(q[3]),s[3]),
            let(cos(q[1]+q[2]),c[12]),let(sin(q[1]+q[2]),s[12]),
            let(cos(q[3]+q[2]),c[23]),let(sin(q[3]+q[2]),s[23]),
            dof: size(DH)[1],
            M1:M[1], M2:M[2], M3:M[3],
            rowList:[], Q:[], Qh:[],
       /*Carico la tabella di DH per eseguire in modo atomico i calcoli*/
            for i:1 thru dof do(
              rowList:append(rowList,[DH[i]]),
              row:rowList[i],
              Q:append(Q,[Q(row[1],row[2],row[3],row[4])])),
       /*Controllo quanti gradi di libertà ha il robot e calcolo l'energia*/
            if (dof = 2) then(
            Q01:Q[1], Q12:Q[2], Q01h:Q01.baricentro(Trsz[1]),
            Q02:trigreduce(Q01.Q12), Q02h:Q02.baricentro(Trsz[2]),
            T1: factor(Tlink(Q01h, 1, M1)), U1: Ulink(Q01h,M1),
            T2: factor(Tlink(Q02h, 2, M2)), U2: Ulink(Q02h,M2),
            T:letsimp(formaQuad(T1[1]+T1[2]+T2[1]+T2[2],2)), U:letsimp(U1+U2)
            ),
            if (dof = 3) then(
            Q01:Q[1], Q12:Q[2], Q23:Q[3],
            Q01h:Q01.baricentro(Trsz[1]),
            Q02:trigreduce(Q01.Q12), Q02h:Q02.baricentro(Trsz[2]),
            Q03:trigreduce(Q02.Q23), Q03h:Q03.baricentro(Trsz[3]),
            T1: factor(Tlink(Q01h, 1, M1)), U1: Ulink(Q01h,M1),
            T2: factor(Tlink(Q02h, 2, M2)), U2: Ulink(Q02h,M2),
            T3: factor(Tlink(Q03h, 3, M3)), U3: Ulink(Q03h,M3),
            T:letsimp(formaQuad(T1[1]+T1[2]+T2[1]+T2[2]+T3[1]+T3[2],3)),
            U:letsimp(U1+U2+U3)
           ),
           /*Funzione di Stampa commentata per migliorare la leggibilità*/
           /*stampa([T1,T2,T3,T],[U1,U2,U3],dof),*/
           return([T,U])
       )$
```

Definisco vettore delle masse.

```
(%i16) M:[M[1],M[2],M[3]]$
```

Definisco le tabelle di DH sottoforma di matrici. In ordine troviamo:

1. Due DOF
2. Cartesiano
3. Cilindrico
4. Scara
5. Sferico Tipo 1
6. Sferico Tipo 2 (Stanford)
7. Antropomorfo

```
(%i17) DH:[matrix([q[1],0,0,L[1]],[q[2],0,0,L[2]]),
        matrix([0,q[1],-%pi/2,0],[-%pi/2,q[2],-%pi/2,0],[0,q[3],0,0]),
        matrix([q[1],L[1],0,0],[0,q[2],-%pi/2,0],[0,q[3],0,0]),
        matrix([q[1],L[1],0,D[1]],[q[2],0,0,D[2]],[0,q[3],0,L[3]]),
        matrix([q[1],L[1],%pi/2,0],[q[2],0,%pi/2,D[2]],[0,q[3],0,0]),
        matrix([q[1],L[1],-%pi/2,0],[q[2],L[2],%pi/2,0],[0,q[3],0,0]),
        matrix([q[1],L[1],%pi/2,0],[q[2],0,0,D[2]],[q[3],0,0,D[3]])]
```

$$(\%o17)\quad \left[\begin{pmatrix} q_1 & 0 & 0 & L_1 \\ q_2 & 0 & 0 & L_2 \end{pmatrix}, \begin{pmatrix} 0 & q_1 & -\frac{\pi}{2} & 0 \\ -\frac{\pi}{2} & q_2 & -\frac{\pi}{2} & 0 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & 0 & 0 \\ 0 & q_2 & -\frac{\pi}{2} & 0 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & 0 & D_1 \\ q_2 & 0 & 0 & D_2 \\ 0 & q_3 & 0 & L_3 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & \frac{\pi}{2} & 0 \\ q_2 & 0 & \frac{\pi}{2} & D_2 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & -\frac{\pi}{2} & 0 \\ q_2 & L_2 & \frac{\pi}{2} & 0 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & \frac{\pi}{2} & 0 \\ q_2 & 0 & 0 & D_2 \\ q_3 & 0 & 0 & D_3 \end{pmatrix}\right]$$

Definisco matrici di traslazione. Ogni riga di una matrice rappresenta la traslazione da applicare alla matrice di trasformazione del link i-esimo per raggiungere il suo baricentro.
L'ordine con cui sono scritte combacia con quello delle tabelle di DH scritte sopra per semplicità.

```
(%i18) Trsz:[matrix([-L[1]/2,0,0],[-L[2]/2,0,0]),
          matrix([0,L[1]/2,0],[0,L[2]/2,0],[0,0,-L[3]/2]),
          matrix([0,0,-L[1]/2],[0,L[2]/2,0],[0,0,-L[3]/2]),
          matrix([-D[1]/2,0,-L[1]/2],[-D[2]/2,0,0],[0,0,L[3]/2]),
          matrix([0,-L[1]/2,0],[-D[2]/2,0,0],[0,0,-L[3]/2]),
          matrix([0,L[1]/2,0],[0,-L[2]/2,0],[0,0,-L[3]/2]),
          matrix([0,-L[1]/2,0],[-D[2]/2,0,0],[-D[3]/2,0,0])]
```

$$(\%o18)\quad \left[\begin{pmatrix} -\frac{L_1}{2} & 0 & 0 \\ -\frac{L_2}{2} & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & \frac{L_1}{2} & 0 \\ 0 & \frac{L_2}{2} & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & 0 & -\frac{L_1}{2} \\ 0 & \frac{L_2}{2} & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} -\frac{D_1}{2} & 0 & -\frac{L_1}{2} \\ -\frac{D_2}{2} & 0 & 0 \\ 0 & 0 & \frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & -\frac{L_1}{2} & 0 \\ -\frac{D_2}{2} & 0 & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & \frac{L_1}{2} & 0 \\ 0 & -\frac{L_2}{2} & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & -\frac{L_1}{2} & 0 \\ -\frac{D_2}{2} & 0 & 0 \\ -\frac{D_3}{2} & 0 & 0 \end{pmatrix}\right]$$

Robot Due Dof

```
(%i19) energia(DH[1],M,Trsz[1])
```

$$(\%o19)\quad \left[\begin{pmatrix} I_{zz_2}+\frac{M_2\left(8\,L_1\,L_2\cos\left(q_2\right)+2\,L_2^2+8\,L_1^2\right)}{8}+I_{zz_1}+\frac{L_1^2\,M_1}{4} & \frac{I_{zz_2}+\frac{M_2\left(4\,L_1\,L_2\cos\left(q_2\right)+2\,L_2^2\right)}{8}}{2} \\ \frac{I_{zz_2}+\frac{M_2\left(4\,L_1\,L_2\cos\left(q_2\right)+2\,L_2^2\right)}{8}}{2} & I_{zz_2}+\frac{L_2^2\,M_2}{4} \end{pmatrix}, 0\right]$$

Robot Cartesiano

```
(%i20) energia(DH[2],M,Trsz[2])
```

$$(\%o20)\quad \left[\begin{pmatrix} M_3+M_2+M_1 & 0 & 0 \\ 0 & M_3+M_2 & 0 \\ 0 & 0 & M_3 \end{pmatrix}, -10\,q_1\,M_3-10\,q_1\,M_2-10\,M_1\,q_1+5\,L_1\,M_1\right]$$

Robot Cilindrico

```
(%i21) energia(DH[3],M,Trsz[3])
```

(%o21) $$\left[\left(\begin{matrix} I_{yy_3}+I_{yy_2}+I_{zz_1}+\frac{M_3\,(8\,q_3^2-8\,L_3\,q_3+2\,L_3^2)}{8} & 0 & 0 \\ 0 & M_3+M_2 & 0 \\ 0 & 0 & M_3 \end{matrix}\right), -10\,q_2\,M_3-10\,L_1\,M_3-10\,M_2\,q_2+5\,L_2\,M_2-10\,L_1\,M_2-5\,L_1\,M_1\right]$$

Robot SCARA

```
(%i22) trigreduce(energia(DH[4],M,Trsz[4]))
```

(%o22) $$\left[\left(I_{zz_3}+I_{zz_2}+\frac{4\,D_1\,D_2\,M_2\cos(q_2)+(D_2^2+4\,D_1^2)\,M_2}{4}+(2\,D_1\,L_3+2\,D_1\,D_2)\,M_3\cos(q_2)+I_{zz_1}+(L_3^2+2\,D_2\,L_3+D_2^2+D_1^2)\,M_3+\frac{D_1^2\,M_1}{4}, (4\,I_{zz_3}+4\,I_{zz_2}+((4\,D_1\,L_3+4\,D_1\,D_2)\,M_3+2\,D_1\,D_2\,M_2)\cos(q_2)+(4\,L_3^2+8\,D_2\,L_3+4\,D_2^2)\,M_3+D_2^2\,M_2)/8, 0; (4\,I_{zz_3}+4\,I_{zz_2}+((4\,D_1\,L_3+4\,D_1\,D_2)\,M_3+2\,D_1\,D_2\,M_2)\cos(q_2)+(4\,L_3^2+8\,D_2\,L_3+4\,D_2^2)\,M_3+D_2^2\,M_2)/8, I_{zz_3}+I_{zz_2}+(L_3^2+2\,D_2\,L_3+D_2^2)\,M_3+\frac{D_2^2\,M_2}{4}, 0; 0, 0, M_3\right), -10\,M_3\,q_3-5\,L_3\,M_3-10\,L_1\,M_3-10\,L_1\,M_2-5\,L_1\,M_1\right]$$

Robot Sferico Tipo 1

```
(%i23) trigsimp(factor(energia(DH[5],M,Trsz[5])))
```

(%o23) $$\left[\left(((4\cos(2\,q_2)+4)\,I_{zz_3}-8\sin(2\,q_2)\,I_{xz_3}+(4-4\cos(2\,q_2))\,I_{xx_3}+(4\cos(2\,q_2)+4)\,I_{zz_2}-8\sin(2\,q_2)\,I_{xz_2}+(4-4\cos(2\,q_2))\,I_{xx_2}+D_2\,(8\,M_3\,q_3\sin(2\,q_2)-4\,L_3\,M_3\sin(2\,q_2))+D_2^2\,((4\,M_3+M_2)\cos(2\,q_2)+4\,M_3+M_2)+q_3\,(4\,L_3\,M_3\cos(2\,q_2)-4\,L_3\,M_3)+q_3^2\,(4\,M_3-4\,M_3\cos(2\,q_2))-L_3^2\,M_3\cos(2\,q_2)+8\,I_{yy_1}+L_3^2\,M_3)/8, -\frac{\cos(q_2)\,I_{yz_3}-\sin(q_2)\,I_{xy_3}+\cos(q_2)\,I_{yz_2}-\sin(q_2)\,I_{xy_2}}{2}, 0; -\frac{\cos(q_2)\,I_{yz_3}-\sin(q_2)\,I_{xy_3}+\cos(q_2)\,I_{yz_2}-\sin(q_2)\,I_{xy_2}}{2}, \frac{4\,I_{yy_3}+4\,I_{yy_2}+4\,M_3\,q_3^2-4\,L_3\,M_3\,q_3+D_2^2\,(4\,M_3+M_2)+L_3^2\,M_3}{4}, -\frac{D_2\,M_3}{2}; 0, -\frac{D_2\,M_3}{2}, M_3\right), 10\,c_2\,M_3\,q_3+D_2\,(-10\,s_2\,M_3-5\,M_2\,s_2)-5\,c_2\,L_3\,M_3+L_1\,(-10\,M_3-10\,M_2-5\,M_1)\right]$$

Robot Sferico Tipo 2 (Stanford)

```
(%i24) trigsimp(factor(energia(DH[6],M,Trsz[6])))
```

(%o24) $$\left[\left(((4\cos(2\,q_2)+4)\,I_{zz_3}-8\sin(2\,q_2)\,I_{xz_3}+(4-4\cos(2\,q_2))\,I_{xx_3}+(4\cos(2\,q_2)+4)\,I_{zz_2}-8\sin(2\,q_2)\,I_{xz_2}+(4-4\cos(2\,q_2))\,I_{xx_2}+q_3\,(4\,L_3\,M_3\cos(2\,q_2)-4\,L_3\,M_3)+q_3^2\,(4\,M_3-4\,M_3\cos(2\,q_2))-L_3^2\,M_3\cos(2\,q_2)+8\,I_{yy_1}+L_2^2\,(8\,M_3+2\,M_2)+L_3^2\,M_3)/8, -(-2\cos(q_2)\,I_{yz_3}+2\sin(q_2)\,I_{xy_3}-2\cos(q_2)\,I_{yz_2}+2\sin(q_2)\,I_{xy_2}+L_2\,(2\,M_3\,q_3\cos(q_2)-L_3\,M_3\cos(q_2)))/4, -\frac{L_2\,s_2\,M_3}{2}; -(-2\cos(q_2)\,I_{yz_3}+2\sin(q_2)\,I_{xy_3}-2\cos(q_2)\,I_{yz_2}+2\sin(q_2)\,I_{xy_2}+L_2\,(2\,M_3\,q_3\cos(q_2)-L_3\,M_3\cos(q_2)))/4, \frac{4\,I_{yy_3}+4\,I_{yy_2}+4\,M_3\,q_3^2-4\,L_3\,M_3\,q_3+L_3^2\,M_3}{4}, 0; -\frac{L_2\,s_2\,M_3}{2}, 0, M_3\right), -10\,c_2\,M_3\,q_3+5\,c_2\,L_3\,M_3+L_1\,(-10\,M_3-10\,M_2-5\,M_1)\right]$$

Robot Antropomorfo

