A high-rate, heterogeneous data set from the DARPA Urban Challenge

Albert S. Huang Matthew Antone[†] Edwin Olson[⋆] Luke Fletcher David Moore Seth Teller John Leonard

MIT CSAIL
Cambridge, MA USA
{ashuang,lukesf,dcm,teller,jleonard}@mit.edu

[†]BAE Systems
Burlington, MA USA
matthew.antone@baesystems.com

[⋆]University of Michigan
Ann Arbor, MI USA
ebolson@umich.edu

***Abstract*— This paper describes a data set collected by MIT's autonomous vehicle Talos during the 2007 DARPA Urban Challenge. Data from a high precision navigation system, 5 cameras, 12 SICK planar laser range scanners, and a Velodyne high density laser range scanner were synchronized and logged to disk for 90km of travel. In addition to documenting a number of large loop closures useful for developing mapping and localization algorithms, this data set also records the first robotic traffic jam and two autonomous vehicle collisions.**

It is our hope that this data set will be useful to the autonomous vehicle community, especially those developing robotic perception capabilities.

I. Introduction

In the 2007 DARPA Urban Challenge, 11 autonomous land vehicles attempted to navigate a 90 km course in a mock urban environment. The race course itself varied widely and included high curvature roads, stretches of highway, and a steep and narrow dirt road. Autonomous vehicles were required to obey traffic rules while avoiding each other and dozens of other human-driven vehicles.

In this paper, we describe the sensing and data collection capabilities of the MIT entry to the Urban Challenge, and present a data set collected by the vehicle during the race. We also discuss its utility towards the development of future algorithms and capabilities. The actual algorithms used are described elsewhere [Leonard et al., 2008].

This data set spans 7 hours of autonomous operation, from early morning through late afternoon. The environment through which the vehicle traveled is recorded in great detail by a number of onboard cameras and laser range finders. The data set additionally records hundreds of encounters with both human- and robot-driven vehicles. Most encounters were at road intersections or in the middle of a road approaching vehicles traveling in the same or opposite direction. Two encounters resulted in collisions with other robot-driven vehicles. There are very few pedestrians documented in this data set, as the course was not deemed safe for people traveling on foot.

It is our hope that this data set will be useful for the development of robotic perception algorithms, and for the development of autonomous vehicles in general. It provides a unique view of the Urban Challenge as seen by one of the robotic contestants, and has been immensely useful to the improvement of our own algorithms and system design.

Fig. 1. The MIT Urban Challenge vehicle used to capture the presented data set. The vehicle is equipped with a high-precision GPS/INS, calibrated cameras, SICK laser range scanners, and a Velodyne 3D scanner.

II. Data Capture

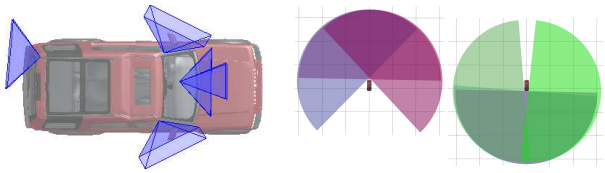
Our vehicle is a modified Land Rover LR3, chosen for its payload capacity, small turning radius, and all-terrain driving capabilities. Custom front and roof fixtures were fitted, permitting sensor positions to be tuned during system development. Wherever possible the fixtures were engineered to protect the sensors from collisions. The stock vehicle was integrated with the following additional sensors:

- Applanix POS-LV 220 GPS/INS
- 12 SICK LMS 291-S05 LIDARs
- Velodyne HDL-64e LIDAR
- 5 Point Grey Firefly MV Firewire cameras

An in-vehicle Quanta blade server system provided computational power, and consisted of 10 quad-core computers connected by gigabit ethernet.

A. Sensor Description

The Applanix POS-LV 220 [Applanix, 2010] was used for world-relative position and orientation estimation of the vehicle. It combines differential GPS, a 1-degree of drift per hour rated IMU, and a 1024-count wheel encoder to estimate the vehicle's position, orientation, velocity, and rotation rates. Only the fused result of the Applanix pose estimates are provided in this data set; the raw Applanix data is not available.



(a) (b)

Fig. 2. Sensor fields of view. (a) The FOV for each of the five cameras is shown in blue. The two forward facing cameras differ primarily in focal length, and the side cameras are pitched down. (b) Our vehicle used seven horizontally-mounted 180° planar LIDARs with overlapping fields of view. The 3 LIDARs at the front and the 4 LIDARs at the back are drawn separately so that the overlap can be more easily seen. An additional 5 roof-mounted pushbroom LIDARs are not shown here.

