Joint Tracing and Detection Using Point Cloud Data

Robert Lake  
University of Virginia  
Charlottesville, VA 22903  
rjl4sw@virginia.edu

Jyoti Kini  
University of Central Florida  
Orlando, FL 32816  
jyoti.kini@knights.ucf.edu

Dr. Mubarak Shah  
University of Central Florida  
Orlando, FL 32816  
shah@crcv.ucf.edu

Abstract

Most online multi-object trackers perform object detection stand-alone in a neural net without any input from tracking. The proposed model is an online joint detection and tracking model that primarily uses point cloud data. This joint tracking and detection model has been implemented using the PointNet feature extractor to parse 3D PointCloud data. We conducted experiments based on the effectiveness of our Model on the JRDB benchmark suite.

1. Introduction

In the age of self driving cars and other autonomous vehicles, there is a significant need for various methods of object tracking and detection. The proliferation of such technologies throughout the industries leads to many different approaches for tracking and detecting through a model. Traditionally, models have functioned successfully by using RGB image data as the primary source of the information. This then would be fed into a model that would extract it’s features and apply tracking and detection in separate sections of the model. The model that we propose uses the primary architectural features of the TraDeS model (a model known for combining tracking and detection) along with the PointNet architecture to enable the model to interpret point cloud data. Within the model we have applied Cost Volume Association and the Motion Feature Warper to the point cloud data to derive tracking and detection for individual subjects.

While autonomous vehicles are a big part of future advances in the space, the ability to track and detect vehicles cannot be the only thing considered when trying to create a model that is truly robust. Traditionally, many datasets such as KITTI have cars as its central focus with other elements on as side areas. This is a factor that lead to our decision to use the JRDB benchmark. With it’s diversified approach to environments, that feature both indoors and outdoors, it will allow for us to test it’s true versatility.

There will be a more detailed explanation of the project in its entirety throughout the rest of this report. It will go over the architecture that was used along with it’s components while discussing the results that were achieved through the testing that we have evaluated from the JRDB benchmark.

2. Related Work

The model that we propose is one that can be considered as a fork of another existing model that has already had existing precedent for good performance called TRaDeS (TRack to DEtect and Segment)[1]. This is the model that our architecture is the most directly based upon due to it’s ability to jointly track and detect.

The TRaDeS model is also based from another model called Center-Track. The fundamental components of the TRaDeS model is present in this one with the major exception of there being one less layer of data for the model to parse. This is a very consequential difference due to TRaDeS using the extra data to create a weight-shared value associations that allow it to make tracking inferences based on the detentions that are present.

The work that we were able to build upon TRaDeS with would not be possible without the existing PointNet feature extractor. In essence it, allowed for our work of retrofitting an existing model to accept point cloud data to be a process something that could be reasonable achieved.

The Dataset and benchmarking software that was used for our work have been both provided by JRDB (Jack Rabbit Dataset and Benchmark) from Stanford University[4]. The primary benefit of working with JRDB was due to the suite being a Unified platform to gather, train and test a model with as little conversions as possible.

3. Network Architecture

That we have designed is divided into three primary components. The Point Net Feature Extractor, Cost Value Association and the Motion Feature Warper.
Figure 1. The model above is primarily based on the TRaDeS architecture. The model has three primary functions. The PointNet feature extractor is first and allows for the data from three individual frames to be loaded into the model at once. The next phases is the Cost Volume Association which is able to identify key features from the max-pooled frames that were fed into it. The last phase is the Motion Feature Warper which allows for the temporal variances between frames to be identified.

3.1. Cost Volume Association

This process is done by comparing the features extracted between two frames in sequential order. After the features are extracted through the PointNet feature extractor we the features themselves are compared between the two frames. The cost volume is derived from gleaming the dense matching similarities between point cloud data gathered from two frames. This cost volume calculation was primarily done through the down-sampling of the feature embedding by a factor 2 which there intern reduces their image’s total point/pixel count to that of only 25 percent of it’s original size. The points that have been gathered and parsed through the cost volume process are then required as the entry feature map for later parts of our model.

3.2. Tracking Offset

Put simply, the tracking offset is a measure of the temporal displacements caused by what is believed to be a single item or person moving within a frame. Temporal displacements are effectively just a measurements of how light data interacts without a solution. There seems to be a strong need for this process to be the core of our model. To be deemed as an effective means to suit the core of a model, it must be able to track the temporal differences from the vertical and horizontal directions respectively. It is believed that only of then is the current offset valuable in determining outcomes from the road before people can even detect them.

