FoReCo
A forecast-based recovery mechanism for real-time remote control of robotic manipulator

Milan Groshev, Javier Sacido & Jorge Martín-Pérez
Universidad Carlos III de Madrid
Telcaria Ideas S.L.
mgroshev@pa.uc3m.es

Abstract
This demonstration presents FoReCo [1], a solution to recover lost control commands in remotely controlled robots. In the demonstration, visitors use a joystick to remotely control a robotic arm under the presence of packet losses in the wireless medium. The lost control commands result in a distorted trajectory of the robotic arm, thus, we deploy FoReCo to recover lost control commands using an ML model that we train with a real-world dataset. The demonstration shows how FoReCo recovers the lost commands, and how the robot arm operates smoothly despite the losses that are present in the wireless medium.

Introduction

Real-time remote control of robotic manipulators is of high interest for industry 4.0 use cases, since it reduces costs and the exposure of operators to hazard situations. However, the presence of packet losses in the the network lead to trajectory deviations that prevent the robot from achieving the expected 99.9999% reliability. The mechatronics and robotic community propose to overcome the lost/delayed control commands using time domain passivity-based [4], and wave-variable passivity-based [5] approaches that assume constant network delays [3], or delays with small variability [4, 5]. Such assumptions on the network delay do not apply for robots connected using IEEE 802.11 wireless technology, which suffers from uncontrolled packet losses and delays. Specially in industrial scenarios with electromagnetic interference jamming the wireless channel.

FoReCo solution

FoReCo [1] does not make assumptions on the network delays and feeds the robot drivers with the lost/delayed commands it recovers through forecasting – see Figure 1. FoReCo receives the remote control commands and checks if they are received at the expected frequency, e.g., each 150ms. In case a command does not arrive on time (due to 802.11 collisions or interference not recovered with error-correction), FoReCo forecasts/infers the out of time command using the recent command history.

In this demonstration FoReCo uses a multinomial logistic regression that we train using a real-world dataset of an experienced robotic arm operator. The ML model receives as **input**:

1. the **position** of each joint $j_i(t), \forall i$;
2. for **how long** each joint has been moving without stopping $\max_{t_0<t}\{t-t_0:\ j_i(\tau)>0, \forall\tau\in[t_0,t]\}, \ \forall i$; and
3. the **angular derivative** of each joint $\frac{d}{d\phi}j_i(t), \forall i$.

Using the input (1)-(3), FoReCo runs the multinomial logistic regression and **outputs** if the lost command was an up/down movement, a righ/left sweep, or a grab/release action. The repetitive pick-and-place task fosters FoReCo's accuracy when it forecasts lost remote control commands.

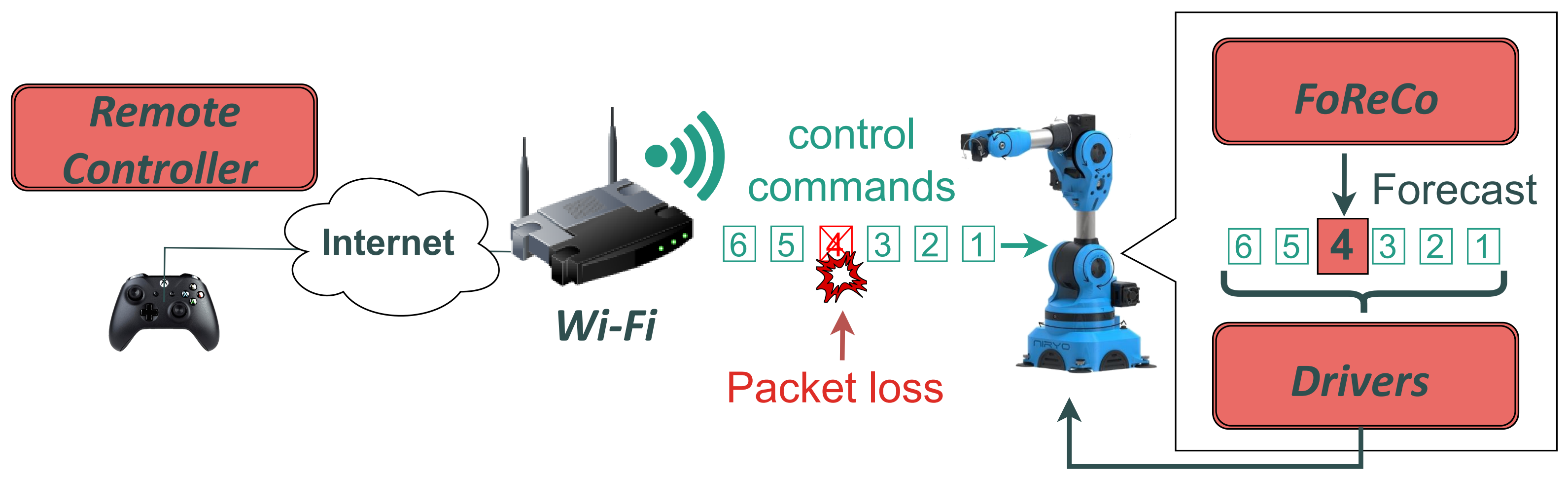

Figure 1: FoReCo infers control commands lost in connection between the remote controller and the robotic arm.

