Department of Informatics
Technical University of Munich

Bachelor's Thesis in Information Systems

proAnno - An Automatic and Intelligent 3D Sensor Data Annotation Framework for Autonomous Driving

proAnno - ein Automatisches und Intelligentes Framework für die Dreidimensionale Annotation von Sensordaten zur Unterstützung des Autonomen Fahrens

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Walter Zimmer, M.Sc.

Author Georgiy Nefedov

Date July 15, 2022 in Garching

Disclaimer

I confirm that this Bachelor’s Thesis is my own work and I have documented all sources and material used.

Garching, July 15, 2022

(Georgiy Nefedov)

Abstract

The development of autonomous driving systems has grown substantially over the last several decades. Still, many problems are unsolved, and further developments are expected in the near future [Yur+20]. Many of the applications for autonomous driving require large amounts of pre-annotated train and test data. As a result, the demand for annotated autonomous driving data has surged, and multiple commercial and open source solutions emerged. proAnno is a web-based application for annotating autonomous driving objects on point cloud data and camera images, developed at the Chair of Robotics, Artificial Intelligence and Real-time Systems at the Technical University of Munich. The application is based on the 3D Bounding Box Annotation Tool (3D BAT) [ZRT19] project.

Significant improvements have been achieved in 3D point cloud annotation applications in recent years. For example, several approaches exist for labeling an object with one mouseclick. The user does not need to set all nine box parameters manually. Instead, he can click on a cluster, and the algorithms will identify all points belonging to the object, predict the object's rotation, and adjust the scale and the position of the annotation. This leads to a significant speedup of the annotation process, reducing the time needed for creating an annotation from 70 seconds to 20 seconds per object on average. In addition, developments in tracking algorithms help to annotate an object on multiple frames with only one manual annotation. Other improvements include 3D-2D fusion and convenient spatial and temporal navigation.

The purpose of this thesis is to extend the proAnno annotation application. First, we will evaluate and compare existing one-click annotation, tracking and object detection approaches. Then, the selected algorithms will be implemented. Further, we will enhance the navigation and reduce the steepness of the learning curve by automating the annotation saving process. Last but not least, we will explore possibilities for further improvements.

Zusammenfassung

Die Entwicklung von autonomen Fahrsystemen hat in den letzten Jahrzehnten erheblich zugenommen. Noch immer sind viele Probleme ungelöst, und weitere Entwicklungen sind in naher Zukunft zu erwarten [Yur+20]. Viele der Anwendungen für autonomes Fahren erfordern große Mengen an vorannotierten Trainings- und Testdaten. Infolgedessen ist die Nachfrage nach annotierten Daten zum autonomen Fahren sprunghaft angestiegen, und es sind mehrere kommerzielle und Open-Source-Lösungen entstanden. proAnno ist eine webbasierte Anwendung zur Annotation von Objekten des autonomen Fahrens auf Punktwolkendaten, die am Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme an der Technischen Universität München entwickelt wurde. Die Anwendung basiert auf dem Projekt 3D Bat [ZRT19].

In den letzten Jahren wurden erhebliche Verbesserungen bei Anwendungen zur Annotation von 3D-Punktwolken erzielt. So gibt es zum Beispiel mehrere Ansätze, um ein Objekt mit einem Mausklick zu annotieren. Der Benutzer muss nicht alle neun Boxparameter manuell einstellen. Stattdessen kann er auf ein Cluster klicken, und die Algorithmen identifizieren alle zum Objekt gehörenden Punkte, sagen die Drehung des Objekts voraus und passen den Maßstab und die Position der Annotation an. Dies führt zu einer erheblichen Beschleunigung des Annotationsprozesses, indem die für die Erstellung einer Annotation benötigte Zeit von durchschnittlich 70 Sekunden auf 20 Sekunden pro Objekt reduziert wird. Darüber hinaus helfen Entwicklungen bei den Verfolgungsalgorithmen dabei, ein Objekt auf mehreren Bildern mit nur einer manuellen Anmerkung zu annotieren. Weitere Verbesserungen sind die 3D-2D-Fusion und die bequeme räumliche und zeitliche Navigation.

Das Ziel dieser Arbeit ist es, die proAnno-Annotation-Anwendung zu erweitern. Zunächst werden wir bestehende Ansätze zur Annotation, Verfolgung und Objekterkennung mit nur einem Klick bewerten und vergleichen. Anschließend werden die ausgewählten Algorithmen implementiert. Darüber hinaus werden wir die Navigation verbessern und die Steilheit der Lernkurve verringern, indem wir das Speichern von Annotationen automatisieren. Zu guter Letzt werden wir Möglichkeiten für weitere Verbesserungen untersuchen.

Contents

Chapter 1

Introduction

Driverless autonomous vehicles move from fantasy to reality. Since 2017, level 3 automated systems have been allowed to pilot the car under certain conditions. In addition, a new bill on autonomous driving will allow the use of level 4 autopilots in some regions of Germany as early as 2022 [BMD]. It is no wonder that the research area of autonomous driving is rapidly expanding.

The first chapter introduces the topic of this work by explaining the motivation in section 1.1. The section 1.2 further presents the autonomous driving research project Providentia++ at the Chair of Robotics, Artificial Intelligence and Real-time Systems at TUM. Next, the labeling tool, which will be the main focus of this work, proAnno, is presented in section 1.3. The last section 1.4 of this chapter briefly introduces the contributions of this work.

1.1 Motivation

Over the last decade, deep learning and artificial intelligence algorithms have been significant drivers in autonomous driving research [Gri+19]. 2D camera images and 3D Light Detection and Ranging (LiDAR) point clouds are the most common input types for such applications, while the latter is the prevailing sensing option.

Machine Learning algorithms often require a lot of custom train and test data. The required datasets are massive as they may contain many sequences with hundreds or thousands of frames and tens of objects per frame. Preparing such datasets is a tremendous workload. Therefore, labeling applications must be efficient and straightforward because each hurdle may cost hours or even days of work for a human annotator. Ideally, this should apply to all the steps of the annotation process, whether it is installation, data import, adding or modifying an annotation, or exporting the data.

The procedure of adding and adjusting a label to a point cloud is the most common action. Hence, improvements in this process impose the highest benefits. A typical LiDAR sensor perceives between 10 and 30 frames per second [Yel], and there are around ten objects per frame on average in the nuScenes dataset [Pha+19b]. As a result, the nuScenes dataset is labeled at 2 Hz, although the dataset was recorded at 10 Hz using a Velodyne HDL32 LiDAR.

Inserting and modifying bounding boxes is only one of the significant hurdles of working with 3D point cloud annotation software. In addition, installing and configuring the application

for custom datasets must be as straightforward as possible to flatten the learning curve for new users. Further operations should be automated to enable the user to concentrate on his primary goal of annotating point cloud clusters; the software should manage everything else out of the box. For example, the annotations should be automatically saved periodically so that the human annotator does not have to download the files manually.

1.2 Providentia++

This work is done in the scope of the Providentia++ project. The Federal Ministry has funded the project for Digital and Transport since 2017. Since 2020, it has been led by the Chair of Robotics, Artificial Intelligence, and Real-time Systems at the Technical University of Munich's Department of Informatics.

Initially, two measurement stations with cameras and Radio Detection and Ranging (Radar) sensors were installed on a test section of the A9 highway near Garching. The sensor data was used to identify vehicles using artificial intelligence and then fused. The resulting digital twin can be used by autonomous cars to make decisions based on events that could not be perceived solely by the vehicle's onboard sensors.

The second phase of the project started in 2020. Light Detection and Ranging (LiDAR) sensors were added to the existing perception methods. In addition, the test stretch was expanded into the residential area, which allows surveying intersections, traffic circles, bus stations, and other urban situations.

Figure 1.1: The image shows the top view of the road stretch covered by the Providentia++ measurement stations. The green stretch and the S40 and S50 sensors belong to the initial project; the remaining stations were erected as part of the second phase.

1.3 proAnno

For the proAnno annotation application, the 3D Bounding Box Annotation Tool (3D-BAT) [ZRT19] is taken as the baseline and developed upon in the scope of the Providentia++ project. The application supports the semi-automatic labeling of ten classes of autonomous driving objects in point clouds. Further, 3D to 2D projection may be enabled to create image annotations from point cloud annotations automatically. The annotator may work in the orthographic birds-eye-view view to reduce the number of degrees of freedom and simplify the labeling or switch to the perspective view and observe the point cloud using all nine degrees of freedom. The proAnno tool was used to annotate the firsth batch of the A9-Dataset [A9], which was published as part of the Providentia++ project. Hopefully, the contributions of this work will enable the expansion of this dataset by providing more advanced and intuitive labeling functionality.

1.4 Contributions

This work comprises several contributions, each improving the annotation quality and pace by improving various flanks of the process. One axis of improvement is the automation of the process with such features as one-click annotation, tracking, and object detection. Ultimately, a human annotator can use these features combined to annotate a sequence of frames with two clicks. Nevertheless, automatically generated labels might not be ideal. Then, the human annotator might need to perform manual adjustments and corrections.

Manual annotations are enhanced by keyboard shortcuts that allow performing almost all the operations. In addition, with visual support like RGB point clouds and HD Maps, manual annotations can be quickened, and the accuracy increased.

One Click Annotation A bounding box of a 3D object has nine degrees of freedom. The initial implementation of proAnno required the user to specify all of them. The user might click on an object with one click annotation, and a new bounding box will appear with the correct rotation, scale, and position. This functionality reduces the number of clicks for annotating an object from tens to one.

Tracking If an object is labeled in one point cloud, the bounding box can be automatically transferred to neighboring frames and the whole sequence. This feature uses the moving average filter with a Euclidean-space-based growing algorithm to track the object throughout the recording. As a result, annotating a vehicle in all frames is simplified to annotating it in one frame. Combined with one click annotation, adding a label to all frames now requires one click for annotating the object in one frame and another click to initiate tracking. Other features like automatic saving flatten the learning curve for inexperienced users by automating routine operations.

Keyboard Shortcuts It is possible to use the keyboard to fly around through the scene, rotate the camera, switch view modes, change selected classes, rotate, scale and translate bounding boxes, change the selection, delete labels, undo, redo, and more. As a result, almost all main functionality can be utilized with key commands only. According to several studies,

keyboard navigation may be more comfortable to use and speed up the work for experienced users [Jor+02; KMA86; Lan+05].

Visual Aid For a human, it might be hard to differentiate a pedestrian from a road sign, a bicycle from a motorbike, etc., if the point cloud is single-colored and without any visual aid. Therefore, proAnno now supports RGB point clouds and HD Maps. Figure 1.2 nicely illustrates how much easier it is to identify objects with HD Map, RGB point clouds, or both.

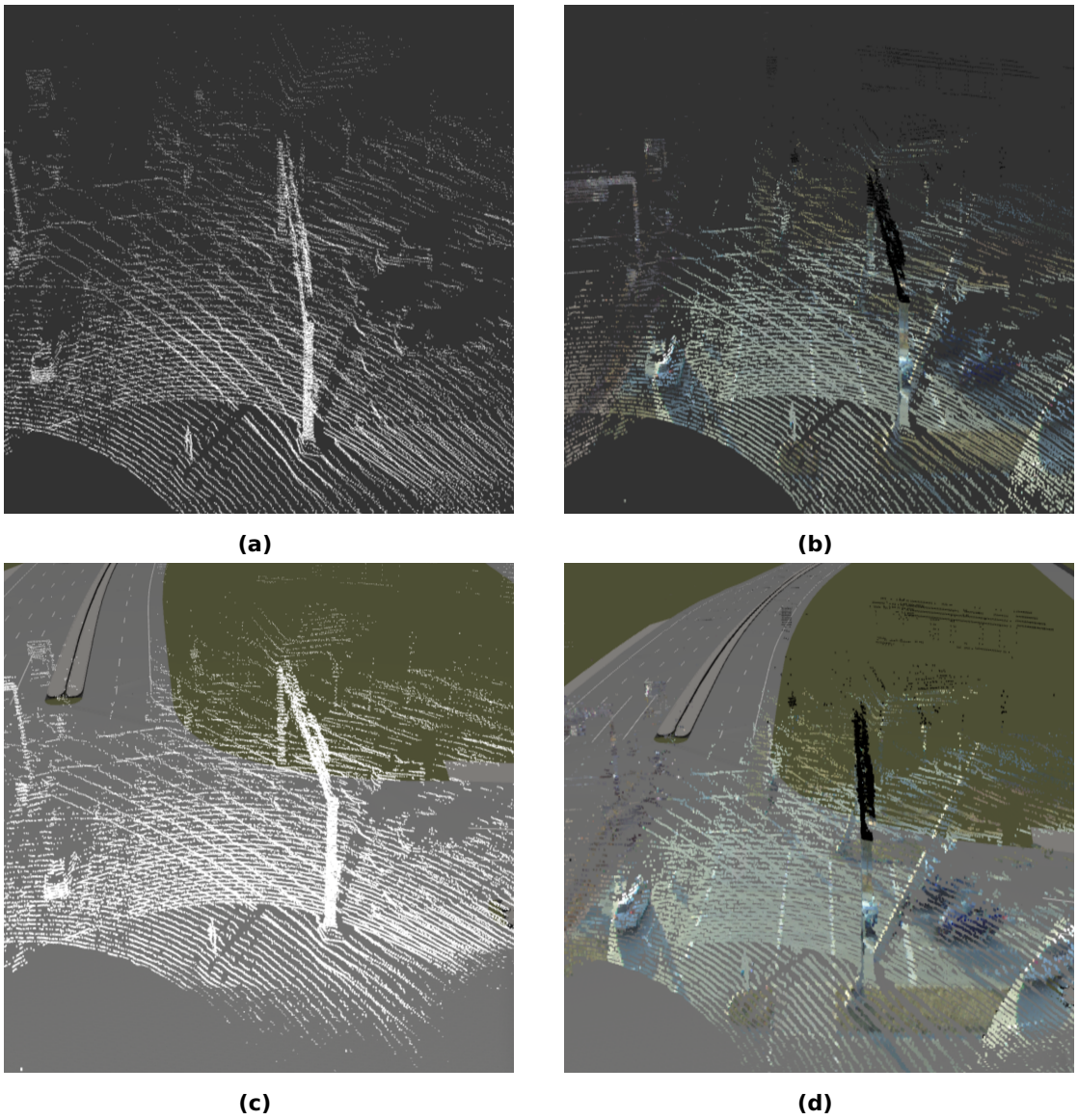


Figure 1.2: A point cloud frame recorded at the S110 Providentia measurement station .
a) Single-colored pointcloud with a dark background
b) RGB point cloud with black background
c) Single-colored point cloud with HD Map
d) RGB point cloud with HD Map.

