2020 IEEE Intelligent Vehicles Symposium (IV)
October 20-23, 2020. Las Vegas, USA

SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System

E Li[1,4], Shuaijun Wang[1,2], Chengyang Li[1], Dachuan Li[1,4],Xiangbin Wu[3], and Qi Hao[1,4*]

***Abstract*— The major challenges of developing 3D point cloud annotation systems for autonomous driving datasets include convenient user-data interfaces, efficient operations on geometric data units, and scalable annotation tools. This paper presents a Portable pOint-cloud Interactive aNnotation plaTform System (*i.e*. SUSTech POINTS), which contains a set of user-friendly interfaces and efficient annotation tools to help achieve high-quality data annotations with high efficiency. The novelty of this work is threefold: (1) developing a set of visualization modules for fast annotation error localization and convenient annotator-data interactions; (2) developing a set of interactive tools for annotators labeling 3D point clouds and 2D images in high speed; (3) developing an annotation transfer method to label the same objects in different data frames. The developed POINTS system is tested with public datasets such as KITTI and a private dataset (SUSTech SCAPES). The experimental results show that the developed platform can help improve the annotation accuracy and efficiency compared with using other open-source annotation platforms.**

I. Introduction

Building accurately annotated autonomous driving (AD) datasets is critical for training and verification of situation perception and data fusion algorithms. While 3D points clouds can provide high-accuracy geometric relations among the ego-vehicle, surrounding objects and the environment, 2D images contain high-resolution information of objects and the environment. Recent AD datasets take advantages of high-resolution, multi-perspective 2D data to improve the annotation quality of 3D data based on their underlying correlations. There are three major components for intelligent AD dataset annotation systems: (1) data pre-processing, (2) data visualizations, and (3) interactive operations, which are enabled by a data management system and a user-data interaction interface, as shown in Fig. 1.

Despite many efforts and successes in developing AI-based pre-processing techniques, data visualizations and interactive operations also impose a number of technical challenges upon developing an efficient data annotation platform, including

This work is partially supported by the National Natural Science Foundation of China (No: 61773197), the Science and Technology Innovation Committee of Shenzhen City (No: GJHZ20170314114424152), the Nanshan District Science and Technology Innovation Bureau (No: LHTD20170007), and the Intel ICRI-IACV Research Fund (CG#52514373).

*Corresponding author: Qi Hao (hao.q@sustech.edu.cn).

[1]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong, China, 518055

[2]Harbin Institute of Technology, 92 West Dazhi Street, Nan Gang District, Harbin, China, 150001

[3]Intel Collaborative Research Institute on Intelligent and Automated Connected Vehicles

[4]Sifakis Research Institute of Trustworthy Autonomous Systems

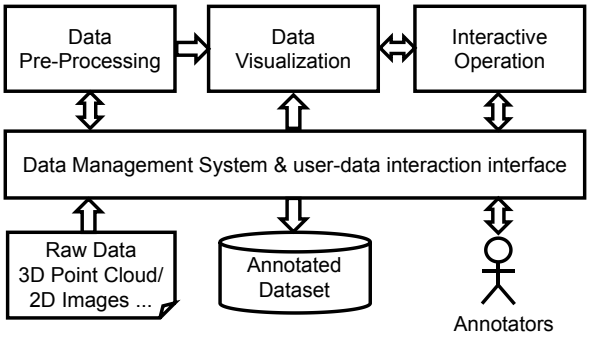


Fig. 1: An illustration of intelligent annotation systems for autonomous driving datasets.

1) **Fast Error Checking**. Given the initial annotation results of 2D and 3D data, it is required to develop a set of data visualization modules that can help annotators quickly localize those annotation errors.
2) **Easy user-data interactions**. Each frame of data usually contains a great number of objects with irregular measurement data in a large field of view (FOV), hence it is necessary to develop a convenient human-computer interface to enable annotators to easily perform corrective operations upon annotation errors.
3) **Flexible annotation extension**. There are a lot underlying correlations among data frames, FOVs, and sensing modalities, so the annotations of the same objects should be scalable and extensible among different frames, areas and modalities.

Current AD dataset annotation systems have developed temporal and spatial navigation tools to help quickly review annotation results [2]–[4], [6], assigning different colors to annotation boxes to distinguish objects [2], [4]–[6], providing various perspective/projective views and multi-camera photo contexts of 3D data to help localize annotation errors [2], [4], [6], and supporting data stream playing and object locking in data streams to help check out the temporal consistency among annotation results [6]. However, more visualization modules such as background removal, object based coloring, and modality/spatial/temporal consistency checking should be further developed to improve the error checking efficiency. On the other hand, interactive operations such as high degrees-of-freedom (DOFs) and multi-view editing, box initialization (one-click annotation) and automatic fitting have also been developed [1]–[4], [6]; but all need further refinement to increase annotation efficiency. Besides, several annotation transfer methods have been developed to achieve

TABLE I: A comparison of 3D point cloud annotation systems in terms of visualization modules and interactive operations

