

Model inference over 5G networks for robotics



Project Title	AI4Europe
Contract N°.	101070000
Type of Action	Horizon CSA
Topic	HORIZON-CL4-2021-HUMAN-01-02
Project start date	1 July 2022
Duration	42 months



Funded by
the European Union

Title	Model inference over 5G networks for robotics
Actual date of delivery	18th December of 2025
Nature of deliverable	Description of experiments on 5G networks and model inference for robotics within the scope of T5.2
Dissemination level	Public
Work Package	WP5
Task(s)	T5.2
Partner responsible	Instituto Tecnológico de Informática
Author(s)	Andrés Meseguer Valenzuela

Abstract	This work studies compute offloading for mobile-robotics perception by abstracting CNN inference from the robot to edge and remote servers within a ROS 2 pipeline. The OMZ person-detection-0200 model is executed using OpenVINO, with video streamed via GStreamer over a private 5G network. Three deployments are compared: on-board inference, edge inference, and remote server inference. Average inference time decreases from 5.177 ms (on-board) to 4.998 ms (edge) and 1.291 ms (remote), while recommended throughput increases from 35.088 FPS to 41.858 FPS and 83.612 FPS. Results confirm that compute placement strongly impacts both latency and achievable frame rate, and that tail behaviour must be considered alongside central tendency when selecting an execution locus.
Keywords	Mobile robotics, Edge computing, Private 5G

Copyright

© Copyright 2022 AI4Europe

This document may not be copied, reproduced, or modified in whole or in part for any purpose without written permission from the AI4Europe. In addition to such written permission to copy, reproduce, or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

All rights reserved.





Contributors

NAME	ORGANISATION
ANDRÉS MESEGUER VALENZUELA	INSTITUTO TECNOLÓGICO DE INFORMÁTICA
JOSE VERA PÉREZ	INSTITUTO TECNOLÓGICO DE INFORMÁTICA
SALVADOR SANTONJA CLIMENT	INSTITUTO TECNOLÓGICO DE INFORMÁTICA

The information and views set out in this report are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf.

Acknowledgement

The authors would like to express their sincere gratitude to Santiago David Gálvez, Antoni Gimeno Bono and Sonia Santiago Pinazo for their valuable support throughout the project and for their insightful guidance and advice.

Table of Abbreviations and Acronyms

Abbreviation/Acronym	Open form
5G	Fifth Generation (mobile network)
AI	Artificial Intelligence
AIoD	AI-on-Demand
ARM	Advanced RISC Machines (ARM architecture)
BGR	Blue–Green–Red (channel ordering)
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
DDS	Data Distribution Service
FLOPs	Floating-Point Operations
FP16	16-bit Floating Point
FP32	32-bit Floating Point
FPS	Frames Per Second
GFLOPs	Giga Floating-Point Operations
GPU	Graphics Processing Unit
IP	Internet Protocol
IR	Intermediate Representation (OpenVINO IR)
IoU	Intersection over Union
LGPL	GNU Lesser General Public License
LPDDR4x	Low-Power Double Data Rate 4X
NVDLA	NVIDIA Deep Learning Accelerator
OMZ	Open Model Zoo
ONNX	Open Neural Network Exchange
QoS	Quality of Service
ROS2	Robot Operating System 2
RTI	Real-Time Innovations (RTI Connex)
RTPS	Real-Time Publish-Subscribe
RTSP	Real Time Streaming Protocol
RTT	Round-Trip Time
SINR	Signal-to-Interference-plus-Noise Ratio
SoC	System on Chip
SSD	Single Shot Detector
TF	TensorFlow
TFLite	TensorFlow Lite
TOPS	Tera Operations Per Second
vCPU	virtual CPU
VPN	Virtual Private Network
WAN	Wide Area Network

Index of Contents

1. Introduction	7
2. Evaluation Description	10
2.1 First Experiment. Local capture and local inference.	10
2.2 Second Experiment. Edge offload over private 5G.	10
2.3 Third Experiment. Remote offload via the same access.	10
2.4 Infrastructure	11
3. Results	14
4. Conclusions	15
Consortium	16

Index of Figures

Figure 1. Robot used for the experiments.	11
Figure 2: 5G network deployed	12
Figure 3 RTT values comparison	13
Figure 4 Throughput behaviour	13

Index of Tables

Table 1. AI catalogues	8
Table 2. Hardware equipment	12
Table 3. Results obtained in terms of inference time.	14
Table 4. Recommended values for FPS configuration	14

