FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Semi-Automatic Video Annotation Tool for Generation of Ground Truth Traffic Datasets

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Florian Groh, BSc
Matrikelnummer 01168186

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao.Univ.Prof. Dipl.-Ing. Mag. Dr. Margrit Gelautz

Wien, 7. Juli 2020

Florian Groh	Margrit Gelautz

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Semi-Automatic Video Annotation Tool for Generation of Ground Truth Traffic Datasets

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Florian Groh, BSc
Registration Number 01168186

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Mag. Dr. Margrit Gelautz

Vienna, 7th July, 2020

Florian Groh

Margrit Gelautz

Erklärung zur Verfassung der Arbeit

Florian Groh, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 7. Juli 2020

Florian Groh

Danksagung

Ich möchte mich bei meiner Betreuerin Margrit für ihre Geduld und Unterstützung im Laufe der Entstehung dieser Diplomarbeit bedanken. Ein großer Dank geht auch an Lisa, die immer an meiner Seite steht. Vielen Dank an meine Familie für die emotionale und finanzielle Unterstützung, welche mir ermöglicht hat all meine Interessen zu verfolgen.

Diese Diplomarbeit wurde von dem Projekt CarVisionLight (Proj.-Nr. 861251) unterstützt, welches von der Forschungsförderungsgesellschaft (FFG) unter dem Programm „IKT der Zukunft“, einer Initiative des Bundesministeriums für Verkehr, Innovation und Technologie (BMVIT), gefördert wird.

Acknowledgements

I would like to thank my supervisor Margrit for her patience and support during the development of this thesis. A big thanks also goes to Lisa, who is always at my side. Many thanks to my family for their emotional and financial support, which has enabled me to pursue all my interests.

This diploma thesis was supported by the project CarVisionLight (Project No. 861251), which is funded by the Austrian Research Promotion Agency (FFG) under the programme "IKT der Zukunft", an initiative of the Federal Ministry of Transport, Innovation and Technology (BMVIT).

Kurzfassung

Im Rahmen dieser Diplomarbeit wurde ein semi-automatisches Annotationstool („CVL Annotator“) für die Generierung von bounding box ground truth Daten in Videos entworfen, implementiert und evaluiert. Das Ziel ist die Annotation von nächtlichen Verkehrsszenen, welche in öffentlich vorhandenen Referenz-Datensätzen nur in beschränktem Ausmaß enthalten sind. Effiziente semi-automatische Annotation bildet eine wichtige Grundlage für die Neuentwicklung von Verfahren des maschinellen Lernens, welche große Mengen an ground truth Daten zum Trainieren und Testen der Netzwerk-Architekturen benötigen. Eine am Beginn der Arbeit durchgeführte Literaturrecherche dokumentiert, dass insbesondere ein Mangel an anspruchsvollen nächtlichen Verkehrsvideos mit hochdynamischen Lichtverhältnissen inklusive Reflexionen von Scheinwerfern und Lichthöfen des Gegenverkehrs besteht. Weiters wird aufgezeigt, dass bestehende Annotationstools zumeist nur auf lineare Interpolation als Unterstützungsmechanismus für die manuelle Annotation setzen. Im Gegensatz dazu wird das neu entwickelte Annotationstool „CVL Annotator“ mit einer Reihe verschiedener state-of-the-art Tracking-Algorithmen ausgestattet. Die Auswahl der Tracker erfolgt auf Basis einer quantitativen Analyse der Algorithmen unter Verwendung eines bestehenden synthetischen Datensatzes. Die Benutzeroberfläche wurde mit der Prämisse, Benutzerinteraktionen zu minimieren und alle für den Benutzer relevanten Informationen auf einen Blick zu visualisieren, entwickelt. Es wurde eine Vorstudie zur Benutzbarkeit durchgeführt, welche das neu entwickelte Annotationstool mit einem bereits veröffentlichten Annotationstool („Scalabel“) vergleicht. Dabei wurden insbesondere der Zeitaufwand und die Anzahl der benötigten Klicks, die für die Erstellung von ground truth Annotationen von Videoverkehrsszenen erforderlich sind, ermittelt. Zusätzlich wurde die Genauigkeit der Annotationsergebnisse verglichen. Es konnten sowohl in der Zeit- und Klickanalyse, als auch in der Genauigkeitsstudie Verbesserungen erzielt werden.

Abstract

In the context of this diploma thesis a semi-automatic annotation tool ("CVL Annotator") for the generation of bounding box ground truth data in videos was designed, implemented and evaluated. The goal is the annotation of night-time traffic scenes, which are only to a limited extent contained in publicly available reference datasets. Efficient semi-automatic annotation is an important basis for the development of machine learning methods that require large amounts of ground truth data for both training and testing of network architectures. A literature review conducted at the beginning of the thesis documents that there is a particular lack of sophisticated night-time traffic videos with highly dynamic lighting conditions including reflections from headlights and halos of oncoming traffic. Furthermore, it is shown that existing annotation tools mostly rely only on linear interpolation as a support mechanism for manual annotation. In contrast, the newly developed annotation tool "CVL Annotator" is equipped with a number of different state-of-the-art tracking algorithms. The selection of the trackers is based on a quantitative analysis of the algorithms using an existing synthetic dataset. The user interface was developed with the premise of minimizing user interaction and visualizing all information relevant to the user at a glance. A preliminary user study was conducted, comparing the newly developed annotation tool with an already published annotation tool ("Scalabel"). In particular, the time required and the number of clicks needed to create ground truth annotations of video traffic scenes were determined. Additionally, the accuracy of the annotation results was compared. Improvements could be achieved in the time and click analysis as well as in the accuracy study.

Contents

CHAPTER 1

Introduction

Modern cars are equipped with an abundance of assistive technologies ranging from Anti-lock Braking System (ABS) to Lane Keeping Assist (LKA), Advanced Front-lighting System (AFS) and more. Over time, the car industry has added so many sophisticated electronic systems to their vehicles, such that we are now at a point where one can argue, that car manufacturers have pivoted to the "PC-on-wheels" business, rather than their actual car business. This diploma thesis is carried out as a part of the CarVisionLight (CVL) Project, which aims to develop a smart AFS using stereo vision and adaptive headlights for "glare free high beam control". The envisioned system adaptively illuminates the scene in front of the car so that most of the scene is illuminated by the bright, long distance, lights (high beams) and only the parts where traffic participants might be blinded are left unlit. In the following section, we will give a more detailed definition of the problem we are trying to solve, and explain our motivation for trying to solve it.

1.1 Motivation and Problem Statement

For a car to be able to have an intelligent headlight system, that adapts to the world around it, the control algorithms need to understand the scene in front of the car. In other words, the system has to identify Areas of Interest (AOI) (cars, trucks, motorbikes, humans, lanes or similar) in its sensor data in an automated and quick fashion. Speaking in terms of the computer vision field, we are dealing with *object detection* and *object tracking*. Both detection and tracking have been challenging and interesting problems in the field of computer vision and a variety of different approaches have been developed for their specific use cases [AT13].

These approaches can vary in complexity and can be as simple as grayscale thresholding (all pixels above a threshold value are counted as significant, the rest is ignored), which might be sufficient in a controlled environment such as a production line where the algorithm could be used to count parts. Or they can be more sophisticated algorithms

such as the "Scale-Invariant Feature Transform (SIFT)". For this algorithm, David Lowe [Low99] combined techniques like *Bayesian probability analysis*, *Hough transform voting* and *linear least squares* in order to find a mathematical representation of feature points in an image, which does not change under rotation, translation or even lighting changes. Another approach, which has gained a lot of popularity in recent years, is to gather large amounts of data, and apply machine learning algorithms to automatically identify features and patterns in order to solve the task of object detection and tracking. Machine learning approaches have risen in popularity in recent years. We can see a particular increase in interest ever since the first deep neural network called "AlexNet" won the ImageNet challenge of 2012 [KSH12]. The success of "AlexNet" has lead many computer vision researchers to focus on machine learning for solving various problems (classification, tracking, style transfer, etc.) on which they achieved great successes. One great advantage of machine learning compared to more classical computer vision approaches is that the authors of the algorithm do not have to identify formal rules or patterns, but instead can let the computer find these patterns in the data itself. But this advantage is also closely linked to the main disadvantage of machine learning: it requires a lot of data. The amount of available data is the most important factor which defines whether the machine learning algorithm can be successfully used to solve a task. That is because the algorithm can only derive its model and tune its parameters based on the data it is given. So it is important that the data covers as many aspects of the problem that needs to be solved as possible. A general rule of thumb is: the more complex the problem, the more data needs to be gathered for the learning algorithm to derive a successful model. Generating such data can be very tedious and time consuming. This is particularly true when dealing with what is called a *supervised* learning algorithm, for which every piece of training data has ground truth information attached to it (e.g. grayscale image of parts on conveyor with the correct number of parts shown) and the goal is to find patterns which lead from the given input data to the given ground truth. *Supervised* datasets generally need the involvement of humans, who create the ground truth data attached to the raw sensor data. In contrast, there also exist *unsupervised* learning algorithms, which do not need any ground truth information attached to the raw data. These algorithms, however, are less equipped to find a particular desired output (e.g. how many parts there are on the conveyor) but instead they can be used to find clusters with similar characteristics in the data, which can then be further analyzed by human analysts. So, in general, *unsupervised* algorithms are more suited for exploring a dataset instead of answering specific questions.

The CVL Project focuses on developing supervised machine learning algorithms to enable an adaptive headlight control system, where the ground truth data is labelled by humans to help the algorithm automatically find patterns in the data, which can be generalized for real-world applications.

To be able to generalize for real-world applications, the training data has to be representative of the kind of problems the algorithms have to solve in the future. Initial research into published automotive data sets for computer vision algorithms at the beginning of

this diploma thesis has shown that most of them ([GLU12][BFC09][COR+16][YXC+18]) do not include nighttime scenes, which are the main focus of the CVL Project. A notable exception are the SYSU and CUHK datasets [CHX+17], containing only nighttime scenes of still images as opposed to videos. There also is research into synthesizing these kinds of images, with datasets such as [RSM+16] or [RHK17]. Unfortunately, the cited datasets do not fit the needs of the CVL Project, as is further explained in Chapter 3 (Datasets). Thus, we deemed it necessary to create a dataset specifically for the needs of this project. During our work on the project and the thesis, many more datasets have been published, some of which include nighttime scenes with high temporal density (e.g. the BDD100K MOT extenstion to [YXC+18]), showing that interest into this topic is very high.

In order to create a ground truth dataset, one needs to use an appropriate tool to annotate the data at hand. Our research into published annotation tools found that most of the tools focus primarily on still image annotation or video annotation using linear interpolation. To reduce annotation time, we specifically address the incorporation of suitable tracking algorithms, which are able to cope with challenges such as high dynamic contrast, halos or reflections, which are typically present in automotive night scenes. Chapter 5 will go into more detail about our evaluation of tracking algorithms.

1.2 Contributions

The main contributions of this work are:

- We perform analysis of the current state of the art in autonomous vehicle datasets.
- We develop a semi-automatic annotation tool, CarVisionLight Annotator (CVLA), for bounding box ground truth generation in videos.
- We carry out a systematic evaluation of tracking algorithms for nighttime footage datasets.
- We conduct a preliminary user study, comparing the speed of annotation using our tool CVLA compared with an existing annotation tool.

1.3 Outline of the Thesis

In Chapter 1, we have given a short **introduction** and problem statement to explain the motivation behind creating a novel annotation tool with semi-automatic tracking capabilities to improve the process of creating an entirely new ground truth dataset from videos. In *Chapter 2*, we present some **related work** and discuss the current state of the art in the scientific community regarding semi-automatic annotation algorithms and publicly available video annotation tools. *Chapter 3* reviews publicly available **datasets**, their respective sizes and variability of scenery and annotation types. We also share details of our own CVL dataset with statistics on bounding box sizes and class

distributions. This is followed by *Chapter 4*, where we talk about the implementation of our **Annotation Tool** and the User Interface (UI) decisions we made to optimize the work flow of annotating videos. *Chapter 5* presents our **Tracker Evaluation**, in which we tested state-of-the-art tracking and propagation algorithms on synthetic nighttime road scenes to assess their applicability for our annotation tool. It also includes our **Preliminary User Study**, in which we evaluate our tool in comparison to a previously published tool using accuracy, time and click analysis as parameters. In Chapters 3, 4 and 5 we include excerpts from our paper "A tool for semi-automatic ground truth annotation of traffic videos", which was published by, and presented at Electronic Imaging 2020. In the end, we will present our final thoughts and provide an outlook on possible future work.

CHAPTER 2

Background and Related Work

This chapter looks into the current state of the art in literature regarding semi-automatic annotation algorithms and the published tools for annotation. We examine existing semi-automatic algorithms, designed to help with the annotation process, followed by a look at publicly available annotation tools, focusing especially on UI decisions and propagation capabilities.

2.1 Semi-Automatic Annotation Algorithms

This section provides an overview of selected current techniques for propagating sparse or dense image labels over time to minimize human workload during video annotation. Generally speaking, the best case scenario is working with ground truth data with the most accurate and dense type of annotation. This is why we first look at annotation algorithms based on segmentation data, in which a pixel-wise mask of an object is propagated over time. Figure 2.1 shows the segmentation of an example image. We first take a look at previous research using Cost Volume Filtering (CVF), followed by some deep learning approaches. Finally, we examine bounding box tracking algorithms, which work with much sparser data in the form of four coordinate values (top, left, width, height). This makes the task of propagating this information over time an easier task than updating pixel-wise masks for video annotation, with the disadvantage of losing some information density.

