Artifact: Codesign of Edge Intelligence and Automated Guided Vehicle Control

Malith Gallage, *Student Member, IEEE,*, Rafaela Scaciota, *Member, IEEE,*, Sumudu Samarakoon, *Member, IEEE*, and Mehdi Bennis *Fellow, IEEE*
Centre for Wireless Communication, University of Oulu, Finland
email: {malith.gallage,rafaela.scaciotatimoesdasilva,sumudu.samarakoon,mehdi.bennis}@oulu.fi

***Abstract*—This is the replication package for the paper, Codesign of Edge Intelligence and autonomous guided vehicle (AGV) Control, which is published at the International Conference on Pervasive Computing and Communications (PerCom), 2023. The paper presents the codesign of edge intelligence and AGV that can be utilized to automate repetitive tasks with a high degree of accuracy and efficiency while having the human-in-the-loop and thereby improving productivity. The artifact contains an introduction that presents the operation of the AGV, the hardware configuration, and scripts and instructions for replicating the results produced in the paper.**

***Index Term*—Edge AI, Image Processing, Autonomous Navigation**

I. Introduction

This is the artifact for the paper, Codesign of Edge Intelligence and autonomous guided vehicle (AGV) Control [1], where a semi-autonomous transportation task aided by edge intelligence. In the paper we used an AGV equipped with a robotic arm that needs to i) pick an object from a source point, ii) follow a path defined by black stripes, and iii) drop the object at a specific drop-point in one of four destinations implicitly defined by a human operator. The human operator defines the destination and exact drop point (delivery information) by placing an irregular black shape inside a destination.

The irregular shape is referred to as a custom drop area also defined by the human operator. The exact drop-point is the center of the custom drop area, which is defined as the center of the largest circle that is placed inside the custom drop area. At the source point, AGV request the delivery information from an edge server. Then, the edge server uses a remote camera to obtain the bird's eye view of all destinations. The artificial intelligence (AI) in the edge server first extracts the delivery information from the camera image and then shares it with the AGV.

The code certified related to this demo is available at github repository https://github.com/ICONgroupCWC/Demo.Percom23. The demo in action can be found in https://youtu.be/DhCSCCZbuHo.

The authors would like to acknowledge the support and contributions of Abdulmomen Ghalka, Dinesh Manimel Wadu, Mithila Amarasena, and Suranga Prasad for developing the image processing solutions.
This work was supported by the projects EU-ICT IntellIoT (grant agreement No. 957218) and Infotech-R2D2.

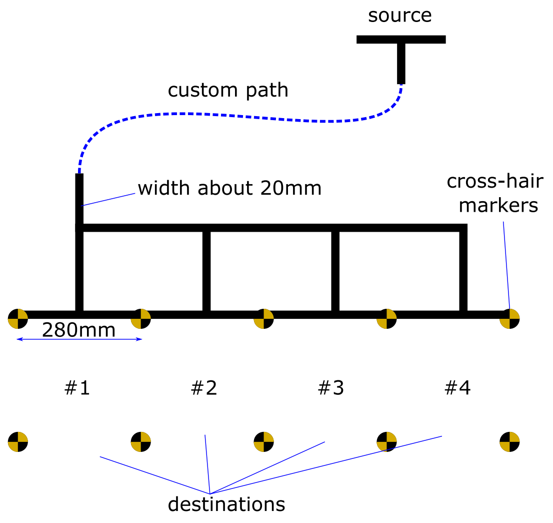

Figure 1: The layout of the robotic platform.

II. Implementation

A. Robot Platform

The robot platform consist of routes spanning from a source point to multiple destinations, with an AGV that transports objects from a source to four destinations. Paths are made from black lines about the width of 2 cm to guide the AGV. Each destination is a square area, where the corners are marked by four cross-hair markers. A camera is mounted in the platform to observe the destination areas. The basic layout of the platform is shown in Fig. 1.

B. AGV

AGV is an off-the-shelf mobile crawler robot known as "Jetank AI kit" which is powered by Nvidia Jetson nano developer module with 16GB eMMC and 4GB RAM [2]. This is capable of running resource-demanding modern computing algorithms related to machine learning and computer vision and supports many popular libraries and frameworks. It uses an onboard camera to sense the environment and determine its control decisions such as navigation among source and destinations. The contents of the code are on Github in the folder `selection_robot`.