Automatic Saving Previously the human annotator had to download the annotations manually as a zip archive. Then, to open the downloaded annotations, it was necessary to unzip

them and move them to a specific location in the input folder. Although not a complex operation, it is a routine task that the annotator can easily forget about, and the annotations will be lost. A newbie also needed to learn where to put the annotations.

Further, a significant part of this work consisted of exploring state-of-the-art solutions. Some of these have been implemented in proAnno, while others are discussed in chapter 5.

Chapter 2

Related Work

Researchers can select between using publicly available pre-labeled datasets or creating custom datasets from their recordings of point cloud sequences. This chapter will discuss currently publicly available tools for 3D labeling. In Section 2.1, we will mention some widely used autonomous driving datasets. We will not provide a detailed description of each dataset; in the scope of this work, the more important question is how the bounding boxes in the point clouds were created for these datasets. Section 2.3 introduces the most popular open-source applications and their feature sets. Further, section 2.4 briefly describes a recently introduced standard for annotation format OpenLABEL by ASAM [Cro].

2.1 Annotated 3D Autonomous Driving Datasets

The need for labeled autonomous driving data is not new. Hence, several publicly available pre-labeled autonomous driving datasets are available. These contain many sequences with various traffic situations, sensor types, and weather conditions. There are obvious advantages in using such datasets instead of collecting and labeling custom data; the researchers do not have to spend much time and money. Instead, they can use a large amount of freely available materials. In the following, we will discuss some of the most extensive and annotated autonomous driving datasets with LiDAR point clouds and bounding box annotations [Dat]. Mainly, we are interested in comparing the 3D labeling tools used for annotating vast amounts of data.

KITTI The dataset [Gei+13] was initially published in 2011, and it is still being advanced [Beh+19; LXG21a]. It was the first publicly available autonomous driving dataset of its kind. The data was collected in rural and urban areas of Karlsruhe, Germany, by researchers from the Karlsruhe Institute of Technology and Toyota Technological Institute in Chicago. Besides LiDAR scans, it includes color and greyscale camera images and GPS/IMU navigation data. With the successor dataset KITTI-360, the KITTI-360 labeling tool was released [LXG21b]. The application was used to label the new dataset's 68 thousand point cloud objects.

A9-Dataset This dataset was created in the scope of the Providentia++ project [A9]. Unlike most other datasets, the data was not collected from sensors installed on a moving vehicle but from cameras, LiDAR, and radar sensors at the project's measurement stations. It contains three camera recordings with image annotations and two point cloud sequences with bounding

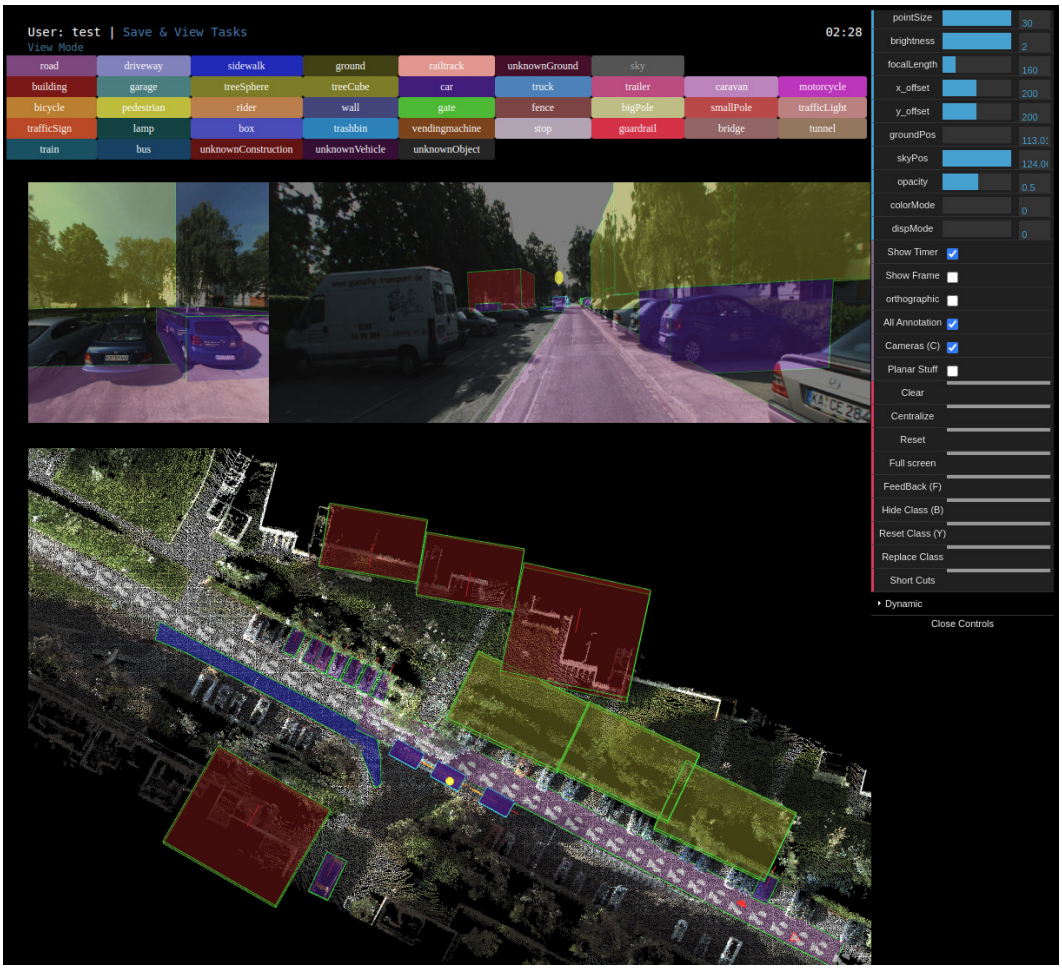


Figure 2.1: KITTI-360 label tool graphical user interface.

box annotations. proAnno is the main tool used for labeling 3D frames in the first batch of this dataset.

Waymo Open Dataset It is the largest published dataset of autonomous driving data containing around 12 million 3D bounding boxes [Sun+20]. However, the work was accomplished by Waymo, and the used labeling applications were not disclosed.

nuScenes Another vast dataset with various annotation types on 2D and 3D frames[Cae+19]. Among others, it contains 1.2 million 3D labels. Scale AI provided these annotations; hence, the tools used to create the bounding boxes are not disclosed.

Other related datasets include ApolloScape[Hua+18], A2D2[Gey+20], A*3D[Pha+19a] and Argoverse[Wil+21]. Most authors used third party services to manually annotate the collected data while some chose not to disclose which tools were used.

2.2 Outsourcing

Plenty of companies are willing to annotate any time of data with various annotations. Outsourcing annotation work is an easy way to get customized annotated data. However, this option has its downsides as well. First, such annotations are usually costly. A project without proper financial backing might be unable to afford to outsource the labeling work because of budget constraints. The Table 2.1 summarizes the costs per 100k 3D annotations in LiDAR frames for several companies. Another problem with outsourcing is possible quality issues. A dataset with bad quality labels is not useful in most cases.

Company	Costs for 100k boxes
UnderstandAI	EUR 3992
Supervisely	$6000
ByteBridge	$9000
Hive	$7500

Table 2.1: Comparison of costs for annotating 100k 3D labels on LiDAR recordings.

2.3 3D Point Cloud Annotation Applications

In the following, we evaluate several applications for annotating autonomous driving data. The list is limited to open-source tools because commercial providers do not disclose details of their implementation, and their functionality does not exceed the toolset of those available for public access. Specifically, we will focus on the feature that could augment proAnno from SUSTechPOINTS, LATTE, SAnE and the KITTI360 labeling tools.

2.3.1 3D Bat

3D Bounding Box Annotation Tool (BAT) is a web-based application developed in javascript with tools for efficient and accurate 3D localization and tracking of objects using full-surround, multi-modal data streams [ZRT19]. It was developed by Walter Zimmer, Akshay Rangesh and Mohan Trivedi at the University of California, San Diego and introduced to the public in May 2019. The implementation of this tool serves as a base for proAnno.

Besides annotating objects in point clouds, the 3D BAT tool supports projecting these annotations to the image domain. Therefore, the same object can be translated to annotations in multiple images from one point cloud annotation.

Scaling, rotating and moving a label box in the perspective view is tedious because the user must change the perspective several times. 3D BAT's master view makes this process convenient by displaying the box's top, side, and front view next to the perspective view making annotation manipulations more convenient.

Another available functionality is smart interpolation which enables semi-automatical labeling. Moreover, the tool is the first to introduce keyboard-only labeling, a handy way to navigate and manipulate annotations.

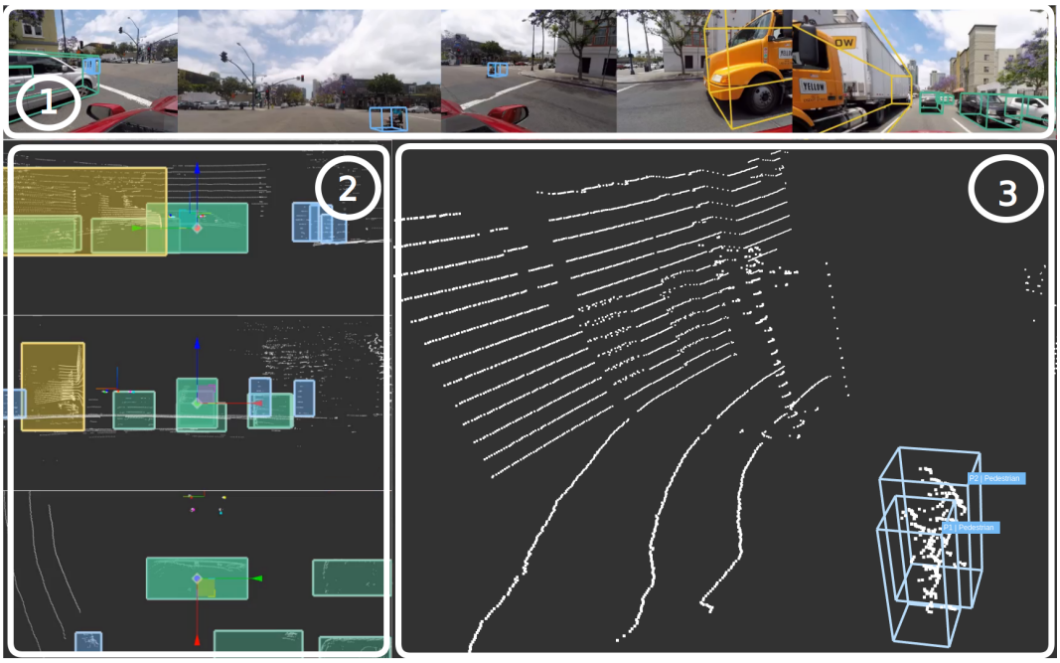


Figure 2.2: Overview of the 3D BAT annotation toolbox. (1) A horizontal scrollable and vertical resizable panoramic camera image provides a full-surround view. (2) The Masterview that consists of a side view (top), a front view (middle) and a top view (bottom) supports the user during the annotation process. (3) 3D view in that the user can navigate and place annotations. [ZRT19]

2.3.2 SUSTechPOINTS

The Portable pOint-cloud Interactive aNnotation plaTform System (POINTS) [Li+20] advances the annotation process for autonomous driving frames with such features as one-click annotation, tracking and easy user-data interactions. It was developed by E Li, Shuaijun Wang, Chengyang Li, Dachuan Li, Xiangbin Wu, and Qi Hao at the Southern University of Science and Technology (SUSTech) [SUS].

One-click annotation is the most attractive feature of SUSTechPOINTS which is implemented in proAnno as part of this work. For labeling an object, a human annotator has two options. First, he can hold down the CTRL button to lock the 3D view and select a 2D rectangle that encloses all the points of the desired object. Then, the application uses a PointNet-based neural net to determine the rotation of the framed object; it pastes a bounding box, applies an iterative growing algorithm to find all points belonging to the object, and shrinks the box to the size. Alternatively, the human annotator can right-click on the object and select the object type from the context menu. Next, the tool will place a corresponding box at the mouse pointer's location and use a Euclidean space growing algorithm to grow the annotation box. Afterward, the rotation and box size will be adjusted as well.

Tracking is available for sequences of data frames. The annotations are copied from one frame to another, and the position is adjusted using a 3D data registration algorithm [Yan+16a]. It is assumed that the object scale does not change; therefore, the scale is taken from the initial bounding box [Li+20].. The application requires a 3D object tracking algorithm to locate the corresponding source and target point clouds before the registration algorithm can be applied. This step can be done manually if there is no tracking algorithm. The user needs to copy

and paste the annotation from one frame to another or add a new label in the next frame and copy the ID from the previous frame.

Alike 3D BAT, SUSTechPOINTS implements a convenient master view that makes bounding box manipulations more convenient. In this implementation, it is also possible to manipulate the annotation through these sub-views. Another positive aspect is that the project is actively maintained and improved.

