Neural network approximator for input allocation

Matteo Dalle Vedove

12-09-2023

University of Trento - Department of Industrial Engineering

Table of Contents

Problem introduction

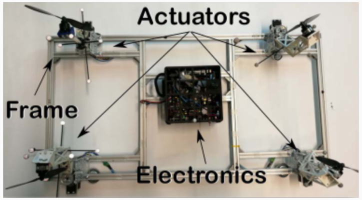

Figure 1:

Image of the ROSPO prototype.

The goal is to test the viability of neural networks for solving allocation problems. Simulation uses a custom model of the **ROSPO** (*ROtor graSPing Omnidirectional*) platform developed at LAAS-CNRS to test allocation schemes in propeller-driven systems.

System description

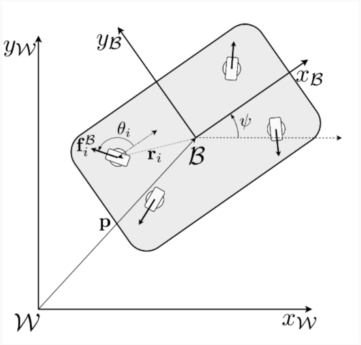

Figure 2:

Schematic representation of the ROSPO platform.

The platform consists of:

- an aluminum frame that uses spherical omnidirectional wheels to slide on the ground;
- a variable number of turret modules.

Each turret use a stepper motor to rotate; on top of each module, a BLDC motor drives a propeller that generates thrust to move the platform.

System description: BLDC motor

The BLDC motor is modelled considering the equivalent single-winding electrical circuit; the dynamics also requires the rotational dynamics of the system;

$$\begin{cases} L\frac{\mathrm{d}i}{\mathrm{d}t} = v - Ri - e \\ \Omega = k_v e \\ J_m T_m = ie \\ J_m\frac{\mathrm{d}\Omega}{\mathrm{d}t} = T_m - T_r \\ T_r = k_r\Omega^2 \end{cases} \Rightarrow \begin{cases} L\frac{\mathrm{d}i}{\mathrm{d}t} & = v - Ri - \frac{\Omega}{k_v} \\ J_m\frac{\mathrm{d}\Omega}{\mathrm{d}t} & = \frac{i}{k_v} - k_r\Omega^2 \end{cases}$$

System description: BLDC motor

The state of the model is $\boldsymbol{x} = (\Omega, i)$, the input $u = v$.

$$\begin{cases} L\frac{\mathrm{d}i}{\mathrm{d}t} & = v - Ri - \frac{\Omega}{k_v} \\ J_m\frac{\mathrm{d}\Omega}{\mathrm{d}t} & = \frac{i}{k_v} - k_r\Omega^2 \end{cases}$$

The force, in norm, generated by each propeller module is

$$F(\boldsymbol{x}) = k_l\Omega^2$$

System description: friction

For control purposes, the following continuous definition of the friction force is used:

$$\boldsymbol{F}_f(\boldsymbol{v}) = \begin{cases} \mu(|\boldsymbol{v}|)\frac{\boldsymbol{v}}{|\boldsymbol{v}|} & \boldsymbol{v} \neq 0 \\ 0 & v = 0 \end{cases}$$

with

$$\mu(s) = \cancel{\gamma_1\big(\tanh(\gamma_2 s) - \tanh(\gamma_3 s)\big)} + \gamma_4 \tanh(\gamma_5 s) + \cancel{\gamma_6 s}$$

System description: ROSPO model

The ROSPO system model is completed by considering the planar mechanics of the system:

$$\begin{cases} m\ddot{\boldsymbol{p}}_{com} & = \mathbf{R}(\psi)\sum_i F_{p,i}\hat{\boldsymbol{u}}(\phi_i) + \sum_j \boldsymbol{F}_{f,j} \\ J\ddot{\psi} & = \sum_i \boldsymbol{r}_i \times F_{p,i}\hat{\boldsymbol{u}}(\phi_i) + \sum_j \boldsymbol{r}_j \times \boldsymbol{F}_{f,j} \\ \dot{\boldsymbol{\phi}} & = \boldsymbol{\delta} \end{cases}$$

Finally the state $\boldsymbol{x}$ of the system comprises position and velocity of the platform and all the components for the N_t number of turrets. Inputs are voltages applied to each BLDC motor and the rotational speed of each turret module.

