Lecture 7+8: Human-robot collaboration and Dynamic Motion Primitives

Victor Risager

February 29, 2024

1 Motivation

Traditional programming is getting out of fashion, because programmers are expensive labor. They become aparent in small to medium size enterprises. In addition higher level programming can facilitate customization of products, quick change overs and partial automation.

2 Franka Robot

Collaborative interface → has buttons on top of the wrist.

- Comes equipped with force sensors in each joint.
- Freedrive
- It has reflexes, which allows it to stop if it hits something, and it does not need to restart the entire program, because it does not have a real emergency stop on it.
- UR only measures the current that goes into the motors, which is low resolution relative to force sensors.

3 Collaborative Robots

The safety level is standardised.
Features:

- Joint torque sensors
- Lightweight design (a person can carry them)
- No pinch design. You can not get pinched in small gaps.
- Tactile skin
- Compliant actuators
- A collaborative robot will never be as stiff as a traditional robot, becuase of the compliant actuators.

Sensors:

- Admittance
- Impedance

Arms:

- KUKA iiwa (old, expensive)
- UR (RTD interface, offical ROS driver) (Aljaz has developped a realtime UR control loop running 500 Hz, Open source)
- Franka emika (20.000 eur, modern, smaller payload, ROS compatible)

4 Collaborative Industrial Systems

You have to define:

- Maximum workspace
- Restricted workspace (some overlab with human workspace (full speed robot))
- Operating workspace, it is a smaller workspace and the robot stops
- Collaborative workspace, concurrent tasks between human and robot.

You can however also run it in a normal workspace.

Mixed Workspaces
Where some of the workspace is collaborative, and some are fully autonomous, and not automated.

5 New Programming Techniques

Robot vendors have their own languages, which are propriotary.
You can use a camera, or other sensors to show the robot how to move.

- Makes teaching complex tasks simple.
- Non experts can quickly program it.
- Simpler to program certain kinds of tasks.
- Quick to reprogram
- $\vdots$

5.1 Demonstration modalities

1. Teleoperation/ manual guidance (kinestetic programming)
    - No mappings required
    - No external hardware required
    - Intuitive for the user.
    - Hard to control due to e.g. friction in the joints. $\rightarrow$ can be physically tirring.
    - Gravity compensation can be bad if the control loop is too slow, e.g. 125 Hz.
2. Sensors on the teacher
    - Exteroceptive sensors
    - $\vdots$
3. Shadowing (Do the same thing as the human)
    - No embodiment mapping, so you do not have to touch the robot
    - Demonstrated by the user
    - Direct transfer of the human skill. Can also be modified. (teleoperation)
    -
4. Observation (make the robot observe (camera))
    - Expensive sensors, but it is the most natural demonstration method.Works like teaching humans.

Record the following:

- Positions and orientation trajectories in cartesian space (use quaternions as orientations)
- Joint configurations
- Force trajectories (noisy)

Time variant data. This entails that we can derive the velocities and accelerations.

6 Trajectory Construction

7 Dynamic motion primitives

The origins of this concept is based on how frogs moves their legs.

$$\tau \dot{z} = K(g - y_0) - Dz + f(x) \tag{1}$$

$$\tau \dot{y} = z \tag{2}$$

Designed on acceleration level. Mass forces and acceleratio goes together. Let the first part of the first equation be the mass, and $f(x)$ be the acting force.
This representation gauranties continuity.
g is the goal/setpoint, and y_0 is the start state of the desired movement. K and D are the positive gains You are essentially computing an error between the setpoint and the state. This will give a PD control, which permits Point to Point control. $f(x)$ represent the movement.

- Not explicitly time dependent. $\rightarrow$ Can run forever
- Demonstrated/ recorded movment can be reproduced.
- You can alter the path online, by e.g. representing obstacles as potential fields.
- Movement can be stopped and resumed again.

They did a tutorial paper on DMP's, which comes with how to's on the code and euqations.
τ is the time constant, that defines how long the trajectory will be. (how long the controller should run, not necessarily how long the trajectory will be.). Usefull when reaching motion. You have to consider the physical constraints of the robot.

7.0.1 Temporal evolution phase system

The system has an exponential decaying function.

$$\dot{x} = -ax \tag{3}$$

The DMP's are on a joint level. They are critically damped.

7.1 Transformation system

$$\tau^2 \dot{y} = K(g - y) - D\tau\dot{y} \tag{4}$$

which is typically written as:

$$\tau \dot{z} = \alpha_y(\beta_y \ldots) \tag{5}$$

7.2 Forcing terms

Weighted mixture of Gaussian Basis Functions. you have define a center and a width.
The top of the basis function, should correspond to a point on the trajectory, which you can compute a weight from, then you could do the inverse.

7.3 Cartesian-space Orientation

Math more or less the same.
You need to compute the difference between two quaternions, using a log function. This is a quaternion function that finds the difference between two quaternions. Needed for the proportional part.
Each degree of freedom should have a controller.
Important to specify a start configuraiton and start position.

8 Reproduction on the robot

Encoding. Can be done offline,

9 2. part: Robot control in context

9.1 Inverse kinematics solver

- Analytical solver
- Numerical solver

9.2 Jacobian

When you have 6 DOF in cartesian space, and 7 DOF joint space, then the jacobian will be a matrix $J \in \mathbb{R}^{6\times 7}$.

9.3 Position Control

You usually have PD control on the actuators, which results in torque. Collaborative robot may also have velocity and torque trajectories. This is also considered a *high gain* controller (very stiff).
Dynamics. You need to kow the dynamic parameters of the robot. Most important forces are the gravity and coriolis effect.

9.4 Position - velocity control

Smother motion, if the controller can take both inputs.

10 Measuring data

10.1 force vs force/torque

Force only measures a single quantity, but you can use e.g. jacobians to map it to torques in the joints.

10.1.1 Force/Torque

Integrated systems. Comes with strain gauges on each pin, which provide multi degree of freedom resolution.
If it is mounted on the wrist, then be carefull not to jam the robot into a table, or do not mount a too long tool on it.
they are susceptive to drift due to temperature increase.

10.1.2 Joint torque sensors

Radial measurement rather than linear.

11 Force control strategies

Note there is no need for the I term.
types:

- PD (cascade)
- Parallel
- Hybrid

If it is possible to use velocity as a controller, DO SO. For force controllers, there will be great constant errors due to integration, unless you have a really fast control loop.

11.1 Parallel force controller

Problem: scale the inputs.

11.2 Hybrid force controller

12 Impedance

Stiffness control. Control the stiffness of the manipualtor to get a force.

12.1 Impedance control

You need the dynamics to do torque control. You need to model the robot with coriolis and friction. you can control Damping and stiffness, but not the inertia. The robot will behave like a spring, entailing that if you lower the damping and stiffness, the more periodic.

12.2 Admittance control

Control the motion of the manipulator after the force has been applied. More similar to a force controller relative to impedance control. Must be equipped with force/torque sensors. Admittance control have 6 springs in the wrist which can be tuned.

13 Coupling with feedforward trajectory control

DMP and Desired force is feed forward. Use a admittance controller as a negative feedback loop, and have the impedance and rigid-body robot controller as the robot controller.

14 Conclusion

- When working with the environment, it is recommended to have some kind of force/torque or force sensors.
- Admittance controllers should be implemented with carefull parameter selection.