Experience: Implementation of Edge-Cloud for Autonomous Navigation Applications

Yuvraj Chowdary Makkena*, Rajashekhar Reddy Tella†, Nisarg Parekh‡, Prem Kumar Saraf§, Annu‖, Hershita Shukla¶, Akhila Matathammal*, Sarat Chandra Sai Danda§, Praveen Chandrahas§, Akshay Ramesh Jadhav‖, Praveen Tammana§, Koteswararao Kondepu†, Rajalakshmi Pachamuthu‖

* TiHAN, IIT Hyderabad, Hyderabad, India

† Department of Computer Science and Engineering, IIT Dharwad, Dharwad, India

‡ Department of Smart Mobility, IIT Hyderabad, Hyderabad, India

§ Department of Computer Science and Engineering, IIT Hyderabad, Hyderabad, India

¶ Department of Integrated Sensor System, IIT Hyderabad, Hyderabad, India

‖ Department of Electrical Engineering, IIT Hyderabad, Hyderabad, India

Email: {yuvraj.makkena, praveent}@cse.iith.ac.in

***Abstract*—Autonomous navigation, while still facing a lot of challenges, has become a reality in the last few years. It has been successfully deployed in limited environments, showing the potential such technologies offer. The availability of increased computational power, coupled with advances in machine learning techniques, including deep learning, have enabled this success. However, there are many challenges that need to be overcome to enable *massive adoption of autonomous navigation* in all environments. One such challenge is to provide reliable, low-latency, and cost-effective data processing solutions for compute-heavy applications. To address this challenge, data processing near the data source, that is, at an edge cloud has been proposed. In this paper, we share our experience in implementing one such edge cloud, designed to bring compute power close to resource-constrained end devices in an autonomous navigation testbed, named as Technology Innovation Hub on Autonomous Navigation and Data Acquisition Systems (TiHAN). By considering some use cases, we show how deploying this edge cloud at the testbed has greatly benefited the performance of autonomous navigation applications.**

***Index Terms*—Edge-compute, Edge-cloud, Kubeedge, testbed, Autonomous navigation, MEC, UAV resource monitoring, Pedestrian detection, Obstacle detection**

I. Introduction

Over the last few years, there has been an enormous interest in autonomous navigation applications such as assisted driving, automated driving, delivery of goods via drones, etc [1]. There is a huge potential for these technologies, including humanitarian applications - from delivering essential medicines to remote areas, to aid in disaster relief works [2]. A drone or an Unmanned Aerial Vehicle (UAV) carrying out such a task would require access to not just satellite positioning system information, but would also need actively to learn about its surroundings to avoid obstacles and navigate safely.

One way this could be achieved is by image inference, with a camera on-board the UAV or UGV capturing images. These images need to be processed in near-real-time if navigation is to be achieved. Navigation and automated driving applications are extremely sensitive to delays in such processing. Such real-time or near-real-time inference is compute-heavy and typically requires access to a Graphical Processing Unit (GPU) for optimal processing [3].

Department of Science and Technology, Government of India

However, mounting these resources on a drone is impractical. GPUs are power-hungry, taking away limited and valuable battery power required for the flight of the UAV itself. In addition, adding these resources increases the weight of the UAV, decreasing its flight times. As newer hardware becomes available, it is impractical to replace the older hardware on these UAVs and UGVs. A similar case can be argued for Unmanned Ground Vehicles (UGV). While not as resource-constrained as UAVs, they are prone to some of the same problems.

Offloading these compute-heavy applications to a cloud is one option. However, public clouds are typically located in far-off places increasing the network latency from the source. This latency is directly related to the distance of the cloud from the source application and will only increase in case of a congested network. Achieving reliable, low-latency, and low-cost compute off-load is the key to enabling these applications.

As an alternative edge cloud moves the processing of data closer to the sources while doing away from the public cloud. Numerous technological benefits are inherent to this, including decreased latency, security, flexibility, and greater reliability to the applications. For instance, the edge cloud enables devices at (or close to the) network edge to instantaneously alert key personnel and equipment about mechanical problems, security risks, and other crucial situations so that prompt action can be taken.

The focus of the current paper is to present the work we have done in deploying a Multi-access Edge Cloud (MEC) (more details in sections II and III). This MEC was then used to provide services to compute-heavy, latency-sensitive autonomous navigation applications (more details in Section IV). We then present the results while using MEC, and compare them with the results obtained by using the public

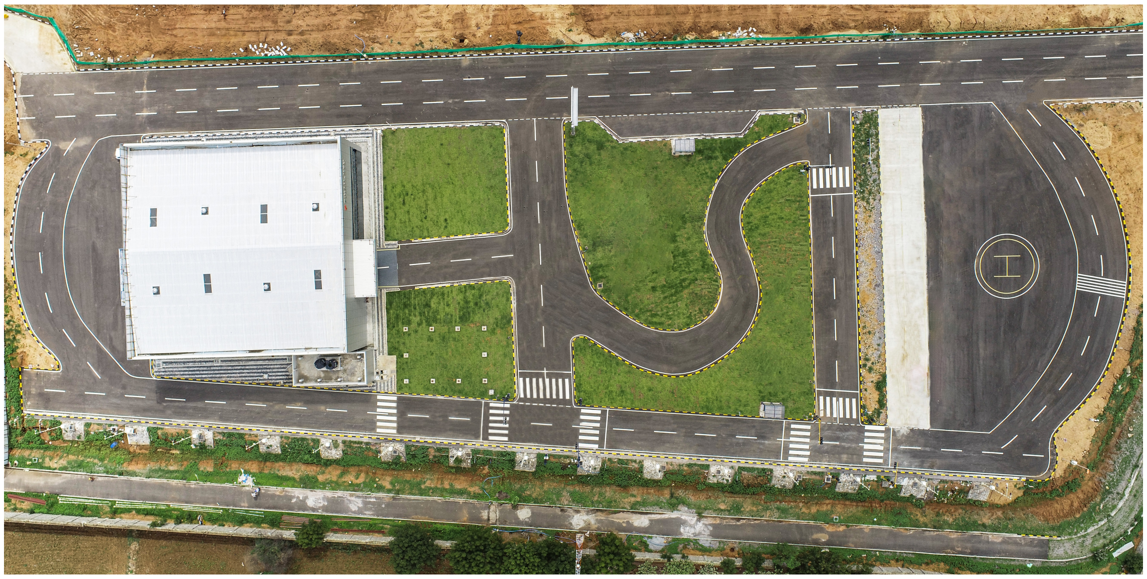


Fig. 1. TiHAN Testbed Layout

cloud and onboard processing. Section V validates our efforts in building the MEC. Our results show significantly better performance when the MEC is used. With this in hand, further work is described in Section VI.