Controls: overview

The proposed solution use a hierarchical control structure:

- an high level controller converts the ROSPO state $\boldsymbol{x}$ and the planned motion $\boldsymbol{p}_{ref}$ information into a commanded virtual input $\boldsymbol{u}_{v,c}$, a wrench to be applied at the com;
- a neural network that can allocate the inputs $\boldsymbol{u}$ such that the system applies the desired wrench $\boldsymbol{u}_{v,c}$.

Controls: high level controller

Based on the reference paper, the high level controller perform a sort of feedback linearization of the non-linear system in order to read an equivalent first-order linear dynamics:

$$\boldsymbol{u}_{v,c} = \mathbf{B}^{-1}\left(\dot{\boldsymbol{u}}_v^\star - \mathbf{A}\boldsymbol{u}_v^\star\right) - \mathbf{K}\tilde{\boldsymbol{u}}_v$$

with

$$\boldsymbol{u}_v^\star = \begin{pmatrix} m\mathbf{R}^\top(\psi)\left(-k_p\tilde{\boldsymbol{p}} - k_d\dot{\tilde{\boldsymbol{p}}} + \ddot{\boldsymbol{p}}_{ref} - \frac{\boldsymbol{F}_f(\boldsymbol{x})}{m}\right) \\ J\left(-k_{p,\psi}\tilde{\psi} - k_{d,\psi}\dot{\tilde{\psi}} + \ddot{\psi}_{ref} - \frac{T_{f,\psi}(\boldsymbol{x})}{J}\right) \end{pmatrix}$$

and $\mathbf{A} = -\gamma_P\mathbf{I}_3$, $\mathbf{B} = \gamma_P\mathbf{I}_3$ and $\mathbf{K} \in \mathbb{R}^{3\times3}$ that stabilizes the linear system.

Controls: allocation overview

I solved the allocation problem by solving independently 2 subproblems:

- computing the voltage v to apply at the BLDC motor so that the module exerts a desired force F_{des};
- control the modules so that they asymptotically reach the desired applied wrench.

Both problems are solved using deep neural networks.

Controls: input voltage

The goal is to generate data on top of which a neural network can be trained to predict at runtime the voltage that should be applied to the motors.

The force F generated by the turret doesn't depend on the input v, but rather on the state $\boldsymbol{x}$ of the motor.

Since we can't optimize istantneously the force, integration of the model in time is performed and optimized against the input by solving the following problem:

$$v^\star(\boldsymbol{x}_0) = \arg\min_v \left\| F^+(v, \boldsymbol{x}_0) - F_{des} \right\|^2$$

Controls: input voltage

Given the optimization problem

$$v^\star(\boldsymbol{x}_0) = \arg\min_{v} \left\| F^+(v, \boldsymbol{x}_0) - F_{des} \right\|^2$$

the numerical solution is find more quickly and with higher reliability by providing the analytical jacobian of the cost function:

$$\frac{\mathrm{d}c}{\mathrm{d}v} = 2(F^+ - F_{des}) \frac{\partial F}{\partial \boldsymbol{x}} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}v}$$

Dataset is creating solving the problem sampling uniformly $(\boldsymbol{x}_0, F_{des})$ on a reasonable subspace $\mathcal{X} \times \mathcal{F}$.

Controls: allocation

The allocation problem is solved similarly. Choosing $\boldsymbol{x} = (F_1, \phi_1, \dots, F_{N_t}, \phi_{N_t})$ and $\boldsymbol{u}$ the component-wise time derivative of $\boldsymbol{x}$, then the allocation problem is solved in a similar fashion of the previous case:

$$\boldsymbol{u}^\star(\boldsymbol{x}_0) = \arg\min_{\boldsymbol{u}} \left\| \boldsymbol{F}_{com}(\boldsymbol{x}_0 + T_s \boldsymbol{u}) - \boldsymbol{F}_{des} \right\|^2_{\mathbf{W}_1} + k \|\boldsymbol{u}\|^2_{\mathbf{W}_2}$$

Development issues

The main issue in applying this technique is associated to the size of the problem.

