Task Spaces

M Toussaint

February 29, 2016

1 Purpose

Task spaces are defined by a mapping $\phi:\ q \to y$ from the joint state $q \in \mathbb{R}^n$ to a task space $y \in \mathbb{R}^m$. They are central in designing motion and manipulation, both, in the context of trajectory optimization as well as in specifying position/force/impedance controllers:

For **trajectory optimization**, cost functions are defined by costs or constraints in task spaces. Given a single task space ϕ, we may define

- costs $\|\phi(q)\|^2$,
- an inequality constraint $\phi(q) \leq 0$ (element-wise inequality),
- an equality constraint $\phi(q) = 0$.

All three assume that the 'target' is at zero. For convenience, the code allows to specify an additional linear transform $\tilde{\phi}(q) \leftarrow \rho(\phi(q) - y_{\text{ref}})$, defined by a target reference y_{ref} and a scaling ρ. In KOMO, costs and constraints can also be defined on $k+1$-tuples of consecutive states in task space, allowing to have cost and constraints on task space velocities or accelerations. Trajectory optimization problems are defined by many such costs/constraints in various task spaces at various time steps.

For simple **feedback control**, in each task space we may have

- a desired linear acceleration behavior in the task space
- a desired force or force constraint (upper bound) in the task space
- a desired impedance around a reference

All of these can be fused to a joint-level force-feedback controller (details, see controller docu). On the higher level, the control mode is specified by defining multiple task spaces and the desired behaviors in these. (The activity of such tasks (on the symbolic level) is controlled by the RelationalMachine, see its docu.)

In both cases, defining task spaces is the core.

2 Basic notation

We follow the notation in the robotics lecture slides. We enumerate all bodies by $i \in \mathcal{B}$. We typically use $v, w \in \mathbb{R}^3$ to denote vectors attached to bodies. $T_{W\to i}$ is the 4-by-4 homogeneous transform from world frame to the frame of shape i, and $R_{W\to i}$ is its rotation matrix only. In the text (not in equations) we sometimes write

$$(i + v)$$

where i denotes a body and $v \in \mathbb{R}^3$ a relative 3D-vector. The rigorous notation for this would be $T_{W\to i}v$, which is the position of i plus the relative displacement v.

To numerically evaluate kinematics we assume that, for a certain joint configuration q, the positions p_k and axes a_k of all joints $k \in JJ$ have been precomputed. The boolean expression

$$[k \prec i]$$

iff joint k is "below" body i in the kinematic tree, that is, joint k is between root and body i and therefore moves it.

Sometimes we write $J_{\cdot k} = ...$, which means that the kth column of J is defined as given.

Let's first define

$$(A_i)_{\cdot k} = [k \prec i][i \text{ rotational}]\ a_k \qquad \text{axes matrix below } i \tag{1}$$

This matrix contains all rotational axes below i as columns and will turn out convenient, because it captures all axes that make i move. Many Jacobians can easily be described using A_i. Analogously we define

$$(T_i)_{\cdot k} = [k \prec i][i \text{ prismatic}]\ a_k \qquad \text{prism matrix below } i \tag{2}$$

This captures all prismatic joints. Note the following relation to Featherstone's notation: In his notation, $h_k \in \mathbb{R}^6$ denotes, for very joint k, how much the joints contributes to rotation and translation, expressed in the link frame. The two matrices A_i and T_i together express the same, but in world coordinates. While typically axes have unit length (and entries of h are zeros or ones), this is not necessary in general, allowing for arbitrary scaling of joint configurations q with these axis (e.g., using degree instead of radial units).

For convenience, for a matrix of 3D columns $A \in \mathbb{R}^{n\times 3}$, we write

$$B = A \times p \iff B_{\cdot k} = A_{\cdot k} \times p$$

which is the column-wise cross product. Also, we define

$$(\hat{A}_i)_{\cdot k} = [k \prec i][i \text{ rotational}]\ a_k \times p_k \qquad \text{axes position matrix below } i \tag{3}$$

which could also be written as $\hat{A}_i = A_i \times P$ if P contains all axes positions.