II. Background

This section describes related work and our proposed MEC features.

In [4], the authors described democratizing the network edge, which focuses on the edge and discusses about the gaps to be filled at edge and access networks by exploiting open source initiatives in order to broaden to the area of research, and offers real-time insights. In [5], the authors compared Amazon AWS and Microsoft Azure edge computing platforms. This study shows that how both platforms offer equal performance for common workloads utilised in edge applications. However, both platforms differ in significant ways for other types of workloads. Comparison of the two platforms in terms of architecture, programmability, performance, and cost is crucial for making an informative decision between them. A benchmarking application called OpenRTiST has proposed in [6] for Augmented Reality. For the evaluation purpose, the authors performed tests on the edge cloud, the public cloud, and the device for the considered application. The reported results shown that the edge cloud outperforms public cloud.

Moreover, microservices offer both possibilities and difficulties for maximising utilisation and quality of service (QoS). They substantially modify a number of underlying presumptions used in the design of existing public cloud systems. In [7], an open-source DeathStarbench bechmark suite has proposed for end-to-end microservices. The authors investigated how the scale-out effects of microservice deployments in datacenters with hundreds of users. Thus, the development of MEC and the provision of services are crucial.

The deployment of our proposed MEC is described as follows:

Technology Innovation Hub on Autonomous Navigation (TiHAN) is an interdisciplinary research initiative on smart mobility technologies to enable autonomous navigation applications. As part of TiHAN, a test bed was set up at the Indian Institute of Technology (IIT) Hyderabad campus which has proving grounds, testing tracks, ground control stations, road infrastructure including smart poles, etc.

Our objective is to build and deploy a MEC to enable various autonomous navigation applications developed at TiHAN. This MEC needs to be geographically co-located with the test bed to achieve low latency. Communication from the User Equipment (UE) to the MEC is via different network access technologies that include wired ethernet, WiFi, DSRC, and other intermediary nodes (e.g., switches). The UEs have an On-Board Unit (OBU), which collects various kinds of data such as Global Positioning System (GPS) positions, the video feed from an on-board camera, Light Detection and Ranging (LiDAR) data, accelerometer, gyroscope, and other parameters in motion. This data is relayed to the MEC via a Road Side Unit (RSU) equipped with the network access technologies mentioned earlier.

Fig. 1 shows the outline of the TiHAN test bed layout. The test bed covers an area of $8100\ m^2$. It has 16 *smart poles* equipped with RSUs and connected to the switch using Optical Fibre Cables (OFC).

III. TiHAN Edge

In the conventional approach, an application communicates with a server providing service across a network that is typically deployed far from the user. Here, communication involves different steps such as request fulfillment, processing data, computing, and delivering the response. This approach does not satisfy the latency requirements of certain autonomous navigation applications. In contrast, in an edge cloud, the servers (*i.e.*, compute resources) are deployed closer to the user (application) where the data generated on the

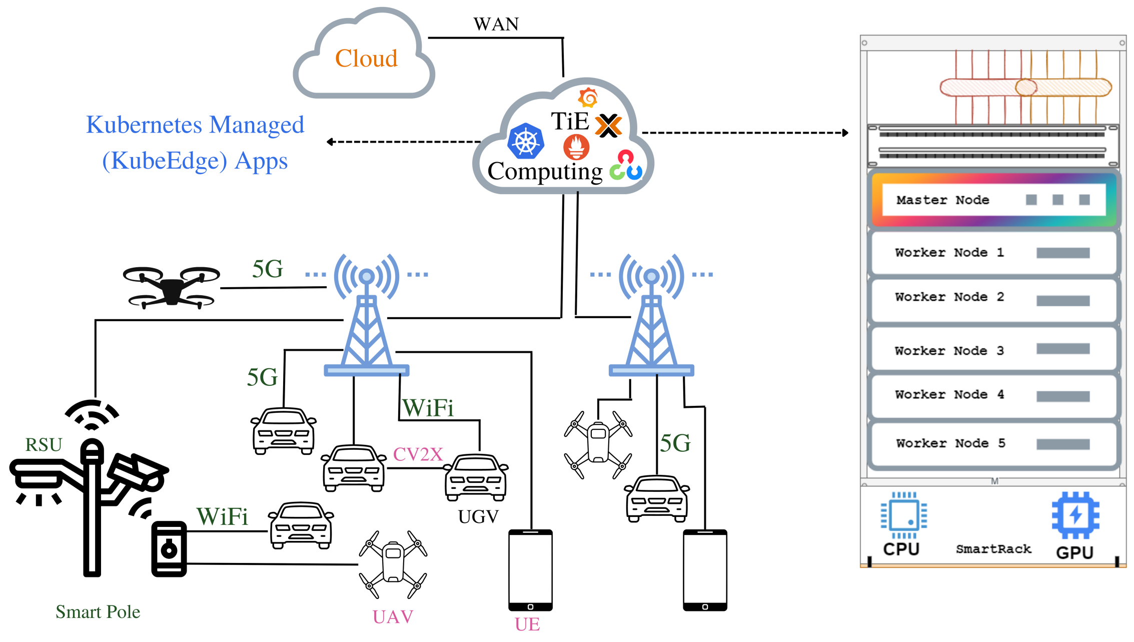


Fig. 2. TiHAN Edge (TiE) Network and Computing Architecture

network edge is processed relatively closer to the user. This approach not only reduces latencies significantly but also enables efficient sharing of edge cloud resources across many applications.

This section provides different networking and computing capabilities supported (and planned to support in the near future) by TiHAN Edge (TiE).

