INDIAN INSTITUTE OF TECHNOLOGY DELHI HAUZ KHAS, NEW DELHI-110016

Dated: 03/10/2024

Open Tender Notice No.IITD/BCSE (SP-4764)/2024

Expression of Interest

Transfer of Technology from Prof. Smruti Ranjan Sarangi's research group (IIT Delhi)

1. Introduction: Over the last several years Prof. Smruti R. Sarangi's group in the Computer Science and Engineering Department (https://srsarangi.github.io/) at IIT Delhi has developed several technologies that are ready to be commercialized. This EoI (Expression of Interest) is to solicit interests from potential buyers. Specifically, the aim is to solicit requests from interested parties and then have an open discussion with them to decide the roadmap for commercialization. There are many strategies possible. Some technologies can be auctioned to the highest bidder with an upfront payment. Some technologies that are not very mature can go for joint development with a non-exclusive tech. transfer agreements. In such cases, royalty sharing in the future can be looked at without upfront payments. Given the complex nature of such an endeavour, for every technology, it is necessary to create a bespoke mechanism for commercialization. Note that this is not a tender nor a formal request for bids.

Interested buyers/bidders can simply respond to the EoI on CPP eProcurement portal or simply send an e-mail to srsarangi@cse.iitd.ac.in

A brief description of the technologies are as follows.
Brief list:

1. An efficient stereo-vision system for mobile robots and drones [link]
2. Carla-based self-driving car simulator [link]
3. Drone-Swarming Simulator [link]
4. Topological Placement Tool [link]
5. Car-Data Logger [link]
6. Secure Compression + Encryption Accelerator [link]

SCHEDULE

Name of Organization	Indian Institute of Technology Delhi
Tender Type (Open/Limited/EOI/Auction/Single)	EOI
Tender Category (Services/Goods/works)	Service
Type/Form of Contract (Work/Supply/ Auction/Service/Buy/Empanelment/ Sell)	EOI
Date of Issue/Publishing	03/10/2024 (15:00 Hrs)
Document Download Start Date and Time	03/10/2024 (15:00 Hrs)
Online Pre Bid Meeting	---
Last Date and Time for Uploading of Bids	24/10/2024 (15:00 Hrs)
Date and Time of Opening of Technical Bids	25/10/2024 (15:00 Hrs)
EMD	**NIL.** However, bidders are required to submit 'Bid Security Undertaking' (Annexure-I)
Bid Validity days (180/120/90/60/30)	90 days from the date of opening of Technical bids
Address for Communication	Prof. Smruti R. Sarangi, Computer Science and Engineering, IIT Delhi Hauz Khas-110016
Contact No.	011-2659 7065
Email Address	srsarangi@cse.iitd.ac.in

2. Technologies

1. An efficient stereo-vision system for mobile robots and drones:

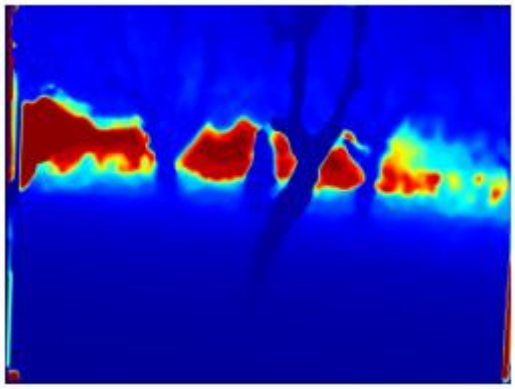
A stereo-camera comprises a pair of two cameras that are used to estimate the depth of objects in the scene. They are extensively used by drones and mobile robots to understand their environment. The challenge is to accurately estimate the depths of different objects in the scene. There is a need to apply different algorithms to "fix" the depth image. The current state-of-the-art systems have the following shortcomings:

i. ML-based algorithms are very slow (less than 2 frames per second).
ii. Non-ML algorithms are faster (17 frames per second), but their quality is very poor. Even the default package that comes along with the Intel RealSense D400 camera finds it very hard to fix all the holes in the depth image.

Our novel approach (link to preprint) is based on sophisticated techniques based on point-cloud fusion and template matching.

We achieve the following:

i. A consistent 27 FPS throughput on NVIDIA Jetson Nano (using Intel RealSense D400).
ii. 31% better quality (PSNR/SSIM) than the state-of-the-art.



iii. Even as compared to the best non-ML solution (17 FPS), our system is much faster (at 27 FPS).

