Reference Notes for our PR2 Controller

M Toussaint

May 14, 2015

Contents

The aim of this document

- Agree on what we talk about!

1 Slow and fast control loop

There are two nested control loops:

In the slow loop ($\sim$ 50Hz, non-strict, non-real-time) the controller has full access to the results of pre-computed optimizations, full models of the robots kinematics (dynamics?) and potentially delayed information on the robot (current pose, forces, contacts, etc). The slow loop may realize computationally complex things, e.g., operational space control, re-adaptation of a plan (phase adaptation, recalibration of task maps), model predictive control, online planning, etc.

The fast loop is 1kHz, strictly and real-time. It has direct access to the current robot state q (needs to compute $\dot{q}$ from filtered differentiation of q) as well as the current readouts of the F/T sensors u_{ft}. **We constrain the fast controller to be a linear regulator in these observables and their integral:**

$$e \leftarrow \gamma e + (f^* - J_{\text{ft}}^\dagger u_{\text{ft}}) \quad \text{or} \quad \dot{e} = (f^* - J_{\text{ft}}^\dagger u_{\text{ft}}) + (1-\gamma)e \tag{1}$$

$$u = u_0 + k_p^{\text{base}} \cdot K_p(q^* - q) + k_d^{\text{base}} \cdot K_d(\dot{q}^* - \dot{q}) + K_I e \; . \tag{2}$$

This a (redundant) parameterization of a regulator linear in $(q, \dot{q}, e)$. We choose this parameterization because $q^*, \dot{q}^*, f^*$ can be interpreted as "references". But actually, we could just drop them (absorb them in u_0) without loosing generality. In addition to this, the fast loop respects control limits by clipping $u \leftarrow \texttt{clip}(u, -u_{max}, u_{max}$ element-wise. u_{max} is a constant set in configuration files, not a fluent. `[TODO: Additional mechanisms should also in the fast loop guarantee velocity and joint limits.]`

The parameter vectors k_p^{base} and k_d^{base} are constants set in the PR2 configuration files. They are hand-tuned so that setting $K_p = K_d = I$ leads to acceptable (rather low gain) behavior. The $\cdot$ denotes an element-wise product.

About the integral term: $J_{\text{ft}}^\dagger$ allows us to linearly project the sensor signals to any other space in which we have a target f^* and integrate the error.

$K_p, K_d, K_I, J_{\text{ft}}^\dagger$ are arbitrary matrices; u_0 an arbitrary control bias. Therefore, the **control mode** of the fast loop is determined by the tuple

$$M = (q^*, \dot{q}^*, f^*, u_0, K_p, K_d, K_I, J_{\text{ft}}^\dagger, \gamma) \; . \tag{3}$$

This is the message that the slow loop needs to send to the fast loop – the slow loop can change the control mode at any time.

Inversely, the fast loop passes the message

$$(q, \dot{q}, f, u) \tag{4}$$

to the slow loop, giving it information on the true current state $(q, \dot{q})$, sensor readings f, and computed controls u.

The core question therefore is how the slow loop computes the message M to realize the desired control behaviors. The list of basic desired control behaviors is:

1. Follow a pre-computed trajectory $(q_{0:T}, \tau)$, where τ is the time resolution

2. Follow the position-reference that is online computed by a operational space (or inverse kinematics) controller; the K_p should such that P-gains can be set/added/removed along *endeffector* spaces rather than only uniformly configuration space

3. Establish a contact

4. Stabalize a contact force

5. Limit F/T (to avoid breaking a handle)

6. Sliding (moving tangentially) on a surface (or along a DOF like an opening door) which is perceived via the F/T signal

2 Operational Space Control: Computing gains by projecting operational space gains

The appendix B derived the necessary equations in all generality. In practise, it is sufficient to modify the K_p only, using the Jacobian of the desired task space. In equation (41) we have

$$\bar{K}_p = A^{-1} J^\top C K_p J \ , \quad A = H + J^\top C J \ , \tag{5}$$

where we assumed $M = \mathbf{I}$ (quasi-dynamic model) and no other tasks. Further, assuming $C = c$ and $K_p = k$ are scalars we have

$$\bar{K}_p = k(H/c + J^\top J)^{-1} J^\top J \ . \tag{6}$$

I actually tested just $kJ^\top J$.

TODO: Let @FeedbackController@ really compute these projected PD behaviors, instead of only $q^*, \dot{q}^*$! Then all of this is automatic!

