

**FACULTY
OF MATHEMATICS
AND PHYSICS**
Charles University

BACHELOR THESIS

Vojtěch Šára

Virtual reality for data labeling

KSVI

Supervisor of the bachelor thesis: RNDr. Tomáš Holan, Ph.D.

Study programme: Computer science

Study branch: Artificial intelligence

Prague 2023

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In date

Author's signature

I want to thank RNDr. Tomáš Holan, Ph.D. for leading me on this journey of a thesis.

Besides that I want to thank Anna, Dominik, Pavel, Jaromír, Laura and Ali for participating in my user study.

Title: Virtual reality for data labeling

Author: Vojtěch Šára

Department: KSVI

Supervisor: RNDr. Tomáš Holan, Ph.D., KSVI

Abstract: Machine learning relies on large scale datasets which are time consuming and expensive to capture and label. This problem is especially pronounced in computer vision and even more so in spatial tasks. Fully labeled 3D datasets with bounding boxes and / or 3D segmentation are still quite rare and relatively small. In this thesis, we will propose a new way of labelling using virtual reality that can speed up the process. This idea has been explored to some extent, but we believe it deserves more attention. Aside providing an introduction to the researched area, we will first introduce our labeling program, describe the design choices and challenges and then provide a detailed analysis of it's effectivity.

Keywords: virtual reality, labeling, computer vision, supervised learning, spatial AI

Contents

Introduction	3
1 Problem formulation	4
1.1 The need for labeled data	4
1.2 Types of labels in CV	4
1.2.1 Bounding box	5
1.2.2 Pose	7
1.2.3 Segmentation	8
1.3 Translating data label types	8
1.4 The ideal dataset	9
2 How are datasets labeled today?	10
2.1 Used methods	10
2.1.1 Involuntary human expert	10
2.1.2 Human expert	11
2.1.3 Human expert with machine assistance	11
2.1.4 Human expert ensembling	12
2.1.5 Machine labeling?	12
2.1.6 Synthetic data	13
2.1.7 Summary of the methods	15
2.2 Notable datasets and how they were labeled	16
2.2.1 KITTI	16
2.2.2 CITYSCAPES	17
2.2.3 Pascal3D+	17
2.2.4 Replica	17
3 Outlining a new solution	18
3.1 Virtual reality as a human-computer interface	19
3.2 Programming our own labeling solution	19
3.3 Creating the user study	20
3.4 Choosing the label type for my experiment	20
3.5 Choosing the data modality	20
4 Design of a VR labeler	22
4.1 User's view of the application	22
4.1.1 The physical user interface	22
4.2 Design philosophy	23
4.2.1 Handling user input	23
4.2.2 Hand tool	24
4.2.3 Bounding box tool	24
4.2.4 Brush tool	25
4.3 Architecture of the Unity solution	25
4.4 Main design problems	26
4.4.1 Movement	26
4.4.2 Data loading and saving	26

4.4.3	UI elements in 3D	27
4.4.4	Point cloud rendering	28
5	Comparing the new method to industry standard	30
5.1	Setup of the user study	30
5.1.1	Productivity of an experienced user	33
5.2	Speed & precision comparison	34
5.2.1	Precision of VR labeling	34
5.2.2	Difference of our labels and the KITTI 360 labels	35
5.3	Shortcomings of VR labeling	35
5.4	Future work	36
	Conclusion	37
	Bibliography	38
	List of Figures	41
	Appendix - file structure	42

Introduction

The rise of machine learning has created a large demand for high-quality data. So much so that it prodded some part of the research community to mentally flip the problem from thinking model-first into thinking data-first. This shifts the focus to creating the largest, most precisely labeled dataset possible and the actual learning is left for the model-first researchers. We strongly align ourselves with the data-first mindset, and we rely on it as a philosophical basis of this thesis.

Dataset creation can be further broken down into two parts - data acquisition and data labeling. This thesis focuses solely on data labeling, more precisely on data labeling for computer vision using virtual reality. We will first try to formulate this problem in chapter 1, then include an overview of dominating ideas in this field in chapter 2. Transitioning from theoretical to practical, chapter 3 will explain the thought process behind conceiving a new labeling program in VR. In chapter 4 the program's design and function will be described, and aside comparing it to other VR labelers, it will be compared to non-VR labeling using the same type of data in chapter 5.

1. Problem formulation

Creating precisely labeled datasets for computer vision is slow.

1.1 The need for labeled data

It could be argued that the need for large labeled datasets will decrease in the future as the models we train get smarter about learning. Either by relying on unlabeled data, or by smarter methods of learning. Humans or animals don't need to see hundreds of thousands of examples before correctly classifying objects, so it's possible computers will eventually reach the same level of "low data learning".

To oppose this, we would add that even if we had such smart models, we would still want to evaluate them properly before deploying them in the real world. So a car that successfully trained to drive itself on only 1 000 scenes from the street should still be subjected to 100 000, or 1 000 000 evaluation scenes to be considered safe to deploy into the real world. For that reason we believe labeled data will always be useful, just that its use might shift from training to evaluation in the future.

1.2 Types of labels in CV

Choosing the right type of label is crucial as this decision directly imposes a hard ceiling on the amount of information that can be learned from a dataset. To demonstrate this on a simple example - if we would label a dataset of images from the street into binary "yes" or "no" for the question of "Does the scene contain a car?", it would be strictly less useful than having a bounding box for each car in every image. Strictly less because converting the second type of label into the first type of label would be trivial, but converting the first into second impossible.

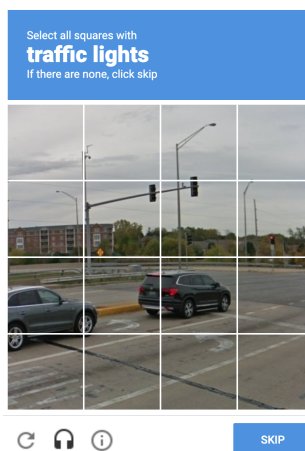


Figure 1.1: Weak labeling program reCAPTCHA

That doesn't mean that datasets with weak labels are useless. Acquiring weak labels can be much simpler as it can be for example done by distracted untrained humans as is the case with reCAPTCHA (Figure 1.1) Ahn et al. [2008].

For the purpose of comparing labels we can define a partial ordering:

Definition 1. *Let L be the set of types of labels. We define a partial ordering \prec_i on L such that for any two types of labels l_1 and l_2 in L , $l_1 \prec_i l_2$ if and only if a label of type l_2 can be converted into a label of type l_1 using a simple algorithm.*

As spatial datasets such as KITTI 360 (Liao et al. [2022]) usually include a 2D modality - in the case of KITTI 360 there are accompanying photographs with 2D projected labels, it feels appropriate to include a quick overview of popular types of labels in both 2D and 3D. We will discuss later in chapter 1.3 ways of translating label types into other label types. In our case of labeling in 3D, we will be mostly interested in ways how our 3D labels could be also automatically projected into \prec_i -wise weaker 2D labels.

1.2.1 Bounding box

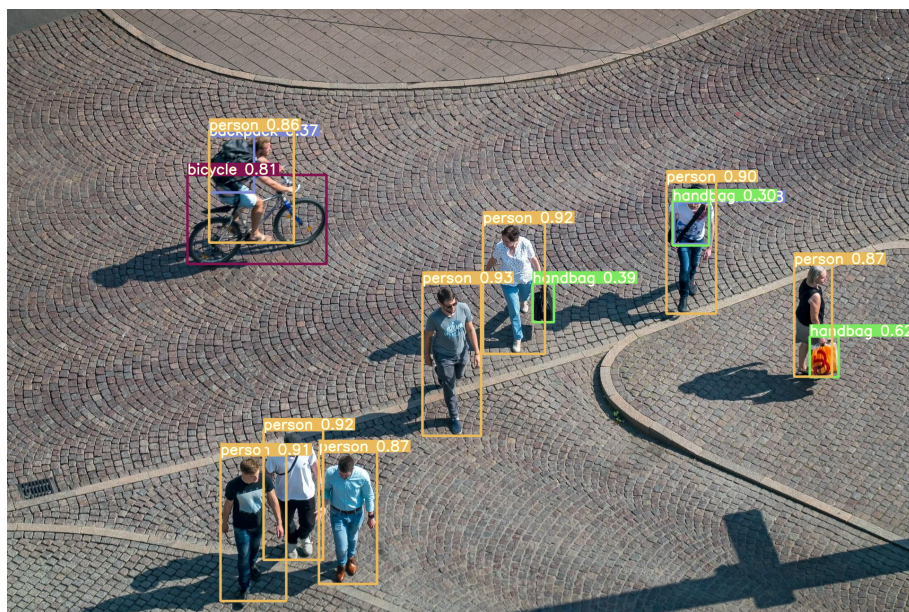


Figure 1.2: 2D Bounding box labels in YOLO v7 (Wang et al. [2022])

Used in object localization and object detection, bounding boxes are widely used in computer vision. It is sufficient for object counting (e.g. Liu et al. [2022]) and generally object detection (e.g. Wang et al. [2022]).

If we are solving a task that requires a deeper understanding of the scene, we can turn to other variants of a bounding box.

Rotated bounding box

A slight improvement for the loss function can be gained by allowing rotated bounding boxes (Wang et al. [2019]).



Figure 1.3: Rotated bounding boxes in aerial photo object detection (Wang et al. [2019])

3D bounding box

Much more important for our topic of labeling spatial data are 3D bounding boxes. A 2D bounding box is \prec_i to a 3D bounding box as again, projecting the 3D box into a 2D box is relatively simple, but the other way around requires some learning-based approach (Mousavian et al. [2016]).

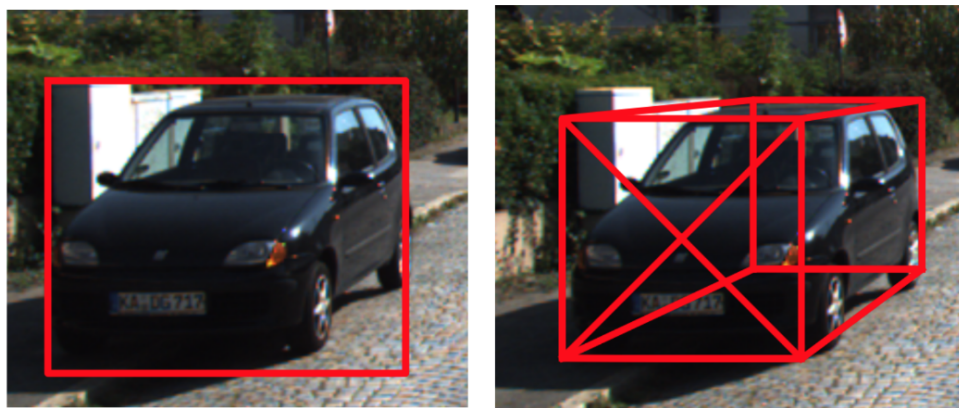


Figure 1.4: Difference between a 2D and a 3D bounding box

With 3D bounding boxes it is again important to consider rotations. In the most typical example of street scene datasets (KITTI 360 (Liao et al. [2022]), Cityscapes (Cordts et al. [2016]), Nuscenes (Caesar et al. [2020])), only z-rotations¹ are allowed. This is quite reasonable as buildings, pedestrians, traffic signs and cars are usually orthogonal to the ground, and the z-rotation constraint makes

¹rotations with the axis orthogonal to the ground

labeling on a screen a lot easier. Nevertheless, in some rare cases where cars are on a steep hill, or flipped on the side after a car crash, generally rotated bounding boxes should provide some benefit. Moreover, in different vision tasks, such as medical imaging, or object grasping, general rotations will be much more prevalent. The more common term for this task is **6D pose estimation**, not to be confused with **Pose estimation**, which I briefly describe in 1.2.2.