		Visualization									Operation				
	Open Source	3D Spatial/ Temporal Navigation	Main View Focus Mode	Box & Object Coloring	Top/Side/ Front Views	Photo Context with 3D-2D Fusion	Focused Context	Multi-Camera Switching	Stream Play	Object Locking	Box Degrees of Freedom	Auto Box Initialization	Sub-View Editing	Interactive Box Fitting	Annotation Transfer
PointAtMe [1]	√	√	×	×	×	√*	×	×	×	×	9	×	×	×	×
3D BAT [2]	√	√	×	√*	√*	√	×	√*	×	×	9	×	×	×	++
LATTE [3]	√	√	×	×	×	√*	×	×	×	×	5	++	×	×	++
Supervise.ly [4]	×	√	×	-	√	√	×	×	-	-	9	×	++	×	-
scale.ai [5]	×	√	×	√*	√	√	×	√*	-	-	7	-	-	×	-
Playment.io [6]	×	√	×	√*	√	√	×	×	√*	√	-	×	++	×	++
Ours (POINTS)	√	√	√	√	√	√	√	√	√	√	9	+++	+++	+++	+++

The symbol "√" means that the system supports the functionality, "×" means does not support, "√*" means partially supports, and "-" means unknown. The number of "+"s represents the performance of a specific functionality, more "+"s means better performance. Some functionalities are explained as below:

1) "Main View Focus Mode": to zoom in and center a selected object in the main view with one click.
2) "Stream Play": to play the sequential 3D data as a video stream
3) "Object Locking": to automatically select the same object in all frames when the stream is playing.
4) "Focused Context": to display the 2D image of the current object as a photo context in a single sub-view
5) "Camera Auto-Switching": to automatically switch the photo context among multiple cameras when navigating objects.
6) "Auto Box Initialization": to automatically fit the 3D annotation box to the whole 3D object data with one click on one point of the object data.
7) "Sub-View Editing": to edit the 3D annotation box in one of the projective sub-views.
8) "Interactive Box Fitting": to automatically fit the 3D annotation box to the 3D object data when editing (e.g. rotating, translating, and resizing).
9) "Annotation Transfer": to transfer annotations across frames by an automatic or semi-automatic way.

The meaning of other functionalities could be understood by their own names.

annotation extensions to multiple frames [2], [3], [6], but no registration-based method has been proposed yet, which can provide more accuracy and higher speeds for annotations.

In this paper, we present a comprehensive web-based AD dataset annotation platform system, which provides more advanced functionalities in visualization modules, interactive tools, and annotation transfer. The contributions of this work include

1) developing an open-source portable 3D point cloud annotation platform with well-designed visualization modules and high-efficiency interactive tools, whose source codes are available for public access on Github (https://github.com/naurril/SUSTechPOINTS).
2) developing a series of enhanced data visualization modules such as stream playing, object locking, background removal, object based coloring, multi-camera swiching, and photo context adjusting to enable fast error checking.
3) developing a series of new interactive tools such as smart box initialization and interactive box fitting to enable fast and accurate 3D data annotating.
4) developing a novel registration-based inter-frame annotation transfer method, which can extend annotation results from one frame to the whole data stream with high accuracy and speeds.

The rest of this paper is organized as follows. Section II reviews the state-of-the-art AD dataset annotation systems and highlights the novelty of our system. Section III describes the system architecture and major functionalities of the proposed POINTS system. Section IV provides the experiment results and related discussions. Section V concludes the paper and outlines future work.

II. Related work

Table I summarizes a detailed comparison of the most popular 3D point clouds annotation systems in terms of visualization modules and interactive operations, where the first 3 systems (PointAtMe [1], 3D BAT [2], LATTE [3]) and ours (POINTS) are open-source software, and the remaining 3 systems (Supervise.ly [4], scale.ai [5], Playment.io [6]) are commercial platforms. It can be seen that in general the commercial platforms can provide advanced functionalities of data visualizations and interactive operations, but their technological details are undisclosed. By comparison, our open-source annotation platform (POINTS) can provide unique functionalities of main view focus mode, focused context and interactive box fitting, plus more advanced functionalities of stream playing, object locking, sub-view editing, auto box initialization and annotation transfer.

3D point cloud annotation systems can be classified as screen based and virtual-reality (VR) based. In the screen based systems, point clouds are visualized on 2D computer screens; users annotate objects by drawing 3D boxes with keyboards, mouse devices and/or gestures (if touch screens are available) [7]; camera images are displayed to help identify objects; various algorithms are developed to assist interactive operations [3]. VR based systems can provide better immersive experiences for users [1], [8], but suffer from limits such as inaccurate operations and motion sickness. Therefore, in this work we choose a screen based interface to develop our annotation system.

Annotation systems can also be classified as web-based and PC-based. The web-based systems, such as 3D-BAT [2] and supervise.ly [4], are portable for users to perform 3D object annotations only using web browsers. The PC-based systems, such as Apollo Suite [9], need software installed to perform annotations. As a result, these systems need to be updated from time to time and annotation datasets need to be downloaded from servers. Our proposed POINTS system