2.1.1 Semantic Segmentation

Cost Volume Filtering

In this section, we take a look at the work by Hosni et al. and its subsequent exentsion by Brosch et al. Hosni et al. have developed a classic computer vision algorithm suitable for many different labelling tasks called CVF. The paper shows promising results for

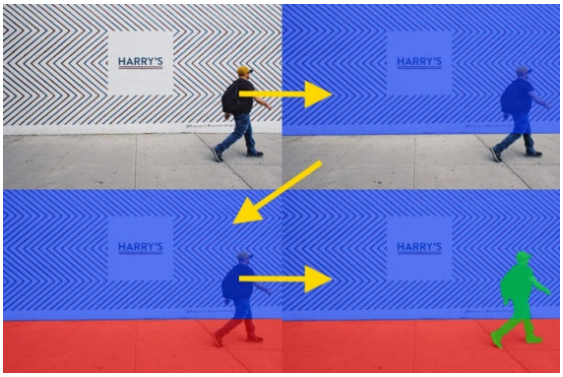

Figure 2.1: Example of pixel-perfect semantic segmentation. Background, floor and human in blue, red and green, respectively. Original Photo by Robert Bye (unsplash user @robertbye)

semi-automatic image segmentation using scribble annotations (see top row of Figure 2.2) by a human annotator [HRB+12]. Brosch et al. extended their work from an interactive image segmentation technique to the temporal domain, to support 2D to 3D conversion of videos [BSG16]. A general overview of how the CVF technique works for image or video segmentation can be found in [BHRG12], and can be summed up in three points:

1. Based on color models that were initialized through foreground scribbles, a cost map that contains each pixel's probability of belonging to the foreground is generated.

2. Smoothing the cost map with an edge-preserving filter [HST10], aggregates the costs across neighboring pixels with similar colors.

3. Finally, pixels are assigned to the fore- or background according to the smoothed costs.

To avoid flickering effects when applying this approach on a frame-by-frame basis, Brosch extended this technique to the temporal domain, by applying a 3D kernel during the filtering step, thus allowing for filtering not only in the x,y domain but also over time.

This approach leads to very good results for 2D to 3D conversion for videos where a clear color distinction between fore- and background areas can be made. Figure 2.2 shows an image from [BHRG12] where this technique works almost perfectly. However, when testing the technique on our nighttime road scenes, we found that the algorithm did not perform as well. The basic assumption that different objects have different color models associated with them, does not apply and therefore, in the case of our test images, the cost maps do not reflect the objects' borders, even after filtering. An example can be seen in Figure 2.3.

Figure 2.2: Example of nearly perfect 2D to 3D conversion using Brosch's extended CVF technique. Original image in Brosch's PhD Thesis [Bro16]

Figure 2.3: Left: a test image from our nighttime road scenes. Right: the filtered foreground region, showing that CVF in this case does not deliver meaningful results.

Deep Learning

Before taking a closer look at the different segmentation propagation techniques, we first want to provide some background information on deep learning:

Recent interest in deep learning started to grow around the year 2012, when AlexNet [KSH12], a deep Convolutional Neural Network (CNN), won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This was the first time that a CNN performed better than classical machine learning techniques and sparked the rebirth of AI research in the computer vision community. What first started as relatively simple task of image classification (the output of the network was limited to a single class correspondence vector) has soon evolved into more sophisticated techniques in which class labels where attached to multiple objects in the scene and finally every pixel in the image, i.e. semantic image segmentation. The first notable achievement in this area was the Region Convolutional Neural Network (R-CNN) architecture by Girshick et al. [GDDM14] which operates in a two-level fashion. On the first level a number of region proposals are extracted using any classic machine learning objectness classifier available and on the second level an image classifier CNN is used to assign classes to the regions. This two-level architecture was succeeded by a the Faster R-CNN architecture [RHGS15], which enables CNNs with shared weights to handle both region proposal and classification, therefore improving speed significantly. It also makes use of a Fully Convolutional Network (FCN), an invention of Long, Schelhamer and Darrell [LSD15] which proved to be a small breakthrough in the deep learning community, as it introduced deconvolutional layers,

which enabled efficient per-pixel outputs for the first time. This architecture then enabled deep learning networks to perform semantic segmentation.

For the purpose of propagating these pixel masks over time, the Densely Annotated VIdeo Segmentation (DAVIS) challenge [PPTM+16] was devised. It offers a set of 50 HD videos with temporally dense ground truth mask data for the main object in the scene. The goal of the challenge is to propagate a segmentation mask from the first frame of an image to the rest of the clip. Through this challenge, several interesting approaches have come forward such as:

- One-Shot Video Object Segmentation (OSVOS) by Caelles et al. [CMPT+17]
  The idea behind OSVOS is that first a parent network is trained offline to distinguish between foreground and background, and then a test network is trained online on the particular ground-truth/image pair of the object of interest to fine tune, *which* part of the foreground we are interested in. Training the test network on the ground truth of the first annotated image in the sequence lets it focus on the specific features found in the object of interest, enabling a further distinction between all of the foreground elements and the one foreground element of interest.
- Lucid Data Dreaming for Video Object Segmentation (Lucid) by Khorea et al. [KBI+17]
  The main idea behind this approach is that by using a single starting frame and synthesizing many plausible future video frames as a training set, it is possible to forego training on a large dataset such as ImageNet.
- Fast and Accurate Online Video Object Segmentation (FAVOS) by Cheng et al. [CTH+18]
  In this technique, the main idea is to track parts of the object of interest with a Siamese Tracker (see Chapter 5) to then create a fore- and background mask of these parts, which can later be combined.

At the time of testing, OSVOS was one of the best performing networks in the DAVIS challenge 2016 and the authors made the source code available to the research community, so we were able to perform some tests with it. Unfortunately, similarly to the CVF approach, we found that OSVOS does not perform well with limited visibility constraints as seen in Figure 2.4. The left side of the image shows the ground truth and the right side displays what was actually calculated by the network.

2.1.2 Bounding Box Tracking

Section 2.1.1 has indicated that our footage is too dark and contains too little texture for the tested semi-automated algorithms to distinguish between fore- and background pixels. We therefore decided to make a compromise in terms of potential accuracy vs. actual accuracy and looked at 2D bounding box trackers. In other words we found that although segmentation has the potential to provide pixel-wise masks, indicating exactly

Figure 2.4: (left) correct mask of oncoming traffic, (right) falsely propagated mask of oncoming traffic by OSVOS network.

which pixels are part of an object, 2D bounding box tracking, being constrained to an axis-aligned box, achieves better results on our predominantly dark footage with the disadvantage of usually including some parts of the background in the box.

Bounding box tracking is a widely researched field in the computer vision community, with benchmark challenges such as the Visual Object Tracking (VOT) Challenge [KML+16] or the Multiple Object Tracking (MOT) Challenge [DRM+19]. Participants of these challenges have come up with a variety of tracking algorithms ([DBKF19], [LYW+18], [LVČZ+17]) which we will take a closer look at in Chapter 5.

Compared to segmentation propagation, the pixels identified by a bounding box will not contain *all* of the pixels and *only* the pixels that belong to the object of interest unless the projection of the object on the image is a perfect box. Depending on the objective of the task at hand, it is important to choose the right bounding box placement and size. As one can see in Figure 2.5, a bounding box which only covers the back side of a preceding vehicle (blue) would be a bad idea for the CVL Project, since a major objective of the project is to not blind other traffic participants. Ideally, we would be using a best-fit 3D bounding box (magenta) to cover all of the important parts of the vehicle, while at the same time minimizing unwanted background pixels. A dataset which offers such bounding boxes would be the KITTI [GLU12] dataset. However, 3D bounding boxes increase the complexity of the tracking problem by adding another geometric dimension, increasing the chance of propagation errors. In the CVL Project we decided on the compromise of using 2D bounding boxes – hence keeping the number of dimensions to two while covering all of the visible pixels of the AOI (red) and accepting that some background pixels will also be covered by this box.

Given that the data recorded during the CVL Project is challenging for segmentation propagation algorithms such as CVF (section 2.1.1) or even deep learning approaches such as OSVOS (section 2.1.1) we have decided to take a deeper look into bounding box tracking algorithms, which will be further examined in Chapter 5.

2.2 Annotation Tools

In this section, we take a look at video annotation tools, published by the scientific community in order to increase the speed of video data ground truth annotation. We

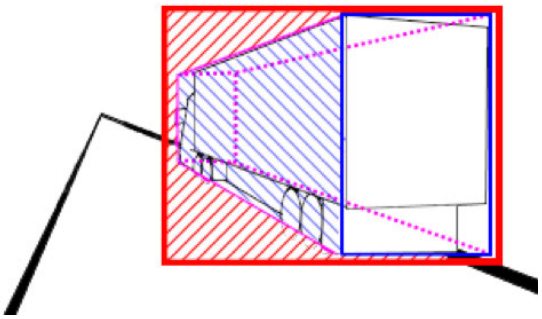

Figure 2.5: Different styles of bounding boxes: (magenta) best-fit 3D bounding box, with minimal background pixels, (blue) 2D bounding box covering parts of object, (red) 2d bounding box covering complete object, but also some area of background pixels.

look at platform choices, UI decisions as well as data propagation mechanisms used. Notable video annotation tools include VATIC [VPR13], ViTBAT [BNFD16], CVAT [Sek18], Scalabel [YXC+18] and BeaverDam [She16].

2.2.1 VATIC

The Video Annotation Tool from Irvine California (VATIC) by Vondrick et al. [VPR13] focuses on making annotations faster by optimizing the user interface for crowdsourcing services such as Mechanical Turk.

Regarding platform choice, the authors' plan was to work with crowdsourcing services, so they chose to build a web application with browser technologies as the user facing front end and a web server as the data keeping back end. The advatage of browser technologies is that they are easily deployable to a wide range of users across the world, with the disadvantage of browsers not having access to the same capabilities as native applications.

Vondrick et al. have tested different UI-choices through a series of user studies using linear interpolation as a data propagation mechanism. They discovered that users tend to annotate videos faster when presented with pre-defined evenly placed keyframes to annotate videos instead of letting them decide where to place the keyframes themselves. At a first glance, this might seem counterintuitive, since evenly pre-defined keyframes need to be placed in a frequency that accounts for the fastest changes in motion in the video over the course of the whole video, whereas user-defined keyframes can be adjusted to the movement needs of the subjects and timings in the video. For example, a car moving in a straight line at constant speed only needs to be annotated at the start and end of a sequence. Contrarily, a pedestrian who changes speed and direction frequently would require keyframes placed at every point in time where their direction and or speed changes. Vondrick et al. have discovered that the process of finding out where to place these keyframes for each velocity or direction change takes more time

than simply adjusting a dense set of pre-defined keyframes, even if many of them are redundant. Additionally, the authors discovered that only these pre-defined keyframes in a slideshow without the frames in between as context can be too little information, because annotators might mix up objects from one keyframe to the next. They therefore decided to display the whole timeline to annotators and enable arbitrary scrubbing through time. Figure 2.6 shows the UI that annotators are given, with the timeline at the bottom.

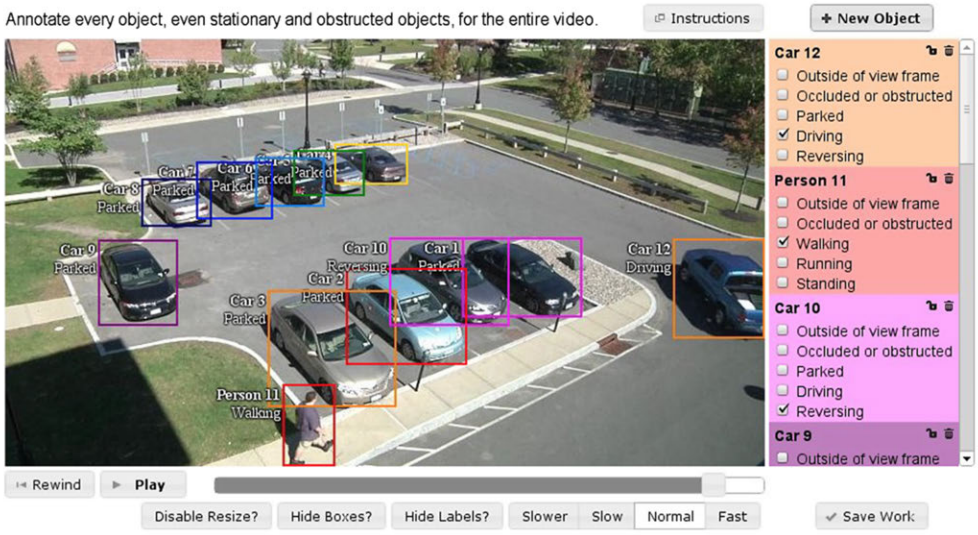

Figure 2.6: VATIC User Interface. Original image in [VPR13]

The main data propagation mechanism used in VATIC is *linear interpolation*. In their paper [VPR13], Vondrick et al. talk about using a dynamic programming algorithm to incorporate *constrained tracking*, where the first and last frame are fixed by user input. However, they report "poor results" and have apparently not added this functionality to their published source code.

2.2.2 ViTBAT

With the Video Tracking and Behavior Annotation Tool (ViTBAT), Biresaw et al. [BNFD16] focused mainly on the process of behavior annotation for both individuals and groups. They offer point, and rectangle annotations for the spatio-temporal aspect of annotation, and a timeline of user-definable behaviors.

When looking at the implementation details, and the platform choice of ViTBAT, we can see that it was developed on top of MATLAB and runs locally on the annotator's machine. Biresaw et al. chose this setup as it enabled them to make use of the large array of functionalities in the MATLAB computer vision toolbox. Additionally, choosing to run the software locally removes the latency associated with network connections and thus theoretically enables higher interactive speeds.

The UI is also implemented on top of MATLAB. An interesting aspect of ViTBAT's UI can be seen on the bottom left of Figure 2.7. Biresaw et al. show a timeline overview of behaviors with a list of all the different behaviors displayed on the left edge of the timeline and the different colors within each behavior representing the different subjects in the video. The lengths of these colored lines represent when and for how long the subjects display the different behaviors.