3 Task Spaces

Position

$$\phi^{\mathsf{p}}_{iv}(q) = T_{W\to i}\ v \qquad \text{position of } (i+v) \tag{4}$$

$$J^{\mathsf{p}}_{iv}(q)_{\cdot k} = [k \prec i]\ a_k \times (\phi^{\mathsf{p}}_{iv}(q) - p_k) \tag{5}$$

$$J^{\mathsf{p}}_{iv}(q) = A_i \times \phi^{\mathsf{p}}_{iv}(q) - \hat{A}_i \tag{6}$$

$$\phi^{\mathsf{p}}_{iv-jw}(q) = \phi^{\mathsf{p}}_{iv} - \phi^{\mathsf{p}}_{jw} \qquad \text{position difference} \tag{7}$$

$$J^{\mathsf{p}}_{iv-jw}(q) = J^{\mathsf{p}}_{iv} - J^{\mathsf{p}}_{jw} \tag{8}$$

$$\phi^{\mathsf{p}}_{iv|jw}(q) = R_j^{\text{-}1}(\phi^{\mathsf{p}}_{iv} - \phi^{\mathsf{p}}_{jw}) \qquad \text{relative position} \tag{9}$$

$$J^{\mathsf{p}}_{iv|jw}(q) = R_j^{\text{-}1}[J^{\mathsf{p}}_{iv} - J^{\mathsf{p}}_{jw} - A_j \times (\phi^{\mathsf{p}}_{iv} - \phi^{\mathsf{p}}_{jw})] \tag{10}$$

Derivation: For $y = Rp$ the derivative w.r.t. a rotation around axis a is $y' = Rp' + R'p = Rp' + a \times Rp$. For $y = R^{\text{-}1}p$ the derivative is $y' = R^{\text{-}1}p' - R^{\text{-}1}(R')R^{\text{-}1}p = R^{\text{-}1}(p' - a \times p)$. (For details see http://ipvs.informatik.uni-stuttgart.de/mlr/marc/notes/3d-geometry.pdf)

Vector

$$\phi^{\mathsf{v}}_{iv}(q) = R_{W\to i}\, v \qquad \text{vector} \tag{11}$$
$$J^{\mathsf{v}}_{iv}(q) = A_i \times \phi^{\mathsf{v}}_{iv}(q) \tag{12}$$
$$\phi^{\mathsf{v}}_{iv-jw}(q) = \phi^{\mathsf{v}}_{iv} - \phi^{\mathsf{v}}_{jw} \qquad \text{vector difference} \tag{13}$$
$$J^{\mathsf{v}}_{iv-jw}(q) = J^{\mathsf{v}}_{iv} - J^{\mathsf{v}}_{jw} \tag{14}$$
$$\phi^{\mathsf{v}}_{iv|j}(q) = R_j^{-1}\, \phi^{\mathsf{v}}_{iv} \qquad \text{relative vector} \tag{15}$$
$$J^{\mathsf{v}}_{iv|j}(q) = R_j^{-1}\, [J^{\mathsf{v}}_{iv} - A_j \times \phi^{\mathsf{v}}_{iv}] \tag{16}$$

3.1 Quaternion

See equation (15) in the geometry notes for explaining the jacobian.

