HD Map Generation for Autonomous Racing

Matthew Sivaprakasam (msivapra@andrew.cmu.edu)
Andrew Saba (asaba@andrew.cmu.edu)
Robotics Institute, Carnegie Mellon University

***Abstract*— Autonomous motorsports aim to replicate the human racecar driver with software and sensors. As in traditional motorsports, Autonomous Racing Vehicles (ARVs) are pushed to their handling limits in multi-agent scenarios at extremely high ($\geq$ $150mph$) speeds. This Operational Design Domain (ODD) presents unique challenges across the autonomy stack. Standard autonomous vehicles (AVs) rely on High Definition (HD) Maps, that provide information such as lanes, right of way information, and more, to simplify many problems. For ARVs, an HD map includes information about the race track, including its boundaries and banking, which the autonomy software can exploit to improve performance. In this project, we created a pipeline to take in noisy sensor LiDAR data and GPS information in order to create an HD map for an ARV, automatically building a globally registered pointcloud and extracting the relevant information.**

I. Introduction

Autonomous vehicles (AVs) are fully automated passenger and commercial vehicles that are able to navigate in the open world from place to place without a driver. AVs have a lot of promise in making everyday travel safer and more accessible. However, a lot of development is still required before they become ubiquitous and can be trusted to replace human drivers. Autonomous racing vehicles (ARVs) provide a platform to test algorithms and sensors in extreme conditions (i.e. very high speeds and accelerations), with the hope that if software and hardware can handle such extreme cases, they can also handle driving down the street in a metropolitan area at 35mph with a very low probability of failure.

High definition (HD) maps are ubiquitous in autonomous vehicles [1,2]. For ARVs, HD maps of the tracks they are racing is also equally important, as these maps provide critical information about the track geometry, where walls and run offs are, and available lanes. As with conventional AVs, it is important to generate these HD maps in an automated fashion, as it enables a faster deployment of AVs on the road and allows the maps to evolve automatically as the world changes.

Fig. 1: Las Vegas Motor Speedway. The extreme banking allows vehicles to travel faster than they normally would. Knowing where the track boundaries are with high accuracy is crucial for calculating optimal racelines.

Like an AV, an ARV needs to know where it can and cannot drive. Additionally, unlike an AV, an ARV also needs to know how banked the track is. Banking refers to the slope of the track surface, either laterally or longitudinally. A high, positive (i.e. in the direction of driving) banking allows a vehicle to drive faster than a flat road. This information is important for a planner or controller to know in order to optimize vehicle speed and performance.

In this project, we aimed to address some key

challenges in HD map generation, but tailored for an ARV. Specifically, address the following challenges:

1) Automatically build a globally registered map of a track from local LiDAR scans and noisy GPS information
2) Automatically extract where the walls and track boundaries are in this map
3) Classify the track and surrounding ground surfaces, in order to build a map of drivable area
4) Build a "banking" map of the track that can be utilized downstream to optimize vehicle performance