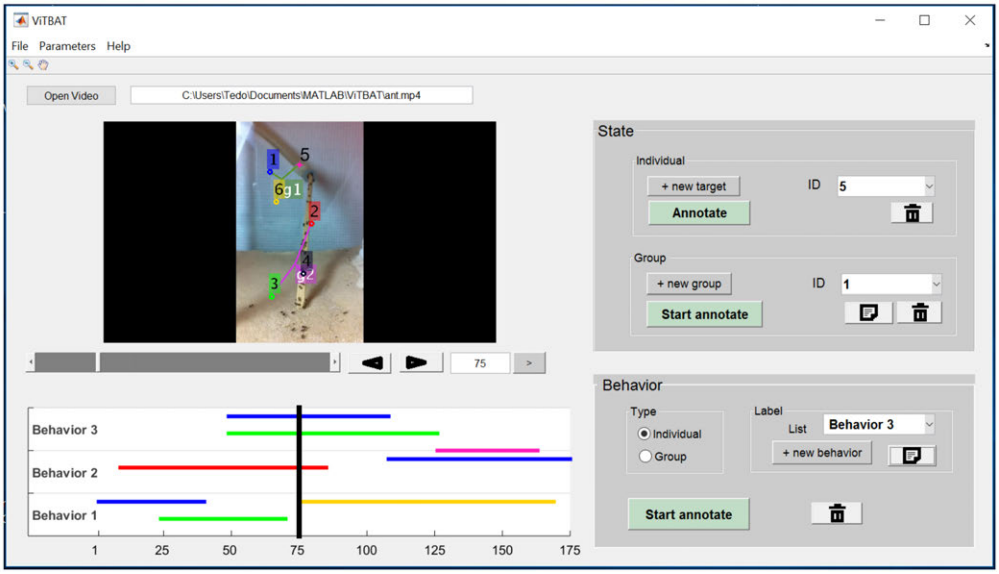

Figure 2.7: ViTBAT User Interface. Original image in [VR11]

Since ViTBAT is more focused on behavior annotation, Biresaw et al. do not use any sophisticated data-propagation mechanisms. Instead they use simple *linear interpolation* to update the positions and sizes of the rectangles in the scene.

2.2.3 BeaverDam

UC Berkeley student Anting Shen wrote his Master's Thesis on his Video Annotation Tool BeaverDam [She16], which focuses on minimizing both the administrator's and annotator's time when annotating large datasets. Shen analyzed the UI user studies performed by Vondrick et al. on VATIC [VR11] and incorporated and improved upon their findings.

One of Shen's major focus-points was the installation process and administration of annotations for reasearchers. Shen argues, that by eliminating "pain points" such as installation errors, a researcher's valuable (expensive) time can be spent on actual research instead of trying to install software. BeaverDam has been tested to install correctly on fresh installs of Ubuntu 14.04 and 16.04. It is built as a web application and geared towards crowd sourcing platforms such as Mechanical Turk.

Regarding BeaverDam's UI, Shen has done an excellent analysis in his thesis. His main findings are:

- **Keyframe placement and visibility**:
  In contrast to Vondrick et al.'s [VR11] discovery that regular keyframe placement leads to faster annotation time, Shen found that this is highly dependant on the video footage. He chose to let annotators place their own keyframes, but shows a keyframe timeline (see Figure 2.8) to give annotators an overview as well as a quick way to jump between keyframes, which in turn increases annotation speed.

- **Fast playback:**
  Caching the whole video in advance eliminates server timeouts on frame changes.

- **Click reduction:**
  Drawing new objects without the need to click "new object", and the object type is pre-selected as the most common class ("car") or the previous selection.

- **Frame exit/enter:**
  Being able to drag bounding boxes outside of the frame increases speed, as annotators do not have to align their mouse perfectly with the image border.

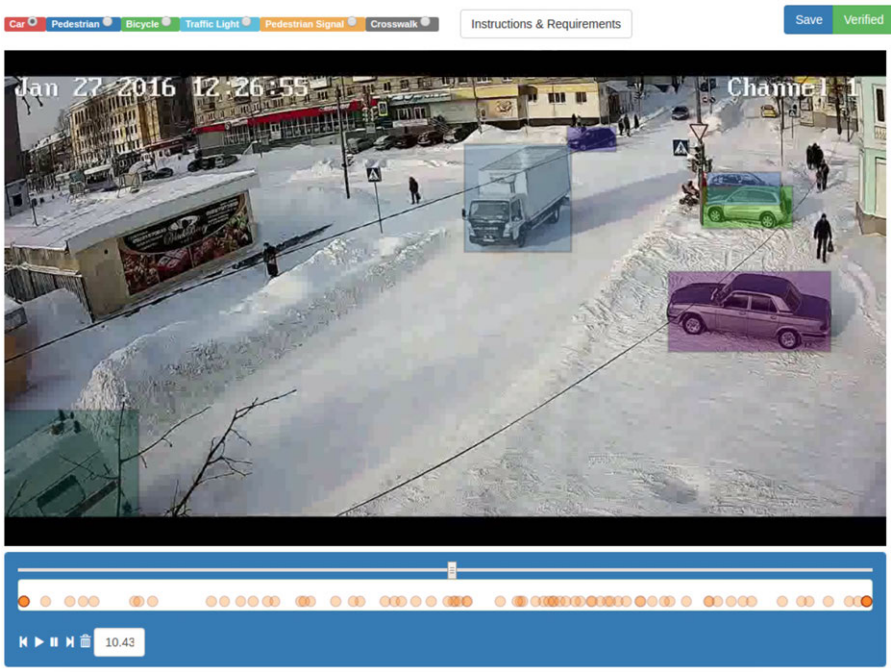

Figure 2.8: BeaverDam User Interface, with keyframe timeline at bottom. Original image in [She16]

In his thesis, Shen claims that BeaverDam has a computer vision tracking module to increase annotation speeds, however, the official implementation does not include[1] such a module, and the only data propagation mechanism is *linear interpolation*.

2.2.4 Scalabel

The Scalabel tool [YXC+18] was developed by Yu et al. at UC Berkeley to annotate their BDD100K dataset of more than 100,000 images. Their main focus was to create a versatile and scalable tool suitable for many different annotation tasks, that might be needed for a driving database, such as bounding box, semantic instance segmentation or lane detection. We will focus on its bounding box video annotation capabilites in particular.

Scalabel is a web application geared toward crowdsourcing applications in order to scale to a large number of annotators quickly. By developing a web application, the requirements to run the annotation workflow are reduced to a computer with an internet browser. There needs to be a back-end server which serves the web application to the annotator's internet browser and provides the dataset and annotation tasks, but the complexity for a new annotator to start on a task is reduced to clicking a link and reading the instructions.

Similar to BeaverDam, the Scalabel UI reduces the amount of clicks compared to VATIC, by remembering the type of object last annotated and removing the need to click/press a button to start annotating an object. Scalabel does not have a timeline overview to see where keyframes have been placed, however, in its video annotation UI a time slider is shown at the bottom to show the current moment in time. Scalabel allows zooming in and out up to a maximum factor of 5x, which can help with annotating smaller objects. See Figure 2.9 for a screenshot of the UI.

Scalabel is geared towards many different annotation techniques and currently only supports linear interpolation as its data propagation mechanism for video annotations using bounding boxes or segmentation masks.

2.2.5 CVAT

The Computer Vision Annotation Tool (CVAT) by Sekachev et al. [Sek18] is an open source tool, maintained by the OpenCV team. Its main focus is on including deep learning components to increase annotation efficiency. Some of the implemented components added to CVAT are *automatic object detection* and *semi-automatic segmentation*.

CVAT is built as a web application and primarily installed via docker. Even though it is a web application, its main focus is not on crowdsourcing annotations, but rather on reproducible builds through containerization and easy access through browser technologies.

The UI of CVAT makes it possible to do both bounding box and segmentation annotation in the same video. For each new annotation the "create shape" button, seen in the

[1] Github issue, where Shen says that no tracking capabilities were implemented: https://github.com/antingshen/BeaverDam/issues/104#issuecomment-310500517

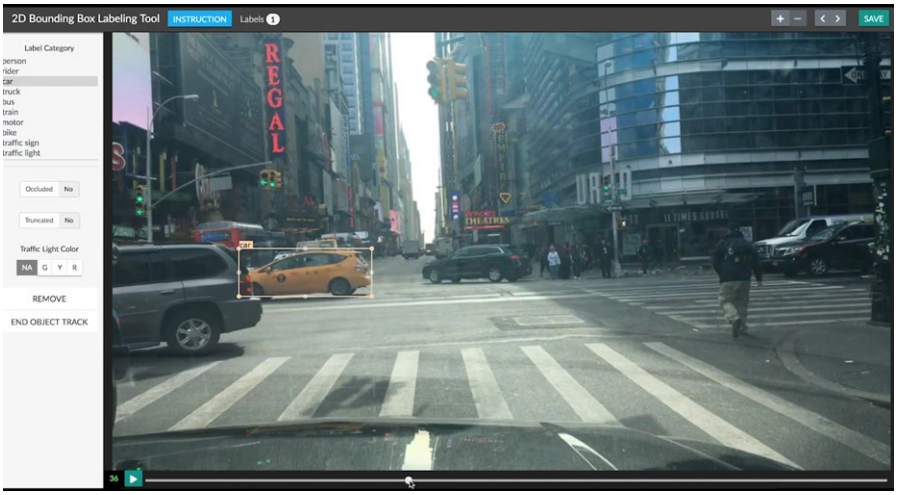

Figure 2.9: Screenshot of Scalabel User Interface.

bottom right of Figure 2.10 needs to be clicked and the type of shape to be used needs to be selected. This does not follow the best practice guideline on minimizing clicks developed by Shen [She16], which recommends using a click-and-drag mechanism inside the current frame for the automatic creation of a new bounding box.

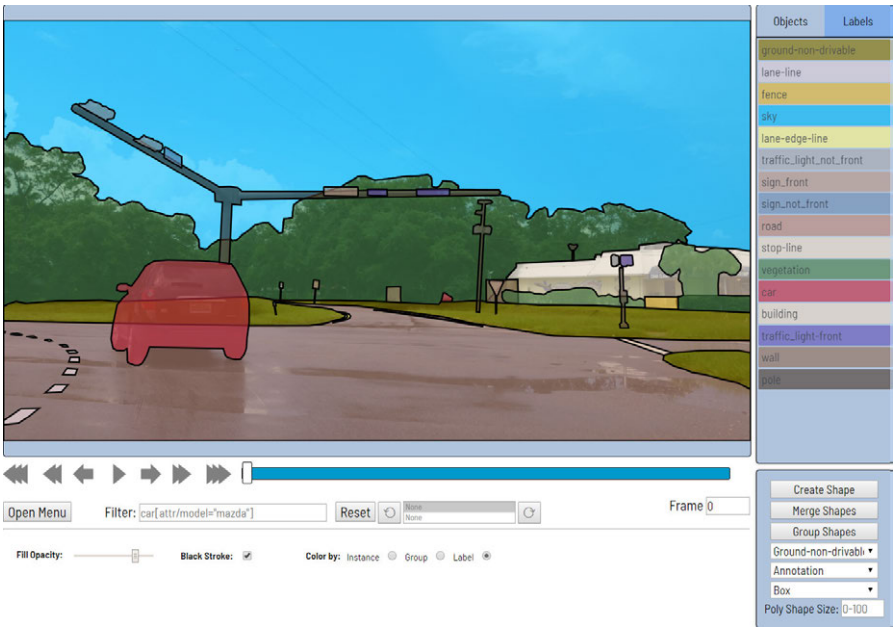

Figure 2.10: Screenshot of CVAT User Interface

With regards to data propagation, onepanel[2] has added a preliminary implementation

[2]https://www.onepanel.io/

for tracking bounding boxes over time. The feature has not yet been added to the official implementation of CVAT, which only includes *linear interpolation* as its data propagation method.

2.2.6 Comparison

This section tries to summarize our findings concerning the different annotation tools mentioned above. We compare the tools regarding platform, user interface and data propagation and provide an overview of our findings in Table 2.1.

Platform

With the exception of ViTBAT, the aforementioned tools work as web applications with the browser acting as the user facing front end and a web server acting as the back end, keeping track of all of the data. The focus on web technologies is primarily rooted in the fact that annotation tasks can then be accessed through a simple URL and can therefore be included into crowdsourcing services such as Mechanical Turk. The authors of ViTBAT, on the other hand, chose to offer a tool that runs locally on the annotator's machine, removing the latency associated with network connections and thus theoretically enabling higher interactive speeds.

User Interface

Regarding the UIs, we found that most tools do not have a temporal overview of the current annotation state with the help of some sort of timeline view. BeaverDam and ViTBAT were the exception to this rule. BeaverDam displays a line of keyframes to show where a user has adjusted the bounding box of an object of interest (see Figure 2.8). ViTBAT, with its focus on behavior annotation, displays a timeline overview of different behaviors in the video to quickly see when subjects perform different behaviors (see Figure 2.7).

Data Propagation

Regarding data propagation, we found that with the exception of CVAT [Sek18], the existing video annotation tools all offer linear interpolation between keyframes. This can be very helpful when dealing with footage from a stationary camera. But when the camera itself is moving, the amount of keyframes needed to follow objects in screen space greatly increases due to abrupt movements in the camera path (e.g. road bumps, sharp turns).

Tool	Platform	Propagation	Temporal overview
VATIC [VPR13]	Web	Interpolation	✗
ViTBAT [BNFD16]	Local	Interpolation	Behavior
Scalabel [YXC+18]	Web	Interpolation	✗
BeaverDam [She16]	Web	Interpolation	Keyframes
CVAT [Sek18]	Web	Tracking	✗

Table 2.1: Comparison of analyzed Video Annotation Tools

CHAPTER 3

Datasets

In this chapter, we review the existing automotive and tracking datasets in the scientific community which we found relevant for our project. We give insight into which requirements we deemed necessary for a dataset to be adequate for the CVL Project and give an overview of the dataset which we were able to create in the process of the project.

Firstly, in Section 3.1, we present an overview of the relevant existing datasets. Depending on the availability of the information provided by their authors, we will summarize their scope and acquisition processes, specifically focusing on the environments in which the images were taken, the amount of time it took to label all the images, what type of classes the images consist of and the total amount of images included in each of the datasets.

Section 3.2 describes our own dataset (CVL Dataset), which we acquired over the course of the project and annotated using our CVLA tool proposed in Chapter 4. For the CVL dataset, we provide some further statistical analyses regarding track lengths, bounding box sizes and class distributions.

Finally, we present a comparison in section 3.3 with the following requirements of our specific application in mind: *non-urban roads (e.g. highway or country roads)*, *nighttime*, *at least 20FPS temporal density*, *realistic lighting in a real-world environment.*

3.1 Ground Truth Datasets

In this section, we provide short summaries of some state-of-the-art datasets and details on how researchers are using different techniques like (i) multi-sensor data, (ii) Computer Vision (CV) algorithms or (iii) Computer-Generated Imagery (CGI) to aid in the creation process of ground truth datasets. The general problem with creating large ground truth datasets is described by Xie et al. in [XKSG16], in which they mention the term “curse of dataset annotation”. It explains the inverse relation between number of annotated images and annotation time needed per image. In other words, the more time consuming

it is to annotate an image (e.g. because we want to assign an object class to every pixel, instead of just a scene descriptor for the whole image), the fewer total images will be in the final dataset as the effort required to provide hundreds of thousands of images is too extensive. Figure 3.1 shows the relationship between annotation time per image (i.e. level of detail) and dataset size.