A. TiE Networking Capabilities

Fig. 2 shows the current schematic of TiHAN Edge (TiE) test bed capabilities, which includes devices such as UAVs, UGVs, and UEs (*e.g.*, mobile phones). These are generally connected to RSU deployed on the smart pole using different protocols such as WiFi, DSRC, and 5G-enabled CV2X technologies. Here, the RSUs connect with the Multi-Access Edge Cloud (MEC) — namely *TiE* — through intermediate TiE access switches. Fig. 3 shows the high-level abstraction of networking components deployed in the TiHAN testbed. The TiE access switch is the primary network source that provides connectivity to the whole test bed. Input to this switch is through an Optical Fiber Cable (OFC) laid from a distribution switch in the IITH campus. The access switch uses OFC to connect the smart poles. Each smart pole has a switch with 10 ports for providing connectivity to RSUs like DSRC boards, industrial WiFi, and other devices like IP Cameras mounted on the smart pole.

B. TiE Computing Capabilities

TiE supports different computing capabilities enabled with multiple hardware and software features. As shown in Fig. 2 (right side), currently, TiE is set up by using *Kubernetes* and supports one master node and multiple worker nodes, where the master node is capable of controlling the worker nodes (applications), and it takes care of cluster management

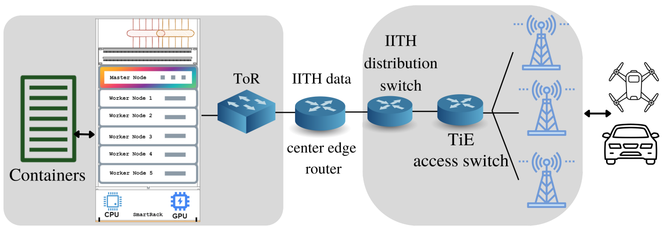


Fig. 3. TiHAN Physical Network TestBed

such as manage and configuring resources (*i.e.*, network, storage, assigning CPU and GPU cores, RAM allocation, etc). *Kubernetes* cluster allocates the required resources for running services associated with each application (*e.g.*, navigation, surveillance, etc). The *pods* are the smallest unit in a *Kubernetes* cluster with one or more containers running in each pod. TiE provides the freedom to use different container images to start services with *pods* like YAML Aint Markup Language (YAML) scripts, Helm charts, and the existing container images. *Kubernetes* can reserve one or multiple containers as required, and also scales the resource horizontally or vertically while maintaining replica sets, volume mount backups, etc.

In addition, the TiE also provides VMs and Linux Containers (LXCs) [8] with the help of ProxMox [9]. The reason for considering the Proxmox setup is to provide Graphical User Interface (GUI) access to users using Virtual Network Computing (VNC) since few applications requests GUI for their use cases like plotting and mapping. The TiE can

TABLE I
CURRENT TIE COMPUTING RESOURCES

	CPU	RAM	GPU	Storage
Workstation 1	12 Cores, 24 Threads, 3.7 Ghz	64 GB, 3200 MHz	Nvidia RTX A4000, Quadro P600	1 TB HDD, 256 GB M.2 SSD
Workstation 2	16 Cores 32 Threads, 4.0 Ghz	32GB, 3200 MHz	1 x NVIDIA Quadro RTX A4000	1 TB HDD, 256 GB M.2 SSD
Workstation 3	16 Cores 32 Threads, 4.0 Ghz	32GB, 3200 MHz	1 x NVIDIA Quadro RTX A5000	1 TB HDD, 256 GB M.2 SSD
Server	8 Cores 16 Threads, 3.2 Ghz	64GB, 3200 MHz	-	1 TB HDD, 256 GB M.2 SSD

TABLE II
TIE COMPUTING RESOURCES IN PROCUREMENT PIPELINE

Server	CPU (2 Socket Scalable)	RAM	GPU	Storage
1	32 Cores, 64 Threads, 3.6 Ghz	256GB, 3200 MHz	4 x Nvidia A100 80GB	6 x 2.4 TB SAS, 2 x 1TB M.2 SSD
2-5	32 Cores, 64 Threads, 3.6 Ghz	256GB, 3200 MHz	1 x Nvidia A40 48GB	4 x 1.8TB SAS, 2 x 1TB M.2 SSD
Total Compute	160 Cores, 320 Threads	1.28 TB	512 GB	43.2 TB SAS, 10TB SSD

also run centralized servers as services and assign different port numbers to each service. For example, we deployed a centralized Robot Operating System (ROS) server and expose it to UGVs.

Moreover, TiE also supports dedicated Graphics Processing Units (GPU) enabled *pods* for processing intensive applications. GPUs are shared using virtual GPU (vGPU) in order to support multiple service requests from different applications. TiE uses Free and Open Source (FOSS) components (Kubernetes and ProxMox), which are flexible and open to configure to our use cases. Both ProxMox and k8s are capable of leveraging GPU to container or VMs using Peripheral Component Interconnect Express (PCIe) pass-through or vGPU. TiE supports huge persistent data storage with volume mounts attached to the containers in order to store the user data (by preventing data loss).

The services running at the TiE can benefit to achieve low latency as the computing happens closer to the application. Table I shows the computing resources of the existing setup with one server and three workstations running Kubernetes cluster and ProxMox parallelly. The current setup is sufficient enough to run the existing services, but in the long run, there are many services we planned to containerize and run on TiE. So we started procuring the resources mentioned in Table II, where each server contains two CPUs each with 16 cores (32 threads), and 256 GB of RAM. In total, TiE supports 160 cores (320 threads) and 1.28 TB of RAM. Moreover, we have two node types, one equipped with 4*Nvidia A100* 80 GB cards and the others with 1 *Nvidia A40* 48 GB Card.

IV. TIE USECASES

We identified the following use cases to validate the performance benefits of the MEC: (1) *Pedestrian Detection*, (2) *Obstacle Detection for UAVs*, and (3) *Monitoring and Prediction of UAV resources*.

A. Monitoring and Prediction of UAV resources

UAVs are most frequently linked for various purposes, such as search and rescue, surveillance, traffic monitoring, weather monitoring, and firefighting. In recent years, UAVs have drawn a lot of interest due to their great mobility, low cost, and adaptable deployment. Low-altitude UAV-enabled wireless networks, in particular, may be easily established and flexibly adjusted to improve network coverage and capacity as compared to terrestrial networks and satellite remote communications [10], [11].