FoReCo deployment

In order to deploy FoReCo we create a Network Service (NS) with six Virtual Network Functions (VNFs) illustrated as red boxes in Figure 2: (1) the **FoReCo** VNF to assess the command recovery; (2) the robot **drivers** VNF to control and monitor the robotic arm hardware; (3) the **niryo-one-interface** and (4) **niryo-one-motion** to provide high-level abstraction for the core robot functionalities; (5) the **niryo-one-web** VNF to provide a set of tools (e.g., web interface, joystick/automated robot control, calibration) that facilitate the user interaction with the robot; and (6) the **niryo-ros-master** as a central entity to coordinate the communication among VNFs. The niryo-one-web/interface/motion/master and the drivers VNFs are implemented using the Robot Operating System (ROS)and the FoReCo VNF is implemented using scikit-learnto forecast/infer lost commands. The resulting NS with the aforementioned VNFs is deployed on top of a joint fog05/Kubernetes cluster where the robot drivers VNF is deployed as a ROS native app using fog05. The niryo-one-web/interface/motion/master are deployed as containers using Kubernetes.

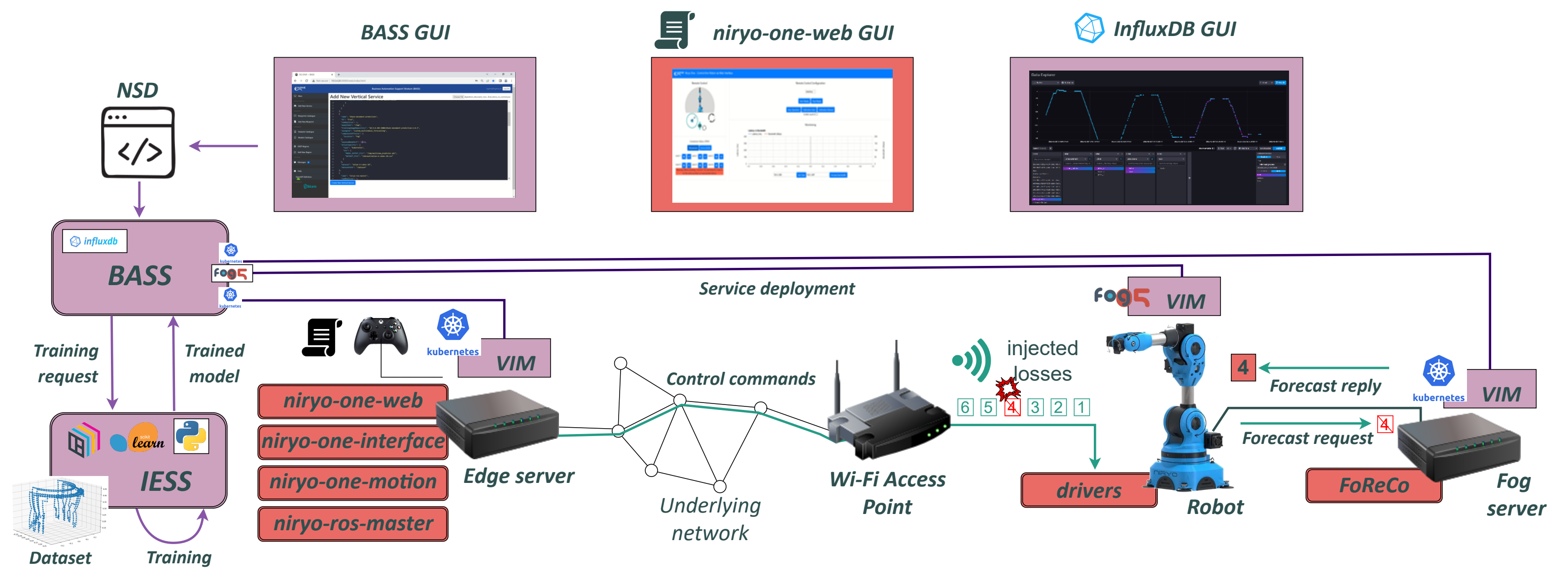

Figure 2: FoReCo-assisted remote control (red) deployed with DEEP modules (purple). Three GUIs are used to deploy the service (left), calibrate/control the robot (middle), and visualize FoReCo command recovery (right).