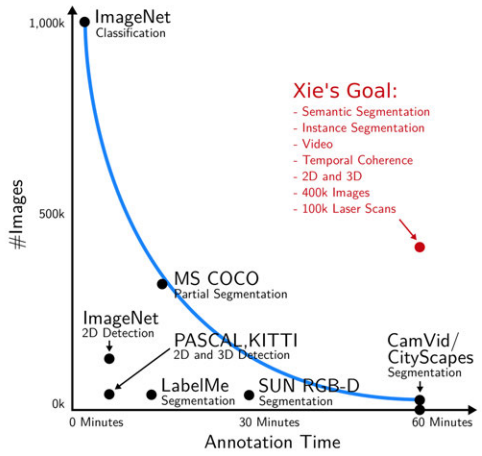

Figure 3.1: Curse of Dataset Annotation, correlation plot between annotation time and number of images. Original image in [XKSG16]

3.1.1 Multi-Sensor Annotation

This section shows examples of datasets that combined data from multiple sensors to aid in the annotation process.

Argoverse

Ming-Fang Chang et al. [CLS+19] present a dataset for autonomous vehicle perception tasks such as 3D tracking. The dataset includes 360 degree images from 7 cameras with overlapping fields of view, stereo imagery facing forward and 3D point clouds from a Light Detection and Ranging (LIDAR) sensor. The data contains lane annotations of about 290km. Argoverse's cars were deployed in two American cities (Miami, Pittsburgh) and captured data at different times of day, including nighttime, and in a wide range of weather conditions. In total, there are 10,572 human-annotated tracked objects in the dataset, with a framerate of 30fps and annotated with 3D bounding boxes. To enable faster annotations on all of the image data provided by the camera sensors, the annotation was done via 3D cuboids on the point cloud data provided by the LIDAR sensor, and then reprojected onto the image data. An overview of this multi sensor data can be seen in Figure 3.2.

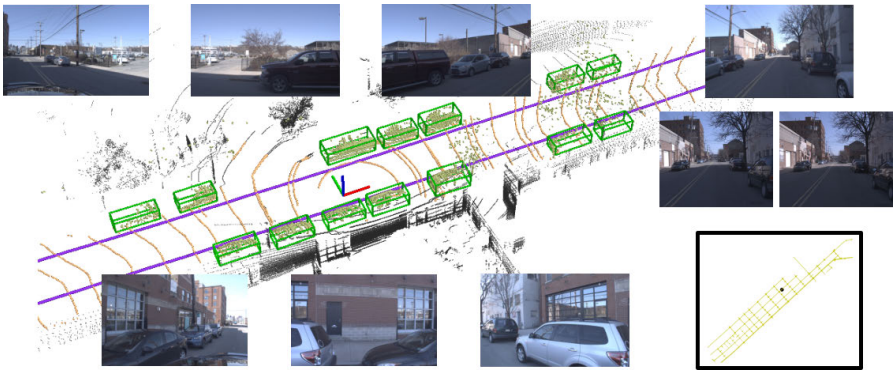

Figure 3.2: Overview of LIDAR and Camera data with 3D bounding boxes. Original image in [CLS+19]

KITTI Dataset

Andreas Geiger et al. [GLU12] used their research vehicle AnnieWay, equipped with color and monochrome stereo cameras, a LIDAR and a Radio Detection and Ranging (RADAR) sensor as well as a GPS/IMU navigation system to record scenes on the roads of Germany. The raw data captured by this set of advanced sensors allowed them to calculate multiple ground truth datasets for the tasks of stereo matching, optical flow estimation, visual odometry, 2D/3D object detection and object tracking [GLU12]. Additionally, Geiger et al. have also published the raw dataset [GLSU13]. All of the data provided in the KITTI dataset was captured in Germany during daytime with good visibility. To ensure a bit of variability within the scenes, the capture sessions took place in urban, suburban and highway areas, the last of which is the most relevant for our purposes. Out of those scenes, the 2D/3D object detection datasets are the only ones which meet the needs of the CVL Project. The datasets consist of 7481 training images and 7518 test images including 3 types of classes: car, cyclist, pedestrian. The annotation of the raw data was done manually by hired annotators, which were given a custom annotation tool showing the 3D pointcloud of the LIDAR system and the color images. It is not stated how long it took for each image to be annotated. Figure 3.3 shows an example 3D object detection dataset.

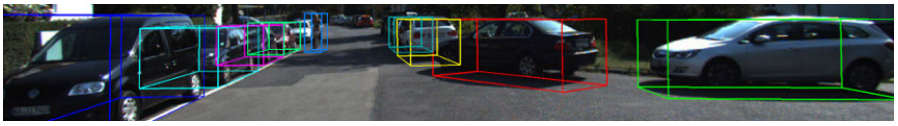

Figure 3.3: Sample of 3D bounding boxes in KITTI dataset. Original image in [GLU12].

3.1.2 Computer Vision Algorithms

In this section we show datasets which used automatic computer vision algorithms as a starting point to aid in the annotation process.

BDD100k

BDD100k by Yu et al. [YXC+18] is a dataset of more than 100,000 videos with fully segmented road scene images from roads in the USA. It features image data from all over the Bay Area as well as New York City. The images have been taken during all times of the day, including nighttime, and in many different weather conditions. Yu et al. have developed an annotation tool called Scalabel (see Section 2.2.4) to create the ground truth labels in a distributed manner. Even though the dataset contains a large amount of video data, annotation data is only provided for a single frame per video. This is done to maximize diversity in the given scenes, however it is unfortunately not suited for evaluating tracking algorithms (In 2020 the BDD100k MOT extension was added to the dataset, including tracking data with a temporal density of about 5 frames per second). To aid the annotation process, a baseline semantic segmentation model was used as starting data, which was then refined by human annotators.

Figure 3.4: Example of fully segmented ground truth in BDD100k dataset. Original image in [YXC+18].

CamVid

Brostow et al. [BFC09] used a consumer camera mounted on the dashboard in front of the passenger seat to record scenes on the roads of the United Kingdom. The videos were recorded during daytime, with high visibility and in a mainly urban to suburban area. The dataset consists of 701 fully segmented images including 32 classes. The classes are categorized as moving objects, road, ceiling and fixed objects. This means that almost every pixel on an image has a corresponding class. As mentioned in Section 2.1, the process of fully annotating an image with meaningful classes is called semantic segmentation. An example of which can be seen in Figure 3.5, which shows a sample frame with its annotation data next to it, where only a small part of the ground truth is

left unlabeled (black). The total amount of man-hours invested into annotating these images is reported as 230 hours, which results in an average annotation time per image of about 20 minutes and was done by volunteer workers recruited via facebook, which received a small compensation. To aid the annotation process, automated segmentation algorithms were used as a starting point which would later be refined by the annotators. This fine-grained approach to annotating the images is well suited for our needs, however the availability of only daytime (sub-)urban scenes makes this dataset unfit for our purposes.

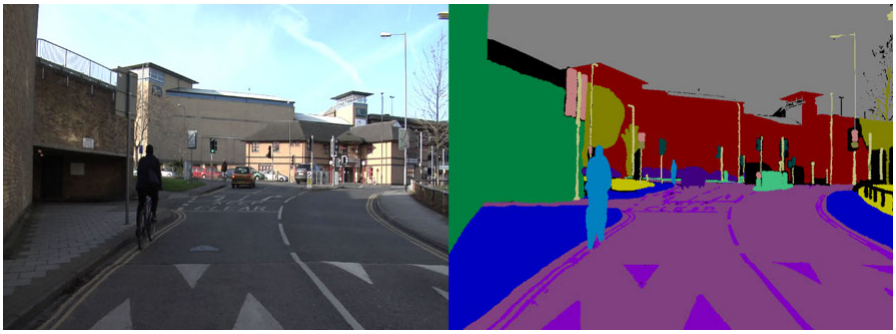

Figure 3.5: Example of fully segmented ground truth in CamVid dataset. Original image in [BFC09].

D^2-City

The D^2-City dataset by Che et al. [CLL$^+$19] contains more than 11,000 traffic videos from various cities around China. In one thousand of the collected videos, Che et al. provide frame-by-frame annotation data for tracking purposes, totalling in more than 700,000 annotated frames. The rest of the videos have been annotated more sparsely: annotation data has been included every couple of frames. Che et al. chose to annotate their dataset with a customized version of the CVAT ([Sek18], see section 2.2.5). All annotations were created manually or by *linear interpolation* within very short time frames with manual adjustments to guarantee the quality of the data.

Figure 3.6: Example scenes from D2-City dataset. Original image in [CLL$^+$19].

3.1.3 Computer Generated Imagery

This section shows synthetically generated datasets, which aim to mimic real world imagery in order to minimize the need for real-world footage for training further computer vision algorithms.

SYNTHIA

As we mentioned before, there are also researchers who create synthetic ground truth data sets in order to enhance the performance of machine learning algorithms. Ros et al. [RSM+16] produced a synthetic dataset using their own virtual world and software created with the Unity game engine. In total, they provide four video sequences comprising 50,000 images each, showing the virtual world in different seasons of the year totalling in more than 200,000 images. See Figure 3.7 for an example. All of the images were taken during virtual daytime and do not contain any night scenes. Additionally, the quality of the images also lacks in realism, as shown in a perceptual experiment by Richter et al. [RHK17] (see next section).

Figure 3.7: Example of four different seasons in SYNTHIA virtual world. Original image in [RSM+16]

VIPER

Stephan Richter et al. [RHK17] also provide a synthetic data set consisting of 254,064 fully annotated images collected while driving, riding, and walking a total of 184 virtual kilometers through the diverse and realistic world of GTA V. The dataset contains scenes with five different environmental conditions (day, sunset, rain, night, snow) and annotation data for 30 different classes including many types of vehicles, as well as road, sky, vegetation or pedestrian. They chose this game specifically because of its highly realistic look and feel. To quantify the realism of the game Richter et al. performed a perceptual experiment of on Amazon Mechanical Turk comparing GTA V images to other synthetic and real datasets. The experiment was an A/B test, where users had to decide which of two images looked more realistic. When users had 8 seconds to analyse which of the presented images of the A/B test looked more realistic, 94% selected images from VIsual PERception (VIPER) instead of those from SYNTHIA. Richter et al. also compared the images with the real CityScapes dataset and surprisingly even compared to real data 11% felt that GTA V images looked more realistic, emphasizing the visual quality found in the VIPER dataset.

Richter et al. had various use cases in mind for their dataset: 2D and 3D bounding box annotations, semantic segmentation, semantic instance segmentation and visual odometry, the first of which is of importance for the CVL Project. Figure 3.8 shows a nighttime scene with its corresponding semantic segmentation applied on the left side. The data was created by using custom shaders and modding software to alter the game's output buffers on the GPU. In a previous paper, Richter et al. [RVRK16] go into further detail

regarding the creation of the semantic segmentation ground truth and mention an average annotation time of about 7 seconds per image. This paper also shows that learning on a mixed dataset of real and synthetic data can improve performance significantly (up to 2.6 percentage points are reported).

Figure 3.8: On the left side of this image we see the semantically segmentation mask overlayed and gradually faded out to one of the nighttime scenes from the VIPER dataset on the right.

3.1.4 Manual Annotation

In this section, we discuss ground truth datasets which were generated entirely by humans, without the help of any computer algorithms.

CityScapes

Cordts et al. [COR+16] also worked on the task of semantic segmentation. They used cars with a stereo camera setup and a GPS sensor to record scenes in 50 different cities in Germany. The videos were recorded during daytime and with high visibility. In total they provide 5000 fully segmented images and 20000 coarsely segmented images containing up to 25 different classes. See Figure 3.9 for an example. The average annotation time per image is reported to be roughly 90 minutes [COR+16]. The annotation was done on stereo imagery by professional annotators in-house to ensure its quality. Scene elements were annotated back to front, such that each boundary had to be marked only once. Additionally, this way of annotating also encodes a simple depth ordering into the classes.

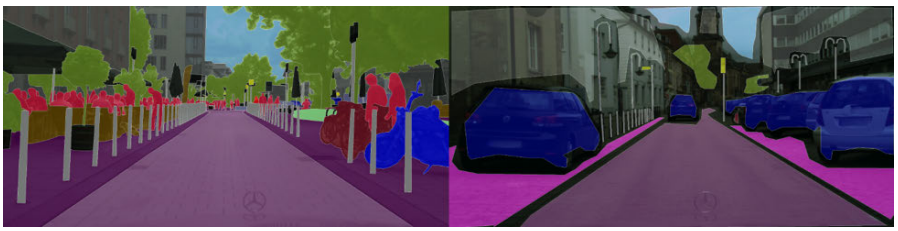

Figure 3.9: Example of the fine (l) and coarse (r) segmentation within the Cityscapes dataset. Original image in [COR+16]

Mapillary

Neuhold et al. [NORBK17] from Mapillary Research provide a dataset containing 25 000 high resolution images with 66 object classes using fine-grained polygon annotations. Compared to other datasets already shown in this document, the Mapillary dataset offers a more diverse look at the world with images taken by individuals all over the globe using a multitude of different cameras (smartphones, tablets, action cameras etc). In addition to having photos from many different cameras, they are also taken across many countries, during different times of day and in different weather conditions (see Figure 3.10). On top of that, the images included in the dataset do not all show scenes from inside or on top of a car, but also street level scenes in general (i.e. they include images taken by pedestrians as well). Annotation was done by 69 professional image annotators, with an average annotation time of about 94 minutes per image.

Figure 3.10: Example of diverse weather and lighting conditions with corresponding annotation data in Mapillary dataset. Original image in [NORBK17]

SYSU and CUHK Night-time Datasets

Chen et al. [CHX+17] provide two datasets consisting only of nighttime scenes. The objects of interest within the images in the datasets are marked by a 2D bounding box, containing the full extent of the vehicles. While the SYSU dataset comprises 402 images with a single class only (car), the CUHK dataset consists of 836 images with five classes (car, taxi, bus, minibus, truck). The images where taken in China (SYSU) and Hong Kong (CUHK). The authors do not state who annotated the data or how long the annotation took to complete. In Figure 3.11 it can be seen that the quality of the datasets does not meet our requirements as the bounding boxes are too large and there is an undetected car, whose driver would be blinded by the headlights.