3.3. Motion-guided Feature Warper

The Motion Feature Warper, is the main predictive apparatus that is featured in both the TRaDeS model and our own [1]. The primary function is to take the tracking offsets from the two frames that were fed into itself in order to predict where the future actions were being detected. The second frame of PointCloud data in particular allows the model to compensate for the lack of a surplus of point cloud data.

3.4. Feature Enhancement

A key benefit of the lack of a unified segmentation and detection layer is that the tracking can become inferential. With things such as Video and possibly LiDAR data, it is very easy for objects to become occluded (getting lost or trapped behind the model). The primary focus of the model’s features enhancements was to make the main methods of detections and detractions more robust and directed than ever before. Not only would this method in theory be able to track and detect through otherwise impermeable objects. Overall, through it’s predictive measures that it approaches the outward draw frames and bounding boxes, it is often able to outperform its own calculations.
4. The Dataset

The dataset that we chose to select for this project was primarily being run through the JRDB benchmark suite. The benchmarking suite make itself different approaches to testing by minimizing as many confounding variables as possible. The benchmark features over twenty different locations and many different separate scenarios. The primary mission of the dataset is to create a truly comprehensive neural network that can be around person and then retrofitted outward.

Another reason why JRDB was a good model to choose over KITTI for the primary knowledge base was that JRDB was much more modern. While this doesn’t seem to make a very big difference but the differences in their architecture become more apparent decision-making breaking point. With a single search through the file system, it becomes apparent that the files in KITTI’s datasets are not properly standardized to work well with more modern applications. The most notable of which is that the PointCloud files were written in .bin (a nonstandard format) while in JRDB, they were paired with the universal (.pcd). Due to these concerns, JRDB was the clear better choice in the search for optimization.

5. Training

\[ L_f(G_1, A_1) = \frac{\sum (G_1 \odot (- \log A_1))}{\sum G_1} \]

Figure 4. The forward loss function that was converted to after after the first revision.

Another proposition for improved performance involved a change in the primary feature extractor for LIDAR data. The suspicions that we faced while running the testing is that the buffer size for the point cloud data is too small to allow for the model to truly learn from the best way through a clearer sample. If a resolution decreased by a factor of two effectively has only ¼ of the total file size in terms of scale. This has the potential to severely hinder the feature extractor of this feature was indeed the intended problem. This possibly has not been fully investigated further since it’s release.

6. Results

<table>
<thead>
<tr>
<th>Name</th>
<th>MOTA</th>
<th>MOTP</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointJointNet</td>
<td>4.59</td>
<td>6.01</td>
<td>0.08s</td>
</tr>
<tr>
<td>PC-DAN</td>
<td>22.56</td>
<td>6.03</td>
<td>0.16s</td>
</tr>
</tbody>
</table>

6.1. Multi Object Tracking Accuracy

Multi Object Tracking Accuracy is the biggest single factor that feeds into the rankings of the JRDB benchmark. The MOTA is calculated using the policy based on the number of identifications versus the number of false positives and negatives. Our model with a revision to the loss functions stills continued to be behind the leading competition. With further revisions, it is possible to narrow the gap in the MOTA but there also has to be considered that our current model requires much less information to be fed into it for the results it can achieve.

6.2. Multi Object Tracking Precision

Multi Object Tracking Precision is a metric that is featured in the benchmark but has little to no bearing on the
overall placement. In short, the MOTP is the average amount of space that the bounding box exceeds the target. The object of the testing is that it will allow for what is important to note about is our network

6.3. Run-time

Run-time is the amount of time that a program is able to execute from the start to the finish. It can be inferred that our model will perform approximately twice fast doing the same general operation that the two of the models deem to be acceptable. This is another variable that shows promise for our network being able to be more efficient due to not explicitly doing the having to track point separately based on an existing detentions.

7. Conclusion

There have been many things learned throughout the model we have created. Our model has been built from very powerful tools. We are able to show that the model is able to infer tracking based on it’s detentions as well as showcase the possible improvements in runtime using this model. We have shown that the MOTP that our model has been able to achieve is up to the standard of the leading models that rely on separate tracking and segmentation. Lastly, while the MOTA is not up to that of it’s competitor networks, there is a lot of improvement that can be made to raise the quality up to the standard that it is fully capable of. The model serves as a solid proof of concept that a model that features joint tracking with LIDAR data only is possible achieve.

References