SALT: A Semi-automatic Labeling Tool for RGB-D Video Sequences

Dennis Stumpf[1], Stephan Krauß[1], Gerd Reis[1], Oliver Wasenmüller[2] and Didier Stricker[1,3]

[1]*German Research Center for Artificial Intelligence GmbH (DFKI), Germany*

[2]*Hochschule Mannheim, Germany*

[3]*University of Kaiserslautern, Germany*

{*dennis.stumpf, stephan.krauss, gerd.reis, didier.stricker*}*@dfki.de, o.wasenmueller@hs-mannheim.de*

Abstract: Large labeled data sets are one of the essential basics of modern deep learning techniques. Therefore, there is an increasing need for tools that allow to label large amounts of data as intuitively as possible. In this paper, we introduce SALT, a tool to semi-automatically annotate RGB-D video sequences to generate 3D bounding boxes for full six Degrees of Freedom (DoF) object poses, as well as pixel-level instance segmentation masks for both RGB and depth. Besides bounding box propagation through various interpolation techniques, as well as algorithmically guided instance segmentation, our pipeline also provides built-in pre-processing functionalities to facilitate the data set creation process. By making full use of SALT, annotation time can be reduced by a factor of up to 33.95 for bounding box creation and 8.55 for RGB segmentation without compromising the quality of the automatically generated ground truth.

1 INTRODUCTION

Generating ground truth data for RGB-D data sets is extremely time-consuming and expensive, as it involves annotations in both 2D and 3D. The 3D bounding boxes of the famous data sets KITTI (Geiger et al., 2013) and SUN RGB-D (Song et al., 2015), for example, are created manually for every individual object in every frame or scene. Furthermore, as such tasks are often outsourced to workers with limited technical background, the annotation quality can vary highly. KITTI, for example, contains frames with missing or incorrectly labeled boxes. This raises the question, if the manual effort in the annotation pipeline for RGB-D data can be reduced without compromising quality.

As it can be seen in Section 2, this question opens up an active field of research where different approaches have already emerged. These approaches, however, all come with their individual limitations. They either require extensive pre-processing steps, are only applicable to static scenes containing rigid objects or limit the ground truth representation (e.g. 4 DoF object poses or segmentation for only one modality). Furthermore, many are designed for application scenarios involving an ego-motion of the camera.

To alleviate these and other limitations, we propose SALT (cf. Figure 1), a simple, yet effective tool to generate bounding box and segmentation ground
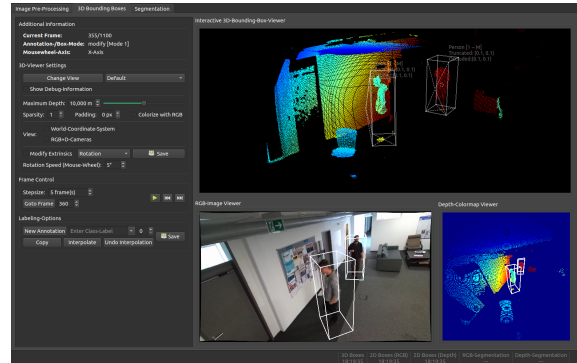

Figure 1: We introduce SALT, a tool to semi-automatically annotate 3D bounding boxes and pixel-level instance segmentation in RGB-D video sequences. This figure depicts a screenshot from the 3D bounding box module of the tool.

truth data for RGB-D video sequences in a semi-automatic fashion. The tool does not require any prior knowledge of the scene and is independent of any objects to be labeled. In summary, our contributions are:

1. A pipeline to rapidly generate 3D bounding boxes for full 6 DoF object poses up to 33.95 times faster than a naive approach.

2. Algorithmically guided generation of pixel-level instance segmentation masks in both 2D and 3D, with 2D bounding boxes as a by-product, speeding up annotation time by a factor of up to 8.55.

Table 1: Overview of state-of-the-art tools (upper half) and methods (lower half) for RGB-D data set creation. Listed are the detection and segmentation ground truth data representation, whether the tool provides built-in calibration functionalities (e.g. undistorting images or modifying extrinsic parameters) and if the functionalities of the tool are accessible through a Graphical User Interface (GUI). Green rows indicate approaches built for annotating indoor scenarios (e.g. by using a Microsoft Kinect) and yellow ones for autonomous vehicles (i.e. LiDAR sensor, ego motion of cameras).

