Computational Offloading for Autonomous Systems: Real-World Experiments and Modeling

Jan Danek, Zdenek Becvar, and Adam Janes
Faculty of Electrical Engineering Czech Technical University in Prague, Prague, Czech Republic
danekja5@fel.cvut.cz, zdenek.becvar@fel.cvut.cz, janesada@fel.cvut.cz

***Abstract*—We focus on computation offloading from moving devices, such as mobile robots or autonomous vehicles to Multi-Access Edge Computing (MEC) servers via mobile network. To this end, we develop and implement a prototype of small autonomous vehicle with capability to offload processing of sensor data to MEC server via mobile network. Then, we investigate an impact of communication channel on delay and energy consumed by the autonomous vehicle for two practical applications, namely road sign recognition and path planning, in the real-world environment with a real physical equipment. Via experiments, we demonstrate benefits of the computation offloading on both energy and delay. The experiments highlight the potential of MEC for the autonomous systems allowing to reduce cost and increase scalability of such autonomous systems. Furthermore, based on the real-world experiments, we derive detailed models of energy consumption and delay for both practical applications.**

***Index Terms*—Multi-access edge computing, mobile network, autonomous driving, robots, testbed, autonomous systems.**

I. Introduction

The autonomous systems require efficient real-time data processing to ensure safe and reliable operation in dynamic environments [1]. The computational demands of autonomous operation algorithms impose a high demand on onboard computing units (CPUs) [2] while deploying powerful CPUs increases costs of the autonomous systems. To address this, offloading a part of the data processing to multi-access edge computing (MEC) servers located near base stations (BS) offers a cost-effective solution while maintaining real-time performance [3]. The offloaded computation to a powerful servers allows to reduce computing (processing) delay. Consequently, despite a non-zero communication delay for a transfer of the computation from a device to the MEC server, the overall task processing delay (including both communication and computing delays) can be reduced.

The offloading of computation to MEC servers has been extensively studied for various applications [4]. For instance, recent papers focuses on optimizing the task processing in MEC servers to increase the number of processed tasks [5], or focuses on optimizing the task processing in MEC servers to increase sensing rate of data from sensors and sensor's data subsequent processing [6]. These works provide insights into the design of MEC systems for general use cases, but do not address the requirements of autonomous systems, namely these works do not consider overall task processing delay critical to ensuring real-time response of the autonomous systems.

Furthermore, works [7], [8] target to minimize overall task processing delay by optimizing the task placement (local computation of offloading) and resource allocation. Similarly, also the paper [9] investigates delay-sensitive task offloading. However, the works on overall task processing delay minimization are based on optimistic and simplified models often lacking real-world validation and neglecting critical factors necessary for robust autonomous system operation, such as real-time execution.

The solution for autonomous systems is developed in [10], where the authors target to minimize energy consumption and overall task processing delay for autonomous vehicles. Furthermore, the paper [11] examines MEC-assisted autonomous driving. While these papers provide interesting insights into leveraging MEC for autonomous systems, namely autonomous vehicles (AVs), these papers rely on theoretical simplified models an evaluate performance in optimistic and controlled simulation environment, failing to capture the challenges of dynamic real-world environment.

In this paper, we aim to demonstrate the feasibility of computation task offloading from the autonomous systems to the MEC server. We focus on processing of data from onboard sensors of the AV and we evaluate an ability to process various types of AV's tasks on the MEC server. We focus on practical aspects of the computation offloading from the AV to the MEC servers over the mobile network. The major contributions of this paper consists in following:

1) We develop and implement a prototype of a model of the AV with capability to offload computation (processing) of onboard sensors to the MEC server via the mobile network.
2) We implement two AV applications: i) road signs recognition, and ii) path planning, as examples of computation heavy applications for potential offloading to MEC servers.
3) We demonstrate benefits of the offloading for both implemented applications in term of energy saving and overall task processing delay reduction if the offloading is enabled. To this end, we perform a series of real-world experiments with realistic data and equipment.
4) Based on the experiments, we derive mathematical models for energy consumption and overall task processing delay for both applications (road sign recognition and

This work was supported by Ministry of Education, Youth and Sport of the Czech Republic under Grant no. LUASK22064, and by Czech Technical University in Prague no. SGS23/171/OHK3/3T/13.

path planning) to allow future theoretical research to be done with realistic models of AV applications.

5) We make the open-source code for all aspects of the developed AV and experiments publicly available on GitLab[1] for future research contributing to transparency and reproducibility of the work.

