[CoRL'24] Visual Manipulation with Leg

1. Link: https://arxiv.org/pdf/2410.11345
2. Arthurs and institution: Xialin He†, Chengjing Yuan, Wenxuan Zhou, Ruihan Yang, David Held, Xiaolong Wang from UIUC, UCSD and CMU
    1. Wenxuan Zhou is a researcher in META, she's doing non-prehensile manipulation and planning
    2. Xialin He is a PHD student, he's doing RL locomotion and manipulation

TL;DR A system that enables quadruped robots to interact with objects using their legs, it has a visual manipulation policy decides how the leg should interact with the object, and a loco-manipulator controller manages leg movements and body pose adjustments, based on impedance control and Model Predictive Control (MPC).

Thoughts and critisims

1. the system architecture is useful
2. their task is object pose alignment, which is an easier level of joint configuration alignment/articulated object pose alignment.
3. instead of apply impedance control for all joints, the user only use it for leg doing manipulation task
4. the perception module needs target pose as input, which has cost to get

Related works

Quadrupedal Locomotion

1. inputs
    1. proprioceptive senses
    2. exteroceptive perception
        1. 2.5D height map
        2. depth sensing
2. methods

1. RL (reduce computation cose, generalizability)
2. model based control (ensure stability, require computation power)

6D Manipulation with Manipulators

focused on dexterous hands and grippers on fixed bases

Non-Prehensile Manipulation

1. defination: a task involves interacting with objects without grasping, has been a focus for tasks like pushing, sliding, and flipping
2. methods: model based, model free RL

Manipulation with Legged Robot

1. manipulate part
    1. arm
    2. base
    3. gripper on leg(prehensile)
    4. leg (non-prehensile, door pushing, basket lifting)

Contributions

Key concepts

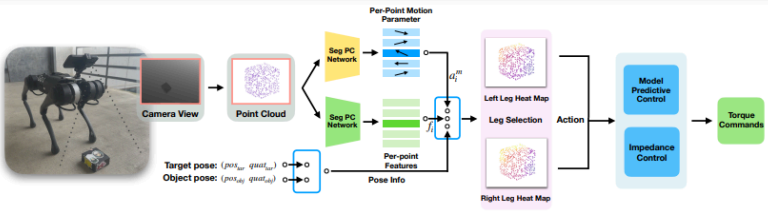


Figure 2: **Visual Manipulation with Legs.** Our system operates in stages: 1) A depth camera captures the object's point cloud. 2) The point cloud and target pose are processed by our network. 3) The manipulation leg is chosen based on the highest Q-value. 4) Pre-contact, contact, and action details are sent to the low-level control system. 5) The control system uses impedance control to direct the selected leg, while a Model Predictive Controller maintains balance, sending torques to the robot.

Task defination

Object Pose Alignment Align an object's 6D pose with a given target pose, represented as a transformation relative to the current object pose. Using an observed object point cloud and the target

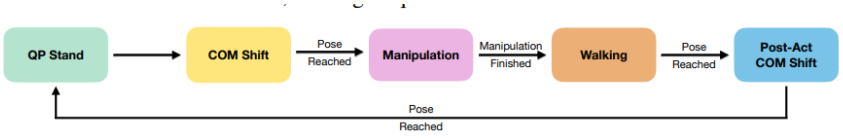
transformation, the quadruped robot interacts with the object using its front legs to achieve alignment

Algorithm

Learning Visual Manipulation Policy

1. framework: RL Actor-critic policy
2. inputs: point clouds
3. outputs: choice of leg, contact point and a vector of motion parameters that define the post-contact movement
4. details
    1. TD3
    2. actor outputs per-point motion param, critic outputs per-point Q-values
    3. use PointNet++ as backbone

Model-Based Loco-Manipulation Controller

1. 