1. Introduction

Modern mobile-robotics applications increasingly hinge on machine-learning techniques whose computational demands steadily rise. As algorithms evolve toward heavier architectures—ranging from compact CNNs to large language models—onboard compute becomes a bottleneck: embedded processors must deliver higher throughput without exhausting the power budget that safeguards mission autonomy. In practice, chasing accuracy with larger models often taxes energy and thermal envelopes, eroding time-between-charges and elongating the application’s end-to-end response, i.e., the actionable latency observable at the robot level. This tension motivates an architectural pivot: instead of insisting that all inference run locally, heavy workloads can be abstracted and offloaded to more capable processors when appropriate. Those processors may live in public cloud infrastructure or at the edge, i.e., in servers on the robot’s local network; the latter option typically reduces network traversal and improves privacy.

Within this landscape, the software backbone is ROS 2, the standard middleware for modern robotic systems. ROS 2 unifies data exchange via DDS (Data Distribution Service), allowing developers to compose sensing, perception, and control graphs with consistent Quality-of-Service semantics. Multiple DDS/RTPS vendors are supported—e.g., eProsima Fast DDS (default in current releases), RTI Connex, Eclipse Cyclone DDS, and GurumDDS—so deployments can select a stack that matches their determinism and footprint requirements. Historical design notes detail why DDS was selected as ROS 2’s communication substrate.

Computer vision tooling plays a parallel enabling role. OpenCV¹, the open-source computer-vision library today operated by the non-profit Open-Source Vision Foundation and maintained by a large community. It offers a rich primitives set for acquisition, image processing, and quick prototyping. GStreamer² complements that layer as a pipeline-based multimedia framework used widely for capture, processing, and streaming; it is released under the LGPL, making it a pragmatic choice for low-latency media transport in robotics.

To reconcile algorithmic ambition with embedded constraints, inference is executed with OpenVINO, an open-source toolkit that converts, optimizes, and runs models efficiently across Intel® CPUs and GPUs—both in cloud and at the edge. OpenVINO’s³ ecosystem includes the Open Model Zoo (OMZ)⁴, a curated set of pre-trained models and demonstration pipelines designed for straightforward deployment on Intel hardware while remaining framework-agnostic at the model-input boundary.

Beyond OMZ, several widely used model catalogues are relevant when assembling edge-deployable inference stacks. A comparative view is provided to make explicit their scope and typical artifacts:

¹ <https://opencv.org>. Accessed on December 16th, 2025.

² <https://gststreamer.freedesktop.org/>. Accessed on December 16th, 2025.

³ <https://docs.openvino.ai>. Accessed on December 16th, 2025.

⁴ https://docs.openvino.ai/2023.3/model_zoo.html. Accessed on December 16th, 2025.

Catalogue	Focus	Typical formats	Edge-readiness
Open Model Zoo (OMZ)	Models and demos tailored for OpenVINO runtimes	OpenVINO IR, ONNX, links to sources	Strong (IR + reference pipelines)
AI-on-Demand (AloD) ⁵	EU portal aggregating AI assets	Mixed	Varies (per asset)
ONNX Model Zoo ⁶	Community ONNX models	ONNX	High (ONNX by design)
TensorFlow Hub ⁷	Reusable TF models	SavedModel, TF.js/TFLite	Medium–High (TFLite/TF.js avail.)
PyTorch Hub ⁸	Research-oriented PyTorch models	PyTorch; ONNX export common	Medium (export needed)
Hugging Face Hub ⁹	Large general-purpose hub	Multiple (PyTorch/TF/ONNX, etc.)	Often high (many ONNX exports)

Table 1. AI catalogues

A practical path forward is to treat perception as a distributed service whose locus of execution can be shifted—on demand—between onboard, edge, and cloud resources while preserving ROS 2 semantics at the application boundary. In this view, the camera, pre-processing, inference, and post-processing stages are composed as a graph with explicit timing points, so that end-to-end latency can be decomposed into capture, encode, transport, decode, model execution, and rendering/actuation components. The benefit of such decomposition is twofold: it enables engineering decisions grounded in measured bottlenecks rather than intuition, and it clarifies which optimizations most effectively reduce the “actionable latency” perceived at the robot level.