3 Controlling the F/T signal—the *sensed* force

3.1 Preliminaries: Understanding force transmission

The following law of force propagation is well known,

$$u = J^\top f \tag{7}$$

where f is a force in the endeffector (e.g., the negative of its gravity load), J the position Jacobian of the endeffector, and u are the torques "perceived" in each of the joints due to the force f. This law is correct only under the assumption that nothing moves. Inversely, this law is typically used to compensate forces: Assume you have a load on an endeffector, gravity pulls it down. The gravity force pulling the load down propagates to torques u in each joint – if you want to compensate this torque the motors need to create the reative torque.

The same also holds for force-torque $f \in \mathbb{R}^6$, where the Jacobian is the stacking of the position and the axial Jacobian.

Typically, f is lower-dimensional than u. So, actually, there should be many u that generate a desired f^*? What is the optimal one? Well, assume $f^* = 0$ for a moment. Then, any choice of u will accelerate the robot (assuming gravity compensation). The only choice to generate $f^* = 0$ and not to accelerate the robot is $u = 0$. Equally, the only choice to generate any f^* without accelerating the robot is $u = J^\top f^*$.

When we include system dynamics in the equation, we have the general

$$u = M\ddot{q} + h + J^\top f \ . \tag{8}$$

where M (the inertia matrix) and h (the coreolis and gravity forces) depend on $(q, \dot{q})$. One way to read this equation is: the torques you "feel" in the joints are the reactive torques of the robot's inertia (that derive the acceleration) plus the torque you feel from the endeff force f.

3.2 Controlling the direct F/T signal with a fixated endeff

Consider the following exercise: Fix the endeffector rigidly, e.g. to a table with a clamp (Schraubzwinge). Write a controller that generates any desired f^* in the F/T sensor with the least effort, and stably, and staying close to a desired homing posture.

If we unrealistically assume that our model is correct then the solution simply is (8); for $\ddot{q} = 0$ and a gravity-compensated robot just (7); where

$$J = J_{\text{ft}} \,, \tag{9}$$

which is the position and axial Jacobian of the F/T sensor w.r.t. q.

However, this equation **does not use any F/T sensor feedback** to generate the desired F/T signal. This cannot work well in practise. We can resolve this with an I-controller on the F/T signal error.

$$e = \int_t dt[f^* - f] \tag{10}$$

$$u = J^\top \alpha e \,. \tag{11}$$

The α here has the meaning of an exponential decay of the signal error—which we can show assuming the perfect model. Under perfect model assumption, the F/T sensor measures

$$f = J^\dagger u \,, \quad J^\dagger J^\top \equiv \mathbf{I} \,, \quad J^\dagger = (JJ^\top)^{-1} J \tag{12}$$

$$\text{Note: } J^\dagger u = J^\dagger J^\top f = (JJ^\top)^{-1} JJ^\top f = f \tag{13}$$

$$\tag{14}$$

Note that $JJ^\top$ is a $d \times d$-matrix and invertible and $J^\dagger$ the appropriate left-pseudo-inverse of $J^\top$. Inserting this perfect-model measurement in the control law (10) we get

$$\dot{e} = f^* - J^\dagger u \tag{15}$$

$$\dot{u} = J^\top \alpha (f^* - J^\dagger u) = \alpha J^\top f^* - \alpha \underbrace{J^\top J^\dagger} \mathbf{I} u = \alpha (J^\top f^* - u) \,. \tag{16}$$

Here, $J^\top J^\dagger = J^\top (JJ^\top)^{-1} J^\top$ is actually the projection that projects any joint torques u into the space that directly relates to endeffector forces and not to accelerations. However, if the u was chosen by some law $J^\top f$, then u will always lie within this projection (will never lead to accelerations of the robot), and therefore it actually is the identity matrix.

Now, the above states that $\dot{u} = \alpha(J^\top f^* - u)$, which says that u exponentially approaches the perfect-model correct torque $J^\top f^*$, which a decay rate α. Therefore, α can be considered a decay rate.

Open: What if we have a $\ddot{q}$ as well? Two possibilities: 1) Reiterate the above reasoning with $\ddot{q}$. 2) Just add the signals.

3.3 Control the indirectly sensed contact force of endeff

Exercise: We have the F/T sensor, but attached to it a hand and a contact point with some relative transformation to the F/T sensor. This point is in contact with a table. What we want to control is the force between point and table, which is just a 1D thing.

