Driving Data Collection Framework Using Low Cost Hardware

Johnny Jacob^(✉) and Pankaj Rabha^(✉)

Intel Corporation, Bengaluru, India
{johnny.jacob,pankaj.rabha}@intel.com
http://www.intel.com

Abstract. Autonomous driving is driven by data. The availability of large and diverse data set from different geographies can help in maturing Autonomous driving technology faster. It is challenging to build a system to collect driving data which is cost intensive especially in emerging economies. Paradoxically these economies have chaotic driving conditions leading to a valuable data set. To address the issue of cost and scale, we have developed a data collection framework. In this paper, we'll discuss our motive for the framework, performance bottlenecks, a two stage pipeline design and insights on how to tune the system to get maximum throughput.

Keywords: Autonomous driving · Data collection · ROS · Sensing Perception · Dataset

1 Introduction

Our motivation for a data collection framework is to enable a community based effort to collect driving data in India. Challenges in this ecosystem are high cost and steep learning curve of technical know-how. A low cost off-the-shelf solution used as-is falls short to meet reliability, quality, performance and real-time requirements. There are several proposed systems (Table 1) for real time data collection. But, as we can see (Table 4) cost of those systems are quite prohibitive for a developing economy. To address this challenge, we have created a recipe for a reference hardware and the associated software framework which is scalable in performance and minimizes initial capital investment. Also, the stack is designed to achieve maximum throughput possible in a commercial automotive grade system with real time constraints.

1.1 Related and Prior Work

Table 1 provides a list of related data collection frameworks. Most of the frameworks use monocular cameras [4,9,10]. And the ones that use stereo support a maximum of 2 instances [5]. Our goal is to capture surround stereo camera

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11133, pp. 617–625, 2019.
https://doi.org/10.1007/978-3-030-11021-5_38

data to enable stereo based algorithm development which needs minimum of 4 stereo cameras. Available published frameworks describe use of high end servers for computation. Ford Campus Vision [9] uses four 2U servers with quad-core processors; LISA-A [10] uses two servers with 4 Xeon processor with a total of 32 threads each. Our work, DDCF[1] stands apart in the usage of low cost compute without compromising the state-of-the-art benchmark in-terms of sampling rate and data resolution. There is no existing robust data collection system using compute which costs less than 1000$. This is the prime motivation for this work.

Table 1. Comparison of related data collection frameworks [from LISA-A[10]]

	Ford campus [9]	TME [1]	LISA-audi	Cityscapes [2]	LISA surround [4]	LISA A [10]
Year	2011	2012	2012	2016	2016	2017
Camera resolution	800 × 600	1024 × 768	1024 × 522	2048 × 1024	2704 × 1440	1600 × 1200
Camera FPS	8 Hz	20 Hz	25 Hz	17 Hz	12 Hz	30 Hz
Total cameras	6	2	1	2	4	8
Stereo rig	n	y	y	y	n	n
Panaromic camera	y	n	n	n	y	y
LiDAR	y	y	y	n	n	y
Radar	n	n	n	y	n	y
GPS/IMU	y	n	y	y	n	y
Vehicle parameters	n	n	y	y	n	y

The rest of the paper is organized as follows: In Sect. 2, we discuss the challenges faced in designing a system for our target community. Section 3 discusses about the system design and architecture. Section 4 discusses the system configuration and sensor suite used. Section 5 discusses the shortcomings and scope for improvement.

2 Challenges

Based on our experience, the community was apprehensive of investing a huge capital upfront and were more inclined towards incremental upgrades[2] to their data collection rig. Field engineers using the data collection vehicle faced challenges in configuring, running and maintaining the system. They expect minimal pre-flight checks, consistency, repeatability and reliability. The system also have to support high data bandwidth sensors (cameras etc.) as well as synchronization among them. The basic requirements of such a system are

1. Scalability in-terms of performance, number of sensors and cost.
2. High resolution 1080p cameras at 30 fps
3. Uncompressed sensor data (E.g. YUYV, RGB, Point Cloud)
4. Synchronization of multimodal sensors: GPS, IMU, LIDAR and Camera.

[1] Driving Data Collection Framework.

[2] System can be scaled based on performance and cost requirements as described in Sect. 3.1.