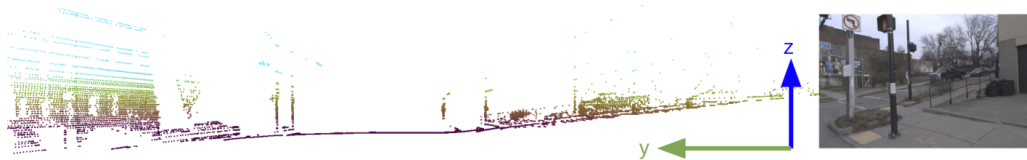


Figure 1.5: Example of an uneven ground requiring generally rotated bounding boxes (Chang et al. [2019])

1.2.2 Pose

Going further up the \prec_i hierarchy, pose is more information rich than a bounding box. Before labeling a skeleton is created with various constraints. During labeling this skeleton is fitted onto a either 2D or 3D view of a scene.

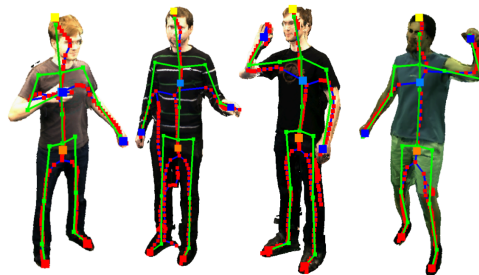


Figure 1.6: 3D human pose labels (Straka et al. [2011])

Most famous is probably the human pose example (Figure 1.5) and it is a type of label most useful in annotating objects with complex and more dynamic geometry (humans, animals, machines). But it can be used even in our favourite example of cars - here it can provide extra information about the orientation / dimensions / shape of the car (Figure 1.6).



Figure 1.7: Example of a car pose (b) (Chabot et al. [2016])

1.2.3 Segmentation

Segmentation breaks into two types and their combination: Semantic, Instance and Panoptic.

- **Semantic seg.** differentiates between classes of objects (car, road...)
- **Instance seg.** differentiates between distinct objects (car 1, car 2...)
- **Panoptic seg.** includes combined information of both.

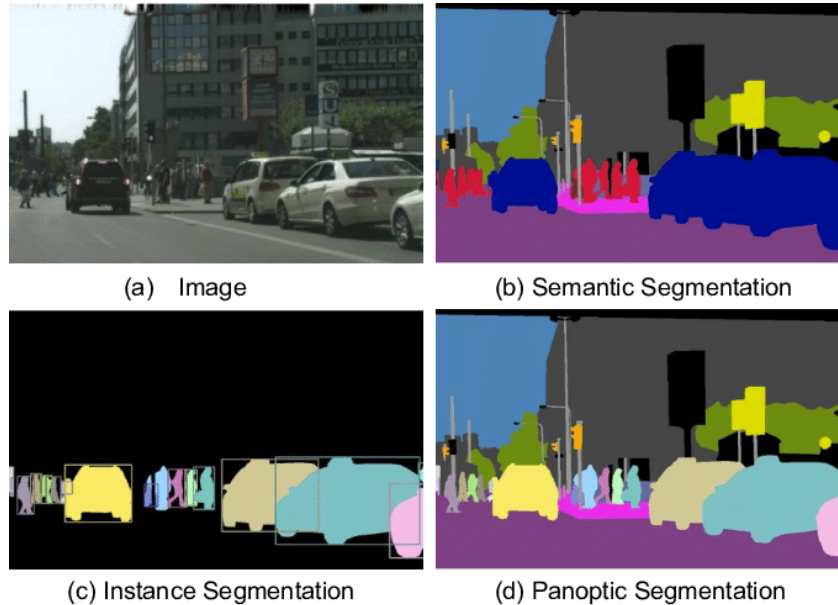


Figure 1.8: Types of segmentations

Segmentation is not \prec_i comparable to pose, but Panoptic segmentation is \succ_i bounding box. This holds true both in 2D and 3D, in both scenarios converting segmentation into a bounding box is just the matter of finding a minimal bounding box of the colored region for each instance.

1.3 Translating data label types

We could construct an entire map of different label types and ways of translating one type into another. Some translations would be trivial and others would require an entirely new body of research to create. As this would stretch outside the scope of this thesis, we will discuss only one translation most relevant to our work. That is point cloud segmentation into 2D per-view segmentation.

This is explained in detail in (Liao et al. [2022]), but in essence, if we have all of the intrinsic and extrinsic parameters of the camera known, it is a relatively simple matter of projecting the colored points onto the image plane. The only complication is occlusions - if we naïvely projected the points of the point cloud, objects occluding each other would mix their labels in the projection. Thus, some meshing process is needed to compute the 2D segmentations correctly.

1.4 The ideal dataset

In this chapter we tried to look on the data labeling problem from a more problem agnostic perspective. Obviously, dataset creation is always bound to some problem - as KITTI 360 is to autonomous driving. On the other hand, it is quite general and it is interesting how it became more general and information rich going from the 2013 original paper (Geiger et al. [2013]) to the 2022 one (Liao et al. [2022]). This will be discussed in more detail in the next chapter.

It is not difficult to conclude that the best dataset would include every type of label possible, every modality from pointcloud, image, to sound, smell... It would be as large as possible and include every encounterable real world situation. Downsampling is always simpler than upsampling. For autonomous driving, we could imagine a Google Earth sized dataset with KITTI-360 level of detail. As KITTI-360 includes 73.7 km driven and the world in total has probably tens of millions of kilometers, not including that we would probably want to visit each location many times to sample it under different conditions.

This seems completely unreasonable, but as cars are equipped with more and more sensors every year, unlabeled datasets of these kinds of proportions are already under construction. Fully labeling them by human annotators might be impossible, so a drastic simplification of what and how the data gets labeled will be needed.

2. How are datasets labeled today?

The area of data labeling is fascinating as it includes many creative ways of extracting useful information from voluntary or involuntary human annotators. These methods are discussed in more general way in 2.1, focusing on not only spatial data, as inspiration from other data modalities might be useful. Before diving into the different methods, we think it is good to establish that data labeling is no longer just a scientific / engineering challenge, but a day to day business reality.

Built-in workflow with Amazon Mechanical Turk

If you use [Amazon Mechanical Turk](#) for labeling, you are charged per object per review instance. We recommend that you use multiple labelers per object to improve label accuracy.

Workflow	Suggested price per labeler
Image classification	\$0.012
Text classification	\$0.012
Named Entity Recognition (NER)	\$0.024
Bounding box	\$0.036
Semantic Segmentation	\$0.84

Figure 2.1: Pricing of labeling data on Amazon (Hudgeon and Nichol [2020])

2.1 Used methods

2.1.1 Involuntary human expert

By far the most popular method of labeling is what I call the "involuntary human expert". The trick is to piggyback getting the label from a user trying to achieve something entirely different from labeling. One example of this is the already mentioned reCAPTCHA. Another strategy I would include in this category is mining already labeled data-label pairs, recent example of this is the LAION-5B dataset (Schuhmann et al. [2022]) behind DALL-E 2. There, the work of internet users linking text to images via the HTML's image tag alternative text option was exploited.

On top of obtaining this set of coarse labels, the authors follow up with advanced filtering done by human experts, which would fall into the "Human expert" method discussed in 2.1.2. This same schema of large scale coarse label mining with human postprocessing can be seen in the ImageNet dataset (Deng et al. [2009]) which made waves across the machine learning field. For the fine-tuned labeling, the aforementioned Amazon Mechanical Turk service was used.

In practice, even datasets without explicit labels are used in the discipline of unsupervised learning. And even though it is true that e.g. the entire corpus of text on wikipedia is "unlabeled", when we train a model for next word prediction on it, like GPT, every word becomes essentially a label for its preceding window of words. Thus, I would argue, that there still is human supervision, it is just more subtle and there isn't an explicit labeling process.

2.1.2 Human expert

Even though human experts can be assigned to a task in a few clicks today with services like Amazon Mechanical Turk, human expertise is still time and money consuming, especially for tasks which require specific knowledge. In tumor detection and many other tasks that cannot be done by unqualified workforce, the cost of a label is the bottleneck.

As we generally delve into more expertise-heavy tasks, it is likely that an increasing amount of effort will be spent on designing highly optimized labeling programs. What role Virtual Reality and other human-computer interfaces will play is yet unclear as these methods are not yet widely used today. The reasons for this will be discussed in chapter 5.

An alternative to paying a less or more expert labeling workforce would be to crowd source the task on the internet. It is hard to motivate people without spending money, but one way this can be done is with gamification. Turning labeling into a fun game is quite a challenge. On top of it, a good game should have some method of scoring your work and for that, the data would need to be pre-labeled, defeating the purpose. Despite that, there were some interesting experiments done in this area, such as (Hebart et al. [2019]) and even the shooting labels paper (Zama Ramirez et al. [2020]) tries to gamify the process with using guns for the labeling process.

2.1.3 Human expert with machine assistance

To lighten the workload of human labelers, we can employ classical or learning-based approaches to pre-label the data coarsely, and leave only selection / refining work on the human labelers. Another option is to provide assistance during labeling, a very recent example of this is (Kirillov et al. [2023]) where a tool was developed that can automatically expand a segmentation to fully annotate an object from a single click.

Such assistants are less common in the spatial domain, and we believe there is a lot of potential benefit in utilizing them. Methods for general 3D object recognition such as those discussed in (Mirbauer et al. [2022]) could be used to pre-label the scene coarsely and then be fine tuned by a human labeler. This would be an interesting avenue to follow with our VR labeling program.

One way of providing machine assistance during VR labeling is discussed in (Zama Ramirez et al. [2020]). In the paper, they employ a filling algorithm, that

for each labeled mesh triangle spreads to its unlabeled neighbours.

We could also include in this category the practice of labeling weak labels and automatically translating them into stronger labels, which is showcased for example in (Chen et al. [2014]). I already discussed this method in chapter 1 under "Translating data label types".

2.1.4 Human expert ensembling

Assuming the errors of the human labelers are uncorrelated, having each data-point labeled by M labelers will after averaging their annotations decrease the error, in the limit making it approach zero. This argument stands on shaky philosophical grounds of truth being merely an arithmetic average of opinions of all people. Brushing over that, the error elimination property is nevertheless real.

In the practice of creating large datasets, this simple (but expensive) trick is often used. For example the dataset ScanNet (Dai et al. [2017]) states:

"scans in ScanNet are annotated by 2.3 workers on average"

2.1.5 Machine labeling?

The ideal way of labeling would be entirely automatic with no human input needed. Rephrasing that goal, we would want to create an auto-labeling computer program P_{AL} that guesses correctly the labels for each input it is presented. But if we knew how to do that, we wouldn't need any labeled data, as we could skip learning, and just use P_{AL} for solving the task directly. It's not hard to see the circle in this problem.

