18th December 2025 IIT Dharwad

* Resolved Rate motion Control * (RRMC)

Core equation $= J(\theta) \cdot \dot{\theta}$

$$V = J(\theta) \cdot \dot{\theta}$$

$V =$ Cartesian velocity of the gripper

$J(\theta) =$ Jacobian matrix

$\dot{\theta} =$ Velocity of motors

Step-I :- define Velocity vector that draws a Circle

Step-II :- Get the jacobian at every step.

Step-III :- Calculate required motor speeds using pseudo-Inverse.

$$\dot{\theta} = J^{-1} \cdot V$$

Step-IV :- Integrate : Convert this speed into tiny position step ($\Delta\theta = \dot{\theta} \cdot dt$) & send it to motors.

* Execution of Bilateral Teleoperation System *

This system has 3 pillars :- 1) Kinematics & Coordinate System
2) Control theory & Signal processing
3) Haptics physics.

This System is a Master-Slave system

Master (Geomagic Touch / phantom omni)
↳ Reads hands position & exerts force on our hand.

Slave (Viper x 300 arm)
↳ Robotic manipulator that attempts to replicate the Master's State.

Working of phantom omni (Haptic device)

- Sensing (encoders): Each joint (Base, shoulder, elbow) has a digital optical encoder. When stylus is moved, encoder count ticks.

- Forward Kinematics: The driver (omni_common) takes the joint angles ($\theta_1, \theta_2, \theta_3$) & does Forward Kinematics to Calculate Cartesian position (x, y, z) of the stylus tip.

- Actuation: To create force, it runs the Jacobian transpose. It calculates, "To create 2 Newtons of force in the X direction, how much torque must I apply to the shoulder motor."

* challange 1 - Frame mismatch

Geomagic frame	Interbotin frame
+X: Right	+X = Forward
+Y: Up	+Y = Left
+Z: Towards me (Pull)	+Z = Up

In the program, a frame Rotation & Translation was performed to overcome challange 1.

$$\boxed{P_{robot} = R \cdot (P_{stylus} \times S) + T_{offset}}$$

S → Scaling factor
R → Rotation matrix
T_{offset} → Translation

- Signal processing

1 euro filter

Initial observation showed that robot motion was jittery and delayed. This is because human hands have a natural tremor (frequency ≈ 8-12 Hz)

1 euro filter is an adaptive low-pass filter.

Basically, on slow movement, it turns on heavy filtering (removes).
on fast movement, it turns off filtering (removes lag)

A simple first order low pass filter

$$\hat{X}_i = \alpha \cdot X_i + (1-\alpha)\,\hat{X}_{i-1}$$

$\hat{X}_i$ = filtered output at time i
X_i = raw input at time i
$\hat{X}_{i-1}$ = previous filtered output
α = smoothing factor ($0 \leq \alpha \leq 1$)

$$\alpha = \frac{1}{1 + \tau/T_e}, \quad \tau = \frac{1}{2\pi f_c}$$

In 1 euro filter, f_c is changed dynamically based on speed.

$$f_c = f_{min} + \beta \cdot |\dot{x}|$$

f_{min} = Cut-off frequency
β = Speed Coefficient (decides how quickly the filter opens up when you start moving)

$|\dot{x}|$ = magnitude of hand's speed (velocity)

Working of Algorithm

When a new position X_i arrives:

1) Calculate Speed: Compute raw velocity, but also filter velocity so a single noise spike doesn't trick the system into thinking we are moving fast.

$$\dot{X}_{raw} = \frac{X_i - X_{i-1}}{T_e}$$

$$\boxed{\dot{X}_{filtered} = \alpha_d \cdot \dot{X}_{raw} + (1-\alpha_d) \cdot \dot{X}_{prev}}$$

2) Calculate Adaptive Cutoff (f_c): Using filtered speed, determine new cutoff frequency.

$$f_c = f_{min} + \beta \, |\dot{X}_{filtered}|$$

3) Calculate Alpha (α)

Convert new f_c into weighing factor

$$\alpha = \frac{1}{1 + \frac{1}{2\pi f_c}}$$

4) Apply standard LPF equation with the new, adaptive α.

- Control Theory: Synchronization

- There was lag initially due to time synchronisation error.

The loop: python script runs at 40Hz (0.025 secs)

- Set moving_time = 0.15s (Basically sending a new command every 0.025 secs but telling the robot to take 0.15s to finish it. This creates a motion buffer.

Since, buffer is constant, motion is continuous.

- **Haptic Theory:**

A Virtual spring was modelled, position error feedback was used (Virtual Coupling).

Classic Hooke's law application

$$F_{feedback} = K \cdot (P_{robot\_actual} - P_{robot\_target})$$

Scenario A: Moving in Air

- Stylus is moved
- Robot follows accurately
- $P_{actual} \approx P_{target}$
- Error is near zero. Force = 0

Scenario B: Touching a box

- push stylus forward
- Robot hits physical box & stops
- Error grows larger
- Force increases (Code sends force to phantom motors, pushing the hand back.

The deadband "(δ)"

$$\text{If } |Error| < \underline{\delta}, \text{ then Force} = 0$$

Summary

Title:- 6 DOF- Bilateral Teleoperation of a Robotic manipulator (Vx300s) with Haptic feedback.

Abstract:- Implemented a master-slave teleoperation System including a Geomagic Touch haptic device interfacing with an Interbotix Vx300s via ROS2.

Features:

1) Coordinate mapping: Transformation matrices to map Varying Workspace & coordinate frames of two devices.

2) Adaptive filtering: Implementation of 1 euro filter to eliminate physiological tremor without inducing Kinematic latency.

3) Haptic feedback: A position based impedence Control utilizing a deadband filter to eliminate free-space drag.

4) Real-time Visualization: Integration of live telemetry plotting (rqt_plot) to verify trajectory tracking performance.

— xx — xx — xx —

* Sequential Equilibrium Inverse Kinematic Optimization *

SEIKO is a whole body Control algorithm that solves for joint positions, torques & Contact forces simultaneously to satisfy task goals while strictly obeying static equilibrium laws.

In defining variables, we solve for a combined vector y

$$y = \begin{bmatrix} q \\ f_c \end{bmatrix}$$

- $q \in R^n$: Joint Configurations (angles)
- $f_c \in R^m$: Contact forces

Constraint A : Kinematics

Standard F.K maps joint spaces to task spaces.

$$x_{task} = \phi(q)$$

For a target pose x_{des}, error is

$$e_{kin} = \phi(q) - x_{des} = 0$$

Constraint B: Static Equilibrium

For a robot to hold a pose, sum of forces & torques must balance.

eq^n of motion for a robot is generally

$$M(q)\cdot\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + J_c(q)^T f_c$$

For SEIKO, $\dot{q} \approx 0$; $\ddot{q} \approx 0$, treats dynamic terms as external wrenches, this simplifies to:

$$g(q) = \tau + J_c(q)^T f_c$$

$g(q)$ = Gravity vector
τ = Joint torques
$J_c(q)^T$ = Transpose of contact jacobian
f_c = external contact forces.