```
(%i25) trigsimp(factor(energia(DH[7],M,Trsz[7])))
```

(%o25) $$\left[\left(((4\cos(2\,q_3+2\,q_2)+4)\,I_{yy_3}+8\sin(2\,q_3+2\,q_2)\,I_{xy_3}+(4-4\cos(2\,q_3+2\,q_2))\,I_{xx_3}+D_3^2\,(M_3\cos(2\,q_3+2\,q_2)+M_3)+D_2\,D_3\,(4\,M_3\cos(q_3+2\,q_2)+4\,M_3\cos(q_3))+(4\cos(2\,q_2)+4)\,I_{yy_2}+8\sin(2\,q_2)\,I_{xy_2}+(4-4\cos(2\,q_2))\,I_{xx_2}+D_2^2\,((4\,M_3+M_2)\cos(2\,q_2)+4\,M_3+M_2)+8\,I_{yy_1})/8, \frac{\cos(q_3+q_2)\,I_{yz_3}+\sin(q_3+q_2)\,I_{xz_3}+\cos(q_2)\,I_{yz_2}+\sin(q_2)\,I_{xz_2}}{2}, \frac{\cos(q_3+q_2)\,I_{yz_3}+\sin(q_3+q_2)\,I_{xz_3}}{2}; \frac{\cos(q_3+q_2)\,I_{yz_3}+\sin(q_3+q_2)\,I_{xz_3}+\cos(q_2)\,I_{yz_2}+\sin(q_2)\,I_{xz_2}}{2},\right.$$

$$\frac{4\,I_{zz_3}+4\,D_2\,D_3\,M_3\cos\left(q_3\right)+4\,I_{zz_2}+D_2^2\left(4\,M_3+M_2\right)+D_3^2\,M_3}{4},\frac{4\,I_{zz_3}+2\,D_2\,D_3\,M_3\cos\left(q_3\right)+D_3^2\,M_3}{8};$$

$$\frac{\cos\left(q_3+q_2\right)I_{yz_3}+\sin\left(q_3+q_2\right)I_{xz_3}}{2},\frac{4\,I_{zz_3}+2\,D_2\,D_3\,M_3\cos\left(q_3\right)+D_3^2\,M_3}{8},\frac{4\,I_{zz_3}+D_3^2\,M_3}{4}\Bigg),-5\,D_3\,M_3\,s_{23}+$$

$$D_2\left(-10\,s_2\,M_3-5\,M_2\,s_2\right)+L_1\left(-10\,M_3-10\,M_2-5\,M_1\right)\Big]$$

(%i26)

Matrici di Inerzia di Corpi Solidi

La matrice di inerzia di un solido a tre dimensioni è descritta dal seguente integrale triplo:

$$\mathbb{I}=\rho\int_V S^T(\hat{p})S(\hat{p})\mathrm{dV}$$

Dove $\hat{p}$ è il vettore delle coordinate del baricentro del solido rispetto al sistema di riferimento inerziale, mentre $S(p)$ è la matrice antisimmetrica seguente:

$$S(\hat{p})=\begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix},\ \hat{p}=\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Da cui segue:

$$S^T(\hat{p})S(\hat{p})=\begin{pmatrix} z^2+y^2 & \text{-xy} & \text{-xz} \\ \text{-xy} & x^2+z^2 & \text{-yz} \\ \text{-xz} & \text{-yz} & x^2+y^2 \end{pmatrix}$$

Consideriamo i seguenti corpi solidi di massa M, sia pieni che cavi:

1. Parallelepipedo di lati A, B, C.
2. Cilindro di raggio R ed altezza H.
3. Sfera di Raggio R.

Per la cavità, consideriamo sia il caso in cui sia completa (diremo senza tappi), sia parziale (con tappi).

(%i1) `p:matrix([x],[y],[z])`

(%o1) $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$

1. **Parallelepipedo**.
Lati A, B, C

(%i2) `a:[-A/2,A/2]`

(%o2) $\left[-\frac{A}{2},\frac{A}{2}\right]$

(%i3) `b:[-B/2,B/2]`

(%o3) $\left[-\frac{B}{2},\frac{B}{2}\right]$

(%i4) `c:[-C/2,C/2]`

(%o4) $\left[-\frac{C}{2},\frac{C}{2}\right]$

Densità del parallelepipedo.

(%i5) `rhoP:M/(A*B*C)`

(%o5) $\frac{M}{A\,B\,C}$

(%i6)
```
parallelepipedo(p,a,b,c):=block(
[s, sTs, II],
    s:matrix([0,-p[3][1],p[2][1]],
             [p[3][1],0,-p[1][1]],
             [-p[2][1],p[1][1],0]),
    sTs:transpose(s).s,
    II:factor(expand(integrate(integrate(integrate(
              sTs,x,a[1],a[2]),y,b[1],b[2]),z,c[1],c[2]))),
    return(II)
)$
```

(%i7) `rhoP*parallelepipedo(p,a,b,c)`

(%o7) $$\begin{pmatrix} \frac{(C^2+B^2)\,M}{12} & 0 & 0 \\ 0 & \frac{(C^2+A^2)\,M}{12} & 0 \\ 0 & 0 & \frac{(B^2+A^2)\,M}{12} \end{pmatrix}$$

1.1 **Parallelepipedo cavo con tappi**.
La cavità è un secondo parallelepipedo posizionato al centro del primo, di lati A_{vt}, B_{vt}, C_{vt} dove $A_{\mathrm{vt}} < A$, $B_{\mathrm{vt}} < B$, $C_{\mathrm{vt}} < C$.

```
(%i8) avt:[-(A[vt])/2,(A[vt])/2]
```

(%o8) $$\left[-\frac{A_{\mathrm{vt}}}{2}, \frac{A_{\mathrm{vt}}}{2}\right]$$

```
(%i9) bvt:[-(B[vt])/2,(B[vt])/2]
```

(%o9) $$\left[-\frac{B_{\mathrm{vt}}}{2}, \frac{B_{\mathrm{vt}}}{2}\right]$$

```
(%i10) cvt:[-(C[vt])/2,(C[vt])/2]
```

(%o10) $$\left[-\frac{C_{\mathrm{vt}}}{2}, \frac{C_{\mathrm{vt}}}{2}\right]$$

```
(%i11) rhovt:M/((A*B*C)-(A[vt]*B[vt]*C[vt]))
```

(%o11) $$\frac{M}{A\,B\,C - A_{\mathrm{vt}}\,B_{\mathrm{vt}}\,C_{\mathrm{vt}}}$$

```
(%i12) paraCavoT(p,a,b,c,avt,bvt,cvt):=block(
       [IIcavo, II, cavo, s, sTs],
                s:matrix([0,-p[3][1],p[2][1]],
                         [p[3][1],0,-p[1][1]],
                         [-p[2][1],p[1][1],0]),
                sTs:transpose(s).s,
                II:factor(expand(integrate(integrate(integrate(
                     sTs,x,a[1],a[2]),y,b[1],b[2]),z,c[1],c[2]))),
                cavo:factor(expand(integrate(integrate(integrate(
                     sTs,x,avt[1],avt[2]),y,bvt[1],bvt[2]),z,cvt[1],cvt[2]))),
                 IIcavo:(II-cavo),
                return(IIcavo)

       )$
(%i13) Ic:factor(expand(rhovt*paraCavoT(p,a,b,c,avt,bvt,cvt)))
```

(%o13) $$\left(\frac{M\,(A_{\mathrm{vt}}\,B_{\mathrm{vt}}\,C_{\mathrm{vt}}^3 + A_{\mathrm{vt}}\,B_{\mathrm{vt}}^3\,C_{\mathrm{vt}} - A\,B\,C^3 - A\,B^3\,C)}{12\,(A_{\mathrm{vt}}\,B_{\mathrm{vt}}\,C_{\mathrm{vt}} - A\,B\,C)}, 0, 0; 0, \frac{M\,(A_{\mathrm{vt}}\,B_{\mathrm{vt}}\,C_{\mathrm{vt}}^3 + A_{\mathrm{vt}}^3\,B_{\mathrm{vt}}\,C_{\mathrm{vt}} - A\,B\,C^3 - A^3\,B\,C)}{12\,(A_{\mathrm{vt}}\,B_{\mathrm{vt}}\,C_{\mathrm{vt}} - A\,B\,C)}, 0; 0, 0, \frac{M\,(A_{\mathrm{vt}}\,B_{\mathrm{vt}}^3\,C_{\mathrm{vt}} + A_{\mathrm{vt}}^3\,B_{\mathrm{vt}}\,C_{\mathrm{vt}} - A\,B^3\,C - A^3\,B\,C)}{12\,(A_{\mathrm{vt}}\,B_{\mathrm{vt}}\,C_{\mathrm{vt}} - A\,B\,C)}\right)$$

1.2 **Parallelepipedo cavo senza tappi**
La cavità è un secondo parallelepipedo posizionato al centro del primo, di lati A_v, B_v, C_v dove $A_v = A$, $B_v < B$, $C_v < C$.

```
(%i14) av:[-A/2,A/2]
```

(%o14) $$\left[-\frac{A}{2}, \frac{A}{2}\right]$$

```
(%i15) bv:[-B[v]/2,B[v]/2]
```

(%o15) $$\left[-\frac{B_v}{2}, \frac{B_v}{2}\right]$$

(%i16) `cv:[-C[v]/2,C[v]/2]`

(%o16) $$\left[-\frac{C_v}{2},\frac{C_v}{2}\right]$$

(%i17) `rhov:M/(A*B*C-A*B[v]*C[v])`

(%o17) $$\frac{M}{A\,B\,C-A\,B_v\,C_v}$$

(%i18)
```
paraCavo(p,a,b,c,bv,cv):=block(
[IIcavoT, II, cavoT, s, sTs],
         s:matrix([0,-p[3][1],p[2][1]],
                  [p[3][1],0,-p[1][1]],
                  [-p[2][1],p[1][1],0]),
         sTs:transpose(s).s,
         II:factor(expand(integrate(integrate(integrate(
              sTs,x,a[1],a[2]),y,b[1],b[2]),z,c[1],c[2]))),
         cavoT:factor(expand(integrate(integrate(integrate(
              sTs,x,a[1],a[2]),y,bv[1],bv[2]),z,cv[1],cv[2]))),
       IIcavoT:(II-cavoT),
       return(IIcavoT)
)$
```

(%i19) `Ic:factor(expand(rhov*paraCavo(p,a,b,c,bv,cv)))`

(%o19) $$\begin{pmatrix}\frac{M\,(B_v\,C_v^3+B_v^3\,C_v-B\,C^3-B^3\,C)}{12\,(B_v\,C_v-B\,C)} & 0 & 0\\ 0 & \frac{M\,(B_v\,C_v^3+A^2\,B_v\,C_v-B\,C^3-A^2\,B\,C)}{12\,(B_v\,C_v-B\,C)} & 0\\ 0 & 0 & \frac{M\,(B_v^3\,C_v+A^2\,B_v\,C_v-B^3\,C-A^2\,B\,C)}{12\,(B_v\,C_v-B\,C)}\end{pmatrix}$$

2. **Cilindro**

Di raggio R ed altezza H

Eseguo cambiamento di coordinate da cartesiane a cilindriche.

$$\begin{cases} x=\rho\cos(\theta) \\ y=\rho\sin(\theta) \\ z=h \end{cases},\quad \begin{array}{l}\rho\in[0,R]\\ \theta\in[0,2\pi]\\ h\in[0,H]\end{array}$$

(%i20) `assume(R>0)$`

(%i21) `rhoC:[0,R]$`

(%i22) `hC:[0,H]$`

(%i23) `dens:M/(%pi*R^2*H)`

(%o23) $$\frac{M}{\pi\,H\,R^2}$$

(%i24)
```
cilindro(rhoC,hC):=block(
[I, s, sTs, p, x, y, z, J, thetaC],
    thetaC:[0, 2*%pi],
    x:rho*cos(theta),
    y:rho*sin(theta),
    z:h,
    p:matrix([x],[y],[z]),
    s:matrix([0,-p[3][1],p[2][1]],
             [p[3][1],0,-p[1][1]],
             [-p[2][1],p[1][1],0]),
    sTs:transpose(s).s,
    J:abs(trigsimp(determinant(
      addcol(diff(p,rho),diff(p,theta),diff(p,h))))),
    I:integrate(integrate(integrate(
      J*sTs,theta,thetaC[1],thetaC[2]),rho,rhoC[1],rhoC[2]),h,hC[1],hC[2]),
    return(I)
)$
```

(%i25) `factor(dens*cilindro(rhoC,hC))`

$$\text{(\%o25)}\quad \begin{pmatrix} \frac{M\,(3\,R^2+4\,H^2)}{12} & 0 & 0 \\ 0 & \frac{M\,(3\,R^2+4\,H^2)}{12} & 0 \\ 0 & 0 & \frac{M\,R^2}{2} \end{pmatrix}$$

2.1 **Cilindro Cavo con Tappi**

La cavità è un secondo cilindro posizionato al centro del primo, di raggio R_{vt} ed altezza H_{vt}. dove $R_{\mathrm{vt}} < R$, $H_{\mathrm{vt}} < H$.

Per eseguire il calcolo dell'integrale usiamo sempre lo stesso cambiamento di coordinate.