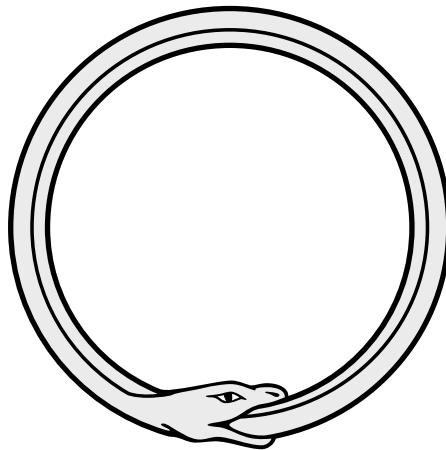


Figure 2.2: Snake Ouroboros eating its own tail - Mohamed Ibrahim, public domain

That being said, machine labeling is not entirely nonsensical and in some cases, it is extremely useful. For example, NVIDIA developed a method (Chaitanya et al. [2017]) for denoising 3D renders by creating a massive dataset of [quickly rendered noise image] - [slowly rendered, with many more samples per

pixel, detailed image] pairs. These pairs were generated by state-of-the-art classical rendering methods.

The computer was essentially forced to find a more efficient algorithm for correctly approximating the lighting of the 3D scene, which impressively enough, it did. Learning-based denoising is now shipped with modern videogames by default under the name DLDSR.

An important note is that this didn't solve any new problem, it just compressed an already functioning solution to a smaller computational footprint. And that is a constraint that holds true for all machine labeling - it can be used to find a faster / smaller solution, but only to an already solved problem.

Another variation of machine labeling might come in the future with "AGI" - we could imagine big, data-center sized general machine learning models used to label data for smaller, more specialized programs, running on a mobile device for example.

Based on this, "Model compression" might be a better term for what I called machine labeling. Nevertheless, it is an interesting area of research, as it is independent of large scale manual labor.

2.1.6 Synthetic data

We will now briefly comment on our favourite solution to the data labeling problem. As synthetic data is defined somewhat broadly, we will be considering only one strategy in computer vision labeled under this term. The strategy is of creating computer generated 3D scenes and novel views into them with large scale randomization of the environment. Datasets like SynLiDAR (Xiao et al. [2021]), Virtual KITTI (Gaidon et al. [2016]) and the original (Richter et al. [2016]) were created this way.

When creating a synthetic dataset, the need of labeling each frame is eliminated by creating labels ahead of time for each object. The human supervision is elevated from intense labeling to pooling large amounts of 3D and 2D assets into a 3D renderer of choice and randomizing every relevant parameter (time of day, noise, lighting, textures...).

It is a strange and delicate discipline of stating what isn't important and what is using randomization. Taking an example from a popular synthetic dataset for human recognition PeopleSansPeople (Figure 2.3), we can see how everything other than human forms is strongly randomized, creating a rather surreal image.

It is shown in the paper that despite "looking strange", this wild randomization performs quite well, and other sources confirm this (Tremblay et al. [2018]), with the general notion that such datasets are most suitable for pre-training, which should be followed by fine-tuning on real data.



Figure 2.3: Sample from the PeopleSansPeople dataset for detecting humans in images Ebadi et al. [2022]

The truth problem

Ground truth is the absolute frontier of each problem solved by machine learning. In most cases our datasets are not including the whole truth - even with millions of pictures, it is hard to capture every imaginable scenario that can happen on a road for example. Other smaller tasks might differ in this and have the truth relatively well covered.

So this is the first lie hiding in what is usually called ground truth - the lie of incomplete data. The other lie that can sneak into datasets is incorrect labels. Due to this, the overall picture looks something like this:

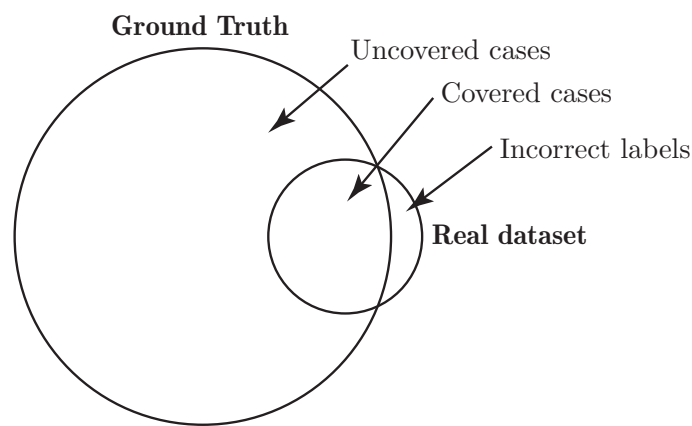


Figure 2.4: Datasets approximate truth

This is for all "classical" labeling - real world datapoint + human annotated label. Considering synthetic data, we will typically have something like this:

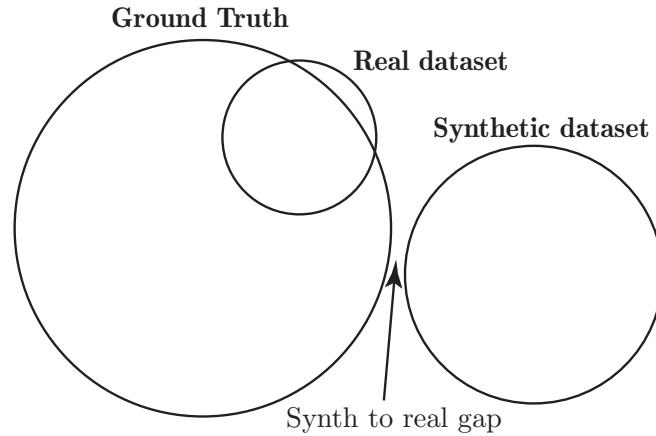


Figure 2.5: The gap between synthetic and real

With a gap between synthetics and reality, but "close enough" to the truth that it is still very useful as pre-training for example. With a very good synthetic dataset, we might get an overlap with the truth. With some of the results from performance of synthetic datasets, it is interesting to consider "super-truth". Basically a lie, an image that would never be produced by the sensory array of the car for example, but which is more useful for training than any other image.

We could imagine a similar strategy to creating adversarial images as in (Ma et al. [2020]) to creating a maximum-information datapoint-label pair. The results might be informative for a dataset creator. This interests me and as I spent a better half of the last year studying synthetic datasets, I felt compelled to include this brief detour. At least there is one very concrete takeaway for this thesis - measuring precision of labeling, which will be discussed in the last chapter.

2.1.7 Summary of the methods

To conclude this section, let's plot all the discussed options on a line based on the amount of human supervision needed per labeled frame. We should see something like this:

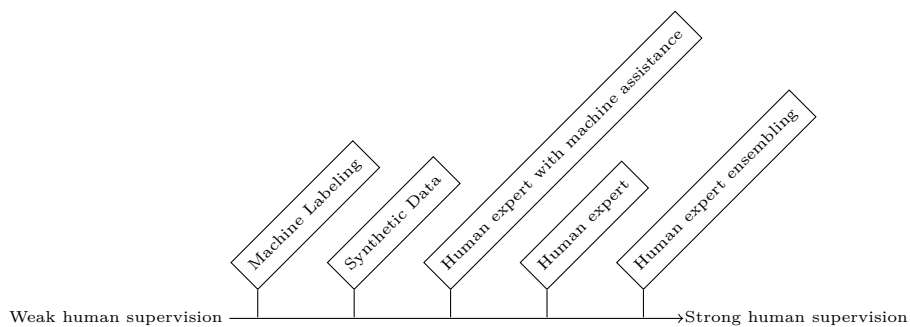


Figure 2.6: Amount of supervision per labeling strategy

2.2 Notable datasets and how they were labeled

2.2.1 KITTI

The original KITTI dataset (Geiger et al. [2013]) had only 3D bounding box labels provided by human experts and annotated only inside the visible frusta from RGB cameras. The technique used for labeling isn't discussed in the paper. The big contribution of this paper were the benchmarks that were established for many different tasks.

First evolution of KITTI was the semantic KITTI dataset (Behley et al. [2019]). In it, the entire point cloud around the car is labeled with per-point semantic segmentation. On top of it, authors of the paper include a relatively detailed account of the labeling process. They preprocess the point clouds into neat 100x100m chunks with small overlap. Each of these chunks is then assigned to a human labeler, alongside a video manual explaining best labeling practices. After labeling, each chunk is inspected by a human supervisor.

More interestingly, labeling speed is mentioned - 1400 hours of labor for labeling 518 tiles, with 10-60min verification by a human supervisor per tile. With a highway tile taking on average **1,5 hours** and an in-city tile taking **4,5 hours**. For a later discussion of speeding up the labeling process, this kind of information will be most useful.

The latest member of the KITTI suite is KITTI-360 (Liao et al. [2022]). Here, panoptic segmentation for full 360 views of the scene are included alongside bounding boxes and other bounding shapes. Authors include a more detailed description of the labeling process, following a trend we anecdotally noticed of increasing attention to labeling over the years. The authors provide the open source labeling program that was created for the purposes of the dataset. On top of it, they go into detail about the annotation procedure with a thought out quality control pipeline, and also comment on the annotation time, saying:

"On average, annotating one full batch (~ 240) frames in 3D required about 3 hours"

The batches aren't of a regular size, but are roughly equivalent to the 100x100m regions of semantic KITTI, so there seems to be a slight speed improvement.

Overall, the KITTI suite was the best we could find in this regard, going very far to explain their labeling process. From further research, we learned that this is not standart and most datasets don't include much information on the labeling process in their accompanying papers. As the labeling process is the source of truth for all methods that will battle for accuracy on the active battlegrounds that the various benchmarks are, we find this troubling. Let us now quickly go through some of the big datasets in the field and summarize what they have to say in this regard.

2.2.2 CITYSCAPES

CITYSCAPES, (Cordts et al. [2016]) describes a two stage process. First, fine, pixel-level labels are created for a smaller subset of training data. These take 1.5h for one single image and the authors claim 5000 annotations were first created this way with a quality in-house labeling workforce. This perfect set of labels is then used to measure the precision of faster labelers - these only had 7 minutes to label each image, but were able to achieve 96-98% accuracy of the fine labels.

2.2.3 Pascal3D+

In Pascal3D+ (Xiang et al. [2014]) a custom labeling program with functionality of CAD model placement into pictures is presented. The labelers were hired and not outsourced, which could have helped the authors monitor and instruct the labelers directly to produce higher quality labels. The process is described as first picking the correct CAD model (SUV / sedan / truck...) and then moving it into the right place, scale and rotation. The speed of labeling isn't commented.