The rest of the paper is organized as follows. First, we outline a model of the system considered in this paper. Then, in Section III, we provide details on the AV and its integration with the mobile network and with the MEC server and we also describe adopted path planning and road sign recognition algorithms. In Section IV, we outline the scenario for experiments. Then, in Section V, we describe results of experiments and define models of the energy consumption and overall task processing delay for both path planning and road sign recognition. Last, Section VI concludes the paper.

II. System Model

In this section, we introduce modelling of the computation offloading from the autonomous system, represented by the AV, to the MEC server via the mobile networks. We also define the overall task processing delay and energy consumption related to both offloading and local computing.

We assume the system with one AV and one BS, see Fig. 1. The BS is enhanced with the MEC server allowing to process computing tasks related to the autonomous driving, such as path planning or road sign recognition.

Each computing task is characterized by a volume of data D_{ul} to be transferred from the AV to the MEC server for the task offloading, by the volume of the task processing result D_{dl} transferred from the MEC server to the AV, and by the computational demand C_D of the task generated by the AV.

The computing task can be processed either locally in CPU of the AV with the computing power C_{AV} or offloaded to the MEC server with the computing power C_{MEC}. The MEC server is equipped with a common CPU with a computation power C_{CPU} and with a graphical processing unit (GPU) with a computing power C_{GPU}. The overall computing power of the server is then defined as a sum of computing powers of both components, i.e., $C_{MEC} = C_{GPU} + C_{CPU}$.

We define a binary variable σ indicating whether the task is processed locally on the AV ($\sigma = 0$) or offloaded to the MEC server ($\sigma = 1$). The communication between the AV and the BS is characterized by downlink bitrate b_{dl} and uplink bitrate b_{ul}. We assume the mobile network to be adopted for AV's communication, hence, the bitrates in both directions results from adopted modulation, coding rate, and the number of resource blocks allocated to the AV [12]. Thus, the uplink bitrate b_{ul} is defined as:

$$b_{ul} = N_{ul}^{RB} \times N_{ul}^{subcarriers} \times N_{ul}^{bits} \times N_{ul}^{s} \times CR_{ul}, \quad (1)$$

where N_{ul}^{RB} is the number of used resource block, $N_{ul}^{subcarriers}$ is the number of subcarriers per resource blocks, N_{ul}^{bits} denotes the number of bits per symbol according to the selected modulation, N_{ul}^{s} represents the number of symbols transmitted per second, and CR_{ul}, is the code rate. The expression for bitrate b_{dl} in downlink is same as for the uplink, just the uplink parameters N_{ul}^{RB}, $N_{ul}^{subcarriers}$, N_{ul}^{bits}, N_{ul}^{s}, CR_{ul} substituting the related downlink parameters.

[1]https://gitlab.fel.cvut.cz/mobile-and-wireless/autonomous-driving

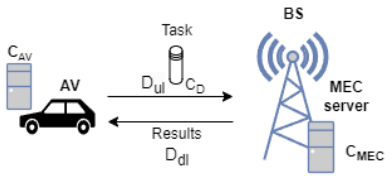


Fig. 1: System model with AV either computing/processing sensor data locally or offloading the computation to MEC server via mobile network.

The communication delay t_n^c for the n-th task includes the uplink communication time $t_n^{ul} = \frac{D_{ul}}{b_{ul}}$ for offloading the task to the MEC server and the downlink communication time $t_n^{dl} = \frac{D_{dl}}{b_{dl}}$ for transferring the result back to the AV, i.e.,:

$$t_n^c = \sigma \times \left(t_n^{ul} + t_n^{dl}\right) = \sigma \times \left(\frac{D_{dl}}{b_{dl}} + \frac{D_{ul}}{b_{ul}}\right). \quad (2)$$

The computing delay t_n^p of the task on the MEC server or locally in the AV is determined as:

$$t_n^p = \sigma t_n^{MEC} + (1-\sigma) t_n^{AV} = \sigma \frac{C_D}{C_{MEC}} + (1-\sigma)\frac{C_D}{C_{AV}}, \quad (3)$$

where $t_n^{MEC} = \frac{C_D}{C_{MEC}}$ is the computing time on the MEC server and $t_n^{AV} = \frac{C_D}{C_{AV}}$ is the computing time on the AV.