2. function
    1. interact with the target object using the learned visual policy
    2. move the robot for long-horizon object manipulation
3. FSM design
    1. QP stand: Quadratic Programming-based Standing Control
        1. stand with 4 legs
    2. COM shift (center of mass)
    3. manipulation: impedance control for one leg, MPC for stance leg
    4. walking: MPC
    5. Post-act COM shift: get back leg
4. details
    1. MPC

reaction force over a finite horizon κ to determine optimal control inputs and trajectory:

$$\min_{x,u} \sum_{i=0}^{k-1} ||x_{i+1} - x_{i+1,ref}||_{Q_i} + ||u_i||_{R_i} \text{subject to } x_{i+1} = A_i x_i + B_i u_i, \underline{c}_i \leq C_i u_i \leq \bar{c}_i, D_i u_i = 0,$$

Here, x_i is the state, u_i the control input, Q_i and R_i are weight matrices, A_i and B_i describe system dynamics, and C_i, $\underline{c}_i$, $\bar{c}_i$ set control constraints. D_i identifies forces for feet not in contact. MPC commands include roll, pitch, yaw, Cartesian position, and target velocities. We primarily manipulate yaw (Z-axis) and provide linear velocity for X and Y directions. More details in Bledt et al. [11].

2. Impedance Control

ment, enabling precise object interaction. Control torques (τ) are computed as:

$$\tau = J^T \left(K_p(p_{\text{des}} - p_{\text{foot}}) + K_d(v_{\text{des}} - v_{\text{foot}}) \right),$$

where p_{des} and v_{des} are the desired foot position and velocity, p_{foot} and v_{foot} are the actual position and velocity, K_p and K_d are proportional and derivative gains, and J maps foot force to joint torques.

Point-cloud Registration Module

1. inputs: source and target point clouds with per-point normals
2. outputs: the transformation from source to target.
3. model: RPM-Net

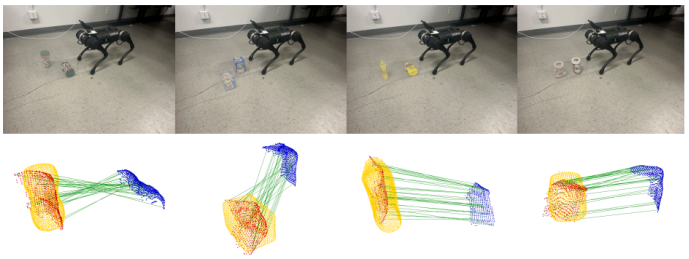


Figure 4: We employ RPM-Net for the registration process of a real robot, emphasizing successful registrations. In the illustration, the blue point cloud represents the source data captured by the camera, the yellow point cloud corresponds to the complete scan of the object, and the red point cloud shows the source data after transformation. Green lines indicate the flow vectors.

4.

5. metrics: flow distance, Chamfer distances

Experiments

1. Task

Task. We evaluate all methods on the following 5 tasks.

- **Box push (Fixed goal)**: Push a box forward (along x-axis in the robot frame) by 15cm.
- **Box push (Random goals)**: Push a box to a target position (in the robot frame) on the ground. The X and Y coordinates of the target position are uniformly sampled from $(10\text{cm}, 20\text{cm})$ and $(-5\text{cm}, 5\text{cm})$ respectively.
- **Box flip + push (Random goals)**: Flip a box left or right by 90 degrees and push to a target position sampled as in the *Box push (Random goals)* task.
- **Multi-object push (Fixed goal)**: Push an object forward (along x-axis in the robot frame) by 15cm.
- **Multi-object push (Random goals)**: Push an object to a target position sampled as in the *Box push (Random goals)* task.

2. Evaluations: sim and real

(a) Flow baseline (b) Ours

Figure 9: **Low-level Control Error.** Red arrows denote commanded movements, yellow arrows represent actual movements.

Claimed limitation

1. QP computation cost
2. uncertainty in observation
    1. IMU
    2. camera
    3. no friction estimation