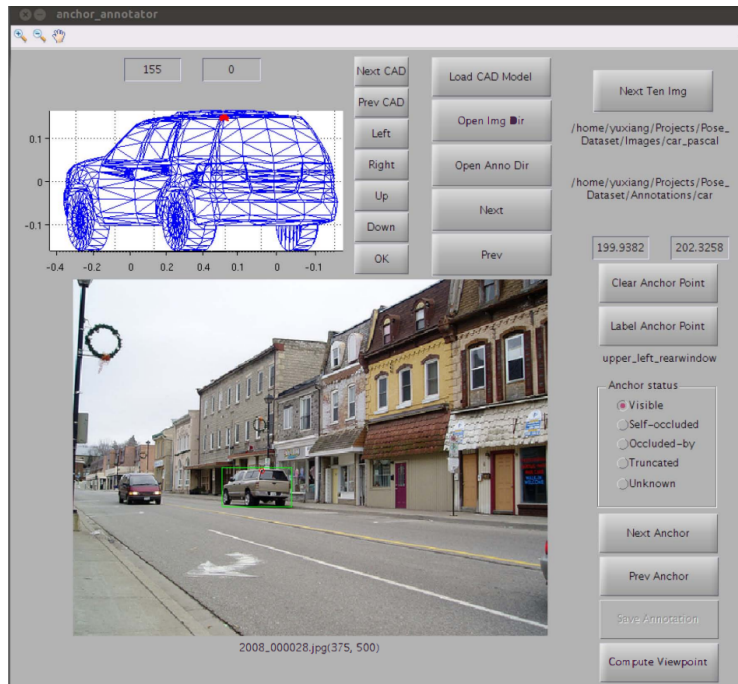


Figure 2.7: Pascal labeling tool

2.2.4 Replica

Going outside just street / road datasets, Replica is a dataset from Facebook research (Straub et al. [2019]) with a very interesting synthetic + real mix. Scenes are collected from real world rooms with a special sensor rig and then reconstructed in 3D using photogrammetry. Labeling is performed by first annotating in 2D from rgb images and as camera parameters are known for each camera, these 2D labels are then projected onto the 3D mesh. A 3D annotation follows with a "mesh painting" feature. The initial 2D segmentation is machine assisted, or completely automatic - this isn't clear from the paper.

3. Outlining a new solution

In chapters 1 and 2, we motivated the problem of data labeling and shown some practical examples of how this process is performed today. Easing into the second part of the thesis, we will demonstrate our very own solution. This chapter will be dedicated to explaining the thought vector from which we are approaching this problem and what were the challenges and inspirations along the way.

We started by outlining the labeling process as it works right now according to the top papers in the field.

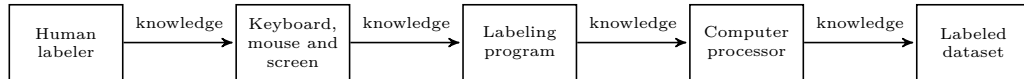


Figure 3.1: Knowledge flowing from human into the dataset

Going from left to right, there are a few candidates for what could be the bottleneck in this process. Starting from the human, expertise can be a huge factor. Aside that, fatigue, lack of attention, poor motivation, eyesight, hand-eye coordination and many other factors can play a role. These are all hard to systematically battle, but from the literature, hiring labelers and supervising them thoroughly seems like the best option, if the budget allows it.

The human inputs his answers into a labeling program. The program might have unoptimized UI, label brushes, bounding boxes and other functions might be hard to use. The labeling program can have bugs and imprecisions. From a review of the available labeling programs, bugs were rare, but counter-intuitive UI wasn't - as these programs are often "single-use" for the purposes of creating one large dataset, the extensive user-in-the-loop iteration of the UI is limited, leading to some shortcomings in this area.

The labeling then translates the knowledge from user inputs into a machine-readable format using either a GPU or a CPU. When labeling millions of points in a pointcloud, or triangles in a large mesh, this can actually manifest as a problem too. Some labeling programs allow for preprocessing - for example splitting the point cloud into smaller chunks with some overlap, which does introduce a bit of inefficiency. Optimization on this front could bring some gains in labeling speed.

The last connection we didn't comment on is the interface between human and software. Typically, this interface consists of a keyboard, mouse and a screen, which is a great option due to its ubiquity. We identified this as the biggest weakness of the entire process. While labeling in some of the mentioned existing tools, the recognition of the objects felt simple and the programs were definitely performant enough to handle much more input from the user. On the other hand, switching between rotating the scene, picking the right tool and labeling always from a 2D viewpoint felt cumbersome and slow. That is where the idea to use VR originated.

3.1 Virtual reality as a human-computer interface

Virtual reality, or VR for short, is the marketing term for a hardware set containing the headset with a display for each eye, some controllers for user input and a system for rendering a view based on the head's position - either outside in or inside out tracking. I will be using the abbreviation VR, as it is practical, although not necessarily rooted in rigorous scientific terminology. For work in 3D, VR has few unique capabilities:

- **Stereo scene view** - every conventional screen prevents seeing perspective. VR headsets on the other hand offer a different viewpoint for each eye, triggering natural depth perception, even without moving the user's head.
- **Natural movement translation** - the user can change his view by simply rotating or moving his head. This grants only limited range, but is enough for a lot of small movements, that would require switching to a "move view" mode in a 2D program.
- **6 dimensions in each hand** - the VR controllers offer natural movement translation similarly to a mouse, but in 3D. On top of it, they register rotation on all the axes, enabling a large space of gestures and dual hand interactions.

For these reasons, we were excited to test this relatively new type of device for the task of data labeling. We found that this idea has been tested already, most notably in (Zama Ramirez et al. [2020]), or (Zhao et al. [2022-07]). We took inspiration in some design choices of our own solution, but also tried to approach the problem with a set of fresh eyes to possibly find novel ideas - in this regard we believe we succeeded and the individual contributions will be mentioned in the following two chapters.

3.2 Programming our own labeling solution

For the reasons above, we decided to write our own labeling program with Unity. The technical details and design decisions will be provided in the next chapter, but the purpose was to meet some of the real problems of designing a labeler in the process of engineering. We wanted to see if such a program could be created with limited resources while being useful. We confirmed that yes, a functional labeling program can be built with Unity and real data from the KITTI 360 dataset can be loaded into it and be successfully labeled. To really test the usefulness, we organised a user study with the following goals:

- Test if an untrained user is able to understand the UI and will correctly classify objects without assistance and without leaving the experiment due to VR sickness.
- Test what precision is achievable with this technique.
- Test what speed is achievable with this technique.

3.3 Creating the user study

To help explore the potential use of virtual reality for labeling, we conducted a small scale user study with seven participants. The idea behind it was to test if our labeling program is intuitive and effective, and to experience the reality of working inside VR. This experiment and its results will be exhaustively covered in chapter 5. The experiment went through an evolution of its own, starting with the idea of a labeling race where participants would compete against each other in labeling speed and accuracy. This turned out to be unnecessarily stressful, so this idea was scratched before inviting the participants. Also, the label type had to be chosen.

A second goal of the user study was to collect data about "VR sickness" and obtain user feedback on the features of the labeler.

3.4 Choosing the label type for my experiment

We first considered making each participant label bounding boxes only. After implementing it in our labeler, we had the idea that semantic segmentation might actually be simpler for the user, because even though bounding boxes around cars and people were easy, it was much less clear how to box a curved road, a tree or a fence. And as segmentation is also \succ_i to bounding box, it seemed like a better choice. The only problem was that instance labels would also have to be provided to be \succ_i to bounding box. These could be added by either adding shades to the label brush, so all cars were blue, just different instances would be different shades, or by combining semantic segmentation with the bounding box.

As we wanted to maximize the time of labeling and minimize the time of onboarding of the labeling tool, in the end we chose to test **only semantic segmentation** for this experiment. The labeling program includes the bounding box labeling and saving logic, so this stronger pair of labels can be created, but we didn't subject to any experiments in this thesis. This makes the comparison to the KITTI-360 labeler weaker, we will discuss this in detail in chapter 5.

3.5 Choosing the data modality

In both the mentioned papers on this same topic, semantic segmentation of a mesh is performed. From quick testing, an uncolored mesh is about as semantically understandable as an uncolored point cloud - with enough camera movement, it is generally not hard for a human eye to recognize shapes in both. Meshes seem to better represent man-made objects and point clouds better capture leaves and other high frequency detail.

The more important aspect is color. Both of the mentioned methods start with a single-color mesh. For our experiment, we were able to use KITTI 360's

color data, which greatly improves visual understanding.

One really helpful addition would be to include the images from the rgb cameras inside the pointclouds, helping the labeler understand the scene even better. This would add quite a lot of extra complexity, so we didn't add this feature, but it would be on top of the list of candidate features. Inside these 2D windows into the 3D scene, we could even display a live version of the translated 2D semantic label which would give the user a very powerful feeling of creating a label for many views at once.

Another future direction could be to switch from point clouds to NERFs. This would give the labeler unprecedented clarity of the 3D scene, but besides being very computationally intensive, it would be hard to represent the labeled area. With point clouds, keeping a simple class index for each point is sufficient. For NERFS, something like a volumetric label field would have to be implemented.

4. Design of a VR labeler

4.1 User's view of the application

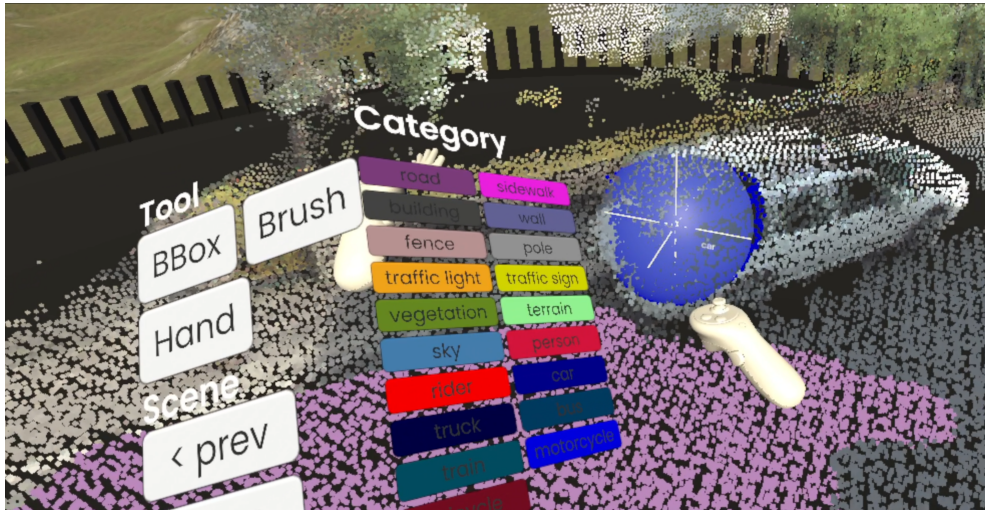


Figure 4.1: User's view into the VR labeler

The user is entering the application to label 3D scenes in VR. He needs to be able to load his own custom data, the label set and then have some way of exporting the labeled dataset. The labeling process should be intuitive and smooth and allow both bounding boxes and semantic segmentation modes. The scene he is labeling must be easy to move around and scale.

To build a program that enables all of the above and allows easy future extension, I chose Unity with the OpenXR library. The choice came from my experience with the tooling, but the same application could be built with Unreal Engine, or be created entirely from scratch with just the OpenXR library. This last choice would be significantly more complex, but could enable a performance boost.

4.1.1 The physical user interface

The application is optimized for the Valve Index on which it has been created. Other headsets are supported by default by the OpenXR library, but haven't been tested. Headsets come in different shapes and sizes, but don't really change anything about the use of the application, the more important device is the controller.

The Valve Index controller offers a rich variety of inputs, but we avoided using those as that would make the application device-locked. The picture is provided so the reader has a better idea of the functionalities that will be explained in the following sections.