The overall task processing delay t_n^o is defined as the total time required to process the n-th task locally or offloaded to MEC server and includes both computing delay and communication delay, i.e.,

$$t_n^o = t_n^p + t_n^c. \quad (4)$$

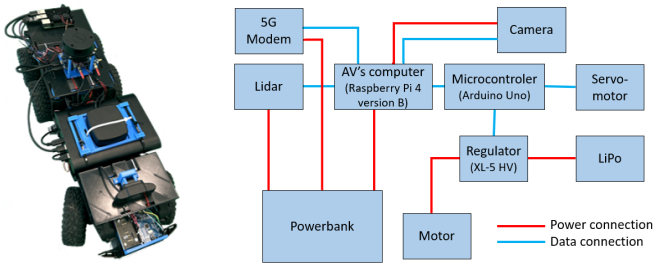
Since the autonomous system can be energy constrained, e.g., in case of the robots or AVs, we define the overall energy consumption E_n^o of the AV for the n-th task processing as:

$$E_n^o = \sigma\left(E_n^p + E_n^c\right) + (1-\sigma)E_n^p = \sigma t_n^o\left(P_p + P_c\right) + (1-\sigma)t_n^o P_p, \quad (5)$$

where $E_n^p = t_n^o P_p$ is the computing energy consumed by the CPU of the AV, $E_n^c = t_n^o P_c$ denotes the communication energy consumed to transmit the data from the AV to the MEC server, P_p and P_c represent the power consumed by the CPU on the AV and by the AV's communication modem, respectively.

III. System Design and Implementation of Autonomous Vehicle Framework

In this section, first outline the whole system for offloading of tasks form the autonomous system, represented as AV, to the MEC server. Then, we describe a developed model of autonomous system represented as AV for experiments from the perspective of key building blocks and components. Afterwards, we present an integration of the MEC server and communication infrastructure to the system. Then, we also



(a) (b)

Fig. 2: Model of AV (a) and block scheme of AV and key components for driving, communication, and computing (b)

describe two implemented offloadable applications for AVs, namely: i) the rapidly exploring random trees star (RRT*) [13] algorithm for path planning and ii) road sign recognition based on neural networks.

A. Overview of implemented system for computation offloading from AV to MEC via mobile networks

The whole implemented system follows the model presented in Fig. 1. The AV generates tasks essential for driverless operation, including path planning using RRT* algorithms and road sign recognition via neural networks [14]. Sensor data, collected from a camera and Light Detection And Ranging (LiDAR), is processed by the AV's computer running the Simultaneous Localization and Mapping (SLAM) algorithm [15]. SLAM creates a grid map [15] of the environment using LiDAR data and localizes the AV's current position. The grid map consists of nodes [15], representing discrete points that indicate whether a location is occupied by an object.

During operation, the AV plans a feasible path in the map and captures images to recognize road signs. To reduce the computational load on onboard hardware, tasks can either be offloaded to the MEC server or processed locally on the AV's computer. The outputs from the path planning and the road sign recognition are inserted to the control algorithm of the AV to navigate the AV in the environment.

B. Implementation of autonomous vehicle

The developed AV is based on a 1:10 scale model of the Mercedes-Benz G63 with a TRX-6 chassis, as shown in Fig. 2a, and is controlled using Robot Operating System (ROS)[2]. Key components include motors, LiDAR, camera, AV's computer, microcontroller, 5G modem, and batteries, detailed in Fig. 2b.

The LiDAR scans the environment, providing data for SLAM to localize the AV and creates a map of environment. The SLAM runs in the AV's computer. The camera, connected via USB, captures images for road sign recognition. The motor controls forward and backward movement of AV, while the servomotor controls turning via the front axle. The motor's speed is controlled by the regulator. The actions for the regulator to control motor and servomotor are determined by the AV's computer and sent via a microcontroller, which facilitates an execution of the actions by the motor and the servomotor.

The communication, allowing the computing tasks to be offloaded to the MEC server, is facilitated by a 5G modem. For local computing, tasks are processed directly on the AV's computer. The AV's computer, LiDAR, and 5G modem are powered by a power bank, while the motors and servomotors use a LiPo battery. Note that detailed specification of individual components and details on their interconnection are available on GitLab repository[3]. Additionally, the control codes for the AV are also accessible on our GitLab repository.

C. Integration of MEC server and communication infrastructure

In this subsection, we describe communication infrastructure and MEC server integration to the system, see Fig. 3.

For local computing, sensor data is processed directly on the AV. For MEC server processing, sensor data and SLAM outputs are transmitted to the MEC server via a software-defined mobile network, using the modem located on the AV. The mobile network is implemented using OpenAir-Interface[4] (OAI), an open-source platform widely used for mobile network experimentation. The network consists of a BS, implemented with a USRP N310 software-defined radio and a connected computer running the OAI. The second part of the mobile network is a core network, running on a separate computer and managing the mobile network.

Tasks are processed within Docker[5] containers, which encapsulate code for the task processing and code's dependencies. This containerization ensures isolation between tasks, preventing conflicts and allowing multiple tasks to run independently on the MEC server. Furthermore, Docker containers provide scalability in the MEC server system.