```
(%i26) assume(R[vt]>0)$
(%i27) rhoCvt:[0, R[vt]]$
(%i28) hCvt:[0, H[vt]]$
(%i29) densvt:M/((%pi*R^2*H)-(%pi*R[vt]^2*H[vt]))
```

$$\text{(\%o29)}\quad \frac{M}{\pi\,H\,R^2-\pi\,H_{\mathrm{vt}}\,R_{\mathrm{vt}}^2}$$

```
(%i30) cilindroVT(rhoC, hC, rhoCvt, hCvt):=block(
       [I, cavo, Icavo, s, sTs, p, x, y, z, J, thetaC],
           thetaC:[0, 2*%pi],
           x:rho*cos(theta),
           y:rho*sin(theta),
           z:h,
           p:matrix([x],[y],[z]),
           s:matrix([0,-p[3][1],p[2][1]],
                    [p[3][1],0,-p[1][1]],
                    [-p[2][1],p[1][1],0]),
           sTs:transpose(s).s,
           J:abs(trigsimp(determinant(
             addcol(diff(p,rho),diff(p,theta),diff(p,h))))),
           I:integrate(integrate(integrate(
             J*sTs,theta,thetaC[1],thetaC[2]),rho,rhoC[1],rhoC[2]),h,hC[1],hC[2]),
           cavo:integrate(integrate(integrate(
                J*sTs,theta,thetaC[1],thetaC[2]),
                rho,rhoCvt[1],rhoCvt[2]),h,hCvt[1],hCvt[2]),
           Icavo:I-cavo,
           return(Icavo)
       )$
(%i31) factor(densvt*cilindroVT(rhoC, hC, rhoCvt, hCvt))
```

$$\text{(\%o31)}\quad \left(\frac{M\,(3\,H_{\mathrm{vt}}\,R_{\mathrm{vt}}^4+4\,H_{\mathrm{vt}}^3\,R_{\mathrm{vt}}^2-3\,H\,R^4-4\,H^3\,R^2)}{12\,(H_{\mathrm{vt}}\,R_{\mathrm{vt}}^2-H\,R^2)},\,0,\,0;\,0,\,\frac{M\,(3\,H_{\mathrm{vt}}\,R_{\mathrm{vt}}^4+4\,H_{\mathrm{vt}}^3\,R_{\mathrm{vt}}^2-3\,H\,R^4-4\,H^3\,R^2)}{12\,(H_{\mathrm{vt}}\,R_{\mathrm{vt}}^2-H\,R^2)},\,0;\,0,\,0,\,\frac{M\,(H_{\mathrm{vt}}\,R_{\mathrm{vt}}^4-H\,R^4)}{2\,(H_{\mathrm{vt}}\,R_{\mathrm{vt}}^2-H\,R^2)}\right)$$

2.2 **Cilindro Cavo senza Tappi**

La cavità è un secondo cilindro posizionato al centro del primo, di raggio R_v ed altezza H_v. dove $R_v < R$, $H_v = H$.

Per eseguire il calcolo dell'integrale usiamo sempre lo stesso cambiamento di coordinate.

```
(%i32) assume(R[v]>0)$
(%i33) rhoCv:[0, R[v]]$
(%i34) hCv:[0, H]$
```

```
(%i35) densv:M/((%pi*R^2*H)-(%pi*R[v]^2*H))
```

(%o35) $$\frac{M}{\pi\, H\, R^2 - \pi\, H\, R_v^2}$$

```
(%i36) cilindroV(rhoC, hC, rhoCv):=block(
       [I, cavo, Icavo, s, sTs, p, x, y, z, J, thetaC],
           thetaC:[0, 2*%pi],
           x:rho*cos(theta),
           y:rho*sin(theta),
           z:h,
           p:matrix([x],[y],[z]),
           s:matrix([0,-p[3][1],p[2][1]],
                    [p[3][1],0,-p[1][1]],
                    [-p[2][1],p[1][1],0]),
           sTs:transpose(s).s,
           J:abs(trigsimp(determinant(
             addcol(diff(p,rho),diff(p,theta),diff(p,h))))),
           I:integrate(integrate(integrate(
             J*sTs,theta,thetaC[1],thetaC[2]),rho,rhoC[1],rhoC[2]),h,hC[1],hC[2]),
           cavo:integrate(integrate(integrate(
                J*sTs,theta,thetaC[1],thetaC[2]),
                rho,rhoCv[1],rhoCv[2]),h,hC[1],hC[2]),
           Icavo:I-cavo,
           return(Icavo)
       )$
(%i37) factor(densv*cilindroV(rhoC, hC, rhoCv))
```

(%o37) $$\begin{pmatrix} \frac{M\,(3\,R_v^2+3\,R^2+4\,H^2)}{12} & 0 & 0 \\ 0 & \frac{M\,(3\,R_v^2+3\,R^2+4\,H^2)}{12} & 0 \\ 0 & 0 & \frac{M\,(R_v^2+R^2)}{2} \end{pmatrix}$$

3. **Sfera di raggio R**
Per il calcolo dell'integrale usiamo le coordinate sferiche

$$\begin{cases} x\colon \rho\cos(\theta)\sin(\phi) \\ y\colon \rho\sin(\theta)\sin(\phi) \\ z\colon \rho\cos(\phi) \end{cases} , \quad \begin{matrix} \rho\in[0,R] \\ \theta\in[0,2\pi] \\ \phi\in[0,\pi] \end{matrix}$$

(%i41) rhoS:[0,R]$

(%i38) densS:M/(4/3*%pi*R^3)

(%o38) $\dfrac{3\,M}{4\,\pi\,R^3}$

```
(%i56) sfera(rhoS):=block(
       [I, s, sTs, p, x, y, z, J, thetaS, phiS],
           thetaS:[0, 2*%pi], phiS:[0,%pi],
           x:rho*cos(theta)*sin(phi),
           y:rho*sin(theta)*sin(phi),
           z:rho*cos(phi),
           p:matrix([x],[y],[z]),
           s:matrix([0,-p[3][1],p[2][1]],
                    [p[3][1],0,-p[1][1]],
                    [-p[2][1],p[1][1],0]),
           sTs:trigsimp(transpose(s).s),
           J:abs(trigsimp(determinant(
             addcol(diff(p,rho),diff(p,theta),diff(p,phi))))),
           assume(sin(phi)>0),
           I:integrate(integrate(integrate(
             J*sTs,theta,thetaS[1],thetaS[2]),
             rho,rhoS[1],rhoS[2]),phi,phiS[1],phiS[2]),
           return(I)
       )$
```

(%i57) densS*sfera(rhoS)

(%o57) $\begin{pmatrix} \frac{2\,M\,R^2}{5} & 0 & 0 \\ 0 & \frac{2\,M\,R^2}{5} & 0 \\ 0 & 0 & \frac{2\,M\,R^2}{5} \end{pmatrix}$

3.1 Sfera Cava di raggio R

La cavità è formata da una seconda sfera di raggio R_v, tale che $R_v < 0$.

(%i58) rv:[0, R[v]]$

(%i59) densSv:M/(4/3*%pi*R^3-4/3*%pi*R[v]^3)

(%o59) $\dfrac{M}{\frac{4\,\pi\,R^3}{3} - \frac{4\,\pi\,R_v^3}{3}}$

```
(%i60) sferaV(rhoS, rv):=block(
       [I, cavo, Icavo, s, sTs, p, x, y, z, J, thetaS, phiS],
           thetaS:[0, 2*%pi], phiS:[0,%pi],
           x:rho*cos(theta)*sin(phi),
           y:rho*sin(theta)*sin(phi),
           z:rho*cos(phi),
           p:matrix([x],[y],[z]),
           s:matrix([0,-p[3][1],p[2][1]],
                    [p[3][1],0,-p[1][1]],
                    [-p[2][1],p[1][1],0]),
           sTs:trigsimp(transpose(s).s),
           J:abs(trigsimp(determinant(
             addcol(diff(p,rho),diff(p,theta),diff(p,phi))))),
           assume(sin(phi)>0),
           I:integrate(integrate(integrate(
             J*sTs,theta,thetaS[1],thetaS[2]),
             rho,rhoS[1],rhoS[2]),phi,phiS[1],phiS[2]),
           cavo:integrate(integrate(integrate(
             J*sTs,theta,thetaS[1],thetaS[2]),
             rho,rv[1],rv[2]),phi,phiS[1],phiS[2]),
           Icavo:I-cavo,
           return(Icavo)
       )$
(%i64) factor(densSv*sferaV(rhoS,rv))
```

(%o64) $\left(\frac{2\,M\,(R_v^4+R\,R_v^3+R^2\,R_v^2+R^3\,R_v+R^4)}{5\,(R_v^2+R\,R_v+R^2)}, 0, 0; 0, \frac{2\,M\,(R_v^4+R\,R_v^3+R^2\,R_v^2+R^3\,R_v+R^4)}{5\,(R_v^2+R\,R_v+R^2)}, 0; 0, 0, \frac{2\,M\,(R_v^4+R\,R_v^3+R^2\,R_v^2+R^3\,R_v+R^4)}{5\,(R_v^2+R\,R_v+R^2)}\right)$

(%i65)

Funzioni di utility

1) Determinare le dimensioni di una matrice.

```
(%i1) size(M):=[length(M), length(transpose(M))]$
```

2) Calcolare la matrice antisimmetrica dato un vettore di $\mathbb{R}^3$

```
(%i2) S(v):=block([s1,s2,s3,S],
          s1:matrix([0],[v[3,1]],[-v[2,1]]),
          s2:matrix([-v[3,1]],[0],[v[1,1]]),
          s3:matrix([v[2,1]],[-v[1,1]],[0]),
          S:s1, S:addcol(S,s2), S:addcol(S,s3),
          return(S))$
```

3) Calcolare la matrice di rotazione $R_v(\theta)$ attorno un asse v di un angolo ϑ, in $\mathbb{R}^3$

```
(%i3) rodrigues(v,theta):=block([S,I,S2],
        S:S(v), I:ident(size(v)[1]), S2:S.S,
        rodr:I+S2*(1-cos(theta))+S*sin(theta),
        return(trigsimp(trigexpand(trigreduce(rodr)))))$
```

4) Calcolare la matrice di rotazione $R_x(\theta)$ in $\mathbb{R}^2$

```
(%i4) rot2(theta):=block([rot2],
        rot2:matrix([cos(theta), -sin(theta)],[sin(theta), cos(theta)]),
        return(rot2))$
```

5) Calcolare matrice di rotazione tramite esponenziale di matrice.

```
(%i5) expLaplace(M, theta):=block(
      [dim, I, sAi, sAiT],
            dim:size(M),
            if(dim[1]#dim[2]) then error("Matrix not Square"),
            I:ident(dim[1]),
            sAi:invert(s*I-M),
            sAiT:sAi,
            for i:1 thru dim[1] do(
             for j:1 thru dim[2] do(
              sAiT[i,j]:ilt(sAi[i,j],s,theta)
             )
            ),
            return(expand(trigreduce(trigsimp(sAiT))))
      )$
```

6) Matrice di Avvitamento dati asse v, angolo ϑ e spostamento d: $\mathrm{Av}(v,\theta,d)=\begin{pmatrix} e^{\boldsymbol{S}_v(\theta)} & d\,v \\ 0 & 1 \end{pmatrix}$

```
(%i6) Av(v, theta, d):=block(
      [exp, S, dv, row, T],
            S:S(v),
            exp:expLaplace(S, theta),
            row:matrix([0,0,0,1]),
            dv:d*v,
            T:addcol(exp, dv),
            T:addrow(T,row),
            return(trigsimp(trigrat(trigreduce(trigexpand(T)))))
      )$
```

7) Matrice della trasformazione completa: $Q_{i-1,i}=\mathrm{Av}(z,\theta_i,d_i)\mathrm{Av}(x,\alpha_i,a_i)$

```
(%i7) Q(thetai,di,alphai,ai):=block(
      [Avz, Avx, x, z, Q], z:matrix([0],[0],[1]), x:matrix([1],[0],[0]),
            Avz:Av(z,thetai,di),
            Avx:Av(x,alphai,ai),
            Q:Avz.Avx,
            return(Q)
      )$
```

8) Nelle procedure, quando necessario, è stata applicata la ridefinizione della notazione seno e coseno nella forma:

$$\cos(\theta_1+\cdots+\theta_n)=c_{1\ldots n}$$
$$\sin(\theta_1+\cdots+\theta_n)=s_{1\ldots n}$$

```
(%i8) ridef(M,q1,q2,q3):=block(let(cos(q3+q2), c[23]),let(sin(q3+q2),s[23]),
      let(cos(q1),c[1]),let(sin(q1),s[1]),let(cos(q2),c[2]),let(sin(q2),s[2]),
      let(D3*cos (q3+q2)+D2*cos (q2), D3*c[23]+D2*c[2]),return(letsimp(M)))$
```

9) Nelle procedure è stato inserito come argomento la lunghezza dei link dei robot, indicando con L le lunghezze dei link che vengono orientati lungo l'asse Z e con D quelle dei link orientati lungo l'asse X

Determino la posizione del baricentro di un link data la corrispondente riga di Denavit-Hartenberg.

```
(%i9) baricentro(row):=block(
      [bc],
          bc:addrow(addcol(ident(3),transpose(row)),[0,0,0,1]),
          return(bc)
      )$
```

Creo una matrice di inerzia in forma simbolica.