Method	Approach	Detection	Segmentation	Calibration	GUI
Ours	interpolation + GrabCut	6DoF, 2D	RGB, D	✓	✓
(Marion et al., 2018)	mesh + reconstruction	6DoF	RGB, D	✓	(✓)
(Suchi et al., 2019)	incremental scene building	2D	D	-	-
(Wong et al., 2015)	learning	4DoF	RGB	-	✓
(Monica et al., 2017)	graph segmentation	-	D	-	✓
(Zimmer et al., 2019)	interpolation	4DoF	-	-	✓
(Wang et al., 2019)	tracking + clustering	4DoF	-	-	✓
(Arief et al., 2020)	tracking + clustering	4DoF	-	-	✓
(Plachetka et al., 2018)	tracking + optimization	4DoF, 2D	RGB, D	✓	✓
(Lee et al., 2018)	learning	4DoF	D	-	✓
(Huang et al., 2020)	3D-2D projection	-	RGB, D	-	✓
(Yan et al., 2020)	clustering	3DoF	-	-	✓
(Hodaň et al., 2017)	mesh + reconstruction	6DoF, 2D	-		
(Xiang et al., 2018)	mesh + reference frame	6DoF	D		
(Grenzdörffer et al., 2020)	mesh + reference frame	6DoF, 2D	D		
(Xie et al., 2016)	label transfer	-	RGB		
(Patil et al., 2019)	interpolation	4DoF	-		
(Dewan et al., 2016)	Bayesian approach	3DoF	D		
(Chang et al., 2019)	accumulation over time	6DoF	-		

2 RELATED WORK

Recent advances in the field of data set creation have shown that automating parts of the annotation process in order to minimize necessary human interaction can result in large-scale, high-quality RGB-D data sets in a short amount of time.

(Suchi et al., 2019) build a static scene by adding one object at a time, leveraging the change in depth for automatic pointwise segmentation. (Monica et al., 2017) use a nearest neighbor graph representation to segment a single point cloud based on individually labeled points. The quality of the generated ground truth of such approaches, however, highly depends on the depth quality and the texture of objects.

(Hodaň et al., 2017) and (Marion et al., 2018) reconstruct the scene, align pre-built meshes with the objects in the point cloud and then project back into the frames used for reconstruction. Similarly, (Xiang et al., 2018) and (Grenzdörffer et al., 2020) align meshes with the point cloud of a reference-frame and then propagate the resulting object configuration to other frames. While mesh based approaches provide full 6 DoF object poses and segmentation masks, they require a pre-defined set of rigid objects and corresponding meshes beforehand. Thus, they are only capable of annotating known, static scenes.

(Wong et al., 2015) use actively trained models to predict the 3D structure of indoor scenes from segmented RGB-D images. (Lee et al., 2018) leverage pre-trained Convolutional Neural Networks (CNNs) to predict a 3D bounding box from a single click on an object's point in the point cloud. (Xie et al., 2016) use a Conditional Random Field (CRF) model to transfer a coarsely labeled 3D point cloud and geometric cues into 2D image space, resulting in densely segmented 2D images. Such learning based approaches, however, require annotated data beforehand, do not guarantee to scale to different data domains and are highly dependent on the used architectures.