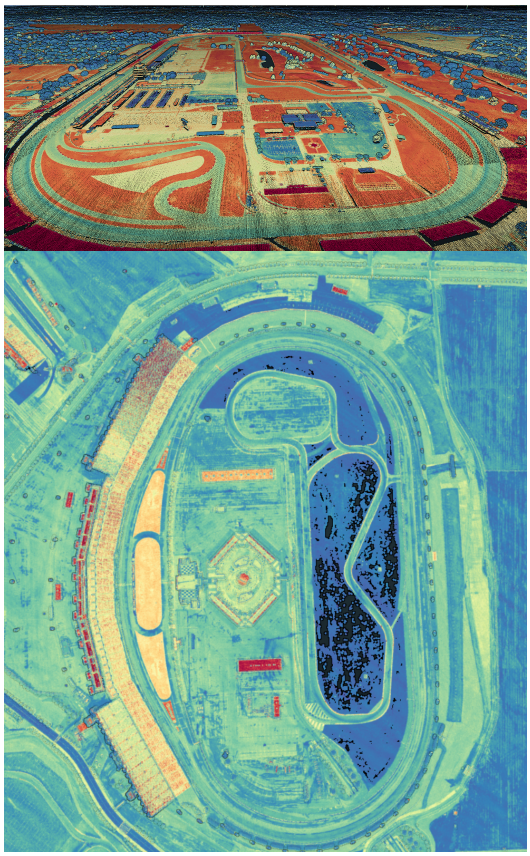

Fig. 2: Pointclouds of the Indianapolis Motor Speedway and Vegas Motor Speedway generated and publicly released by USGS. Unfortunately, this type of data is not available at tracks outside of the US, such as in Italy.

II. Related Work

For autonomous vehicles (AVs), automatic HD map generation is a very open area of research [1] [2]. In many of these approaches, sensor data is processed to detect critical components of the maps, including crosswalks, signs, lanes, etc. Many of these features are not present on race tracks, which simplifies the problem setup. However, unlike an AV, there is very little training data for race tracks, necessitating either an unsupervised approach or an approach that is very data efficient.

For point cloud registration, a lot of previous work has been done, a lot of which are variants of iterative closest point (ICP). For example, the authors from [3] develop an ICP pipeline that is more robust than previous works. However, even this pipeline has failed to register data from an ARV, showing that ICP optimizations can be brittle and sensitive to the conditions under which the data set was collected. To address these shortcomings, we intend to use noisy GPS data as an initial guess for how to register the point clouds and use ICP to refine the alignment. Our hope is that the initial guess will make the optimization less brittle.

III. The Dataset

A. The Platform

The target platform for the HD Maps created from this project is the Dallara AV-21, the official vehicle of the Indy Autonomous Challenge (IAC). Every competitor must use the same hardware, including vehicle setup, autonomy sensors, and compute. The vehicle is a modified version of the Indy Lights IL-15 chassis, retrofitted with a package of automated vehicle sensors, drive by wire, and compute. The engine is a 4 Piston Racing-built Honda K20C. Sensors onboard the AV-21 include 3 Luminar Hydra LiDARs, 3 Aptiv Medium Range Radars, 2 NovAtel PwrPak7D-E1 GNSS, and 6 Mako G-319 Cameras. In total, these sensors provide redundant 360° coverage and over $200m$ of sensing range. Figure 3 shows the AV-21 platform and sensor locations.

B. Data Collection

The target track for the HD Map generation is the Autodromo Nazionale Monza, in Monza, Italy. Because the AV21 platform is not able to be driven manually, a surogate vehicle was equipped with LiDARs, cameras, and GPS to simulate the full

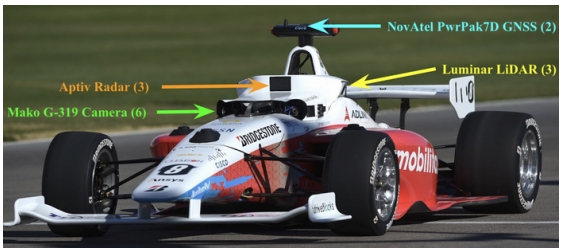

Fig. 3: ARV with available sensors.

suite of the AV21 and was driven on Monza. This data was shared to us, providing a dataset to build the necessary HD map features for this project.