```
(%i10) inerzia(j):=block(
       [II],
       II:matrix([I[xx[i]],I[xy[i]],I[xz[i]]],
                 [I[xy[i]],I[yy[i]],I[yz[i]]],
                 [I[xz[i]],I[yz[i]],I[zz[i]]]),
       II:subst(j, i, II),
       return(II)
       )$
```

Estendo la funzione diff() per la derivata di una funzione composta (secondo specifiche del manuale)

```
(%i11) derivata(f, n):=block(
       [df],
       df:0,
       for i:1 thru n do(
         df: df+diff(f,q[i])*v[i]
       ),
       return(df)
       )$
```

$\frac{\partial L(q,\dot{q})}{\partial q}$

```
(%i12) diffPos(f,dof):=block(
       [df, i, dfi],
       df: matrix([diff(f,q[1])]),
       for i:1 thru dof-1 do(
         dfi:diff(f,q[i+1]),
         df:addrow(df,matrix([dfi]))
       ), return(df)
       )$
```

$\frac{d}{dt}\frac{\partial L(q,\dot{q})}{\partial \dot{q}}$

```
(%i13) diffVelT(f,dof):=block(
       [df, i, dfi],
       /*dL/dq_punto = diff(f,v),v
         d/dt q_punto = a*/
       /*Aggiungo le derivate rispetto alle posizioni per la dipendenza da q(t)*/
       df: matrix([diff(diff(f,v[1]),v[1])*a[1]+diff(diff(f,v[1]),q[1])*v[1]]),
       for i:1 thru dof-1 do(
         dfi:diff(diff(f,v[i+1]),v[i+1])*a[i+1]+
             diff(diff(f,v[i+1]),q[i+1])*v[i+1],
         df:addrow(df,matrix([dfi]))
       ), return(df)
       )$
```

$\frac{\partial \mathfrak{F}(\dot{q})}{\partial \dot{q}}$

```
(%i14) diffAttr(f,dof):=block(
       [df, i, dfi],
       df: matrix([diff(f,v[1])]),
       for i:1 thru dof-1 do(
         dfi:diff(f,v[i+1]),
         df:addrow(df,matrix([dfi]))
       ), return(df)
       )$
```

Definisco procedura per estrarre la forma quadratica di una funzione.

```
(%i15) formaQuad(func, n) := block(
           [B],
           B: zeromatrix(n,n),
           /*Derivo due volte: ottengo i termini quadratici nelle velocità*/
           for k:1 thru n do(
               B[k, k]: diff(func, v[k], 2)),
           /*Derivo i termini misti:*/
           /*ottengo i termini sopra e sotto la diagonale principale.*/
           for i:1 thru n do(
               for j:i+1 thru n do(
                   B[i, j]: 1/2*diff( diff( func, v[j] ), v[i] ),
                   B[j, i]: B[i, j]
               )
           ),
           return (B)
       )$
```

Definisco procedura per il calcolo dell'energia cinetica di un link, data la corrispondente matrice di DH traslata nel baricentro.

```
(%i16) Tlink(Qh, dof, M):=block(
       [Tr, Tt, w, wt, II, R, Rt, Rd, Sw, result],
            R:submatrix(4,Qh,4),
            d:submatrix(4,Qh,1,2,3),
            Rd:derivata(R, dof),
            dd:derivata(d, dof),
            Sw:trigsimp(Rd.transpose(R)),
            w:matrix([Sw[3][2]],[Sw[1][3]],[Sw[2][1]]),
            wt:transpose(w),
            dd:trigsimp(trigreduce(trigexpand(dd))),
            ddt:transpose(dd),
            II:inerzia(dof),
            Tt:1/2*M*trigsimp(trigreduce(trigexpand(ddt.dd))),
            Tr:1/2*trigsimp(trigreduce(trigexpand(wt.R.II.transpose(R).w))),
            result:[Tr, Tt, II],
            return(result)
       )$
```

Definisco procedura per il calcolo dell'energia potenziale di un link, data la corrispondente matrice di DH traslata nel baricentro.

```
(%i17) Ulink(Qh, M):=block(
       [g, U, d],
           /*Approssimo accelerazione gravitazionale a 10 m/s^2*/
            g:[0,0,10],
            d:submatrix(4,Qh,1,2,3),
            U:-M*g.d,
            return(U)
       )$
```

Calcolo l'energia di un robot, fornendo la Massa dei link in forma di lista ($[M_1, M_2, M_3]$) e la relativa tabella di Denavit-Hartenberg.

```
(%i18) energia(DH, M, Trsz):=block(
       [T, U, E1, E2, E3, Q, i, row, rowList, dof, M1, M2, M3,
        Q01, Q12, Q23, Q03, Q01h, Q12h, Q23h, Q03h, g, U1, U2, U3],
        let(cos(q[1]),c[1]),let(cos(q[2]),c[2]),let(cos(q[3]),c[3]),
            let(sin(q[1]),s[1]),let(sin(q[2]),s[2]),let(sin(q[3]),s[3]),
            let(cos(q[1]+q[2]),c[12]),let(sin(q[1]+q[2]),s[12]),
            let(cos(q[3]+q[2]),c[23]),let(sin(q[3]+q[2]),s[23]),
            dof: size(DH)[1],
            rowList:[], Q:[], Qh:[],
       /*Carico la tabella di DH per eseguire in modo atomico i calcoli*/
            for i:1 thru dof do(
              rowList:append(rowList,[DH[i]]),
              row:rowList[i],
              Q:append(Q,[Q(row[1],row[2],row[3],row[4])])),
       /*Controllo quanti gradi di libertà ha il robot e calcolo l'energia*/
            if (dof = 2) then(
            M1:M[1], M2:M[2],
            Q01:Q[1], Q12:Q[2], Q01h:Q01.baricentro(Trsz[1]),
            Q02:trigreduce(Q01.Q12), Q02h:Q02.baricentro(Trsz[2]),
            T1: factor(Tlink(Q01h, 1, M1)), U1: Ulink(Q01h,M1),
            T2: factor(Tlink(Q02h, 2, M2)), U2: Ulink(Q02h,M2),
            T:letsimp(formaQuad(T1[1]+T1[2]+T2[1]+T2[2],2)), U:letsimp(U1+U2)
            ),
            if (dof = 3) then(
            M1:M[1], M2:M[2],M3:M[3],
            Q01:Q[1], Q12:Q[2], Q23:Q[3],
            Q01h:Q01.baricentro(Trsz[1]),
            Q02:trigreduce(Q01.Q12), Q02h:Q02.baricentro(Trsz[2]),
            Q03:trigreduce(Q02.Q23), Q03h:Q03.baricentro(Trsz[3]),
            T1: factor(Tlink(Q01h, 1, M1)), U1: Ulink(Q01h,M1),
            T2: factor(Tlink(Q02h, 2, M2)), U2: Ulink(Q02h,M2),
            T3: factor(Tlink(Q03h, 3, M3)), U3: Ulink(Q03h,M3),
            T:letsimp(formaQuad(T1[1]+T1[2]+T2[1]+T2[2]+T3[1]+T3[2],3)),
            U:letsimp(U1+U2+U3)
           ),
           return([T,U])
       )$
```

```
(%i19) euleroLagrange(DH,M,F,u,Trsz):=block(
       [e, T, U, L, i, dFddq, dLdq, dLddq, dq, dqT, eq:[], solEL:[]],
           /*T: energia cinetica, U: energia potenziale gravitazionale*/
           /*F: attrito statico, dLdq: derivata di L rispetto q*/
           /*dLddq: derivata di L rispetto a q punto, e: energia*/
           /*q: variabili di giunto*/
           /*EL: equazioni eulero lagrange*/
           e:energia(DH,M,Trsz),
           /*print(e),*/
           dof: size(e[1])[1],
           for i:1 thru  dof do(
             solEL:append([0],solEL)
           ),
           dq:matrix([v[1]]),
           for  i:1 thru dof-1 do(
             dq:addrow(dq,matrix([v[i+1]]))
           ),
           dqT:transpose(dq),
           T:1/2*dqT.e[1].dq,
           U:e[2],
           L:T-U,
           dLdq:diffPos(L,dof),
           dLddq:diffVelT(L,dof),
           dFddq:diffAttr(F, dof),
           for i:1 thru dof do(
             eq:append(eq,dLddq[i]-dLdq[i]+dFddq[i]-u[i]),
             solEL[i]:solve(eq[i],a[i])
           ),
           return(solEL)
       )$
```

Funzione per la sintesi di equazioni interpretabili da MatLab per la simulazione delle equazioni di Eulero-Lagrange.

Se si vuole il risultato sottoforma di stringa, inserire 1 come secondo argomento, zero altrimenti.

```
(%i20) sintesys(solEl, bool):=block(
       [system, numSol, x:[x[1],x[2],x[3],x[4],x[5],x[6]], a:[], i],

       numSol:length(solEl),
       for i:1 thru numSol do (
          a:append([0],a)
       ),
        /*Converto soluzioni in lista semplice*/
       for i:1 thru  numSol do(
         a[i]:map(rhs,solEl[i])[1]
       ),
       if(numSol = 2) then(
         system:matrix([x[3]],[x[4]],[a[1]],[a[2]]),
         system:subst([q[1]=x[1],q[2]=x[2],v[1]=x[3],v[2]=x[4]],system)
       ),
       if(numSol = 3) then(
         system:matrix([x[4]],[x[5]],[x[6]],[a[1]],[a[2]],[a[3]]),
         system:subst([q[1]=x[1],q[2]=x[2],q[3]=x[3],
                     v[1]=x[4],v[2]=x[4],v[3]=x[6]],system)
       ),
       if (bool = 1) then return(string(system)),
       return(system)

       )$
```

Funzione di test per verificare corretto funzionamento delle funzioni derivata, basata sull'esempio presentato a lezione 31.

```
(%i21) euleroLagrangeTest():=block(
       [M, T, U, L, F, dof, i, dFddq, dLdq, dLddq, eq:[], solEL:[0,0,0], u],
           /*T: energia cinetica, U: energia potenziale gravitazionale*/
           /*F: attrito statico, dLdq: derivata di L rispetto q*/
           /*dLddq: derivata di L rispetto a q punto, e: energia*/
           /*q: variabili di giunto*/
           /*eq: equazioni di eulero Lagrange*/
           /*solEL: soluzione equazioni eulero lagrange*/
           M:[M[1],M[2],M[3]],
           T: (1/2)*M[1]*v[1]^2+(1/2)*M[2]*v[2]^2+(1/2)*M[3]*v[3]^2,
           U: U:(1/2)*K[1]*(q[1])^2+(1/2)*K[2]*(q[1]-q[2])^2+
              (1/2)*K[3]*(q[2]-q[3])^2+(1/2)*K[4]*(q[3])^2+(1/2)*K[5]*(q[2])^2,
           F:(1/2)*D[1]*v[1]^2+(1/2)*D[2]*v[2]^2+(1/2)*D[3]*v[3]^2,
           L:T-U,
           u:matrix([u[1]],[u[2]],[u[3]]),
           dof:3,
           dLdq:diffPos(L,dof),
           dLddq:diffVelT(L,dof),
           dFddq:diffAttr(F, dof),
           for i:1 thru dof do(
             eq:append(eq,dLddq[i]-dLdq[i]+dFddq[i]-u[i]),
             solEL[i]:solve(eq[i],a[i])
           ),
           print("Energia Cinetica T = ",T),
           print("Energia Potenziale U = : ",U),
           print("Forze di Attrito F = ",F),
           print("L = T-U = ",L),
           print("Forze Esterne: ",u),
           print("d/dt dL/dv = ", dLddq),
           print("dL/dq = ",dLdq),
           print("dF/dv = ",dFddq),
           return(solEL)
       )$
(%i22) test:euleroLagrangeTest()
```

Energia Cinetica T $= \frac{M_3\,v_3^2}{2}+\frac{M_2\,v_2^2}{2}+\frac{M_1\,v_1^2}{2}$

Energia Potenziale U $=$: $\frac{q_2^2\,K_5}{2}+\frac{q_3^2\,K_4}{2}+\frac{K_3\,(q_2-q_3)^2}{2}+\frac{K_2\,(q_1-q_2)^2}{2}+\frac{K_1\,q_1^2}{2}$

Forze di Attrito F $= \frac{D_3\,v_3^2}{2}+\frac{D_2\,v_2^2}{2}+\frac{D_1\,v_1^2}{2}$

L $=$ T-U $= -\frac{q_2^2\,K_5}{2}-\frac{q_3^2\,K_4}{2}+\frac{M_3\,v_3^2}{2}-\frac{K_3\,(q_2-q_3)^2}{2}+\frac{M_2\,v_2^2}{2}-\frac{K_2\,(q_1-q_2)^2}{2}+\frac{M_1\,v_1^2}{2}-\frac{K_1\,q_1^2}{2}$

Forze Esterne: $\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}$

d/dt dL/dv $= \begin{pmatrix} M_1\,a_1 \\ M_2\,a_2 \\ M_3\,a_3 \end{pmatrix}$

dL/dq $= \begin{pmatrix} -K_2\,(q_1-q_2)-K_1\,q_1 \\ -q_2\,K_5-K_3\,(q_2-q_3)+K_2\,(q_1-q_2) \\ K_3\,(q_2-q_3)-q_3\,K_4 \end{pmatrix}$

$$\text{dF/dv} = \begin{pmatrix} D_1\, v_1 \\ D_2\, v_2 \\ D_3\, v_3 \end{pmatrix}$$

(%o22) $$\left[\left[a_1 = \frac{K_2\, q_2 - q_1\, K_2 - D_1\, v_1 + u_1 - K_1\, q_1}{M_1}\right], \left[a_2 = -\frac{q_2\, K_5 - K_3\, q_3 + q_2\, K_3 + D_2\, v_2 - u_2 + K_2\, q_2 - q_1\, K_2}{M_2}\right], \left[a_3 = -\frac{q_3\, K_4 + D_3\, v_3 - u_3 + K_3\, q_3 - q_2\, K_3}{M_3}\right]\right]$$