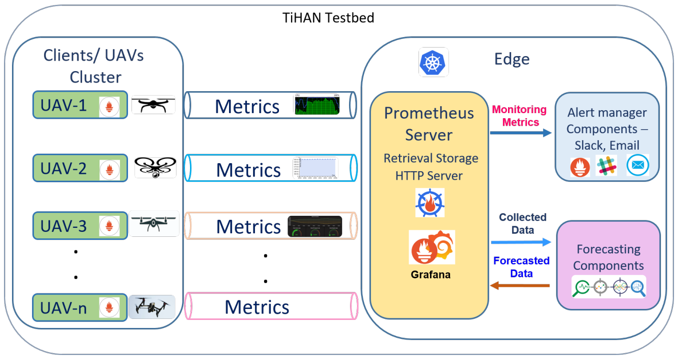


Fig. 4. Resource Monitoring Framework

Recently, monitoring [12], [13] and alerting have drawn more and more attention along with the expansion of information collected from the devices and the processed at the other end. The performance monitoring of all interconnected services is complicated unless there is a software tool that manages all incoming connections and monitors them efficiently. We use "Prometheus" [14] monitoring tool to monitor the system resources of connected devices. Metrics are continuously gathered from numerous services and aggregated for viewing and monitoring at various levels of granularity. The data received from the exporter (node exporter) can be visualized by using the "Grafana" [15] tool. The proposed monitoring framework is implemented at the Edge by using all open-source tools.

Fig. 4 shows the proposed monitoring framework implemented and evaluated at the TiE testbed. The framework is mainly classified into two parts: *Client/UAV* cluster and the *TiE*. Here, the *Client/UAV* cluster contains multiple UAVs, and the resource usage on these devices is exported periodically to the *edge*. On *TiE*, we deployed a monitoring framework consisting of *Prometheus*, *Grafana*, *Alert manager*, and *Forecasting* modules.

The detailed description of the cluster and modules is as follows:

UAV cluster: The Raspberry Pi boards are deployed at the considered UAVs. In this, specifically for monitoring purposes, a Node exporter is deployed to continuously export the system resource usage metrics to the Prometheus server deployed at the Edge. Node exporter is a collector agent that translates system-level metrics, such as CPU and memory usage, to time-series data format and consequently sends it to the Prometheus server for storage. Metrics from the Node exporter can be scraped based on pre-defined interval time. The requests from the applications running on the UAVs are processed and associated actions are performed by the UAV cluster.

Edge: The Kubernetes [16] based Edge deployed at the TiE is considered to implement the monitoring framework edge side modules. Kubernetes makes it possible to deploy containers in Platform-as-a-Service (PaaS) clouds, with a focus on cluster-based systems. Kubernetes supports to deploy multiple "pods". The following monitoring framework related pods are deployed:

1) Prometheus — It is open-source software for environmental monitoring and alerting [17]. It records real-time indicators in a Time Series Data Base (TSDB) built using the HTTP pull model and has a flexible query and instant alerting functions. It makes use of its own query language - Prometheus Query Language (PromQL) - which enables users to pick and aggregate data as well as perform conversions and manipulate metrics;
2) Grafana — It is an open-source system for multi-platform analysis and interactive visualization [15]. Grafana supports a considerable number of data sources to facilitate data visualization. The resource metrics of the UAV cluster are scraped by Prometheus at the Edge. The Grafana tool is used to display the metrics in a more organized way;
3) Alert manager — As the number of activities performed by the UAV increases and the required resources also increases. In order to ensure that all the activities at the UAV complete efficiently within the pre-defined threshold values, the *alert manager* feature in the Grafana can be used to notify the corresponding persons through different channels (e.g., slack, email) if any of the activity consumes more resources than the pre-defined threshold values. For example, CPU, memory and networking resource usage alerts can be created for each activity performed by the UAV.
4) Forecasting — It is a procedure used to make predictions or estimates about the system resources (e.g., CPU, Memory usage). The forecasting module could help to take the action based on the resource availability at the UAV to offload any additional activity. For example, if required to offload video capture activity to the UAV, it is important to know the available resource, hence a decision can be made to offload such activity or not. On other hand, forecasting helps to scale-in or scale-out the resource vertically or horizontally before offloading the activity. Thus, the forecasting module helps efficient utilization of resources. However, the wrong prediction may leads to over- or under-provisioning the resources. Thus, efficient predictions algorithms are required to avoid inefficient utilization of the resources. In this, different time series models are implemented to understand the prediction accuracy.

Results – Drone Usage Monitoring and Prediction Usecase: Fig. 5 shows the system resource captured with the Grafana dashboard at the TiE while UAV is in IDLE mode and FLYING mode. During the IDLE mode, the UAV consumes CPU usage of 38% and the RAM usage of 22% was observed. Note that some programs running on the Raspberry Pi also contributes to the reported system resource usage. In the case of FLYING mode, the UAV consumes CPU usage of 87% and memory usage of 23%. Here, around 40% of CPU usage increase can be observed when moving from IDLE to FLYING mode. However, there is no significant memory increase between the modes as the UAV is not performing any memory-hungry tasks.

Fig. 6 shows an example of slack alert generation. The following tasks are performed to generate a slack alert: (i) a sample slack channel is created; (ii) the channel is integrated into the Grafana dashboard; and (iii) configured the alert rules. Fig. 6 shows a sample CPU usage rule created by setting the value to 0.01 (as the UAV is in ideal mode). In order to generate an alert, the propellers of the UAV are powered on and the CPU usage of the UAV is increased to 0.432. Thus, the pre-defined CPU alert rules are triggered and generated alerts to the slack channel as shown in Fig. 6.

Fig. 7 shows an example of implemented forecasting CPU utilization. The following steps are performed to forecast system resources (CPU): (i) exported CPU utilization data (in the form of CSV) from the Grafana dashboard. For the evaluation purpose, a shell script is implemented to export the latest two hour data from the Grafana dashboard, and scraped samples every $2sec$. However, this periodicity can be adjusted according to the use case considered; (ii) the exported data is feed to the considered time series model — Long Short-Term Memory (LSTM) algorithm to forecast the CPU utilization of the UAV, and the LSTM parameters are configured as described in [18]. Fig. 7 shows the obtained results as a function of CPU utilization. The LSTM algorithm forecasts close to the actual values. The future work of forecasting will focus on implementing other forecasting models. The importance of forecasting in the considered monitoring framework could help in offloading additional activities (jobs) at the UAV, which are running under pre-defined system resource thresholds.