(Wang et al., 2019), (Arief et al., 2020) and (Yan et al., 2020) infer 3D bounding boxes from the spatial extends of clustered points. The former two additionally provide CNN assisted pre-segmentation and use tracking to propagate annotations into subsequent frames. (Dewan et al., 2016) use a Bayesian approach and motion cues to automatically detect and track objects. (Plachetka et al., 2018) track manually annotated 3D bounding boxes and corresponding polygons (for RGB segmentation). (Chang et al., 2019) accumulate manually selected points over time to automatically infer fixed size 3D bounding boxes. The drawback of these approaches is that the spatial extents of
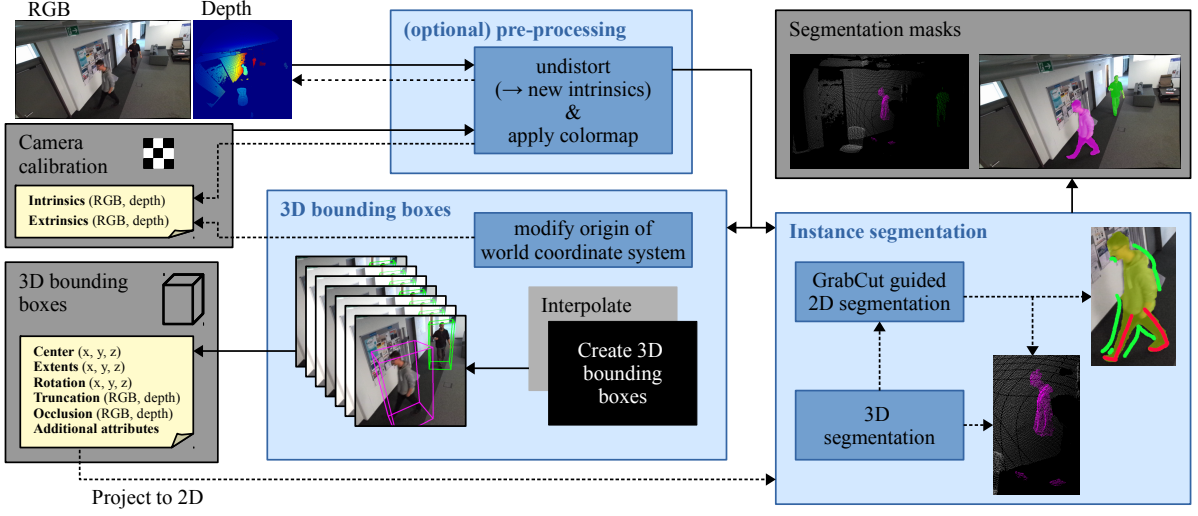
Figure 2: Overview of the pipeline of our proposed tool SALT. Input are the individual frames of an RGB-D sequence from a fixed viewpoint, as well as the corresponding camera calibration (intrinsics and extrinsic transformation between RGB and depth). The footage is either already undistorted, or can be undistorted using our tool. The 3D bounding box module can be used to not only generate 3D bounding boxes in a semi-automatic fashion, but also to modify the extrinsics of the RGB-D camera system w.r.t. the scene through simple drag and drop of the world coordinate system's origin. The instance segmentation module can segment depth maps and RGB images with the support of GrabCut (Rother et al., 2004), or in the case of depth maps, by selecting the points directly from inside the point cloud. Already segmented depth maps can additionally aid the segmentation of the corresponding RGB image.

an object in a point cloud may not always represent the full spatial extents due to occlusion, sparse measurements or reflective textures. In such cases, tracking and clustering may yield inaccurate results.

(Huang et al., 2020) manually segment background and moving objects in 3D and 2D respectively and by combining both, they retrieve a fully segmented image. (Zimmer et al., 2019) and (Patil et al., 2019), similarly to our approach, annotate 3D bounding boxes for only a subset of frames, while interpolating the remaining ones. Both, however, only interpolate linearly, which may not accurately capture real dynamic motion.

3   METHOD

In this section, we describe the core features of SALT in greater detail. A graphical overview can be viewed in Figure 2, which can be used as a guiding reference throughout this section.

3.1   Pre-processing

The tool internally uses undistorted RGB and depth frames (16 bit depth and 8 bit color mapped depth). The user has the option to either provide these images directly, or use the tool to generate them from the raw

footage. In both cases, the appropriate intrinsic parameters of both cameras and the extrinsic stereo calibration (rotation R and translation t between the individual cameras) have to be provided.

Figure 3: Simplified, schematic visualization of our undistortion pipeline for RGB images with radial distortion. The naive approach undistorts while retaining the original spatial dimension (green), resulting in loss of information through scaling. Our approach assures to keep the valid area (red) of roughly the original spatial dimension, retaining as much information (i.e. original pixels) as possible.

Our pipeline undistorts in such a way that as many pixels of the raw input image as possible are retained. As it can be seen in Figure 3, undistorting an image while retaining the resolution and all pixels results in areas (indicated in black) that do not contain any pixel values. The naive approach would be to crop out the area containing only valid pixels (indicated in red) and scale it back to the original resolution. But this results in loss of information, as pixels outside of the valid area are dropped and pixels inside are already scaled down. Our pipeline, on the other hand, deter-

mines the ideal resolution at which one side of the valid area matches the original image resolution. This assures that during undistortion, as few pixels as possible are lost. Depth maps are processed in a similar fashion. The tool undistorts each pixel individually and projects them back to the original image space. Depending on the distortion model, the ideal coordinates of projected pixels could fall outside of the original resolutions' boundaries. To capture those pixels as well, the necessary resolution and focal lengths are computed and a new camera matrix is generated.

While we do not claim novelty for this undistortion process, we argue that it is necessary to generate a data set that is as flexible as possible. This means that, at a later point in time, if the images in the data set need to be undistorted differently, already generated segmentation masks lose as few pixels as possible when being re-distorted.

