Supplementary Document: Human-Robot Co-Transportation using Disturbance-Aware MPC with Pose Optimization

Al Jaber Mahmud, Amir Hossain Raj, Duc M. Nguyen, Weizi Li, Xuesu Xiao, and Xuan Wang

1 Whole-Body Dynamics with Torque-Level Control

Configuration and State Variables Since the Fetch mobile manipulator has a differential-drive base and a 7-DoF robotic arm, we define the configuration vector,

$$q \;=\; \begin{bmatrix} x_{\text{base}} & y_{\text{base}} & \phi & \theta_1 & \theta_2 & \cdots & \theta_7 \end{bmatrix}^\top \;\in\; \mathbb{R}^{10}, \tag{1}$$

where x_{base}, y_{base} are the planar coordinates of the base in an inertial frame, ϕ is the heading angle of the base, $\theta_1, \ldots, \theta_7$ are the 7 joint angles of the manipulator arm. Considering second-order dynamics, we collect the joint velocities into a vector $\dot{q} \in \mathbb{R}^{10}$. Thus, we define the state vector of the system as,

$$X \;=\; \begin{bmatrix} q^\top & \dot{q}^\top \end{bmatrix}^\top \;\in\; \mathbb{R}^{20}.$$

Torque Inputs: The Fetch robot's base has two wheel actuators (left and right), and seven joint actuators for the arm, yielding a total of 9 actuators. We combine these nine torques into a single vector,

$$\tau \;=\; \begin{bmatrix} \tau_{\text{base}}^\top & \tau_{\text{arm}}^\top \end{bmatrix} \;\in\; \mathbb{R}^9,$$

where $\tau_{\text{base}} \in \mathbb{R}^2$ contains the left and right wheel torques, and $\tau_{\text{arm}} \in \mathbb{R}^7$ consists of the arm joint torques.

Rigid-Body Dynamics in Joint-Space: In general, the standard rigid-body dynamics (including nonholonomic base) can be written in the form

$$M(q)\,\ddot{q} \;+\; C\big(q,\dot{q}\big)\,\dot{q} \;+\; G\big(q\big) \;=\; \mathcal{B}(q)\,\tau, \tag{2}$$

where, $M(q) \in \mathbb{R}^{10\times 10}$ is the positive-definite mass/inertia matrix, $C(q,\dot{q})\,\dot{q} \in \mathbb{R}^{10}$ accounts for Coriolis and centripetal terms, $G(q) \in \mathbb{R}^{10}$ captures gravitational forces/torques, $\mathcal{B}(q) \in \mathbb{R}^{10\times 9}$ is an input selection matrix that maps the 9-dimensional torque vector τ to the 10-dimensional joint-space accelerations.

Input Selection Matrix $\mathcal{B}(q)$: We partition $\mathcal{B}(q)$ according to base DOFs vs. arm DOFs:

$$\mathcal{B}(q) \;=\; \begin{bmatrix} \mathcal{B}_{\text{base}}(q) & \mathbf{0}_{3\times 7} \\ \mathbf{0}_{7\times 2} & I_{7\times 7} \end{bmatrix}.$$

The lower-right block $I_{7\times 7}$ indicates that each arm joint torque directly maps to its corresponding joint angle. The upper-left block $\mathcal{B}_{\text{base}}(q) \in \mathbb{R}^{3\times 2}$ describes how the two-wheel torques map into forces/torques in (x_{base}, y_{base}, ϕ). If each wheel has radius r and is offset laterally by $\pm b$ from the center point, then the forward force produced by one wheel torque τ_{wheel} is $F_{\text{wheel}} = \frac{\tau_{\text{wheel}}}{r}$. When

the robot's heading is ϕ, the forward direction aligns with $(\cos\phi,\ \sin\phi)$, and the yaw torque about the center is offset by $\pm b$. Hence, a commonly used form is:

$$\mathcal{B}_{\text{base}}(\phi) \;=\; \begin{bmatrix} \frac{\cos\phi}{r} & \frac{\cos\phi}{r} \\ \frac{\sin\phi}{r} & \frac{\sin\phi}{r} \\ -\frac{b}{r} & +\frac{b}{r} \end{bmatrix},$$

reflecting that each wheel torque contributes to the net forward force and yaw torque in the inertial frame.

