

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.scaciota,timo.sasilva,sumudu.samarakoon,mehdi.bennis}@oulu.fi

Abstract—This work presents a harmonic design of autonomous guided vehicle (AGV) control, edge intelligence, and human input to enable autonomous transportation in industrial environments. The AGV has the capability to navigate between a source and destinations and pick/place objects. The human input implicitly provides preferences of the destination and exact drop point, which are derived from an artificial intelligence (AI) module at the network edge and shared with the AGV over a wireless network. The demonstration indicates that the proposed integrated design of hardware, software, and AI design achieve a technology readiness level (TRL) of range 4-5.

Index Term—Edge AI, Image Processing, Autonomous Navigation

I. INTRODUCTION

The rapid growth of customer demands and increasing costs of resources, labor, and energy have driven industries to seek new technologies that improve productivity and efficiency. In particular, modern logistics is one of the key players to adopt autonomous automation technologies including human-in-the-loop, in which, human operators can provide the preferences required for automation [1]. Autonomous guided vehicles (AGVs) are one of the pivotal components to enable "smart logistics" in manufacturing plants [2]. Realizing full/semi-autonomy with the fusion of control systems, communication networks, and computation servers at the edge over repetitive tasks calls for artificial intelligence (AI)-based solutions [3].

With the increasing interests in utilizing AGVs in industrial applications, developing efficient and reliable AI-driven solutions for navigation tasks [3] is of utmost importance. Towards this, line following robots have received significant attention due to their ease of use and low complexity and robust operations [4]. In this work, we demonstrate the codesign of an intelligent crawler robot, which uses its camera to sense the environment and, navigate and accurately deliver objects to their corresponding location with the aid of an edge AI server to meet preferences set by a human operator.

II. SYSTEM ARCHITECTURE

We consider the robotic platform illustrated in Fig. 1 consisting of paths from a source point to multiple destinations, with an AGV that transports objects from source to destinations, a camera observing the destinations, and an edge

The authors would like to acknowledge the support and contributions of Abdulkomen 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.

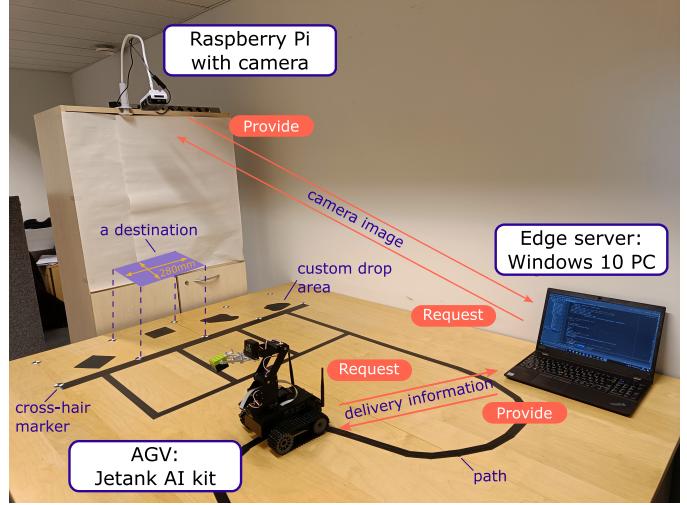


Figure 1: The robotic platform highlighting the components and wireless connectivity.

server equipped with computing capabilities. The AGV uses an onboard camera to sense the environment and determine its control decisions, i.e., navigation among source and destinations. The destination and precise drop location of the object, referred to as *delivery information* hereinafter, are determined by the edge server using camera images and shared with the AGV upon request. The delivery information is derived based on custom drop areas defined by an external party (e.g., human operator).

Communication between the AGV and the edge server takes place through a REST API over a WiFi network. The server provides delivery information in JSON format via its exposed endpoint, upon request by the AGV which takes place only once when each of the transport job is initiated. To derive delivery information, the edge server obtains a bird's eye view of the destinations by accessing the camera through its REST API over WiFi. After completing the delivery job successfully, the AGV returns to the source and repeats the procedure.

The paths between the source and the destinations are defined by black lines drawn on the surface to aid AGV's navigation. Here, a single path starting from the source, branches out to four paths leading to four adjacent destinations. Each destination is a square of width 280 mm and the corners of the area are marked by cross-hair markers. Hence, four destinations are defined by ten cross-hair markers as shown in

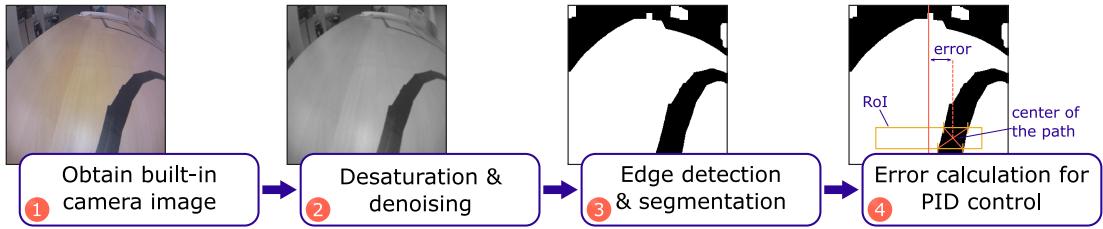


Figure 2: The key steps of the intelligent navigation.