The MEC server operates as a stand-alone server and is connected to the BS via a wired local area network (LAN).

D. Applications for offloading

In this section, we describe two applications generating computing tasks for offloading, i.e., RRT* for path planning and neural networks for road sign recognition.

1) Path planning RRT algorithm:* The RRT* algorithm [13] is an efficient path-planning algorithm commonly used for navigation of autonomous systems, respectively AV, in a high-dimensional space by incrementally constructing a tree, containing all feasible paths from the AV's start position, through random sampling, to a AV's final position. The RRT* is fed with the environment map, AV's current position from SLAM, and the destination. Based on these inputs, the RRT* returns a feasible path as an output.

The RRT* algorithm starts at the AV's position and iteratively samples random points to extend the path towards

[2]https://www.ros.org

[3]https://gitlab.fel.cvut.cz/mobile-and-wireless/autonomous-driving

[4]http://www.openairinterface.org/

[5]https://www.docker.com

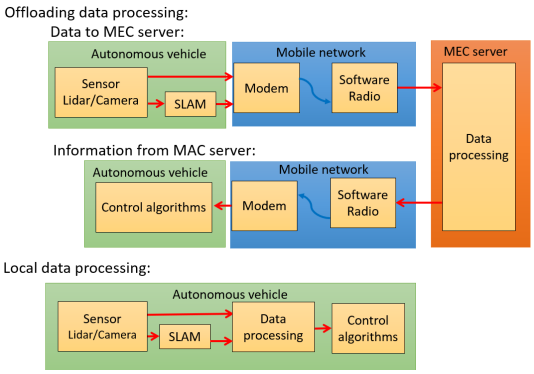


Fig. 3: Flow of data from sensors of AV to driving control algorithm of AV for offloading to MEC server and for local data processing in AV.

each new point while minimizing the path cost and avoiding obstacles. The RRT* considers the AV's kinematic constraints, such as turning radius, ensuring practical paths.

2) Road sign recognition: As described in [14], road sign recognition involves two stages: detection and classification. The detection stage identifies the bounding box of a road sign within an image captured by the AV's camera, while the classification stage categorizes the road sign in the cropped image, cropped by founding bounding boxes.

For detection, we use You Only Look Once version 3 (YOLOv3) [16], chosen for its high precision and low computational demands compared to alternatives such as Single Shot Detection [17] or Faster R-CNN [17]. Our implementation is based on a GitHub repository[6].

To prevent overfitting, the training process applies transformations such as flipping, brightness/contrast adjustments, and noise addition. Inputs include transformed images and annotated bounding boxes of road signs. We use the Adaptive Moment Estimation (ADAM) optimizer [18] with a learning rate of 5×10^{-5} and weight decay of 5×10^{-4}. The original loss function from [16] is employed, excluding classification loss, since only one class (road signs) is recognized. We create our own dataset for training to fit our experiments needs. This dataset is captured and annotated on Czech Technical University in Prague. Dataset is available on GitHub[7].

In the inference phase, inputs are represented by images captured by the AV's camera. The outputs of YOLOv3 are represented by a cropped image of the detected road signs.

Widely used neural networks for image classification, such as Residual Network [19] or Visual Geometry Group-16 [20], are unsuitable for deployment on autonomous systems due to their high computational demands and insufficient accuracy for this application. To address these challenges, we developed a neural network specifically for road sign classification, as shown in Fig. 4. The architecture efficiently extracts features using convolutional layers with 5x5 and 3x3 kernels to capture

[6]https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/object_detection/YOLOv3

[7]https://gitlab.fel.cvut.cz/mobile-and-wireless/autonomous-driving

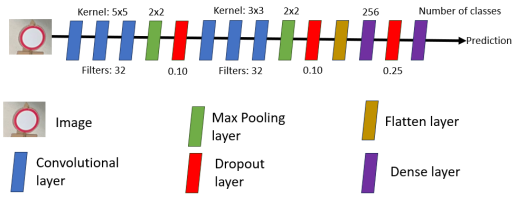


Fig. 4: Neural network for road sign classification.

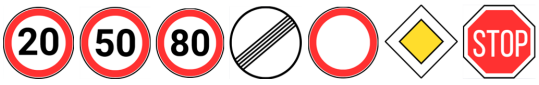


Fig. 5: Road signs used for testing.

both large and fine details. Dense layers and dropout regularization ensure robust classification while mitigating overfitting during training process.