B. Pedestrian Detection

Future vehicles have the ability to share data with other vehicles (V2V), roadside infrastructure (V2I), the Internet (V2N), pedestrians (V2P), etc. Vehicle-to-Everything (V2X) has been proposed to describe all of these types of vehicular communication [19]. The automobile industry believes in two

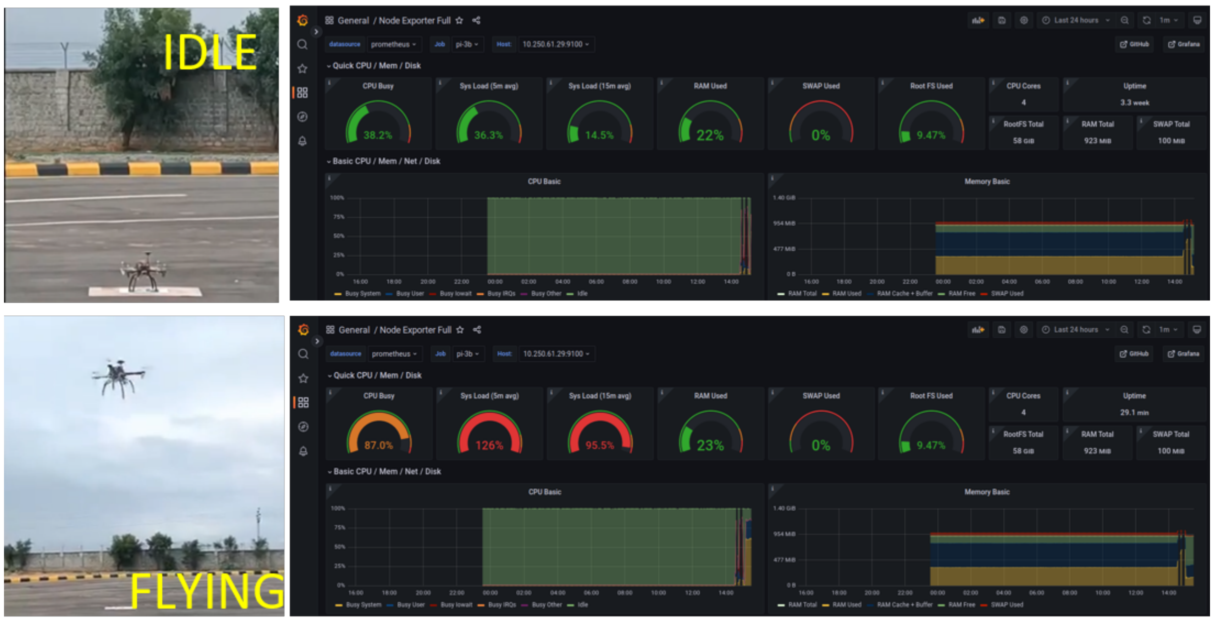
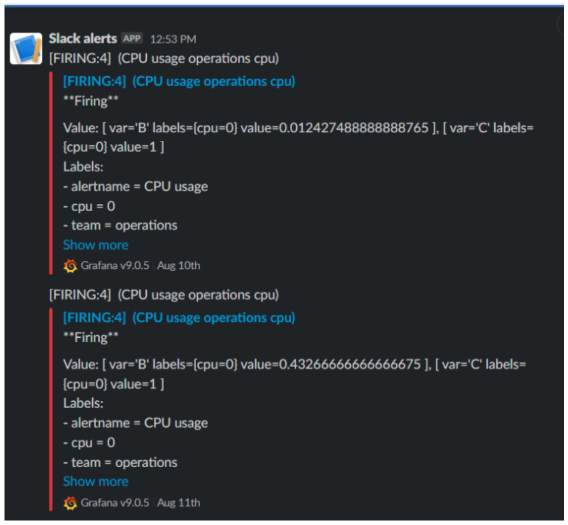


Fig. 5. Grafana Dashboard - IDLE and FLYING mode UAV system resources



Slack alerts APP 12:53 PM
[FIRING:4] (CPU usage operations cpu)
[FIRING:4] (CPU usage operations cpu)
Firing
Value: [var='B' labels={cpu=0} value=0.012427488888888765], [var='C' labels={cpu=0} value=1]
Labels:
- alertname = CPU usage
- cpu = 0
- team = operations
Show more
Grafana v9.0.5 Aug 10th

[FIRING:4] (CPU usage operations cpu)
[FIRING:4] (CPU usage operations cpu)
Firing
Value: [var='B' labels={cpu=0} value=0.43266666666666675], [var='C' labels={cpu=0} value=1]
Labels:
- alertname = CPU usage
- cpu = 0
- team = operations
Show more
Grafana v9.0.5 Aug 11th

Fig. 6. An example of slack alerts generation with Grafana-based alert manager

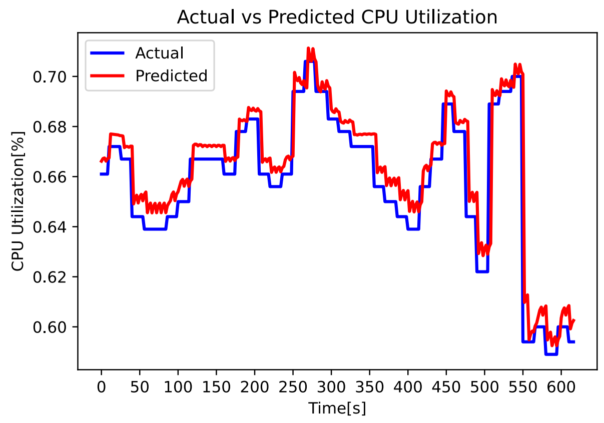


Fig. 7. Actual vs Predicted CPU usage

main trends with the relevance of the 5G and beyond as an automotive vision, such as automated driving and improvement in road safety services [20]. The latter could significantly lower the number of fatal accidents.