Against that backdrop, the present work focuses on a single, representative perception primitive—person detection—as an anchor task to study offloading trade-offs without confounding variables from multi-class pipelines. An Open Model Zoo model is selected due to a lightweight backbone with well-documented inputs/outputs and mature OpenVINO integration, making it appropriate for embedded execution while being sufficiently expressive to stress the transport and scheduling layers. The resulting setup allows isolating where computation belongs under different network and hardware constraints, and how much “distance” between sensor and accelerator can be introduced before the control loop is measurably impaired.

The experimental design therefore contrasts three canonical placements of the inference stage: (i) on-robot, where frames are captured and inferred locally on an embedded SoC; (ii) edge, where frames traverse a private 5G access and are inferred on a server located

⁵ <https://aiod.eu/get-started/> Accessed on December 16th, 2025.

⁶ <https://onnx.ai/models/> Accessed on December 16th, 2025.

⁷ <https://www.tensorflow.org/hub> Accessed on December 16th, 2025.

⁸ <https://pytorch.org/hub/> Accessed on December 16th, 2025.

⁹ <https://huggingface.co/docs/hub/index> Accessed on December 16th, 2025.

within the same administrative domain and IP segment; and (iii) remote cloud/campus, where frames traverse the same access link but add a wide-area hop to a server in a data-center environment. All three paths retain the same model, pre-/post-processing, and application logic. Only the locus of inference—and thereby the transport distance and compute envelope—changes. This symmetry is intentional: it ensures that observed differences in frame rate and latency can be traced to compute and network placement, not to model or software drift.

2. Evaluation Description

Three experimental scenarios are defined to probe the latency–throughput landscape of local vs. offloaded inference, holding the model constant and varying only where inference runs and how media is transported.

2.1 First Experiment. Local capture and local inference.

The first experiment keeps the entire perception loop on board in the robot. An RGB camera mounted on the platform streams frames that are acquired locally through OpenCV, with no encoder or network hop in the path. Each frame is analyzed internally following a process that starts with the image capture, minimal pre-processing (e.g., color conversion and resize to the model’s native 256×256), inference, and lightweight post-processing to filter detections in addition to draw the human detections.

The evaluation gravitates around a single, well-scoped model: OMZ’s person-detection-0200. Rather than casting a wide net, it focuses exclusively on spotting people, coupling a lean MobileNetV2 backbone with two SSD heads that read from the 1/16 and 1/8 feature maps and rely on clustered prior boxes. The model consumes a 1×3×256×256 BGR tensor and produces detections using the following format: *image_id*, *label*, *confidence*, *x_min*, *y_min*, *x_max*, *y_max*.

Previous validation¹⁰ reports a per-inference computational cost of ≈0.786 GFLOPs (for a 256×256 input, under standard FLOP counting) and 1.817 million parameters. In practical terms, this places the network in the “compact” range: the weight footprint is roughly 7.3 MB at FP32 (≈3.7 MB at FP16), which is friendly to embedded memory budgets and cache behavior. The stated accuracy—AP@[0.50:0.95] = 0.2398—corresponds to the COCO-style average precision averaged over IoU thresholds from 0.50 to 0.95 in 0.05 increments for the single person class, reflecting the expected speed–accuracy trade-off of a lightweight detector. Given this profile, the architecture aligns well with near-real-time operation at moderate resolutions on mobile platforms.

2.2 Second Experiment. Edge offload over private 5G.

The second experiment starts with the image capturing with the camera mounted on the robot. However, in this case the image is streamed directly with 5G Modem through a 5G private network. Frames are streamed to an edge server directly attached to that network using RTSP within GStreamer pipeline. The server receives the streaming using a python script with gstreamer, to bridge the application with OpenCV and OpenVINO.

2.3 Third Experiment. Remote offload via the same access.

Similarly to the previous experiment, the third evaluation is based on the streaming of the camera data through the 5G private network and public network used to connect to

¹⁰ https://docs.openvino.ai/2023.3/omz_models_model_person_detection_0200.html Accessed on December 16th, 2025.

a second site (with VPN). This experiment is designed to apply the human detection within a server located in the second site, with a larger computing capacity.

2.4 Infrastructure

The robot platform used for the evaluation is an X100 (XMachines¹¹) which runs ROS 2 Humble¹², equipped with an NVIDIA® Jetson Xavier™ NX module. The Xavier NX¹³ provides up to 21 TOPS (INT8), integrating a 384-core Volta GPU with 48 Tensor Cores, a 6-core Carmel ARMv8.2 CPU, and dual NVDLA engines—capabilities that make it suitable for multi-stream perception under tight power limits.



Figure 1. Robot used for the experiments.