The neural network is trained on images containing a single road sign, with labels corresponding to the road sign class. We employ the ADAM optimizer with a learning rate of 0.001 and weight decay of 0.001, using cross-entropy [21] as the loss function. For inference, the input is the cropped image of the road sign, extracted by YOLOv3. The network outputs a probability distribution over the classes, with the highest probability indicating the predicted class.

Training is conducted using the German Traffic Sign Recognition Benchmark [22], which includes 43 road sign types. To simplify the training and evaluation process, we selected seven commonly encountered road sign types: Speed limit (20 km/h, 50 km/h, and 80 km/h), End of all prohibitions, No entry for vehicles, Priority road, and Stop sign, as shown in Fig. 5. This subset allows focused testing within limited time constraints while ensuring relevance to real-world scenarios.

IV. Experimental Setup and Scenario

In this section, we discuss the setup and scenarios for experiments to evaluate the benefits of the computation tasks offloading compared to the local computing. Performance metrics are also defined in this section.

A. Experimental Setup

The MEC server is represented by a computer with a CPU i7-115G7, 16 GB RAM, and GPU Nvidia RTX 2060. The computing power of the MEC server for CPU and GPU (C_P and C_{GPU}) is 20.4 GFLOPS and 52000 GFLOPS, respectively. Note that the graphic card is used only for road sign recognition. Calculations on AV are performed on Raspberry Pi 4 version B with computing power C_{AV} of 9.69 GFLOPS. For communication between the AV and MEC server, the 5G mobile network is used. The 5G operates at 3.5 GHz (n78) with carrier bandwidth 20 MHz. We consider modulation and coding scheme 24 (Modulation 64QAM and Code rate R_X 0.754) [12], 106 resource blocks, downlink and uplink periodicity of 5 ms, downlink and uplink pattern: 6 slots for downlink and 3 slots for uplink.

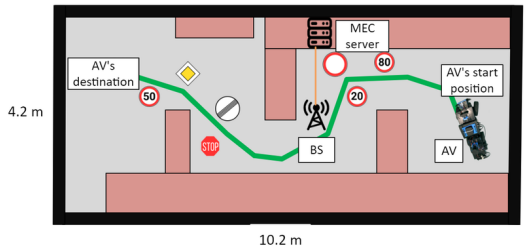


Fig. 6: Map of environment for experiments showing space for AV movement (grey area), walls (black lines), obstacles (red boxes), example of computed path for AV (green line), AV's start and destination, base station, MEC server, and road signs.

TABLE I: Volume of Tasks and Computational Demand

Task	D_{ul} [kb]	D_{dl} [kb]	C_D [GFLOPS]
Road sign recognition	282.4	0.624	523.260
Path planning	944	10.64	0.380

The experiments are done with the AV model described in Section III. To ensure consistent conditions across different experiments, we regulate the quality of the communication channel by limiting the transmission bitrate in both uplink and downlink directions to the same values in range from 0.1 to 3 Mb/s on the AV site by program Wondershaper. Limiting the bitrate allows us to guarantee the same conditions for all experiments. Note that modification of parameters such as bandwidth, transmission power or modulation and coding scheme could not guarantee the same data rate due to channel conditions that can change quickly in an unpredictable way and cannot be controlled in a real-world environment during experiments. To ensure delivery of the offloaded computation to the MEC server and return of results back to the AV, transmission control protocol (TCP) is adopted at transport layer above the Internet protocol (IP).

The task volume of data in uplink D_{ul} and in downlink D_{dl} and computation demand C_D are presented in Table I.

To ensure reliable information on energy consumption and computation time for the road sign recognition and the path planning, each task is repeated 100 times for every bitrate. The results are then averaged out over all 100 experiments.

For every task we prepare a dataset providing the same input data to the algorithm (road sign recognition and path planning) to ensure consistency between experiments. The datasets contain images for road sign recognition tasks, AV's start position, AV's destination and map of the environment for path planning tasks and is available on our GitHub[8]. An example of the experiments is demonstrated in video at our YouTube channel[9].

[8]https://gitlab.fel.cvut.cz/mobile-and-wireless/autonomous-driving
[9]https://youtu.be/aPKcAR9Qli4?si=f1m0Y7pn2Ys0vKSg

B. Performance metrics

Two performance metrics are considered: i) energy consumption and ii) delay, as defined in Section II. The energy consumption is measured using a USB power meter with 1 mWh resolution and $\pm 1.41\%$ accuracy.

The delay measurement is recorded on both the AV and MEC server. The computing delay t_n^p is defined as the time difference between the data input to the road sign recognition or RRT* algorithm and the time when output of these algorithms is ready. The communication delay t_n^c is measured as the sum of the time taken to send data from the AV to the MEC server (uplink communication time t_n^{ul}) and the time taken to receive the processed results from the MEC server to the AV (downlink communication time t_n^{ul}).

