Contents

1. Delimitation about the problem

Currently, GALP service station clients are presented with conventional, generic, and static vehicle services. Taking the car washing services into mind, we find these services lacking in many ways.

Considering the importance of sustainability, connectivity, flexibility of services, intelligent commodities, we find that it is highly relevant to create a new GALP service that can present better and newer services, improving not only the client's life, but also its relationship with GALP.

2. Hypothesis of solution

CUIDA – Car Unified Inspection for Damage and Dirt with AI, is an AI-based station inspection system. CUIDA creates new, attractive, and custom offers, promoting sustainable, cost-efficient cleaning and repairing services. Connecting stations with GALP network partners using a new shared service paradigm.

3. Team members

João Borges – Project management, mobile app development.

César Melo – AI development and Edge-AI implementation.

António Moreira – Project management, MQTT and XDK implementation.

José Barradas – GALP strategy planning.

Cristina Coutinho – GALP mobile app and UI/UX design.

4. Team strategy

The overall CUIDA system is comprised by an inspection chamber which has 3 iteratives stages, as shown in Figure 1:

1. **(E1)** – Visual dirt inspection – Once the vehicle enters the inspection system, a sensor detects the vehicle, **S1,** and triggers the cameras, **C1**, and an AI algorithm which detects the dirtiness level in different car parts, **N1**. After the inspection, the client is presented with a suggestion of different car washing services, through a cell phone application, **J1.**
2. **(E2) –** Car washing system – After the client finishes the selection of car washing services, the vehicle enters the washing system for a customized washing procedure.
3. **(E3) –** After the vehicle is cleaned, a second inspection system is triggered, through a second sensor, **S2.** A second AI algorithm detects different damaged parts, **N1**, which then enables the cell phone application to present different car repair services from GALP partner network, **J1**.

Each AI algorithm was based on a supervised learning procedure, thus datasets were the main hindering factor to reach higher TRLs. Considering the lack of public datasets with context to the project scope, and the lack of time, the team focused on implementing the full development pipeline, aiming for a maturity level of TRL4.

The computational concept follows an Edge-AI approach, where each AI algorithm is computed in a Jetson Xavier AGX, **N1**.

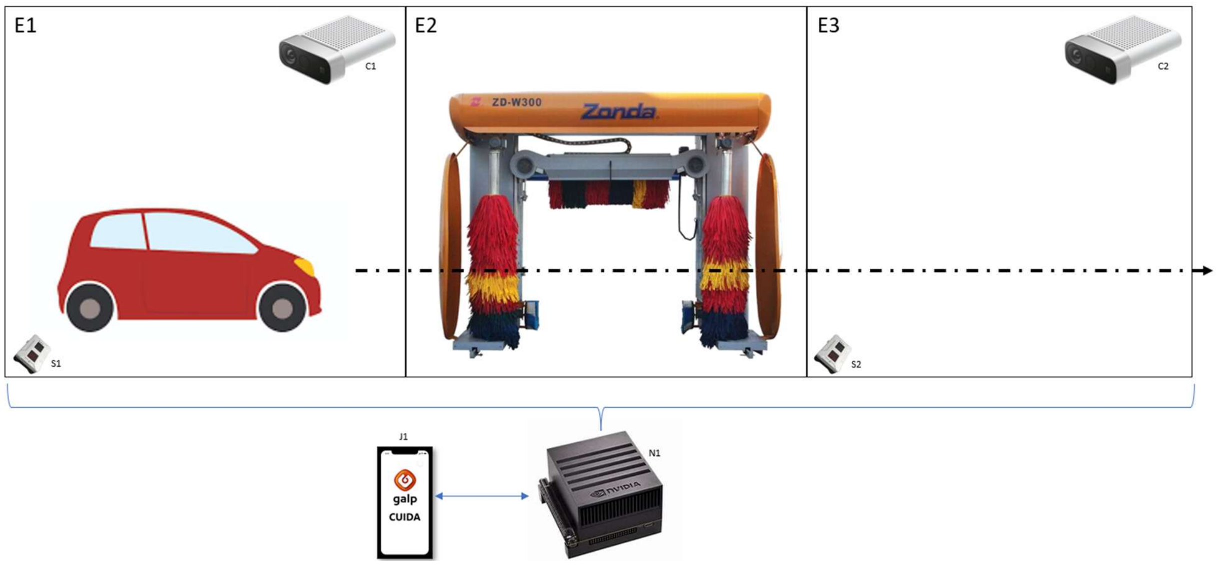

Figure 1 - CUIDA inspection pipeline

5. Technologic pre-requisites

For the development of the presented strategy, some technologies and tools were used:

- MQTT (Message Queuing Telemetry Transport) – Open source message protocol for sensors and IoT devices, based on a publish/subscribe architecture. This protocol was used for the communication between the sensors, the cell phone, and the Jetson, via messages sent to two different topics (cuida – for messages regarding if it is needed a damage or dirt inspection; cuidapp – for messages with information regarding the results from the algorithms inference).

- Pytorch – Open-source machine learning framework, used for computer vision and natural language applications, designed to provide good flexibility and high speed for deep neural networks. Used for the YOLOv5l training task.

- YOLOv5 (You Only Look Once) – Open-source object detection model, based on pytorch framework, known for low computational requirements and good metrics. Used architecture for the damage and dirt detection models generation.

- Open-CV – Open source machine learning library, used for computer vision applications. Used to read the images provided by the cameras, for algorithmic inference.

- CUDA – Open source parallel computing platform created by NVIDIA, that increases the power of GPUs and the speed of applications. It was implemented on the embedded system Jetson Xavier AGX.

- Jetson Xavier AGX – NVIDIA Edge-AI computational system, using NVIDIA Jetpack 4.6.1.

6. Used Data sets

Like it was said before, there is a lack of public datasets associated with both problematics, car damage and dirt detection. In the following, are presented the features for each dataset, used for algorithmic training task.

6.1. Dirt Dataset Generation

To aid the dirt detection algorithm to understand the presence of dirt in a vehicle, images were collected and aggregated from the following websites:

https://stock.adobe.com/ ; https://www.pexels.com/ ; https://pixabay.com/ ; https://unsplash.com/

Since it was also necessary images associated to cleaned cars, some samples from the dataset available in https://www.kaggle.com/datasets/anujms/car-damage-detection were collected too.

Because some of the images did not fit in the context, and due to the lack of pretended annotations alongside the images, it was generated a second version of each of these datasets, where the labels were generated manually using a tool for bounding boxes label generation, with the following classes: (0) Wheels Dirt; (1) Wheels Clean; (2) Lateral Dirt; (3) Lateral Clean; (4) Top Dirt; (5) Top Clean (as shown in Figure 2). These classes were defined to allow the car washing system to provide 3 customized cleaning services (i.e. wheels, lateral, top). The final dataset is comprised by 208 samples.

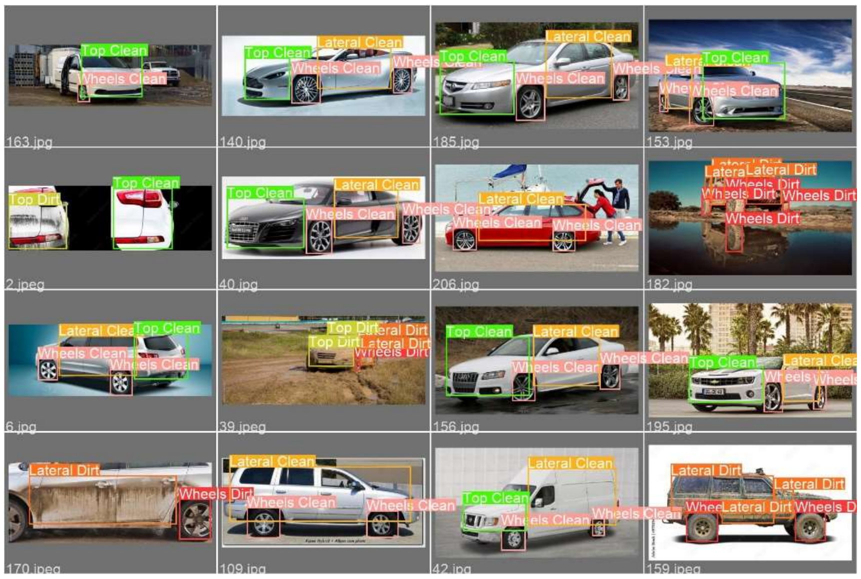

Figure 2 - Vehicle dirt detection dataset

6.2. Damage dataset generation

In a similar fashion as the previous section, public datasets were collected and labeled from the websites:

https://peltarion.com/knowledge-center/documentation/terms/dataset-licenses/car-damage

https://www.kaggle.com/datasets/lplenka/coco-car-damage-detection-dataset

https://www.kaggle.com/datasets/anujms/car-damage-detection

The final dataset reached a total of 1087 samples, as shown in Figure 3. Moreover, classes were focused on different types of repairing shops: (0) scratch; (1) broken glass; (2) deformation; (3) broken.