We will transfer the C++ and CUDA code to an interested party.

3. Carla-based self-driving car simulator

We have created a simulator for self-driving cars based on the open-source Carla simulator. Our simulator was initially used for research in several self-driving technologies. Now, the code is reasonably mature and can be transferred to third parties.

We have implemented the following in our simulator:

1. The Perception Stack
    a. Lane detection
    b. Object detection and tracking
    c. Traffic light detection
    d. Localization (SLAM)
2. Motion planning and control
    a. Global planning
    b. Local planning
    c. PID controller for motion control
3. Agents
    a. Manual (possible to drive) using the keyboard or an attached steering device
    b. Carla (default)
    c. Our autonomous driving agent

Figure 1: Obstacle detection (dynamic)

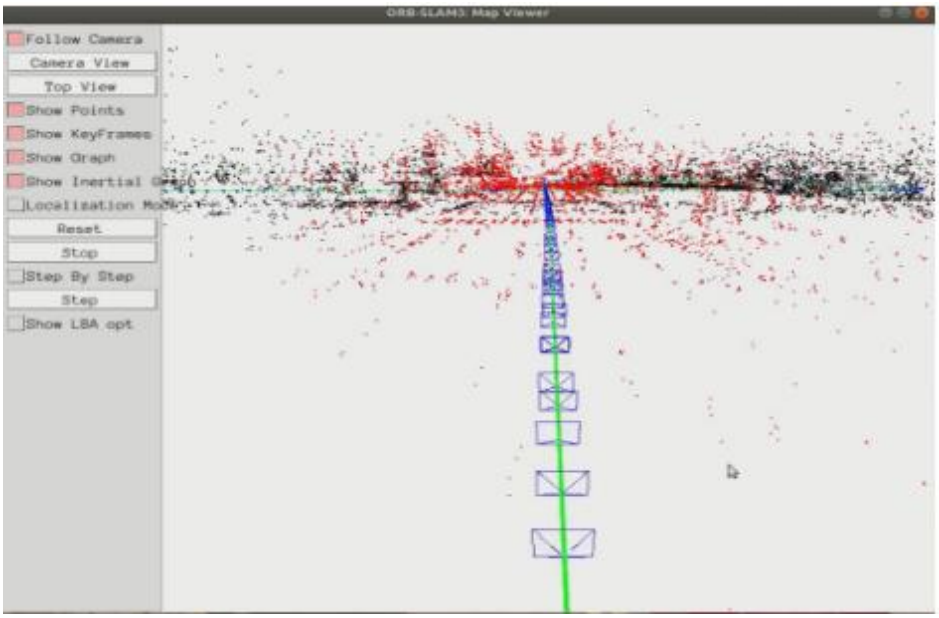


Figure 2: ORB-SLAM result

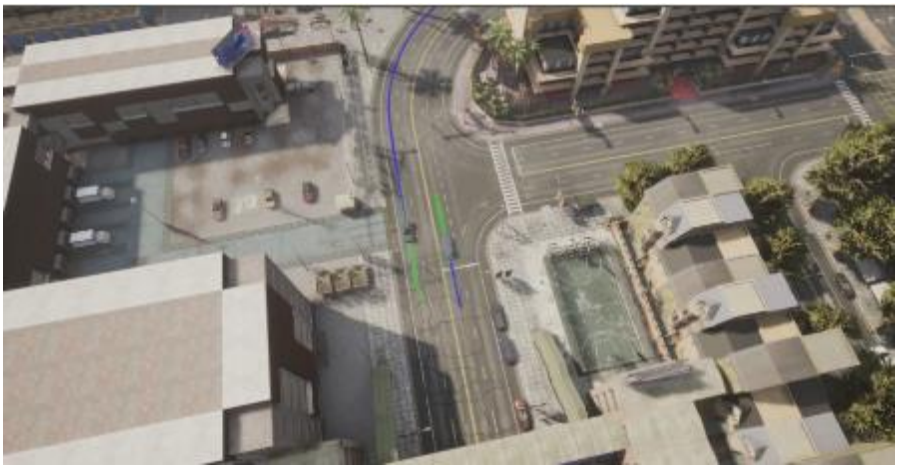


Figure 3: Object tracking and motion planning

4. Drone-Swarming Simulator

Drone swarming is expected to be a major driver of innovation in the space of drones and mobile robots. Often it is not possible for a swarm of drones to fly with continuous GPS guidance. GPS signals provide an approximate location reference and there are a lot of GPS-denied environments as well. Furthermore, there could be obstacles. We want the entire drone swarm to fly together, navigate obstacles, and continue to either follow a set path or follow the leader in all such situations. Even if the set of drones get partitioned and separated due to obstacles, we expect that they should join the swarm as soon as they can. To the best of our knowledge, there is no theory or simulation that can achieve all this.