VOT Challenge

In 2013, a dataset for visual object tracking purposes called VOT2013 [KPL+13] with an accompanying visual object tracking benchmark challenge was released. The challenge has continued every year since then and new ground truth data, such as RGB-D images, has continuously been added to the dataset. Annotation was done manually by the

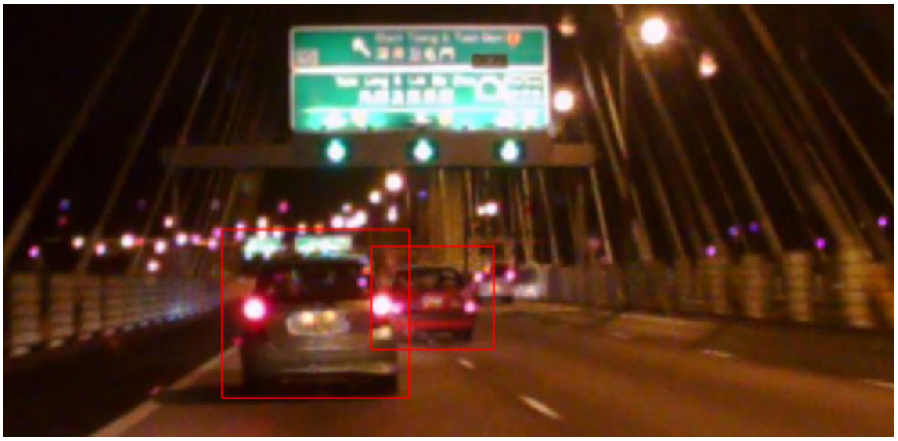

Figure 3.11: Example image from the CUHK dataset showing an unmarked car as well as bounding boxes which are too big

expert annotators in the VOT committee. Unfortunately, even though the dataset focuses on providing a wide variety of different scenes regarding visible objects and lighting conditions, it does not contain many automotive scenes (an example can be seen in Figure 3.12), and completely lacks nighttime automotive scenes. However the authors' contribution is not limited to the dataset. They have also introduced a novel evaluation methodology for single-target tracker performance [KML+16] which we will take a closer look at in Chapter 5.

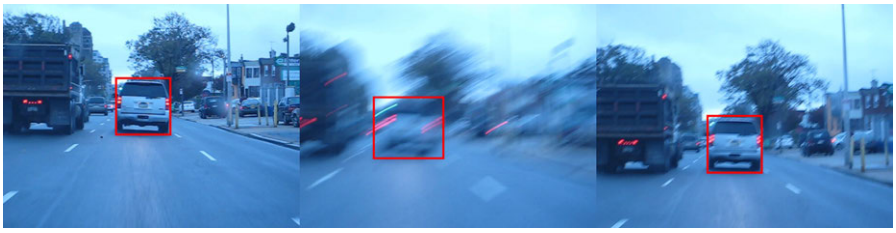

Figure 3.12: Example scene from the VOT Challenge Dataset

3.2 CVL Dataset

The CVL Dataset was born out of the necessity to have an automotive dataset with high temporal density, non-urban scenes and nighttime footage. The raw image data was gathered at 30 frames per second by our project partner ZKW over the course of two years (2018-2020) and annotated at TU Wien as part of this diploma thesis. To ensure variability in the data, three different camera sensors with different dynamic ranges, and resolutions were used. Additionally, the cameras were positioned at a variety of different places both inside and outside of the vehicle. In total, there are 22420 annotated images from 49 videos in the dataset, 44 videos or 18600 images of which were taken at night,

and 5 videos or 3820 images taken during daytime. These annotated images contain 320 tracked objects 81 of which are at daytime and 239 at night. Figure 3.13 shows example images of these two variations and gives an example of how dark and low in visible information some of our nighttime footage is. Subsection 3.2.1, shows further examples and illustrates some of the challenging scenarios included in the dataset. Subsection 3.2.2 gives further statistics and analyses regarding our dataset.

Figure 3.13: Day (left) and nighttime (right) example images from our CVL Dataset. Bigger red bounding boxes denoting oncoming vehicles, small bounding box in nighttime image denotes a bicycle and gives an example of how little visible information was recorded.

3.2.1 Example Footage

In this section, we want to highlight the challenging nature of the CVL Dataset. Due to the predominantly dark environments in which our material was recorded, most of the time the only light sources in the scene are the headlights of the cars. This fact results in reflections, lens flares and exposure complications such as light blooming or motion blur.

In Figures 3.14 and 3.15 we can see examples of how dark our footage can get. Figure 3.14 shows the footage of a single car driving with turned on high beams on a non-urban road. Even with the brightest light available in the test car it is hard to see the pedestrian next to the road, when they are more than 70m away from the vehicle. This might not be a scenario one encounters every day, but it might still happen on a few occasions (e.g. it could be a person with car trouble standing outside their vehicle).

Figure 3.14: Pedestrian on side of the road at distances of about 70m, 35m and 10m (from left to right).

Figure 3.15 shows a simple scene of oncoming traffic on a non-urban road. We can see that the upper two thirds of the scene are completely dark, and also the road fades into black just behind the oncoming car.

Figure 3.15: Oncoming traffic on non-urban road, lit only by headlights.

Figure 3.16 shows the large amount of motion blur occurring in some of the recorded scenes. This motion blur effect results out of the trade-off between trying to capture as much information as possible by choosing a long exposure time, against trying to get fast update times and sharp images with clear object boundaries. This trade-off has to be made at night with limited amounts of light in the scene, whereas during daytime the sun offers an abundance of light, enabling short exposure times and sharp images.

Figure 3.16: Motion blur on traffic signs.

Another problem we encounter in our dataset is lens flare and over-exposed headlights. These are caused by long exposure times and wide open apertures and can be seen in Figure 3.17 and 3.18. The images show another effect of the trade-off between wanting

to capture as much light as possible and the desire of a sharp image. Figure 3.18 is a pronounced example of this effect, where the combination of over-exposure and lens flares washes out almost the entire field of view. In Figure 3.19 we see an example of lens flares on the brake lights of the preceding car on the right, and motion blur on the traffic sign as well as the oncoming car.

Figure 3.17: Over-exposure (white reflection on road) and lens flares (star-shaped streaks around headlights, red dots on grass patch).

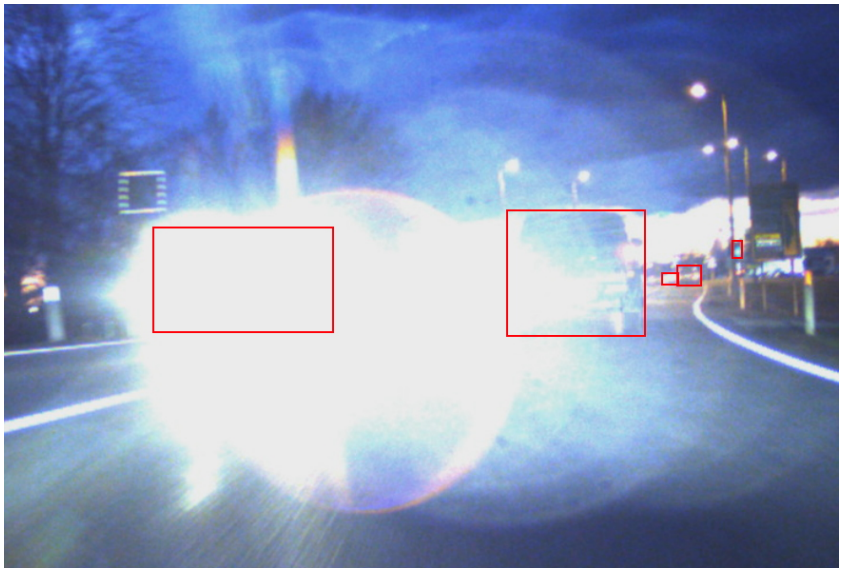

Figure 3.18: Over-exposure and lens flare of oncoming headlight affecting almost the entire field of view.

Figure 3.19: Examples of motion blur (left car, traffic sign) and lens flare (right car).

3.2.2 Statistics

This section focuses on the statistical analysis of the CVL Dataset. In Figure 3.20 we can see the partition of our data into day and night scenes, about 20% of our image data has been taken during the day, and the rest was taken at night. The color codes of dark blue for nighttime, and light blue for daytime footage will be continued throughout all of the figures in this section.

The list of object classes annotated in the CVL dataset is as follows:

- bicycle
- car
- motorcycle
- pedestrian
- traffic light
- traffic sign
- truck

Figure 3.21 shows which object class has the largest presence in our dataset (cars at 68.1%) and which class has the lowest presence (bicycle at 0.5%). This information does not come as a surprise, as most of the data was taken on country roads at nighttime in

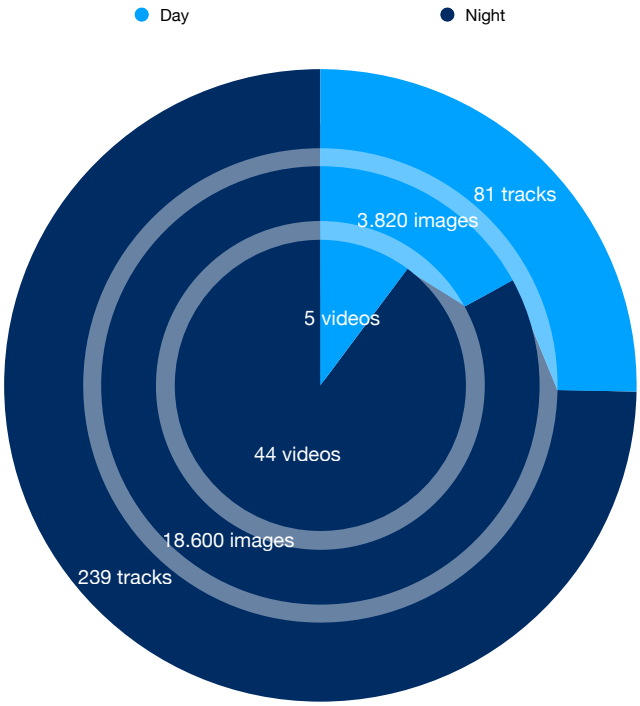

Figure 3.20: Day/Night partition of the CVL Dataset

Austria, where not a lot of bicycles are to be expected. The relatively large presence of pedestrian data can be explained by the fact, that these come from a test setup, where we recorded a number of scenes with a pedestrian at the side of the road under different lighting conditions.

Another interesting observation can be made about the distribution of bounding box area sizes available in the CVL dataset (Figure 3.22). About a fifth (20.1%) of the bounding box areas are under the size of 400 pixels (denoted as 20^2 in the figure caption, as it is easier to imagine a box of 20x20 pixels to represent all bounding boxes of up to 400 pixels) and more than half (50.3%) of the annotated areas are under the size of 1600 pixels (or 40^2). This can be attributed to the fact that Austria's country roads usually see rather low traffic at night and visibility typically reaches quite far, hence a lot of small objects can be observed in the distance. Additionally, most of the larger bounding boxes in our dataset belong to oncoming traffic which quickly passes our field of view and is therefore only visible for a couple of frames.

The fact that most of the bounding boxes in our dataset belong to oncoming traffic can also be observed in Figure 3.23, where we can see that more than half (57.5% or 184) of our total of 320 object tracks are shorter than 50 frames or 1.66s, and 80% of the object tracks are shorter than 150 frames or 5s. The number of times we observed and annotated preceding traffic for longer than 501 frames or 16.66s is seven.

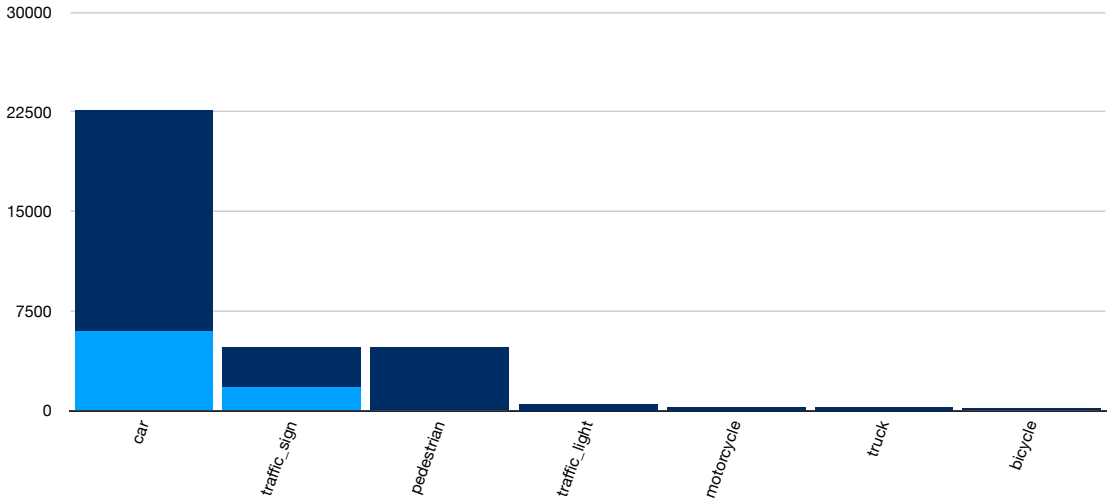

Figure 3.21: Histogram of number of bounding boxes per class available in the CVL Dataset. Dark blue refers to nighttime footage, light blue is daytime footage

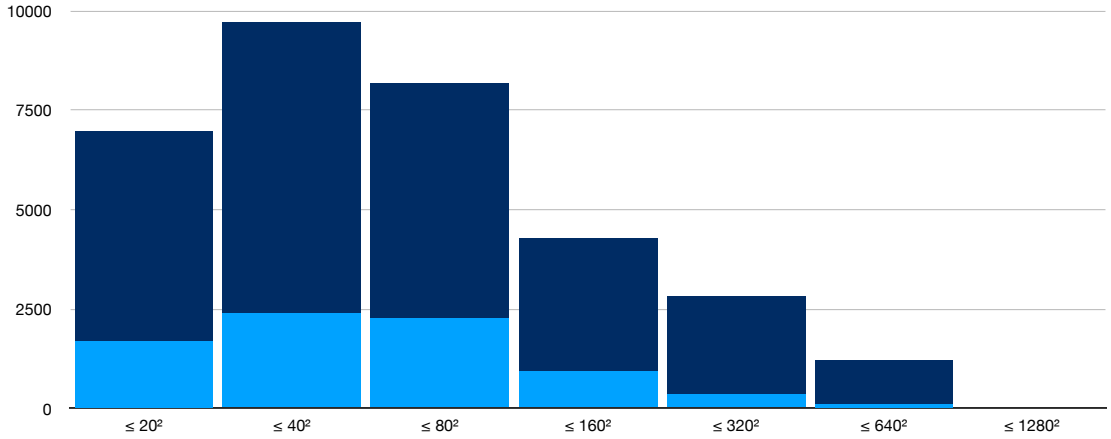

Figure 3.22: Histogram of different bounding box sizes observed in the CVL Dataset. Dark blue refers to nighttime footage, light blue is daytime footage

3.3 Comparison

Table 3.1 gives an overview of various datasets (both road scenes and general scenes) we have reviewed. Several datasets (e.g. VOT2017 [KML+16], VIPER [RHK17], D^2-City [CLL+19], BDD100K [YXC+18]) have the temporal density needed for our application. While temporally dense nighttime scenes are included in some of them ([CLS+19], [CLL+19]), we noticed a shortage of footage from non-urban roads. In the case of synthetically generated videos, such as in ([RHK17]), we observed a lack of natural lighting variability (high dynamic contrast, glaring, halos or reflections).