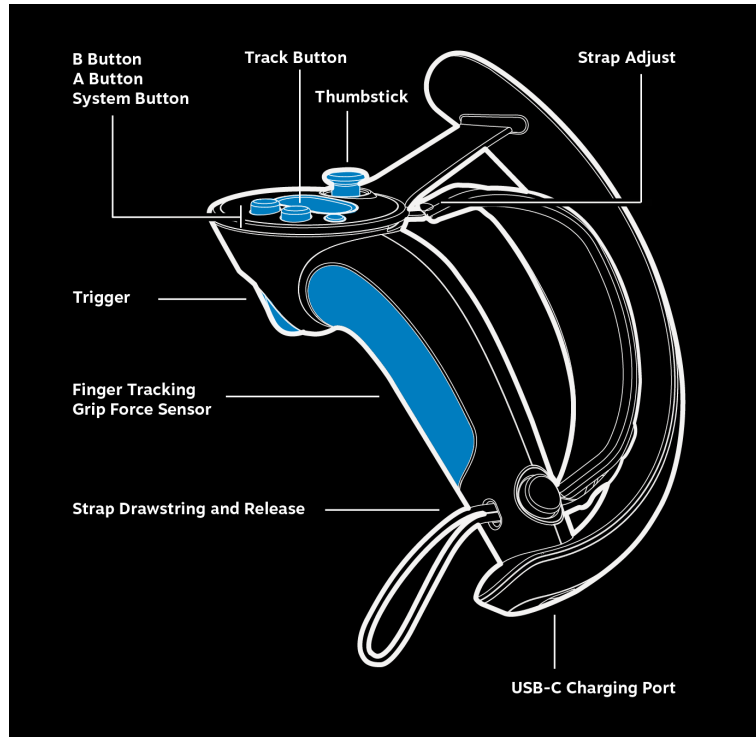


Figure 4.2: Valve Index controller (source: Valve)

4.2 Design philosophy

Ease of use was a very important parameter for the design as we wanted to onboard the users as fast as possible. We think we were successful in this regard as it took the participants in our small scale user study on average 2,6 minutes to start being productive with the tool. As this application is meant to be used by power users, it would make sense to trade off some of the simplicity for a wider variety of options in further development. The ambition was more to try to find a new starting point for the interface, stripping features to the simplest form possible.

4.2.1 Handling user input

There are two main strategies in interface design. One is to decorate the various elements with "gizmos" - leaving the user with one main tool and the different actions are performed on interaction with the elements. The other option is to separate different functionalities into different tools that can be cycled through. As we wanted to enable painting of the point cloud with a 3D brush, the tool mindset seemed much more suitable.

There are three tools available: hand tool, bounding box tool and brush tool. These function similar to the analogous 2D tools in Microsoft Paint or Adobe Photoshop for example. Hands are used to move the scene around by the user, bounding box is used to draw boxes using both controllers and the brush tool sits in the right hand only and is used to color the point cloud points. On top of this, the left hand has a sticky menu with a label palette, previous / next scene

button and the tool selector. This left / right hand split is inspired by Google’s Tilt Brush.

4.2.2 Hand tool

The left hand is always in the hand tool mode, the right hand only if it is selected. When only the left hand is active, the user can move the scene with just the left hand. Enabling right hand too allows the user to also scale the scene by holding both hands and moving them closer together. We considered adding rotation, but opted not to as we wanted to limit fast movements of the scene as they can cause nausea. In hindsight, we would lean back again to allowing rotation as the users were asking for it.

4.2.3 Bounding box tool

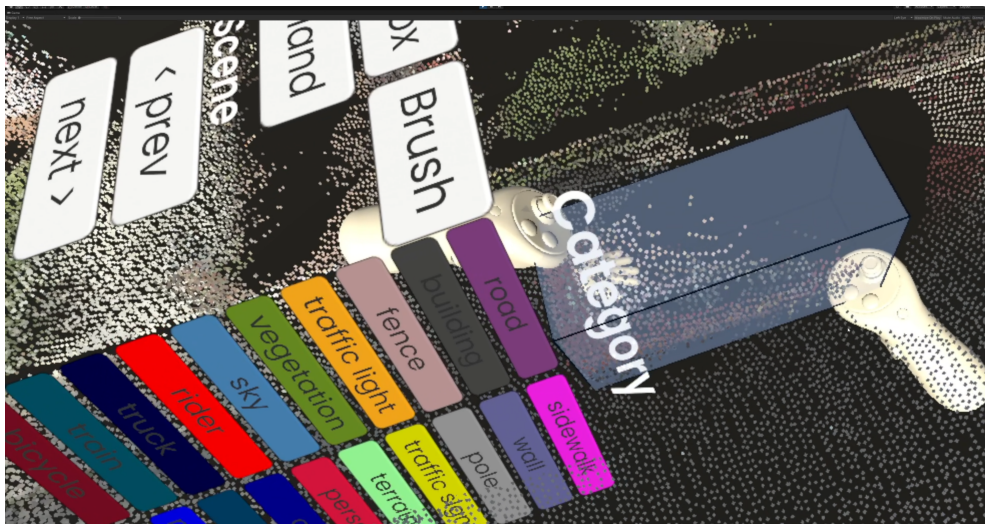


Figure 4.3: Bounding box mode

There were some interesting options for the bounding box tool, and we believe there is still some better way that we didn’t find. In our solution, the left hand defines the left bottom corner of the box and the right hand defines the opposite corner. The rotation of the box is dictated by the left hand. This causes that when controllers are misaligned, no box is visible, which is a confusing state for the user who selected bounding box. This could be improved and on top of it, all rotations except orthogonal to the ground plane could be locked, ideally on a button press, as generally rotated bounding boxes are sometimes needed, as was discussed in chapter 1.

Despite these minor drawbacks, defining bounding boxes felt fast compared to 2D methods. On that note, it might also be interesting to add a fine-tuning mode after placing the box. This would create a two stage process, which could help the precision quite a lot.

4.2.4 Brush tool

Seen in Figure 3.1, the brush tool became the dominant tool for our experiment, as we saw it as simpler and at the same time faster at transferring the knowledge of the labeler into the dataset. It is a sphere attached to the right controller, which can be enlarged / shrunk down by the right controller joystick.

It proved to be intuitive even for completely inexperienced users. The only thing we considered in this department was other brush shapes for labeling hard to get places, but that could be done with a small enough sphere brush. Inside the sphere, a small text is denoting which label is selected.

4.3 Architecture of the Unity solution

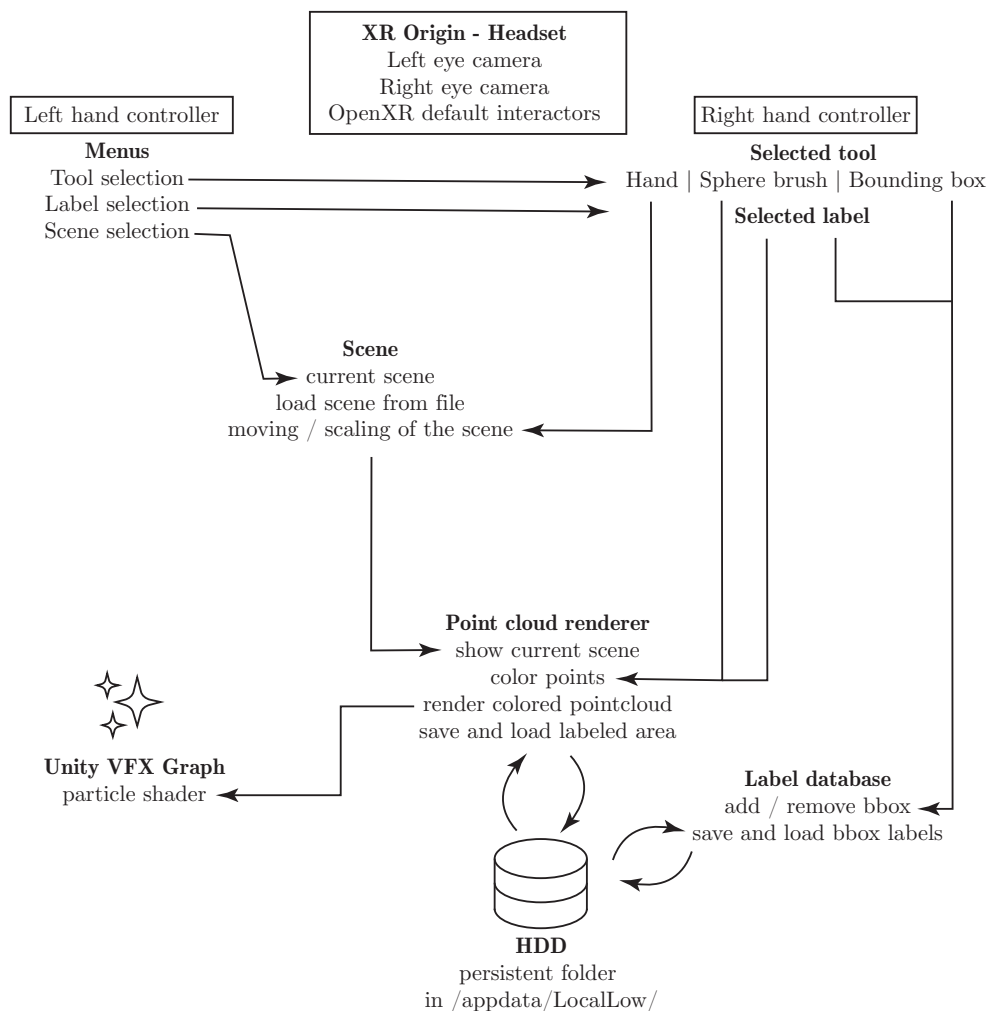


Figure 4.4: Top down view of the solution architecture

4.4 Main design problems

4.4.1 Movement

We considered multiple options for movement. Our first idea was to let the user move around an otherwise static scene. This is supported by default in all VR applications as if moving the user’s head / body wouldn’t properly change their viewpoint in the scene, motion sickness would be almost instant (tested this hypothesis on myself). If the user doesn’t have a large area dedicated for physical movement while in VR, this limits the reach quite drastically.

The typical ways of extending the users movement are:

- **teleportation** - using the controller, the user can point and click to teleport. This is the simplest and most common way of movement in VR applications. From experience, it isn’t as intuitive and requires some getting used to.
- **smooth locomotion** - using a joystick on the controller, you can go forwards / backwards / sideways more similar to moving around in screen and keyboard environment. This is much more motion sickness inducing compared to teleportation as during movement, the movement of the scene doesn’t correspond to the movement the body is experiencing.

After testing out both options, we weren’t happy with neither, and instead turned to moving the labeled object relative to the more or less stationary user. This solution proved to be quite intuitive and fast, so we chose it as the final solution. It also allows to see the scene at different scales and angles, which helps the user recognise objects better. The only drawback of this might be that rapid change of scale / refocusing eyes on different depths fast can cause fatigue or motion sickness.

4.4.2 Data loading and saving

The program is fully prepared for loading custom data with a custom label set. To load a custom pointcloud, a python script is provided, which chops up the point cloud into regions so that it will fit into the constraints of the program. These are dependant on the computer and most influenced by the performance of the GPU. A computer with a GEFORCE GTX 1660 SUPER can handle about 500 000 points in one pointcloud interactively. If the user has different computing capabilities, changing the size of the regions in preprocessing will give the biggest improvement.