We published a research paper in this area with new theoretical results. We have a Unity-based simulator that implements the paper. It should prove to be a valuable resource for groups working on autonomous drones or mobile robots.

Figure 4: A set of nodes flying in a cityscape

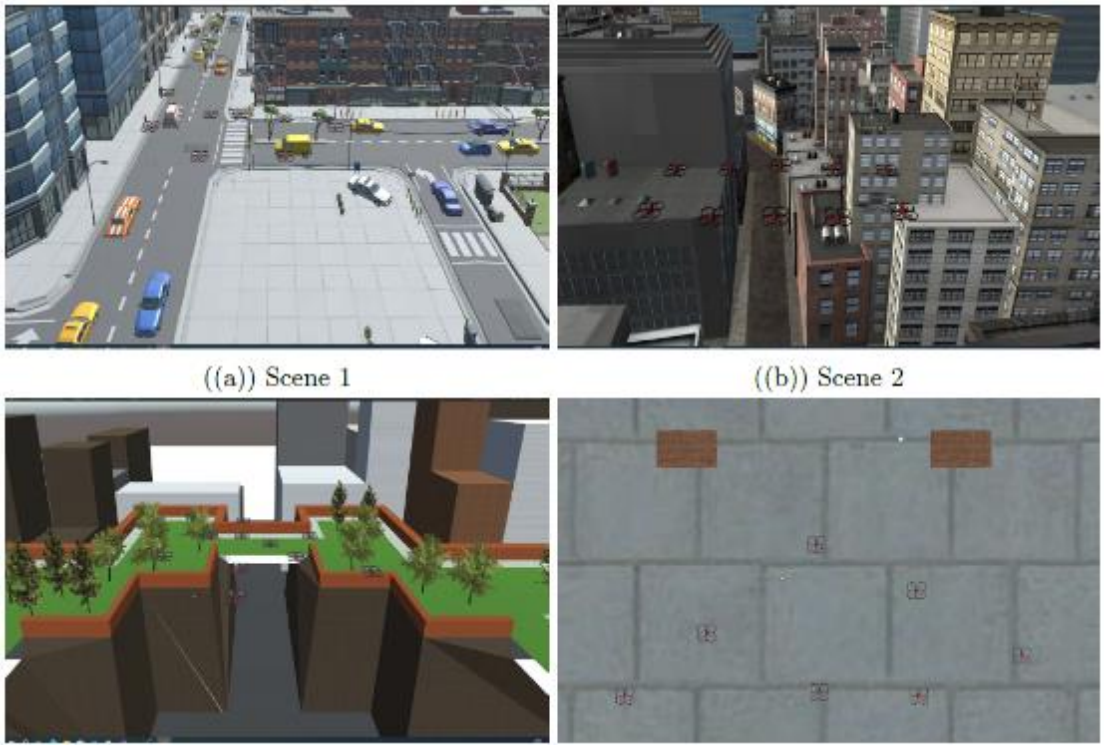


Figure 5: Different scenarios in our simulator

We will transfer the entire code to any interested party (subject to the terms decided in the EoI meeting). The algorithm was implemented on a Beaglebone Black Board. The frame rate was approximately 75 FPS (frames per second).

To detect and navigate around obstacles, sense the position relative to other drones, and continue to navigate without GPS signals, each drone relies on depth maps (related to product #1: stereo-camera).

5. Topological Placement Tool

This placement tool [published in Smart Cities 2023, link] was originally designed to place charging stations in a city in a quick and very intuitive manner. The user simply needs to specify the demand points (points of high demand). The GUI-based tool that relies on OpenStreetMaps uses a bunch of algorithms to find the best possible placement of charging stations. The entire process can be conveniently visualized by the user of the tool. It can be used for any kind of facility location problem. It is per se not limited to charging station placement.

It first uses a clustering algorithm to partition the network into intuitive clusters. There are many options for clustering the network (based on OpenStreetMaps). The best algorithm is the ToMaTo clustering algorithm that uses results from theoretical topology. Once the clustering is done, the demand points can be specified.