```
(%i23) sintesys(test,0)
```

(%o23) $$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{-D_1\, x_4 + K_2\, x_2 - x_1\, K_2 - K_1\, x_1 + u_1}{M_1} \\ -\frac{x_2\, K_5 + D_2\, x_4 - K_3\, x_3 + x_2\, K_3 + K_2\, x_2 - u_2 - x_1\, K_2}{M_2} \\ -\frac{D_3\, x_6 + x_3\, K_4 + K_3\, x_3 - u_3 - x_2\, K_3}{M_3} \end{pmatrix}$$

Definisco le tabelle di DH sottoforma di matrici. In ordine troviamo:

1. Due DOF
2. Cartesiano
3. Cilindrico
4. Scara
5. Sferico Tipo 1
6. Sferico Tipo 2 (Stanford)
7. Antropomorfo

```
(%i24) DH:[matrix([q[1],0,0,L[1]],[q[2],0,0,L[2]]),
          matrix([0,q[1],-%pi/2,0],[-%pi/2,q[2],-%pi/2,0],[0,q[3],0,0]),
          matrix([q[1],L[1],0,0],[0,q[2],-%pi/2,0],[0,q[3],0,0]),
          matrix([q[1],L[1],0,D[1]],[q[2],0,0,D[2]],[0,q[3],0,L[3]]),
          matrix([q[1],L[1],%pi/2,0],[q[2],0,%pi/2,D[2]],[0,q[3],0,0]),
          matrix([q[1],L[1],-%pi/2,0],[q[2],L[2],%pi/2,0],[0,q[3],0,0]),
          matrix([q[1],L[1],%pi/2,0],[q[2],0,0,D[2]],[q[3],0,0,D[3]])]
```

(%o24) $$\left[\begin{pmatrix} q_1 & 0 & 0 & L_1 \\ q_2 & 0 & 0 & L_2 \end{pmatrix}, \begin{pmatrix} 0 & q_1 & -\frac{\pi}{2} & 0 \\ -\frac{\pi}{2} & q_2 & -\frac{\pi}{2} & 0 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & 0 & 0 \\ 0 & q_2 & -\frac{\pi}{2} & 0 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & 0 & D_1 \\ q_2 & 0 & 0 & D_2 \\ 0 & q_3 & 0 & L_3 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & \frac{\pi}{2} & 0 \\ q_2 & 0 & \frac{\pi}{2} & D_2 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & -\frac{\pi}{2} & 0 \\ q_2 & L_2 & \frac{\pi}{2} & 0 \\ 0 & q_3 & 0 & 0 \end{pmatrix}, \begin{pmatrix} q_1 & L_1 & \frac{\pi}{2} & 0 \\ q_2 & 0 & 0 & D_2 \\ q_3 & 0 & 0 & D_3 \end{pmatrix}\right]$$

Definisco matrici di traslazione. Ogni riga di una matrice rappresenta la traslazione da applicare alla matrice di trasformazione del link i-esimo per raggiungere il suo baricentro.
L'ordine con cui sono scritte combacia con quello delle tabelle di DH scritte sopra per semplicità.

```
(%i25) Trsz:[matrix([-L[1]/2,0,0],[-L[2]/2,0,0]),
            matrix([0,L[1]/2,0],[0,L[2]/2,0],[0,0,-L[3]/2]),
            matrix([0,0,-L[1]/2],[0,L[2]/2,0],[0,0,-L[3]/2]),
            matrix([-D[1]/2,0,-L[1]/2],[-D[2]/2,0,0],[0,0,L[3]/2]),
            matrix([0,-L[1]/2,0],[-D[2]/2,0,0],[0,0,-L[3]/2]),
            matrix([0,L[1]/2,0],[0,-L[2]/2,0],[0,0,-L[3]/2]),
            matrix([0,-L[1]/2,0],[-D[2]/2,0,0],[-D[3]/2,0,0])]
```

$$(\%o25)\quad \left[\begin{pmatrix} -\frac{L_1}{2} & 0 & 0 \\ -\frac{L_2}{2} & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & \frac{L_1}{2} & 0 \\ 0 & \frac{L_2}{2} & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & 0 & -\frac{L_1}{2} \\ 0 & \frac{L_2}{2} & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} -\frac{D_1}{2} & 0 & -\frac{L_1}{2} \\ -\frac{D_2}{2} & 0 & 0 \\ 0 & 0 & \frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & -\frac{L_1}{2} & 0 \\ -\frac{D_2}{2} & 0 & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & \frac{L_1}{2} & 0 \\ 0 & -\frac{L_2}{2} & 0 \\ 0 & 0 & -\frac{L_3}{2} \end{pmatrix}, \begin{pmatrix} 0 & -\frac{L_1}{2} & 0 \\ -\frac{D_2}{2} & 0 & 0 \\ -\frac{D_3}{2} & 0 & 0 \end{pmatrix}\right]$$

Esempio Robot 2 DoF Planare

```
(%i26) dueDof:DH[1]$
(%i27) Mdd:[M[1],M[2]]$
(%i28) udd:matrix([u[1]],[u[2]])$
(%i29) Fdd:(1/2)*D[1]*v[1]^2+(1/2)*D[2]*v[2]^2$
(%i30) dd:euleroLagrange(dueDof,Mdd,Fdd,udd,Trsz[1])
```

$$(\%o30)\quad \left[\left[a_1 = -\frac{4\,D_1\,v_1 - 4\,u_1}{4\,I_{zz_2} + 4\,L_1\,L_2\,M_2\cos(q_2) + 4\,I_{zz_1} + (L_2^2 + 4\,L_1^2)\,M_2 + L_1^2\,M_1}\right], \left[a_2 = -\frac{2\,L_1\,v_1^2\,L_2\,M_2\sin(q_2) + 4\,D_2\,v_2 - 4\,u_2}{4\,I_{zz_2} + L_2^2\,M_2}\right]\right]$$

```
(%i31) sintesys(dd,0)
```

$$(\%o31)\quad \begin{pmatrix} x_3 \\ x_4 \\ -\frac{4\,D_1\,x_3 - 4\,u_1}{4\,I_{zz_2} + 4\,L_1\,L_2\,M_2\cos(x_2) + 4\,I_{zz_1} + (L_2^2 + 4\,L_1^2)\,M_2 + L_1^2\,M_1} \\ -\frac{2\,L_1\,L_2\,M_2\,x_3^2\sin(x_2) + 4\,D_2\,x_4 - 4\,u_2}{4\,I_{zz_2} + L_2^2\,M_2} \end{pmatrix}$$

Esempio Robot Cilindrico

```
(%i32) cilin:DH[3]$
(%i33) Mc:[M[1],M[2],M[3]]$
(%i34) uc:matrix([u[1]],[u[2]],[u[3]])$
(%i35) Fc:(1/2)*D[1]*v[1]^2+(1/2)*D[2]*v[2]^2+(1/2)*D[3]*v[2]^2$
(%i36) cil:euleroLagrange(cilin,Mc,Fc,uc,Trsz[3])
```

$$(\%o36)\quad \left[\left[a_1 = -\frac{4\,D_1\,v_1 - 4\,u_1}{4\,I_{yy_3} + 4\,I_{yy_2} + 4\,I_{zz_1} + 4\,M_3\,q_3^2 - 4\,L_3\,M_3\,q_3 + L_3^2\,M_3}\right], \left[a_2 = \frac{10\,M_3 - v_2\,D_3 - D_2\,v_2 + u_2 + 10\,M_2}{M_3 + M_2}\right], \left[a_3 = \frac{2\,u_3 + 2\,v_1^2\,M_3\,q_3 - v_1^2\,L_3\,M_3}{2\,M_3}\right]\right]$$

```
(%i37) sintesys(cil,0)
```

$$(\%o37)\quad \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\frac{4\,D_1\,x_4 - 4\,u_1}{4\,I_{yy_3} + 4\,I_{yy_2} + 4\,I_{zz_1} + 4\,M_3\,x_3^2 - 4\,L_3\,M_3\,x_3 + L_3^2\,M_3} \\ \frac{-D_3\,x_4 - D_2\,x_4 + 10\,M_3 + u_2 + 10\,M_2}{M_3 + M_2} \\ \frac{2\,M_3\,x_3\,x_4^2 - L_3\,M_3\,x_4^2 + 2\,u_3}{2\,M_3} \end{pmatrix}$$

Esempio Robot Cartesiano

```
(%i38) cartes:DH[2]$
```

```
(%i39) Mcart:[M[1],M[2],M[3]]$
(%i40) ucart:matrix([u[1]],[u[2]],[u[3]])$
(%i41) Fcart:(1/2)*D[1]*v[1]^2+(1/2)*D[2]*v[2]^2+(1/2)*D[3]*v[2]^2$
(%i42) cart:euleroLagrange(cartes,Mcart,Fcart,ucart,Trsz[2])
```

(%o42) $$\left[\left[a_1 = \frac{10\,M_3 + 10\,M_2 - D_1\,v_1 + u_1 + 10\,M_1}{M_3 + M_2 + M_1}\right], \left[a_2 = -\frac{v_2\,D_3 + D_2\,v_2 - u_2}{M_3 + M_2}\right], \left[a_3 = \frac{u_3}{M_3}\right]\right]$$

```
(%i43) sintesys(cart,0)
```

(%o43) $$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{-D_1\,x_4 + 10\,M_3 + 10\,M_2 + u_1 + 10\,M_1}{M_3 + M_2 + M_1} \\ -\frac{D_3\,x_4 + D_2\,x_4 - u_2}{M_3 + M_2} \\ \frac{u_3}{M_3} \end{pmatrix}$$