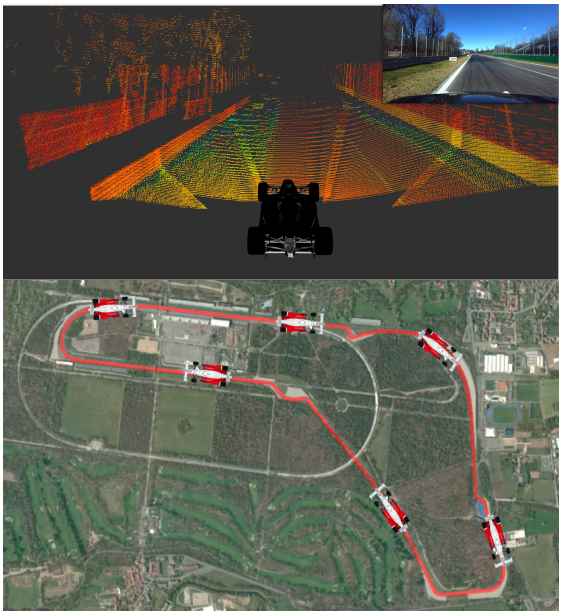

Fig. 4: The GPS trajectory for a lap around the Monza Circuit, and an example local lidar scan at a specific timestep.

IV. Method and Results

A. Accurate Global Pointcloud Accumulation

While the GPS signal is relatively accurate, it's not as accurate as needed in order to calculate the information we desire, which is all sensitive to noise (an example of this noise is shown in Fig. 6). To fix this issue we use ICP [4,5] on each scan to fit it to the previous scans (algorithm outlined in Alg. 1), using the GPS data simply as an initialization for the ICP process. At a high level, we take a local pointcloud and its corresponding GPS position, use ICP to fit it to the previous scans, and then merge it into the previous scans. To save computation, we downsample the pointcloud to a .1 meter discretization, and only feed a buffer of the past 10 scans (instead of the whole global pointcloud) into the ICP process. When we run loop this over a sequence of the ARV driving around the track, we get the result shown in 5. The resulting pointcloud is of much higher quality than what we would have obtained using GPS alone.

Algorithm 1: Iterative Closest Point (ICP)

Input: Local Pointcloud P_l,
Reference Pointcloud P_r,
Number of iterations N,
Initial guess from local frame to reference frame T_r^l
Output: Transform from local frame to reference frame T_{lr}^*
for $i \leftarrow 1$ ***to*** N **do**
  TempP$_l \leftarrow transform($P$_l, T_{lr})$
  Correspondences $\leftarrow closest(Temp$P$_l,$ P$_r)$
  TempP$_l \leftarrow Temp$P$_l$[correspondences]
  TempP$_r \leftarrow$P$_r$[correspondences]
  $T_r^l \leftarrow calcTransform(Temp$P$_l$,TempP$_r$)
end
return T_r^l

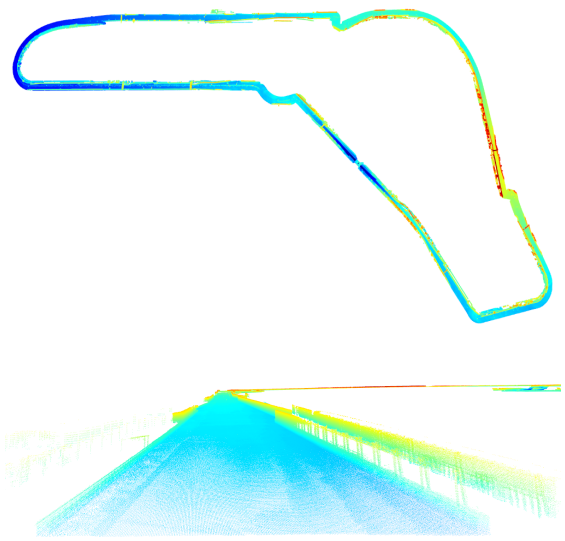

Fig. 5: A visualization of the global pointcloud.

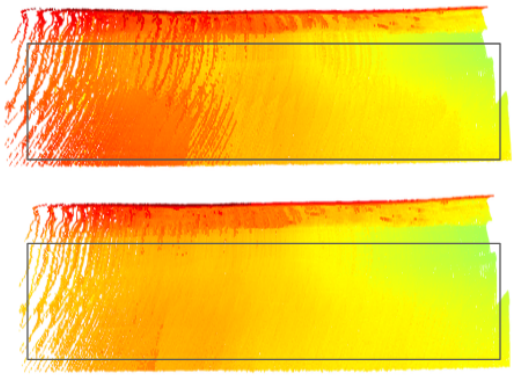

Fig. 6: A portion of the accumulated pointcloud, colored by height. The top shows the pointcloud accumulated using GPS alone. The bottom shows the results using ICP, which provided a smoother estimate of the track (highlighted in the grey boxes).