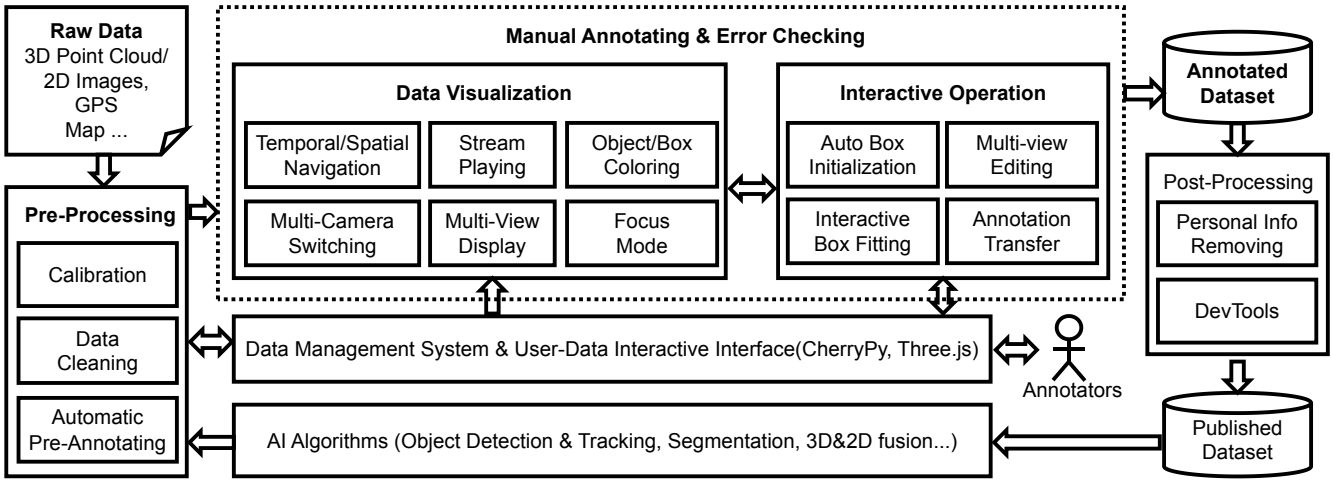


Fig. 2: The system architecture of the proposed portable point cloud interactive annotation platform system (POINTS).

is web-based, enabled by the CherryPy [10] web application development framework. We believe that web-based systems are more portable and scalable for massive data annotating when many annotators are involved.

There are no standard benchmark metrics for annotation system evaluation yet, to the best of our knowledge. Annotation efficiency can be measured by the annotation time used by annotators [1], [2], [11]. PointAtMe evaluates the annotation accuracy by using labor-intensive annotation results as the ground truth [1]. In this work, we choose the metrics used by PointAtMe [1] to evaluate the efficiency and accuracy of our developed tools, and use its experiment results as the baseline to compare with.

III. System Architecture and Module Functionalities

A. System Architecture

The web-based POINTS focuses on developing visualization modules and interactive tools for 3D bounding box and tracking ID annotation. The platform web server manages all the data, programmed with the CherryPy [10] web application development framework. The platform frontend provides almost all the module functionalities, programmed with the `WebGL` library Three.js [12]. Fig. 2 shows the system architecture of POINTS. In the data pre-processing stage, intrinsic/extrinsic sensor parameters are estimated first; only those data frames consistent to calibration parameters, useful for algorithms training and verification are selected; then optionally a set of AI algorithms including detection, tracking, and 2D&3D fusion are used to generate initial annotation results. The data visualization modules provide helpful temporal/spatial navigation tools for users, distinguish data of different objects, and show multiple views of each object from different perspectives. The interactive data operations include editing labels in multiple views, automatically adjusting the size of 2D/3D boxes to fit the data of each object, and automatically extending (transferring) annotations of the same objects from one frame to different frames.

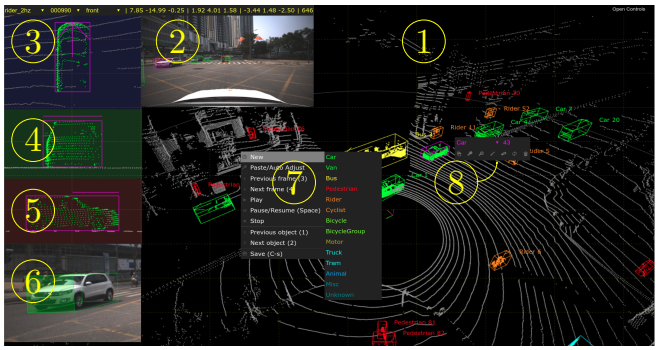
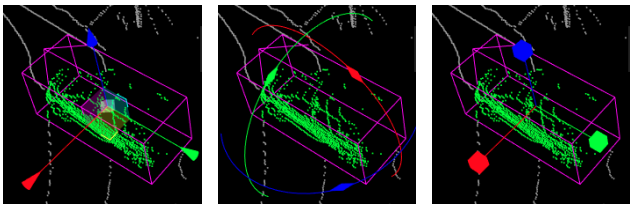


Fig. 3: The main UI of POINTS. ① the perspective view of the 3D point cloud; ② the photo context, which is resizable and can be automatically switched among multiple camera images; ③ the top view of the selected object; ④ the front view of the selected object; ⑤ the side view of the selected object; ⑥ the focused photo context, which can be automatically chosen for the selected object ; ⑦ the context menu, which provides tools of context operations; ⑧ the floating fast toolbox, which provides most used tools.