Required data format

The preprocessing script requires the point cloud to be stored in a file with the following format

$$x, y, z, r, g, b$$

where one row = one point, xyz are the spatial coordinates of the point and rgb are the colors (optional, coloring point clouds is quite difficult and requires synchronization of rgb cameras with the lidar, which is monochromatic)

All the regions are automatically aligned to the origin of the Unity scene by moving the centroid of all the points to 0,0,0 and giving all points the same transformation. This is done for cases where the original point cloud's origin would be off-screen.

The resulting sectioned point cloud must be moved to the application folder.

To load custom labels, they must be in the following format, and inside the label directory.

Saving work

During labeling, the work is auto-saved every 10 seconds. This auto-save frequency can be modified but as the save files are relatively small and saving them uses parallelism, It is advised to leave the interval at 10 seconds. This functionality was used to track progress for the participants of the user study, effectively sampling their productivity every 10 seconds.

Upon exiting and reentering the application, the last saved labels are loaded for each scene, so the user can easily hop back into his previous work.

4.4.3 UI elements in 3D

UI becomes a relatively complex discipline in VR compared to traditional 2D layouts that we are used to. It enables some exciting new options such as 3D icons, which we used for the tools.

This is combined with classical 2D UI for the hand menu and label selection, where additional raycasting logic has to be used, but thankfully the OpenXR library has a very friendly API for such solutions as they are required for essentially every VR application.

As VR controllers are very different, building functionality around specific features of the controllers is a little risky as it can impact compatibility. Every controller on the market has a main “trigger” button - that is why we chose to map most functionality on this one button, from drawing, to moving, to selecting in the menu - this best emulates using a mouse. For changing the brush size, we chose the joystick, which is also usually present in some form, either as a physical joystick or a capacitive touchpad.

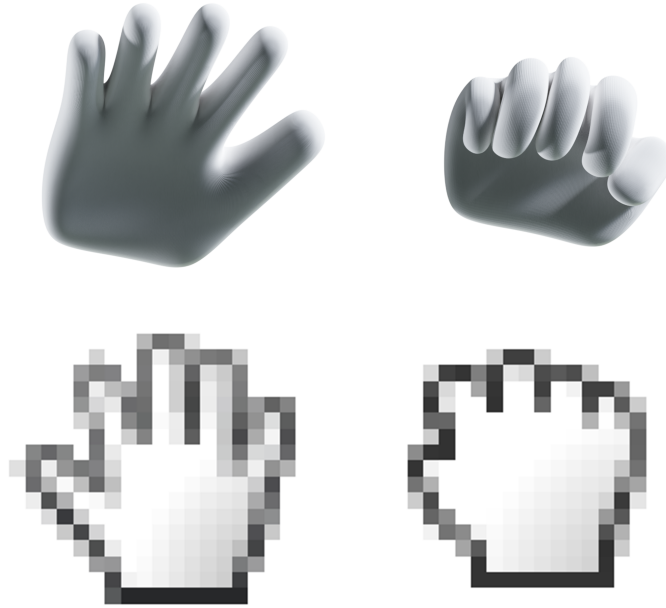


Figure 4.5: 3D hand icons and their 2D cousins

4.4.4 Point cloud rendering

Another interesting part of our solution is the point cloud rendering. In (Zama Ramirez et al. [2020]) point clouds were displayed by turning them into meshes with voxelization. Combining this with LOD's and other mesh optimizations, this is a solid option, but from our testing, using particles was more performant. Unity provides two types of particle systems, the older CPU based one and a newer, GPU based one inside the VFX graph. We tried the CPU one first, but in the end learned to utilize the VFX graph. The performance boost was quite significant, enabling about 3x more particles to be rendered on our hardware.

While coloring the point cloud with labels, a rather ineffective update of the entire particle system is performed. Slicing the particle system into multiple, using octrees or a simple grid, would increase the performance of this operation drastically. We didn't implement this optimization as the renderer itself, without updating, already occupies most of the GPU's memory, so alleviating this bottleneck would allow only a small amount of performance to be gained.



Figure 4.6: User inside the labeling environment

5. Comparing the new method to industry standard

In this ultimate chapter, we will confront our solution with reality and with other labeling programs. The main findings will be then summarized in the conclusion.

5.1 Setup of the user study

The participant group was not very age-diverse as all participants were $22 \pm \epsilon$. The group consisted of two women and four men including myself. As I developed the application and had a lot of training beforehand, my result won't be included in the analysis of the user behaviour. It will be discussed in the context of "what speed / precision is achievable with more experience with the tool".

All participants were instructed to sit down and adjust the VR headset to comfort. Upon entering the virtual reality, the users saw an empty podium surrounded by a mountain environment, just to be grounded in space and not in a white room. They could get familiar with the controllers and upon pressing "next scene" they would be introduced to the first scene. I sat beside each participant, using a second monitor to see their view and guided them verbally through the process.

Some participants were unsure at first, but once they selected the brush tool and labeled the first few points, intuition kicked in and the scenes started to take color rather quickly. Most users needed some time to grasp moving the scene around, but quickly learned to use the left hand to move the scene and the right hand to label. Four of the six users even figured out a strategy that we didn't think of before - moving the scene with one hand while labeling it with the second. This reminded me of an idea shared in (Zhao et al. [2022-07]):

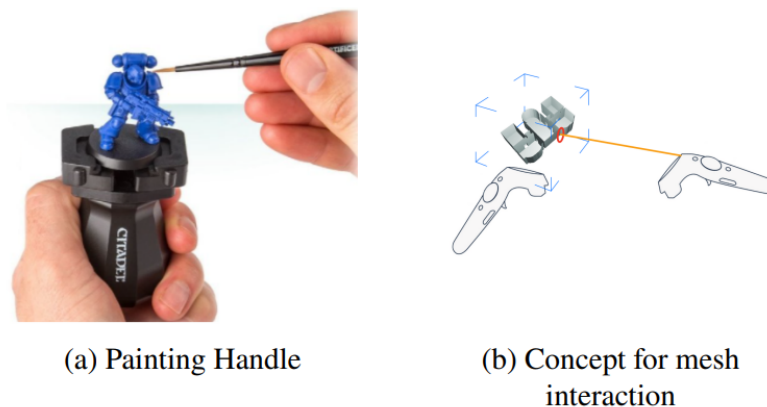


Figure 5.1: Two hand interaction idea from SemDpray (Zhao et al. [2022-07])

During each labeling session with a participant, the progress was saved every 10 seconds. This created a rich set of data displayed on the following two pages.