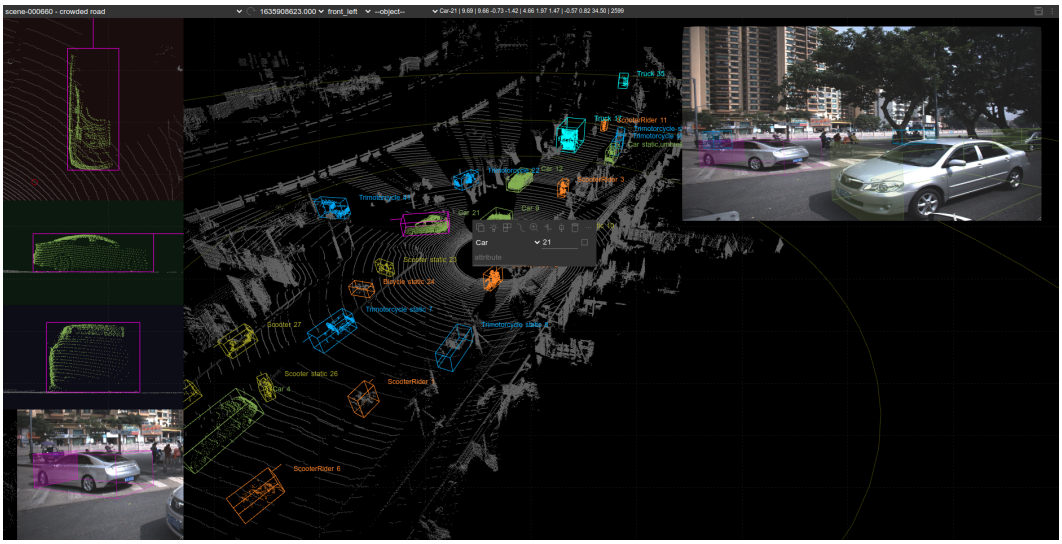


Figure 2.3: Main User Interface of the SUSTechPOINTS annotation application. The perspective view as the main view with side views (top, right, back) on the left side and camera images. [Li+20]

2.3.3 LATTE

LATTE was developed at the University of California, Berkeley by Bernie Wang, Virginia Wu, Bichen Wu, and Kurt Keutzer. Like proAnno, it supports a perspective view and top orthographic view. In addition, this application implements sensor fusion to simplify the annotation process. First, the point cloud is projected to the corresponding image. Then, Mask-RCNN [He+17] 2D detectors are used to detect objects on the image. After detection, the images inside the bounding boxes are segmented. The points inside the mask are then projected back to the point cloud and pre-labeled, in other words, highlighted. Then, the human annotator draws the bounding box around these highlighted points. These are again projected to the image domain, and the part of the image with the annotated object is displayed to the annotator to confirm.

One-click annotation is implemented in LATTE as well. However, an approach entirely different from POINTS is used. First, the ground is removed by analysing the singular values of the point cloud dispersion in the pre-processing step. Then, a clustering algorithm is used to cluster the objects from the 2D top view. The algorithm finds the nearest cluster and adds a label when a human annotator clicks on a point cloud. Such labels differ from the labels produced by the previously described annotation applications; the object is annotated from the 2D top view. They comprise only 5 degrees of freedom (DOF), compared to 9. Nevertheless, it is sufficient for most autonomous driving applications.

The application is modular and therefore supports different tracking algorithms. The standard implementation uses Kalman [WB95] filtering to predict the position of the object center in the neighbouring frame. The correction step is later performed manually by a human annotator. Therefore this approach is not fully automatic. The Kalman filter and clustering algorithm used in LATTE are discussed in the following.

Kalman Filter

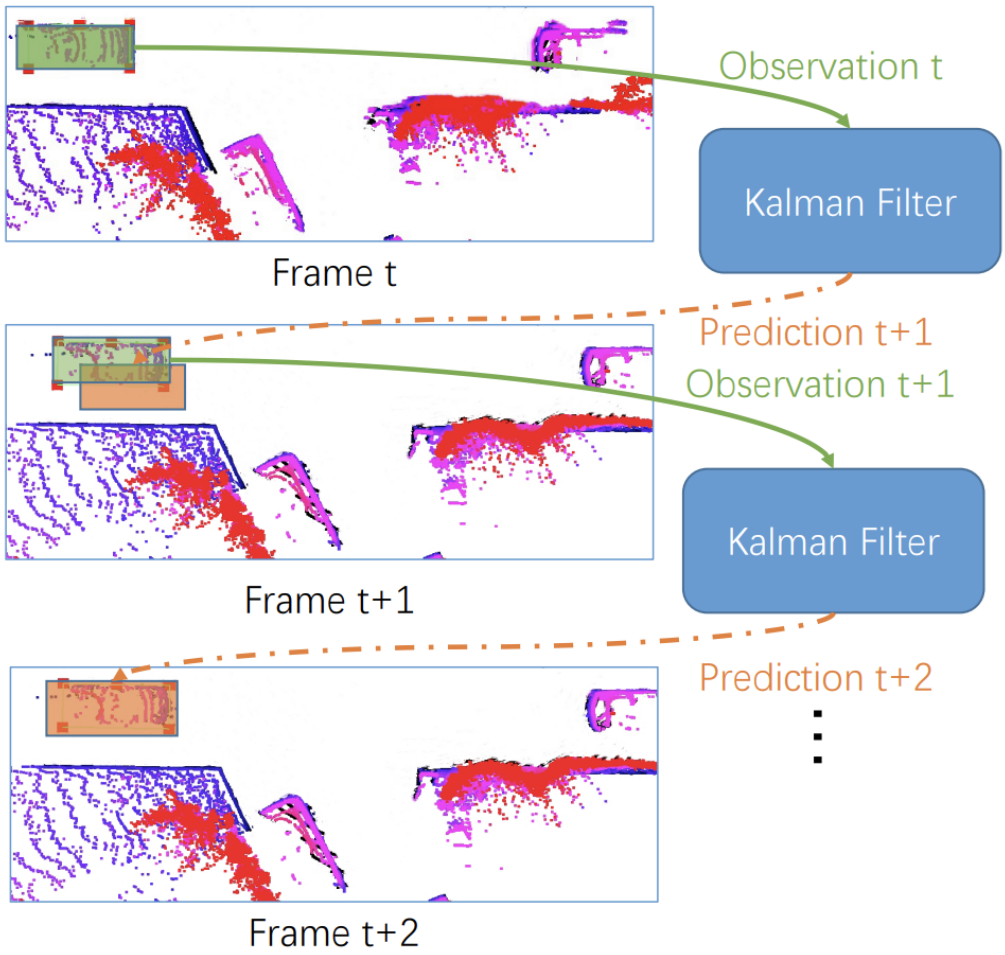


Figure 2.4: Visualisation of Kalman filter applied to two annotation transfer operations on three consecutive autonomous driving point cloud frames. The predictions are displayed in orange and the observations/measurements are green. [Wan+19]

The Kalman filter [WB06] is a model-based filter for estimating the state of an object based on previous measurements. Since its invention in the 1960s, it has become the default choice for many use cases. Primarily, it is often used in radar and tracking applications. The algorithm has proven reliable and superior to other algorithms measured by the mean squared error. Along with other model-based filters, such as dynamic data reconciliation (DDR) [BTM06], the Kalman filter is well suited for processes with significant dynamics, as in the case of autonomous driving objects.

The Kalman filter uses a form of feedback control to estimate a process. One iteration of the algorithm consists of the steps of prediction and measurement. In the first step, data from previous measurements are used to estimate the object's state in the next frame. After that, a human annotator must adjust the bounding box, and the corrected annotation serves as the ground truth measurement. This measurement updates the filter, and the next iteration can be performed.

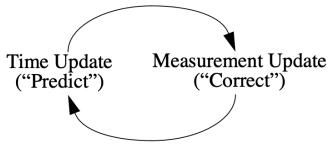


Figure 2.5: The ongoing Kalman filter cycle. [WB06]

The state of the object in frame k is modeled as

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{2.1}$$

with A relating the state in frame $k-1$ to the state at frame k and B relating an optional control input u to the state x_k, and w_{k-1} being the process noise with normal distribution and covariance Q. The measurement (the corrected bounding box) in frame k is modeled as

$$z_k = Hx_k + v_k \tag{2.2}$$

where Hx_k describes the ground truth state and v_k is the measurement error, or in our case, the annotation error. The variable follows normal distribution as well.

In the following, variables with a super minus describe state estimate provided knowledge of the states up to frame $k-1$ while variables without the minus estimate the state given the measurement z_k. The time update step solves two equations

$$\hat{x}^- = A\hat{x}_{k-1} + Bu_k \tag{2.3}$$

$$\hat{P}_k^- = A\hat{P}_{k-1}A^T + Q \tag{2.4}$$

and the measurement update step solves three equations

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.6}$$

$$P_k = (I - K_k H)P_k^- \tag{2.7}$$

The P_k is the estimate error covariance defined by $P_k = E[e_k e_k^T]$, R_k is the measurement error covariance, and the K_k variable denotes the Kalman gain factor. This factor decides how much the prediction and how much the measurement is weighted for the prediction of the next frame.

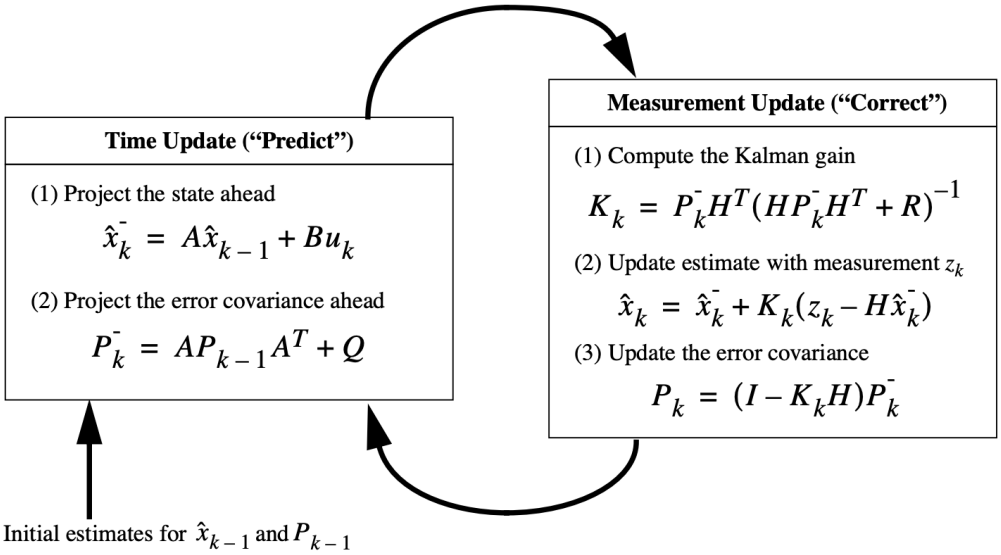


Figure 2.6: The ongoing Kalman filter cycle with equations for each step [WB06].

Clustering

The clustering algorithm in LATTE requires point cloud frames without ground. The ground plane is estimated iteratively. Initially, the estimated ground plane is spanned by the x and y axes, and the z-axis represents the distance from the ground. The iteration begins with sampling the closest points to the ground plane. Then, singular value decomposition is performed, and the three singular vectors represent a coordinate system where the first two vectors span the new ground plane. The third is the normal vector representing the distance from the ground. These steps are repeated with the new coordinate system until the ground plane estimation converges.

After that, the points are projected to the ground plane, and the algorithm searches for clusters in this 2D projection. The clustering is based on density-based spatial clustering of applications with noise (DBSCAN). A human annotator needs to have the 2D top-view open and click on a point in the desired point cloud to annotate an object in one click. Then, the algorithm finds the nearest cluster to the selected location and adds a label.

The performance of annotation applications is an essential aspect. Lidar frames can contain many thousands of points, and iterating over them all might impact the calculation time. Therefore, LATTE assumes upper bounds for the dimensions of a car and prunes the point clouds. For this, information on the distribution of bounding box sizes in autonomous driving data is used.

One limitation of the described approach is that point clouds are usually sparse. Therefore, the quality of the top-view 2D projection is generally not good enough to separate closely located objects. Furthermore, noise is another hurdle for clustering algorithms. A noisy cluster may result in an incorrect bounding box shape and rotation prediction. The Smart Annotation and Evaluation (SAnE) [Ari+20] annotation tool aims to address these issues by denoising pointwise segmentation. The denoising is done by increasing penalties for wrong predictions near object boundaries during the training process. This approach also eliminates the need to

remove the ground plane points. Another improvement of SAnE is using the Nearest Neighbor (NN) search instead of DBSCAN because the latter is computationally expensive. The authors also claim that their approach achieves at least similar results for region proposals.

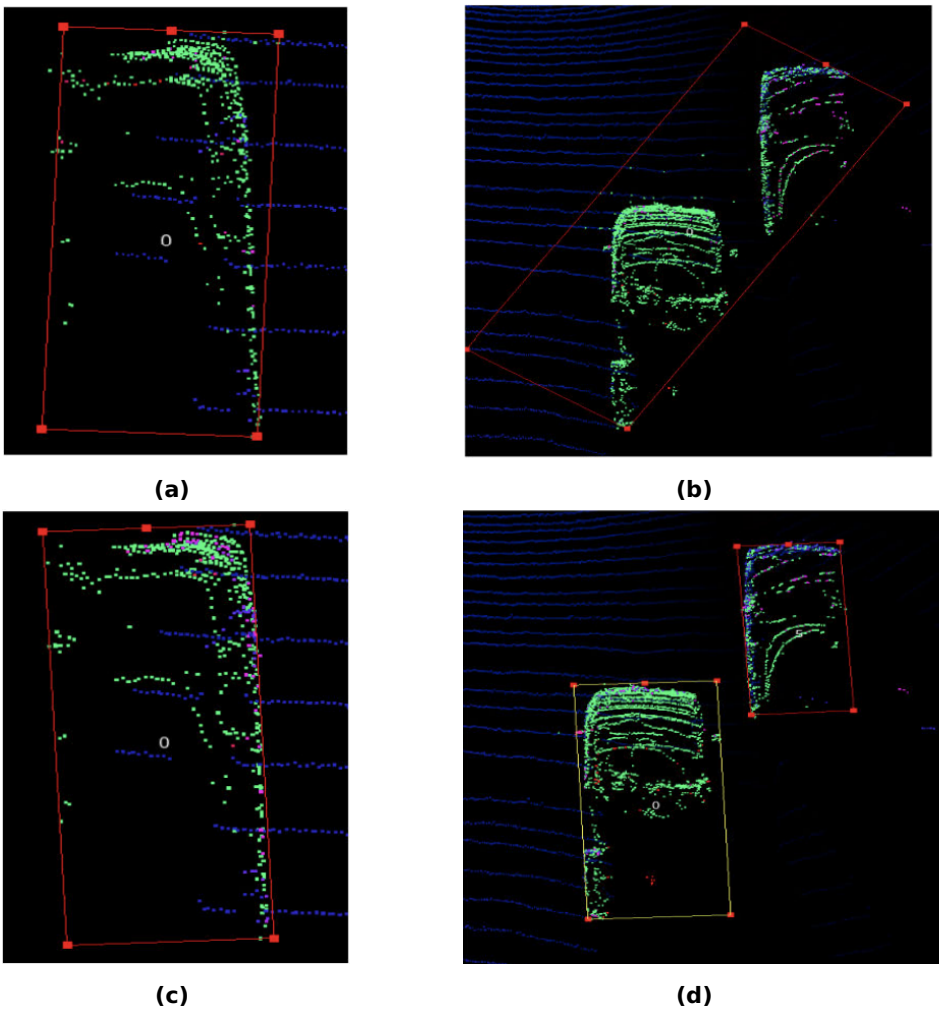


Figure 2.7: Figures a and b show the impact of noise on the accuracy of one-click annotations for a single object and two closely located cars. Figures c and d show the result of a one-click annotation after denoising the frame.