3.2 3D Bounding Boxes

As depicted in Figure 1, the user accesses the scene with an interactive 3D point cloud viewer. To identify objects in the scene and adjust parameters more easily, the boxes are projected and displayed in the corresponding RGB and depth frames in real time.

Let $\{F_1, F_2, \ldots, F_N\}$ be a sequence of N RGB-D frames. When creating a new 3D bounding box B_k for frame F_k, a user first assigns a unique interpolation ID and a class label. The process of drawing the box and adjusting its parameters involves simple drag-and-drop operations (translation and scaling) and the use of the mouse wheel (rotation) to reduce the cognitive workload.

Let the object contained in B_k be visible for M frames. In order to retrieve the boxes for all frames $B_{all} = \{B_k, B_{k+1}, B_{k+2}, \ldots, B_{k+M}\}$ with minimal effort, two core features are provided: *copy* and *interpolate*. Given the sequential nature of videos, copying the boxes into subsequent frames facilitates the annotation process, since only minor adjustments, if any, need to be made to copied boxes. This can then be used in conjunction with interpolation: The user only needs to fully annotate box B_k, copy it into every m-th frame (keyframe) and apply small adjustments, resulting in $B_{key} = \{B_k, B_{k+m}, B_{k+2m}, \ldots, B_{k+M}\}$. A simple interpolation $f_{\mathrm{int}}(\cdot)$ then generates the box parameters for all intermediate frames. More formally:

$$B_{all} = f_{\mathrm{int}}(B_{key}) \qquad (1)$$

The advantage of using a simple interpolation instead of, for example, a more complex tracking algorithm is its robustness against sparse or occluded measurements. Even objects without any points (i.e. visible only in the RGB frame) can be annotated.

Commonly, translation and scaling parameters are interpolated individually using simple linear interpolation (e.g. (Zimmer et al., 2019) and (Patil et al., 2019)). We follow this approach for the scaling parameters (width, height and length), but implement a hybrid interpolation method for the translation parameters. More precisely, if the Euclidean distance c_Δ of the center coordinates is above a certain threshold ε for at least 4 consecutive keyframes, we apply cubic spline interpolation between those keyframes, otherwise linear interpolation is used:

$$f_{\mathrm{int}}(\cdot) = \begin{cases} \text{cubic} & \text{if } c_\Delta > \varepsilon \text{ for } \geq 4 \text{ keyframes} \\ \text{linear} & \text{otherwise} \end{cases} \qquad (2)$$

Rotation angles (yaw, pitch and roll), on the other hand, cannot be interpolated individually. Given the Euler angles φ, a single 3D orientation is constructed from individual rotations around the z, y and x axis in a fixed order. For this reason, we apply Spherical Linear Interpolation (SLERP), as introduced in (Shoemake, 1985). Instead of using three Euler angles, keypoint orientations are represented as quaternions. Thus an interpolation step involves moving between quaternions on a sphere around a fixed rotation axis with constant velocity, resulting in unambiguous rotational movement.

Furthermore, the tool automatically provides a truncation value t by determining the cut-off area of bounding boxes at image boundaries when projected into the RGB and depth frames. Occlusion o, on the other hand, can be manually specified and will be propagated to interpolated boxes. By combining these values, a visibility score v is computed as:

$$v = (1-t) \cdot (1-o) \qquad t, o \in [0, 1] \qquad (3)$$

This value is used to colorize boxes depending on their visibility, in order to guide the user in visually understanding the scene and annotations more intuitively. Furthermore, the occlusion and truncation values can be used to categorize the difficulty of individual samples similarly to KITTI (Geiger et al., 2013).

Finally, the world coordinate system's origin (i.e. extrinsic calibration of the RGB and depth camera) can be displayed and adjusted from inside the point cloud viewer using the bounding box control scheme.

3.3 Instance Segmentation

The segmentation module is, in its core, aided by the GrabCut algorithm (Rother et al., 2004). Initially, a user supplies a region of interest R_k (represented as 2D rectangles) in the image space, containing the object that shall be segmented. This generates an initial mask M_k with pixels in- and outside of R_k

being designated fore- and background respectively. Given M_k and the corresponding image, the algorithm learns Gaussian Mixture Models (GMMs) for fore- and background and improves upon the initial pixel assignment, resulting in a new mask M'_k, defined as:

$$M'_k = f_{GrabCut}(M_k) \qquad (4)$$

Depending on the input image, the process of Equation 4 may not yield an optimal segmentation result. In such cases, M'_k can be modified by the user in the form of coarse scribbles (cf. Figure 4), manually assigning fore- or background to the pixels. By running $f_{GrabCut}(\cdot)$ on the modified mask, the GMMs are updated further, improving the estimated segmentation mask. This process can be repeated iteratively until convergence. Asking the user to indicate fore- and background with coarse scribbles instead of polygons or exact pixel-level segmentation reduces the cognitive and physical workload by a wide margin.
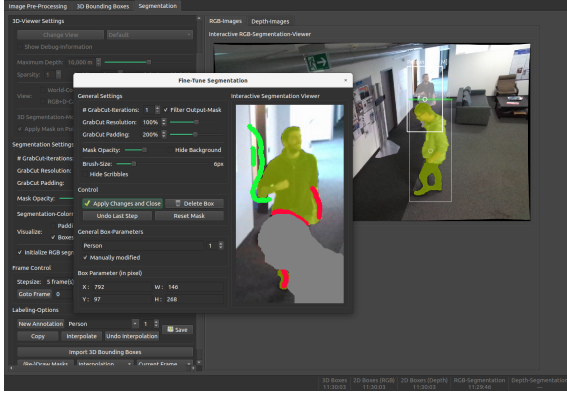

Figure 4: Screenshot of the segmentation module of SALT. Masks (yellow) are created by using GrabCut (Rother et al., 2004) and can be optionally fine-tuned. For this, a user provides coarse scribbles for fore- and background (red and green respectively) for the algorithm to iteratively improve upon its initial mask. Overlapping masks (gray) are used to initialize pixels as background.

To further improve both run time and accuracy, multiple extensions are applied to the default Grab-Cut implementation. First, the 2D rectangles can be created and interpolated similarly to the 3D bounding boxes, or inferred by projecting already annotated 3D bounding boxes into 2D. Furthermore, instead of generating M_k by using the whole frame F_k, we only use the rectangle and a small, padded area outside of its boundaries. We argue that the local surroundings of objects are sufficient for the algorithm to determine background pixels. If two masks $M_{k,1}$ and $M_{k,2}$ overlap, the pixels belonging to $M_{k,2}$ can be set as background when running $f_{GrabCut}(M_{k,1})$ (cf. Figure 4). Moreover, images fed into the algorithm can be downsampled beforehand to improve run time and

the generated masks can be filtered using simple morphological operations. Finally, individual masks can be copied into subsequent frames, speeding up annotation time of non-moving objects.

Even though GrabCut relies on color cues in the RGB color space to estimate masks, we empirically found that it is also applicable to the depth frames when provided as color mapped versions. We additionally convert them into grayscale images to make it easier for the user to differentiate between scribbles, masks and actual image contents. Given the three dimensional nature of depth maps, however, it is easier to differentiate between foreground and background when looking at the scene from multiple angles. Thus, we additionally provide the option to create depth segmentation masks by using an interactive 3D point cloud viewer similar to the one used for the 3D bounding box module. The process of creating a mask in 3D is visualized in Figure 5.
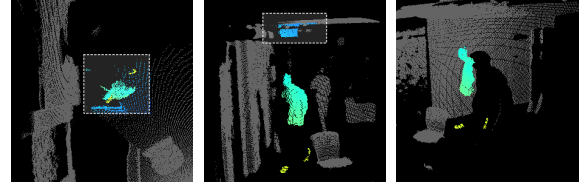

Figure 5: Segmenting a depth map using the 3D point cloud viewer. A user adds (left) or deletes (center) multiple points at once from different views using a 2D rectangle, resulting in a fully segmented object (right).

When a new mask is created in the RGB image, the tool will optionally search for a corresponding mask in the depth map. If found, its points will be projected into the RGB image space and used to initialize pixels as foreground in M_k, making the initial guess as precise as possible. Depending on the quality of the camera calibration and the density of the depth map, this feature can potentially remove the need to manually fine-tune the annotation.