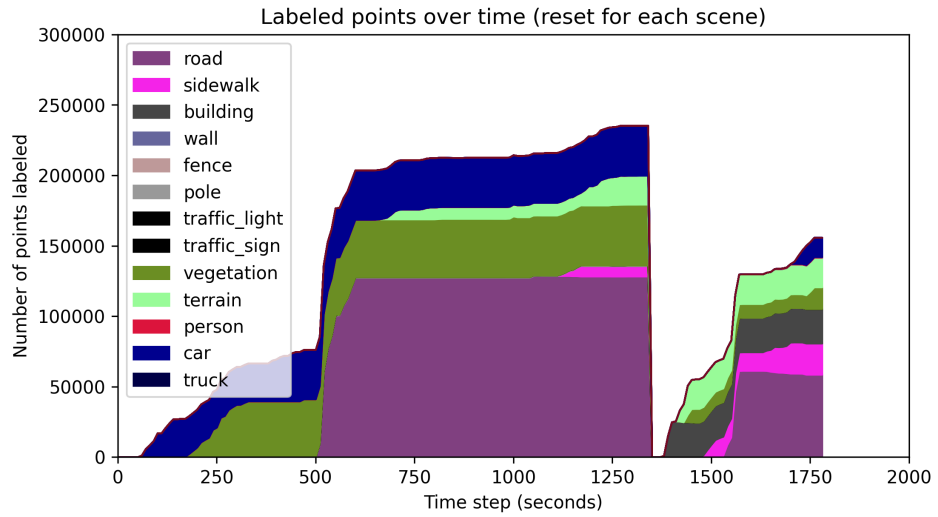


Figure 5.2: Subject #1

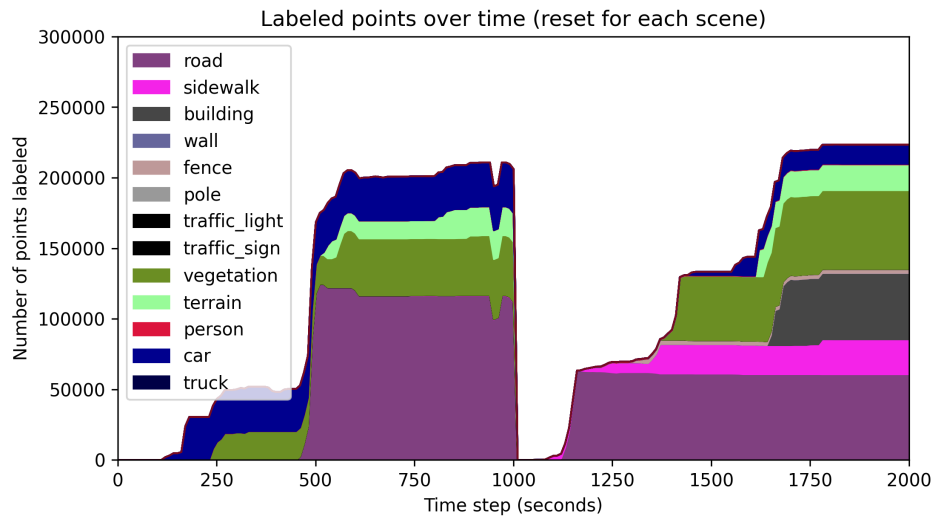


Figure 5.3: Subject #2

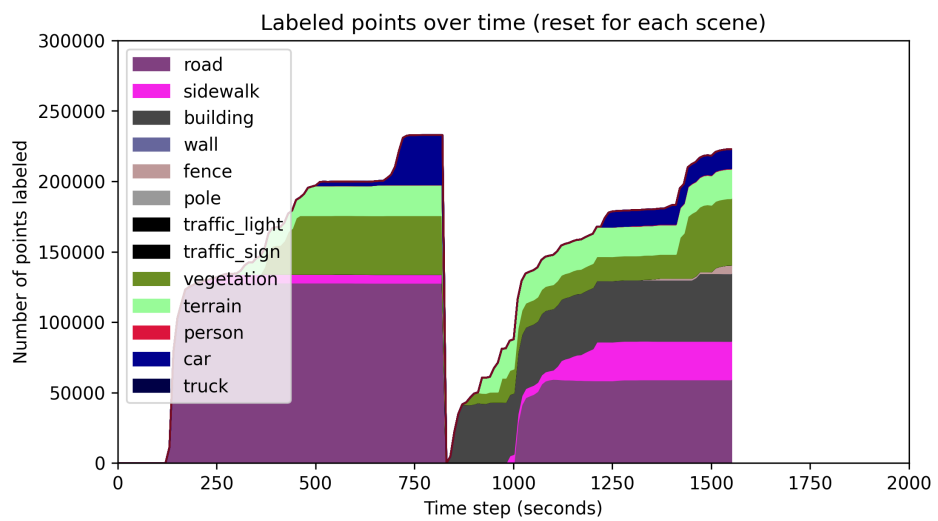


Figure 5.4: Subject #3

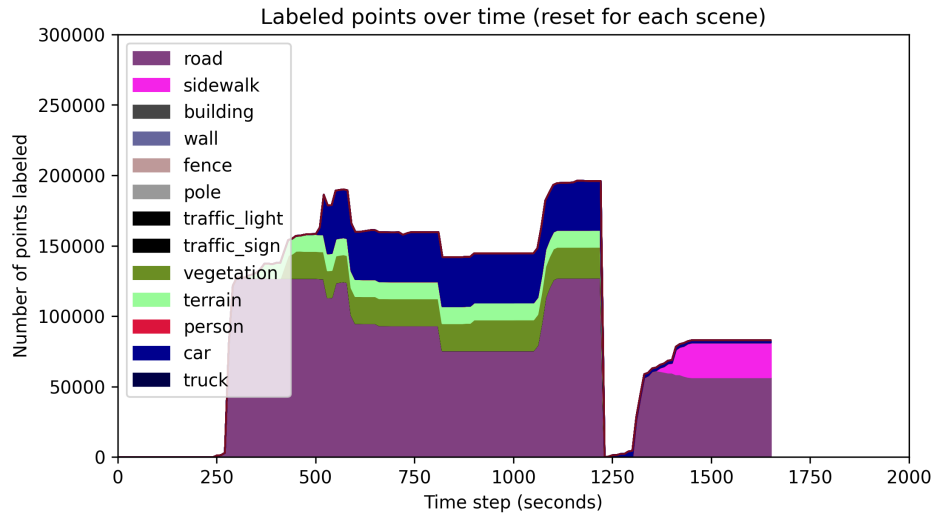


Figure 5.5: Subject #4

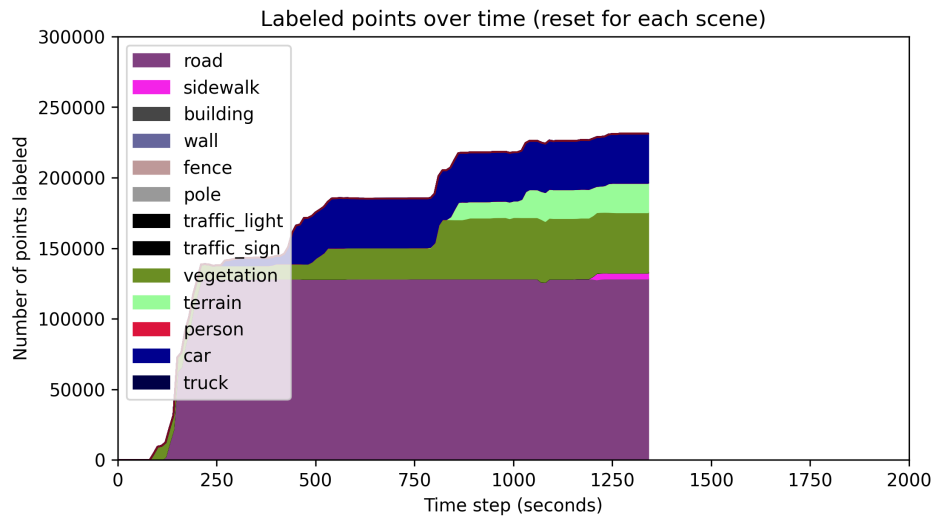


Figure 5.6: Subject #5

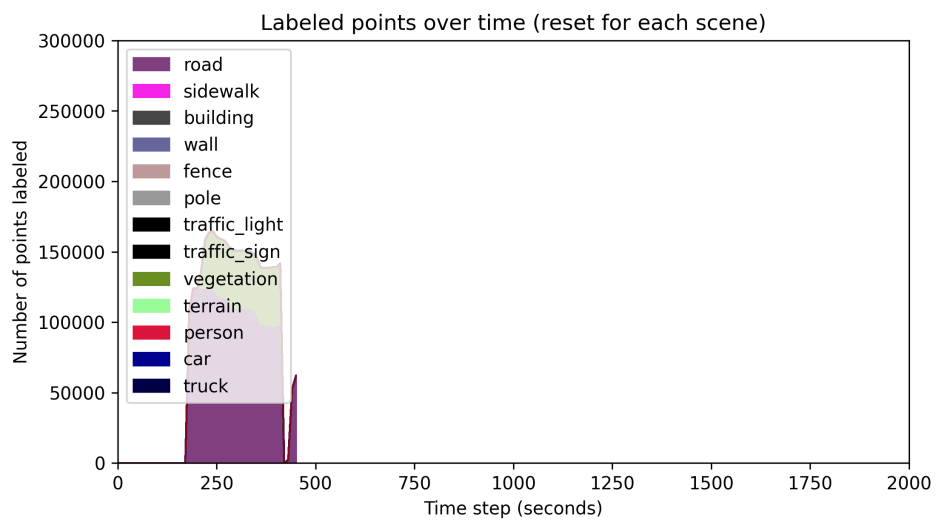


Figure 5.7: Subject #6

All the graphs are rectified to the same range on both axes to make all sessions comparable. The Y axis is the number of **correctly** labeled points. This counter is reset when the user switched to another scene - for this comparison, only the performance on the first two scenes was measured. From the graph, it is for example easy to see how the class road was the easiest to label - most users started with this category and labeling it caused a fast spike in correctly labeled points. Other classes had a slower trajectory as they required more careful labeling.

Ignoring color in these graphs, we can see how productivity evolved. A surprising result to me is how fast all participants were in starting to be productive. Some plateaus can be seen in the graphs - these were usually caused by labeling areas incorrectly (most common mistake was sidewalk / road) which didn't contribute to the total area labeled. For the same reason, the graph decreased sometimes, but without any interference, participants usually realized their mistake and later fixed incorrect labeling. Subject #6 skipped through the first two scenes quite quickly, that is the reason for their graph's fast cutoff.

5.1.1 Productivity of an experienced user

Alongside with the relatively short user sessions, we conducted a full run of labeling the whole batch, which was performed by the author of the labeling program, who practiced labeling before doing this run. This was done to emulate a more realistic labeling scenario - someone with technical ability, who had some time to get used to the tool and get a sense of the dataset. This is the run in its full glory:

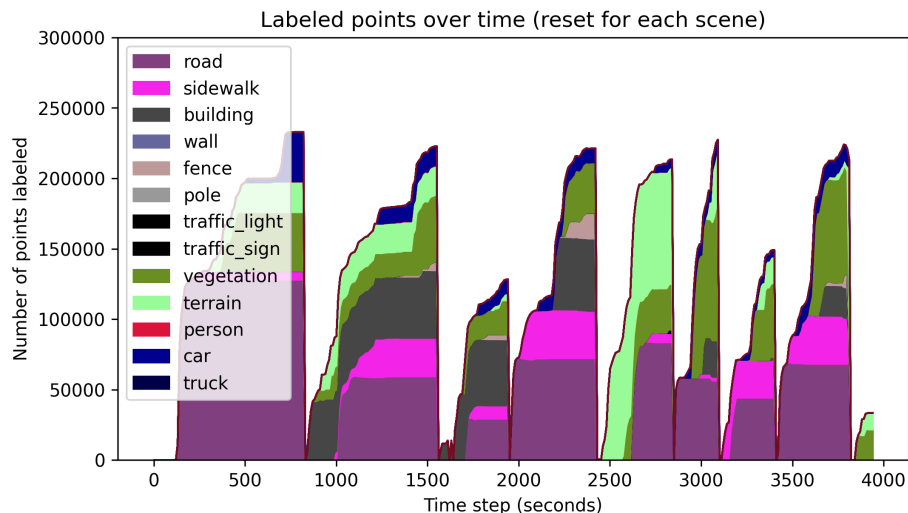


Figure 5.8: A full batch labeling session

There can even be seen some improvement in speed during this experiment - the later scenes are labeled faster than the initial ones, hinting at a "getting into the zone" effect. In the following sections, we will break down this labeling run. It would be best to conduct this run multiple times to see if the speed can be improved further and mainly to make the result more statistically significant.

5.2 Speed & precision comparison

Comparing speed and the precision of VR labeling to keyboard + screen labeling is the main result of this thesis, but there are many asterisks to this comparison which need to be commented.

Starting with the raw result, we were able to label an entire batch of the KITTI 360 dataset in 65 minutes. In the paper, the authors say that labeling a batch took around 3 hours on average. This gives us a very hopeful result, but the comparison is not entirely fair for the following reasons.

5.2.1 Precision of VR labeling

My labeling of the scene was in a 88.43% agreement with the KITTI dataset. It is not exactly the same as accuracy as it is possible there were mistakes in the KITTI labeling. We are certainly not accusing the KITTI researchers of hiring sloppy labelers, the argument aims more in the direction of questioning the objective truth. Where does the label "vegetation" end, and where does "terrain" begin on an overgrown hill? This is a very interesting question that reveals a third "lie" in the ground truth. The lie of language itself. Many aspects of the environment around us are manifestations of continuous spectrums and imposing the discrete data structure of language is a lossy operation.

It is interesting to see the breakdown of the mistakes:

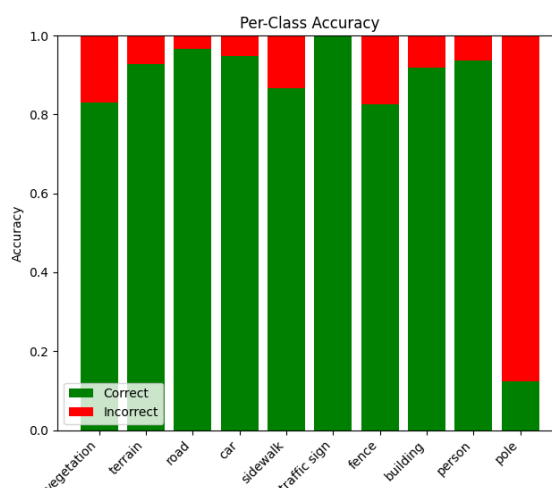


Figure 5.9: Accuracy of a VR labeling session compared to KITTI 360

The biggest mistake in my labeling was incorrectly determining vegetation from terrain. This isn't clear from the figure, but stems from the fact that these classes have much more points in the point cloud compared to e.g. class "pole". The most important classes, person, car and road, which are also easier to exactly separate from other objects, have a better accuracy - 93.69%, 94.76% and 96.54% respectively.

The accuracies of other users were slightly worse, but all hovered around the 85% figure. The reason for the higher accuracy of my run was extensive training on other data beforehand to get a grasp of the dataset, which wasn't feasible for the other users.

5.2.2 Difference of our labels and the KITTI 360 labels

In the mentioned 65 minutes, only a semantic segmentation of the point cloud was created. This is the most time-consuming part, but it further complicates a direct comparison to the KITTI 360 dataset as they created a panoptic segmentation in the 3-hour time. It seems like adding an instance segmentation to reach panoptic wouldn't take another 2 hours, but as we didn't test it, we have to include this discrepancy.