To configure the Jetank, use a browser to access the Jupyter Lab on JETANK. Then, open a terminal and install the JETANK codes as follows:

```
git clone
  ↪ https://github.com/waveshare/JETANK.git
cd JETANK
chmod +x install.sh
chmod +x config.sh
./install.sh
./config.sh
```

After configuring the Jetank, the notebook provided in the `selection_robot` folder can be used to start the autonomous navigation of the AGV. In the notebook, the 'Test Mode' button that is created while running the code needs to be enabled to physically move the AGV. Moreover, it is essential to modify the IP addresses and the ports of edge server and remote camera under the entries `EDGE_SERVER_IP` and `CAMERA_IP`.

C. Remote camera

The camera that observes the destinations consists of a Raspberry Pi V2 camera module, which comes with a robust 8MP Sony IMX219 image sensor. It connects with a Raspberry Pi 4 Model B computer, which hosts a web server that serves high resolution images of the storage area upon the requests from the edge server. This camera provides static images up to 3280×2464 px resolution and it has a manually adjustable focal length. The contents of the code are on Github in the folder `image_server`. Image server is run on python 3.7 or higher and it can be run with the default python libraries.

The service is activated as follows:

```
python image_server.py
```

When activated you can see the image from the address.

```
http://[IP:PORT]/capture
```

where the `[IP:PORT]` is the IP address and the port of the camera that is run on the Raspberry Pi (e.g. "127.0.0.1:8080").

D. Edge Server

A powerful multi-purpose 64-bit Windows 10 computer acts as the edge server and it hosts the AI service and shares the delivery information with the AGV upon request. If you need to create the Edge Server in an OS system you have to find the tenserflow version that works in the OS system. The Edge server derives the delivery information by using the camera images with a bird's eye view. The REST APIs of the AI service is generated from swagger specifications and it runs on python-flask. The swagger specifications are given in `swagger.yaml` and is stored under `edge_ai_delivery\swagger_server\swagger\`.

The code of the edge server is given in the Github repository under the folder `edge_ai_delivery`, which requires to be the root. This installation needs python 3.8 and a windows operating system. If the operating system differs, the corresponding tensorflow version should be installed. For this demo, Anaconda distribution is used to create the python environment and install dependencies [3]. The required dependencies are given in the requirements.txt file. In the computer, open a terminal and install the dependencis at the root folder as follows:

```
pip install -r requirements.txt
```

Prior to running the server, the pre-trained cross-hair marker detection model should be added under the directory `edge_ai_delivery/swagger_server/models/`. Then if the required dependencies are satisfied, run the following command to start the server. The working directory for this should be `edge_ai_delivery`. In the computer, open a terminal and run the server at the root folder as follows:

```
python -m swagger_server
```

The AI service uses a pre-trained cross-hair marker detection model named `workpieceStorage_marker_model_small.hdf5`. Due to the large size, the github repository cannot host the model and thus, it is shared on link https://doi.org/10.6084/m9.figshare.22093568.v1. The model needs to be stored under `edge_ai_delivery\swagger_server\models\` for the successful operation of the edge server.

When the server is running, the following hypertext transfer protocol (HTTP) call can be used to obtain delivery information. For the ease of use, we recommend the "Postman" API platform for testing [4].

```
http://[IP:PORT]/AI_Service/
  ↪ compute_deliveryInformation?
  ↪ storageId=[ID]&cameraHostname=[HOST]&
  ↪ cameraId=0
```

Here, the `[IP:PORT]` is the IP address and the port of the AI service that is run on the edge server (e.g. "127.0.0.1:8080"). The `[ID]` is the destination ID from the set $\{1, 2, 3, 4\}$. Here, using negative number returns a visualization of the delivery information that can be used for debugging purposes[1]. The `[HOST]` is the IP address and the port of the remote camera (e.g. "127.0.0.1:8080"). Note that the locally stored `test_image.jpg` can be used instead of contacting the remote camera by setting `[HOST]` to be empty.

References

[1] M. Gallage, R. Scaciota, S. Samarakoon, and M. Bennis, "Codesign of edge intelligence and automated guided vehicle control," in *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2023.

[2] Waveshare, "Jetank ai kit," Available at https://www.waveshare.com/jetank-ai-kit.htm (2022/11/11).

[3] "Anaconda software distribution," 2020. [Online]. Available: https://docs.anaconda.com/

[4] Postman, Inc., "Postman," Available at https://www.postman.com/ (2023/01/28).

[1]By changing `SHOW_PLOTS` to `False` on the line 15 in the file `edge_ai_delivery\swagger_server\models\init.py`, the popup windows can be avoided.