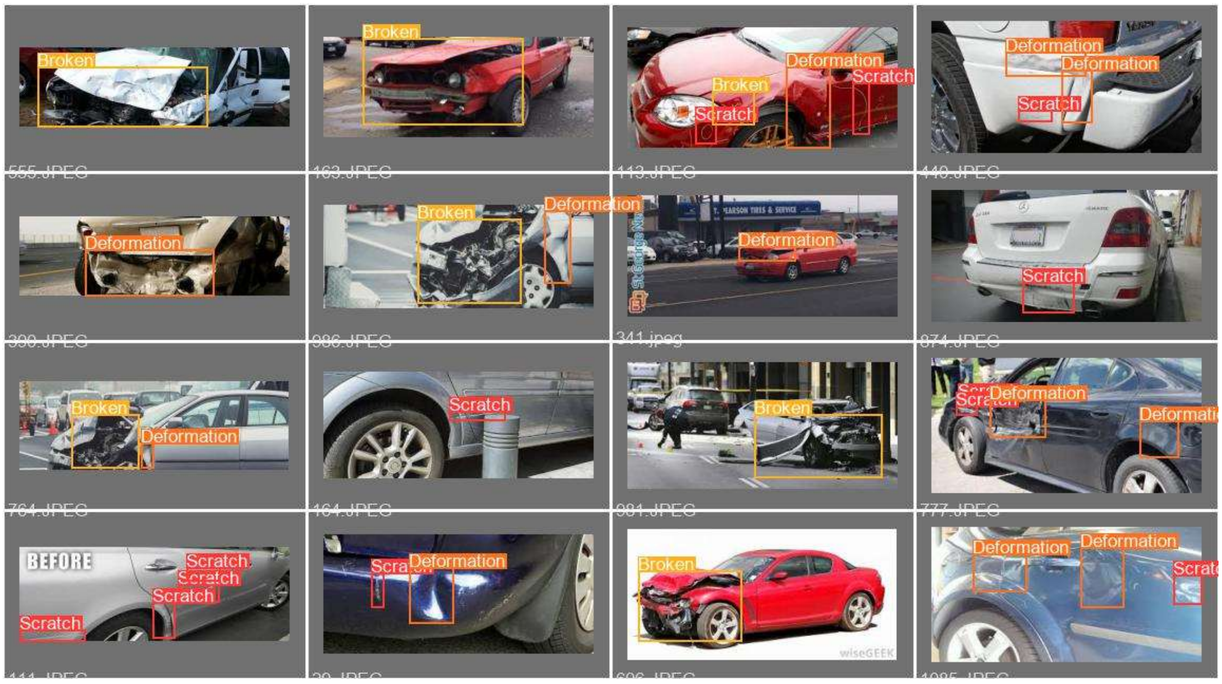

Figure 3 - Vehicle damage detection dataset

7. How the solution can be used

In this section the Edge-AI execution of the CUIDA system will be presented.

- When the sensor S1 detects the vehicle, a "[XDK_DIRT]:0" message is sent from the sensor via MQTT to topic "cuida", informing the system to do a dirt inspection;

- Depending on the results obtained from the "dirt_best.pt" model inference, different messages are sent from the system to the mobile app, via "cuidapp":
  - If dirt on the wheels detected: "[JETSON_DIRT_R]";
  - If dirt on the lateral detected: "[JETSON_DIRT_L]";
  - If dirt on the top is detected: "[JETSON_DIRT_T]"

- Depending on the messages received, the mobile app will show pre-selected washing services;

- After the wash process, the sensor S2 detects the vehicle, and a "[XDK_DMG]:0" message is sent via MQTT to topic "cuida", informing the system to do a damage inspection;

- Depending on the results obtained from the "damage_best.pt" model inference, different messages are sent from the system to the mobile app, via "cuidapp":
  - If Scratch detected: "[JETSON_DMG_R]"
  - If Broken Glass detected: "[JETSON_DMG_v]"

- If Deformation detected: “[JETSON_DMG_R]”
- If Broken detected: “[JETSON_DMG_P]”

- Depending on the messages received, the mobile app will show different car repair services from GALP partner network;

The user guide and a demonstrator of the user interface are both provided in the repository.