```
(%i44)
```

Visione Artificiale

La seguente procedura espone il riconoscimento di un oggetto attraverso l'elaborazione al calcolatore di un'immagine che lo contiene.

Pre-Processamento

Per effettuare il riconoscimento, vengono fatte delle operazioni pre-processamento dell'immagine, riducendone la qualità se questa è a risoluzione eccessiva rispetto all'obiettivo da perseguire, per poi trasformarla prima in scala di grigi e poi in bianco e nero.

Processamento

Successivamente, il processamento dell'immagine viene fatto eliminando il più possibile i disturbi dovuti ad eventuali riflessi ed ombre, che possono risultare in "buchi" all'interno dell'oggetto che vogliamo identificare.

Inoltre, vengono applicati operatori di Dilatazione ed Erosione per migliorare i contorni dell'oggetto.

Orientamento

L'orientamento viene calcolato prima attraverso la trasformata di Radon, ovvero, la proiezione dell'intensità dell'immagine su un piano orientato ad un angolo specifico. In particolare, è stato scelto un set di angolo che va da 0 a 179 per avere uno spettro dell'immagine completo. Nella foto mostrato sotto, l'oggetto è riconoscibile nel suo orientamento nel punto in cui l'intensità luminosa è maggiore.

Successivamente, viene identificato il massimo della matrice delle proiezioni, per identificare gli angoli delle diagonali principali dell'oggetto identificato e trovare il suo baricentro.

Infine, l'orientamento viene calcolato con riferimento all'orizzontale.

Il Riconoscimento

Il riconoscimento degli oggetti viene fatto attraverso un algoritmo che stabilisce la forma dell'oggetto in base al rapporto tra area e perimetro, secondo la seguente logica:

$$Area = \frac{perimetro \cdot apotema}{2}$$

Poiché l'apotema è un numero fisso, descritto da

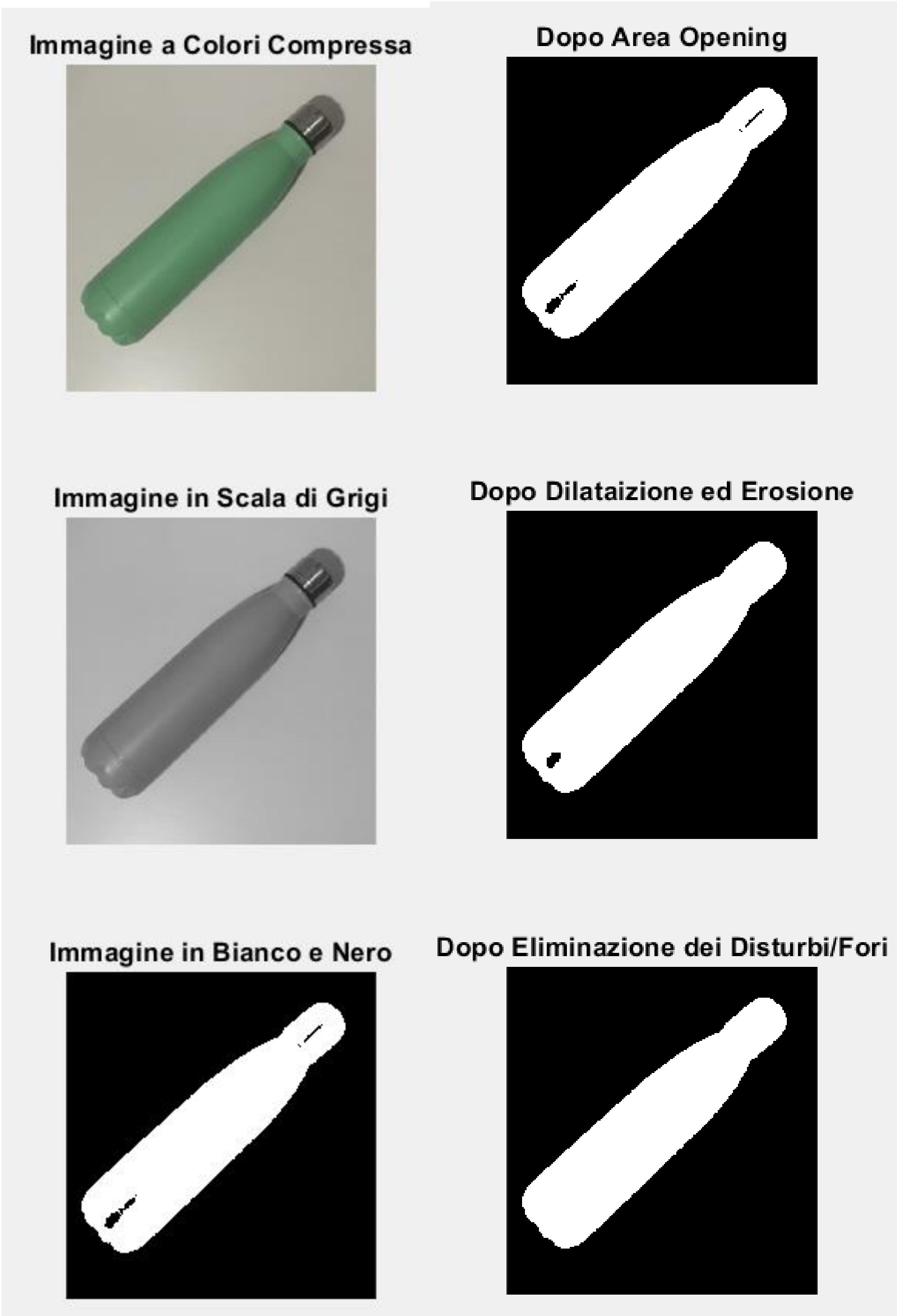
$$apotema = \frac{perimetro}{\#lati}$$

Possiamo dire che:

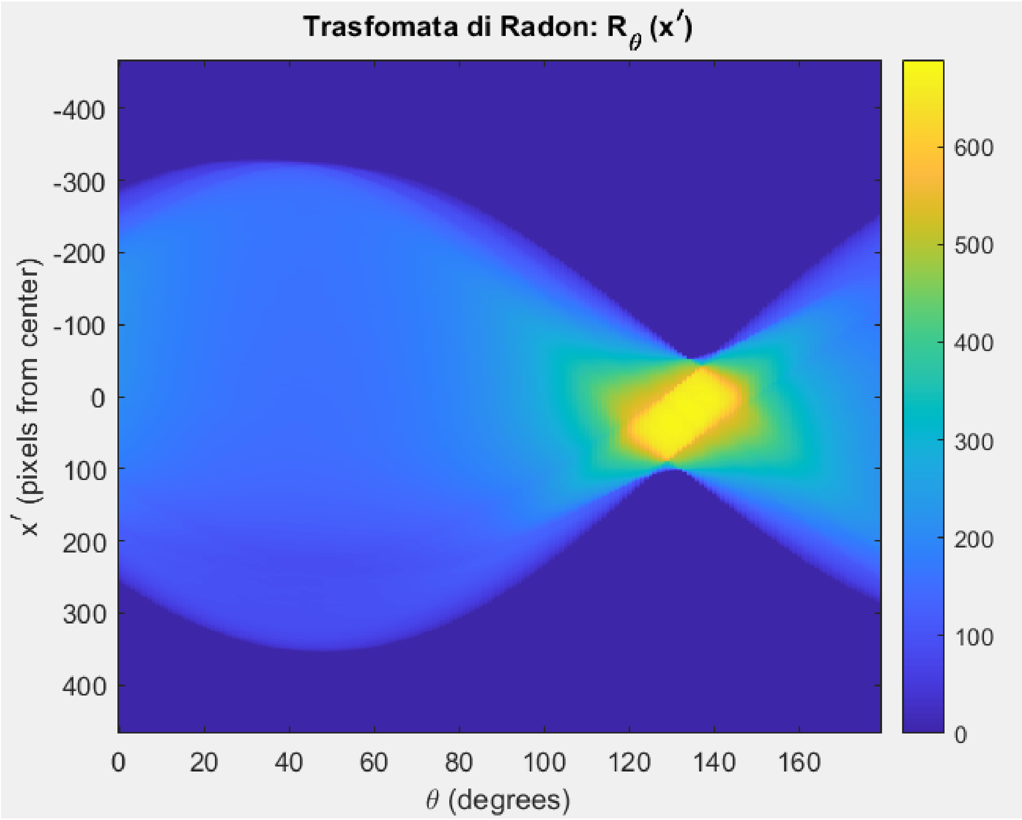
$$apotema = \frac{2 \cdot Area}{perimetro}$$

Questo ultimo rapporto, viene confrontato con il numero fisso di figure come triangoli, quadrati, pentagoni, esagoni e cerchi. Ovviamente, il confronto è fatto inserendo una tolleranza rispetto al numero fisso, per inquadrare gli oggetti nell'intorno del numero fisso, per avere una stima accettabile.

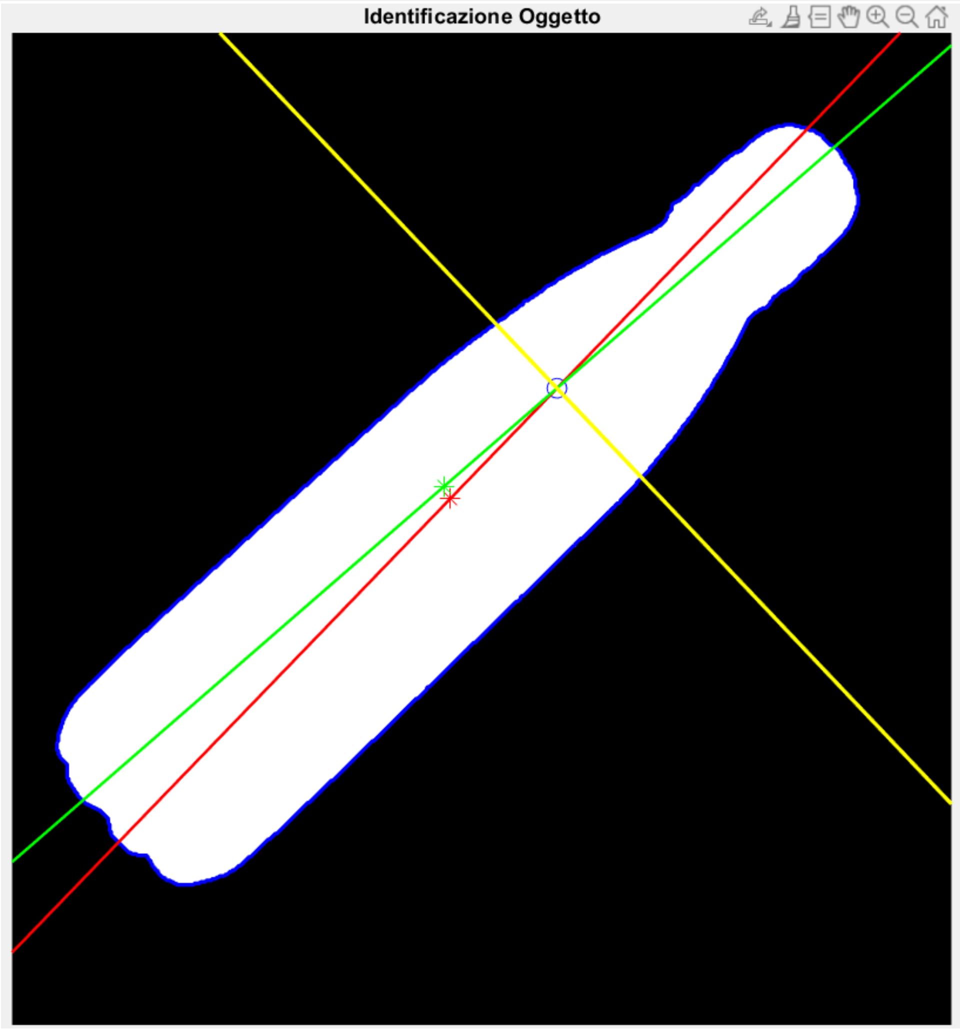
Per i cerchi, l'algoritmo è modificato appositamente rispetto all'area del cerchio.



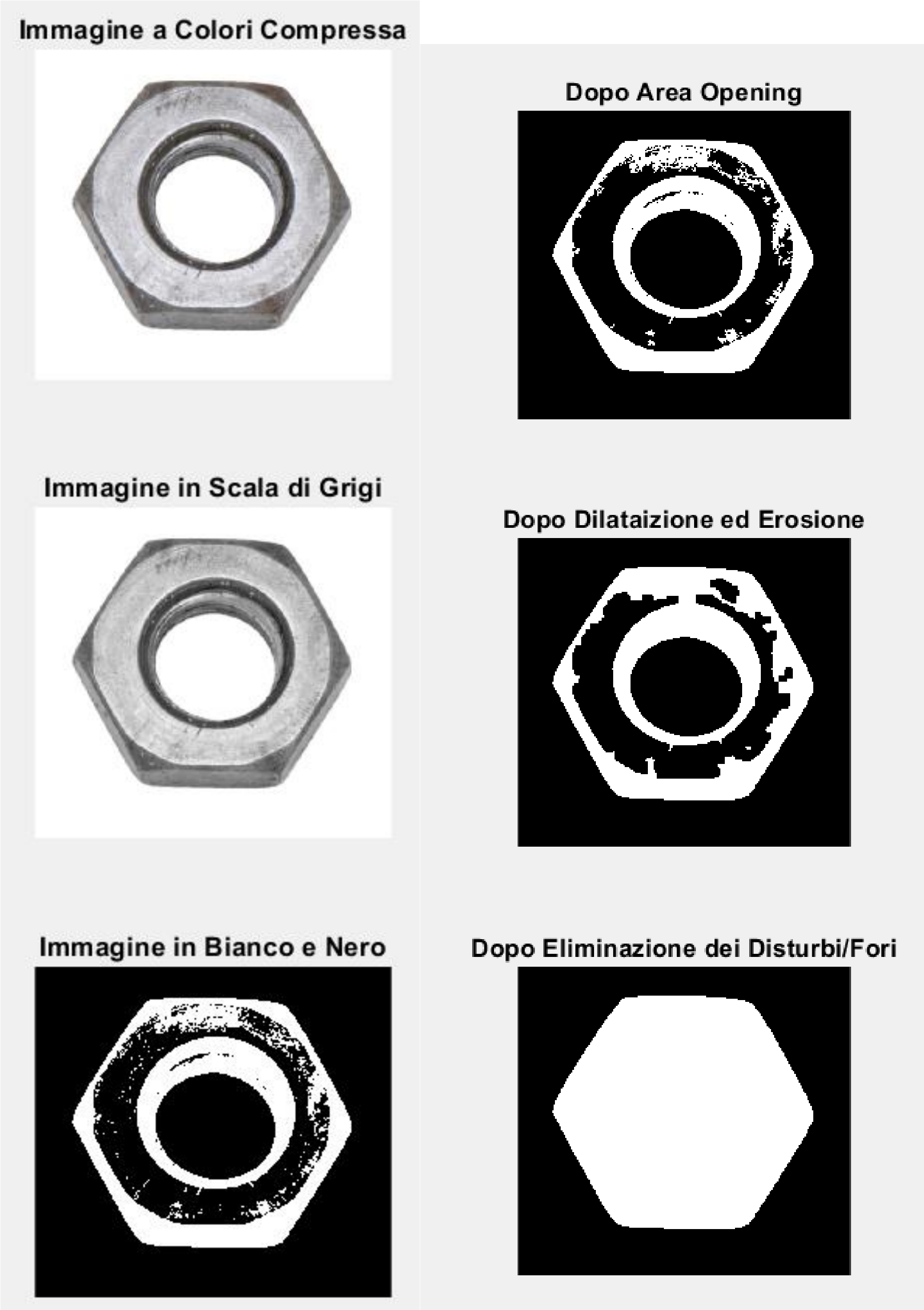