The SICK LIDAR [Sick, 2010] units generate 180 point scans at 1 degree spacing, with each scan offset 0.25 degrees from the previous scan. The timing of each 180-point scan is individually synchronized using a method robust to jitter in our Linux cluster. The SICK LIDARs are employed in two configurations: "skirt" units roughly parallel to the ground plane at bumper-height, and "pushbroom" units mounted on the roof with a downward pitch.

The Velodyne HDL64E [Velodyne, 2007] is a 3-D laser scanner composed of 64 lasers mounted on a spinning head. It produces approximately one million range samples per second, performing a full 360-deg sweep at 15 Hz. Raw data packets from the Velodyne are captured and logged.

The Point Grey Firefly MV cameras provide color imagery of the vehicle's surrounding area, and image timestamps were acquired by exploiting the independent 8 kHz Firewire clock. All five cameras are roof mounted, and are positioned and oriented as depicted in Fig. 2(a). One camera is rear-facing, two face forward with different focal lengths, and two cameras provide side views. All cameras are spatiotemporally decimated to 376x240 and 10 Hz in the log file. The wide-angle forward-facing camera was logged at a higher resolution of 752x480 and framerate of 22.8 Hz in a separate file.

Sensors are distributed across computers to minimize latency and maximize fault-tolerance. As sensor data is received by the host computers, it is timestamped using the host computer clock along with clocking information available from the device. This fusion of synchronization information makes the timestamps robust to timing jitter that could otherwise occur on the non real-time Linux machines.

Each packet of sensor data is then retransmitted over the network using the Lightweight Communications and Marshalling (LCM) protocol [Huang et al., 2009], where it is captured by a logging process, timestamped again, and stored on disk. As a result, events have two timestamps: the time that the event was logged to disk, and the recovered time of the event itself.

B. Inter-host Time Synchronization

The clocks on each of the 10 computers are synchronized with a simple protocol that designates a single server as the "master" clock. Every other computer continually estimates its clock skew relative to the master clock by measuring the round-trip-time (RTT) of small packets periodically transmitted to the master. Non-master clocks are then slewed to correct for the estimated skew. In practice, measured RTT was often less than 100 μs, which we consider to be a conservative upper bound on the clock skew. The Network Time Protocol (NTP) was also considered [Mills, 1992], but the trivial topology of our network allowed for more accurate synchronization with a simpler method.

C. Coordinate Frame

When fusing sensor data and GPS-referenced coordinates into a common reference frame, our system uses a local coordinate frame defined such that the vehicle pose is the integrated result of its successive relative pose estimates [Moore et al., 2009]. This provides a reliable way to build locally valid maps using recent sensor data with minimal noise, and is not subject to sudden jumps and discontinuities in a globally-referenced frame such as GPS.

This data set contains two sets of pose estimates. The first set describes the pose of our vehicle within this local frame. The second set provides GPS estimates and also defines the transformation between GPS coordinates and the local frame.

III. Sensor Calibration

Our acquisition platform encompasses a number of sensors, each statically mounted in a different location with a different orientation, and each with its own internal coordinate system (Fig. 2). To perform fusion and reasoning in a single consistent 3D reference frame, we associate with our data sets a static set of configuration parameters. These parameters, recorded in the `calibration` section of the text file `lr3.cfg`, define the geometric transformations that align each sensor's measurements to a fixed reference. We assign to every sensor a 6-DOF extrinsic pose consisting of 3D position and orientation with respect to the body frame of the vehicle. Each camera also includes parameters describing its lens that allow transformation of pixel coordinates into 3D rays.

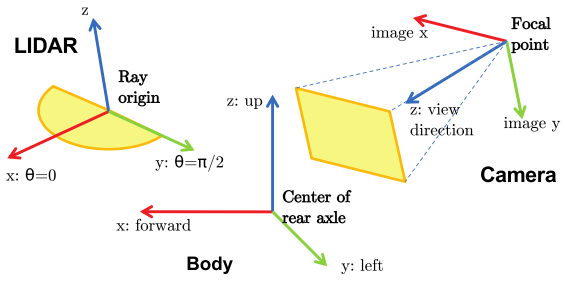


Fig. 3. Coordinate system conventions for the vehicle body frame, and for cameras and laser range sensors relative to the body.