Fig. 1 with the furthest marker at the left defined as the origin of the 2D coordinates. The custom drop areas are located inside each of the destination, which are possibly irregular shapes. The precise drop location is defined as the center point of the largest circle placed inside the custom drop area. It is worth highlighting that the drop point coordinates need to be computed with respect to the actual coordinates using a camera image having measurements in pixels (px). The placement of cross-hair markers is predefined and this knowledge is utilized to translate the px distances to the actual measurements during image processing at the edge server.

III. SYSTEM IMPLEMENTATION

The demo setup is composed of several hardware components and two key software solutions providing the intelligence (i) for navigation and (ii) object placement. The details of these components and the implementation are discussed next.

A. Hardware

Fig. 1 shows all the hardware components used in the system. AGV is a an off-the-shelf mobile robot known as "Jetank AI kit" which is powered by Nvidia Jetson nano developer module with 16GB eMMC and 4GB RAM [5]. The robot is equipped with 4-Dof mechanical arm and a wide-angle camera with 160° field-of-view. The camera consists of a Raspberry Pi V2 camera module and connects with a Raspberry Pi 4 Model B computer, which hosts a web-server that serves images of the storage area upon the requests from the edge server. A powerful multi-purpose 64-bit Windows 10 computer acts as the edge server and it hosts the AI service and share the delivery information with the AGV.

B. Intelligence for The Navigation

A vision-based line following system is implemented on the AGV to successfully navigate around the platform by using the on-board camera. The camera and the Jetson nano board support processing images with resolution 300×300 px up to 30 frames per second. During the navigation, the following three steps are repeated: i) acquiring camera images, ii) image pre-processing and line detection, and iii) actuating control decisions of the AGV as illustrated in Fig. 2.

The initialization is to set both the robot arm and the camera orientation to a position providing an obstruction free view of the path. During navigation, camera images are obtained repeatedly, and converted to gray scale images. Using a 3×3 Gaussian kernel, Gaussian blurring is performed to reduce the noise in the image (see step 2 in Fig. 2). Next, two

morphological operators, erosion and dilation, are applied on the image to remove miniature inconsistencies on the detected path caused by to the glare of the surrounding light sources while preserving the structure and the shape of the path. Then, the image is segmented by using OTSU method [6], which results in a binary image as shown under step 3 in Fig. 2. Therein, the path (dark surface) and the horizon (low lighting) of the image appear in black color while the rest appears in white. To neglect the undesired black areas corresponding to the horizon, a rectangular window that is likely to contain the path has been defined as the *region-of-interest (RoI)* (see step 4 of Fig. 2). By assigning weights ones and zeros to black and white px, respectively, the centroid of the RoI is calculated and referred to as the *center of the path*. The displacement of the center of the path from the vertical symmetrical axis of the image is denoted as the *error*. This error value is used in a proportional-integral-derivative (PID) controller to determine the control commands (i.e., angular velocities of left and right wheels) ensuring a smooth navigation [7]. The coefficients of the proportional, derivative, and integral are 1, 1, and 0, respectively, which are obtained during the tuning phase of the AGV.

At junctions, the AGV switches to an alternative control procedure that relies on the type of the junction and the knowledge on the direction of moving and the delivery information. The junction is detected and its type is determined by analyzing the boundaries of the region of interest. Next, based on the current goal, predefined sequences of control commands are issued to turn the AGV by 90° or 180°. After turning, the AGV hands over the control to the PID algorithm. The AGV identifies the terminal points, source and destinations, by the absence of path segments in the RoI. As the AGV reaches to source or destination, it activates pick/drop procedures accordingly. Here, the robot arm is programmed to move to a given 2D coordinate along its moving plane using inverse kinematics and grab or release the object.

C. Intelligence for The Object Placement

The role of the edge server is to provide the intelligence in terms of computing the delivery information, which includes the destination and the drop location. The destination is one of the four square areas defined by cross-hair markers. A human operator defines a custom drop area by placing a dark and possibly irregular flat surface made out of paper. The drop point is the center of the largest circle (not necessarily to be unique) that can be placed inside the user-defined area.