4 EXPERIMENTS

In order to evaluate the effectiveness of our tool, we create two toy data sets: 1000 frames of two people walking around, and 100 frames of various objects being placed into a shelf (referred to as *ds_people* and *ds_objects* respectively). The first data set can be considered easy to annotate given the high framerate and depth map quality as well as simple object poses, while reflective surfaces, low framerate, low depth map resolution and complex object poses of *ds_objects* make it a challenging data set. Annotated

Table 2: Quantitative comparison of user annotations with the high-quality ground truth for both toy data sets in terms of Average Scaling Error (ASE) (↓), Average Translation Error (ATE) (↓), Average Orientation Error (AOE) (↓), Intersection over Union (IoU) (↑), Mean Absolute Error (MAE) (↓) and t_{avg} (↓). Coverage (↑) represents the percentage of annotated objects w.r.t. the ground truth (1.0 means all objects were annotated by the user). BB_{3D} are 3D bounding box annotations, S_{RGB} and S_{depth} are the RGB and depth segmentation masks. Naive represents user annotations with a naive tool (averaged across all users).

| Data | Metric | ds_people | | | | ds_objects | | | |
		User 1	User 2	User 3	Naive	User 1	User 2	User 3	Naive
BB_{3D}	ASE	0.320	0.338	0.294	0.296	0.226	0.332	0.239	0.393
	ATE (in cm)	9.781	9.264	9.182	8.103	2.543	1.867	2.253	2.753
	AOE (in rad)	0.257	0.150	0.149	0.141	0.156	0.185	0.115	0.241
	t_{avg} (in s)	1.53	1.85	3.86	51.95	8.62	7.63	11.93	88.71
	Coverage	0.960	0.999	0.966	1.0	1.0	1.0	1.0	1.0
S_{RGB}	IoU	0.943	0.948	0.953	0.942	0.955	0.958	0.956	0.948
	MAE	0.015	0.014	0.012	0.018	0.011	0.010	0.011	0.013
	t_{avg} (in s)	47.64	69.01	75.28	146.6	18.19	24.41	44.39	155.56
	Coverage	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
S_{depth}	IoU	0.999	1.000	0.998	-	0.874	0.904	0.886	-
	MAE	0.000	0.000	0.000	-	0.019	0.014	0.016	-
	t_{avg} (in s)	24.40	31.82	32.87	-	42.53	65.55	55.54	-
	Coverage	0.964	0.976	0.976	-	1.0	0.984	1.0	-

samples of these data sets can be viewed in Figure 6.

We asked three different users to annotate these sequences using SALT. For 3D bounding boxes, they were instructed to annotate every 40th and 5th frame of *ds_people* and *ds_objects*, respectively, as well as frames in which objects enter or leave the scene. Segmentation masks were annotated for every 20th and 5th frame, respectively. We compare the user annotations with a high-quality ground truth created with high care by a different expert user. Results are displayed in Table 2.

When evaluating the image segmentation masks, we report the common metrics Mean Absolute Error (MAE) and Intersection over Union (IoU):

$$IoU = \frac{\sum_{j=1}^{w} \sum_{i=1}^{h} \left(M_{i,j} \cdot M_{i,j}^* \right)}{\sum_{j=1}^{w} \sum_{i=1}^{h} \left(M_{i,j} + M_{i,j}^* - M_{i,j} \cdot M_{i,j}^* \right)} \quad (5)$$

$$MAE = \frac{1}{h \cdot w} \sum_{j=1}^{w} \sum_{i=1}^{h} \left| M_{i,j} - M_{i,j}^* \right|, \quad (6)$$

with h, w being the height and width of the 2D rectangle area containing the mask (cf. Section 3.3) and $M_{i,j}, M_{i,j}^* \in \{0, 1\}$ being the corresponding pixel values at position (i, j) of user annotations and ground truth, respectively.

For evaluating the 6 DoF 3D bounding boxes, we report the Average Scaling Error (ASE), Average Translation Error (ATE) and Average Orientation Error (AOE) as introduced in the *nuScenes* benchmark

(Caesar et al., 2020). However, we modify the AOE to take all three rotation angles into account, since *nuScenes* only reports *yaw*. More precisely, given the 3x3 rotation matrices R_{gt} and R_u for ground truth and user annotation respectively, the difference in orientation $\Delta\varphi$ can be computed as:

$$\Delta\varphi = arccos \left(\frac{tr \left(R_u R_{gt}^T \right) - 1}{2} \right) \quad (7)$$

All of the aforementioned evaluation metrics are averaged across all annotations of the individual data sets for each user. Furthermore, we measure the total annotation time per data set for each user, and compute the resulting average annotation time per individual object t_{avg}. For the 3D bounding boxes, both the manually created as well as the interpolated boxes are considered when computing t_{avg}.