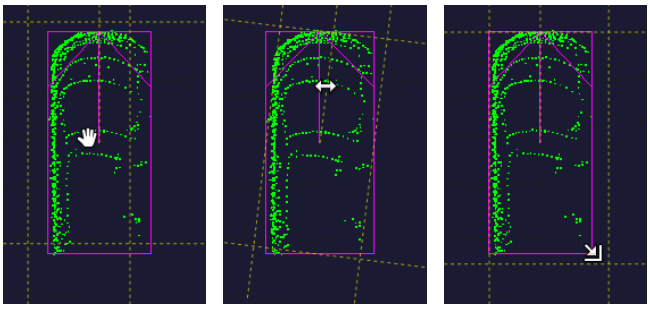
B. Data Visualization Modules

Data visualization modules help edit and review the annotation results in terms of 3D boxes. Fig. 3 shows the main user interface (UI) of POINTS, from which all the visualization functionalities can be achieved. The main UI contains the following functionalities: **Sub-views**. The main window is divided into one main view (① in Fig. 3) and five sub-views, and the top, front and side projective views of the selected object are displayed on the left side (③,④,⑤ in Fig. 3)with the background hidden. The context photo of the selected object is displayed in the top middle sub-view (② in Fig. 3) and an extra focused context is displayed in the bottom left sub-view (⑥ in Fig. 3); the 3D annotation



(a) Translation (b) Rotation (c) Scaling

Fig. 4: An illustration of annotation box operations in the perspective view.



(a) Translation (b) Rotation (c) Scaling

Fig. 5: An illustration of annotation box operations in the top projective view. Dotted lines suggest the next box pose after each operation.

box of the selected object is projected onto these two sub-views. **Focus-mode**. A focus mode is provided to help check the details of the selected object. When the mode is activated through the fast toolbox (⑧) in Fig. 3), the selected object will be automatically centered in the main view and zoomed in with most of the background hidden. **Camera auto-switching**. Multiple camera images can all be displayed on the top of the main view [2], or only one of them displayed to provide more focus [1], [3], [5], which requires automatic switching to the most relevant camera for a selected object. Manual camera selection should also be provided. Note that this functionality needs accurate calibration parameters of sensors. **Object coloring**. All boxes and points can be colored by their categories. A selected object will be highlighted with a different color. **Navigation**. A user can quickly check the annotation results in the 3D space, or along the time axis, by changing the point of view, selected objects and the no. of frames. **Stream play**. A user can play the sequential data in terms of scenes like a video. **Object locking**. When a data stream is playing, a selected object can be locked, highlighted with a specific color, in all frames. **Box information**. The detailed information (scene, frame, object category, coordinates, dimension, orientation and the number of points) of each selected box will be displayed on the top of the main window.

C. Interactive Operations: 3D Box Tools

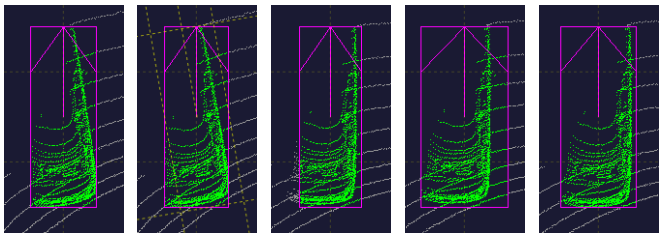
3D box initialization. A user can create a new 3D box by right clicking on an object and then select the object type from the pop-up context menu. Our platform performs the following functionalities to help create a box easily: 1) The initial box orientation is upward (*i.e* the degree of yaw along the z-axis is zero) within the main view of the x-y plane. 2) A box prototype (in accordance with the object type) is placed at the position of the mouse pointer. 3) A Euclidean distance based growing algorithm is invoked to estimate all the points belonging to the object. 4) An auto-fitting algorithm (Section III-D) is used to shrink the box size in case there is a margin between the box and the points of the object.

Note that in most cases, the annotation can be completed with such a single click. Different from the one-click-annotation of LATTE [3], which uses a clustering algorithm to search the points belonging to the object under selection, our method uses box prototypes to overcome numerical difficulties when the points of an object is too sparse for clustering algorithms. A user can also create a new box by drawing a 2D rectangle within the main view, enclosing relevant points. The box is initialized by setting its orientation as above and automatically fitted to the enclosed points. The object type is automatically chosen by simply matching the dimension of the object with those of prototypes.

3D box editing in the perspective view. When an object is selected, a floating fast toolbox will appear next to it, as shown in Fig. 3. A user can change the object type, tracking id, activate the focus mode, among others. For tracking ID, a user can input or select existing value in the fast toolbox, or select 'auto' option letting the platform to generate a unique new ID in the scene. A user can click on an object or the button of the fast toolbox to enable box editing, and drag the handlers to resize, rotate or translate the box, as shown in Fig. 4. Corresponding keyboard shortcuts are also provided. **3D box editing in the projective sub-views**. Editing a 3D box in the perspective view is inconvenient as the user has to change the viewpoint from time to time. As shown in Fig. 3, the annotation results are more clearly displayed in the projective sub-views, hence our platform enables the user to rotate, resize or translate the box in the projective sub-views by using mouse operations, as shown in Fig. 5. All operations can also be done with keyboard shortcuts.