2.3.4 SAnE

The Smart Annotation and Evaluation tool (SAnE) for point cloud data [Ari+20] was developed by researchers at the Norwegian University of Life Sciences and the Carnegie Mellon University Hasan Asy'ari Arief, Mansur Arief, Guilin Zhang, Zuxin Liu, Manoj Bhat, Ulf Geir Indahl, Håvard Tveite, and Ding Zhao. It is inspired by LATTE and, therefore, shares many features. The main difference between these applications is the background logic. For instance, SAnE automates the correction step of the tracking algorithm with a greedy search, backtracking and refinement. For fitting the boxes during one-click annotation, an l-shape fitting algorithm is used. We will discuss both features in more detail in the following.

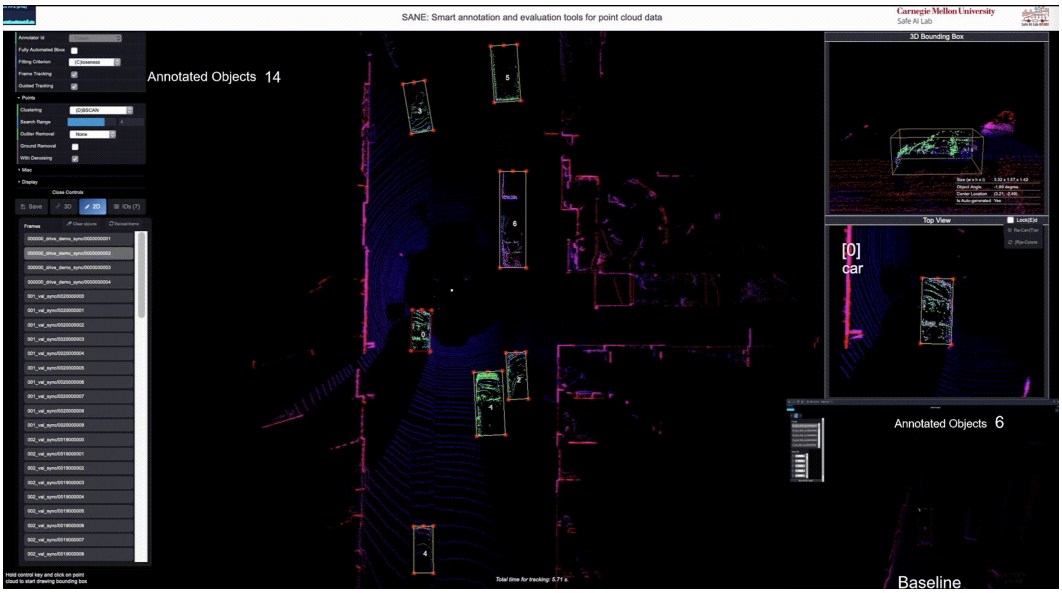


Figure 2.8: Main view of the SAnE application. [has]

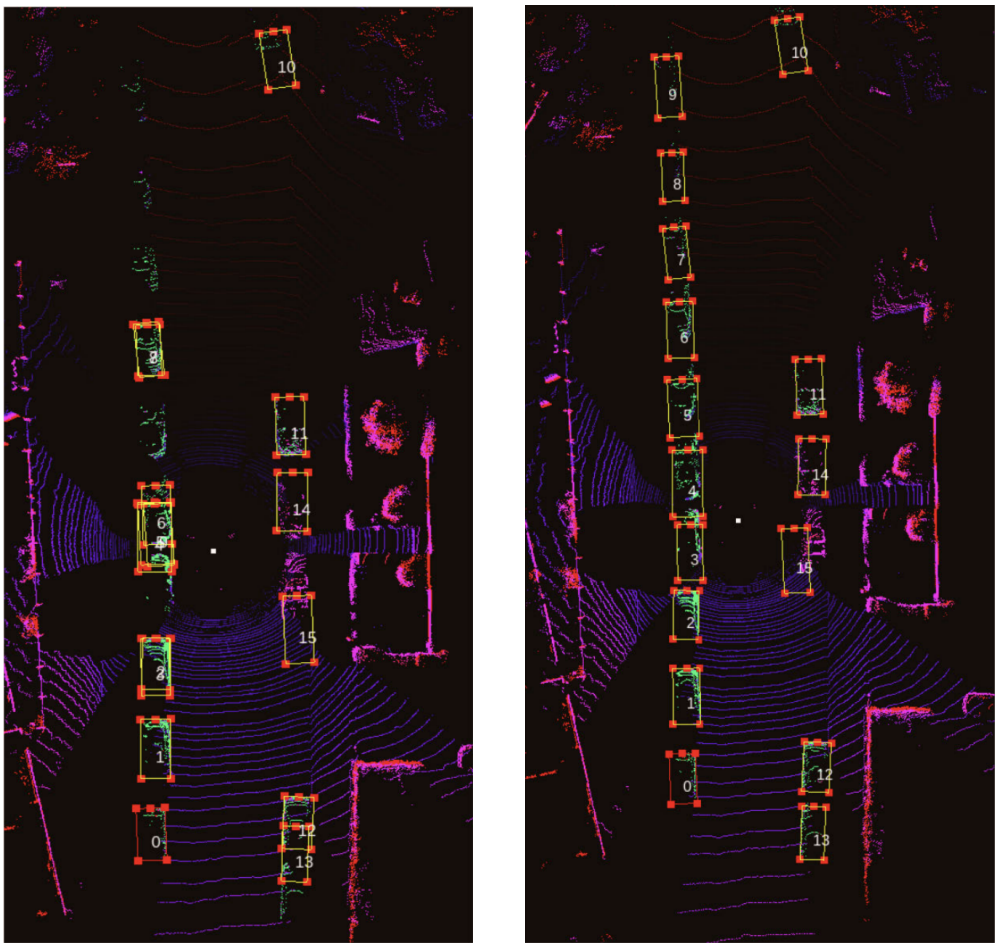
Tracking

Alike the previously described approaches, the tracking algorithm in the SAnE annotation tool performs annotation transfer between frames in a prediction and an adjustment step. First, for prediction, an extended motion model is utilized. After that, the adjustment procedure consists of three subroutines: greedy search to regress each bounding box to the nearest cluster, backtracking to resolve overlapping bounding boxes, and refinement to adjust the bounding boxes to the nearest cluster without overlap [Ari+20].

Greedy Search After predicting the new location of the bounding box, a recursive greedy search moves each bounding box to the nearest cluster. Briefly summarized, the algorithm moves the bounding box around the predicted center position. Each iteration counts the number of enclosed points and their average distance to the nearest bounding box edge. The objective is to maximize the former while minimizing the latter.

Backtracking In the previous step, it might happen that two bounding boxes will be regressed to the same cluster. However, because of the assumption that each cluster represents one object, such prediction is undoubtedly wrong. Therefore, the backtracking algorithm aims to resolve the overlap by reversing the last tracking adjustment of the bounding box with the most significant distance between the original bounding box center and the regressed one.

Refinement After backtracking, the bounding boxes are probably not optimal because not all of them are regressed to fit the nearest cluster. However, the positions are closer to the object's actual location now than at the beginning of the iteration. Therefore, the greedy algorithm can be applied again, but with a smaller padding parameter to prevent repeated overlap.



(a) (b)

Figure 2.9: SAnE's backtracking algorithm applied on a frame with overlapping bounding boxes after greedy search [Ari+20]. a) after greedy search, before backtracking b) backtracking applied

L-Shape Fitting

The L-Shape fitting algorithm used in LATTE is applied to the top view projection of the point cloud cluster. Hence, it only generates a 2-dimensional bounding box, disregarding the height of the object and the roll and pitch rotations. Nevertheless, such top-view bounding boxes are sufficient for most autonomous driving applications.

To find the best fitting box Xiao Zhang et al. formulated an optimization problem with a least-squares-like loss function. The parameters are θ - the direction of the vehicle, P and Q are two disjoint sets of points of the cluster, and the line expressions $c_1 = x \cdot cos\theta + y \cdot sin\theta$ and $c_1 = -x \cdot sin\theta + y \cdot cos\theta$. A point is assigned to set P or Q if it is closer to c_1 or c_2 correspondingly. The optimization problem is formulated as follows:

$$\begin{aligned} \underset{\theta,P,Q,c_1,c_2}{\arg\min} \quad & \sum_{i \in P} (x_i \cdot cos\theta + y_i \cdot sin\theta - c_1)^2 + \sum_{j \in Q} (-x_j \cdot sin\theta + y_j \cdot cos\theta - c_2)^2 \\ \text{subject to} \quad & P \cup Q = G, P \cap Q = \emptyset \\ & c_1, c_2 \in R \quad 0^\circ \leq \theta \leq 90^\circ. \end{aligned}$$

Although we cannot feasibly solve this optimization problem, it is possible to find an approximate solution. The algorithm iterates over all angles between 0 and 90 to find the minimum

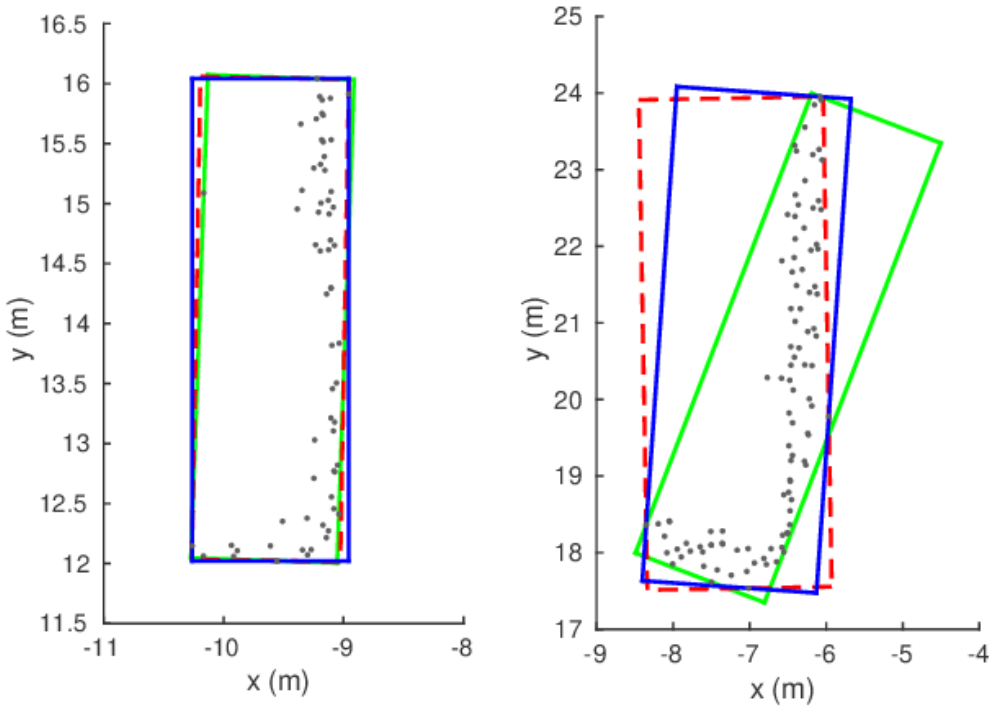


Figure 2.10: Two rectangle fitting examples. The grey dots represent the scanned point cloud points in BEV. In green, red and blue are the fitted top view bounding boxes criteria area minimization, closeness maximization, and variance minimization. [Zha+17]

box enclosing the whole cluster, split the points in P and Q and calculates the loss for this angle.

If θ^* denotes the θ that yields the minimal loss is chosen, then the corresponding edges of the box are calculated as

$$E_1 = x \cdot cos\theta^* + y \cdot sin\theta^* - min\{c_1^*\} \tag{2.8}$$

$$E_2 = x \cdot cos\theta^* + y \cdot sin\theta^* - max\{c_1^*\} \tag{2.9}$$

$$E_3 = x \cdot sin\theta^* + y \cdot cos\theta^* - min\{c_2^*\} \tag{2.10}$$

$$E_4 = x \cdot sin\theta^* + y \cdot cos\theta^* - min\{c_2^*\} \tag{2.11}$$

with $c_1 = X \cdot (cos\theta^*, sin\theta^*)$ and $c_2 = X \cdot (-sin\theta^*, cos\theta^*)$.

Alternatively to the squared error loss function, [Zha+17] proposes two other possible criteria: area minimization and closeness to the nearest edge. These three loss functions and the PCA were compared on CMU's SRX vehicle data. The car has six IBEO LiDARs mounted, and it collected the data on local roads in Pittsburgh. The resulting real and absolute errors are compared in the Table 2.2. The illustration 2.10 demonstrates the different fitted boxes using these criteria on two sample point clusters.

The table reveals that the variance criteria, i.e., the least-squares loss function, yield the lowest error. This is a possible reason why LATTE uses this criterion as the default option.

	Real Error ($\theta - \theta_g$)		Absolute Error ($\|\theta - \theta_g\|$)	
Method	Mean (deg)	STD (deg)	Mean (deg)	STD (deg)
PCA	-5.60	10.68	10.09	6.57
Area	0.65	14.80	11.78	8.46
Closeness	0.01	3.65	2.47	3.36
Variance	-0.15	2.19	1.55	1.66

Table 2.2: Rotation error comparison [Zha+17]

2.3.5 KITTI-360 Label Tool

As mentioned in the previous section, the team behind the KITTI-360 dataset developed this application specifically for labeling their point cloud frames [LXG21b]. Therefore, the KITTI-360 label tool is more than just an annotation application compared to other mentioned applications. This application is designed for multiple users with different levels of privileges. Admin users can add tasks that contain sequences of point cloud frames to be labeled. Then, human annotators pick up these tasks and label them. In other words, it implements the entire collaboration workflow around the labeling application.