Another difference is the lack of dynamic labels in our labeling. The KITTI 360 dataset includes dynamic objects that moved through the batch as it was being captured. This too happened inside the 3 hours on the KITTI side, further explaining the large difference.

5.3 Shortcomings of VR labeling

Prolonged sessions in VR are known to cause "VR sickness", this is commented for example in (Zhao et al. [2022-07]), where they claim about 30 minutes is the maximum amount of time spent inside the headset before some of these effects start taking place. This was confirmed in our small experiment, where most of the participants responded that besides a slight "adjustment back to reality" after taking the headset off, they felt fine in their ≤ 30 min sessions. One respondent reported nausea, ending the experiment sooner because of it. I myself spent about 65 minutes in the headset and experienced strong eye strain, needing to take a break from all screens for some time, but didn't experience much "VR sickness", believing that prior long exposure to VR helped me build up immunity. Taking tiny breaks with taking the headset off for a minute seemed to help quite a bit.

Unfortunately, discomfort is still the limiting factor of this entire endeavor. As this problem taunts the entire VR space today, we believe there are strong incentives to build more comfortable headsets and this problem will fade over time. But that is only speculation now and it is yet unclear what level of comfort is achievable in the coming years.

5.4 Future work

The bounding box feature wasn't properly tested in the user study for the time constraints, and this would be certainly good to explore as seeing users trying to use it could spur new design ideas. Additionally, to match the KITTI level, dynamic scenes should be handled too. We could imagine a slider in front of the user for scrubbing through the dynamic point cloud.

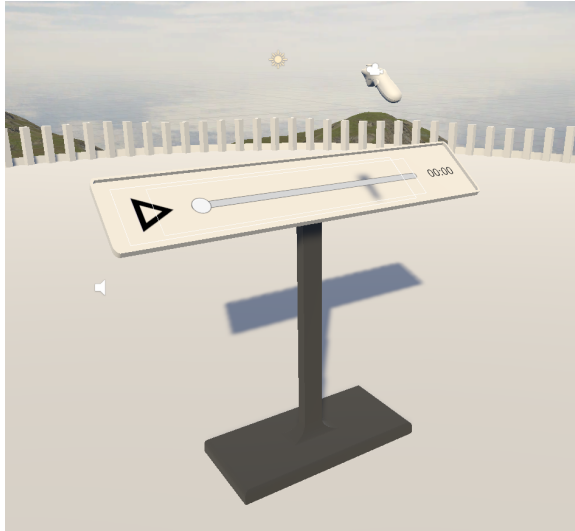


Figure 5.10: Idea for a dynamic point cloud scrubbing player

The data loading could be expanded to allow more file formats and besides point clouds support also meshes. Some intuitive user interface for loading and saving data would be ideal, but it would have to be decided whether to implement this pre-VR session in classical 2D UI or somehow within VR, which would be better but potentially complicated.

Conclusion

We introduced a variation on a VR labeling program similar to (Zama Ramirez et al. [2020]) and (Zhao et al. [2022-07]). We studied the efficiency of such a tool and found promising results regarding the speed and precision of this type of labeling.

We conclude that within a larger development effort, the proposed program could be expanded into an industry-grade labeling tool. The main problem to solve would be comfort. On the tested hardware, labeling sessions lasting over 30 minutes caused discomfort to the point of losing the competitive edge to traditional, slower, but more comfortable screen-and-keyboard labeling.

From this we conclude that further success in this area relies on an improbable, but possible future development of a breakthrough in VR comfort.

Furthermore, we strove to explore alternatives and objectively compare them. We hope someone in the practice of dataset creation might come across this thesis and find some of the ideas inspiring. We identify creating our own dataset as a good possible next step for our research. In that context, the theory of labeling discussed here would be put to test and the usefulness of VR and other methods would be decided by the engineering process.

Bibliography

- Luis Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science (New York, N.Y.)*, 321:1465–8, 09 2008. doi: 10.1126/science.1160379.
- Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences, 2019.
- Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020.
- F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau. Accurate 3d car pose estimation. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3807–3811, 2016. doi: 10.1109/ICIP.2016.7533072.
- Chakravarty Chaitanya, Anton Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics*, 36:1–12, 07 2017. doi: 10.1145/3072959.3073601.
- Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps, 2019.
- Liang-Chieh Chen, Sanja Fidler, Alan L. Yuille, and Raquel Urtasun. Beat the mturkers: Automatic image labeling from weak 3d supervision. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3198–3205, 2014. doi: 10.1109/CVPR.2014.409.
- Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- Salehe Erfanian Ebadi, You-Cyuan Jhang, Alex Zook, Saurav Dhakad, Adam Crespi, Pete Parisi, Steven Borkman, Jonathan Hogins, and Sujoy Ganguly. Peoplesanspeople: A synthetic data generator for human-centric computer vision, 2022.

- Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.
- Martin N. Hebart, Adam H. Dickter, Alexis Kidder, Wan Y. Kwok, Anna Corriveau, Caitlin Van Wicklin, and Chris I. Baker. Things: A database of 1,854 object concepts and more than 26,000 naturalistic object images. *PLOS ONE*, 14(10):1–24, 10 2019. doi: 10.1371/journal.pone.0223792. URL <https://doi.org/10.1371/journal.pone.0223792>.
- Doug Hudgeon and Richard Nichol. Machine learning for business: Using amazon sagemaker and jupyter, 2020. URL <https://aws.amazon.com/sagemaker/data-labeling/pricing/>.
- Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.
- Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d, 2022.
- Chang Liu, Yujie Zhong, Andrew Zisserman, and Weidi Xie. Countr: Transformer-based generalised visual counting, 2022.
- Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition*, 110:107332, 05 2020. doi: 10.1016/j.patcog.2020.107332.
- Martin Mirbauer, Miroslav Krabec, Jaroslav Křivánek, and Elena Šikudová. Survey and evaluation of neural 3d shape classification approaches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8635–8656, 2022. doi: 10.1109/TPAMI.2021.3102676.
- Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5632–5640, 2016.
- Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games, 2016.
- Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

- Matthias Straka, Stefan Hauswiesner, Matthias R  ther, and Horst Bischof. Skeletal graph based human pose estimation in real-time. In *British Machine Vision Conference*, 2011.
- Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The replica dataset: A digital replica of indoor spaces, 2019.
- Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization, 2018.
- Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.
- Yashan Wang, Yue Zhang, Yi Zhang, Liangjin Zhao, Xian Sun, and Zhi Guo. Sard: Towards scale-aware rotated object detection in aerial imagery. *IEEE Access*, PP:1–1, 11 2019. doi: 10.1109/ACCESS.2019.2956569.
- Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, pages 75–82, 2014. doi: 10.1109/WACV.2014.6836101.
- Aoran Xiao, Jiaying Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Transfer learning from synthetic to real lidar point cloud for semantic segmentation, 2021.
- Pierluigi Zama Ramirez, Claudio Paternesi, Luca Luigi, Luigi Lella, Daniele Gregorio, and Luigi Di Stefano. Shooting labels: 3d semantic labeling by virtual reality. pages 99–106, 12 2020. doi: 10.1109/AIVR50618.2020.00027.
- Yiming Zhao, Cyprien Fol, Yuchang Jiang, Tianyu Wu, and Iro Armeni. Semd-pray: Virtual reality as-is semantic information labeling tool for 3d spatial data. In Lavinia Chiara Tagliabue, Athanasios Chassiakos, Daniel Hall, Dragana Nikolic, and Ranjith Kuttantherappel Soman, editors, *Proceedings of the 2022 European Conference on Computing in Construction*, pages 590 – 597, Turin, 2022-07. Universit   degli Studi di Torino. ISBN 978-88-7590-226-1. doi: 10.3929/ethz-b-000586092. European Conference on Computing in Construction (2022 EC³); Conference Location: Rhodes, Greece; Conference Date: July 24 –26, 2022; Presentation held on July 26, 2022.

List of Figures

1.1	Weak labeling program reCAPTCHA	4
1.2	2D Bounding box labels in YOLO v7 (Wang et al. [2022])	5
1.3	Rotated bounding boxes in aerial photo object detection (Wang et al. [2019])	6
1.4	Difference between a 2D and a 3D bounding box	6
1.5	Example of an uneven ground requiring generally rotated bounding boxes (Chang et al. [2019])	7
1.6	3D human pose labels (Straka et al. [2011])	7
1.7	Example of a car pose (b) (Chabot et al. [2016])	7
1.8	Types of segmentations	8
2.1	Pricing of labeling data on Amazon ()Hudgeon and Nichol [2020])	10
2.2	Snake Ouroboros eating its own tail - Mohamed Ibrahim, public domain	12
2.3	Sample from the PeopleSansPeople dataset for detecting humans in images Ebadi et al. [2022]	14
2.4	Datasets approximate truth	14
2.5	The gap between synthetic and real	15
2.6	Amount of supervision per labeling strategy	15
2.7	Pascal labeling tool	17
3.1	Knowledge flowing from human into the dataset	18
4.1	User’s view into the VR labeler	22
4.2	Valve Index controller (source: Valve)	23
4.3	Bounding box mode	24
4.4	Top down view of the solution architecture	25
4.5	3D hand icons and their 2D cousins	28
4.6	User inside the labeling environment	29
5.1	Two hand interaction idea from SemDpray (Zhao et al. [2022-07])	30
5.2	Subject #1	31
5.3	Subject #2	31
5.4	Subject #3	31
5.5	Subject #4	32
5.6	Subject #5	32
5.7	Subject #6	32
5.8	A full batch labeling session	33
5.9	Accuracy of a VR labeling session compared to KITTI 360	34
5.10	Idea for a dynamic point cloud scrubbing player	36
5.11	Overview of the file structure provided with this thesis.	42

Appendix - file structure

In the following figure a map of the most important files in the provided .zip is shown.

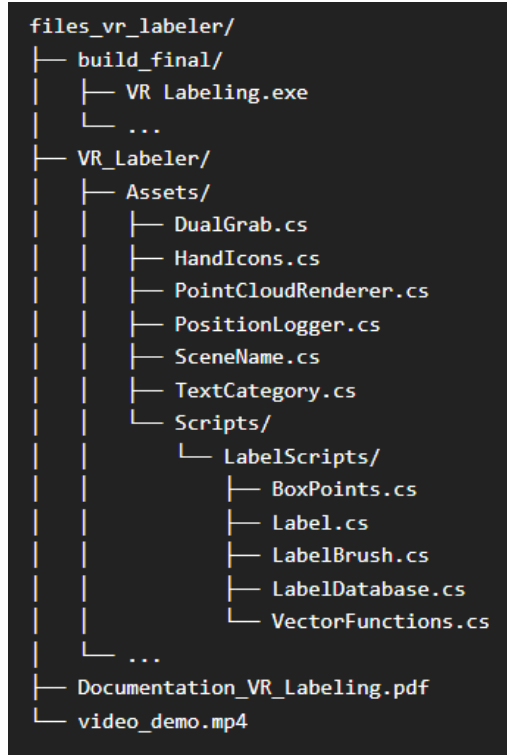


Figure 5.11: Overview of the file structure provided with this thesis.

Besides the Unity project inside VR_Labeler with the most important classes of the program shown in the figure above, an .exe build is provided in build_final/ and a short video of the program is provided in the root folder and the program documentation. The video is added for a quick overview of the solution without requiring the proper hardware to run the program.