According to previous studies, around 50% of all fatal accidents happen at road junctions. Thus, improving communication between road signals and vehicles could help in reducing road fatal accidents. The objective of the proposed use case is to evaluate the performance of pedestrian detection. Here, TiE resources are exploited to understand the latencies involved to detect the pedestrian. Therefore, all calculations will be carried out at the TiE before being forwarded to the RSU which incorporates speed optimization and reduced latency.

For example, if there are 1000 vehicles, each with a camera, each vehicle must conduct Vulnerable Road Users (VRU) detection individually. Instead, all of the processing is done at the edge cloud using a single detection algorithm, and it broadcasts the data using DSRC communication through RSUs to all of the OBUs. This means the proposed use case with help of TiE reduces overall processing power with improved vehicle visibility as compared to the normal one.

In this use case, we employ a novel method of using traffic cameras to detect pedestrians. Fig. 8 shows the possible approach to detecting the pedestrian with help of IP-based cameras. Since the traffic cameras are fixed, their GPS positions are fixed and known. Based on the camera angle, when a pedestrian is detected, it is possible to identify the location of the pedestrian with a fairly high degree of accuracy. However, this approach would entail continuous processing of the video stream from the traffic to identify, detect and broadcast information about pedestrians to any UGVs. A system to continuously process a live video stream would

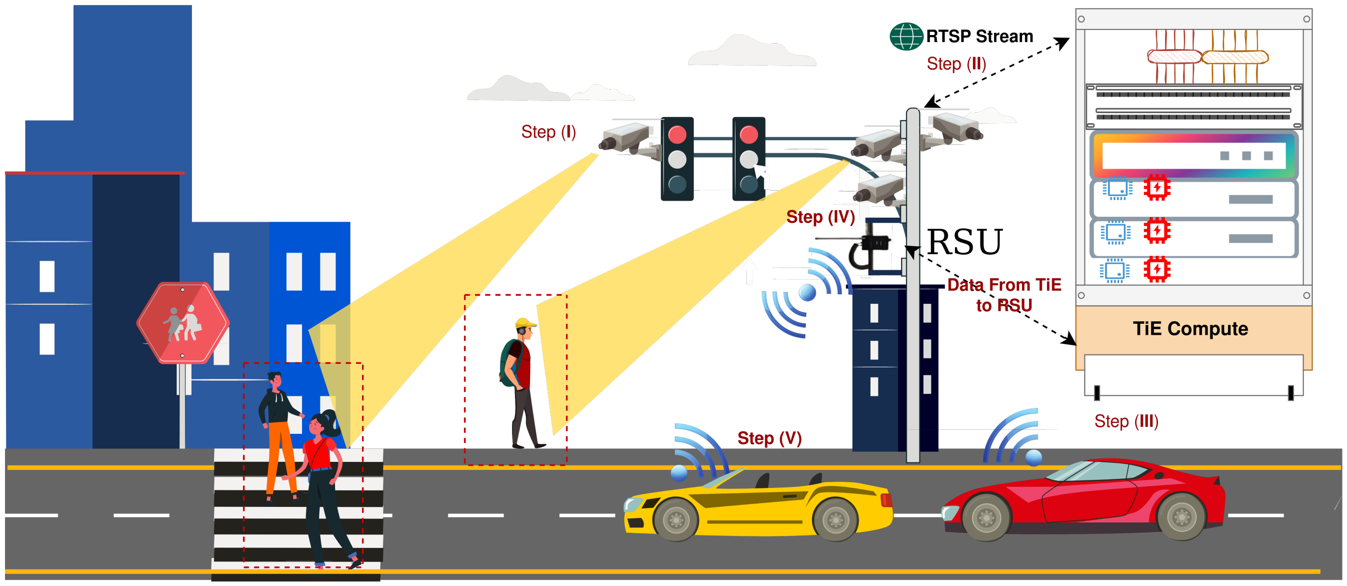


Fig. 8. Pedestrian Detection using IP Camera

TABLE III
RTT FROM TIE TO RSU AND AWS TO RSU

	RTT [in millisec]	
	TiE to RSU	**Public Cloud (AWS) to RSU**
Minimum	0.341	15.963
Average	0.468	19.74
Max	0.869	34.068
Mdev	0.16	6.404

require significant computing power. This is a task that is well suited to be deployed on the edge.

Fig. 8 shows the steps involved in the considered pedestrian detection use case: (i) a service is implemented running on the edge consuming the live video feed from the traffic camera; (ii) the live video is streamed with the help of Real Time Streaming Protocol (RTSP) for processing, detecting and computing the location of pedestrians: (iii) the computed processing information is forwarded to the RSU; (iv) Upon receiving the information, the RSU relay to any OBU within its vicinity or based on a publish-subscribe model.

OBU is a component that sends and receives DSRC transmissions and is mounted to a car. The OBU module has a quad-core NXP Semiconductor (formerly Freescale) i.MX6 processor with 1 GB of RAM and 4 GB of NAND memory. It features two IEEE 802.11p radios for DSRC communication: one is dedicated to safety applications and the other switches between control and other service channels to deliver DSRC-based applications.

RSUs are devices mounted on traffic signals or lamp posts near roadside intersections. These devices are alongside the road to communicate with traffic control systems and broadcast and receive DSRC communications (traffic signal controllers). It gets placed along the roadside to improve V2I communication. The RSU and OBU both have i.MX6 Quad Core processor, 1 GB RAM, and 4GB Flash. In addition, it functions with two DSRC radios, one for service and the other for safety. The RSU also has Ethernet interface for backhaul connectivity with the server, a built-in 4G modem, and Wi-Fi connectivity. A solar panel or a Power-Over-Ethernet (PoE) cable can offer a 12V DC supply voltage to the RSU if the link is an Ethernet connection.

Results – Pedestrian Detection Usecase: Two different computing setups are considered to understand the impact of latency. One from the TiE and the other from the public cloud (Amazon Web Server — AWS) instance located in Mumbai, India as this is geographically the closest location from the service provider to the TiHAN testbed. The deployed AWS instance is configured with 4 Cores and 8GB RAM, and similar configurations are also considered for the reserved container at the TiE.