Another distinguishing aspect is that the vast amount of annotation classes allow annotating every object in the frame. So besides annotating instances like cars, pedestrians, or cyclists, it is possible to annotate stuff like roads, sidewalks, and driveways where there is no individual object of this category. Stuff is usually planar. The annotations for individual objects may be of different shapes, while only a box shape is supported in most other tools. Dynamic objects that change their position between frames can be annotated semi-automatically by placing the annotation in one frame and drawing its trajectory.

The authors split their data into batches of around 240 frames to evaluate the tool's performance and measured the time needed for labeling such sequence. They came to an average time of 3 hours per batch or 45 seconds per frame. Overall, the KITTI-360 label tool is designed for effective collaboration and supports a wide range of autonomous driving annotations.

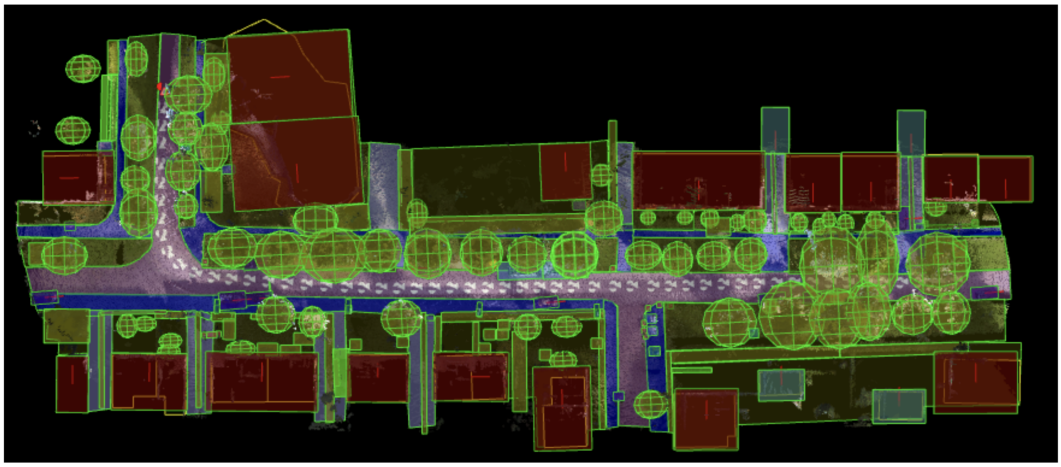


Figure 2.11: Birds eye view of a road scan with all objects annotated with the KITTI-360 label tool.

2.4 OpenLABEL Standard

Dataset	Year	Annotation Types	Annotation File Format
Cityscapes	2016	Pixel Level	PNG, JSON
Synthia	2016	Pixel Level	PNG, TXT
Mapillary-Vistas	2017	Pixel Level	PNG M
BDD100K	2017	2D Bounding Box Pixel Level	PNG, JSON
KITTI	2012	2D/3D Bounding Box	TXT/SSV
ApolloScape	2018	2D Bounding Box Pixel Level	PNG, JSON
nuScenes	2018	3D Bounding Box	JSON
Lyft	2019	3D Bounding Box	JSON
BLVD	2019	3D Bounding Box	TXT
Waymo	2019	2D/3D Bounding Box	Tensorflow Records/Protobuf
Drive & Act	2019	Actions 3D Bounding Box	CSV
A9-Dataset	2022	2D/3D Bounding Box	JSON (OpenLABEL)

Table 2.3: Open source autonomous driving datasets file formats [NSO21].

Without standardization, there would be as many annotation file formats in different languages as the number of annotation tools. For instance, the most popular autonomous driving datasets all use different formats. SSV files are used in KITTI. The nuScenes dataset uses a custom format, also adopted in Lyft Level 5 dataset. Waymo defined its dataset format in Google's proto format. A custom format is used at the PASCAL Visual Object Classes Challenge. KITTI [Gei+13], COCO [Lin+14], and YOLO [Red+15] are the most popular formats in autonomous driving. As a result, many development kits contain tools for converting annotation files between a vast amount of specifications. Therefore, using an existing dataset for training a model requires understanding the format of the dataset of interest, choosing a format one is comfortable working with and converting it to the required format first. The Table 2.3 compares the file formats used in autonomous driving. Adhering to an exchangeable format will make the interaction between autonomous driving vehicles smoother, reducing the risk of causalities. Further, annotation quality will increase because, at the current state, annotators need to complete training before annotating each task, as every dataset needs to be annotated with a different tool. Also, only one conversion to the OpenLABEL format would be necessary for existing datasets.

The Association for Standardization of Automation and Measuring Systems (ASAM) with

Deepen AI released the OpenLABEL paper for a labeling standard. Besides the file format, the OpenLABEL covers the labeling methodology and the labeling structure. The association works with numerous customers in the autonomous driving industry and understands the need for standardization. They argue that unstandardized formats are a safety threat if used in a level two or higher autonomous driving system (ADS). Hampered Vehicle2Vehicle Interaction, Precluded sharing, Lowered Annotation quality, and the Deprecation of old labels are the main issues mentioned on their website [Cro]. Consequently, the proAnno annotation tool and the A9-Dataset will fully adopt the OpenLABEL format to comply with the new industry standard.

Format Description The OpenLABEL standard is defined in the JavaScript Object Notation (JSON) schema. This schema file will be used to verify the format of the annotation files. The flexibility of the format allows for storing simple object-level and more complex multi-stream-level annotations with scene semantics. The basis of an annotation file conforming to the schema may include six concepts: Elements, Frames, Streams, Coordinate Systems, Metadata, and Ontologies.

Elements Objects, Actions, Events, Contexts and Relations fall under this concept. All elements must have the mandatory attributes `name` and `type`; other element data varies by element class. The meaning of an object and context is rather self-explaining. The paper [Cro] gives "isWalking" as an example of an action and "startsWalking" for an event. The difference is that action extends through several timeframes and describes a prolonging activity, whereas an event is a one-time instance.

Listing 2.1: An excerp of an annotation file followong the OpenLABEL schema. Here, elements of type action, event and context are defined. [Cro]

```
{
   "openlabel": {
       ...
       "actions": {
           "0": {
               "name": "following1",
               "type": "following",
               "frame_intervals": [{"frame_start": 0, "frame_end": 10}]
            }
       },
       "events": {
           "0": {
               "name": "crossing1",
               "type": "startsCrossing",
               "frame_intervals": [{"frame_start": 5, "frame_end": 5}]
            }
       },
       "contexts": {
           "0": {
               "name": "",
               "type": "Urban"
            }
        }
```

```
24     }
25 }
```

Relations are also modeled as elements; a relation is a triple of a subject, an object, and a predicate. Nevertheless, unlike a standard RDF triple, the schema allows multiple subjects and objects in relation.

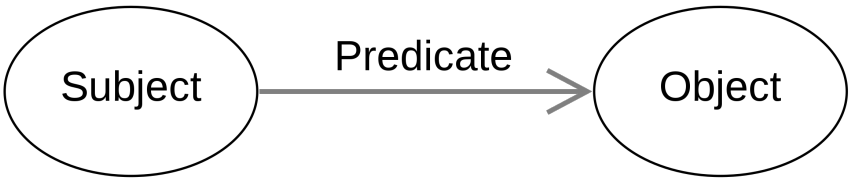


Figure 2.12: Basic subject-object-predicate triple.

Frames Frames represent moments in time. A frame can reference objects and store non-static information about them, for example, the bounding box parameters in the corresponding moment in time. Each frame may also store its own parameters like `timestamp`. The `frame_intervals` define which frames exist in the annotation file.

Ontologies Ontologies are used primarily to remove ambiguity from some values. For example, the paper mentions `Pedestrian` and `Person` as two possible types for the same object class, which could be easily confused. The `ontology_uid` key allows adding a pointer to an ontology where the type is defined, therefore making the type unambiguous.

Streams To handle data streams from multiple sources like cameras, LiDARs, and radars, the stream concept is available in the OpanLABEL schema. A stream may be defined globally or within a frame and can be assigned to an object to indicate that this object was sensed in this stream.

Coordinate Systems Data from diverse sources are represented in different coordinate systems. For example, camera images have a 2D coordinate system; LiDAR point clouds have a 3D one. Likewise, front-view captures have different coordinate systems from side-view data. Therefore, coordinate systems and transforms should be defined in an OpenLABEL annotation file as a mechanism to convert values between different coordinate systems. Each label should specify the corresponding coordinate system.

Further, the OpenLABEL standard goes into detail in explaining all the mentioned concepts. It also provides manuals on synchronization between different streams, declares data types and labels geometries, explains how to label an object with various label types, and defines the concept of a scenario. Explaining these would be out of the scope of this work, for details we recommend the paper [Cro].

Chapter 3

Solution

This chapter describes the contributions of this work to the proAnno labeling tool starting with the most significant advances, and compares alternative approaches for these features. The one-click annotation functionality is presented in details in Section 3.1. Further, Section 3.2 describes several object tracking/annotation transfer algorithms, comparing the Kalman filter with the moving average filter as well as different bounding box adjustment procedures. Section 3.3 depicts the use of HD maps for point cloud annotation. Keyboard navigation shortcuts and enhancements are presented in Section 3.4 and Section 3.5 describes the automatic saving feature.

3.1 One Click Annotation

Annotating a single object in a 3D point cloud requires setting nine parameters. Moreover, the desired values are not known upfront. Therefore, it is more than simply typing nine values in form fields. For example, to set the object's position, the user needs to look at the object from at least two perspectives and drag the object to the proper position. The same is true for scale and rotation. The purpose of the one-click annotation is, as the name tells, to allow the labeling of an object with a single click. The user selects a point in the space, and the cluster of points around this point is automatically annotated.

The following needs to happen to enable the desired behaviour:

1. Identify the points that belong to the object and create a box enclosing these

2. Predict the rotation of the object and adjust the box

3. Adjust the box scale and position to remove the margin between the box and points of the object

We compared the approaches for one-click annotation implemented in three of the most advanced labeling tools: LATTE, SAnE, and SUSTechPOINTS. While the first two have much in common, the latter implementation relies on a completely different approach. For proAnno, we decided to implement an algorithm similar to SUSTechPOINTS because it offers several advantages over LATTE and SAnE. First, it can be applied to more object classes. The L-Shape approach is suitable only for vehicles. Second, the chosen algorithm predicts seven degrees of freedom (DOF) compared to five in the alternative approach.

3.1.1 Determine Points of an Object

For one-click annotation, the first step is identifying all object points given only one point as input. Two different techniques are used by some of the most advanced open-source annotation tools for autonomous driving. Both LATTE and SAnE use DBSCAN [Est+96] based clustering algorithms to estimate the clicked object's top-view bounding box. This approach has some drawbacks if the data is sparse or noisy. To overcome these, SUSTechPOINTS uses box prototypes.

Box Prototypes

Clustering algorithms are prone to misclassification if the point cloud is too sparse. To overcome this numerical difficulty, the Portable Point-cloud Interactive Annotation Platform System (SUSTechPOINTS) [Li+20] determines the points which belong to the selected object using box prototypes. The human annotator can right-click on a position in the point cloud, and a context menu with object types will appear. Selecting an item will place an object with default scales for the selected object type at the mouse pointer's position. After that, a Euclidean space-based growing algorithm grows until there are no points in the point cloud that are not part of the selected object but are located within a threshold distance to any side of the annotation box.

3.1.2 Box Fitting

After assessing the belonging points, the next step is the bounding box fitting to the object's rotation, scale, and position. Both SAnE and LATTE implement the Efficient L-Shape Fitting for Vehicle Detection Using Laser Scanners [Zha+17] algorithm, but other methods can be plugged into LATTE as well; for example, methods based on the Principal Component Analysis (PCA). In the following, we will discuss the algorithm implemented in LATTE and the approach of SUSTechPOINTS, which separates rotation prediction from scale and position adjustment.

Rotation

The SUSTechPOINTS algorithm and the one we implemented predicts an object's direction separately from the scale and the position. When all the points belonging to the cluster are identified, they are fed to a pre-trained deep neural network that predicts the object's rotation. The DNN used in this operation is PointNet-based. Most objects in autonomous driving are placed parallel to the ground plane; in other words, the pitch and roll rotation is absent. Therefore, only the predicted yaw rotation is applied to the bounding box.

Scaling and Positioning

After estimating the vehicles heading direction, it is straightforward to compute its scale and adjust its position. First, we create a rotation matrix from the Euler angles as described in [Sla]:

$$R = R_z(\phi) \cdot R_y(\theta) \cdot R_x(\psi)$$
$$= \begin{bmatrix} cos\theta \cdot cos\phi & sin\psi \cdot sin\theta \cdot cos\phi - cos\psi \cdot sin\phi & cos\psi \cdot sin\theta \cdot cos\phi + sin\psi \cdot sin\phi \\ cos\theta \cdot sin\phi & sin\psi \cdot sin\theta \cdot cos\phi + cos\psi \cdot sin\phi & cos\psi \cdot sin\theta \cdot cos\phi - sin\psi \cdot sin\phi \\ -sin\theta & sin\psi \cdot cos\theta & cos\psi \cdot cos\theta \end{bmatrix}$$

Then, $R \cdot p,\ \forall p \in P$ will rotate the cluster so that its edges are parallel or orthogonal to the coordinate system base. The scale of the object on each axis is defined by the $max_{axis} - min_{axis},\ axis \in \{x, y, z\}$ where max_{axis} defines the maximum point projection value on the axis and min_{axis} defines the minimum point projection value.

The maxima and minima of a cluster are available as global coordinates. Therefore, the bounding box position along the x-axis is calculated as $x = \frac{max_x + min_x}{2}$. The formula applies for y and z as well.