The extrinsic calibration of a sensor is specified as a rigid-body transformation that transforms 3D points in the sensor's internal coordinate frame to the vehicle body frame.

(Fig. 3). The internal coordinate system of a range sensor is defined by the zero-bearing (x axis) and $\pi/2$ bearing (y axis) directions at zero elevation. In camera coordinates, the origin is the lens's 3D focal point, the x axis points from left to right in the image, and the z axis points in the view direction orthogonal to the projection plane. We specify all five parameters of the pinhole projection matrix K, namely

$$\mathrm{K} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

where f_x and f_y represent the horizontal and vertical focal lengths in pixels, s represents the pixel skew, and (c_x, c_y) represents the principal point in pixels. We also employ an analytically invertible radial lens distortion model consisting of the center of distortion and a single control parameter [Tordoff and Murray, 2000].

Since management of these parameters is cumbersome, we devised a number of offline, semi-automated techniques to calibrate the sensors. The 6-DOF poses of the laser range units are estimated jointly using vertical traffic posts arrayed in an arbitrary pattern on level ground. We slowly drive the vehicle amid and around the posts, associating post-like features observed in the laser scans over time and thus forming "tracks" for each post. Relying on highly accurate time synchronization and navigation state (3D vehicle position and orientation), we perform nonlinear stochastic optimization over the unknown sensor poses to minimize the covariance of post tracks when transformed into the local frame (from sensor to body frame via the current parameter estimates, and from body to local frame via the navigation data). The cost minimized is

$$\mathcal{C} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} (p_{ij}(\Psi, A(t_{ij})) - \bar{p}_i)^2$$

where N is the number of posts detected, M_i is the number of observations of post i, p_{ij} is the jth observation of post i in the local frame, and $\bar{p}_i$ is the best known position estimate of post i in the local frame as determined by the sensor. The points p_{ij} are transformed from each sensor's internal reference frame by the current 6-DOF parameter estimates Ψ consisting of sensor roll, pitch, yaw, and 3D position, as well as the (known) rigid body-to-local transformation A at the time of observation t_{ij}.

Camera positions are measured by hand, and lens parameters are determined automatically by applying a variant of Zhang's method [Zhang, 2000]. Camera orientations are determined automatically by driving the vehicle through various turn maneuvers in a highly-textured environment, and comparing epipolar geometry observed in the images [Hartley and Zisserman, 2004] with known vehicle body motion, again relying on accurate navigation state. Each estimated camera rotation, combined with the known vehicle rotation over the corresponding time interval, provides a constraint on the sensor-to-body alignment; the set of all such constraints guides a nonlinear least squares cost minimization algorithm to produce the optimal camera orientation with respect to the vehicle body. The cost minimized is

$$\mathcal{C} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} (l_{ij}(p_{ij}, \Psi, A(t_i)) \cdot q_{ij})^2$$

where N is the number of sequential image pairs used for calibration, M_i is the number of point feature correspondences across image pair i, and l_{ij} is the epipolar line associated with the undistorted image point feature pair consisting of p_{ij} in one image and q_{ij} in the other. Here, l_{ij} is computed from p_{ij} as a function of the current estimates of the camera's roll, pitch, and yaw angles Ψ, its known 3D position, and the known rigid body-to-local transformation A at the time of image acquisition t_{ij}.

Mission	Duration	Distance	LCM Log Size	Camera Log Size
1	10,947 s	30.0 km	74 GB	18 GB
2	5,428 s	21.5 km	38 GB	10 GB
3	10,414 s	41.2 km	72 GB	18 GB

TABLE I
MISSION SUMMARIES

IV. DATA

Each vehicle in the Urban Challenge was required to autonomously complete three mock supply missions. In each mission, the vehicle was required to visit a series of GPS waypoints in a specific order. The amount of time our vehicle spent on each mission, in addition to the distance traveled, is shown in Table I.

This data set consists of two files for each mission. One file contains the high-resolution image data for the wide-angle forward-facing camera, captured at 22.8 Hz. This file is simply a TAR archive of JPEG files; the name of each file corresponds to the time at which the image was acquired, measured in microseconds since 00:00:00 Jan 1, 1970 UTC.