$$\phi^{\mathsf{q}}_i(q) = \text{quaternion}(R_{W\to i}) \qquad \text{quaternion} \in \mathbb{R}^4 \tag{17}$$
$$J^{\mathsf{q}}_i(q)_{\cdot k} = \begin{pmatrix} 0 \\ \frac{1}{2}(A_i)_{\cdot k} \end{pmatrix} \circ \phi^{\mathsf{q}}_i(q) \qquad J^{\mathsf{q}}_i(q) \in \mathbb{R}^{4\times n} \tag{18}$$
$$\phi^{\mathsf{q}}_{i-j}(q) = \phi^{\mathsf{q}}_i - \phi^{\mathsf{q}}_j \qquad \text{difference} \in \mathbb{R}^4 \tag{19}$$
$$J^{\mathsf{q}}_{i-j}(q) = J^{\mathsf{q}}_i - J^{\mathsf{q}}_j \tag{20}$$
$$\phi^{\mathsf{q}}_{i|j}(q) = (\phi^{\mathsf{q}}_j)^{-1} \circ \phi^{\mathsf{q}}_i \qquad \text{relative} \tag{21}$$
$$J^{\mathsf{q}}_{i|j}(q) = \text{not implemented} \tag{22}$$

A relative rotation can also be measured in terms of the 3D rotation vector. Lets define

$$w(r) = \frac{2\phi}{\sin(\phi)}\bar{r}\,, \quad \phi = \text{acos}(r_0)$$

as the rotation for a quaternion. We have

$$\phi^{\mathsf{w}}_{i|j}(q) = w(\phi^{\mathsf{q}}_j)^{-1} \circ \phi^{\mathsf{q}}_i) \qquad \text{relative rotation vector} \in \mathbb{R}^3 \tag{23}$$
$$J^{\mathsf{w}}_{i|j}(q) = A J^{\mathsf{q}}_i - J^{\mathsf{q}}_j \tag{24}$$
$$\tag{25}$$

3.2 Alignment

parameters: shape indices i, j, attached vectors v, w

$\phi^{\text{align}}_{iv|jw}(q) = (\phi^{\mathsf{v}}_{jw})^\top \phi^{\mathsf{v}}_{iv}$

$J^{\text{align}}_{iv|jw}(q) = (\phi^{\mathsf{v}}_{jw})^\top J^{\mathsf{v}}_{iv} + \phi^{\mathsf{v}\,\top}_{iv} J^{\mathsf{v}}_{jw}$

Note: $\phi^{\text{align}} = 1 \leftrightarrow$ align $\quad \phi^{\text{align}} = -1 \leftrightarrow$ anti-align $\quad \phi^{\text{align}} = 0 \leftrightarrow$ orthog.

3.3 Gaze

2D orthogonality measure of object relative to camera plane

parameters: eye index i with offset v; target index j with offset w

$$\phi^{\text{gaze}}_{iv,jw}(q) = \begin{pmatrix} \phi^{\mathsf{v}\,\top}_{i,e_x}(\phi^{\mathsf{p}}_{jw} - \phi^{\mathsf{p}}_{iv}) \\ \phi^{\mathsf{v}\,\top}_{i,e_y}(\phi^{\mathsf{p}}_{jw} - \phi^{\mathsf{p}}_{iv}) \end{pmatrix} \quad \in \mathbb{R}^2 \tag{26}$$

Here $e_x = (1,0,0)$ and $e_y = (0,1,0)$ are the camera plane axes.

Jacobians straight-forward

3.4 qItself

$\phi^{\text{qItself}}_{iv,jw}(q) = q$

3.5 Joint limits measure

parameters: joint limits $q_{\text{low}}, q_{\text{hi}}$, margin m

$\phi_{\text{limits}}(q) = \frac{1}{m}\sum_{i=1}^{n}[m - q_i + q_{\text{low}}]^+ + [m + q_i - q_{\text{hi}}]^+$

$J_{\text{limits}}(q)_{1,i} = -\frac{1}{m}[m - q_i + q_{\text{low}} > 0] + \frac{1}{m}[m + q_i - q_{\text{hi}} > 0]$

$[x]^+ = x > 0?x : 0 \qquad [\cdots]$: indicator function

3.6 Collision limits measure

parameters: margin m

$\phi_{\text{col}}(q) = \frac{1}{m}\sum_{k=1}^{K}[m - |p_k^a - p_k^b|]^+$

$J_{\text{col}}(q) = \frac{1}{m}\sum_{k=1}^{K}[m - |p_k^a - p_k^b| > 0](-J^{\mathsf{p}}_{p_k^a} + J^{\mathsf{p}}_{p_k^b})^\top \frac{p_k^a - p_k^b}{|p_k^a - p_k^b|}$

A collision detection engine returns a set $\{(a, b, p^a, p^b)_{k=1}^K\}$ of potential collisions between shape a_k and b_k, with nearest points p_k^a on a and p_k^b on b.