State-Space Formulation in Continuous Time: From the rigid-body dynamics (2), we write

$$\ddot{q} \;=\; M(q)^{-1}\Big[\mathcal{B}(q)\,\tau \;-\; C(q,\dot{q})\,\dot{q} \;-\; G(q)\Big].$$

Hence,

$$\underbrace{\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix}}_{\dot{X}} \;=\; \begin{bmatrix} \dot{q} \\ M(q)^{-1}\big[\mathcal{B}(q)\,\tau - C(q,\dot{q})\,\dot{q} - G(q)\big] \end{bmatrix}. \tag{3}$$

This encodes the continuous-time rigid-body dynamics at the torque level.

Discrete-Time Approximation for Control: For real-time control (e.g., Model Predictive Control), this dynamics is discretized with a small time step Δt. A forward Euler approximation of (3) yields:

$$q(k+1) \approx q(k) \;+\; \dot{q}(k)\,\Delta t, \tag{4}$$

$$\dot{q}(k+1) \approx \dot{q}(k) \;+\; \Delta t\, M\big(q(k)\big)^{-1}\Big[\,\mathcal{B}\big(q(k)\big)\,\tau(k) - C\big(q(k),\dot{q}(k)\big)\,\dot{q}(k) - G\big(q(k)\big)\Big]. \tag{5}$$

i.e.,

$$X(k+1) \;=\; X(k) \;+\; \Delta t\, F\big(X(k),\,\tau(k)\big),$$

where $F(\cdot)$ follows from the continuous-time right-hand side of (3).

End-Effector Updates in the Inertial Frame: Once we have $X(k+1)$, we extract $q(k+1)$ and use forward kinematics to obtain the updated end-effector pose $x(k+1)$ in the inertial frame:

$$x(k+1) \;=\; f\big(q(k+1)\big),$$

where $f(\cdot)$ is the forward-kinematics map. Similarly, the end-effector velocity is

$$\dot{x}(k+1) \;=\; J\big(q(k+1)\big)\,\dot{q}(k+1),$$

with $J(\cdot)$ being the end-effector Jacobian.

2 LSTM Implementation for Human Trajectory Prediction

Standard LSTM Equations: Consider a time-indexed sequence of input vectors $\{r(t)\}_{t=1}^{T}$ (manipulator's end-effector pose to be tracked to adapt to human movements, where each $r(t) \in \mathbb{R}^6$ includes 3D position (r^x, r^y, r^z) and 3D orientation $(r^\alpha, r^\beta, r^\gamma)$). At each time step t, the LSTM maintains a hidden state $h(t) \in \mathbb{R}^d$ and a cell state $c(t) \in \mathbb{R}^d$. The update from time $(t-1)$ to t is given by:

$$i(t) = \sigma\big(W_{ir}\, r(t) + W_{ih}\, h(t-1) + b_i\big), \tag{6}$$
$$f(t) = \sigma\big(W_{fr}\, r(t) + W_{fh}\, h(t-1) + b_f\big), \tag{7}$$
$$o(t) = \sigma\big(W_{or}\, r(t) + W_{oh}\, h(t-1) + b_o\big), \tag{8}$$
$$\tilde{c}(t) = \tanh\big(W_{cr}\, r(t) + W_{ch}\, h(t-1) + b_c\big), \tag{9}$$
$$c(t) = f(t) \odot c(t-1) \;+\; i(t) \odot \tilde{c}(t), \tag{10}$$
$$h(t) = o(t) \odot \tanh\big(c(t)\big). \tag{11}$$

where $i(t)$ is the Input Gate that controls how much of the candidate cell state $\tilde{c}(t)$ is added to the current cell state $c(t)$. $f(t)$ is the Forget Gate that controls how much of the previous cell state $c(t-1)$ is retained. $o(t)$ is the Output Gate that controls how much of the current cell state $c(t)$ is exposed to the hidden state $h(t)$. $\sigma(\cdot)$ is the sigmoid function, $\tanh(\cdot)$ is the hyperbolic tangent, and $\odot$ denotes element-wise multiplication. $W_{(\cdot)\,r}, W_{(\cdot)\,h}, b_{(\cdot)}$ are the trainable weight matrices and bias vectors for the LSTM gates.