The second file contains the remaining sensor data (including low-resolution image data for all cameras), and is stored in the LCM log file format. This format is useful for storing many heterogeneous streams of synchronized data, and is described in the accompanying source code. Conceptually, an LCM log file is an ordered list of events, where each event corresponds to a timestamped data packet. Each event is also associated with a named channel, e.g. "POSE" for the Applanix-derived pose estimates. Table II describes each channel in this data set.

A. LCM Types

In order to efficiently represent complex data structures in a compact and easily accessible manner, each data packet in a log event is serialized with the LCM marshalling tools. LCM provides a language-agnostic type-specification language that allows users to define custom data types. Automatically generated language bindings then provide a fast and easy way to both represent these data structures in a specific language, and to serialize the data structures to and from a binary blob.

Channel	Description	Rate
POSE	Vehicle pose (local frame)	100 Hz
GPS_TO_LOCAL	Vehicle pose (GPS)	100 Hz
CAM_THUMB_RFC	Camera front wide∠, low-res.	10 Hz
CAM_THUMB_RFC_6mm	Camera front narrow∠. low-res.	10 Hz
CAM_THUMB_RFR	Camera left, low-res.	10 Hz
CAM_THUMB_RFL	Camera right, low-res.	10 Hz
CAM_THUMB_RFR	Camera rear, low-res.	10 Hz
BROOM_L	SICK pushbroom left	75 Hz
BROOM_CL	SICK pushbroom left-center	75 Hz
BROOM_C	SICK pushbroom center	75 Hz
BROOM_CR	SICK pushbroom right-center	75 Hz
BROOM_R	SICK pushbroom right	75 Hz
SKIRT_FL	SICK skirt front-left	75 Hz
SKIRT_FC	SICK skirt front-center	75 Hz
SKIRT_FR	SICK skirt front-right	75 Hz
SKIRT_RC_HI	SICK skirt rear-high	75 Hz
SKIRT_RC_LO	SICK skirt rear-low	75 Hz
VELODYNE	Velodyne	15 Hz

TABLE II

LCM CHANNELS

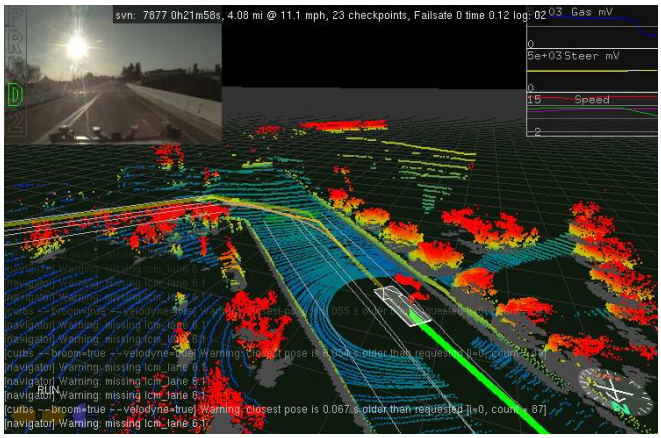


Fig. 5. A screenshot of the Linux visualization software. This application provides an interactive and more intuitive 3D view of data in our log files.

```
struct laser_t
{
  int64_t  utime;

  // range data (meters)
  int32_t  nranges;
  float    ranges[nranges];

  // intensity data, in sensor-specific units
  int32_t  nintensities;
  float    intensities[nintensities];

  // the angle (in radians) to the first point
  float    rad0;

  // the number of radians between each
  // successive sample
  float    radstep;
}
```

Fig. 4. The LCM type definition specifying how SICK data in our log files is formatted. Additional type definitions in the accompanying source code specify the structure of other logged data.

Instead of describing the binary layout of each data type, we provide the LCM type definitions, and a brief overview on how the type definition dictates the structure of the binary blob. Fig. 4 shows the type definition describing how SICK data is represented in our log files. Additional type definitions in the accompanying source code describe the format of the pose estimates, camera data, and Velodyne data.

Each encoded packet begins with an 8-byte type signature, which is used by LCM for runtime type checking and can be safely ignored in this context. Next, each field is packed in the order it appears in the type definition. The size of an integer field is indicated by its field name (`int8_t, int16_t, int32_t, int64_t`), and multi-byte integers are packed in network byte order. Single- and double-precision floating point values (`float` and `double`, respectively) are encoded using the IEEE 32- and 64-bit formats, also appearing in network byte order. The `byte` field can be treated as identical to the `int8_t` field.