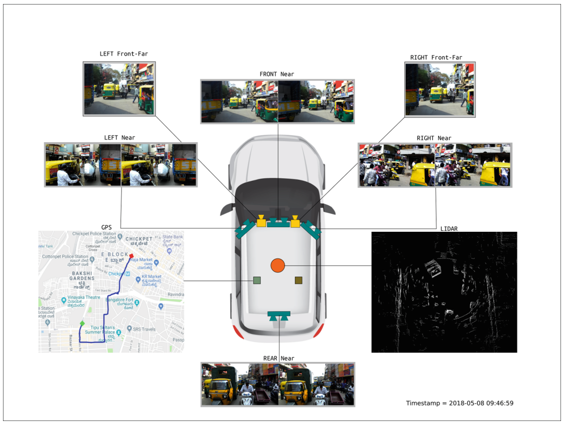

Fig. 1. Sensor layout: an electric car with 4 stereo cameras, 1 LIDAR and 1 GPS/IMU

Apart from the support of multiple sensors, the selection of a framework or middle-ware for such a system is a major challenge. Among many options ROS[3] is the most suitable choice instead of a complete grounds up implementation. But deploying ROS in a low cost platform had performance issues such as using *rosbag record* to write image data to disk involves encoding of the image buffer, multiple in-memory copies and serialization. This degrades performance in terms of frames per second.

3 Driving Data Collection Framework

In this section we describe the proposed framework. We describe about the architecture and the design of the system. We also elaborate on the optimization approaches we took.

3.1 Architecture and Design

Scalability. We wanted scalability in terms of performance and cost as the most important criterion for designing the system. We leverage ROS's distributed architecture to connect multiple low cost hosts across a network hub. If a host H_1 has maxed out its I/O bandwidth and compute with a set of sensors S_1, to add more sensors, a new host H_2 with additional set of sensors S_2 can be connected using an Ethernet hub. New hosts H_n with sensors S_n can be added as needed (Table 2). This enables low cost incremental scalability. Several changes and enhancements are made to the standard ROS framework to meet our requirements. These are described below.

[3] Robot Operating System http://wiki.ros.org/kinetic.

Table 2. Scalable configurations possible with DDCF

	DDCF × 1 host	DDCF × 2 hosts	DDCF × 3 hosts	DDCF × N hosts
Camera resolution	3840 × 1080	3840 × 1080	3840 × 1080	3840 × 1080
Camera FPS	30 Hz	30 Hz	30 Hz	30 Hz
Total cameras	4	8	12	4 * N
Stereo rig	y	y	y	y
Panaromic camera	n	n	n	n
LiDAR	y	y	y	y
Radar	n	n	n	n
GPS/IMU	y	y	y	y
Vehicle parameters	y	y	y	y

Messaging Architecture. The first problem we faced with the setup is the data throughput. ROS has a distributed message passing framework. It allows to run different processes/threads independently for capturing data. But, this approach has a limitation especially in data heavy sensors like cameras. In standard ROS messaging system involves multiple copies and a serialization and de-serialization, which introduces latency. To overcome this issue of ROS we introduced a modified message parsing approach. In this approach we separated the data and the metadata. Only the metadata is published as ROS messages while every image frame is written directly to disk as binary files. The published metadata of every frame is recorded as ROS bags. This approach is depicted in the Figs. 2 and 3.

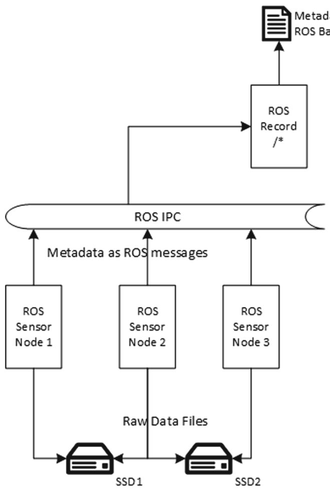

Fig. 2. Stage 1: capture - data is written to SSD as raw binary files. Metadata is published as ROS messages.

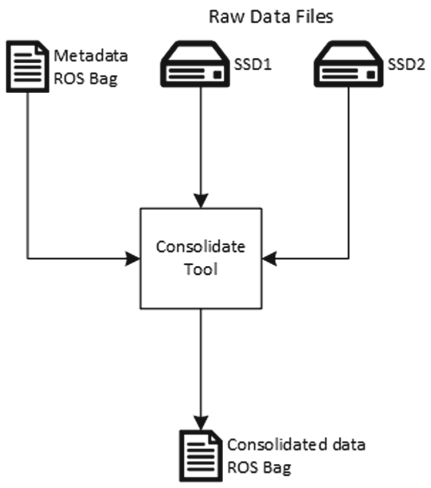

Fig. 3. Stage 2: consolidation - raw binary files are converted to coherent stream using metatdata and published as a single ROS bag.

Two Stage Pipeline. A data collection system demands uninterrupted data capture without information loss. A standard single stage pipeline design consisting of *capture* and *record* did not meet the real time performance we needed from the system. Hence the pipeline is broken into two stages. First stage is capture, all raw sensor data is written to disk as binary files with appropriate metadata recorded as ROS bags.

Second stage is consolidation, which is run offline to build a coherent data stream i.e., combine the raw data from binary files and metadata from ROS bags and output a single ROS bag as a final output.