```
Area: 92688.000000, Perimetro: 1547.889000
Forma dell'Oggetto: Irregolare
Baricentro dell'Oggetto: X[bc] = 51.751498, Y[bc] = 95.337385
Orientamento dell'Oggetto: -46.500000°
Tempo di Processamento: 0.154444 ms
```



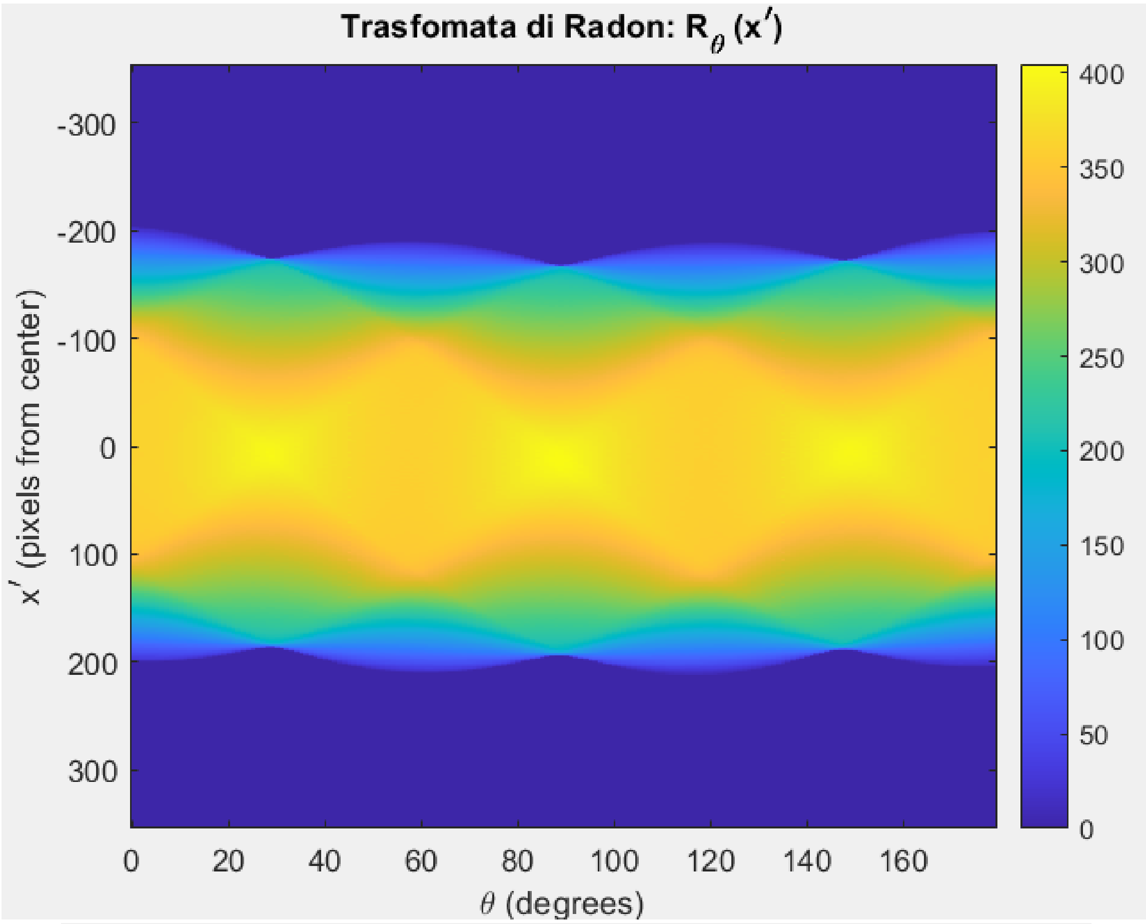
Trasfomata di Radon: $R_\theta (x')$
-400
-300
-200
-100
0
100
200
300
400
x' (pixels from center)
0
20
40
60
80
100
120
140
160
θ (degrees)
600
500
400
300
200
100
0

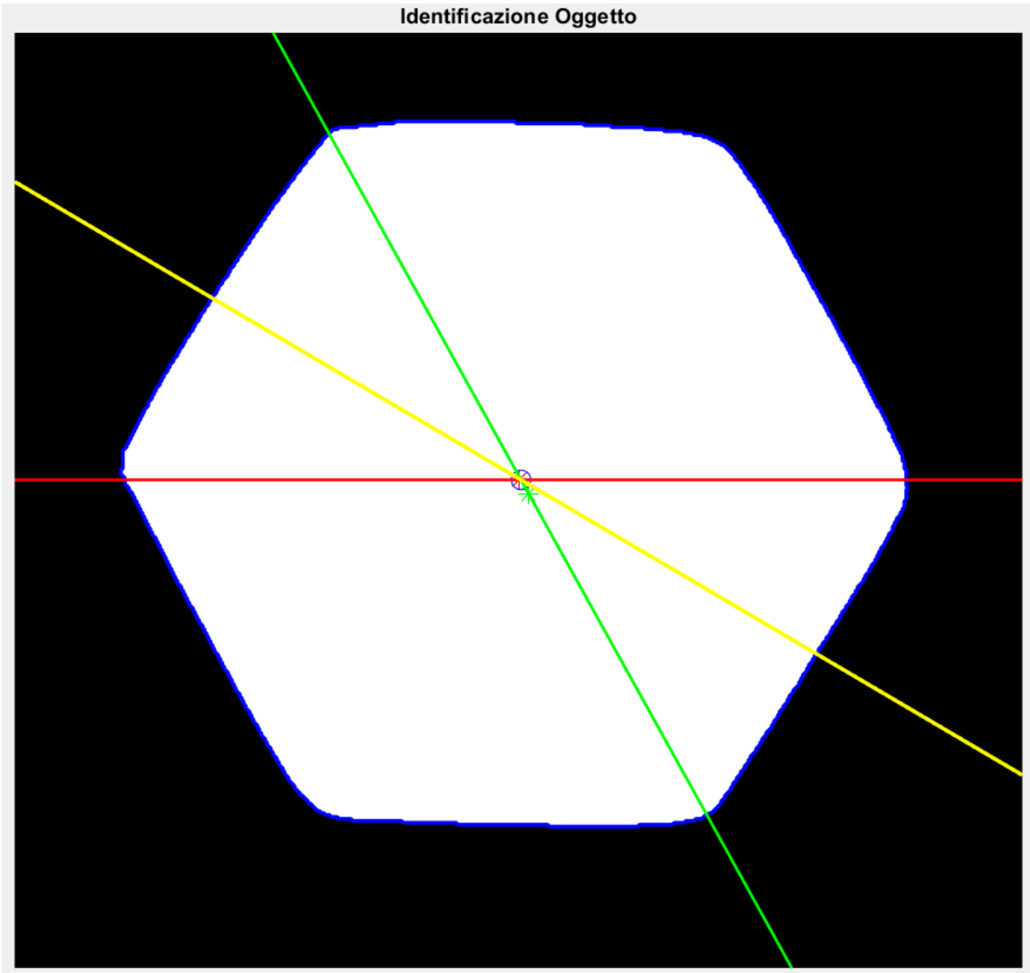


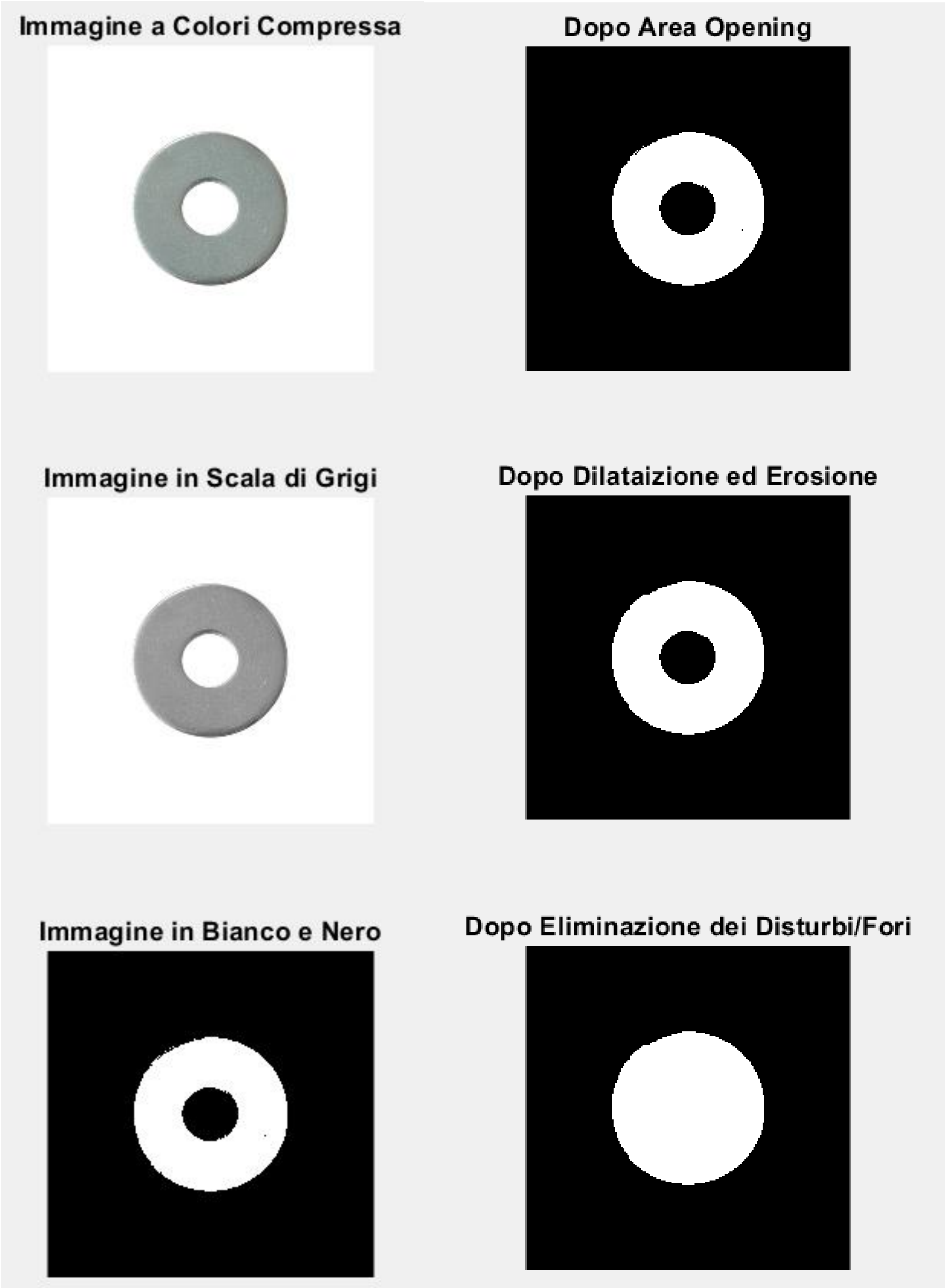
Identificazione Oggetto



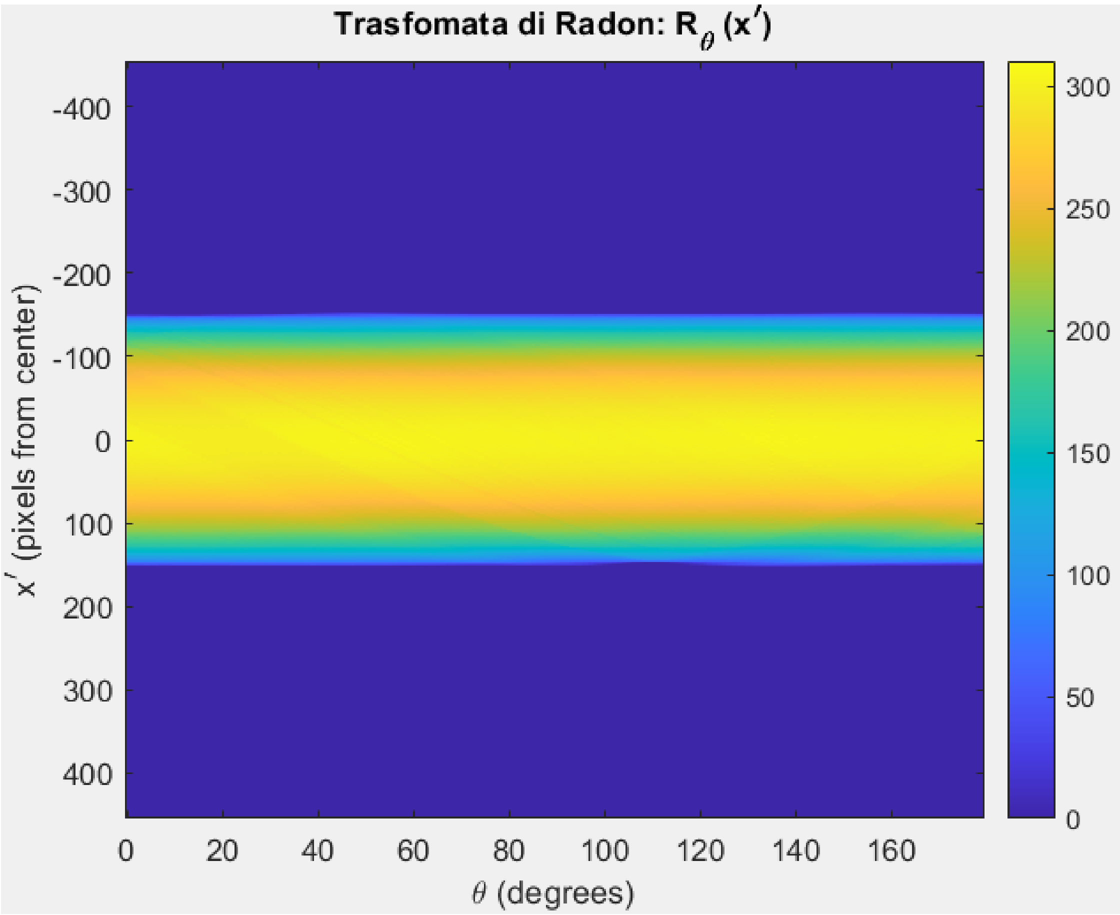