8. Metrics and results

8.1. Dirt Detection Model

Considering the problematic at hand, the team focused on a state-of-the-art object detection algorithm, i.e. YOLOv5. Moreover, the YOLOv5 model was selected to reach a better compromise between precision and computational performance, i.e. Figure 4, assuring compatibility with the Edge-AI requirements.

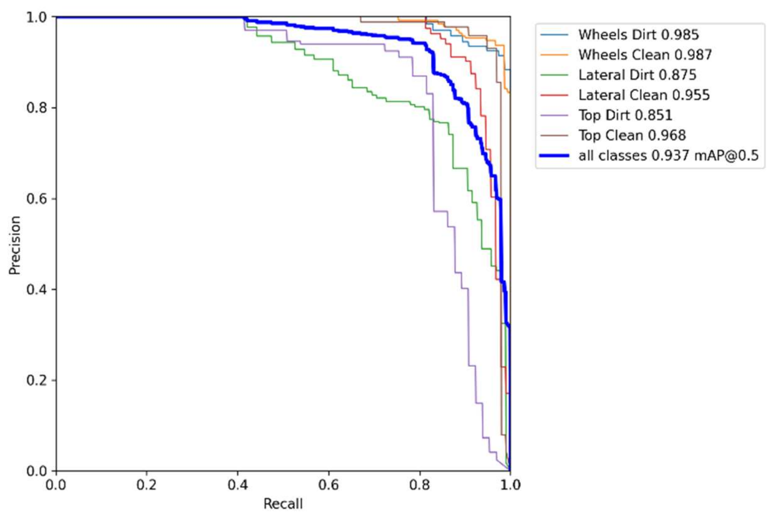

Figure 4 - Dirt Detection YOLOv5 mAP

8.2. Damage Detection Model

Damage detection makes use of the same YOLOv5 model, Figure 5, following the same considerations as described in the dirt model.

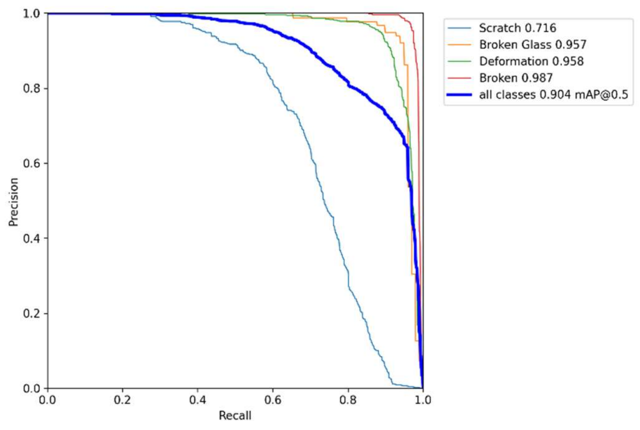

Figure 5 - Damage detection YOLOv5 mAP

8.3. Inference procedure

To automate the inspection procedure between CUIDA station and client, sequential automated process was implemented, which aggregates the station, Edge-AI, and cell phone functionalities (Figure 6). Following the description made in section 4:

0. Initially the user enters the CUIDA station, E1, while using CUIDA app;
1. If S1 detects the vehicle, N1 detects the dirt level in the vehicle;
2. N1 sends the results to J1, in a form of pre-selected washing services;
3. The client proceeds the washing station, E2;
4. After finishing the car washing, the client proceeds to the 2nd inspection chamber. If S2 detects the vehicle, N1 detects the damage in the vehicle;
5. N1 sends the results to J1, in a form of GALP partners which can provide a service.

