Stardust-R Early Stage Researcher 10 Assignment

Davide Torielli

July 12, 2019

Assignment Recap

- Space Mobile manipulator with 7-DOF arm
- Objective: following a desired end-effector trajectory (already "generated")
  - i.e. given, *desired*, pose and velocity for the End-Effector at each certain time-steps (approximately 0.15 seconds range)
- No external forces/disturbances
- Four modalities:
  - Free-Floating
  - Free-Flying
  - Rotation Free-Flying
  - Free-Floating & Null-Space Exploitation

Mathematical models

Kinematic Control Layer

Given the desired EE pose $\bar{\boldsymbol{x}} = [\bar{\phi}\ \bar{\theta}\ \bar{\psi}\ \bar{x}\ \bar{y}\ \bar{z}]^T$ and velocity $\dot{\bar{\boldsymbol{x}}} = [\bar{\omega}_x\ \bar{\omega}_y\ \bar{\omega}_z\ \dot{\bar{x}}\ \dot{\bar{y}}\ \dot{\bar{z}}]^T$ at each time step Δt, the Kinematic Layer provides *desired* joint-space velocities for each joint : $\dot{\bar{\boldsymbol{q}}} = [\dot{\bar{q}}_1\ \dot{\bar{q}}_2\ \dot{\bar{q}}_3\ \dot{\bar{q}}_4\ \dot{\bar{q}}_5\ \dot{\bar{q}}_6\ \dot{\bar{q}}_7]^T$

For the first three modalities, this is done with Jacobian pseudoinverse, computed with SVD (Singular Value Decomposition).
In the forth (*Null-Space Exploitation*), a TPIK approach is used (details after).

These notations are for the manipulator, but for the vehicle in the Free-Flying cases considerations are similar.

Mathematical models

Kinematic Control Layer for the arm

$$\dot{\tilde{\boldsymbol{q}}} = \boldsymbol{J}^{\#}\left(k_p\tilde{\boldsymbol{x}} + k_v\dot{\tilde{\boldsymbol{x}}}\right)$$

- $\boldsymbol{J} = \boldsymbol{J}_{0/ee} - \boldsymbol{J}_{\boldsymbol{0}0/ee}\,\boldsymbol{H_0}^{-1}\boldsymbol{H_{0m}}$ The generalize Jacobian. The second part takes into account effects of joints on the vehicle *[Siciliano and Khatib, 2008]*
- $\boldsymbol{J}_{0/ee}$ (6×7) is the Manipulator Jacobian. It expresses the rate of change of EE velocity $\dot{\boldsymbol{x}}$ with respect to the joints velocity vector $\dot{\boldsymbol{q}}$. $\boldsymbol{J_0}$ is the analogous but with respect to the vehicle velocities $\dot{\boldsymbol{q}}_{\boldsymbol{0}}$
- $\boldsymbol{H_0}$ is the inertia matrix only for the vehicle part, which is always invertible because it is positive definite
- $\boldsymbol{H_{0m}}$ is the coupling inertia matrix which expresses how joints accelerations influence the forces acting on the vehicle

Mathematical models

Kinematic Control Layer for the arm (continued)

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\left(k_p \tilde{\boldsymbol{x}} + k_v \dot{\tilde{\boldsymbol{x}}}\right)$$

- $\#$ denotes the pseudoinverse operator
- $\tilde{\boldsymbol{x}}$ is the error between *desired* end-effector position and *real* one
  - Angular part computed as Misalignment Vector between the two rotation matrix *[Simetti, n.d.; Casalino, n.d.]*
  - Linear part is simply a difference: $\bar{\boldsymbol{x}}_l - \boldsymbol{x}_l$
- $\dot{\tilde{\boldsymbol{x}}} = \dot{\bar{\boldsymbol{x}}} - \dot{\boldsymbol{x}}$ is the error between *desired* end-effector velocity and *real* one
- k_p and k_v are positive gains

Mathematical models

Kinematic Control Layer for the vehicle

For the Free-Flying cases, we want the vehicle to stay in its original pose (i.e. inertial frame). So, we calculate *desired* velocity for the vehicle:

$$\bar{\dot{\boldsymbol{q}}}_{\mathbf{0}} = \boldsymbol{P}_{\mathbf{0}}{}^{\#} \left(k_p \tilde{\boldsymbol{x}}_{\mathbf{0}} \right)$$

For the Rotation Free-Flying modality only the angular part (the first three rows) is computed and considered.