For the image acquisition, an Intel® RealSense™ D455 RGB-D camera is used. The D455¹⁴ extends the stereo baseline to 95 mm, reducing Z-error to < 2% at 4 m; the RGB sensor employs a global shutter aligned with the depth FOV ($\approx 86^\circ \times 57^\circ$), and the depth stream reaches 1280×720 @ 90 FPS within a 0.6–6 m operating range—practical characteristics for mobile perception and human-presence detection.

The following table summarizes the infrastructure used during the evaluation:

¹¹ <https://www.xmachines.ai/> Accessed on December 16th, 2025.

¹² <https://docs.ros.org/en/humble/index.html> Accessed on December 16th, 2025.

¹³ <https://developer.nvidia.com/blog/jetson-xavier-nx-the-worlds-smallest-ai-supercomputer/> Accessed on December 16th, 2025.

¹⁴ <https://www.intel.la/content/www/xl/es/products/sku/205847/intel-realsense-depth-camera-d455/specifications.html> Accessed on December 16th, 2025.

Target	CPU	Accelerator / GPU	Memory	Location / notes
On-robot (Jetson Xavier NX)	6-core Carmel ARMv8.2	384-core Volta GPU + 48 Tensor Cores; up to 21 TOPS (INT8)	8–16 GB LPDDR4x (module-dependent)	Mounted on robot; ROS 2 Humble. (NVIDIA)
Edge server	11th-Gen Intel® Core™ i7-1165G7 @ 2.80 GHz	—	16 GB RAM	Same private network as robot. (Intel)
Cloud server	12 vCPUs @ 2.1 GHz	Tesla T4 16GB	20 GB RAM	Hosted remotely.

Table 2. Hardware equipment

In terms of wireless communications, a 5G private network is used to transmit information from the mobile robot. This network is allocated in 40 band (2350 to 2390MHz) and makes usage of Amarisoft Core Network and radio software. For this aim, an Amarisoft Callbox Mini¹⁵ is utilized to deploy the network satisfactorily.

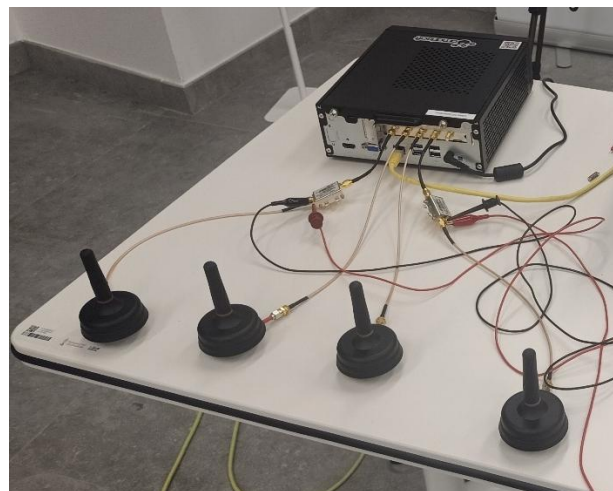


Figure 2: 5G network deployed

The private 5G network delivers latency suitable for the intended workload. Figure 3 summarizes latency as a function of radio link quality (SINR): at SINR \approx 7–6 (dB), the round-trip latency remains below 20 ms with similar distributions. As SINR approaches the noise floor, however, latency increases markedly and becomes more variable, consistent with degraded radio conditions.

¹⁵ <https://www.amarisoft.com/test-and-measurement/device-testing/device-products/amari-callbox-mini> Accessed on December 16th, 2025.

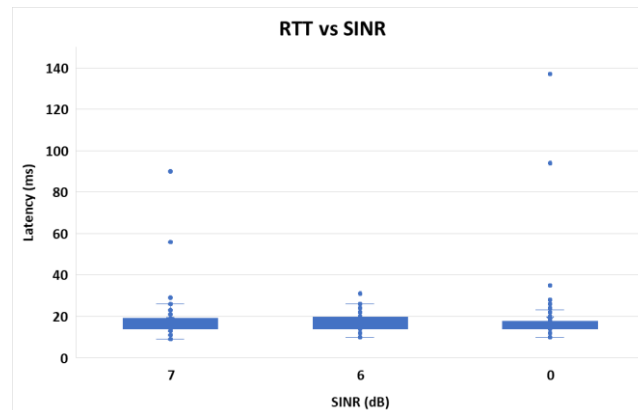


Figure 3 RTT values comparison

In terms of throughput, the radio configuration is designed for uplink transmission as the camera is transmitted towards the core network. Following this approach, the 5G private network achieves a throughput above 40Mbps in Uplink. The following figure showcases the behavior of the bitrate depending on the SINR.