Once FoReCo is packed inside the aforementioned NS, we use the DEEP platform [2] for its automated ML training, deployment, life-cycle management, and termination. While the BASS module of the DEEP simplifies and automates the creation and management of the NS by being the logical entry point for the user, the IESS module facilitates the training and provisioning of FoReCo by providing an intent-driven approach and AutoAI solutions. The deployment through the DEEP platform proofs that FoReCo is ready to be integrated in real world platforms.

FoReCo results

Decrease error by a 95%

To assess the performance evaluation of FoReCo, we have tested it in a simulation environment that follows a mathematical model for IEEE 802.11 packet losses upon interference [1]. The considered model allows to increase the number of robots present in a robot factory and study the impact of packet collisions in the shared wireless medium. Figure 3 illustrates the average trajectory deviation of 25 robot manipulators that are remotely controlled in a factory floor.

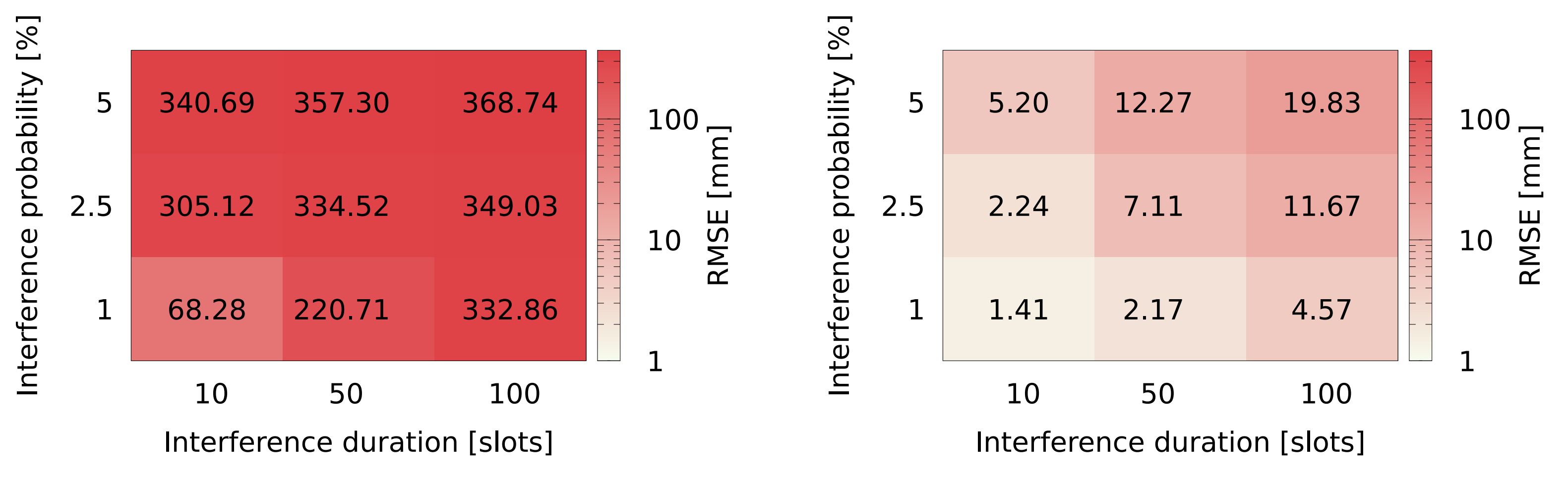

Figure 3: Trajectory error as the interference duration and probability increase with or without FoReCo (right and left).

Smoothness upon losses & interference

The experimental validation [1] shows that FoReCo reduces the trajectory error by guessing the lost control commands, yet preserving the smoothness of the remote control. Figure 4 shows how FoReCo kept as smooth control even with periodic jams on the IEEE 802.11 channel. On top, Figure 4 shows that once the channel recovers the PID control of the state of the art solution suffers from a ping-pong effect, whilst FoReCo keeps interacting with the drivers with a stable feed of control commands that prevent rotors' damage.