This is best addressed by thinking of the F/T sensor as if it was a 6D joint (like a ball joint). If we have a force f at some endeffector then we "feel" this force in all joints of the robot as $u = J^\top f$. This includes the F/T sensor joints! So the Jacobian of the endeffector

variable (be it 1D or 3D) w.r.t. the sensor pseudo-ball-joint exactly gives the measurement equation. Let's denote this Jacobian as

$$J_{\text{ft}} \in \mathbb{R}^{d\times 6} \,, \tag{17}$$

where $d = 1$ if it is only the distance to the table, or $d = 3$ if it is all forces. Further, let's denote by

$$J \in \mathbb{R}^{d\times n} \tag{18}$$

the Jacobian w.r.t. all the real robot joints.

As above, the *perceived* endeffector force (this time perceived by the F/T sensor) is

$$u_{\text{ft}} = J_{\text{ft}}^\top f \quad \Rightarrow \quad f = J_{\text{ft}}^\dagger u_{\text{ft}} \,, \tag{19}$$

where $u_{\text{ft}} \in \mathbb{R}^6$ is the F/T signal. Again we may use an I-controller to correct for the error between desired endeffector force f^* and perceived one:

$$\dot{e} = f^* - f = f^* - J_{\text{ft}}^\dagger u_{\text{ft}} \tag{20}$$

$$u = J^\top \alpha e \,. \tag{21}$$

Note that the last equation generates joint torques proportional to the normal endeffector Jacobian J because e is an error in endeffector force space (not F/T signal space).

This fits to our contoller setup by

$$J_{\text{ft}}^\dagger \leftarrow J_{\text{ft}}^\dagger \,, \quad f^* \leftarrow f^* \,, \quad \gamma \leftarrow 1 \,, \quad K_I \leftarrow \alpha J^\top \,. \tag{22}$$

When force control is turned off, we need to remember to set $\gamma = 0, e = 0$ to ensure that next time it is turned on again it doesn't blow.

Open: What happens for $\gamma < 1$? Is this equivalent to $\alpha < 1$? Perhaps not. ($\sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma}$)

3.4 q-control under force constraints

Assume we have a P(I)D controller on q—typically a PID in some task space that has been projected to joint space. We would like to execute that desired reference behavior but subject to constraints on the sensed endeffector force

$$f_{lo} \le J_{\text{ft}}^\dagger f \le f_{hi} \,. \tag{23}$$

These are $2d$ constraints.

As with lagrange parameters, we can simply activate the constraints when violated: When one of the components violates the constraint, control the force to be exactly $f_{lo|hi}$. For the latter, use the f-error-integral method as above. This should eventually have higher priority to any other gains (keep other I-gains limited!).

3.5 I-gains on position?

4 Technical Details and Issues

4.1 Ctrl-Message documentation

One message type for setting the control mode AND feedback from the controller.

Setting the control model: $(q^*, \dot{q}^*, f^*, u_0, K_p, K_d, K_f)$. Can be set any time.

Feedback from the controller: $(q, \dot{q}, f, u)$. Published with 1kHz.

4.2 Filtering of the differentiation of q

[Peter: please fill in]

5 Enforcing control, velocity, joint and force limits

Enforcing control limits is really simple: Just clip the computed u.

Enforcing velocity limits turned out difficult: The velocity signal is so noisy, a direct feedback coupling was bad. Also, the IF-case of velocity-limit-violation turned off and on quickly and introduced even more noise (rattling motors...)

I have ignored limits totally so far – should be handled (as collisions) in the slow loop.

FORCE LIMITS! Not idea how to handle this.

Maybe a route to a more principled approach to all of these: Take the Augmented Lagrangian way to handle constraints as a template: First associate only a soft squared penalty with margin penetration. Then compute/update the respective dual parameters that push you out of the margin.

A Reference: General OPEN-LOOP ideal contact force controller

A.1 General case

This controller is sort-of open loop! It does not take into account any F/T feedback. What is receives as a specification is $\ddot{y}^*$ (desired endeff accel) and λ^* (desired contact force); as well as models of the maps ϕ, J_ϕ, g, J_g. We discuss below how this can properly be made a feedback regulator. We write the problem as a general constraint problem

$$\min_{u,\ddot{q},\lambda} \quad \|u - a\|_H^2 \tag{24}$$

$$\text{s.t.} \quad u = M\ddot{q} + h + J_g^\top \lambda \tag{25}$$

$$J_\phi \ddot{q} = c \tag{26}$$

$$\lambda = \lambda^* \tag{27}$$

$$J_g \ddot{q} = b \tag{28}$$

Notes:

