Technical Analysis: UR3e & Hand-E Drawing System

Research on Programmatic Trajectories and Hardware Constraints

1. The Robot Arm: UR3e

The UR3e is a 6-axis collaborative robot. In a drawing application, it functions as a high-precision positioning system.

1.1. Coordinate System and Poses

The robot's state is defined by a Pose ($p[x, y, z, rx, ry, rz]$).

- **Position (x, y, z):** Coordinates in meters relative to the center of the base.
- **Orientation (rx, ry, rz):** A rotation vector where the direction is the axis of rotation and the magnitude is the angle in radians.

1.2. Motion Limitations

- **Singularities:** The arm can encounter mathematical "dead zones" (singularities) where the joints cannot calculate a valid move. This often happens when the arm is fully extended or when the wrist joints (J4 and J6) align.
- **Reach:** For the UR3e, the maximum reach is approximately 500mm. However, drawing requires the arm to stay within a "dexterous workspace" where orientation can be maintained.
- **Speed vs. Accuracy:** While the robot can move at 1 m/s, drawing on PLA requires lower speeds ($v < 0.05$ m/s) to ensure the acrylic ink adheres correctly.

2. The Gripper: Hand-E

The Robotiq Hand-E is designed for high-precision tasks. For holding an acrylic pen, the control logic focuses on force management.

2.1. Holding the Pen

The pen is held between two parallel fingers. Control is achieved by writing to specific registers:

- **rPR (Position):** 0 is fully open, 255 is fully closed. To pick a pen, the gripper moves to a target position.
- **rFR (Force):** 0 is minimum force, 255 is maximum (approx. 130N). For plastic acrylic pens, a force value between 50 and 100 is recommended to prevent crushing.
- **rSP (Speed):** Determines how quickly the fingers close.

2.2. Object Detection

The gripper uses the `gOBJ` register to provide feedback:

- **Value 1 or 2:** Object detected (the pen is securely held).
- **Value 3:** Gripper reached position without finding an object (pick failure).

2.3. Drawing

Once the pen approaches the desired drawing surface, the arm will slowly move towards the surface, applying enough pressure for the pen to start drawing. Then the arm will move following a set of points using the movep (move process) function.

To prevent the pen from slipping out of the gripper, the drawing direction must remain parallel to the finger alignment.

3. Programmatic Trajectory Generation

3.1. Constant Speed with `movep`

For drawing, `movep` (Move Process) is the critical command. Unlike `movel` (Linear), which may vary speed at corners, `movep` maintains a constant Tool Center Point (TCP) speed. This is essential for acrylic pens to prevent ink blobs.

3.2. TCP Calibration

The **Tool Center Point** must be shifted from the robot's flange to the tip of the pen.

- **Command:** `set_tcp(p[0, 0, Z_offset, 0, 0, 0])`
- This ensures that all x, y, z movements in the script refer to where the ink meets the duck.

4. Code Implementation Logic

The final script uses the `URBasic` library to send URScript strings. The logic follows a sequence of:

1. Initialize socket connection to the gripper.
2. `movej` to a safe hover position over the pen holder.
3. `movel` to descend and `SET rPR 255` to grasp.
4. Apply `set_tcp` for the active pen.
5. Execute the drawing path using a series of `movep` waypoints given by tracing team

5. Sources

- https://github.com/Toys-R-Us-Rex/ur3e-control/tree/main/manuals
- https://s3-eu-west-1.amazonaws.com/ur-support-site/105210/99403_UR3e_User_Manual_en_Global.pdf