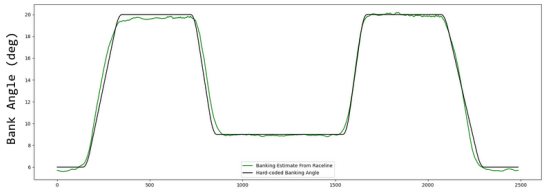

Fig. 7: A comparison between our banking estimate and a hard-coded estimate based on the known min/max banking angles at the extremes of the track.

B. Bank Angle Calculation

In order to reach higher speeds, it is important for the ARV to know the banking angle of the road at every point on the track. We use the high-definition pointcloud to calculate a bank angle for each point in the pointcloud.

For each point, we find the N nearest points to it. Doing some covariance analysis using the Open3D library [6] allows us to find the principal axis among these points and the normal vector of the point can then be inferred. The normal vector in this case describes an outward direction perpendicular to the surface of the track. If we orient the whole global pointcloud so that it is parallel to an XY plane, the bank angle at a given point is simply the arc-cosine of the z component of its normal vector. The result of this approach on a turn in the Monza track is shown in Fig. 8.

In order to validate this approach independent of our ICP processing described previously, we also tested it on some US tracks using pointclouds from USGS data. Fig. 9 shows the bank angle on every point of the Lucas Oil Speedway. The lowest bank angles are clearly in the straightaway, with steeper banking on the outer edges of the turns. Using this approach we can observe not only longitudinal changes in bank angle but also lateral changes, which is important on tracks with progressive banking. Fig. 9 shows a qualitative evaluation. In order to evaluate the actual banking values, we compare our estimates with another team's bank estimate at the Las Vegas Motor Speedway as shown in Fig. 7. Their estimate was calculated by interpolating between the publicly available extremes of the track. We found that our approach serves as an accurate enough automatic pipeline for when public banking data isn't available or when manual calculation requires too much effort.

Fig. 8: A visualization of how the bank angle changes during a turn on the track.

C. Track Detection

The track consists of four main components:

1) Track, paved, able to be driven on
2) Run-off, paved, able to be driven on
3) Gravel traps, not able to be driven on
4) Grass and ground, not able to be driven on

For this portion of the project, we focused on being able to distinguish between the drivable and non-drivable portions of the track. This comes down to being able to segment pavement and non-pavement from the LiDAR points.

Because pavement is smoother than non-pavement, our initial thought was to perform principle component analysis (PCA) for clusters of points in the cloud. Using the primary components, we generated several hand-tuned features, shown in Figure 11. These features are different ways to

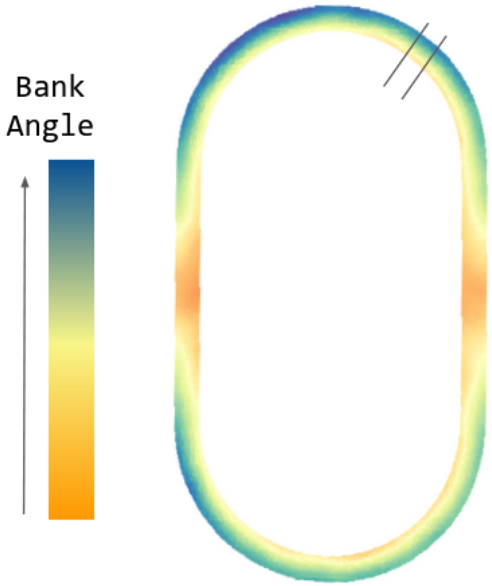

Fig. 9: A bank map we calculated for the Lucas Oil Motor Speedway. Specifically, note that we can not only model how the banking changes as the car moves forward along the track, but also as it moves laterally.