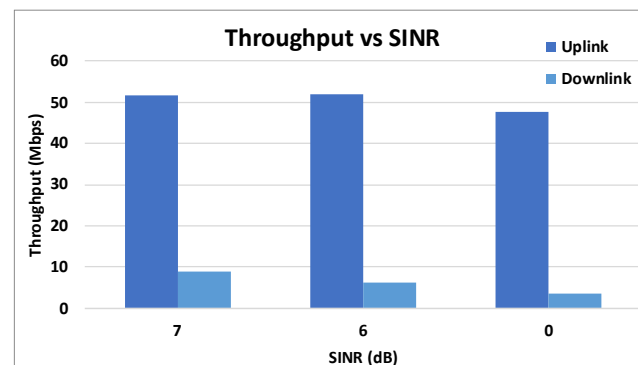


Figure 4 Throughput behaviour

As depicted in Figure 4, the loss of signal quality affects mainly to downlink stream. However, the 5G private network is validated and provides suitable performance for the transmission of an RGB camera which requires from 10.3Mbps considering a resolution of 1080p and H.264 compression.

In addition to the 5G private network, data must be transmitted through a WAN in the third experiment to receive the streaming at a remote server in a second site. For this aim, a test is performed to measure the latency between the 5g private network site (Paterna, Spain), and the second site (located in Valencia, Spain). The results showed that the RTT between both sites was 17.47ms (mean), 17.00 (median) and 0.89ms (Standard Deviation). Therefore, the connectivity is suitable for the transmission use case.

3. Results

The three experiments detailed in the previous section were executed achieving results in terms of inference time. Table 1 reports the model-only inference time statistics for the three execution placements

Inference Time (ms)	First Experiment	Second Experiment	Third Experiment
Average	5.177	4.998	1.291
Median	5.138	4.218	1.070
Maximum	28.499	23.890	11.960

Table 3. Results obtained in terms of inference time.

Narratively, the trend is unambiguous: the Third Experiment achieves the lowest central latency and the tightest tail, with an average of 1.291 ms and a maximum of 11.960 ms. Relative to the First Experiment, this represents a ~75% reduction in mean latency and a ~58% reduction in the observed maximum. The Second Experiment offers a modest improvement over the First in mean latency (~3.5% lower) but a much larger gain in the median (~18% lower), indicating fewer typical delays despite occasional spikes.

These latency differences translate into higher sustainable throughput: recommended frame rates rise from 35.088 FPS (First) to 41.858 FPS (Second, ~19% higher) and to 83.612 FPS (Third, ~138% higher than First and ~100% higher than Second). The gap between mean and median in the First and Second experiments signals tail latency episodes that depress effective throughput; by contrast, the Third experiment's lower maximum suggests more predictable execution and thus a higher safe FPS cap. The recommended values for FPS are defined in the following table:

	First Experiment	Second Experiment	Third Experiment
Recommended FPS	35.088	41.858	83.612

Table 4. Recommended values for FPS configuration

In practical terms, the results indicate that moving inference to the compute-rich environment of the Third scenario yields the best combination of low central latency, reduced worst-case delays, and highest stable FPS. The Second scenario remains a balanced choice when minimizing wide-area traversal is important, while the First scenario offers full autonomy at the expense of lower throughput and more pronounced latency tails.

4. Conclusions

In conclusion, the measurements indicate that relocating inference to the highest-capacity compute target (Third Experiment) provides the strongest performance–stability trade-off: the average inference time drops to 1.291 ms ($\approx 75\%$ lower than the fully local baseline) and the recommended throughput rises to 83.612 FPS ($\approx 138\%$ higher than 35.088 FPS in the First Experiment). Edge offloading (Second Experiment) also improves typical behaviour, reducing the median inference time by $\approx 18\%$ (from 5.138 ms to 4.218 ms) and increasing the recommended frame rate to 41.858 FPS ($\approx 19\%$ higher than the local case), although non-negligible tail events remain visible when compared to the third scenario.

Overall, the results support a compute-abstraction/offloading strategy whenever connectivity is controlled and stable, while preserving on-board execution as a robust fallback for disconnected or privacy-constrained conditions; importantly, placement decisions should be driven by end-to-end latency and variability rather than inference time alone, since network transport can introduce additional delay and jitter.

Consortium