Some fields appear as fixed-length or variable-length arrays. Fixed-length arrays are declared in the type definition using a C-style syntax, and decoding one simply involves decoding a fixed number of individual elements. The length of a variable-length array is determined by another field in the type, named in the type definition. Once the length of a variable-length array has been determined by decoding the value of the length-specifying field, the array is decoded in the same fashion as a fixed-length array.

B. Source Code and Data

The data files and accompanying source code can be downloaded from:

`http://dgc.mit.edu/wiki/index.php/PublicData`

This data set includes a modified version of the automatically generated C language bindings for parsing the binary blobs in each log event, and no external software libraries are required to decode the log files. Additionally, C source code is provided that is able to iterate through the events in a log file and also project sensor data into a common coordinate frame (the local frame [Moore et al., 2009]). Short example programs are provided that illustrate how to accomplish each of these tasks.

In addition to these portable examples, we also distribute a Linux-specific tool for graphical visualization of logged data. The visualization tool projects all sensor data into the local frame and provides an interactive viewing environment. It served as one of the primary development tools during the design and implementation of our vehicle, and we highly recommend its use when access to a Linux development platform is available. Fig. 5 shows a screen capture.

V. SUMMARY

We have presented a data set that documents in great detail the DARPA Urban Challenge from the perspective of the MIT vehicle. Data from calibrated cameras, laser range scanners, and navigational sensors provide a 7-hour data set of autonomous vehicle operation that spans 90 km of travel through a variety of road conditions. This data set is being

provided to the community to aid progress in autonomous vehicle research, for example through comparative evaluation of different algorithms implemented with the data.

Acknowledgments

Sponsored by Defense Advanced Research Projects Agency, Program: Urban Challenge, ARPA Order No. W369/00, Program Code: DIRO. Issued by DARPA/CMO under Contract No. HR0011-06-C-0149. The authors wish to thank all the members of Team MIT for their contributions to this project. The authors also gratefully acknowledge the sponsorship of: MIT School of Engineering, MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT Department of Aeronautics and Astronautics, MIT Department of Electrical Engineering and Computer Science, MIT Department of Mechanical Engineering, The C. S. Draper Laboratory, Franklin W. Olin College of Engineering, The Ford-MIT Alliance, Land Rover, Quanta Computer Inc., BAE Systems, MIT Lincoln Laboratory, MIT Information Services and Technology, South Shore Tri-Town Development Corporation and Australia National University. Additional support has been provided in the form of in-kind donations and substantial discounts on equipment purchases from a variety of companies, including Nokia, Mobileye, Delphi, Applanix, Drew Technologies, and Advanced Circuits.

REFERENCES

[Applanix, 2010] Applanix (2010). Applanix POS-LV 220: Land vehicle position and orientation system. http://applanix.com/products/land/pos-lv.html.

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

[Huang et al., 2009] Huang, A. S., Olson, E., and Moore, D. (2009). Lightweight communications and marshalling for low latency interprocess communication. Technical Report MIT-CSAIL-TR-2009-041.

[Leonard et al., 2008] Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., and Williams, J. (2008). A perception-driven autonomous vehicle. *Journal of Field Robotics*, 25(10):727–774.

[Mills, 1992] Mills, D. (1992). Network time protocol (version 3) specification. RFC 1305, Internet Engineering Task Force.

[Moore et al., 2009] Moore, D., Huang, A. S., Walter, M., Olson, E., Fletcher, L., Leonard, J., and Teller, S. (2009). Simultaneous local and global state estimation for robotic navigation. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3794–3799, Kobe, Japan.

[Sick, 2010] Sick (2010). Laser measurement technology LMS2xx / LMS291 / outdoor / mid-range sensor. https://mysick.com/PDF/Create.aspx?ProductID=33771.

[Tordoff and Murray, 2000] Tordoff, B. and Murray, D. (2000). Violating rotating camera geometry: The effect of radial distortion on self-calibration. In *Proc 15th Int Conf on Pattern Recognition, Barcelona, Spain*, volume 1, pages 423–427.

[Velodyne, 2007] Velodyne (2007). Velodyne's HDL-64E: A high definition lidar sensor for 3-d applications. http://www.velodyne.com/lidar/downloads/whitepaper.aspx.

[Zhang, 2000] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334.