Network Structure: We employ a two-layer LSTM architecture, where each layer has $d = 50$ hidden units. The overall architecture is:

- **Layer 1**: An LSTM with 50 hidden units.
- **Layer 2**: Another LSTM with 50 hidden units, receiving the hidden output from Layer 1.
- **Output (Dense) Layer**: A fully connected layer that projects the final LSTM hidden state to the desired output dimension for each future time step.

Let

$$\big(h_1(t),\, c_1(t)\big) \quad \text{and} \quad \big(h_2(t),\, c_2(t)\big)$$

denote the hidden and cell states of the first and second LSTM layers, respectively. At each time t:

$$(h_1(t),\, c_1(t)) \;=\; \mathrm{LSTM}_1\big(r(t),\, h_1(t-1),\, c_1(t-1)\big).$$

The hidden state $h_1(t)$ is then passed to the second layer:

$$(h_2(t),\, c_2(t)) \;=\; \mathrm{LSTM}_2\big(h_1(t),\, h_2(t-1),\, c_2(t-1)\big).$$

Finally, the dense layer projects $h_2(t)$ to obtain the one-step prediction:

$$\tilde{r}(t+1) \;=\; W_{\text{out}}\, h_2(t) \;+\; b_{\text{out}}.$$

where, $\tilde{r}(t+1) \in \mathbb{R}^6$ is predicted human pose at time $(t+1)$. To generate multi-step predictions, the LSTM is unrolled iteratively for 10 steps beyond the last input time step, producing:

$$\big\{\, \tilde{r}(t+1),\, \tilde{r}(t+2),\, \ldots,\, \tilde{r}(t+10) \big\}.$$

Input and Output Specification:

- **Input Sequences:** We collect a sequence of 15 previous consecutive true human poses as input:
$$\big\{\, r(t-14),\, r(t-13),\, \ldots,\, r(t)\big\},$$
These 15 time-step data are fed into the LSTM in chronological order.

- **Output Sequences:** At time t, the network outputs a sequence of 10 future predicted poses:
$$\big\{\tilde{r}(t+1),\, \tilde{r}(t+2),\, \ldots,\, \tilde{r}(t+10)\big\}.$$
This is done by unrolling the LSTM for 10 additional steps beyond the last input.

Data Collection and Normalization We train on synthetic human motion trajectory data that represents ground truth. Each trajectory is segmented into overlapping windows of 15 input poses and 10 corresponding future poses as labels:

$$\text{Input: } \big\{r(t-14),\, \ldots,\, r(t)\big\}, \quad \text{Label: } \big\{r(t+1),\, \ldots,\, r(t+10)\big\}.$$

Each dimension (e.g., $x, y, z, \alpha, \beta, \gamma$) is normalized by subtracting the mean and dividing by the standard deviation computed over the entire training set. Normalization helps stabilize the training process.

Loss Function and Optimization: The training objective is to minimize the Mean Squared Error across all 10 predicted future steps:

$$\mathcal{L} \;=\; \frac{1}{10\,N}\sum_{i=1}^{N}\sum_{j=1}^{10}\Big\|\, r_i(t+j) \;-\; \tilde{r}_i(t+j)\,\Big\|_2^2,$$

where N is the total number of training samples, $r_i(t+j)$ is the ground-truth pose at time $t+j$ for the i-th sample, and $\tilde{r}_i(t+j)$ is the model prediction. The network weights

$$\Big\{W_{(\cdot)\,r},\; W_{(\cdot)\,h},\; b_{(\cdot)},\; W_{\text{out}},\; b_{\text{out}}\Big\}$$

are optimized via Adam optimizer, a gradient-based method.