```
Area: 111304.000000, Perimetro: 1207.590000
Forma dell'Oggetto: Esagono
Baricentro dell'Oggetto: X[bc] = 1.317034, Y[bc] = 10.000000
Orientamento dell'Oggetto: -30.500000°
Tempo di Processamento: 0.153015 ms
```

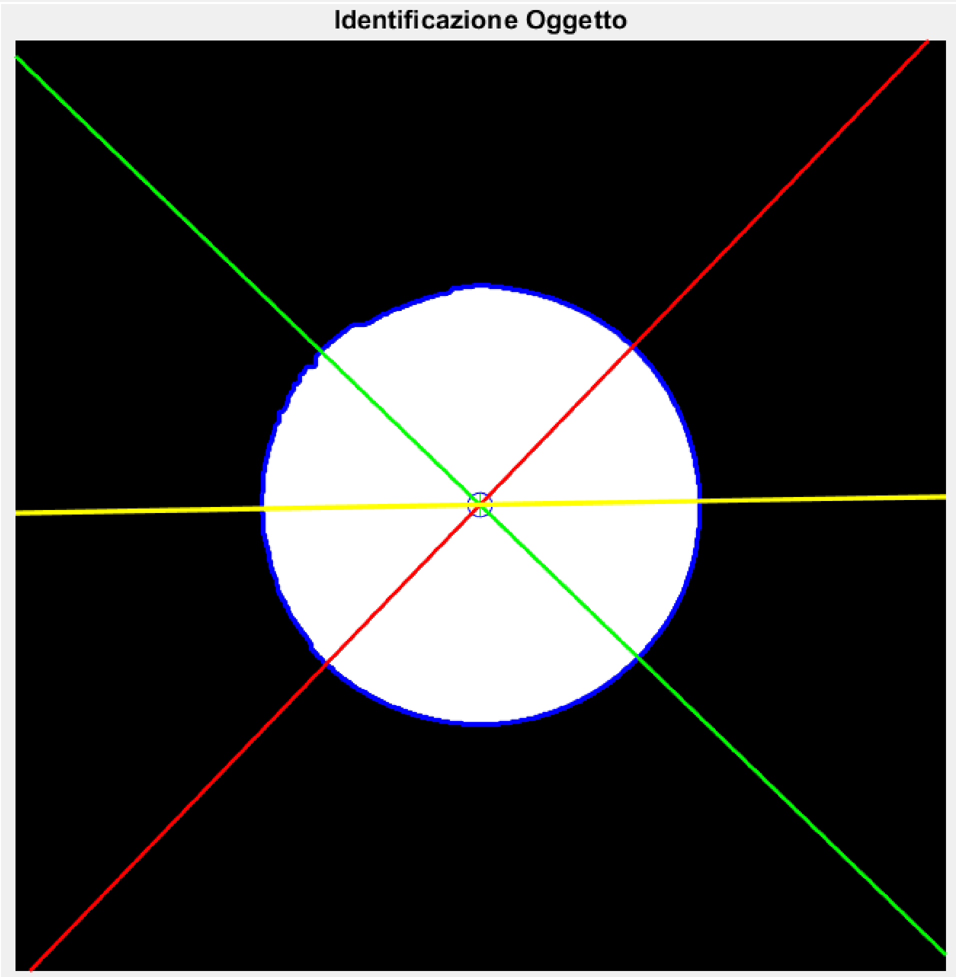

Trasfomata di Radon: $R_\theta(x')$
-300
-200
-100
0
100
200
300
x' (pixels from center)
0
20
40
60
80
100
120
140
160
θ (degrees)
400
350
300
250
200
150
100
50
0


Identificazione Oggetto




```
Area: 71741.000000, Perimetro: 958.739000
Forma dell'Oggetto: Circolare
Baricentro dell'Oggetto: X[bc] = 0.000000, Y[bc] = 0.000000
Orientamento dell'Oggetto: 1.000000°
Tempo di Processamento: 0.138703 ms
```


Trasfomata di Radon: $R_\theta(x')$
-400
-300
-200
-100
0
100
200
300
400
x' (pixels from center)
0
20
40
60
80
100
120
140
160
θ (degrees)
300
250
200
150
100
50
0


Identificazione Oggetto

```matlab
%% Visione Artificiale
% Determinare perimetro, area, forma, centro ed orientamento di un oggetto
% all'interno in un'immagine.
% Andrea Efficace 0300125
% In questo script verranno utilizzate funzioni dell'Image Processing
% Toolbox di maxima
clearvars
close all
clc
syms x diag1(x) diag2(x)
%% Lettura Immagini e caricamento in workspace
% Calcolatrice Verticale con luce diffusa -> Non Identificata
% filename = 'images/verticalCalc.jpg';
% Calcolatrice in Diagonale con sfondo bianco -> Identificata
% filename = 'images/rotateCalcW.jpg';
% Calcolatrice Orizzontale con Ombra -> Identificata con pxToDel = 40000
% filename = 'images/horizontalCalcShadow.jpg';
% Calcolatrice Orizzontale sfondo bianco -> Identificata
% filename = 'images/horizontalCalcW.jpg';
% Calcolatrice Orizzontale sfondo nero -> Identificata
% filename = 'images/horizontalCalcB.jpg';
% Rondella Circolare -> Identificata
% filename = 'images/rondella.jpg';
% Mensola Triangolare -> Identificata
% filename = 'images/triangolo.jpg';
% Dado Esagonale  -> Identificato
% filename = 'images/dadoEsagono.jpg';
% Controller  -> Bordi troppo diversi da loro, non identificata.
            % -> Identificata con maschera Forte.
% filename = 'images/controller.jpg';
% Borraccia -> Identificata
filename = 'images/rotateBottle.jpg';
% Bottiglia Vetro con Disegno -> identificata
% filename = 'images/verticalGlass.jpg';

imgRGB = imread(filename);  % Caricamente Immagine
%imgRGB = imrotate(imgRGB, 35, 'nearest','crop');

%% Pre-Processamento: Compressione dell'immagine

% Per mantenere un rapporto di grandezze tra l'aspetto dell'immagine in
% ingresso quella post compressione, una delle due dimensioni viene decisa
% dalla libreria in automatico.

maxRes = [480, 640];       % Risoluzione massima desiderata dell'immagine
width = size(imgRGB, 2);    % Larghezza iniziale dell'immagine
height = size(imgRGB, 1);   % Altezza iniziale dell'immagine

if (height > maxRes(1))
   imgRGB = imresize(imgRGB, [maxRes(1) NaN]);
end
if (width > maxRes(2))
   imgRGB = imresize(imgRGB, [NaN maxRes(2)]);
end

width = size(imgRGB, 2);    % Larghezza dell'immagine post-compressione
height = size(imgRGB, 1);   % Altezza dell'immagine post-compressione
res = width*height;         % Risoluzione dell'immagine post-compressione
center = [floor((width+1)/2), floor((height+1)/2)];
%% Avvio Processamento dell'immagine
tic;        % Avvio contatore del tempo di processamento

%% Converto l'immagine a scala di grigi
imgGRS = rgb2gray(imgRGB);

%% Converto l'immagine in B/N
```

```
imgBW = imbinarize(imgGRS);

%% Calcolo il negativo dell'immagine se questa è troppo chiara
% utile per le operazioni successive

% Prima somma per le colonne, seconda per le righe
nPixelW = sum(sum(imgBW));
if(nPixelW >= res/2)
    imgBW_old = imgBW;
    imgBW = imcomplement(imgBW);
end

%% Elimino difetti nell'immagine
% utile per eliminare artefatti grafici (es: ombre)
% pxToDel = 40;        % Soglia di pixel da considerare come rumore.
pxToDel = 40000;      % Soglia per Ombra
imgBW2 = bwareaopen(imgBW,pxToDel);

%% Applico Maschere di Dilatazione ed Erosione
% Definisco operatore morfologico: Applico maschere quadrate
mask = strel('square', 10);
% Maschera per il controller
% mask = strel('square', 250);
% imclose() applica operatori morfologici su immagini in b/n  o greyscale
imgBW3 = imclose(imgBW2, mask);
%% Eliminazione "buchi" dall'immagine
% permette di eliminare artefatti grafici simili a riflessi sulla
% superficie dell'oggetto, o buchi nello stesso.
imgBW4 = imfill(imgBW3, 'holes');
%% Determino Area e Perimetro dell'Oggetto nell'Immagine

% bwboundaries() determina i contorni di un oggetto in un'immagine b/w
% il parametro 'noholes' serve per specificare all'algoritmo di cercare
% soltanto oggetti compatti, velocizzando il processo.
% Tale algoritmo può identificare più oggetti in una singola immagine.
[B, L] = bwboundaries(imgBW4, 'noholes');
% regionprops() calcola il valore effettivo di perimetro ed area.
% Salvo il risultato in un oggetto.
props = regionprops(L, 'Area', 'Perimeter');
% Salvo Area e Perimetro di tutti gli oggetti identificati come lista.
areas = [props.Area];
perims = [props.Perimeter];
% Discrimino l'oggetto da identificare da tutti quelli nell'immagine
objArea = max(areas);
objPerim = max(perims);
% Sapendo che Area = (perimetro * apotema)/2 e che l'apotema è pari ad un
% numero fisso per perimetro/numero_lati, troviamo che
apothem = objArea*2/objPerim;
epsilon = 0.05;
objShape = "Irregolare";
fixed = [0.289,0.5,0.688,0.866];
polig = ["Triangolo", "Quadrilatero", "Pentagono", "Esagono"];
for i=1:4
    numLati = i+2;
    if(apothem/objPerim*numLati < fixed(i)+epsilon)
      if(apothem/objPerim*numLati > fixed(i)-epsilon)
        objShape = polig(i);
        break;
      end
    end
end
% Stesso discorso dei poligoni regolari, ma su una circonferenza
if (strcmp(objShape, "Irregulare") == 1)
    epsilon = 0.55;
    myPi = objPerim^2/(4*objArea);
    if(myPi < pi+epsilon)
        if(myPi > pi-epsilon)
```

```
            objShape = "Circolare";
        end
    end
end

%% Eseguo Trasformata di Radon per determinare baricentro ed orientamento
% Calcoliamo le proiezioni, in modo tale da identificare le diagonali
% principali dell'oggetto da riconoscere. La loro intersezione ne
% determinerà il baricentro.

% Definisco Rispetto a quanti angoli effettuare la proiezione.
theta = 0:179;
% Eseguo le proiezioni.
% R è la trasformata, Xp l'angolo in radianti relativo alla trasformata.
[R, xp] = radon(imgBW4, theta);

% Determino la proiezione con altezza maggiore per determinare la diagonale
maxRadon = max(R);

%% Determino il massimo per identificare il punto più alto della
% proiezione con altezza maggiore.
% [pk, locs] = findpeaks() identifica il massimo locale del vettore dato in
% ingresso e l'indice di quel valore all'interno del vettore.
% SosrtStr specifica che i risultati andranno ordinati
% NPeaks specifica quanti massimi locali trovare nel vettore.
[pk, locs] = findpeaks(maxRadon, 'SortStr', 'descend', 'NPeaks', 2);

% Trovo angolo in radianti il cui indice corrisponde all'elemento di una
% delle colonne di R il cui valore è pari al picco individuato da pk
theta1 = locs(1);
offset1 = xp(R(:, locs(1)) == pk(1));
theta2 = locs(2);
offset2 = xp(R(:, locs(2)) == pk(2));

% Determino le diagonali
diag1(x) = tand(theta1+90) * (x - offset1*cosd(theta1)) + offset1*sind(theta1);
diag2(x) = tand(theta2+90) * (x - offset2*cosd(theta2)) + offset2*sind(theta2);

% Identifico il baricentro nel punto d'intersezione delle diagonali
x_bc = solve(diag1 == diag2);
y_bc = diag1(x_bc);

% Determino Orientamento a partire da due angoli identificati dalla
% trasformata di Radon

orient = (theta1+theta2)/2;
% Controllo se c'è discordanza tra i quadranti identificati dagli angoli
if(sign(sind(theta1)) ~= sign(sind(theta2)) || sign(cosd(theta1)) ~= sign(cosd(theta2)) )
    orient = orient-90;
end
% Mi assicuro che l'angolo trovato sia tra -90° and 90°
orient = atand(sind(orient)/cosd(orient));
%% Fine Processamento
elapseTime = toc;
%% Visualizzo Immagine a Colori
% figure
% imshow(imgRGB)
% title('Immagine a Colori');

%% Visualizzo immagine a colori compressa
figure
subplot(3,1,1)
imshow(imgRGB)
title('Immagine a Colori Compressa');

%% Visualizzo immagine in scala di grigi
subplot(3,1,2)
```

```
imshow(imgGRS)
title('Immagine in Scala di Grigi');

%% Visualizzo immagine in B/N
subplot(3,1,3)
imshow(imgBW)
title('Immagine in Bianco e Nero');

%% Visualizzo immagine senza artefatti
figure
subplot(3,1,1)
imshow(imgBW2)
title('Dopo Area Opening');

%% Visualizzo immagine dopo operatori morfologici
subplot(3,1,2)
imshow(imgBW3)
title('Dopo Dilataizione ed Erosione');

%% Visualizzo immagine Pre-Processata
subplot(3,1,3)
imshow(imgBW4)
title('Dopo Eliminazione dei Disturbi/Fori');

%% Visualizzo la Trasformata di Radon
figure
imagesc(theta, xp, R); colormap();
xlabel('\theta (degrees)');
ylabel('x^{\prime} (pixels from center)');
title('Trasfomata di Radon: R_{\theta} (x^{\prime})');
colorbar
%% Grafico delle Diagonali
figure
imshow(imgBW4)
hold on
boundary = B{1};
for i = 2:length(B)
    if(size(B{1},1) > size(boundary, 1))
        boundary = B{1};
    end
end

plot(boundary(:,2), boundary(:,1), 'b', 'LineWidth', 2);
title('Identificazione Oggetto');
hold on
plot(center(1) + offset1*cosd(theta1), center(2) - offset1*sind(theta1), 'r*', 'MarkerSize', 10);
plot(center(1) + offset2*cosd(theta2), center(2) - offset2*sind(theta2), 'g*', 'MarkerSize', 10);
plot(center(1) + x_bc, center(2) - y_bc, 'bo', 'MarkerSize', 10);
diag1Plot = center(2) - tand(theta1+90) * (x - center(1) - offset1*cosd(theta1)) - offset1*sind(theta1);
diag2Plot = center(2) - tand(theta2+90) * (x - center(1) - offset2*cosd(theta2)) - offset2*sind(theta2);
lineOrient = center(2) - tand(orient) * (x - center(1) - x_bc) - y_bc;
fplot(diag1Plot, 'r', 'LineWidth', 1.5);
fplot(diag2Plot, 'g', 'LineWidth', 1.5);
fplot(lineOrient, 'y', 'LineWidth', 2);

%% Report dell'Identificazione
fprintf("Area: %f, Perimetro: %f\n", objArea, objPerim);
fprintf("Forma dell'Oggetto: %s\n", objShape);
fprintf("Baricentro dell'Oggetto: X[bc] = %f, Y[bc] = %f\n", x_bc, y_bc);
fprintf("Orientamento dell'Oggetto: %f°\n", orient);
fprintf("Tempo di Processamento: %f ms\n", elapseTime);
```