C. Scenarios of experiments for different tasks

In this subsection, we describe the scenarios used to measure the consumed energy and the offloading delays for the road sign recognition and the path planning.

1) Road sign recognition: The images for the experiments are selected so that there is an even representation of cases from zero to four road signs in one image. During the measurement a total of 100 images are processed for each bitrate. Individual road signs are deployed randomly along the path, see Fig. 6.

2) Path planning: The map of the environment and an example of the path for the AV are shown in Fig. 6. To ensure fair comparisons, we pre-generate a fixed set of random points for all experiments, guiding the exploration of the RRT* algorithm. Unlike the classical RRT* algorithm [13], where random points are dynamically generated to expand the path tree, our adopted approach eliminates randomness that could otherwise affect the computing delay t_n^p. Random selection of the tree growth directions in RRT* could significantly impact computing delay t_n^p, making comparisons across bitrates unreliable. By freezing the same set of points, we ensure consistency in the generated paths.

V. Performance Evaluation

In this section, we present the results of experiments. The results are presented in subsections corresponding to individual types of the offloaded tasks, i.e., road sign recognition, or path planning. In the last subsection, we introduce models for overall task processing delay t_n^o and consumed energies E_n^c and E_n^p as a function of the communication bitrate. Note that in Figs 7., 8., 9. and 10, $\sigma = 1$ indicates offloading the tasks to the MEC server while $\sigma = 0$ represent the local task processing.

A. Road sign recognition

Fig. 7 shows the average overall task processing delay t_n^o for the road sign recognition as a function of bitrate in downlink and uplink (both are the same). For the offloading, the communication delay t_n^c decreases with increasing bitrate, while computing time t_n^p remains constant. Consequently, t_n^o decreases up to 2 Mb/s, after which it saturates due to constant

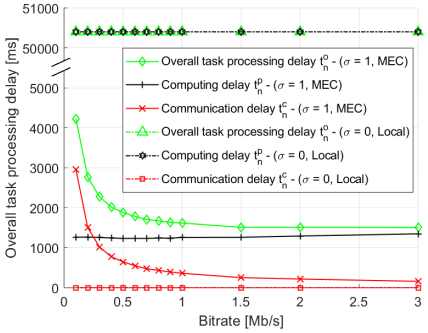


Fig. 7: Impact of bitrate on overall task processing delay t_n^o for road sign recognition

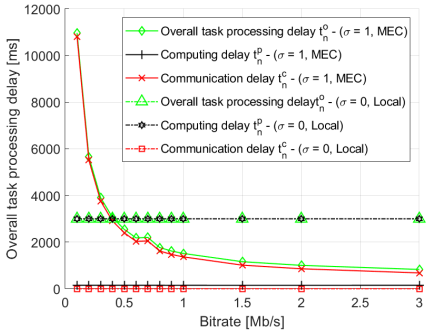


Fig. 8: Impact of bitrate on overall task processing delay t_n^o for path planning

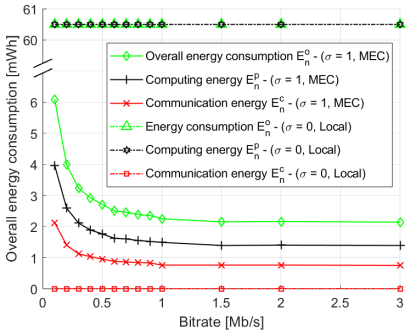


Fig. 9: Impact of bitrate on overall energy consumption E_n^o for road sign recognition

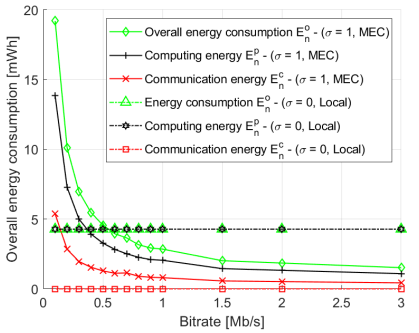


Fig. 10: Impact of bitrate on overall energy consumption E_n^o for path planning

signal processing and communication overhead delay, which dominates t_n^c and creates a lower bound regardless of further increase in bitrate. The signal processing and communication overhead delay is constant for every bitrate, and is introduced in HW (USRP and modem) and can be seen as overhead added on top of data communication and we include it in the developed models in the next subsection.