Subsequently, an ILP solver can be used to find the best possible locations to place the facilities such as charging stations. The basic reason for the speedup is that the tool uses the medial axis transform and persistent homology-based methods. We divide the entire

city into a set of 5 basic shapes (subject to certain topological transformations). We compute solutions for each of these shapes and appropriately modify them for the actual shape that is there in the layout of the city. After a round of interactive analyses, the final assignment is generated. We have tried this method for the largest 50 cities of the world and gotten good results.

Extending the framework to model other general facility location problems is quite easy.

Some screenshots follow:

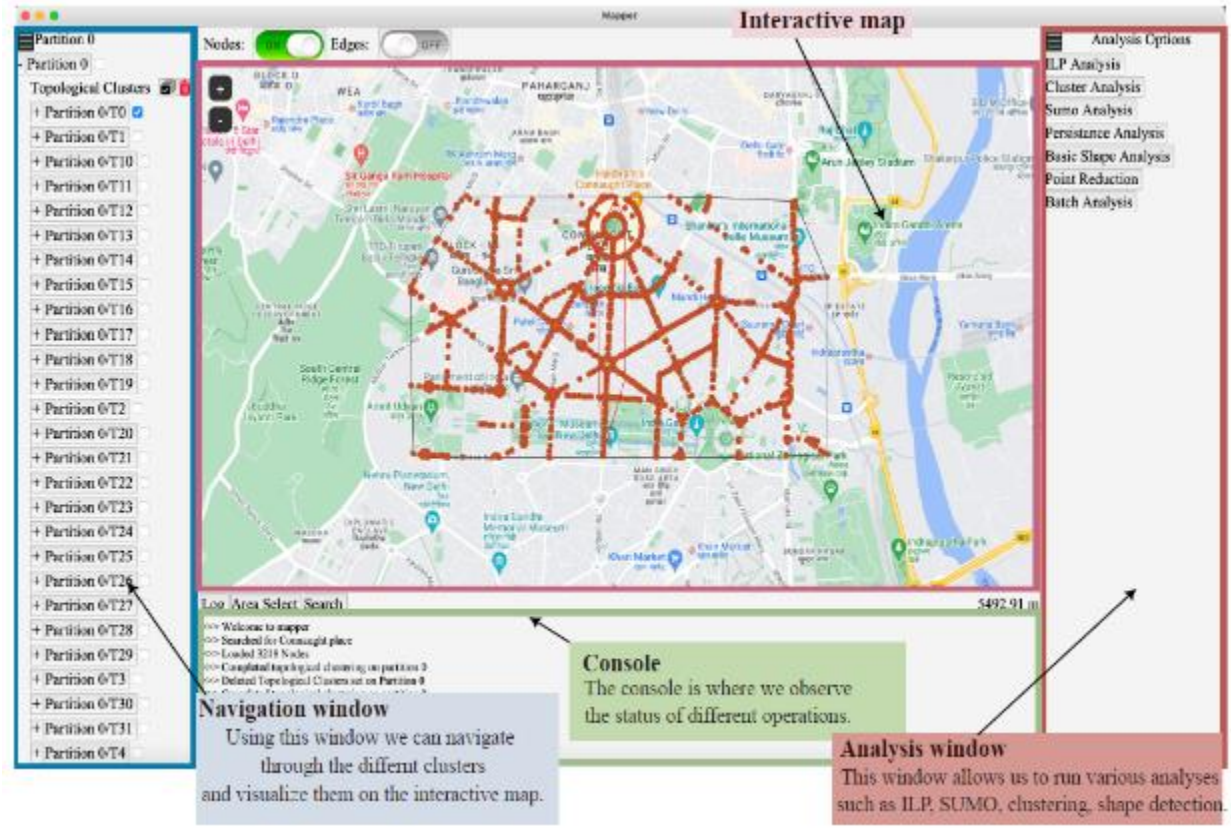


Figure 6: Layout of the tool

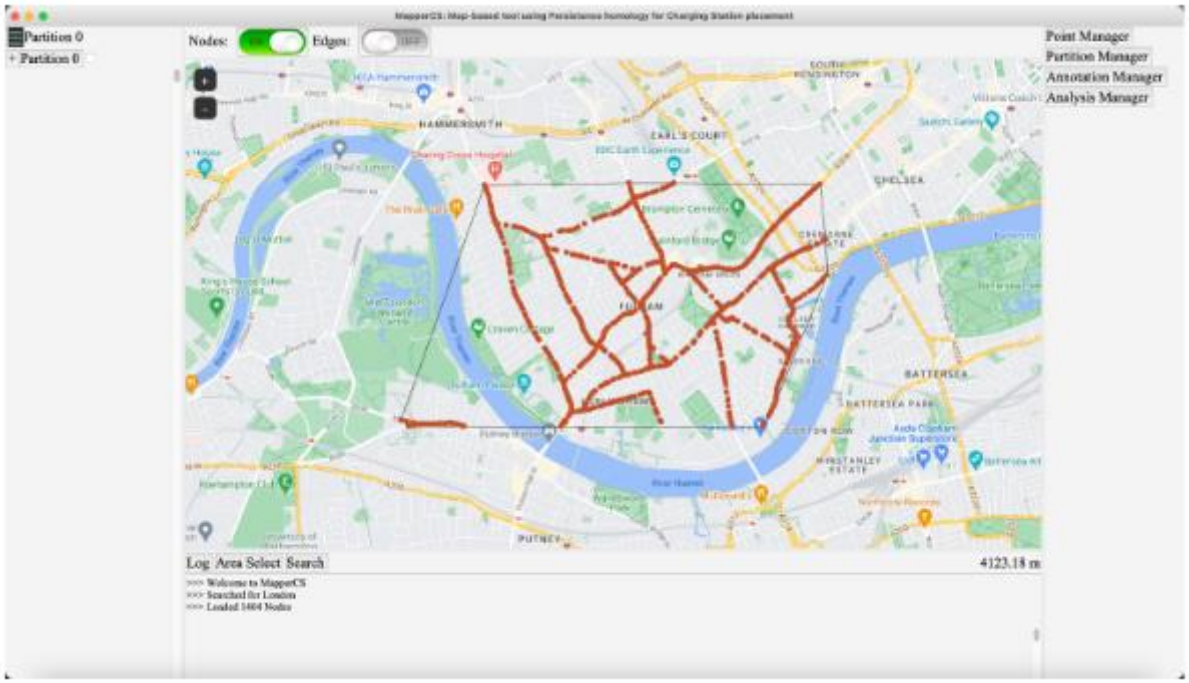


Figure 7: A demonstration of the area select feature

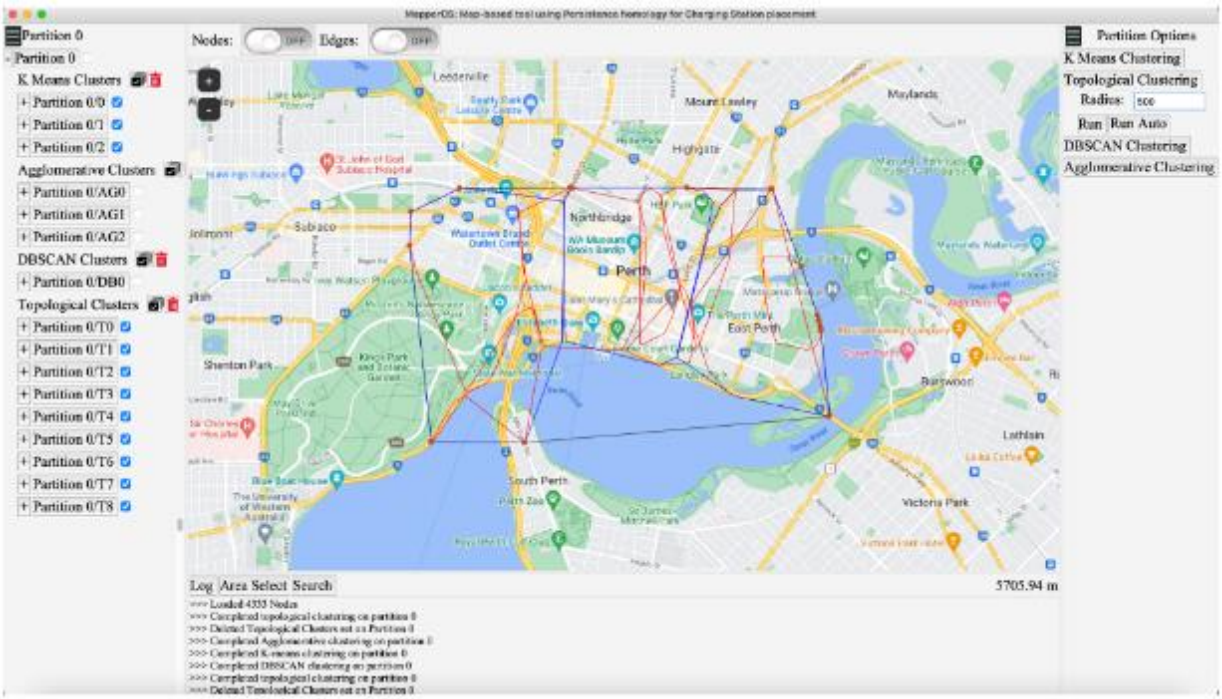


Figure 8: Analysis of Perth (after topological clustering)

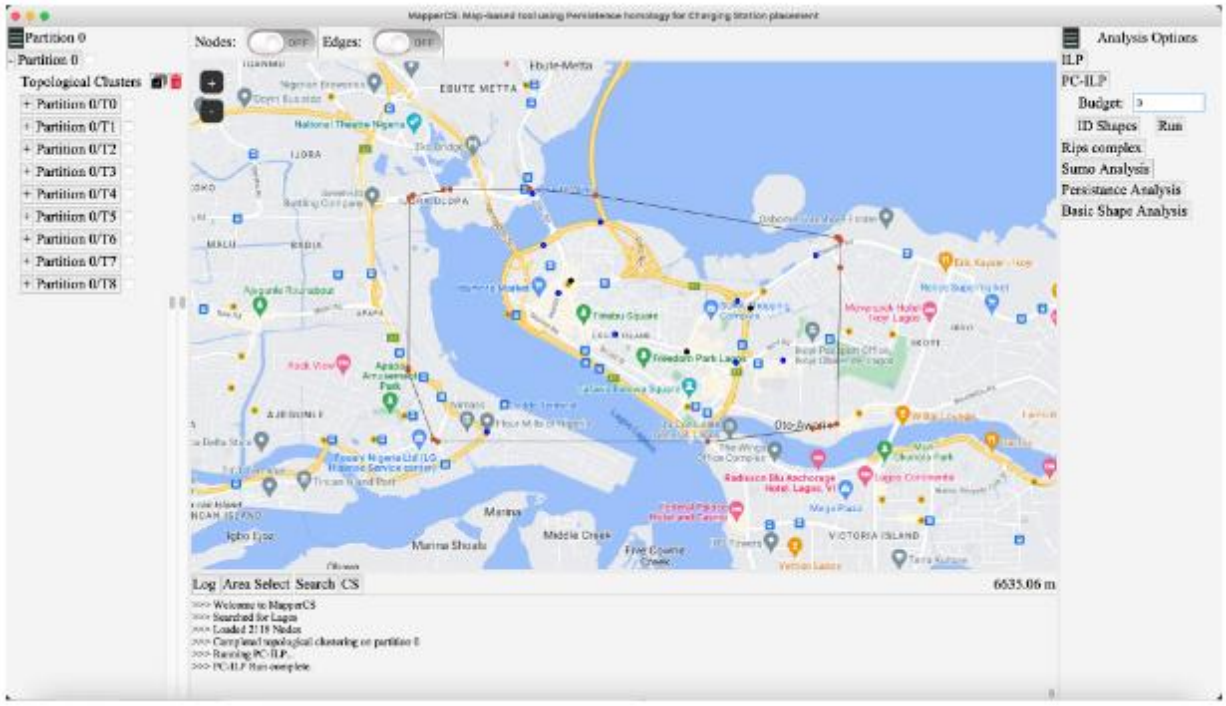


Figure 9: A demonstration of our proposed algorithm on a layout of Seoul

6. Car-Data Logger

This is a data logging solution for CAN bus data in vehicles. Our aim is to record all the CAN bus signals in a vehicle such that they can be used for analysis later. This is important for identifying driving patterns, forensic investigations (post an accident) and for vehicle diagnostics [paper published in International Conference on Intelligent Transportation Systems, 2023, link]. Unfortunately, current state-of-the-art solutions in this space have the following shortcomings.

1. ECUs generate data at about 500 kbps. This can easily overwhelm the capacity of automotive-grade SD cards or FRAM memories. Having larger memories of this kind can increase the costs significantly particularly in the cost-sensitive market.
2. Current Event Data Recorders (EDRs) do not have facilities to compress the data and store them in a lossy format, where the degree of loss increases with time.