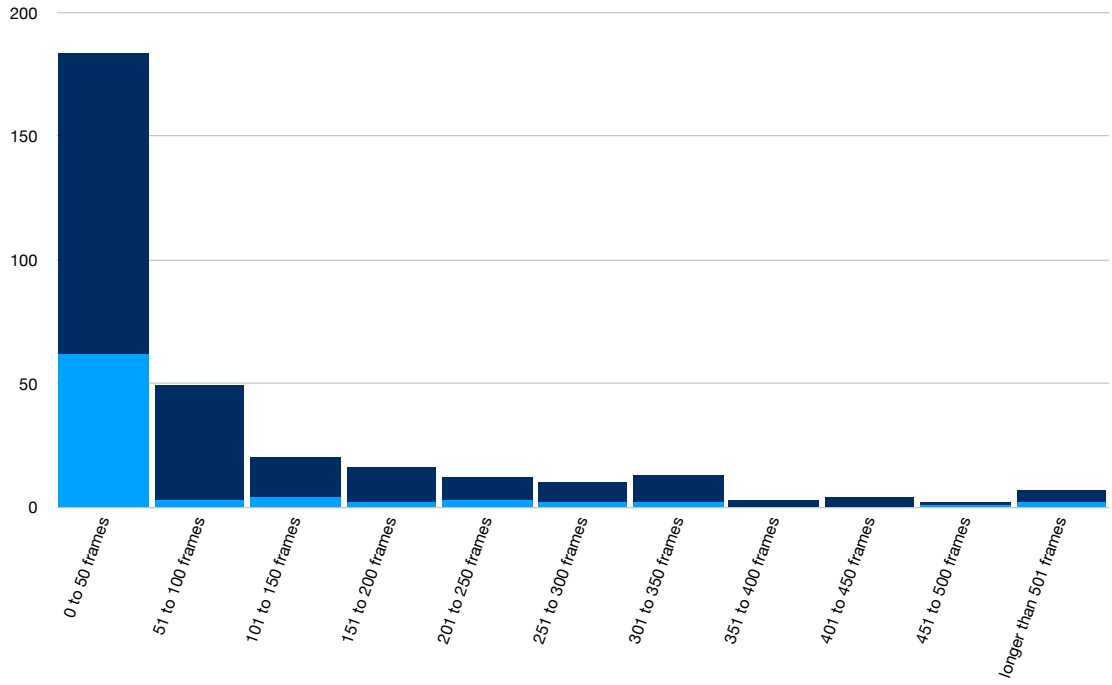

Figure 3.23: Histogram of different track lengths in the CVL Dataset. Dark blue refers to nighttime footage, light blue is daytime footage

We can tell from this investigation that currently there is no publicly available ground truth dataset which fully meets our requirements. This served as motivation for us to annotate the CVL dataset and to develop an annotation tools that support efficient ground truth generation for self-recorded nighttime traffic scenes.

Dataset	**Non-Urban**	**Night**	**$\geq$ 20 FPS**	**Real**
Argoverse [CLS+19]		✓	✓	✓
BDD100k [YXC+18]	✓	✓		✓
CamVid [BFC09]				✓
CityScapes [COR+16]				✓
D^2-City [CLL+19]		✓	✓	✓
KITTI [GLSU13]	✓		✓	✓
Mapillary [NORBK17]	✓	✓		✓
SYNTHIA [RSM+16]	✓	✓		
SYSU & CUHK [CHX+17]		✓		✓
VIPER [RVRK16]	✓	✓	✓	
VOT Challenge [KML+16]			✓	✓
CVL Dataset (*Ours*)	✓	✓	✓	✓

Table 3.1: Overview of datasets regarding selected requirements.

CHAPTER 4

CarVisionLight Annotation Tool

This Chapter will describe the tracker-assisted annotation tool CarVisionLight Annotator (CVLA), which was developed in the context of this diploma thesis. While existing annotation tools reviewed in Section 2.2, contain only linear interpolation as their propagation method, the incorporation of state-of-the-art object tracking algorithms between frames of human input data (keyframes) is a major component of our work. In Chapter 5 we perform an extensive test of different state-of-the-art algorithms on synthetic night scenes from the VIPER [RHK17] dataset to determine which tracking algorithms to include in our program. The following subsections explore implementation choices regarding platform, UI framework and deployment system, as well as user interface choices and our tracker integration.

4.1 Implementation

One of our goals for implementing CVLA was to be able to easily include state-of-the-art tracking algorithms as well as providing an easy solution for deploying and installing the application. We first present a fully web-based application, running entirely in the browser, and then focus on a locally installed python application. The following paragraphs explain the motivation behind our final decision for a local installation based on python[1], PyQt[2] and fbs[3].

4.1.1 Browser Implementation

Our first approach for developing CVLA was fully based on browser technologies, with JavaScript running directly in the annotator's browser. One of the biggest advantages of

[1] https://www.python.org
[2] https://riverbankcomputing.com/software/pyqt/intro
[3] https://build-system.fman.io

such web applications is the fact that the user only needs internet access and a browser, with which they open a Uniform Resource Locator (URL) to be able to start working with the tool. However, there are some limitations as to what kind of technologies can be developed with a JavaScript app alone. In comparison with a local installation, which has a broad range of possibilities regarding the types of technologies that could be included, web applications are limited to browser standards and their official Application Programming Interfaces (API), or alternatively they can rely on the computing power of a centralised back-end server.

Our first build of CVLA, see Figure 4.1, is based entirely on a JavaScript implementation and a simple optical flow tracker. It includs a first implementation of a timeline view, which enables a quick overview of where user input is present (see Section 4.2 for more information). A limiting factor of this implementation is the restricted access to the Graphics Processing Unit (GPU) of the host machine. As of writing this thesis, JavaScript applications in current browsers can only interact with the GPU using the Web Graphics Library (WebGL) interface whose primary use case was designed for 3D graphics applications, such as video games. This limitation is particularly bad for state-of-the-art tracking algorithms, as most of them (e.g. Siamese Region Proposal Network (SiamRPN) [LYW+18], Accurate Tracking by Overlap Maximization (ATOM) [DBKF19]) are based on deep neural networks, which make use of the GPU's capabilities for running parallelized code. The community has built solutions around these limitations like tensorflow.js[4], but it is necessary to convert each model to be compatible, and this does not work for some state-of-the-art modules like Jiang et al.'s Precise ROI Pooling [JLM+18] used in ATOM. Apart from running JavaScript code, modern browsers can also run WebAssembly code. For this, programs and libraries, which are written in C or similar lower level programming languages, and which have to be compiled to run on the Central Processing Unit (CPU), can be compiled to WebAssembly to enable faster runtimes in the browser. One such library is OpenCV[5]. It has a WebAssembly version called OpenCV.js, however it also comes with a few limitations, namely a complete lack of the tracking module, which includes trackers such as MedianFlow, Kernelized Correlation Filters (KCF) ([KMM10] and [HCMB14]).

With these limitations in mind, we had the option of implementing a back-end server with tracking capabilities or a locally installable program. Implementing the tracking capabilities on a back-end server introduces a question of data transmission speeds and lag. Since the video data needs to be processed on the back-end server, it somehow needs to find its way on to that machine. This could be done either through just in time delivery, where each frame gets sent to the server as it gets processed, introducing lag, or through prior upload, introducing long wait times before being able to get started. Since both options are not ideal for our use case, we finally decided to implement CVLA as a locally installable program built with python.

[4] https://www.tensorflow.org/js
[5] https://opencv.org

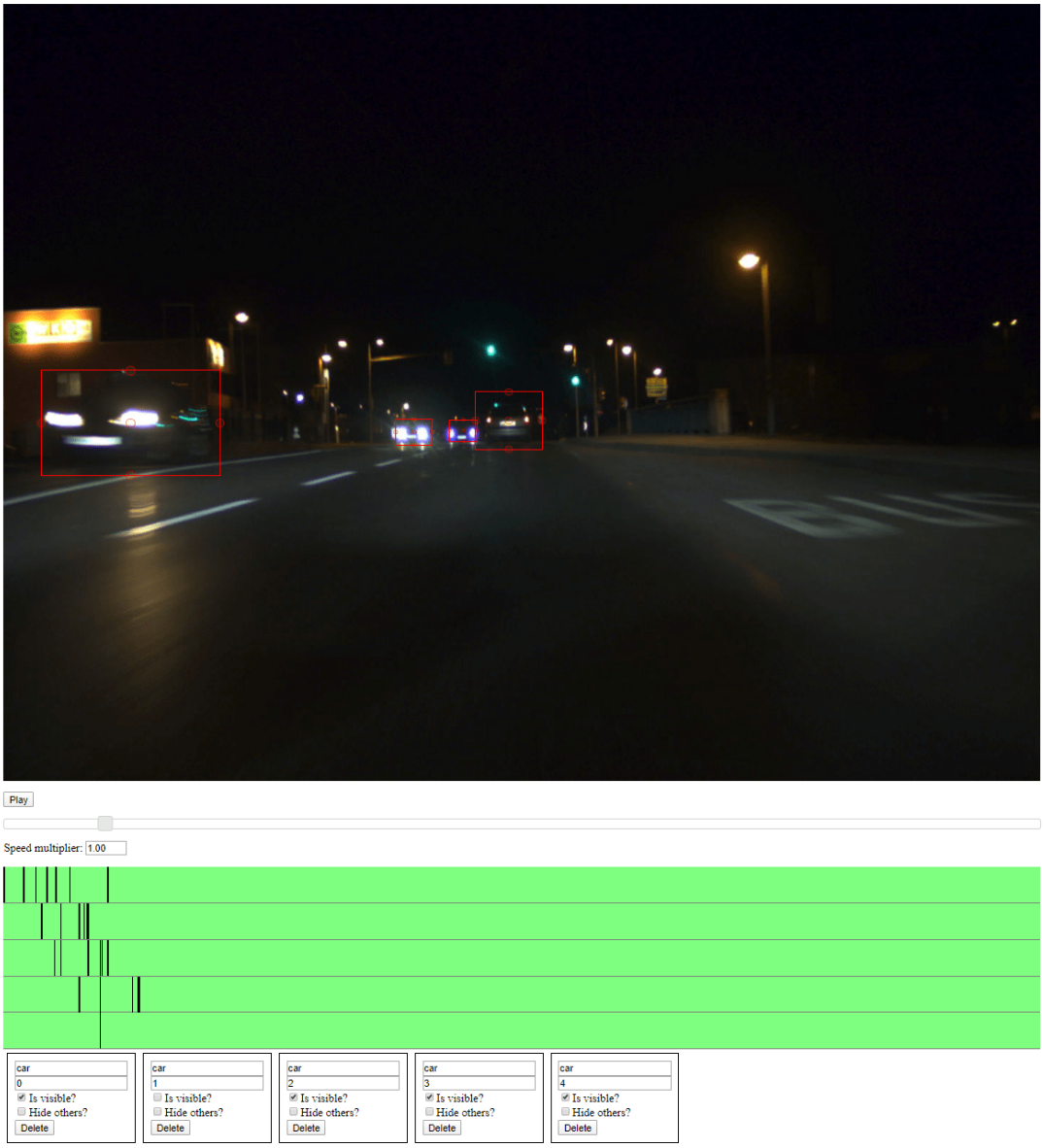

Figure 4.1: Screenshot of first browser implementation with rudimentary keyframe visibility in green timeline.

4.1.2 Python Implementation

As we already mentioned in the previous section, after a first implementation using browser technologies, we decided to switch to a locally installable solution using the programming language python. This decision was based on the fact that a local installation can access all of the compute power available, including GPU access, and that there is no network

latency, which could slow down the annotation experience, involved. We chose python as the programming language, as it is the most prominent language in Artificial Intelligence (AI) and machine learning today. Libraries such as *Tensorflow*[6], *PyTorch*[7] and *Keras*[8] are some of the most widely used tools in the community and they are all based on python. Therefore, the incorporation of open source state-of-the-art AI models can be ensured. For the UI framework, we chose the Qt[9] platform, which is widely used in C++ as a cross platform interface. It has a python wrapper called PyQt[10], which enabled us to write our UI once. Also it works on all platforms (Windows, macOS, Linux). These considerations were taken into account in order to minimize so called "friction points" such as requiring future researchers and annotators to build from source or manage dependency issues. We aimed at creating an easily deployable system, which can be installed by simply downloading an executable file for the appropriate Operating System. To make this possible, we work with a deployment helper script by Michael Herrmann called fman build system[11]. It enables us to create portable executables with all of the dependencies bundled in for Windows, macOS and Linux. Section 4.2 will explain some of our UI decisions in detail.

Software Architecture

The software architecture of CVLA is based around the Singleton design pattern [Gam95] for data storage and UI event notifications (see yellow boxes in Figure 4.2). With these two Singleton objects, we can ensure that the data displayed in the different areas of the application is always in the same state, and that adjustments made in one particular view (e.g. moving a bounding box in the CanvasView) will automatically appear in the other views of the application (e.g. appearance of the keyframe indicator in the TimelineView).