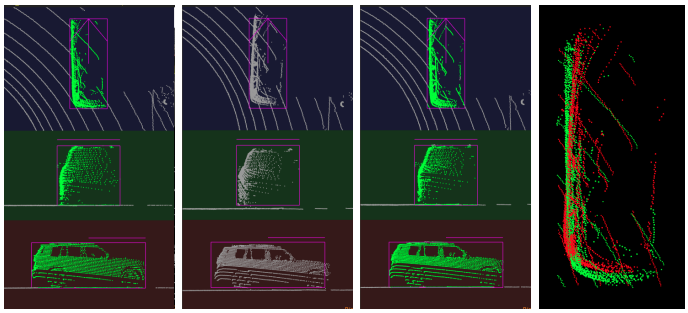
D. Interactive Operations: Supporting Algorithms

Interactive box fitting. When a bounding box is resized or rotated, our platform can fit the box to the object automatically by finding the minimal box enclosing all the points of the object. Fig. 6 illustrates the process of box auto-fitting after rotation; as the result, all the points of the car are kept inside the bounding box. With this functionality, only two steps are needed to annotate a new object: (1) draw a rectangle to enclose all the points of an object; (2) rotate the 3D bounding box once. Note that the algorithm to find the minimal box may fail when the points of ground are present. For example, in Fig. 6d, because the ground points (circular arcs) are present, the auto-fitted box becomes a little bit larger than the object. One possible solution is to remove the ground points using algorithms, but it is not reliable when the ground is not smooth (as often can be seen in



(a) (b) (c) (d) (e)

Fig. 6: An illustration of box auto-fitting after rotation. (a) A bounding box enclosing all points of a car. (b) A counter-clockwise rotation of the bounding box. (c) Part of the points of the car going outside the bounding box without auto-fitting. (d) Auto-fitting of the bounding box to the points of the car, where the final bounding box becomes a little bit larger than the car because of the presence of ground points (circular arcs). (e) Auto-fitting of the bounding box to the points of the car, with the ground plane problem solved by our algorithm, the bounding box fits the object well.



(a) (b) (c) (d)

Fig. 7: An illustration of registration based annotation transfer. (a) The reference object and its bounding box. (b) The object in another frame with an initial bounding box. (c) The adjusted bounding box using our transfer algorithm. (d) The input of the registration algorithm consisting of two set of point clouds from (a) and (b).

the KITTI dataset [13] where cars are often parked across the curb (Fig. 8)). We use a simple but effective method to solve this problem. Specifically, a large part of the object bottom (0.2m along the z-axis by default) is skipped while the object size along x-y dimensions (width and length) are accurately estimated, such that the ground points can be removed effectively. In real-world traffic scenarios, almost all objects are much higher than 0.2m. After such a object size estimation, the reduced 0.2m will be compensated along the z-dimension (height). The Interactive Box Fitting algorithm is summarized in Algorithm 1.

Annotation transfer among frames. AD datasets are often organized in terms of data streams [14]–[16]. Each stream consists of a sequence of data frames. There are many similarities between neighboring frames, which makes it possible to transfer annotations among them. Annotation results of one frame could be directly copied to the next frame directly and then adjustments are performed by users

Algorithm 1: Interactive Box Fitting Algorithm.

Input: point cloud $\mathcal{P} \in \mathbb{R}^{n\times 3}$; original box $b = (p, s, r)$; operation $o = (r', s')$, where $p, s, r \in \mathbb{R}^3$ means position, scale and rotation, respectively, rotation is represented by Euler angles, r', s' means rotating angles and scale changes of the operation o.

Output: box b' after operation.

find all points $\mathcal{R}$ inside box b;
construct coordinate system B' with original point p and axes angles $r + r'$;
construct range c by applying s' on s, represented as x, y, z coordinate ranges;
compute $\mathcal{P}'$ by representing $\mathcal{P}$ in coordinate system B';
initialize $\mathcal{R}' = \emptyset$;
for *point* x *in* $\mathcal{P}'$ **do**
  if x *is in* $\mathcal{R}$ **then**
    add x into $\mathcal{R}'$
  end
  else if x *is in range* c **then**
    add x into $\mathcal{R}'$
  end
end
compute coordinates range d of $\mathcal{R}'$ by finding maximum and minimum values along each coordinate axis. when computing x, y ranges, ignore points whose z value is in the lowest 0.2m part;
compute position delta δp and scale s'' of new box b' in coordinate system B' by range d;
compute $\delta p'$ by representing δp in original coordinate system of $\mathcal{P}$
return $b' = (p + \delta p', s'', r + r')$;

[2]. Those box adjusting algorithms can use the object sizes estimated from previous frames [3]. Our system utilizes a 3D data registration algorithm [17] to automatically adjust box poses using the sizes and orientations of objects acquired from previous frames, which can produce accurate annotation results in most cases.

The automatic annotation transfer needs a 3D object tracking algorithm to build the correspondence between reference objects and target objects from different frames. The registration algorithm is then used to compute the relative geometric transform between the reference and target objects, and the bounding boxes in the target frame will be adjusted accordingly. If there is no 3D object tracking algorithm, a user needs to first select a reference box in the source frame, and copy & paste it at an appropriate position (overlapping with the target object) in the target frame. The relative geometric transform between the reference and target objects will be computed and the bounding box in the target frame will be adjusted accordingly. The reference and target objects are cropped out and all their points are transformed to the corresponding bounding box coordinates systems automati-

cally before feeding them into the registration algorithm.

Fig. 7 illustrates an example of annotation transfer, whose performance relies on the effectiveness of the registration algorithm. The registration works well when the orientations of reference and target objects have small deviations.

Given the reference and target objects (denoted by $\mathcal{R}$ and $\mathcal{T}$, respectively) and their bounding boxes (b_r; b_t), a registration algorithm aims to find the transform T to match $\mathcal{T}$ with $\mathcal{R}$, minimizing a distance function $dist(T(\mathcal{T}); \mathcal{R})$. For brevity we derive the formula for cases where only rotation is needed (*i.e.* zero translation).