The features of our novel solutions are as follows:

1. Lossy + lossless storage. The degree of information loss depends on time.
2. A novel online data compression algorithm.
3. Implementation on the ARM Beaglebone board.
4. Can store 3.2X more data than conventional solutions (at line speed).
5. A traditional EDR with a lossless fills a 4 GB log in 3 days, whereas our system fills it in 10 days.
6. It preserves most anomalies.

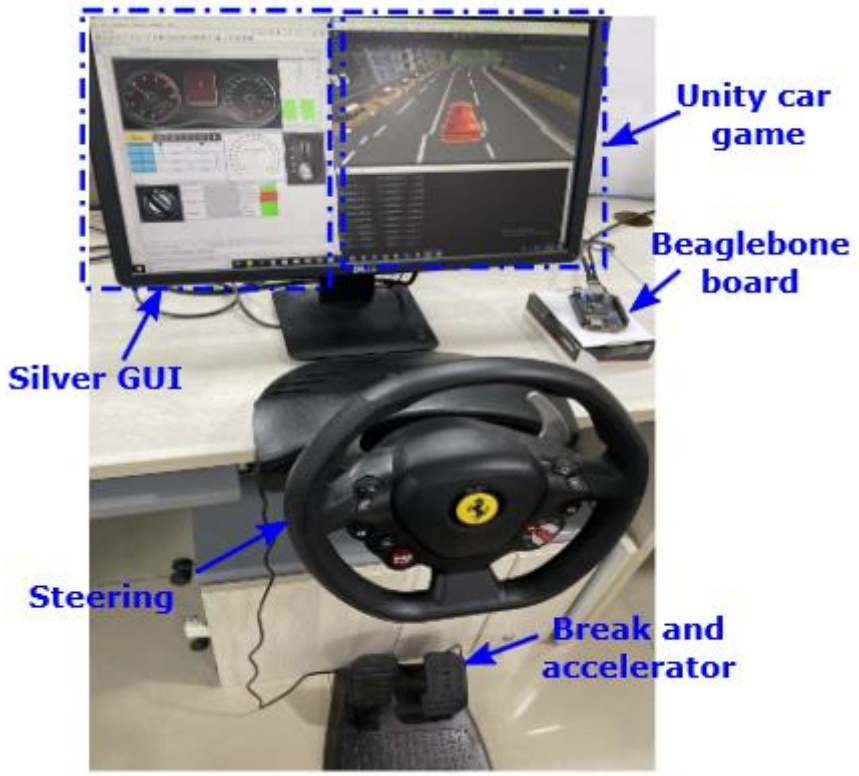


Figure 10: A demonstration of the system. The Synopsys Silver tool generates the CAN bus traces. The Unity simulator is used to drive the car. Note the Beaglebone board that runs the online compression algorithm.

Deliverable: In this case, the deliverable is the code that implements the algorithm.

7. Secure Compression + Encryption Accelerator

Up till now compression and encryption were separate functions, where the standard paradigm is to compress the data first and then encrypt it. This is a slow process and often the size of the compressed text (subsequently encrypted without addition/removal of bytes) gives an idea of the plaintext. There are ways to fix this using zero-padding and adding dummy data.

We propose a new paradigm (preprint paper) based on well-known results in chaos theory. The steps are as follows:

1. We generate a chaotic logistic map to permutate bytes in the plaintext.
2. We use a novel Fisher-Yates merge-shuffle algorithm.
3. This is done in the shadow of a DRAM miss.
4. Next, we perform the compression using the *zstd* library (as we have done in the preprint paper). The RTL code for the zstd algorithm will not be bundled with the software. Instead, we will provide the implementation of another fast compression algorithm based on base-delta compression.
5. Next, we use a Henon and Lorentz map for substitution.

This is an ultra-fast compression+encryption scheme, which is very well suited for IoT applications. Note that it does not provide industrial-grade security, which one can expect from AES-128 or AES-256. The *zstd* version of the algorithm did pass all NIST tests for

a wide variety of datasets. However, with base-delta compression, the results have not been thoroughly evaluated yet.

Notwithstanding this limitation, this is a good piece of hardware for IoT applications that require best-effort encryption and reduction in the volume of network traffic. The latter is easily achieved through compression. Hence, in cases, where best-effort encryption is required and hackers have limited motivation and access to computational hardware, such a scheme should prove to be beneficial.