To further evaluate the achievable annotation time speedup using SALT, we implement a naive version of the tool. This means that 3D bounding boxes are created without copying or interpolating, while segmentation masks are manually drawn without being assisted by the GrabCut algorithm. We let all three users annotate the same, randomly sampled subset of frames using the naive approach and report the average annotation time t_{avg} and accuracy per object across all three users. As objects are not always distinguishable in the depth maps, applying this approach on the depth segmentation has empirically
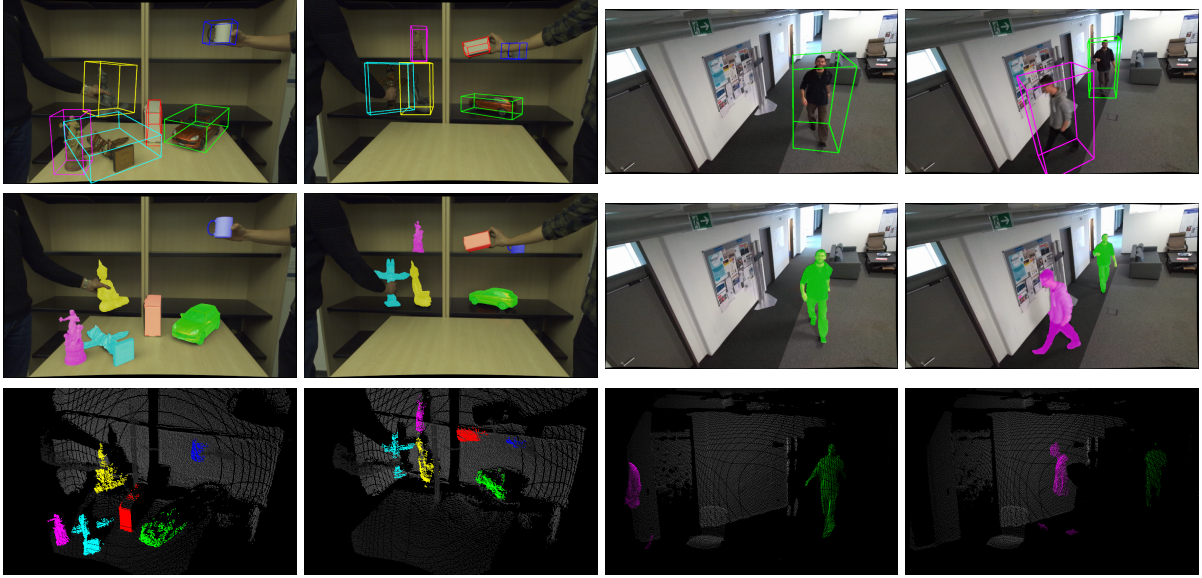
Figure 6: Qualitative results (3D bounding boxes, RGB and depth segmentation masks) of user annotated data using SALT. The two leftmost columns are from the ds_objects data set, the others from ds_people. Segmented depth maps are depicted as colorized point clouds. Due to the low depth resolution of ds_objects, points are projected onto a 3x3 pixel region.

shown to fail. Therefore, we only report the naive results of RGB segmentation and 3D bounding boxes.

As can be seen in Table 2, using SALT allows a reduction in annotation time without compromising quality compared to a high-quality ground truth. RGB segmentation can be sped up by a factor of up to 3.08 for ds_people and 8.55 for ds_objects, while 3D bounding box creation is up to 33.95 and 11.63 times faster, respectively, when compared to a naive tool. The gap between those factors is a result of the difference in difficulty for our data sets. Objects in ds_objects only move during short periods of time, allowing masks to be copied into subsequent frames, reducing the annotation time even further. Movements in ds_people involve less complex object poses, which makes annotating the bounding boxes easier. Even though User 3 has never used the tool before, and therefore achieves the slowest annotation time, the overall accuracy across all three users is on par and in some cases even higher than with the naive tool.

Lastly, we evaluate different interpolation approaches for 3D bounding box propagation in terms of ASE and ATE. More precisely, given the user annotated keyframes, we apply *linear*, *cubic* and our *hybrid* interpolation (cf. Section 3.2). The results are listed in Table 3 and suggest that relying solely on linear interpolation is not optimal. For our data sets, we achieved best results when applying linear interpolation on the scaling parameters and our hybrid interpolation approach on the translation parameters. In case of translation, linear interpolation performs worst as

Table 3: Quantitative comparison of different interpolation approaches for scale and translation parameters of the 3D bounding boxes.