We denote the bounding box coordinate systems of $\mathcal{R}$, $\mathcal{T}$ and the world coordinate system by their corresponding basis matrices B_r, B_t and B_w. Note that the box is identical to the box coordinates system (*i.e.* b_r is identical to B_r) since we only concern the orientation (corresponds to 3 axes) and position (corresponds to the origin point) of the box.

Assume a point $t \in \mathcal{T}$ corresponds to a point $r \in \mathcal{R}$ after the registration, we will have

$$Tt^{B_t} = r^{B_r} \tag{1}$$

$$TB_t^{-1}t^{B_w} = r^{B_r} \tag{2}$$

where r^{B_r} means a point r represented in the B_r coordinate system. If the new coordinate system of $\mathcal{T}$ is denoted as B'_t, then we will have

$$t^{B'_t} = r^{B_r} \tag{3}$$

$$(B'_t)^{-1}t^{B_w} = r^{B_r} \tag{4}$$

From Eq. (2) and Eq. (4), we have

$$B'_t = (TB_t^{-1})^{-1} \tag{5}$$

$$= B_tT^{-1} \tag{6}$$

The annotation transfer algorithm and an object crop and transform algorithm are summarized in Algorithm 2 and Algorithm 3.

Algorithm 2: Annotation Transfer Algorithm.

Input: reference and target point cloud $\mathcal{P}_r, \mathcal{P}_t \in \mathbb{R}^{n\times 3}$; reference box $b_r = (p_r, s_r, r_r)$; initial position p_t of target box, where $p, s, r \in \mathbb{R}^3$ means position, scale and rotation, respectively, rotation is represented by Euler angles;

Output: target box b_t

initialize target box $b_t = (p_t, s_r, r_r)$;
compute points of reference object $\mathcal{R}$ represented in coordinate system B_r by calling Algorithm. 3 with input $\mathcal{P}_r$ and b_r;
compute points of target object $\mathcal{T}$ represented in coordinate system B_t by calling Algorithm. 3 with input $\mathcal{P}_t$ and b_t;
compute relative rotation ρ and translate τ by registering $\mathcal{T}$ to $\mathcal{R}$ with registration algorithm [17] ;
compute τ' by transform τ from box coordinate system B_t into world coordinate system B_w;
by using Eq. (6), compute $b_t = (p_t - \tau', s_r, r_r - \rho)$;
return b_t;

Algorithm 3: Object Crop and Transform Algorithm.

Input: point cloud $\mathcal{P} \in \mathbb{R}^{n\times 3}$; box $b = (p, s, r)$ where $p, s, r \in \mathbb{R}^3$ means position, scale and rotation, respectively, rotation is represented by Euler angles.

Output: set of points $R \in \mathbb{R}^{m\times 3}$ in box b, represented in b's box coordinate system.

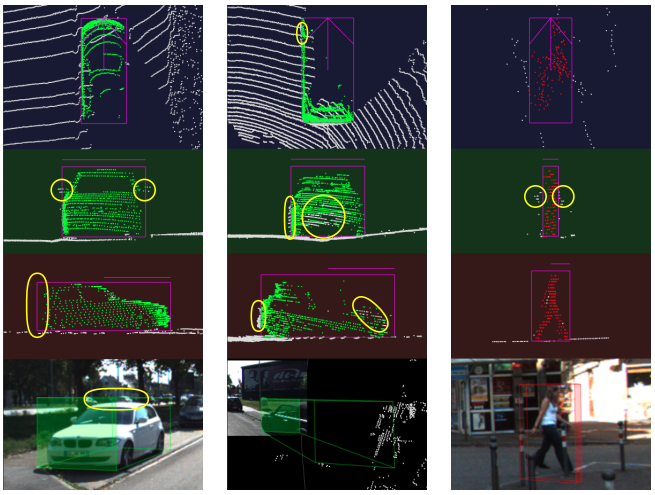
initialize $R = \emptyset$;
for *point* x *in* $\mathcal{P}$ **do**
    compute x' by representing x in b's coordinate system;
    if x' *lies inside box* b **then**
        append x' into R
    end
end
return R;

IV. Results And Discussions

Generally speaking, it is hard to evaluate an 3D point cloud annotation platform as there are no standard benchmark metrics. In this work, we use KITTI [13] and SUSTech SCAPES to test the developed POINTS. The whole evaluation contains three parts: 1) Annotation error checking efficiency. We use the KITTI dataset [13] to test POINTS's new visualization functionalities which help users to check the annotation errors. 2) Annotation efficiency. We compare POINTS and PointAtMe [1] in the terms of annotation errors and speeds. 3) Annotation transfer quality. We use four target frames for annotation transfer from one reference frames with three types of objects: car, rider, and bus.

A. Annotation Error Checking Efficiency

Examples from the KITTI 3D Object Detection dataset [13] are used to demonstrate the effectiveness and efficiency of POINTS in checking annotation errors. The same examples are chosen as shown in Fig. 5 of [1]. Although our platform supports all 9 degrees of freedom (DOFs) for 3D bounding boxes, only 7 DOFs (without pitch and tilt) are labeled in the KITTI dataset. Therefore, we only test those examples with 7 DOFs. Fig. 8 shows examples of annotation errors checking with the KITTI 3D object detection dataset using POINTS. It contains the top, front, side views and photo context of two cars and a pedestrian. It can be seen that with the help of the 4 sub-views and object coloring (SectionIII-B) annotation errors, such as front mirrors, hands, part of cars outside the bounding boxes and the margins between bounding boxes and objects can be quickly identified by just selecting the objects from one to another (Section III-B). Meanwhile, tracking IDs can also be easily checked with the help of the functionalities of object locking and stream playing (Section III-B).