3.2 Tracking

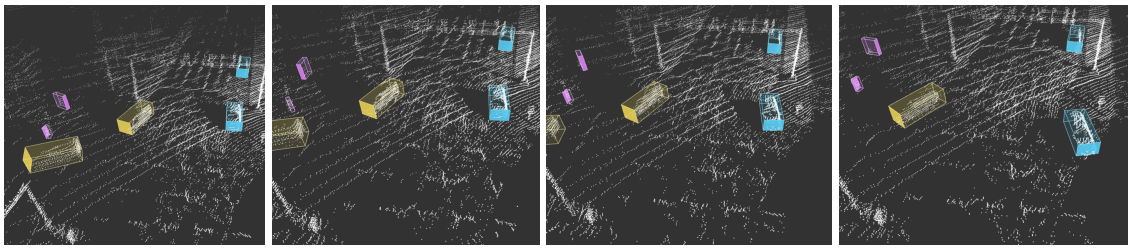


Figure 3.1: Four frames of a sequence demonstrating the result of tracking applied to various cars, vans and pedestrians. Three out of four frames are skipped in this figure for better visibility.
A BEV recording of the whole sequence: https://youtu.be/oE8-SDQf63U

Datasets in autonomous driving usually consist of traffic recordings with a frame rate of over 10 Hz [Gei+13; Sun+20; Cae+19; A9; Hou+20]. Hence, the label's position and direction do not change much between two consecutive frames. As a result, it is possible to track objects through whole data sequences. For example, a 20-second long recording with a frequency of 10 Hz contains each vehicle up to 200 times. Tracking allows annotating the object in one point cloud and transferring the bounding box to other frames while adjusting its position and rotation.

The annotation transfer to a neighboring frame in the SUSTechPOINTS and proAnno tools consists of the prediction and adjustment steps. First, the existing bounding boxes are fed into a moving average filter for predicting the location and rotation of the label in the next frame. After that, the annotation is automatically adjusted using a Euclidean growing algorithm to update the position and rotation. The LATTE application utilizes the Kalman[WB06] filter for predicting the object's pose in the next frame. A human annotator then adjusts the label position and rotation in the measurement step. The SAnE application implements a custom guided-tracking algorithm which was discussed in Section 2.3.4

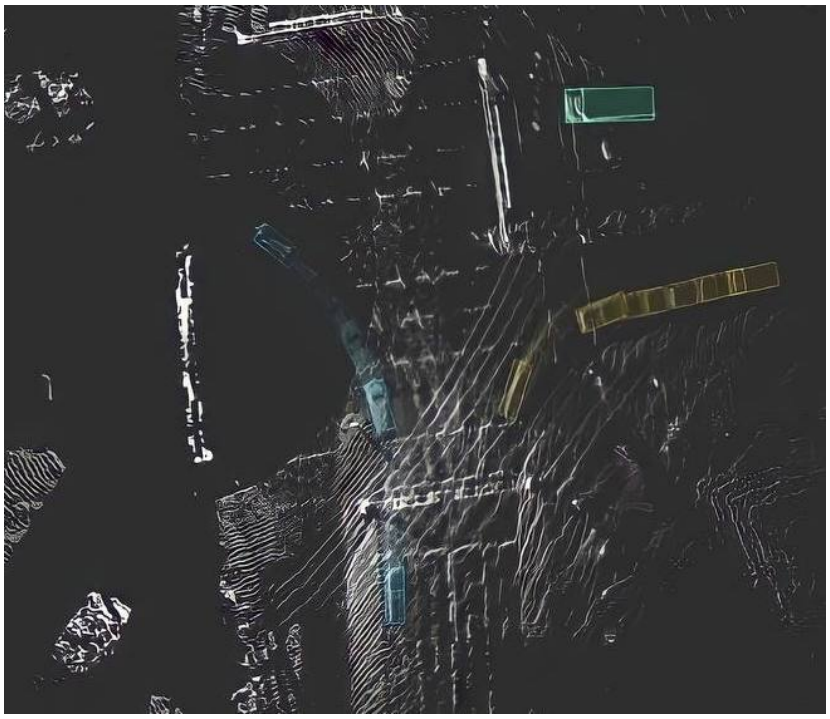


Figure 3.2: A series of birds-eye-view captures of frames of one sequence. Two blue cars, two yellow vans, one green truck and a pink pedestrian (bottom-right) were tracked in proAnno and can be found of this figure.

3.2.1 Moving Average Filter

The moving average filter is implemented in proAnno as part of this work. It can also be found in the SUSTechPOINTS tool. Like with the Kalman filter, the annotation transfer happens in two steps. First, the moving average filter is used to predict the object's position and rotation in the next frame. After that, the bounding box needs to be adjusted, and the moving average filter updates its state using the adjusted label.

Initially, the filter is initialized from a box and does not have any information about the velocity. In an update step, the velocity v_k for frame k of the vehicle gets updated to $v_k = d \cdot (x_{k-1} - \hat{x}_{k-1}) + (1-d) \cdot v_{k-1}$. Here, d defines the decay, x_{k-1} and $\hat{x}_{k-1}$ are the actual and predicted bounding boxes for frame $k-1$.

In our implementation, the bounding box adjustment is automated. Therefore, a label inserted in one frame can be transferred to all sequence frames in one click. For the auto adjustment, the Euclidean-space growing algorithm from one-click annotation is used to estimate the cluster's boundaries. However, since we assume rigid objects, only the position and rotation of this operation are applied.

3.3 HD Map

Autonomous driving point clouds are usually sparse. In most cases, these frames do not contain color information, making it difficult for a human annotator to navigate through the cloud and detect the boundaries of different objects. For example, differentiating a pedestrian from

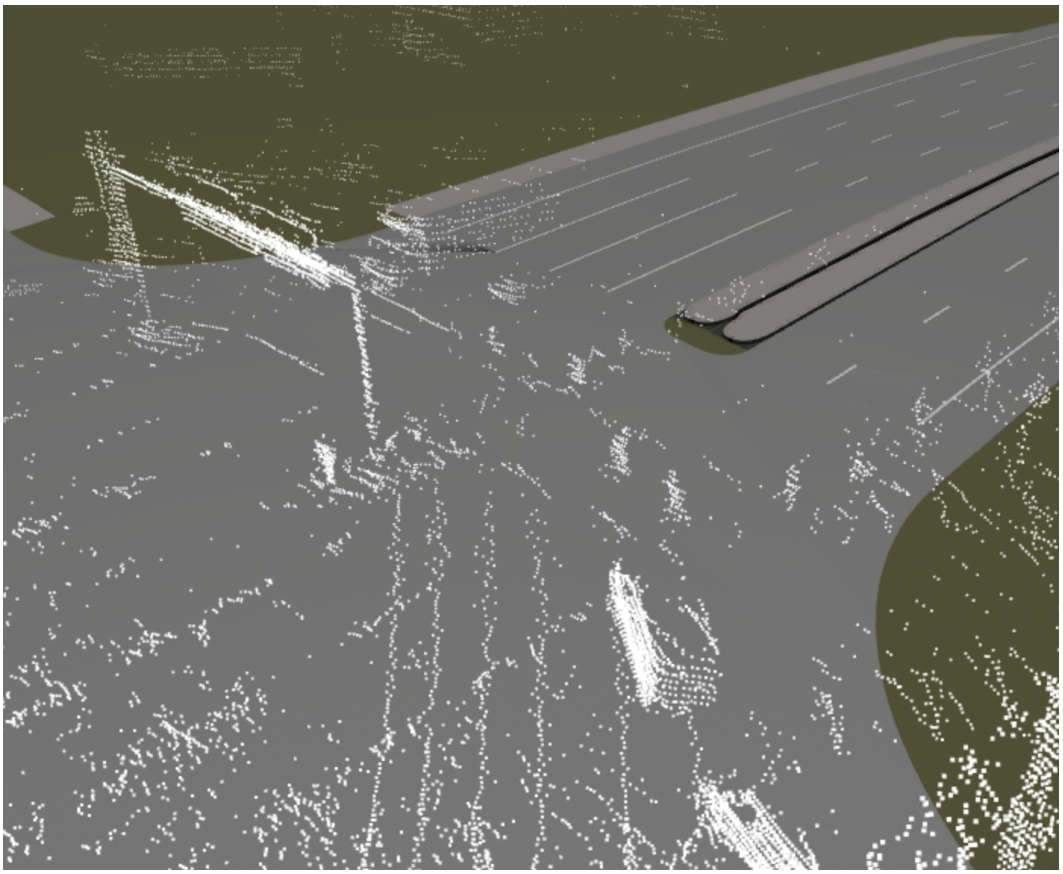


Figure 3.3: A pointcloud frame taken at the S110 measurement station on an Ouster OS1-64 LiDAR. Augmented with Providentia++ test stretch HD Map.

a road sign or a bicycle from a motorcycle is undoubtedly more straightforward if one can see whether the object is on or off the road. Therefore, displaying the street lanes, the boundaries of the road, and the environment's textures increases the accuracy and annotation speed. Therefore, we added an option to extend the point cloud with HD maps in proAnno. The user only needs to place the corresponding `.obj` and corresponding texture files in the folder with a channel. The HD map data needs to be per channel as this feature assumes static camera positioning.

3.4 Keyboard Navigation

Performing a massive annotation task which includes identifying clusters in the point cloud, navigating to the object, selecting the suitable class for the annotation, and adding a label, requires the human annotator to perform many manual steps. Although such automation techniques as one-click labeling, tracking, or object detection drastically reduce the workload, the worker still needs to navigate through frames and sequences to correct misclassified labels and inaccurate bounding boxes.

Several studies have shown that advanced computer users need less time choosing a command using short keyboard combinations than selecting the same command in the user interface using a mouse [Jor+02; KMA86; Lan+05]. However, a more recent study has shown that

this does not hold for two-level shortcuts [Oma+10]. One key in such a sequence determines a category of the selection, and the second key picks an item from this category. Nevertheless, more experienced users usually chunk longer key combinations and perform the command selection faster [Lan+93; MDO11].

Some keyboard shortcuts were already implemented in proAnno. For example, time navigation was possible with the N and P keys that switch to the next and previous frames accordingly. Toggling between the top orthographic view and perspective view is possible with the C key and interpolating annotation boxes with the I key. The annotator can select rotation, scaling, or translation mode when an object is selected by holding Ctrl and pressing R, S, or T, respectively.

It is in the scope of this work to enhance the keyboard commands and make it possible to use all the functionality of proAnno using keyboard controls only.

Tabbing Tabbing navigation allows the user to navigate through focusable elements using the TAB key. In the case of proAnno, the human annotator can switch the selected bounding box using this key. If no label is currently selected, the one with the lowest index will be selected. For iterating over the elements in reverse order, SHIFT + TAB should be used. When in focus mode, the object can be modified on nine degrees of freedom; therefore, three modes need to be separated: rotation, scaling, and translation. The SPACE button switches the current mode between these three. Finally, pressing the ESCAPE key when an object is in focus will remove the selection.

Bounding Box Manipulation Besides switching the mode using the SPACE bar, the desired mode can be selected using CRTL and R for rotation, S for scaling, or T for translation modes. Manipulations along the Y-axis are enabled with the W and S buttons. For example, pressing W when the translation mode is enabled will translate the object to higher Y values. In scaling mode, the object will increase in size along the Y-axis. Similarly, the A and D keys manipulate the bounding box along the X-axis and E and Q on the Z-axis.

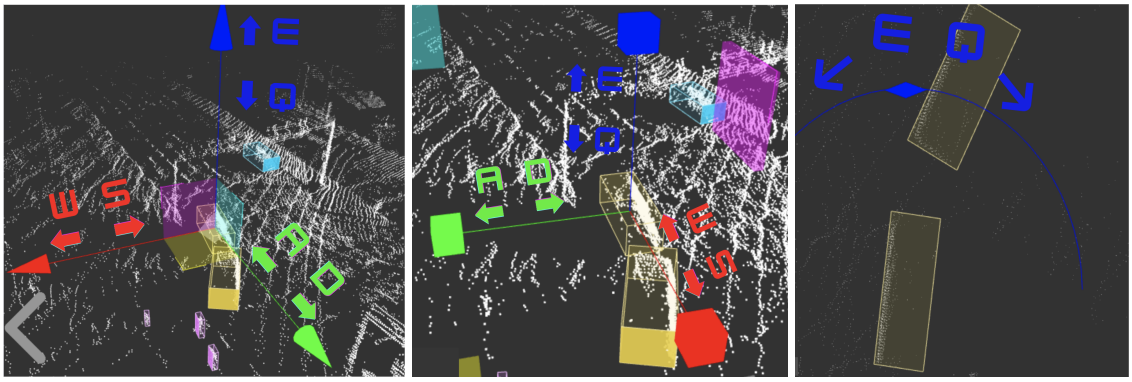


Figure 3.4: Shortcuts for object translation (left), scale (middle), and rotation (right)

Almost all objects in autonomous driving data are positioned parallel to the ground plane. Therefore, pitch and roll rotations are rarely used. Therefore, in the rotation mode, only the yaw rotation is available. As a result, the complexity of setting an annotation is reduced from nine to seven degrees of freedom.

Camera Translation and Rotation It is essential to make viewpoint adjustments straightforward. proAnno supports dollying in and out using the Q and E buttons, flying the camera with the WASD keys, and rotating the camera with arrow keys.

Further Shortcuts The exhaustive and up-to-date list of shortcuts can be found in the readme of the proAnno project on gitlab. Other commands include num pad navigation over annotation classes, undo and redo functionality with the standard key combination CTRL + Z and CTRL + SHIFT + Z, playing the sequence like a video using SPACE, and many others.

3.5 Automatic Saving

The former implementation of proAnno did not implement automatic saving. The user needed to download the annotations manually in a zip file to save the progress. Saving should have been done periodically because a sudden error would mean a loss of all the unsaved progress. As a result, after finishing annotating a sequence, the human annotator had dozens of downloaded zip files with various progress. Opening these downloaded annotations in proAnno required extracting the files into the correct subfolder in the input folder. All in all, the process of saving the current annotation state was far from straightforward, leading to a sharp learning curve and, therefore, a lot of lost time for inexperienced users. As part of this thesis, automatic saving was added to proAnno. A POST request with all annotations is periodically performed to the backend server. The server then updates the annotation files in the input folder. Not a single click is now required from the user to save current annotations.

Chapter 4

Evaluation

The two common issues when annotating train data by hand are the quality of labels and the time it takes to annotate a bunch of objects. Therefore, we evaluate the improvements in proAnno on both metrics. Besides simplifying the annotation process, we expect that one-click annotation achieves a higher or at least the same intersection over union (IoU) rate with the ground truth as manual annotation. On the other hand, the tracking feature speeds up the annotation procedure tremendously by automatically copying and adjusting the labels in consecutive frames. This chapter explains our conducted experiments and discusses the results regarding annotation speed and IoU.