- $\tilde{\boldsymbol{x}}_{\mathbf{0}}$ is the error between desired vehicle pose and real one
- k_p is a positive gain
- $\boldsymbol{P}_0$ (6×6) is the Map from vehicle velocities projected in inertial frame to vehicle velocities projected in vehicle frame:

$$\dot{\boldsymbol{x}}_{\mathbf{0}} = \boldsymbol{P}_{\mathbf{0}} \dot{\boldsymbol{q}}_{\mathbf{0}} \qquad \boldsymbol{P}_{\mathbf{0}} = \begin{bmatrix} \underset{3\times3}{\boldsymbol{J}_{\boldsymbol{k},\mathbf{0}}} & \underset{3\times3}{\mathbf{0}} \\ \underset{3\times3}{\mathbf{0}} & \underset{3\times3}{{}^{w}\boldsymbol{R}_{v}} \end{bmatrix} \qquad \boldsymbol{J}_{\boldsymbol{k},\mathbf{0}} \textit{ as in[Antonelli, 2013]}$$

Mathematical models

Dynamic Control Layer

Given the *desired* joint-space velocities for each joint $\dot{\boldsymbol{q}}$, the dynamic layer provides the torques for each joint : $\boldsymbol{\tau} = [\tau_1\ \tau_2\ \tau_3\ \tau_4\ \tau_5\ \tau_6\ \tau_7]^T$

This is done with a Computed Torque Scheme (see appendix for stability proof). The desired accelerations $\ddot{\boldsymbol{q}}$ are neglected and put to zero.

In the Free-Flying cases, an *augmented* Computed Torque Scheme also provides torques and forces for the vehicle.

Mathematical models

Dynamic Control Layer

$$\boldsymbol{\tau} = \boldsymbol{H}\,(\ddot{\bar{\boldsymbol{q}}} + k_d \dot{\tilde{\boldsymbol{q}}}) + \boldsymbol{C}\,\dot{\boldsymbol{q}} + \boldsymbol{D}$$

- $\boldsymbol{H}(\boldsymbol{q})$ $(N \times 7)$ is the Generalized Inertia Matrix
- $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ $(N \times 7)$ is the Convective Inertia Matrix that takes into account *coriolis* and *centrifugal* effects
- $\boldsymbol{D}(\boldsymbol{q})$ $(N \times 1)$ is a vector for gravitational effects, thus in this case is $\mathbf{0}$
- $\ddot{\bar{\boldsymbol{q}}}$ $(N \times 7)$ are the desired accelerations (as said, put to zero)
- $\dot{\tilde{\boldsymbol{q}}} = \dot{\bar{\boldsymbol{q}}} - \dot{\boldsymbol{q}}$ $(N \times 1)$ is the error between *desired* velocities (KCL output) and *real* velocities
- k_d is a positive gain (can be different for arm and vehicle parts in the Free-Flying cases)