As an initial setup, the focus is devised to understand the latency, thus the Round Trip Times (RTTs) are evaluated from RSU to both TiE and the public cloud. The results of the RTT comparison are summarised in Table III. On average, the latency measured at the TiE is 40 times lesser than that of a public cloud server. The huge difference in the network latency means that the end-to-end latency (i.e., RTT and compute time) will still end up being lower for the edge cloud scenario. Note that the live steam video processing and detection time will be added to the latency reported in Table III. The use case is implemented and tested on pedestrian detection using the YOLO-V5 ML model trained on Coco dataset [21] [22].

C. *Obstacle Detection for UAVs*

In the previous section, we outlined how a pedestrian detection service might be designed and deployed on the TiE. For a more general-purpose application that can be used by both UAVs and UGVs, we consider obstacle detection as a use case. This experimental setup is designed to detect

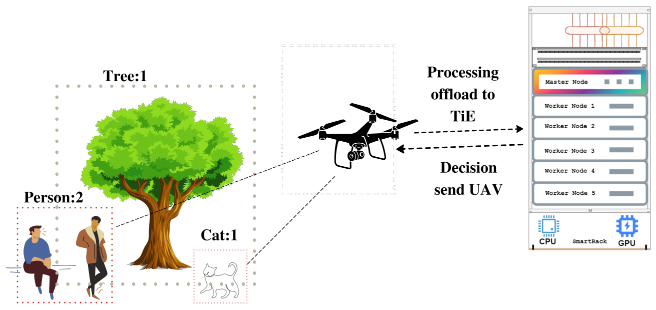


Fig. 9. Obstacle Detection from UAV

obstacles using the UAV (enabled with Raspberry Pi) deployed in the TiHAN testbed. Note that the same be extended to other applications —- object detection for video surveillance, mapping, object or obstacle detection for UGVs, etc. We have chosen the UAV setup to demonstrate the benefits of edge cloud in a resource-constrained environment.

In this, three different computing setups are considered to understand the impact of the processing and latency. The three considered computing resources are located at: (i) the TiE; (ii) the public cloud; and (ii) the on-board (Raspberry Pi). As stated earlier, the AWS instance is also considered for obstacle detection use case. Note that the similar tests have been performed in evaluating a benchmarking application called OpenRTiST [6].

Fig. 9 shows the obstacle detection using the UAV. An onboard camera on the UAV takes images at frequent intervals (every 500ms) and is transferred to the TiE/AWS/On-board in the current experimental scenario. The images were taken in different detail levels (resolutions), producing images of varying sizes. Images with higher resolution have higher detail and even smaller objects can be identified. However, this comes with a cost - larger image size and more computation time. On the other hand, a low-resolution image will have a smaller size and will need lower computing power, but will miss out on details and some objects may not be identified.

Results —- Obstacle Detection Usecase: The obstacle detection code is deployed at the TiE, reserved AWS, and Onboard Raspberry pi. The UAV located in the TiHAN testbed is used as a client and started detecting the objects. This code uses a REST API call and use 7 images of sizes ranging from 8.2 KB to 10.5 MB. The experiment is repeated for 50 times for each image and measured the end-to-end time, that is, the total time taken from the request initiation to the time response was received. We considered three different image resolutions for this use case: a low-resolution image (8.2KB), a medium-resolution image (2.4MB), and a high-resolution image (10.5MB). For objection detection, we used the YOLO-v3 ML model trained on the coco dataset [23]. The results are summarized in Table IV. For all image sizes, the results show that there is a clear benefit of using the edge cloud (TiE) compared to the public cloud and Onboard processing. Fig. 10 shows transfer time (*i.e.*, response time) as a function of multiple image sizes. This also confirms the advantage of using edge cloud deployment independent of the image size.

V. Discussion on Technical and Practical Challenges

A. GPU Sharing

Sharing GPU resources across applications is not a straightforward task. The applications mentioned above, like pedestrian detection, telecontrol, or obstacle detection, have image inference components requiring heavy GPU resources. The key question is *How do we share a GPU across multiple applications?*. One approach is to use pass-through mode, which is a technology in most hypervisors. They use PCIe pass-through mechanism and share GPU from the host machine to the VM or container directly. The second option is vGPUs in NVIDIA cards which can slice up the GPU into multiple cores and assign them to the VM or container requesting GPU resources. Since vCPU is quite common, this notion of vGPU being common did not work – the number of types of GPU cards supporting this feature is limited and expensive, and currently, only a few types of cards from NVIDIA support vGPU.

B. Latency

The acceptable end-to-end latency for 5G URLLC applications is in the order of tens of milliseconds, whereas the observed latency in TiE is in the order of hundreds of milliseconds (*e.g.*, image inference at TiE). This requires careful inspection of packets both at each hop and across hops. However, it is challenging to understand, correlate, and pinpoint the hop(s) causing significant delays because intermediate hops are heterogeneous devices with hardware from different vendors, thus, they must be time synchronized with either external or internal sources. Initially, we tried with an external UTC as a source, but accuracy is not always guaranteed. Thus, without time synchronization, analyzing delays at the milliseconds level is infeasible or hard. Moreover, software running on some embedded devices provides timestamps at the granularity of seconds, not at milliseconds. In our future work, we plan to profile per-hop latencies and reduce delays either by optimizing the device code or by using a fast network path.

VI. Conclusion and Future work

In this paper, we presented TiE, an edge cloud for autonomous navigation applications, and demonstrated three use cases that leverage compute resources at TiE and offloaded compute-heavy tasks. As a next step, we plan to identify a small set of common services essential for a wide range of autonomous navigation applications and expose those services as APIs. By doing so, any UAV/UGV, irrespective of hardware or software running on board, can make calls to these APIs, thus making UAV/UGV software management easier while at the same time enabling the services to a large number of devices. For example, consider image inference.

TABLE IV
END-TO-END LATENCY: TIME TAKEN (IN SECONDS) TO TRANSFER, PROCESS, AND DETECT OBJECTS IN AN IMAGE.