The dependence of communication delay t_n^c on bitrate for offloading follows a hyperbolic relationship in line with (2). The overall task processing delay t_n^o is from 11.9 times (for 0.1 Mbit/s) to 36.5 times (for 3 Mbit/s) longer for the local processing compared to the offloading. The offloading remains efficient even at low bitrates, as the reduction in t_n^p for the offloading compared to the local processing outweighs the communication delay t_n^c. The road sign recognition, requiring high computational power but transferring only small data volumes, benefits from the offloading at any bitrate.

Fig. 9 shows the average overall energy consumption E_n^o for the road sign recognition. The offloading reduces E_n^o by 3.1 times (for 0.1 Mb/s) to 30.7 times (for 3 Mb/s) compared to the local processing. In case of the offloading E_n^o gets saturated at about 1 Mb/s due the saturation of the communication delay t_n^c while both computing energy consumption and communication consumption remain constant.

The offloading proves to be more energy-efficient across all tested bitrates for the road sign recognition, as the reduction in the computing energy E_n^p by the offloading compared to the local processing exceeds the increased communication energy E_n^c introduced by the offloading.

B. Path planning

Fig. 8 presents the overall task processing delay t_n^o for the path planning as a function of bitrate. As in Fig. 7, for the offloading, the communication delay t_n^c decreases with increasing bitrate while the computing time delay t_n^p remains constant, leading to a hyperbolic dependence of t_n^c on bitrate, as expected according to (2).

The overall task processing delay ranges from 4.5 times increase (for 0.1 Mb/s) to 3.7 times decrease (for 3 Mb/s) in case of the local processing compared to the offloading. The offloading becomes more efficient only for bitrates above 0.5 Mb/s due to relatively large volumes of data transferred to the MEC server compared to the road sign recognition. For the offloading of the path planning, the trade-off between the communication delay t_n^c and the computing time delay t_n^p makes the offloading beneficial only at higher bitrates.

The overall energy consumption E_n^o for the path planning is shown in Fig. 10. The offloading increases E_n^o by up to 4.5 times for low bitrates (0.1 Mb/s), but reduces E_n^o by up to 3.5 times for high bitrates (3 Mb/s) compared to the local processing. As in Fig. 9, the overall energy consumption E_n^o for offloading first decreases with increasing bitrate due to a reduction in the communication delay t_n^c. Then, the overall energy consumption E_n^o for offloading saturates at about 2 Mb/s, as the signal processing and communication management latency becomes the dominant factor.

While both computing energy E_n^p and communication energy E_n^c for offloading decrease with increasing bitrate, the offloading becomes more energy-efficient only for bitrates above 0.6 Mb/s. This threshold is the trade-off point, where the reduction in E_n^p achieved by the offloading exceeds the increase in E_n^c due to the communication.

C. Modelling of overall task processing delay and energy consumption

The experimental results presented in the previous section allow for the creation of realistic models of overall task processing delay, and computing and communication energy consumption. The overall task processing delay is modeled as follows:

$$t_n^o = t_n^c + t_n^p = t_{oh} + \frac{D_{ul}}{b_{ul}} + \frac{D_{dl}}{b_{dl}} + t_n^p, \tag{6}$$

where t_{oh} is the delay due to signal processing and communication overhead (e.g., delay in USRP N310 or in modem).

The computing energy E_n^p and the communication energy E_n^c are modeled as follows:

$$E_n^p = t_n^o P_p = (t_n^c + t_n^p) P_p = \left(t_{oh} + \frac{D_{ul}}{b_{ul}} + \frac{D_{dl}}{b_{dl}} + t_n^p \right) P_p, \tag{7}$$

$$E_n^c = t_n^o P_c = (t_n^c + t_n^p) P_c = \left(t_{oh} + \frac{D_{ul}}{b_{ul}} + \frac{D_{dl}}{b_{dl}} + t_n^p \right) P_c, \tag{8}$$

where P_c and P_p are the powers consumed for the communication (by modem) and for the task computing (CPU or GPU), respectively.

The values of variables in the models derived from Figs 7., 8., 9. and 10 are in Table II. The average power consumption of the modem P_c remains the same for both applications, but the average power consumption of the CPU in the AV P_p is higher for path planning. The transferred volumes of data D_{ul} and D_{dl} match the sizes defined in Table I, taking into account overhead caused by upper layer protocols, such as TCP/IP. The overhead is different for both application (road sign recognition and path planning) and arise in variations in volumes of transmitted data, resulting in different numbers of sent packets. Each packet contains a fixed overhead of fixed size implied protocols requirements by TCP/IP.

TABLE II: Values for models of t_n^c, E_n^p, and E_n^c for road sign recognition (RSR) and path planning (PP).