with N = 7 for FFloating N = 6+7 for FFlying N=3+7 for Rot-FFlying

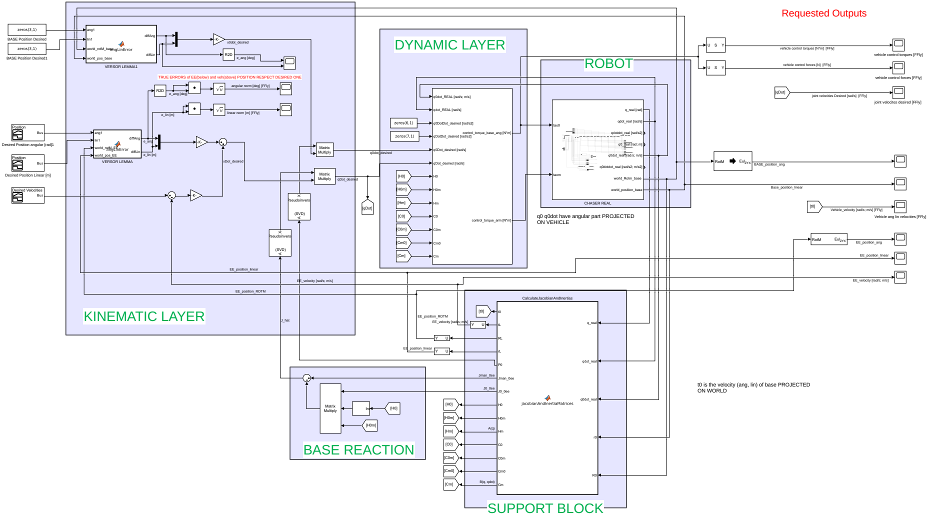
Assumptions

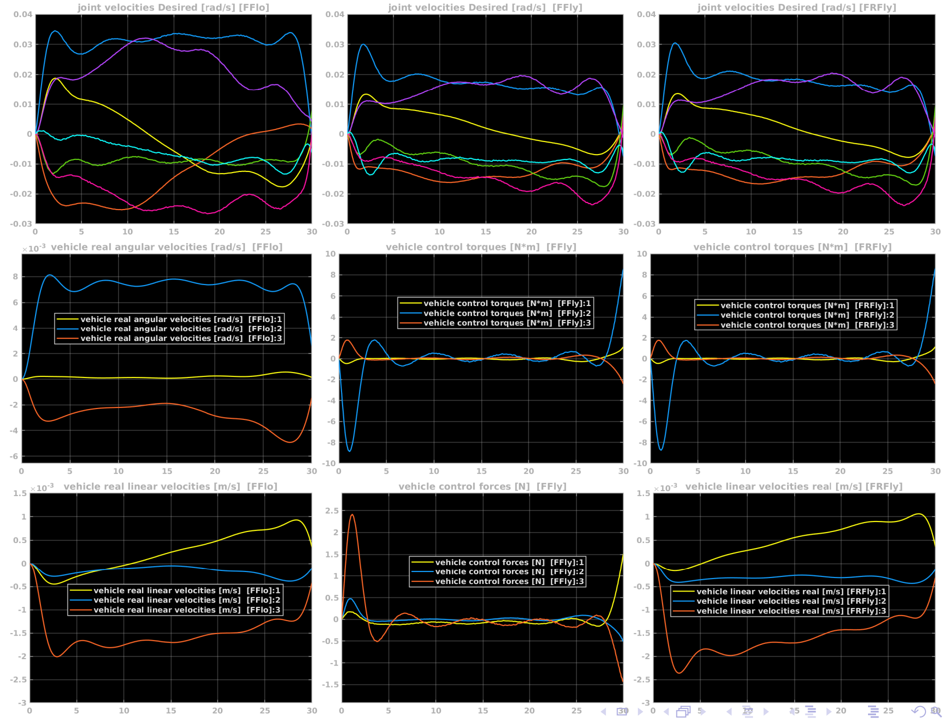
- The assignment ones (first page of slides)
- State of the joints $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and of the vehicle $\boldsymbol{q_0}, \dot{\boldsymbol{q}}_\mathbf{0}$ known
- Pose of the vehicle respect to the inertial frame (i.e. $\boldsymbol{x_0}$) always known
- Inertia Matrices $\boldsymbol{H}, \boldsymbol{C}$ known without uncertainties (even if the computed torque is good also when we have only rough approximations)
- Fully Actuated chaser (interesting only for Free-Flying cases)
- No other errors/disturbances

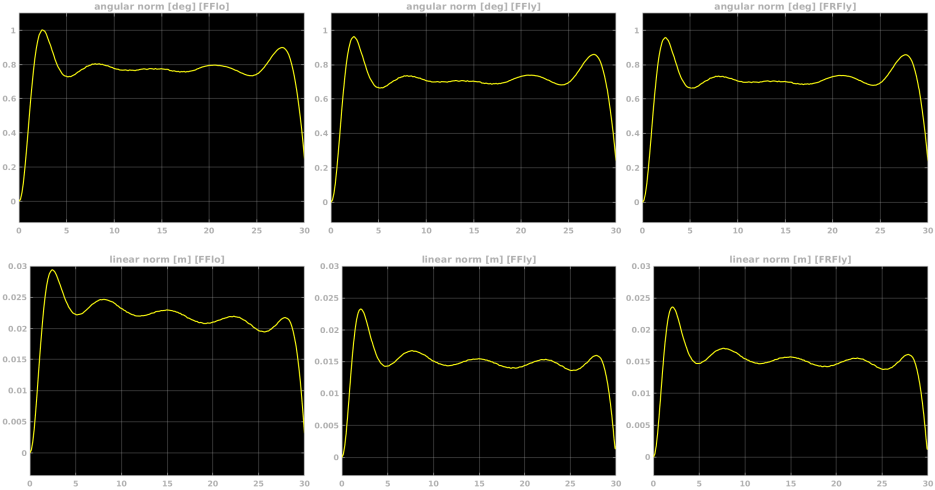
Tools Used

- Matlab & Simulink
- Simscape Multibody to simulate the robot (i.e. forward dynamics)
  - Given the torques for the joints (and for the vehicle), it provides vehicle and joints state
- SPART *[Virgili-Llop, n.d.]* a matlab toolkit to compute manipulator Jacobian and Inertia Matrices
- iCAT algorithm from my university to deal with TPIK (for modality 4) *[Simetti and Casalino, 2016]*