	Metric	User 1	User 2	User 3
ds_people	ASE (linear)	0.320	0.338	0.290
	ASE (cubic)	0.331	0.339	0.299
	ATE (linear)	11.740	11.363	10.921
	ATE (cubic)	9.781	9.264	9.345
	ATE (hybrid)	9.781	9.264	9.182
ds_objects	ASE (linear)	0.226	0.332	0.239
	ASE (cubic)	0.225	0.332	0.239
	ATE (linear)	2.685	1.987	2.330
	ATE (cubic)	2.617	1.898	2.286
	ATE (hybrid)	2.543	1.867	2.253

it assumes constant velocity between keyframes, thus failing to capture dynamic object movements. Cubic interpolation on the other hand will induce jitter as a byproduct of the polynomial fitting shortly before movement when an object is standing still for some time. This drawback is solved by our hybrid approach, as we apply cubic interpolation only during object movement, and linear otherwise.

5 CONCLUSION

In this paper, we introduced SALT, a tool to semi-automatically annotate RGB-D video sequences. The

tool provides a pipeline for creating 6 DoF 3D bounding boxes, as well as instance segmentation masks for both RGB images and depth maps. We have shown that by making full use of the provided features, annotation time can be reduced by a factor of up to 33.95 for 3D bounding box creation and 8.55 for RGB segmentation without compromising annotation quality. In some cases, the quality of automatically generated data even improved, demonstrating that our provided functionalities reduce the cognitive workload of the user and make the annotation process more intuitive.

For future work, several additions can be considered to further enhance the efficiency of our proposed pipeline. For example, distinguishing objects of interest from static background in 3D point clouds is a challenging task. By adding ground-plane and background removal algorithms as part of the preprocessing pipeline for depth maps, distracting elements can be removed from the scene, allowing the user to identify and annotate objects with greater ease. Furthermore, depth points inside 3D bounding boxes can be projected to an already segmented RGB image to automatically infer depth segmentation masks, leaving the user to only remove or add individual points, if necessary.

ACKNOWLEDGMENTS

REFERENCES

Arief, H. A., Arief, M., Zhang, G., Liu, Z., Bhat, M., Indahl, U. G., Tveite, H., and Zhao, D. (2020). Sane: Smart annotation and evaluation tools for point cloud data. *IEEE Access*, 8:131848–131858.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628.

Chang, M., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8740–8749.

Dewan, A., Caselitz, T., Tipaldi, G. D., and Burgard, W. (2016). Motion-based detection and tracking in 3d lidar scans. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4508–4513.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 32(11):1231–1237.

Grenzdörffer, T., Günther, M., and Hertzberg, J. (2020). Ycb-m: A multi-camera rgb-d dataset for object recognition and 6dof pose estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3650–3656.

Hodaň, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., and Zabulis, X. (2017). T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888.

Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., and Yang, R. (2020). The apolloscape open dataset for autonomous driving and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2702–2719.

Lee, J., Walsh, S., Harakeh, A., and Waslander, S. L. (2018). Leveraging pre-trained 3d object detection models for fast ground truth generation. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2504–2510.

Marion, P., Florence, P. R., Manuelli, L., and Tedrake, R. (2018). Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3235–3242.

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

Patil, A., Malla, S., Gang, H., and Chen, Y. (2019). The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9552–9557.

Plachetka, C., Rieken, J., and Maurer, M. (2018). The tubs road user dataset: A new lidar dataset and its application to cnn-based road user classification for automated vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2623–2630.

Rother, C., Kolmogorov, V., and Blake, A. (2004). ”grabcut”: Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH ’04, pages 309–314.

Shoemake, K. (1985). Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’85, pages 245–254.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgbd: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576.

Suchi, M., Patten, T., Fischinger, D., and Vincze, M. (2019). Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic rgb-d datasets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6678–6684.

Wang, B., Wu, V., Wu, B., and Keutzer, K. (2019). Latte: Accelerating lidar point cloud annotation via sensor fusion, one-click annotation, and tracking. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 265–272.

Wong, Y.-S., Chu, H.-K., and Mitra, N. J. (2015). Smartannotator: An interactive tool for annotating indoor rgbd images. *Computer Graphics Forum (Proc. Eurographics)*, 34(2):447–457.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2018). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems (RSS)*.

Xie, J., Kiefel, M., Sun, M., and Geiger, A. (2016). Semantic instance annotation of street scenes by 3d to 2d label transfer. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3688–3697.

Yan, Z., Duckett, T., and Bellotto, N. (2020). Online learning for 3d lidar-based human detection: Experimental analysis of point cloud clustering and classification methods. *Autonomous Robots*, 44:147–164.

Zimmer, W., Rangesh, A., and Trivedi, M. (2019). 3d bat: A semi-automatic, web-based 3d annotation toolbox for full-surround, multi-modal data streams. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1816–1821.