Besides UI information relevant for keeping different views up to date like frame number and brightness values, our data Singleton also stores actual bounding box and tracking data. Figure 4.2 shows a class diagram of how our concepts of FollowedObjects and FrameMeta enable this data storage. Each individual object of interest is stored as a FollowedObject, with a Unique Identifier (UID) and a list of FrameMeta objects for every frame in the video. In the FrameMeta class we store informations such as its frame number, the associated bounding box coordinates, the FrameStatus (Keyframe, Untracked, Tracked, Lost), and whether the bounding box is visible in this frame. For keyframes, we store the propagation algorithm chosen by the annotator.

Propagation algorithms have to implement an abstract tracker interface (see Listing 4.1), which is designed so that CVLA's tracking capabilities can easily be expanded and be kept up to date with the state of the art.

[6] tensorflow.org
[7] pytorch.org
[8] keras.io
[9] qt.io
[10] riverbankcomputing.com/software/pyqt/
[11] build-system.fman.io

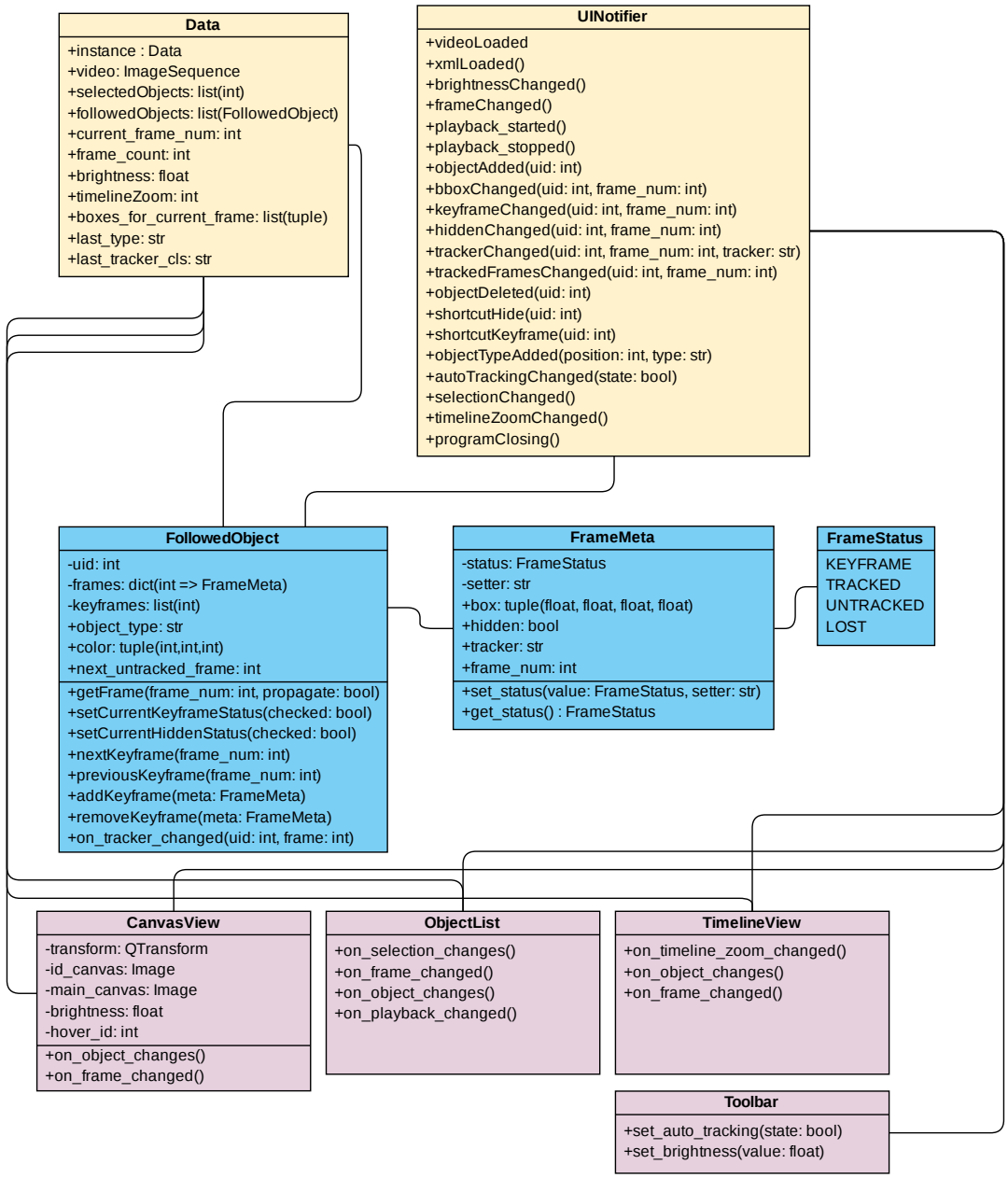

Figure 4.2: UML Class Diagram, showing the associations between Singleton classes (yellow), data structure classes (blue) and UI classes (purple).

```python
class AbstractTracker:
    # setup class variables
    def __init__(self):
        # tracker_implementation represents the class, that
        # actually does the tracking
        self.tracker_implementation = None

        # most data propagation algorithms work in a
        # consecutive order, last_frame_idx keeps track of the
        # last frame that was last updated using the tracker,
        # such that no unintentional jumps can happen (e.g.
        # frame 5 -> 10 is not allowed, but 5 -> 6 is)
        self.last_frame_idx = -1

        # the start and end meta data including bounding boxes
        self.start_meta = None
        self.end_meta = None

    # override this method if the implemented tracker makes
    # use of an end state keyframe (e.g. linear interpolation)
    def has_endstate(self):
        return False

    # initialize tracker with the start and end metadata
    def init_tracker(self, start, end=None):
        self.last_frame_idx = start.frame_num
        self.start_meta = start
        self.end_meta = end

    # to be implemented by subclass
    def update(self, meta):
        pass

    def is_initialized(self):
        return self.tracker_implementation is not None

    # to be implemented by subclass, this is needed for creating new
    # keyframes, which pre-select (copy) the tracker from the previous
    # keyframe
    def copy(self):
        pass
```

Listing 4.1: Tracker Interface

4.2 User Interface

Our tool aims to reduce user interaction by minimizing mouse clicks and incorporating tracking capabilities while also giving a clear overview of the annotation data in a timeline view. The following paragraphs explain the main functionalities and visualizations, which were chosen to aid in annotation speed and visual interpretation of the annotation data. A quick recap of our UI can be seen in Figure 4.3, where a screenshot of the main window

of CVLA shows the four major sections of the program:

- CanvasView: This area always shows the current frame of the video.
- TimelineView: In this area, time goes from left to right. Keyframes are shown as white circles, and the colored horizontal bars describe objects and their respective visible periods. This view's vertical scrollbar is linked with the ObjectList and the colors of the objects are shared between this view, the ObjectList and the CanvasView for reidentification purposes.
- ObjectList: Here, we show a list of all the annotated objects in the scene. A drop-down box on the left lets annotators choose or type what kind of object they are annotating. Next to it, the checkboxes show the visible state of the object at the current frame, and whether it is a keyframe for the given object. The drop-down menu on the right lets annotators decide which propagation algorithm they want to choose for the frames between the previous and next keyframe.
- Toolbar: In the toolbar, annotators can adjust the gamma correction of the video in order to increase visibility on dark footage, and they can enable automatic data propagation in a background thread.

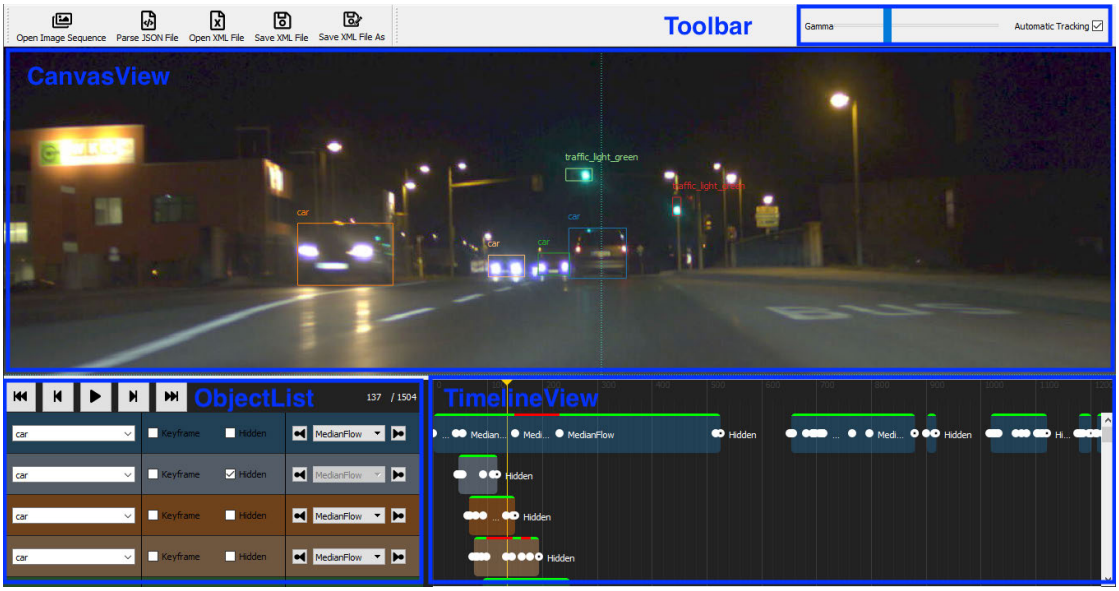

Figure 4.3: Screenshot of main window of the CVLA tool.

4.2.1 Information at a Glance

One of our focus points in designing the UI for CVLA was having the ability to have a quick understanding of the annotation data with regards to where keyframes were added,

as well as seeing at what point in time objects enter and leave the scene. Regarding keyframe information, we used the concept of a timeline with keyframes shown as white dots, like Shen proposed in his work for the video annotation tool BeaverDam [She16]. We also added the ability to see whether an object is hidden at a given moment in time without jumping to that frame using colored visibility bars. Whenever there is a colored bar in the timeline, it denotes that the corresponding object is annotated as visible, while a dark gray background refers to them being hidden.

This approach in combination with the ObjectList on the left makes it possible for annotators to quickly tell which kinds of objects are present in the scene, and by scrolling down they get an overview of the number of objects visible. Figure 4.3 shows the TimelineView of all the objects present in one of our videos. One can see that the topmost object has long periods without white dots, which means that the chosen tracking algorithm (MedianFlow [KMM10]) was accurate, and the annotator did not see the need to change the position or size of the bounding box. Another observation we can make, is that in this video most objects are visible at the beginning or at the end of the video, while the topmost object is visble almost throughout the duration of the video.

4.2.2 Minimizing Clicks

In his thesis for the BeaverDam tool [She16], Shen optimized the interface with the goal of minimizing the amount of clicks necessary when compared with VATIC [VR11]. We took his recommondations and implemented them into CVLA. When annotators want to draw a bounding box around an object, there is no need to press a dedicated "create object" button as in VATIC or CVAT. All an annotator needs to do is click and drag a bounding box around the object they want to annotate. This can speed up the annotation process significantly, especially when there are many objects in the scene. Additionally, CVLA does not require the user to choose the object class of the currently annotated object after every bounding box was drawn. Instead it pre-selects the class to be the same as the previous annotation. This class selection process was also proposed by Shen and it encourages users to first select all the objects of the same type before changing to a different type, which has proven to be a good way to reduce clicks [She16]. In CVLA, the same logic is applied for selecting a data propagation technique for new keyframes. The previously chosen propagation method is pre-selected, but it can be overwritten by user choice anytime.

4.2.3 Object Tracker Inclusion

As mentioned above, the inclusion of state-of-the-art trackers was a major component of our work with CVLA. In this section, we talk about the UI choices we made to let annotators easily select trackers and see the current tracking status of frames at a glance. As mentioned in the section above, we pre-selected a propagation algorithm upon annotation of a new object in the scene. Annotators then have two UI elements, which let them change this pre-selected algorithm to another one. One is the drop-down menu

Figure 4.4: Object Type (l) and Tracker (r) selection dropdown menus. Hidden and Keyframe checkboxes and keyframe navigation buttons.

in the ObjectList section to the left of the timeline (Figure 4.4 right), and the other can be seen in Figure 4.5. The name of the selected tracker is printed between keyframes, and when clicked on also triggers a drop-down menu.

At the top of Figure 4.5 our tracker-bar UI element can be seen. This bar represents a way to show the tracked status of an object at a given frame, without having to go to that frame to check the tracker status. For this visualization we employed a technique used in video editing software like Adobe Premiere[12], which shows a red or green bar to denote whether certain effects have already been rendered or not. Instead of showing the status of rendering effects, we use the tracker-bar element on top of each object track to indicate whether the bounding box has already been propagated by the selected algorithm or not.

Data propagation can be done in the background by enabling "Automatic Tracking" (top right of Figure 4.3), or when going through the video on a frame-by-frame basis.