	Small Image [8.2kb]			Medium Image [2.4MB]			Large Image [10.5MB]		
	TiE	Public Cloud	On board	TiE	Public Cloud	On board	TiE	Public Cloud	On board
Minimum	0.324	0.635	4.682	0.694	1.129	5.898	1.443	2.006	7.958
Average	0.343	1.784	9.503	0.695	2.117	5.944	1.443	2.477	7.982
Median	0.343	1.784	9.503	0.695	2.117	5.944	1.443	2.477	7.982
Maximum	0.345	2.904	14.244	0.696	3.105	5.99	1.444	2.949	7.982
90%	0.345	0.955	4.781	0.705	2.25	5.956	1.464	3.121	8.099

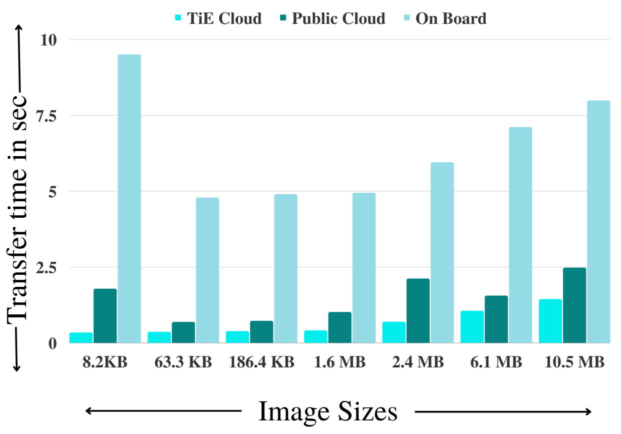


Fig. 10. Average Transmission time of TiE, public cloud, and onboard

There are multiple micro-tasks, including but not limited to traffic light detection, traffic sign identification, road marking detection, scene understanding, and depth perception. These tasks can be deployed as individual services on the edge cloud, in the spirit of micro-services, and process requests from UAVs/UGVs. Also, we plan to profile the performance of exposed services, especially those that require multiple interactions among micro-services, and evaluate performance under different loads. Enabling private 5G for the testbed and securing the exposed services are also part of our future work.

VII. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their insightful feedback. We also thank all team members involved in building the TiHAN testbed. This work was supported by DST National Mission Interdisciplinary Cyber-Physical Systems (NM-ICPS), Technology Innovation Hub on Autonomous Navigation and Data Acquisition Systems (TiHAN) at Indian Institute of Technology (IIT) Hyderabad.

REFERENCES

[1] E. Frachtenberg, “Practical Drone Delivery,” *Computer*, vol. 52, no. 12, pp. 53–57, 2019.
[2] B. P. A. Rohman, M. B. Andra, H. F. Putra, D. H. Fandiantoro, and M. Nishimoto, “Multisensory Surveillance Drone for Survivor Detection and Geolocalization in Complex Post-Disaster Environment,” in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 9368–9371.
[3] C.-C. Yeh, Y.-L. Chang, P.-H. Hsu, and C.-H. Hsien, “GPU Acceleration of UAV Image Splicing Using Oriented Fast and Rotated Brief Combined with PCA,” in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 5700–5703.
[4] L. Peterson, T. Anderson, S. Katti, N. McKeown, G. Parulkar, J. Rexford, M. Satyanarayanan, O. Sunay, and A. Vahdat, “Democratizing the Network Edge,” *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 2, p. 31–36, may 2019.
[5] A. Das, S. Patterson, and M. P. Wittie, “Edgebench: Benchmarking edge computing platforms,” 2018. [Online]. Available: https://arxiv.org/abs/1811.05948
[6] S. George, T. Eiszler, R. Iyengar, H. Turki, Z. Feng, J. Wang, P. Pillai, and M. Satyanarayanan, “OpenRTiST: End-to-End Benchmarking for Edge Computing,” *IEEE Pervasive Computing*, vol. 19, no. 4, pp. 10–18, 2020.
[7] Y. Gan, Y. Zhang *et al.*, “An open-source benchmark suite for microservices and their hardware-software implications for cloud amp; edge systems,” in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 3–18.
[8] D. Bernstein, “Containers and Cloud: From LXC to Docker to Kubernetes,” *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.
[9] Proxmox. [Online]. Available: https://www.proxmox.com/en/
[10] S. Bonafini, C. Sacchi, F. Granelli, R. Bassoli, F. H. Fitzek, and K. Kondepu, “3D Cloud-RAN Functional Split to Provide 6G Connectivity on Mars,” in *2022 IEEE Aerospace Conference (AERO)*, 2022, pp. 1–13.
[11] S. Bonafini, C. Sacchi, R. Bassoli, K. Kondepu, F. Granelli, and F. H. Fitzek, “End-to-end performance assessment of a 3D network for 6G connectivity on Mars surface,” *Computer Networks*, vol. 213, p. 109079, 2022.
[12] K. Hitchcock, *Monitoring*. Berkeley, CA: Apress, 2022, pp. 203–240.
[13] H. Fathoni, H.-Y. Yen, C.-T. Yang, C.-Y. Huang, and E. Kristiani, “A Container-Based of Edge Device Monitoring on Kubernetes,” in *Frontier Computing*, J.-W. Chang, N. Yen, and J. C. Hung, Eds. Singapore: Springer Singapore, 2021, pp. 231–237.
[14] N. Sukhija and E. Bautista, “Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus,” in *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation)*, 2019, pp. 257–262.
[15] Grafana. [Online]. Available: https://github.com/grafana/grafana
[16] C.-C. Chang, S.-R. Yang, E.-H. Yeh, P. Lin, and J.-Y. Jeng, “A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning,” in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.
[17] Prometheus. [Online]. Available: https://prometheus.io
[18] J. Martín-Pérez, K. Kondepu, D. De Vleeschauwer, V. Reddy, C. Guimarães, A. Sgambelluri, L. Valcarenghi, C. Papagianni, and C. J. Bernardos, “Dimensioning V2N Services in 5G Networks Through Forecast-Based Scaling,” *IEEE Access*, vol. 10, pp. 9587–9602, 2022.
[19] V2X. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White-Paper-on-Automotive-Vertical-Sectors.pdf
[20] [Online]. Available: https://www.itu.int/hub/2020/05/to-5g-or-not-to-5g-automotive-safety-may-hang-in-the-balance/
[21] Coco Dataset. [Online]. Available: https://cocodataset.org/
[22] Yolo v5. [Online]. Available: https://github.com/ultralytics/yolov5
[23] Yolo v3. [Online]. Available: https://github.com/ultralytics/yolov3