Filesystem. Filesystems plays a major role in data throughput especially in real time systems. In our scenario, we wanted to use a standard file system which can meet our throughput requirements. We experimented with different file systems that are available in Linux. With emphasis on ease of use, we consciously avoided SSD specific file systems even though they have higher throughput. As shown in Table 3, Btrfs[4] has the highest write throughput for an application such as this.

Latency Optimization. The data capture pipeline was optimized by using a zero in-memory copy approach for persisting raw data which is very similar to the approach adopted in several other ROS based implementations for autonomous driving e.g., Apollo Baidu[5]. We were able to get near real-time performance without going for a strict real time OS or bare-metal embedded system.

[4] https://btrfs.wiki.kernel.org/.
[5] http://apollo.auto/.

Table 3. Comparison of file systems performance

Filesystem	Image resolution (pixels)	MB per frame	Average write time per frame
Ext 4	3840 × 1080	12.4 MB	310 ms
XFS	3840 × 1080	12.4 MB	200 ms
Btrfs	3840 × 1080	12.4 MB	167 ms
Btrfs+LZO	3840 × 1080	12.4 MB	290 ms

3.2 Sensor Calibration

Intrinsics and Stereo Calibration. Intrinsics of cameras is calibrated using Zhang's [11] checkerboard pattern approach. And stereo calibration is performed using OpenCV tools.

Extrinsics for Non-overlapping Field of View. For extrinsic calibration between cameras with non-overlapping field of view, we use a modified version of Pagel [8] using AprilTag [7]. Using AprilTag array instead of checker board pattern improved repeatability. Since the tag array can be uniquely identified, calibration of fixed targets needs to be done only once.

Extrinsics of Camera and a LIDAR. For extrinsic calibration of camera and a LIDAR, we used intermediate results of Dhall et al. [3]. It was challenging to calibrate a 16 line LIDAR with a stereo camera because of sparse point cloud. Multiple iterations were performed to reduce error.

4 System Configuration

The compute hardware is a low cost setup such as an Intel NUC. Since the application demands a high disk write throughput, we recommend using SSD with write speed of 520 MB/s for storage. RAID[6] is desirable but not mandatory as it increases cost. If a compute host has multiple disks, the disk I/O is balanced after experimentation by assigning specific disks to sensor nodes. Optimal I/O loading is capped at 80% bandwidth of disk's write capability.
Our hardware is composed of

- Intel Core i5 Processor
- 4 × USB 3.0
- Ethernet
- 1 × 8 GB RAM
- 2 × 1 TB SSD

This hardware specification costs about 900 USD (Fig. 4). The compute is powered by the electric car's battery. Using a low powered compute, the electric

[6] Redundant Array of Independent Disks.

car's range was extended by 50%[7]. Table 4 shows comparison with similar data collection frameworks and estimated cost[8].

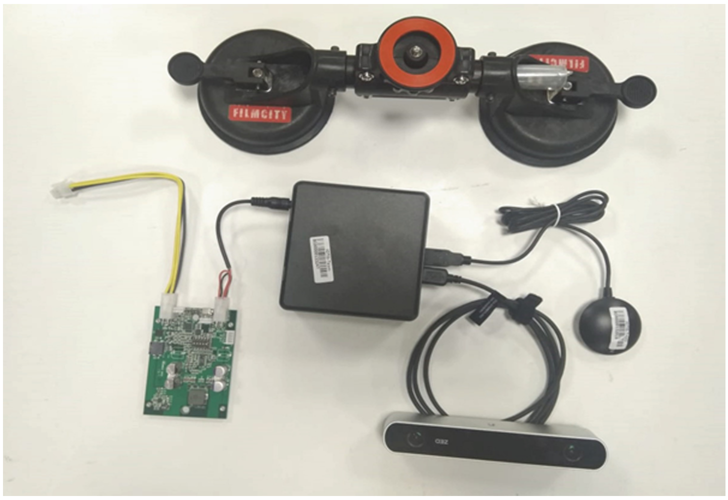

Fig. 4. An example of low cost data collection hardware kit built using the framework. From top left to right: a suction mount, DC voltage regulator, low cost compute, GPS and stereo camera

Table 4. Cost comparison of related data collection frameworks

Framework	Estimated compute cost
LISA-A [10]	5800$
DDCF × 1 host [6]	900$
DDCF × 2 hosts[a] [6]	1800$
DDCF × 3 hosts[a] [6]	2100$

[a]See footnote 2

Figure 1 shows our sensor layout on a electric car. Our test vehicle (Fig. 5) has the following sensors:

- 4 × Zed Stereo Cameras[9]
- 1 × VLP 16 LIDAR
- 1 × Advanced Navigation Spatial GPS and IMU

Software stack has been chosen with readily available components:

[7] In comparison to our initial hardware which was a dual socket Xeon 2U rugged server.
[8] Cost estimation is based on the description of compute.
[9] GiGE cameras can be used for synchronized surround vision data.