(a) (b) (c)

Fig. 8: An illustration of annotation errors checking with the KITTI 3D object detection dataset using POINTS. (a) The top, front, side views and photo context of a car, where two front mirrors are out of the bounding box, and the bounding box has back and top margins over the car. (b) The top, front, side views and photo context of a car, where one front mirror, the left side and the back part of the car are out of the bounding box. (c) The top, front, side views and photo context of a pedestrian, where the two hands are out of the bounding box.

B. Annotation Efficiency

Annotation efficiency can be measured by annotation accuracy and speed [1] [2]. POINTS aims to maximize the annotation accuracy within the minimum time. As shown in the previous section (IV-A) and also indicated in [1] (Table I), some of the original KITTI labels are not suited as ground truth to compare our annotations with, following [1] we carefully annotated some KITTI scenes and use the results as the ground truth. Three users are employed to annotate the same set of data after a few hours of practicing. For each user we measure the average annotation time (seconds) per object and average annotation errors per object. For a perfect annotation, all the points belonging to an object should be inside the bounding box and otherwise outside. Three metrics are used: 1) the number of error points per object, 2) the relative number of False Positive (FP) points per object, and 3) the relative number of False Negative (FN) points per object.

As shown in Table II, the annotated accuracy obtained by those unexperienced users using POINTS are very close to the ground truth, much higher than using PointAtMe (baseline) with less than half of the annotation time. The success can be attributed to the developed techniques such as smart box initialization, interactive box fitting and convenient visualization modules. Users only need to ensure the points of one object are all inside one 2D rectangle initially, and do not need to refine the 3D bounding box by themselves, reducing many human errors. Meanwhile, the annotation speed is also improved by more than two times compared to the baseline. It can be seen that POINTS can really help unexperienced users to achieve high annotation efficiency.

TABLE II: A comparison of annotation efficiency between POINTS and PointAtMe using the KITTI dataset.

Method	Ours (POINTS)	PointAtME [1]
Time/Object (sec.)	24 ± 6	55.2 ± 12.1
Errors (Points/Object)	<1	28.7 ± 5.1
FP ratio / Object	0	7.16%
FN ratio / Object	$<1\%$	3.27%

C. Annotation Transfer Quality

TABLE III: Annotation transfer results among five frames for three scenarios using POINTS with the SUSTech SCAPES dataset.

#	Type	Ref. Obj.	0.5s	1s	1.5s	2.5s	Err/Obj	FN Ratio
1	Car	-/1958	15/1387	25/935	19/677	1/389	15	1.7%
2	Rider	-/329	0/240	0/190	0/149	0/94	0	0
3	Bus	-/2327	5/1208	3/656	1/396	0/173	2.25	0.28%

The value is in the form of Error points/Total points. "Ref. Obj." means the reference object (*i.e.* the manually annotated object). The "Err/Obj" and "FN" are the same as in Table II.

Our private dataset (SUSTech SCAPES) is used to evaluate the annotation transfer quality among five frames at 0s, 0.5s, 1s, 1.5s, and 2.5s. We choose three scenarios with different object types: 1) scenario 1: a car moving away from the ego-car along a straight line while the ego-car waiting for the red light, 2) scenario 2: a rider crossing the road while the ego-car approaching to the rider, and 3) scenario 3: a bus and the ego-car both moving forward on the same road with reverse directions. The frame at 0s is chosen as the reference frame, and hence reference objects have the largest number of points. Table III shows that the distance between the ego-car and the target object increases as the time goes by, because the numbers of 3D data points of those objects decrease accordingly. For example, in scenario 3, the bus only has 173 data points in the frame of 2.5s, 7.4% of the reference object's 2327 data points in the frame of 0s. Elaborate annotation results of those scenarios are chosen as the ground truth. The Error Points Per Object and the relative FN points number per object are used to evaluate the annotation transfer quality.

The results of transfer annotation are shown in Table III and Fig. 9. It can be seen that for scenario 1, the annotation transfer errors of the car are less than 2.7%, with the relative number of FN points of 1.7%; for scenario 2, the annotation transfer errors of the rider are 0, with the relative number of FN points of 0%; for scenario 3, the annotation transfer errors of the bus are less than 0.5%, with the relative number of FN points of 0.28%. Compared with Table II, it can be seen that the accuracy of transfer annotation is lower than manual annotations, but still higher than baseline (although we used a different dataset). With the help of such a annotation transfer technique, each object in a data stream needs only almost once manual annotation at the beginning.