3.7 Shape distance measures (using SWIFT)

- allPTMT, //phi=sum over all proxies (as is standard)
- listedVsListedPTMT, //phi=sum over all proxies between listed shapes
- allVsListedPTMT, //phi=sum over all proxies against listed shapes
- allExceptListedPTMT, //as above, but excluding listed shapes
- bipartitePTMT, //sum over proxies between the two sets of shapes (shapes, shapes2)
- pairsPTMT, //sum over proxies of explicitly listed pairs (shapes is n-times-2)
- allExceptPairsPTMT, //sum excluding these pairs
- vectorPTMT //vector of all pair proxies (this is the only case where dim(phi)¿1)

3.8 GJK pairwise shape distance (including negative)

3.9 Plane distance

4 Application

Just get a glimpse on how task space definitions are used to script motions, here is a script of a little PR2 dance. (The 'logic' below the script implements kind of macros – that's part of the RAP.) (`wheels` is the same as qItself, but refers only to the 3 base coordinates)

```
cleanAll #this only declares a novel symbol...

Script {
  (FollowReferenceActivity wheels){ type=wheels, target=[0, .3, .2], PD=[.5, .9, .5, 10.]}
  (MyTask endeffR){ type=pos, ref2=base_footprint, target=[.2, -.5, 1.3], PD=[.5, .9, .5, 10.]}
  (MyTask endeffL){ type=pos, ref2=base_footprint, target=[.2, +.5, 1.3], PD=[.5, .9, .5, 10.]}
  { (conv FollowReferenceActivity wheels)  (conv MyTask endeffR) }  #this waits for convergence of activities
  (cleanAll) #this switches off the current activities
  (cleanAll)! #this switches off the switching-off

  (FollowReferenceActivity wheels){ type=wheels, target=[0, -.3, -.2], PD=[.5, .9, .5, 10.]}
  (MyTask endeffR){ type=pos, ref2=base_footprint, target=[.7, -.2, .7], PD=[.5, .9, .5, 10.]}
  (MyTask endeffL){ type=pos, ref2=base_footprint, target=[.7, +.2, .7], PD=[.5, .9, .5, 10.]}
  { (conv FollowReferenceActivity wheels)  (conv MyTask endeffL) }
  (cleanAll)
```

```
  (cleanAll)!

  (FollowReferenceActivity wheels){ type=wheels, target=[0, .3, .2], PD=[.5, .9, .5, 10.]}
  (MyTask endeffR){ type=pos, ref2=base_footprint, target=[.2, -.5, 1.3], PD=[.5, .9, .5, 10.]}
  (MyTask endeffL){ type=pos, ref2=base_footprint, target=[.2, +.5, 1.3], PD=[.5, .9, .5, 10.]}
  { (conv MyTask endeffL) }
  (cleanAll)
  (cleanAll)!

  (FollowReferenceActivity wheels){ type=wheels, target=[0, 0, 0], PD=[.5, .9, .5, 10.]}
  (HomingActivity)
  { (conv HomingActivity) (conv FollowReferenceActivity wheels) }
}


Rule {
    X, Y,
    { (cleanAll) (conv X Y) }
    { (conv X Y)! }
}

Rule {
    X,
    { (cleanAll) (MyTask X) }
    { (MyTask X)! }
}

Rule {
    X,
    { (cleanAll) (FollowReferenceActivity X) }
    { (FollowReferenceActivity X)! }
}
```