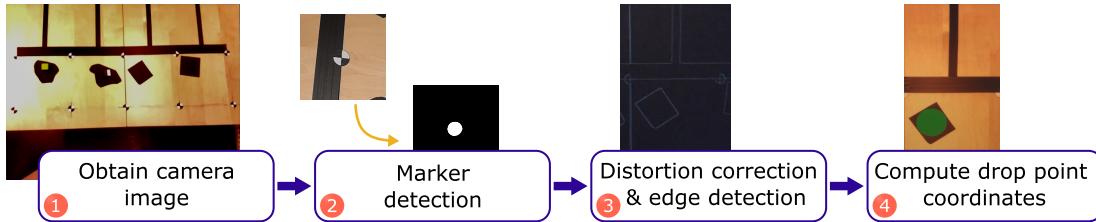


Figure 3: The key steps of the edge AI service.

Given the camera image, derivation of the delivery information requires several steps including cross-hair marker detection, isolate the destination and camera distortion correction, identify the custom area within the destination, and compute drop coordinates. The flow of the edge server operation is illustrated in Fig. 3.

A pretrained machine learning (ML) model is used for the cross-hair marker detection. For the supervised training, 256×256 px images extracted from the camera is used as the inputs while the labels are images with the same dimensions having white areas corresponding to the cross-hair markers and black areas reflecting everything else. To avoid handcrafting a dataset, a data augmentation procedure is adopted. First, using a single camera image (see step 1 in Fig. 3 as an example) as the master sample, a master label is generated by filtering out the background (anything except cross-hair markers). Then, different sizes of rectangles with different orientation are extracted randomly from the master sample and correspondingly from the master label. The extracted samples and labels are then resized to 256×256 px images to generate the training and testing datasets of sizes 1024 and 128, respectively. Next, a ML model based on the UNET architecture [8] with the input and output dimensions of 256×256 is trained and tested with the aforementioned dataset. During inference, the camera image is partitioned into 10 segments, resized, and fed into the cross-hair marker detection model. Once the cross-hair markers are detected (see step 2 in Fig. 3), the four destinations can be isolated. For the predefined destination, using the prior knowledge of markers to define a square, the detected markers are used to remove camera distortions. The main benefit of the marker detection capability is that the service provided by the edge server is robust against camera reposition and variations in lighting conditions up to a certain limit.

The isolated destination is extracted as a separate image segment and the edge detection procedure based on OTSU method explained in Sec. III-B is used to identify the edges of the custom drop area as illustrated under step 3 in Fig. 3. To determine the drop point, which is the center of the largest circle that can be placed inside the drop area (e.g., step 4 in Fig. 3), the edges are first approximated in to a polygon. Then, the desired point is the furthest point from all the edges that lies inside the polygon. This geometric solution is provided as a built-in function named `polylabel()` in a python package

named `polylabel` [9]. Since the drop point coordinates are returned in px with respect to the extracted image segment of the destination, they are translated to the actual physical coordinates based on prior knowledge of the setup dimensions and then shared with the AGV.

D. Demo, Resources, and Future Developments

The software related to this demo is available at [github](https://github.com/ICONgroupCWC/Demo.Percom23) <https://github.com/ICONgroupCWC/Demo.Percom23>. With a similar platform developed as per specifications provided in Sec. II and the use of the hardware specified under Sec. III-A along the aforementioned software, this demo can be reproduced. This demo in action can be seen from <https://youtu.be/DhCSCCZbuHo>.

The lighting conditions (over/under-exposure and harsh shadows) directly impact the performance of the AGV. To improve the overall performance, brightness and, exposure corrections methods and deep learning models for denoising and filtering will be investigated in future.

REFERENCES

- [1] H. Tang, X. Cheng, W. Jiang, and S. Chen, "Research on equipment configuration optimization of AGV unmanned warehouse," *IEEE Access*, vol. 9, pp. 47946–47959, 2021.
- [2] S. Li, J. Yan, and L. Li, "Automated guided vehicle: the direction of intelligent logistics," in *Proc. of IEEE Intl. conf. on SOLI*, 2018, pp. 250–255.
- [3] J.-S. Shaw, C. J. Liew, S.-X. Xu, and Z.-M. Zhang, "Development of an AI-enabled AGV with robot manipulator," in *Proc. of IEEE ECICE*, 2019, pp. 284–287.
- [4] M. Bošnak and I. Škrjanc, "Obstacle avoidance for line-following AGV with local maps," in *Proc. of IEEE Intl. SACI*, 2021, pp. 193–198.
- [5] Waveshare, "Jetank ai kit," Available at <https://www.waveshare.com/jetank-ai-kit.htm> (2022/11/11).
- [6] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [7] S. Bennett, "Development of the PID controller," *IEEE Control Systems Magazine*, vol. 13, no. 6, pp. 58–62, 1993.
- [8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. of Intl. Conf. on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [9] V. Agafonkin, "A new algorithm for finding a visual center of a polygon," Available at <https://blog.mapbox.com/a-new-algorithm-for-finding-a-visual-center-of-a-polygon-7c77e6492fbc> (2022/11/11).