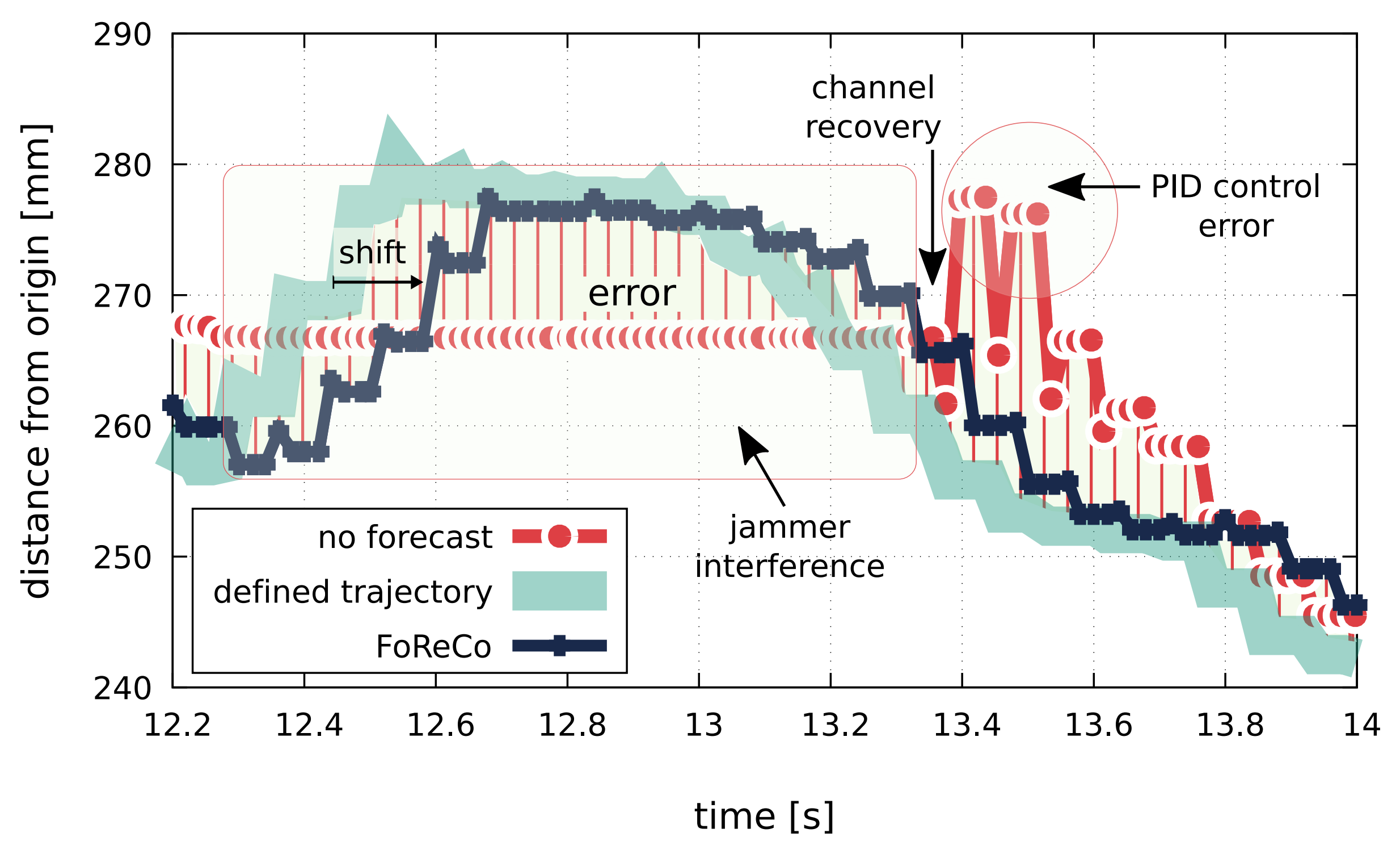

Figure 4: Robot trajectory upon IEEE 802.11 jammer interference

References

[1] Milan Groshev et al. "FoReCo: a forecast-based recovery mechanism for real-time remote control of robotic manipulators". In: *IEEE Transactions on Network and Service Management* (2022), pp. 1–1. DOI: 10.1109/TNSM.2022.3173436.

[2] Carlos Guimarães et al. "DEEP: A Vertical-Oriented Intelligent and Automated Platform for the Edge and Fog". In: *IEEE Communications Magazine* 59.6 (2021), pp. 66–72. DOI: 10.1109/MCOM.001.2000986.

[3] Christian Ott et al. "Subspace-oriented energy distribution for the Time Domain Passivity Approach". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 665–671. DOI: 10.1109/IROS.2011.6094697.

[4] Da Sun et al. "Neural Network-Based Passivity Control of Teleoperation System Under Time-Varying Delays". In: *IEEE Transactions on Cybernetics* 47.7 (2017), pp. 1666–1680. DOI: 10.1109/TCYB.2016.2554630.

[5] Da Sun et al. "Wave-Variable-Based Passivity Control of Four-Channel Nonlinear Bilateral Teleoperation System Under Time Delays". In: *IEEE/ASME Transactions on Mechatronics* 21.1 (2016), pp. 238–253. DOI: 10.1109/TMECH.2015.2442586.

Acknowledgements

This work has been partly funded by the European Commission through the H2020 project Hexa-X (Grant Agreement no. 101015956), and the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D 6G-EDGEDT and 6G-DATADRIVEN.