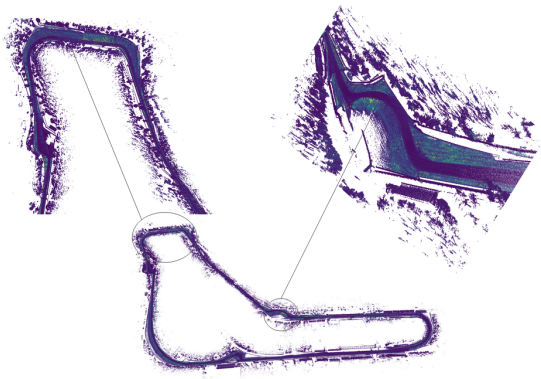

Fig. 10: Intensity serves as a good feature for distinguishing the track from grass.

project the principle eigenvectors of the clusters of points, which describe some physical property (i.e. anisotropy, planarity, sphericity, etc.) of the cluster. For example, the feature in the bottom right of the figure is curvature. This feature space produces very distinct sets of features for the flat areas of the track and the walls, but struggles to pick up on the finer geometric differences between point returns from grass and pavement. In the end, point intensity proved to be more useful as a feature than the hand crafted features.

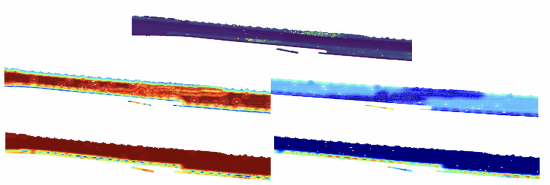

Fig. 11: Several feature maps, extracted from the pointcloud for a portion of the track. (Top-Middle) The original pointcloud, colorized by the intensity of each point. It is possible to visually make out the portions of the track that are paved and not paved. Despite filtering, voxelization, and extensive hand tuning, it was difficult to hand craft a feature set to distinguish paved and unpaved ground, outside of directly using the point intensity values.

To extract the track bounds, the pointcloud, colored by intensity, is projected into a bird's eye view (BEV) image, which amounts to what the pointcloud would look like if you were looking straight down onto it. Once in image space, computer vision tools are at our disposal, including simple color thresholding. By simply removing "pixels" that are not close to the hand-tuned thresholds, we were able to efficiently and effectively segment out the portions of the track we were interested in.

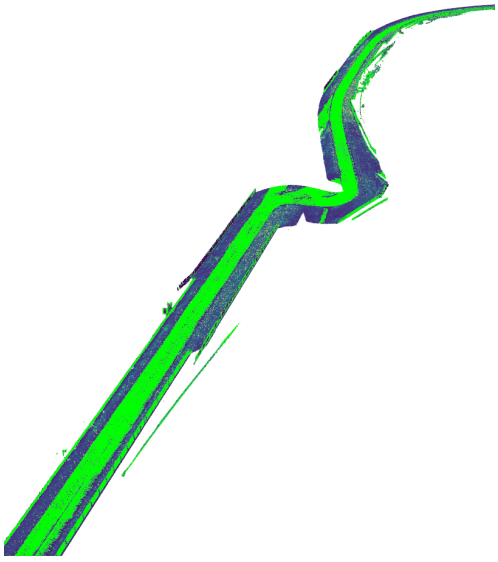

Fig. 12: A section of the global pointcloud, with track points colored in bright green.