Task	t_n^p [ms]	t_{oh} [ms]	D_{ul} [kb]	D_{dl} [kb]	P_p [W]	P_c [W]
RSR	1271.00	64.41	288.36	0.64	0.93	0.50
PP	149.00	333.30	1033.35	11.65	1.27	0.50

VI. Conclusions

We have demonstrated potential of computation offloading from the autonomous systems, represented as AV, to the MEC server in real environment with a real equipment for two practical applications commonly used by AVs. The computation offloading from the AV to the MEC server achieves up to 80% reduction in the overall task processing delay and up to 75% in the energy consumption for the path planing. Reductions of up to 96% in the overall task processing delay and up to 97% in the energy consumption are observed for the road sign recognition. Testing across varying network conditions shows that offloading the road sign recognition is always beneficial, whereas the path planning is more efficient locally for low bitrates up to 0.5 Mb/s. We have also developed models for the computing and communication energy, and overall task processing delay.

Future work should expand the system to multiple BS and AVs, explore new offloading decision algorithms, and compare additional path planning and following methods to optimize performance.

References

[1] L. Liang, et al., "Vehicle Detection Algorithms for Autonomous driving: A Review," Sensors 2024, no. 10, 2024.

[2] M. Baziyad, M. Saad, R. Fareh, T. Rabie and I. Kamel, "Addressing Real-Time Demands for Robotic Path Planning Systems: A Routing Protocol Approach," IEEE Access, vol. 9. pp. 38132-38143, 2021.

[3] Z. Bečvář and P. Mach, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," IEEE Communications Surveys & Tutorials, vol. 19, no. 3, 2017.

[4] D. Shi, et al., "Task offloading strategies for mobile edge computing: A survey," Computer Networks, volume 254, 2024.

[5] L. Liu, et al., "Deep Reinforcement Learning-based Dynamic SFC Deployment in IoT-MEC Networks," IEEE ICCC, 2022.

[6] X. Li, S. Bi, Z. Quan and H. Wang, "Online Cognitive Data Sensing and Processing Optimization in Energy-Harvesting Edge Computing Systems," IEEE Transactions on Wireless Communications, 2022.

[7] C. -L. Chen, et al., "Latency Minimization for Mobile Edge Computing Networks," IEEE Trans. Mob. Comput., vol. 22, no. 4, 2023.

[8] B.K. Osibo, et al., "An edge computational offloading architecture for ultra-low latency in smart mobile devices". Wireless Netw. 28, 2022.

[9] S. Song, et al., "Delay-sensitive tasks offloading in multi-access edge computing," Expert Systems with Applications, vol. 198, 2022.

[10] R. Xiong, , et al., "Reducing Power Consumption and Latency of Autonomous Vehicles With Efficient Task and Path Assignment in the V2X-MEC Based on Nash Equilibrium," IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 10, 2024.

[11] Y. Zhang, C. Chen, H. Zhu, J. Wang, "Task Offloading for MEC-V2X Assisted Autonomous Driving," 2024 IEEE 99th VTC, 2024.

[12] 3GPP, "5G Mobile System Architecture, TS 23.501, version 17.5.0," 3GPP Technical Specification, 2022.

[13] P. Zhao, Y. Chang, W. Wu, "Dynamic RRT: Fast Feasible Path Planning in Randomly Distributed Obstacle Environments," Journal of Intelligent & Robotic Systems, vol. 107. 2023.

[14] A. Vennelakanti, et al., "Traffic Sign Detection and Recognition using a CNN Ensemble," 2019 IEEE ICCE, pp. 1-4, 2019.

[15] S. Macenski, I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," The Jurnal of Open Source Software, vol. 6., 2021.

[16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 21 09 2018. [Online]. Available: https://arxiv.org/abs/1804.02767. [Accessed 12 07 2024].

[17] Z.-Q. Zhao, P. Zheng, S.-T. Xu and X. Wu, "Object Detection With Deep Learning: A Review," IEEE Transaction on Neural Networks and Learning System, vol. 30, no. 11, pp. 3212-3232, 2019.

[18] E. Hassan, M.-Y. Shams, N.-A. Hikal, "The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study," Multimed Tools Appl, vol. 82, pp. 16591-16633, 2023.

[19] S. Nazmul and A. Maida, "Enchancing ResNet Image Classification Performance by using Parametrized Hypercomplex Multiplication," 2023.

[20] M. A. Wani. et al., "Basic of Supervised Deep Learning," Advances in Deep Learning, pp. 13-29, 2020.

[21] A. Mao, et al., "Cross-entropy loss functions: Theoretical analysis and applications," International conference on Machine learning, 2023.

[22] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," Neural Networks, 2012.