- The role of a becomes clearer when we treat the blue constraint below
- The 2nd constraint relates to an arbitrary task map $\phi : \ q \mapsto y$ with Jacobian J_ϕ, $c = \ddot{y}^* - \dot{J}_\phi \dot{q}$ and some desired task space acceleration $\ddot{y}^*$.
- We have a set of functions $g : \ q \to \mathbb{R}^m$ with Jacobian J_g which play the role of inequality constraints
- The 3rd constraint captures desired contact forces with the constraints
- The blue constraint expresses that a) assuming the contact is active there must not be acceleration w.r.t. g or b) if the contact is not active we might want to control the acceleration towards it (to make it active). (This constrains the dynamics. Without this constraint the dynamics could assume that the contact forces λ are generated while the endeff is moving (e.g., by strange external forces). This constraint makes it consistent.)

To derive closed form solutions, each of these equality constraints can be handled in two ways: relax it to become a squared penalty (and then potentially taking the infinite precision limit); or resolve it.

We resolve the 1st and 3rd constraint, and relax the 2nd and 4th to later take the limit. The solution is

$$\mathbf{h} := h + J_g^\top \lambda^* \tag{29}$$
$$f(\ddot{q}) = \|M\ddot{q} - (a - \mathbf{h})\|_H^2 + \|J_g\ddot{q} - b\|_B^2 + \|J_\phi\ddot{q} - c\|_C^2 \tag{30}$$
$$\ddot{q}^* = (M^\top H M + J_g^\top B J_g + J_\phi^\top C J_\phi)^{-1}[M^\top H(a - \mathbf{h}) + J_g^\top B b + J_\phi^\top C c] \tag{31}$$

The limit $B \to \infty$:

$$\ddot{q}^* = (X + J_g^\top B J_g)^{-1}[J_g^\top B b + x] \tag{32}$$
$$= J_{g\,XB}^{\#} b + (\mathbf{I} - J_{g\,XB}^{\#} J_g)\,(X^{-1}x) \tag{33}$$

And note that $(X^{-1}x)$ is the solution to only having the other terms. Given $\ddot{q}^*$, the optimal control is computed as $u = M\ddot{q} + h + J_g^\top \lambda^*$. We still did not take the limit of the C-term (endeffector position control). We could using the hierarchical limit case.

B Reference: Pullback of operational space linear controllers

The above assumes that at any instance in time we want a certain task-space acceleration $\ddot{y}^*$ and translates this to an optimal joint control in that instant in time. If we want to implement a certain feedback behavior in the task space, that is, we have a desired feedback control law $\pi : y, \dot{y} \mapsto \ddot{y}$, we can evaluate π at every point in time and project to operational space control.

$$\ddot{y} = \ddot{\phi}(q) = \ddot{(}Jq) = \dot{(}\dot{J}q + J\dot{q}) = 2\dot{J}\dot{q} + J\ddot{q} \tag{34}$$
$$\ddot{y}^* = K_p y + K_d \dot{y} + k \tag{35}$$
$$J\ddot{q} \stackrel{!}{=} c = \ddot{y}^* - 2\dot{J}\dot{q} = K_p y + K_d\dot{y} + k - 2\dot{J}\dot{q} \tag{36}$$
$$\approx K_p(J(q - q_0) + \phi(q_0)) + K_d J\dot{q} + k \tag{37}$$
$$= K_p J q + K_d J\dot{q} + k' \,, \quad k' = k + K_p(\phi(q_0) - Jq_0) \tag{38}$$
$$\ddot{q}^* = A^{-1}[... + J^\top C c] = A^{-1}[... + J^\top C(K_p J q + K_d J\dot{q} + k')] \tag{39}$$
$$= A^{-1}[...] + A^{-1}J^\top C K_p J q + A^{-1}J^\top C K_d J\dot{q} + A^{-1}J^\top C k' \tag{40}$$
$$= \bar{K}_p q + \bar{K}_d\dot{q} + \bar{k} \,, \quad \bar{k} = A^{-1}[...] + A^{-1}J^\top C k' \,, \quad \bar{K}_p = A^{-1}J^\top C K_p J \,, \quad \bar{K}_d = A^{-1}J^\top C K_d J \tag{41}$$

C How to make this FEEDBACK?

W.r.t. y (endeff pos) it is clear how to make this feedback: We can impose a PD behavior on the endeffector

$$\ddot{y}^* = k_p(y^* - y) + k_d(\dot{y}^* - \dot{y})$$

and send this desired endeff accel to the general controller.

What about λ^*??