When training a neural network that approximate the function $f : \mathbb{R}^3 \to \mathbb{R}$ that determines the input voltage, generating the dataset can be performed in a reasonable amount of time and training requires low resources.

Training the allocator requires approximating a function $f : \mathbb{R}^{11} \to \mathbb{R}^3$ (considering 4 turret modules). Sampling a space with such high dimensionality is a challenge: in 30 minutes 1 million samples are collected; training on such amount of data requires 1 hour but still the result is not satisfactory.

Results: force tracking

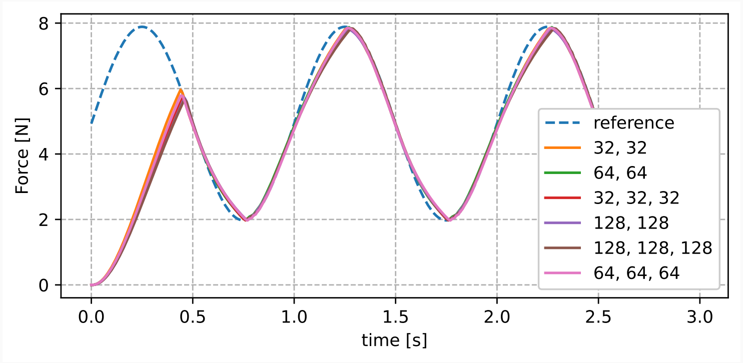

Figure 3:

Response of the neural networks controller in tracking a reference force.

Results: ROSPO trajectory tracking

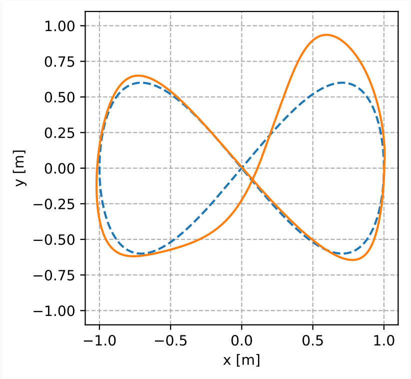

Figure 4:

Path followed by the ROSPO platform.

This image shows the path followed by the system while tracking a 8-shaped reference described as

$$\boldsymbol{p} = \begin{pmatrix} \rho_x \cos(c_1 t) \\ \rho_y \sin(c_2 t) \\ 0 \end{pmatrix}$$

Results: ROSPO trajectory tracking

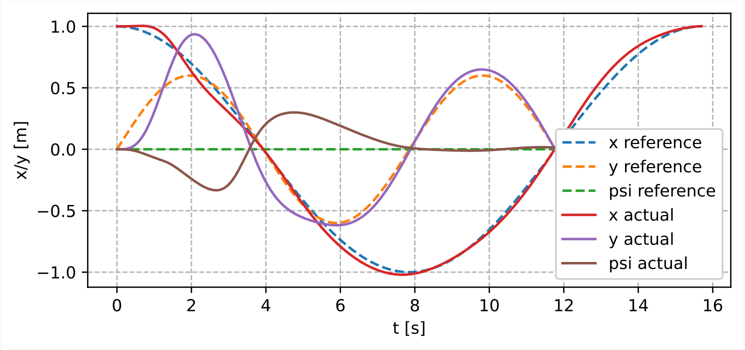

Figure 5:

Reference and actual position of the ROSPO platform during the simulation.

Conclusions

Using deep neural networks to fit the optimal allocation function is non optimal due to the high dimensionality of the problem involved and the numerical complexity that comes with it.

More complex data-driven control approach might use:

- reinforcement learning in order to train the system at runtime;
- recurrent neural networks trained on dataset generated with model predictive controls.