Intersection over Union IoU measures the volume of intersection of two objects relative to the total volume of both objects. An IoU of 100% means the volume of overlap is the same size as the union of the volumes of both objects. Usually, applications try to maximize this metric. What is considered a good value depends on the application domain. For example, most object detection competitions classify a detection as a true positive at an IoU of 50% or 25%, while for some applications, these might be small values.

$$UoI = \frac{BB \cap GT}{BB + GT - BB \cap GT} \tag{4.1}$$

BB - Bounding Box GT - Ground Truth

4.1 Application-Level Comparison

The first experiment highlights the accuracy and annotation speed difference between distinct users with different configurations. It aims to understand whether the modifications of this work improve the accuracy and speed for novice annotators.

4.1.1 Experiment Setup

We took a sequence of point clouds consisting of one hundred frames for our experiment. With a frequency of 10 Hz, this totals to a time-frame of 10 seconds. The recording was taken on the Providentia++ S110 measurement station positioned above a crossing in an urban area. The point clouds capture two cars turning left and two vans turning right, three pedestrians crossing the road, one truck waiting on a red right, and another trailer passing the crossing

speedily. The turning cars present a complex labeling case because of their position and rotation change between consecutive frames. It is a hurdle when labeling the objects by hand and a trial for the tracking algorithm.

We created the ground truth annotations for the sequence. Further, we asked a newbie without prior experience annotating to annotate the same sequence. The novice was instructed about the newly available features and how they should be used. The baseline for our measurement is provided by Marcus Grabler, who manually created the labels for a half of this sequence using the proAnno version before our modifications, without one-click annotation, tracking, or keyboard controls.

4.1.2 Results

The Table 4.1 summarizes the outcomes. One can clearly observe the improvements in the labels' quality and the reduced time needed for annotation. While without improvements, a student annotated 50 frames in three hours, with improvements, the time was 2 hours and 21 minutes for 100 frames. The quality in terms of 3D IoU with the ground truth bounced from 31% to 58%. Further, it is important to mention here that the pre-modification version of proAnno already supported copying annotations from one frame to the next. This feature has already reduced the annotation time for sequences where the objects rarely changed their position between consecutive frames. In our sequence, a truck and a car waited at a red light and did not move for the whole duration of the sequence. Another car and two vans remained in their positions for 10 - 20 consecutive frames. This is a feasible explanation for the phenomenon that the annotation time difference is not as large as in our next experiment, where we compare the annotation time of a single object.

Annotator	Incl. changes	# of frames	Time	IoU
Expert	Yes	100	37 mins	Ground Truth
Novice	No	50	3 hours	31%
Novice	Yes	100	2 hr 21 mins	58%

Table 4.1: Compare the time needed to annotate a sequence of frames as well as the intersection over union (IoU) with the ground truth.

4.2 Feature-Level Comparison

The previous experiment was performed on a whole sequence, and the human annotators were not restricted to which functionality of proAnno they could use. It means the compound functionality of both versions of proAnno was compared. Although this might be one of the most reasonable metrics for comparing labeling applications, we are still interested in analyzing the impact of specific parts of this work on the annotation process. In the following experiments, we will inspect the benefits of the specific added functionalities.

4.2.1 One-Click Annotation

In this experiment, we want to understand the benefits of the new one-click annotation functionality. This feature directly impacts the annotation speed and accuracy for one single object.

Experiment Setup

For this experiment, we ask a novice and an expert to annotate only one frame containing nine objects. Both users must complete the task using the pre-modification version of proAnno and our implementation. We also computed the intersection over union (IoU) of the labels from expert and novice annotators.

Results

The Table 4.2 exposes that the average time it takes to annotate a single object with one-click annotation is 22 seconds for an expert and 31 seconds for a novice. This includes manually adjusting the bounding box after the one-click annotation engine places its prediction. Creating the same bounding box without the help of one-click annotation took, on average, 73 seconds for an expert and 113 seconds for a novice. Followingly, this feature speeds up the action by over three times. The measured IoU of the labels from the expert and the novice created with the enabled feature is 86%. Such a high overlap value can be explained by the fact that the labels are mostly set automatically with the same algorithm. The difference arises mainly from differing manual adjustments.

Annotator	without	with
Expert	73 sec	22 sec
Novice	113 sec	31 sec

Table 4.2: Difference in per-object annotation time for expert and novice users. The annotation time without one-click annotation feature can be seen in the second column, with the one-click annotation in the third column.

4.2.2 Tracking and Keyboard Navigation

Another feature we want to evaluate is tracking. This feature automates transferring annotations between frames by predicting the object's position in the next frame using a moving average filter and using the same box-growing algorithm used in one-click annotation to adjust the position of the prediction. Moreover, the same PointNet-based neural net as in the one-click annotation is used to adjust the predicted bounding box rotation.

Tracking performance may differ greatly for each object depending on the object's size, other nearby objects, the object's trajectory, and other parameters. Followingly, any experiment on a small sample would not yield significant results. In the worst case, when tracking fails completely, the tracking algorithm copies the object between frames but incorrectly predicts its rotation and position. Such cases can be compared to manually copying the label between frames and adjusting its position and rotation. Thus, in the worst case, there is no

improvement over the pre-modification proAnno version. Nevertheless, we performed an experiment to evaluate annotation speed gain from labeling in a best-case scenario. Here we also compare the keyboard controls with mouse-only controls.

Experiment Setup

We selected a car in a 100-frame sequence. The car does not move in the first 20 frames, starts driving, turns left on a crossing, and disappears from the scene in the last ten frames. We applied tracking on this car and annotated the same vehicle manually in all frames using the copy-to-next-frame feature.

We annotated the car in the whole sequence with the new tracking function and in 50 frames with the manual approach. The later requires the manual annotator to select an object, click the copy-to-next-frame button, change the frame, and adjust the position and rotation of the object. The bounding box adjustment is by far the most expensive step. The first 20 frames can be completed reasonably quickly as the car does not move, and the most expensive step of the iteration can be skipped. In the successive 30 frames, a slight rotation on one axis and position adjustments on two axes were needed.

The manual part was performed two times. In the first iteration, we did not use the keyboard controls. Each adjustment was performed using a trackpad. In the next iteration, we used keyboard shortcuts. These iterations highlight the impact of keyboard controls.

Results

The results are summarized in Table 4.3. Automated tracking took 17 seconds to label the car in all 100 frames. Then, the annotator took another 96 seconds to iterate over the sequence, verify that the labels were set accurately, and delete the bounding boxes in the last ten frames. Copying the objects to the next frame and manually adjusting the positions using the keyboard took almost 9 minutes. Without the keyboard controls, this time increased to over 12 minutes. Thus, it is four times more expensive to annotate a vehicle in this specific example without tracking. Without keyboard controls, it is another 1.5 times more expensive.

Annotation transfer approech	# of frames	time
Automated with tracking	100	1 min 53 sec
Manual without keyboard	50	8 min 46 sec
Manual with keyboard	50	12 min 14 sec

Table 4.3: Difference in annotation time using tracking feature, keyboard controls and mouse only.

4.3 Conclusion

While it is difficult to quantify the improvements to proAnno precisely, this chapter has shown that the implemented features significantly positively impact both annotation speed and accuracy. Particularly, the bounding box fitting algorithm in the one-click annotation function improved the IoU with the ground truth almost two-fold from 31% to 58%. Furthermore, in our best-case example, the tracking algorithm made transferring annotations to consecutive

frames up to four times faster than the manual copy-paste-adjust approach with keyboard controls and up to six times faster than the manual approach with trackpad-only controls. A bigger study would be required to empirically estimate the impacts of other features like HD maps, RGB point clouds, and automatic saving.

Chapter 5

Future Work

This work provides automation for some most frequent processes in point cloud annotation in proAnno. Nevertheless, there is plenty of space for improvement. The central improvement axes are accuracy, annotation speed, and application usability. This chapter discusses several concrete improvement ideas that were not implemented in the scope of this work. Section 5.1 discusses the advantages of making proAnno compatible with the forementioned OpenLABEL standard. The next section 5.2 discusses different possibilities of detecting objects in a frame, automating the labeling process even further. Possible improvements in tracking is examined in the section 5.4. Last but not least, section 5.5 lists desirable smaller functions and shortcuts that would improve usability and flatten the learning curve for getting started with proAnno.

5.1 OpenLABEL Compatibility

As described in section 2.4, a common standard for all labeling tools will simplify the cooperation between different applications and reduce the error probability in autonomous driving systems, therefore improving the safety on roads. Currently, proAnno defines a custom annotation file schema based on the 3D BAT format [ZRT19]. It uses the JSON file format and overlaps with the OpenLABEL schema at various points. Therefore, it is undoubtedly critical for proAnno to be conforming to industry standards. Compatibility with the annotation file standard can be achieved by switching entirely to the OpenLABEL annotation file format or implementing a bidirectional converter for proAnno schema and OpenLABEL.

5.2 Object Detection

The implemented one-click functionality enables annotating each object with only one click. Further, tracking allows transferring the annotation to all frames of the sequence. Therefore, a human annotator only needs to annotate an object with a single click in any point cloud. Although these features provide a significant speedup compared to the manual annotation of each object and compete with the most advanced 3D point cloud annotation applications, the labeling process can be automated even further with the help of object detectors. Ultimately, a zero-click annotation could be achieved.

An object detector finds objects in a point cloud and predicts their class, position, rotation, and size. Object detection is an extensive topic with various existing approaches and imple-

mentations of detectors. However, most publicly available pre-labeled autonomous vehicle datasets are recorded from the top of a driving vehicle. Therefore, object detectors might need to be trained on a different, road-side dataset to perform well in proAnno. Luckily, such research is performed in the scope of the Providentia++ project, and the results could be helpful in the further development of proAnno. In this section, we discuss single-stage, two-stage and unsupervised 3D object detectors and compare them for the use case of proAnno.

5.2.1 Single-Stage

A single-stage object predictor consists of a neural network that generates features directly from the given point cloud and one or two detection heads for classification and bounding box prediction. This is the preferred approach for real-time object detection as single-stage solutions usually outperform two-stage implementations in terms of speed. VoxelNet [ZT17], SECOND [YML18], PointPillars[Lan+18], SA-SSD [He+20] and 3DSSD [Yan+20] are some prominent single-stage trainable object-detection methods.

5.2.2 Two-Stage

Two-stage 3D object detectors are fed with raw point clouds as well. They perform the detection in two steps. In the beginning, the Region of Interest (RoI) is generated in the first step. These regions of interest are represented as bounding boxes. Then, the points within these boxes are fed into the second stage network, which generates features for each specific RoI. Finally, the features from both stages are combined to predict each object's class and bounding box. The benefit of the two-stage approach is the improved accuracy over the single-stage methods. Some example methods are PV-RCNN [Shi+19], PV-RCVV++ [Shi+21] and BtcDet [XZN21]

The DASE-ProPillars [Gra22] object detector by Marcus Grabler stands as a possible candidate for an object detector on proAnno. It improves on the SE-ProPillars, the work of Jialong Wu, which is based on ProPillars [Fer+21] and PointPillars [Lan+18]. DASE-ProPillars is a Domain Adaptive single-stage 3D object detector in LiDAR point clouds. The model is trained on road-side autonomous driving frames from the A9 Dataset [A9], the synthetic datasets proSynthLidar, proSemiSynthLidar, D16 and A11 from the Providentia++ and the Regensburg Next projects. The Table 5.1 demonstrates the mean average precision (mAP) of the DASE-ProPillars method when detecting objects of class Car. It also compares the results with the predecessor SE-ProPillars.

IoU threshold	0.5	0.25
SE-ProPillars	7.03	97.07
DASE-ProPillars	42.3	92.81

Table 5.1: 3D mAP of the class car under 0.5 and 0.25 IoU threshold, with 40 recall positions for the class car. [Zha+17]

5.2.3 Unsupervised

An alternative approach is unsupervised object detection. In the scope of the Providentia++ project, Marcel Brucker is working on implementing an unsupervised object detector. It has several advantages over a supervised solution. First of all, it does not require training. Furthermore, it does not need to be re-trained if used on a different data type. As we discussed previously, switching between road-side and vehicle-side recordings might be problematic with a model trained on only one kind of data. Another advantage is the performance. The current implementation can detect objects at around 10 Hz. The downturn, however, is that the object's heading needs to be adjusted to make the predictions more accurate. Nevertheless, despite the rotation errors, the overall detection quality seems proper for our objective of using the detected bounding boxes as input for the one-click annotation algorithm, which will fit the rotation and dimensions of the object.

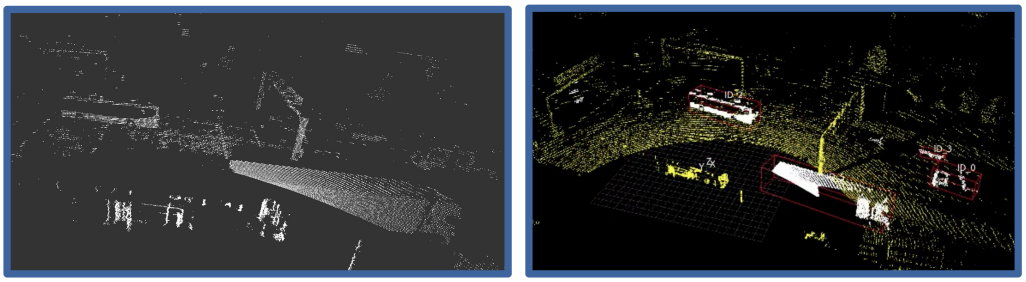


Figure 5.1: Multiple objects detected by the unsupervised algorithm.

The bounding boxes predicted by 3D object detectors usually have lower accuracy than those generated with one-click annotation. Therefore, the predicted bounding boxes should be refined using the box fitting algorithm from the section 3.1. Ultimately, a whole sequence of point clouds could be annotated with a single click and reasonable accuracy. The human annotator would only be needed for refining the labels, if at all.

Figure 5.2: Unsupervised object detection pipeline [Mar22]

5.3 Automatic Class Matching

Besides finding the object, determining its rotation, and fitting the box, we want the application to determine the class of the object it just found automatically. The problem can be solved in a plethora of ways. One of them is a classical classification problem with a neural net. The model should receive box dimensions and be able to classify the object. A more straightforward approach is matching the class to prototype boxes of different classes and choosing the best match. Regardless of the approach, this feature is essential for further automation.