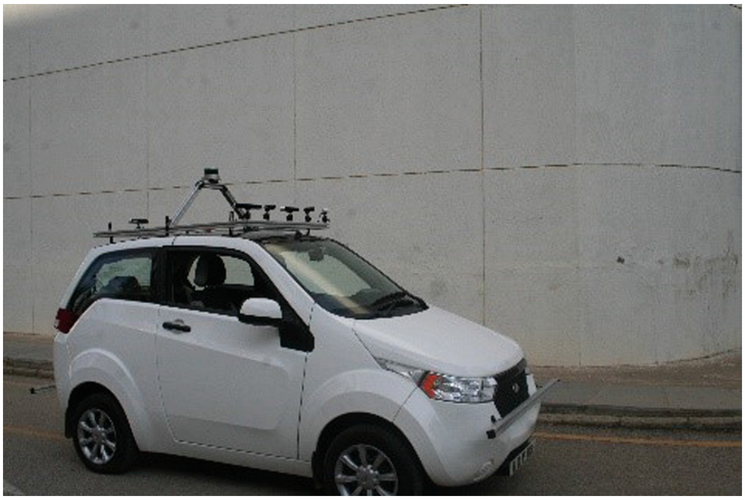

Fig. 5. An electric car mounted with stereo cameras, LIDAR, GPS and IMU

- Ubuntu 16.04 LTS (in run level 3)
- ROS Kinetic

- Sensor nodes from open source community.
- New ROS messages for managing data and meta-data in disk [6]
- Tools for consolidation of data from different streams [6].

5 Conclusion and Future Work

Our framework is designed for use in capturing of data for any driving scenarios and is being improved continuously as an opensource project [6].

Using USB cameras, synchronization between cameras was not possible. We plan to add support for more sensors like PCIe based cameras etc.

Processing of raw data from capture has to be done offline. A live second stage consolidation node to perform lazy consolidation of data during capture will extend the capture time and optimize the use of available storage.

We are working towards our goal of creating an approachable recipe in terms of time, effort and cost for anybody to create a data collection rig using low cost hardware. We believe this will enable a wider participation of community in creation of datasets for autonomous driving research.

References

1. Caraffi, C., Vojíř, T., Trefný, J., Šochman, J., Matas, J.: A system for real-time detection and tracking of vehicles from a single car-mounted camera. In: 2012 15th International IEEE Conference on Intelligent Transportation Systems, pp. 975–982, September 2012. https://doi.org/10.1109/ITSC.2012.6338748
2. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. CoRR abs/1604.01685 (2016). http://arxiv.org/abs/1604.01685
3. Dhall, A., Chelani, K., Radhakrishnan, V., Krishna, K.M.: LiDAR-camera calibration using 3D-3D point correspondences. ArXiv e-prints, May 2017
4. Dueholm, J.V., Kristoffersen, M.S., Satzoda, R.K., Ohn-Bar, E., Moeslund, T.B., Trivedi, M.M.: Multi-perspective vehicle detection and tracking: challenges, dataset, and metrics. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 959–964, November 2016. https://doi.org/10.1109/ITSC.2016.7795671
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361, June 2012. https://doi.org/10.1109/CVPR.2012.6248074
6. Kumar, A., Kambaluru, S., Vanguri, V.R.P., Jacob, J., Rabha, P.: Driving data collection reference kit opensource repository (2018). https://github.com/intel/driving-data-collection-reference-kit. Accessed 17 July 2018
7. Olson, E.: AprilTag: a robust and flexible multi-purpose fiducial system. Technical report, University of Michigan APRIL Laboratory, May 2010
8. Pagel, F.: Calibration of non-overlapping cameras in vehicles. In: 2010 IEEE Intelligent Vehicles Symposium, pp. 1178–1183, June 2010. https://doi.org/10.1109/IVS.2010.5547991
9. Pandey, G., McBride, J.R., Eustice, R.M.: Ford campus vision and lidar data set. Int. J. Robot. Res. **30**(13), 1543–1552 (2011). https://doi.org/10.1177/0278364911400640
10. Rangesh, A., Yuen, K., Satzoda, R.K., Rajaram, R.N., Gunaratne, P., Trivedi, M.M.: A multimodal, full-surround vehicular testbed for naturalistic studies and benchmarking: design, calibration and deployment. CoRR abs/1709.07502 (2017). http://arxiv.org/abs/1709.07502
11. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000). https://doi.org/10.1109/34.888718