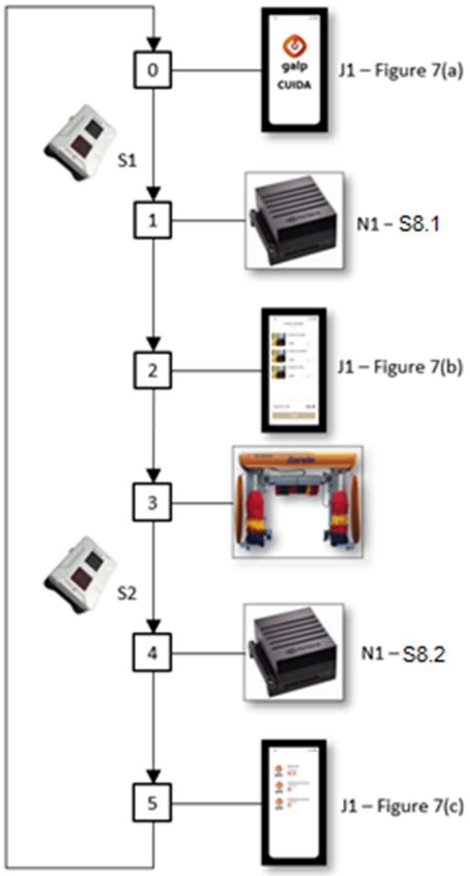

Figure 6 - CUIDA inference pipeline

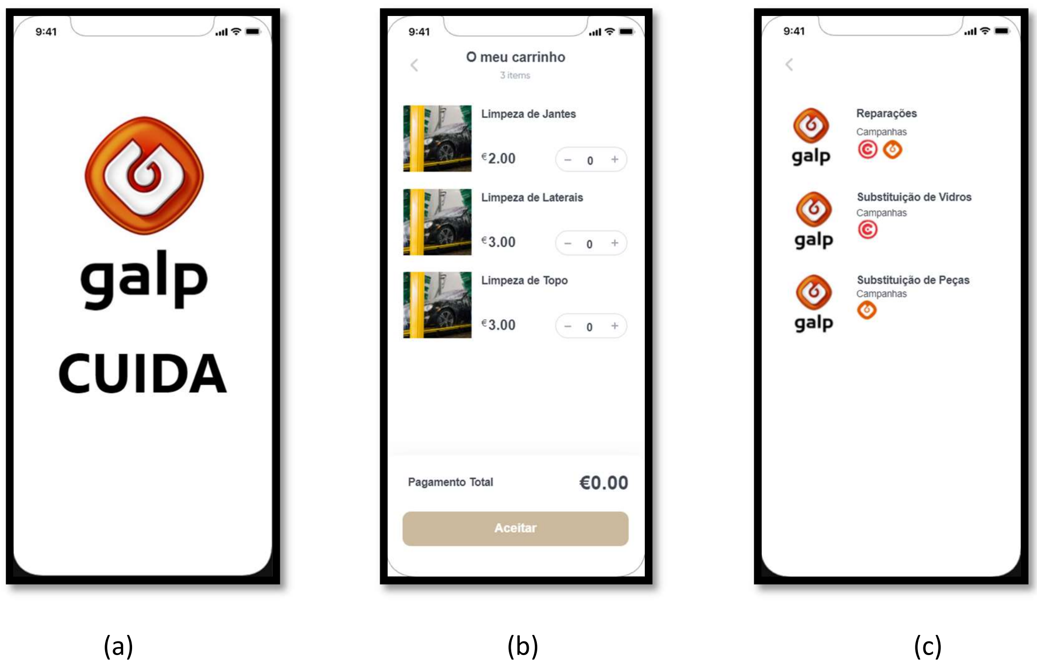

Figure 7 - CUIDA smartphone application.

9. Conclusions and Future Work

CUIDA is an inspection pipeline for dirt and damage, which can be installed into GALP service stations. This project is a proof of concept that presents an opportunity for GALP to improve its competitiveness and provide newer state-of-the-art services that foster the integration/offer of other solutions after the cleaning process, adding value to the process and helping users maintaining cars.

In the future, with the goal to make it commercial, higher TRLs are required. Not neglecting the required technological maturity of the automated system, the biggest effort should be placed in the algorithmic side. Further dataset generation campaigns are required, through manual annotation or synthetic data generation, to provide higher precision and generalization of the models.