Free Flying mode




joint velocities Desired [rad/s] [FFlo]
joint velocities Desired [rad/s] [FFly]
joint velocities Desired [rad/s] [FRFly]
vehicle real angular velocities [rad/s] [FFlo]
vehicle real angular velocities [rad/s] [FFlo]:1
vehicle real angular velocities [rad/s] [FFlo]:2
vehicle real angular velocities [rad/s] [FFlo]:3
vehicle control torques [N*m] [FFly]
vehicle control torques [N*m] [FFly]:1
vehicle control torques [N*m] [FFly]:2
vehicle control torques [N*m] [FFly]:3
vehicle control torques [N*m] [FRFly]
vehicle control torques [N*m] [FRFly]:1
vehicle control torques [N*m] [FRFly]:2
vehicle control torques [N*m] [FRFly]:3
vehicle real linear velocities [m/s] [FFlo]
vehicle real linear velocities [m/s] [FFlo]:1
vehicle real linear velocities [m/s] [FFlo]:2
vehicle real linear velocities [m/s] [FFlo]:3
vehicle control forces [N] [FFly]
vehicle control forces [N] [FFly]:1
vehicle control forces [N] [FFly]:2
vehicle control forces [N] [FFly]:3
vehicle linear velocities real [m/s] [FRFly]
vehicle linear velocities real [m/s] [FRFly]:1
vehicle linear velocities real [m/s] [FRFly]:2
vehicle linear velocities real [m/s] [FRFly]:3

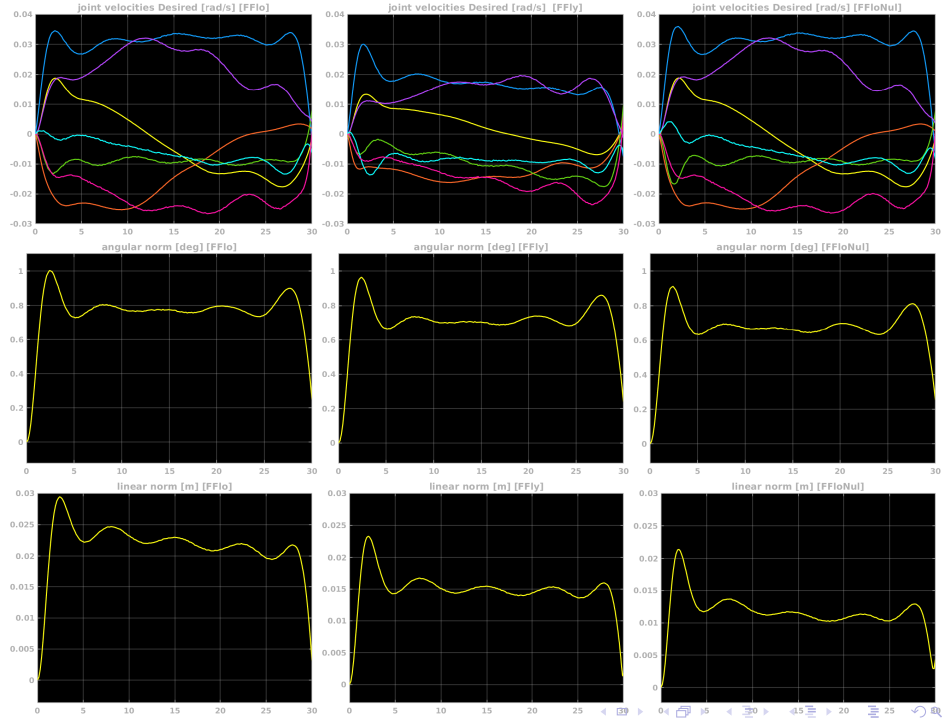

angular norm [deg] [FFlo]
angular norm [deg] [FFly]
angular norm [deg] [FRFly]
linear norm [m] [FFlo]
linear norm [m] [FFly]
linear norm [m] [FRFly]
0
0.2
0.4
0.6
0.8
1
0.005
0.01
0.015
0.02
0.025
0.03
5
10
15
20
25
30