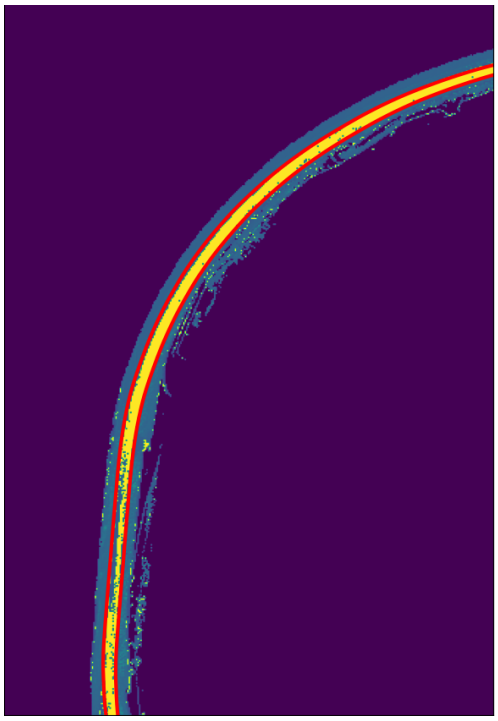

Fig. 13: A section of the global pointcloud projected to birds-eye-view, with detected track points colored in yellow. The actual track bounds are shown in red.

We employ other filtering on the 2D BEV image, including only looking for points in the image above a certain color gradient. Because we are interested in the track boundaries, we care about the points that lie on the edge of pavement and unpaved ground, so there is a distinct color gradient at this edge. Once we have our candidate points, we fit a line and remove outliers.

D. Wall Filtering

To extract the wall location, we could have utlized the feature maps we created, as shown in Figure 11. However, we wanted an even simpler and faster approach, so we developed a method where we "slice" the pointcloud vertically, thereby removing points that are low and are likely the ground and points too high to be a wall. Once we have this slice, we can perform a line fit similar to how the track bounds are extracted and get the wall boundaries.

V. Conclusion

In this work, we presented a pipeline for taking noisy sensor measurements, specifically GPS and LiDAR pointclouds, and generating an HD map of a racetrack for an autonomous racing vehicle (ARV). First, we showed how to generate the global map, by utilizing ICP to refine an initial guess from GPS. Secondly, with this globally registered cloud, we can extract important information about the track banking, which is especially important to maximize the dynamic potential of the ARV. Finally, we showed how we can take the registered cloud and extract features for segmenting out the track walls and boundaries, including the drivable areas.

We believe that there is much further work in pre- and post-processing the pointclouds, to reduce the noise and further improve the accuracy of the registration. Additionally, we believe there is room for using learned features to segment out the relevant portions of the track. While the PCA features did not work as expected, we believe this is more of a function of not having just quite the correct set of features, which a learned model can fix by learning how to extract the pertinent information from the sensor data.

Finally, future work can can extend this approach to become multi-modal, incorporating additional information, such as colorization of the cloud via RGB cameras, or additional features, such as language or other semantics, to encode important information about the track. For example, human racecar drivers build maps of the track that are much richer than pure geometries, including information about the parts of track with the best visibility, parts that get the most sun and as such are warmer so tires grip better, areas that are trickier to navigate, and so much more. Encoding a richer set of knowledge into a form that an autonomy software stack can use is a very open research problem, and doing so in a way that requires little-to-no manual overhead will streamline a number of important tasks.

REFERENCES

[1] L. Mi, H. Zhao, C. Nash, X. Jin, J. Gao, C. Sun, C. Schmid, N. Shavit, Y. Chai, and D. Anguelov, "Hdmapgen: A hierarchical graph generative model of high definition maps," *arXiv*, 2021.

[2] Z. Bao, S. Hossain, H. Lang, and X. Lin, "High-definition map generation technologies for autonomous driving," 2022.

[3] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "Kiss-icp: In defense of point-to-point icp – simple, accurate, and robust registration if done the right way," 2022.

[4] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[5] E. Recherche, E. Automatique, S. Antipolis, and Z. Zhang, "Iterative point matching for registration of free-form curves," *Int. J. Comput. Vision*, vol. 13, 07 1992.

[6] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.