5.4 Tracking

Object tracking usually consists of a prediction, and a correction step, as section 3.2 explains. In this work, we implemented the moving average filter for the prediction step and adjusted the bounding box with a Euclidean-space-based growing algorithm. However, advanced tracking applications, for example, aircraft radars, mainly rely on other methods. Therefore, there are several alternatives to the moving average filter. Kalman filter is one of the most popular ones for tracking. For instance, the tracking algorithm in LATTE implements the Kalman filter for tracking bounding boxes in point cloud sequences. This is because it is more adaptive to an object's trajectory and velocity changes. However, despite its popularity, the Kalman filter is not state-of-the-art anymore. A more advanced alternative is the Generalized Labeled Multi-Bernoulli Filter. The latter is also used in the Providentia++ live system.

Improving the predictions would ultimately improve the tracking quality in proAnno. Therefore, replacing the moving average filter with a more suitable approach will significantly improve the tool's accuracy and labeling automation.

5.5 Miscellaneous

More minor adjustments together drastically enhance the useability. Finally, some ideas for future improvements are listed in this section.

Multiselection When having a bunch of bounding boxes per frame, it might be helpful to have the possibility to perform some operations on multiple objects simultaneously. For example, the world shifts slightly between frames if the researcher did the data recording from a moving subject like a car. In such cases, copying all labels to a consecutive frame and translating the position of the bounding in bulk is very handy.

Further Keyboard Shortcuts This work covered the vast majority of all functionality of proAnno with keyboard commands. Nevertheless, there are still some unimplemented features that would require a shortcut—for example, performing a redo operation, copying and pasting with `CTRL+C/CTRL+V`, navigating through the menu and changing settings, updating the point size, adding a label with a key combination, and many more.

Copy and Paste Interpolation requires setting several control point bounding boxes in various sequence parts. Tracking can also use control points to adjust the prediction accuracy of the filter. Currently, it is possible to copy a bounding box to a consecutive frame. The copy and paste functionality should be expanded to allow copying any point cloud in the sequence.

Chapter 6

Acknowledgments

This thesis would not have been possible without support from different people and their assistance. Foremost, I would like to thank my advisor Walter Zimmer for introducing me to the topics and for guidance throughout the whole duration of my work at the Chair of Robotics, Artificial Intelligence and Real-time Systems. I also greatly appreciate collaborating with my colleagues Marcus Grabler, Marcel Brucker and Jialong Wu. They kindly introduced me to the field of object detection and helped me understand their work. Many thanks to all the authors of the techniques used in this work for bringing a fabulous imagination of autonomous vehicles closer to reality. Last but not least, thanks to my family and friends for supporting and encouraging me to try always to achieve the best results.

List of Figures

List of Tables

Bibliography

[A9] A9. URL: https://innovation-mobility.com/a9-dataset/ (visited on 05/04/2022).

[Ari+20] Arief, H. A., Arief, M., Zhang, G., Liu, Z., Bhat, M., Indahl, U. G., Tveite, H., and Zhao, D. “SAnE: Smart Annotation and Evaluation Tools for Point Cloud Data”. In: *IEEE Access* 8 (2020), pp. 131848–131858. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3009914.

[BTM06] Bai, S., Thibault, J., and McLean, D. D. “Dynamic data reconciliation: Alternative to Kalman filter”. In: *Journal of Process Control* 16.5 (2006), pp. 485–498. ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2005.08.002. URL: https://www.sciencedirect.com/science/article/pii/S095915240500096X.

[Beh+19] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. *SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences*. 2019. DOI: 10.48550/ARXIV.1904.01416. URL: https://arxiv.org/abs/1904.01416.

[BMD] BMDV. URL: https://www.bmvi.de/SharedDocs/EN/Articles/DG/act-on-autonomous-driving.html (visited on 05/10/2022).

[Cae+19] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. *nuScenes: A multimodal dataset for autonomous driving*. 2019. DOI: 10.48550/ARXIV.1903.11027. URL: https://arxiv.org/abs/1903.11027.

[Cro] Croce, N. URL: https://www.asam.net/project-detail/asam-openlabel-v100/ (visited on 05/13/2022).

[Dat] Datasets, O. *Open Datasets*. URL: https://scale.com/open-datasets.

[Est+96] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[Fer+21] Fernandes, D., Afonso, T., Girão, P., Gonzalez, D., Silva, A., Névoa, R., Novais, P., Monteiro, J., and Melo-Pinto, P. “Real-Time 3D Object Detection and SLAM Fusion in a Low-Cost LiDAR Test Vehicle Setup”. In: *Sensors* 21.24 (2021). ISSN: 1424-8220. DOI: 10.3390/s21248381. URL: https://www.mdpi.com/1424-8220/21/24/8381.

[Gei+13] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[Gey+20] Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A. S., Hauswald, L., Pham, V. H., Mühlegg, M., Dorn, S., Fernandez, T., Jänicke, M., Mirashi, S., Savani, C., Sturm, M., Vorobiov, O., Oelker, M., Garreis, S., and Schuberth, P. "A2D2: Audi Autonomous Driving Dataset". In: *CoRR* abs/2004.06320 (2020). arXiv: 2004.06320. URL: https://arxiv.org/abs/2004.06320.

[Gra22] Grabler, M. "Real-Time and Robust 3D Object Detection within Multi LiDAR Systems on the Autonomous Driving Test Stretch Using Cross-Sensor Domain Adaptation". PhD thesis. unpublished thesis, 2022.

[Gri+19] Grigorescu, S., Trasnea, B., Cocias, T., and Macesanu, G. "A survey of deep learning techniques for autonomous driving". In: *Journal of Field Robotics* 37 (Nov. 2019). DOI: 10.1002/rob.21918.

[has] hasanari. *SAnE Github Repository*. URL: https://github.com/hasanari/sane.

[He+20] He, C., Zeng, H., Huang, J., Hua, X.-S., and Zhang, L. "Structure Aware Single-Stage 3D Object Detection From Point Cloud". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11870–11879. DOI: 10.1109/CVPR42600.2020.01189.

[He+17] He, K., Gkioxari, G., Dollár, P., and Girshick, R. *Mask R-CNN*. 2017. DOI: 10.48550/ARXIV.1703.06870. URL: https://arxiv.org/abs/1703.06870.

[Hou+20] Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. 2020. DOI: 10.48550/ARXIV.2006.14480. URL: https://arxiv.org/abs/2006.14480.

[Hua+18] Huang, X., Cheng, X., Geng, Q., Cao, B., Zhou, D., Wang, P., Lin, Y., and Yang, R. "The ApolloScape Dataset for Autonomous Driving". In: *CoRR* abs/1803.06184 (2018). arXiv: 1803.06184. URL: http://arxiv.org/abs/1803.06184.

[Jor+02] Jorgensen, A. H., Garde, A. H., Laursen, B., and Jensen, B. R. "Using mouse and keyboard under time pressure: Preference, strategies and learning". In: *Behaviour & Information Technology* 21.5 (2002), pp. 317–319. DOI: 10.1080/01449290021000048448. eprint: https://doi.org/10.1080/01449290021000048448. URL: https://doi.org/10.1080/01449290021000048448.

[KMA86] Karat, J., McDonald, J. E., and Anderson, M. "A Comparison of Menu Selection Techniques: Touch Panel, Mouse and Keyboard". In: *Int. J. Man-Mach. Stud.* 25.1 (July 1986), pp. 73–88. ISSN: 0020-7373. DOI: 10.1016/S0020-7373(86)80034-7. URL: https://doi.org/10.1016/S0020-7373(86)80034-7.

[Lan+93] Lane, D. M., Napier, H. A., Batsell, R. R., and Naman, J. L. "Predicting the Skilled Use of Hierarchical Menus With the Keystroke-Level Model". In: *Human–Computer Interaction* 8.2 (1993), pp. 185–192. DOI: 10.1207/s15327051hci0802_4. eprint: https://www.tandfonline.com/doi/pdf/10.1207/s15327051hci0802_4. URL: https://www.tandfonline.com/doi/abs/10.1207/s15327051hci0802_4.

[Lan+05] Lane, D., Napier, H., Peres, S., and Sandor, A. "Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts". In: *Int. J. Hum. Comput. Interaction* 18 (May 2005), pp. 133–144. DOI: 10.1207/s15327590ijhc1802_1.

[Lan+18] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. *PointPillars: Fast Encoders for Object Detection from Point Clouds*. 2018. DOI: 10.48550/ARXIV.1812.05784. URL: https://arxiv.org/abs/1812.05784.

[Li+20] Li, E., Wang, S., Li, C., Li, D., Wu, X., and Hao, Q. "SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1108–1115. DOI: 10.1109/IV47402.2020.9304562.

[LXG21a] Liao, Y., Xie, J., and Geiger, A. *KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D*. 2021. DOI: 10.48550/ARXIV.2109.13410. URL: https://arxiv.org/abs/2109.13410.

[LXG21b] Liao, Y., Xie, J., and Geiger, A. "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D". In: *arXiv.org* 2109.13410 (2021).

[Lin+14] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

[Mar22] Marcel, B. "Unsupervised LiDAR-based 3D Object Detection". MA thesis. TUM, 2022, p. 159.

[MDO11] Miller, C. S., Denkov, S., and Omanson, R. C. "Categorization Costs for Hierarchical Keyboard Commands". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 2765–2768. ISBN: 9781450302289. DOI: 10.1145/1978942.1979351. URL: https://doi.org/10.1145/1978942.1979351.

[NSO21] Nieto, M., Senderos, O., and Otaegui, O. "Boosting AI applications: Labeling format for complex datasets". In: *SoftwareX* 13 (2021), p. 100653. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2020.100653. URL: https://www.sciencedirect.com/science/article/pii/S2352711020303666.

[Oma+10] Omanson, R. C., Miller, C. S., Young, E., and Schwantes, D. "Comparison of Mouse and Keyboard Efficiency". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 54.6 (2010), pp. 600–604. DOI: 10.1177/154193121005400612. eprint: https://doi.org/10.1177/154193121005400612. URL: https://doi.org/10.1177/154193121005400612.

[Pha+19a] Pham, Q.-H., Sevestre, P., Pahwa, R. S., Zhan, H., Pang, C. H., Chen, Y., Mustafa, A., Chandrasekhar, V., and Lin, J. *A*3D Dataset: Towards Autonomous Driving in Challenging Environments*. 2019. DOI: 10.48550/ARXIV.1909.07541. URL: https://arxiv.org/abs/1909.07541.

[Pha+19b] Pham, Q.-H., Sevestre, P., Pahwa, R., Zhan, H., Pang, C., Chen, Y., Mustafa, A., Chandrasekhar, V., and Lin, J. "A*3D Dataset: Towards Autonomous Driving in Challenging Environments". In: (Sept. 2019).

[Red+15] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. DOI: 10.48550/ARXIV.1506.02640. URL: https://arxiv.org/abs/1506.02640.

[Shi+19] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., and Li, H. *PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection*. 2019. DOI: 10.48550/ARXIV.1912.13192. URL: https://arxiv.org/abs/1912.13192.

[Shi+21] Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., and Li, H. *PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection*. 2021. DOI: 10.48550/ARXIV.2102.00463. URL: https://arxiv.org/abs/2102.00463.

[Sla] Slabaugh, G. G. *Computing Euler angles from a rotation matrix*.

[Sun+20] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[SUS] SUSTech. *Southern University of Science and Technology Webpage*. URL: https://www.sustech.edu.cn/en/.

[Wan+19] Wang, B., Wu, V., Wu, B., and Keutzer, K. "LATTE: Accelerating LiDAR Point Cloud Annotation via Sensor Fusion, One-Click Annotation, and Tracking". In: *arXiv preprint arXiv:1904.09085* (2019).

[WB95] Welch, G. and Bishop, G. *An Introduction to the Kalman Filter*. Tech. rep. USA, 1995.

[WB06] Welch, G. and Bishop, G. "An Introduction to the Kalman Filter". In: *Proc. Siggraph Course* 8 (Jan. 2006).

[Wil+21] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., Ramanan, D., Carr, P., and Hays, J. "Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*. 2021.

[XZN21] Xu, Q., Zhong, Y., and Neumann, U. "Behind the Curtain: Learning Occluded Shapes for 3D Object Detection". In: (2021). DOI: 10.48550/ARXIV.2112.02205. URL: https://arxiv.org/abs/2112.02205.

[YML18] Yan, Y., Mao, Y., and Li, B. "SECOND: Sparsely Embedded Convolutional Detection". In: *Sensors* 18.10 (2018). ISSN: 1424-8220. DOI: 10.3390/s18103337. URL: https://www.mdpi.com/1424-8220/18/10/3337.

[Yan+16a] Yang, J., Li, H., Campbell, D., and Jia, Y. "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11 (Nov. 2016), pp. 2241–2254. DOI: 10.1109/tpami.2015.2513405. URL: https://doi.org/10.1109%2Ftpami.2015.2513405.

[Yan+16b] Yang, J., Li, H., Campbell, D., and Jia, Y. "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11 (Nov. 2016), pp. 2241–2254. DOI: 10.1109/tpami.2015.2513405. URL: https://doi.org/10.1109%2Ftpami.2015.2513405.

[Yan+20] Yang, Z., Sun, Y., Liu, S., and Jia, J. *3DSSD: Point-based 3D Single Stage Object Detector*. 2020. DOI: 10.48550/ARXIV.2002.10187. URL: https://arxiv.org/abs/2002.10187.

[Yel] Yellepeddi, A. *Seeing Farther with LIDAR Using Tracking*. URL: https://www.analog.com/media/en/technical-documentation/tech-articles/seeing-farther-with-lidar-using-tracking.pdf.

[Yur+20] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. “A Survey of Autonomous Driving: Common Practices and Emerging Technologies”. In: *IEEE Access* 8 (2020), pp. 58443–58469. DOI: 10.1109/ACCESS.2020.2983149.

[Zha+17] Zhang, X., Xu, W., Dong, C., and Dolan, J. M. “Efficient L-shape fitting for vehicle detection using laser scanners”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 54–59. DOI: 10.1109/IVS.2017.7995698.

[ZT17] Zhou, Y. and Tuzel, O. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. 2017. DOI: 10.48550/ARXIV.1711.06396. URL: https://arxiv.org/abs/1711.06396.

[ZRT19] Zimmer, W., Rangesh, A., and Trivedi, M. M. “3D BAT: A Semi-Automatic, Web-based 3D Annotation Toolbox for Full-Surround, Multi-Modal Data Streams”. In: *CoRR* abs/1905.00525 (2019). arXiv: 1905.00525. URL: http://arxiv.org/abs/1905.00525.