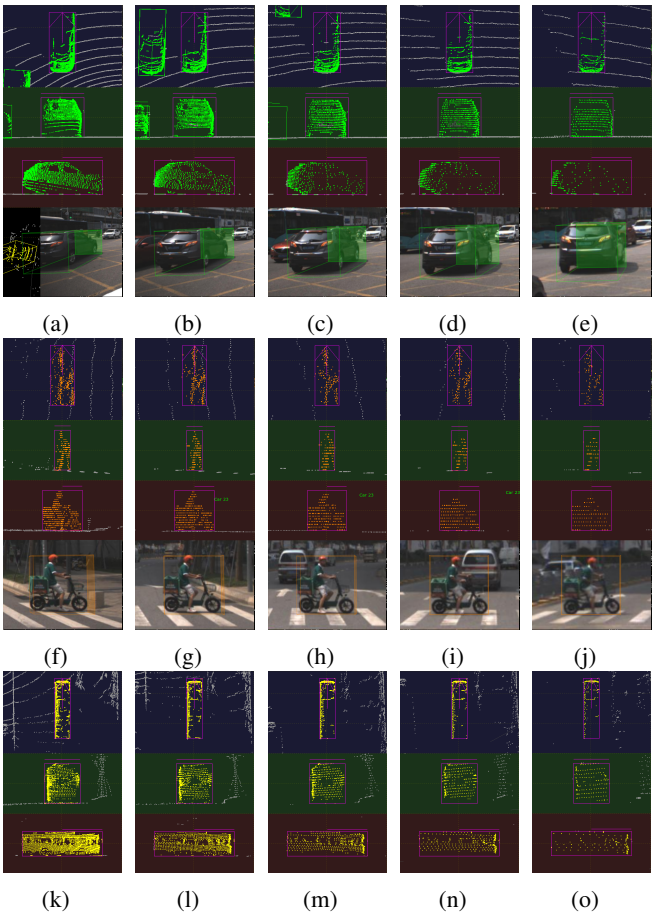


Fig. 9: Annotation transfer results among five frames for three scenarios using POINTS with the SUSTech SCAPES dataset. The three rows correspond to three scenarios. The left column (a,f,k) are references frames whose bounding boxes are annotated manually, the remaining columns are frames of 0.5s, 1s, 1.5s and 2.5s, respectively, whose bounding boxes are generated by the annotation transfer algorithm of POINTS. For scenarios 1 and 2 (a-j), the top, front, side views and photo context are shown, for scenario 3 (k-o) only top, front, side views are shown. In scenario 1 (a-e), the back part (b-d) and the front mirror (e) of the car are out of the bounding box. In scenario 2 (f-j), there is no errors. In scenario 3 (k-o), the back part (l, m) and the front mirror (m, n) of the bus are out of the bounding box.

Such a technique can also help solve the problem of distant or occluded objects when the number of data points are very sparse. We believe that this technique is complementary to other existing annotation transfer algorithms, such as direct pasting and linear interpolation, and if integrated with tracking algorithms POINTS could improve the annotation efficiency further.

V. CONCLUSION

In this paper, we have presented a framework for developing portable 3D point cloud iterative annotation platform systems (POINTS) for AD applications. A number of important data visualization modules and interactive operation tools, as well as supporting algorithms, have been specifically introduced. The major contributions of this work include developing a series of enhanced data visualization modules, new interactive tools, and a registration-based annotation transfer method to achieve high-accuracy and high-efficiency human-in-the-loop 3D data annotation. The source codes of the developed SUSTech POINTS are available on Github for public access. Compared with PointAtMe, the proposed POINTS can achieve much lower annotation errors in shorter annotation time, tested with KITTI and SUSTech SCAPES datasets. The developed annotation transfer method can transfer annotations from one frame to multiple frames with errors of no more than 3%. The future work includes integrating more AI-based algorithms, such as 3D object tracking algorithms, with our annotation platform (POINTS) to achieve more automatic annotation functionalities with high quality.

REFERENCES

[1] F. Wirth, J. Quchl, J. Ota, and C. Stiller, "PointAtMe: Efficient 3D Point Cloud Labeling in Virtual Reality," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1693–1698.

[2] W. Zimmer, A. Rangesh, and M. M. Trivedi, "3D BAT: A Semi-Automatic, Web-based 3D Annotation Toolbox for Full-Surround, Multi-Modal Data Streams," *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1816–1821, 2019.

[3] B. Wang, V. Wu, B. Wu, and K. Keutzer, "LATTE: Accelerating LiDAR Point Cloud Annotation via Sensor Fusion, One-Click Annotation, and Tracking," *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 265–272, 2019.

[4] SUPERVISELY. [Online]. https://supervise.ly/.

[5] SCALE. [Online]. https://scale.com/sensor-fusion-cuboids.

[6] Playment. [Online]. https://playment.io/.

[7] C. S. MIT and A. I. Laboratory. LabelMe3D. [Online]. http://labelme.csail.mit.edu/Release3.0/.

[8] M. Veit and A. Capobianco, "Go'then'tag: A 3-d point cloud annotation technique," in *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, March 2014, pp. 193–194.

[9] P. Wang, X. Huang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The apolloscape open dataset for autonomous driving and its application," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[10] T. C., "CherryPy — A Minimalist Python Web Framework," url-http://www.cherrypy.org.

[11] R. Monica, J. Aleotti, M. Zillich, and M. Vincze, "Multi-label point cloud annotation by selection of sparse control points," in *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 301–308.

[12] R. C. et al., "Three.js," urlhttps://github.com/mrdoob/three.js, 2010.

[13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," *ArXiv*, vol. abs/1903.11027, 2019.

[15] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9552–9557, 2019.

[16] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft Level 5 AV Dataset 2019," urlhttps://level5.lyft.com/dataset/, 2019.

[17] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2241–2254, 2016.