Exploitation of Null Space

Task Priority Inverse Kinematic

- Objective: exploit arm redundancy to reduce effect of vehicle y-angular velocity r_y on the end-effector.
- Task Priority Inverse Kinematic. Following the real angular velocity r_y will be the higher priority task. In this way, the $\dot{\boldsymbol{q}}$ are calculated taking into account effects of r_y on end-effector

```
rhop = zeros(8,1); %control vector [q0Dot_y qDot1 ... qDot7]^T
Qp = eye(8); % matrix for null projections
JvehConstr = [1, zeros(1,7)]; %for effects of rhop on y-angular
    velocity
Jgoal = [J0_0ee(:,2), J];%for effects of rhop on EE velocity
[Qp, rhop] = iCAT_task(AvehConstr, JvehConstr, Qp, rhop,
    q0dotReal_y,  0.0001,   0.01, 10);
[Qp, rhop] = iCAT_task(Agoal, Jgoal, Qp, rhop, goalRef,  0.0001,
    0.01, 10);
qDotDesired = rhop(2:8); % take only commands for joints
```


joint velocities Desired [rad/s] [FFlo]
joint velocities Desired [rad/s] [FFly]
joint velocities Desired [rad/s] [FFloNul]
angular norm [deg] [FFlo]
angular norm [deg] [FFly]
angular norm [deg] [FFloNul]
linear norm [m] [FFlo]
linear norm [m] [FFly]
linear norm [m] [FFloNul]

Conclusions

- Among the three cases, Free-Flying is the best, but the energy consumed does not compensate this improvement
- Rotation Free-Flying can be a good compromise also because attitude control can be more energy efficient than whole pose control (reaction wheels most of the time versus thruster all the time)
- In general, improvement at kinematic level are necessary, results of the fourth case are a proof
- Real scenarios are much more complicated and each situation has to be studied specifically

Thank you for the attention!

Appendix A
Stability proof of computed torque for DCL

Given the dynamic model:
$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{D}(\boldsymbol{q}) = \boldsymbol{\tau}$
with $\boldsymbol{H}$ positive definite

Let the Lyapunov function:

$$V = \frac{1}{2}||\dot{\tilde{\boldsymbol{q}}}||^2 = \frac{1}{2}\dot{\tilde{\boldsymbol{q}}}^T\dot{\tilde{\boldsymbol{q}}} \qquad \dot{V} = \dot{\tilde{\boldsymbol{q}}}^T\ddot{\tilde{\boldsymbol{q}}} = \dot{\tilde{\boldsymbol{q}}}^T(\ddot{\bar{\boldsymbol{q}}} - \ddot{\boldsymbol{q}})$$

$$\dot{V} = \dot{\tilde{\boldsymbol{q}}}^T(\ddot{\bar{\boldsymbol{q}}} - \boldsymbol{H}^{-1}(\boldsymbol{\tau} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{D})) = \dot{\tilde{\boldsymbol{q}}}^T(\ddot{\bar{\boldsymbol{q}}} - \boldsymbol{H}^{-1}\boldsymbol{u})$$

$\boldsymbol{u}$ contains $\boldsymbol{\tau}$ which is arbitrary, so I impose $\boldsymbol{u}$ as:

$$\boldsymbol{u} = \boldsymbol{H}(\ddot{\bar{\boldsymbol{q}}} + k_d\dot{\tilde{\boldsymbol{q}}}) \qquad \text{so} \qquad \dot{\boldsymbol{V}} = -k_d\dot{\tilde{\boldsymbol{q}}}^T\dot{\tilde{\boldsymbol{q}}} = -2k_d\boldsymbol{V}$$

With the resultant negative definite $\dot{\boldsymbol{V}}$ the error $\dot{\tilde{\boldsymbol{q}}}$ converges.
So the wanted control torque is:

$$\boldsymbol{\tau} = \boldsymbol{u} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{D} = \boldsymbol{H}(\ddot{\bar{\boldsymbol{q}}} + k_d\dot{\tilde{\boldsymbol{q}}}) + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{D}$$

References

Antonelli, Gianluca (2013). "Underwater Robots". In: Springer International Publishing, p. 23. DOI: 10.1007/978-3-319-02877-4.

Casalino, Giuseppe. *Inverse Versor Lemma explained: extract from lecture notes (pdf in ref folder)*.

Siciliano, Bruno and Oussama Khatib (2008). "Springer Handbook of Robotics". In: Springer International Publishing, pp. 1406–1407. DOI: 10.1007/978-3-319-32552-1.

Simetti, Enrico. *Versor Lemma explained: extract from lecture notes (pdf in ref folder)*.

Simetti, Enrico and Giuseppe Casalino (2016). "A Novel Practical Technique to Integrate Inequality Control Objectives and Task Transitions in Priority Based Control". In: *Journal of Intelligent & Robotic Systems* 84. DOI: 10.1007/s10846-016-0368-6.

Virgili-Llop, Josep et al. *SPART: an open-source modeling and control toolkit for mobile-base robotic multibody systems with kinematic tree topologies*. https://github.com/NPS-SRL/SPART.