Design

(a) Design Criteria and Desired Functionality:

- The system must accurately detect and track AR markers on blocks using the topic ar pose marker.
- Must calculate proper yaw orientations using euler from quaternion transformations.
- Must successfully execute two stacking patterns
  1. Pyramid: 3-block base layer, 2-block middle layer, 1-block top
  2. Vertical: Sequential stacking of blocks vertically
- Must maintain precise positional control for block placement using either Moveit controller or the custom implemented PID controller.
- Must properly coordinate gripper operations with movement.
- Must avoid collisions between blocks during stacking.

(b) Design Description: The system uses a modular architecture with several key components:

1. Perception System:
   - Uses AR marker detection via ROS.
   - Implements scanning motion to detect all blocks.
   - Calculates yaw adjustments using quaternion transformations.
2. Motion Planning:
   - Uses either MoveIt for trajectory planning.
   - Implements LinearTrajectory for movement paths
   - Uses PathPLanner for safe trajectories.
3. Control System:
   - Offers choice between PID and open-loop controllers.
   - Implements velocity-based control for smooth motion.
   - Uses feedback from joint states for positional control.

Design Choices and Trade-Offs:

1. **Block Detection Strategy**:
   - Choice: Scanning motion with multiple angles (0.2, 0.1, 0, -0.1, -0.2 radians)

- Trade-off: More comprehensive detection but longer initialization time.

2. Movement Strategy:
    - Choice: Linear trajectories with fixed timing.
    - Trade-off: Simpler implementation but potentially slower than optimized paths.

3. Stacking Approach:
    - Choice: Layer by layer building with intermediate tuck positions.
    - Trade-off: More stable but requires more movements. Some movements computed by IK might be deficient, meaning blocks might overlap, this reduces robustness.

(d) Real Engineering Impact:

1. Robustness:
    - Positives:
        - Muliple scanning angles improve marker detection reliability.
        - Error handling for marker detection timeouts.
        - Intermediate tuck positions reduce error accumulation
    - Limitations:
        - Fixed movement times might not adapt to different loads.
        - Dependent on AR marker visibility and camera resolutions.
        - The intermediate positions take a lot of time, making the project less useful.

2. Durability
    - Positives:
        - Uses controlled velocities to reduce mechanical stress.
        - Implements smooth trajectories.
    - Limitations:
        - Fixed height increments stresses the system with varying block sizes. We use same sied blocks throughout the project.
        - Control isn't always predictable with the robot, causing jerky movements from time to time.

3. Efficiency:
    - Positives

  - Modular code structure allows easy modifications.
  - Resuses movement patterns
- Limitations
  - Multiple tuck operations increase completion time.
  - Fixed timing regardless of distance makes the system slow.

Some Technical Details:

Getting the yaw adjustment:

To get the yaw adjustment of each block, we use euler from quaternion. The transformations are inherently in quaternion, so we needed to convert them to euler coordinates to get the correct yaw adjustments and rotate while placing them. The equation for that is as follows:

$$\psi = tan^{-1}(\frac{2(q_w q_z + q_x q_y)}{1 - 2(q_y^2 + q_z^2})$$

Trajectory Planning

We follow a linear trajectory to the end-point. For a given time interval, this trajectory has constant acceleration for the half of the time interval, and constant deceleration for the remaining half:

$$x(t) = 0.5at^2 + x_0, 0 \leq t \leq T/2$$

$$x(t) = v_{\max}(t - T/2) - 0.5a(t - T/2)^2 + (x_0 + x_f)/2, T/2 \leq t \leq T$$

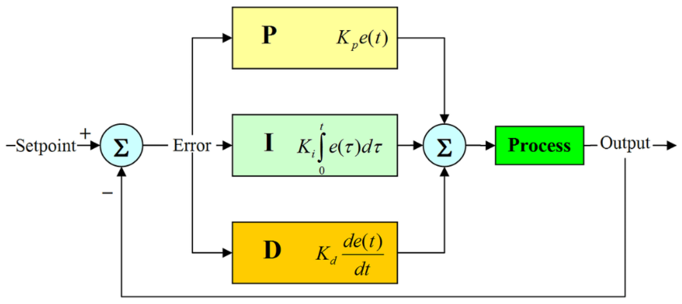
where $v_{\max}$ is the maximum velocity reached at $t = T/2$, x_0 the initial position and x_f the target position.

Implementation of PID control

We use PID control where error = (target - current) position. The proportional term provide immediate correction proportional to the error, the derivative term provides damping to prevent overshooting, and the integral term eliminates steady state error by discounting finite horizon errors.

$$u = u_{ff} + K_p e + K_i \int_0^t e dt + K_d \dot{e}$$

where u_{ff} is the desired velocity, e error, K_p the proportional term, K_d the derivative term, K_i the integral term. Of course it is usually intractable to compute the integral error, so we discount previous integral errors and add the current error.


Setpoint
+
−
Σ
Error
P
$K_p e(t)$
I
$K_i \int_0^t e(\tau) d\tau$
D
$K_d \frac{de(t)}{dt}$
Σ
Process
Output