4.2.4 Visibility Enhancement

Due to our focus on night scenes in the CVL project, the main kinds of images we are concerned with are predominantly dark. In some situations, this makes it hard to see the objects of interest clearly, which is why we included a gamma (γ) correction slider as a way to improve the visibility in dark areas of the image. In image processing, gamma

[12] www.adobe.com/at/products/premiere.html

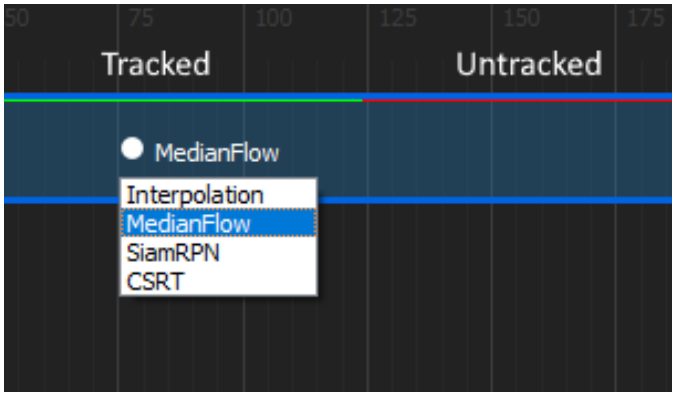

Figure 4.5: Tracker status bar on top: Green indicates tracked frames, red indicates untracked frames. Tracker selection drop-down by clicking on Keyframe dot.

correction is used to encode non-linear image intensities in a linear space to match the non-linearity of how humans perceive light intensities – our eyes notice slight differences in dark areas much more than in we do in bright areas [Han95]. A typical gamma correction function can be seen in equation 4.1 and a plot of it is seen in Figure 4.6.

$$V_{out} = V_{in}^{\gamma} \tag{4.1}$$

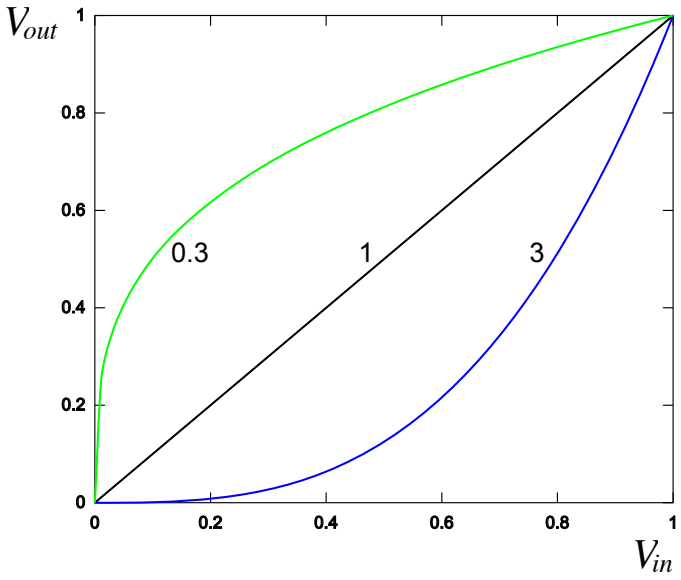

Figure 4.6: Plot of gamma correction curves at $\gamma = 0.3$, 1, 3 of gamma.

A nice feature of this gamma correction function, which we can use to our advantage,

is that with gamma values smaller than 1, dark intensity values get a much stronger brightness increase than large or bright intensity values. This makes the gamma slider, which can be seen in Figure 4.7, an easy way to brighten dark parts of the video in order to improve visibility on our typically dark footage.

Figure 4.7: Gamma slider and automatic tracking checkbox.

CHAPTER 5

Evaluation

In this Chapter, we examine the results of our tracker evaluation and the preliminary user study we conducted. For our tracker evaluation we focus on three main metrics (overlap, reset rate and mean time) to decide which algorithms to incorporate into our annotation tool. In the preliminary user study we focused on relative improvements between CVLA and Scalabel [YXC+18] regarding annotation time and accuracy.

5.1 Tracker Evaluation

In this section, we take a look at a number of state-of-the-art bounding box tracking algorithms and analyze their respective performances with regards to nighttime road scenes in non-urban environments. To evaluate these algorithms, we need an adequate dataset, representative of the scenes we might encounter in real life. Unfortunately, as we have explained in Section 3.3, we were not able to find a public dataset that meets all of the requirements we identified for our application in Section 3. Out of all of the considered datasets, VIPER [RHK17], appears as the most promising to us. Although this dataset does not feature real-world imagery, it does meet the rest of the requirements regarding scene environment and temporal density. Due to the fact that the data is computer-generated, we expect a domain shift between the synthetic and real-world footage. However, Richter et al. [RVRK16] have shown that that shift has only minor negative effects on performance values, when a small amount of real-world data is added to the dataset. Additionally, computer-generated ground truth data, like the images in the VIPER dataset, have perfect accuracy and contain no human bias, which is another reason why they are suitable as an evaluation basis.

The following subsections describe our considerations regarding metrics, and scene choice. First, we give some insight into the kinds of scenes we chose from the VIPER dataset to test against. Secondly we address the weakly correlated performance metrics *overlap*

and *robustness* described by Kristan et al. [KML+16] and Čehovin et al. [ČLK16] (see Section 5.1.3).

5.1.1 Clip Selection

As mentioned in Section 3.1.3, the VIPER dataset consists of 254,064 fully annotated images from over 184 virtual kilometers and five different environmental conditions. The CVL Project is mainly concerned with night footage, so we chose a subset of 28,348 images from 26 different recordings of the dataset. From the 30 available annotation classes, the three most common types of vehicles (car, van, truck) were chosen as most representative for our use case. To qualify for our evaluations, the instances of these vehicle annotations had to meet the following criteria:

- At least 10 consecutive frames
  Object appearances shorter than 10 frames long were not interesting for our use case as we wanted to find a suitable tracker for a semi-automatic annotation process with user input spaced further apart than half a second of video time.
- Bounding box size of at least 30 pixels and minimum bounding box side length of 3 pixels
  From our initial informal observations, we found that boxes smaller than 50 pixels in area or with side lengths shorter than 3 pixels usually did not belong to cars far in the distance but rather tiny parts of occluded cars, which might still be visible, for example, through leaves of a bush (see Figure 5.1).
- Bounding box size difference between consecutive frames of at most 20%
  Different to other tracking evaluation challenges such as VOT [KML+16], we had a semi-automatic annotation process in mind, where a human annotator is always in the loop. As such, we chose a maximum area difference of 20% between consecutive frames to minimize scenes with abrupt occlusions. The aim was to analyze scenes from VIPER, where a majority of the objects are visible the whole time. Our concern was not on automatic re-identification of shortly occluded boxes, as we want this to be done by a human annotator.

5.1.2 Tracker Selection

To assess which trackers to include in our annotation tool, we performed a test of five different state-of-the-art trackers regarding either their qualitative metrics or their speed. The trackers we chose to evaluate are:

- ATOM [DBKF19]
- SiamRPN [LYW+18]

Figure 5.1: Example of a bounding box with area of less than 50 pixels shown in red. Upper right corner zooms in on the area, and highlights the car parts visible through the bush leaves in light blue.

- MedianFlow [KMM10]
- KCF [HCMB14]
- CSRT [LVČZ+17]

The first two were chosen because of their good results in the VOT challenge [KML+16], whereas the last three were chosen for their fast update times.

ATOM

Danelljan et al. [DBKF19] have published the source code for their tracking algorithm ATOM. They propose a new approach for finding accurate bounding box positioning and sizing, going beyond a simple multi-scale bounding box approach and towards a higher-level knowledge based technique. The proposed algorithm consists of a target estimation and a classification component. The high-level knowledge is integrated into the target estimation component via offline learning, where they train the component to predict the overlap between the target object and an estimated bounding box. The classification component is trained and updated online to achieve good re-identification in the presence of distractors.

SiamRPN

Bounding boxes highly depend on the pose and viewpoint of the object, which often cannot be inferred by just a single input patch, but requires prior knowledge of the

object in question. To get accurate estimations of bounding boxes in concurrent frames, Li et al. [LYW+18] make use of a Siamese deep neural network called SiamRPN with a bounding box regression module. Li et al. have integrated prior knowledge to this regression module through heavy offline learning of bounding boxes.

MedianFlow

Kalal et al. [KMM10] have developed the MedianFlow tracking algorithm. It automatically tracks objects both forwards and backwards in time. Through this mechanism, it is able to calculate a Forward-Backward error, which helps in selecting a reliable trajectory, as well as successfully detecting tracking failures. First empirical tests that we conducted have shown, that this tracker works well for rigid objects whose appearance stays relatively stable and when the motion is fairly predictable. Additionally, thanks to its' forwards-backwards tracking, MedianFlow detects occlusions of objects well.

KCF

Henriques et al.'s [HCMB14] tracker KCF is a discriminative tracking-by-detection based algorithm. These types of algorithms work by learning a classifier, which discriminates between the object of interest and its surroundings by feeding it multiple patches of environment data and constantly updating the model for the object of interest, when new frames are added. To improve these algorithms, one could feed many different environment patches to the classifier, which will learn that these are negative examples. However, this can have a major impact on calculation speed. Henriques et al. have discovered a way to increase the number of negative patches, with no significant speed cost added to the bounding box regression by transforming the data into the Fourier domain (going from $O(n^3)$ to $O(n \log n)$, and therefore being able to achieve frame rates of more than 100fps. See [HCMB14] for further details.

CSRT

Lukežič et al. developed the Channel and Spatial Reliability Tracker (CSRT)[LVČZ+17], which builds on the idea of Discriminative Correlation Filters (dcfs) first introduced by Bolme et al. [BBDL10] and extends them with channel and spatial reliability maps. The spatial reliability map is used to restrict the correlation filter to the parts suitable for tracking. This is achieved by taking the initial bounding box and using the dominant colors within to create a foreground/background color model. Channel reliability weights are a way of ensuring that each feature channel gets an appropriately weighted filter response. With Histogram of Oriented Gradients [DT05] and color names [VDWSVL09], CSRT uses two well-known feature vectors.

5.1.3 Performance Metrics

Čehovin et al. [KML+16] and Kristan et al. [ČLK16] have researched various methods for comparing and evaluating trackers and concluded that a two-score system using two

weakly correlated measures such as accuracy and robustness is a better choice than combining these values into a hybrid measure. As described in their respective works, the two weakly correlated measures we use are mean overlap (eq. 5.1) and reset rate (eq. 5.3). Whereas the regular overlap measure is described in equation 5.2. R_t^G describes the ground truth region at time t, and R_t^T is the tracker's proposed region. Overlap is also known as Intersection over Union (IoU) as it describes the ratio between the intersection of the ground truth and proposed region with the union of both of these regions. Figure 5.2 shows a visual explanation of this. The IoU measure (ϕ_t) is described in equation 5.2, where R_t^G denotes the ground truth region at time t, and R_t^T is the tracker's proposed region. Figure 5.2 shows a visual explanation of this.

$$\hat{\phi} = \sum_t \frac{\phi_t}{N} \tag{5.1}$$

$$\phi_t = \frac{R_t^G \cap R_t^T}{R_t^G \cup R_t^T} \tag{5.2}$$

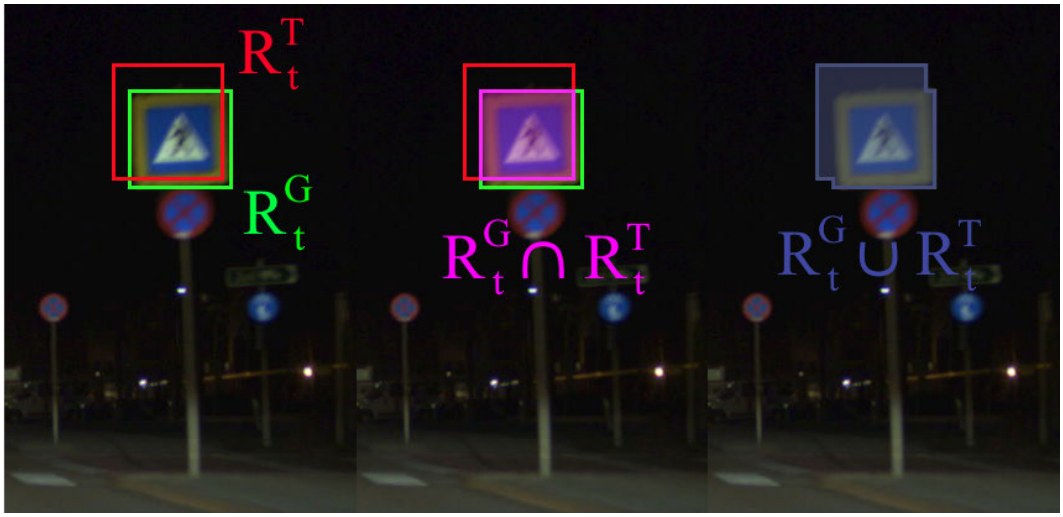

Figure 5.2: Visual explanation of IoU measure. Leftmost image shows ground truth (R_t^G, green) and tracker's proposed region (R_t^T, red), middle image shows region intersection ($R_t^G \cap R_t^T$, magenta), and rightmost image shows union ($R_t^G \cup R_t^T$, blue).

Reset rate ($\hat{r}$) describes the amount of frames where the IoU went below a threshold (τ) and had to be reset, divided by the total number of frames N. See figure 5.3 for an example of an IoU / time graph, with two resets. The reset threshold we chose was 51%, and unlike [KML+16] who disregard measurements of the first five frames after

re-initialisation, we take all frames except for initialisation frames into account. This decision was made because we are focusing on a semi-automatic approach with a human in the loop and we noticed that the frames immediately after tracker initialisation tend to be very important indicators of subsequent tracker success. A human in the loop would notice bad tracking capabilities after he or she reset the bounding box and by utilizing every frame except initialisation frames, we account for this as well.

A third important metric is mean execution time as interactivity is important when working with a human in the loop. Unless a tracker produces 100% accurate results all the time, it is paramount that the annotator does not have to wait too long for the tracker results to show up in order to be able to make necessary changes in a timely fashion.

$$\hat{r} = \frac{||\{t|\phi_t < \tau\}_{t=1}^{N}||}{N} \tag{5.3}$$

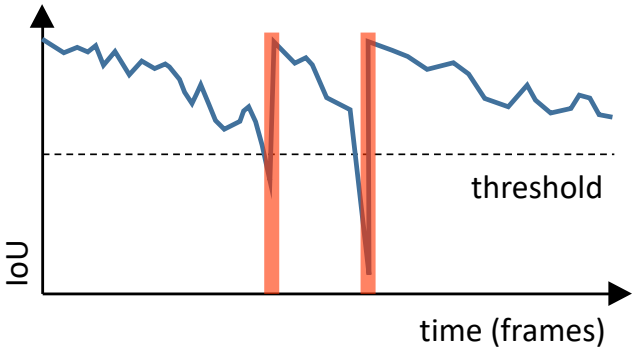

Figure 5.3: Plot of IoU over time, with two tracker resets – where IoU falls below threshold – shown in red.

5.1.4 Evaluation Results

Our evaluations on night scenes from the VIPER dataset suggest that current state-of-the-art trackers (e.g., ATOM [DBKF19],SiamRPN [LYW+18]) are not necessarily more suitable for our application than the simpler MedianFlow [KMM10] approach. Table 5.1 shows the results of this evaluation. ATOM [DBKF19] was the best performing tracker with a mean IoU of 72.6% and a reset rate of 7.3%. However, MedianFlow [KMM10] performed nearly as well, with 71.7% mean IoU and 7.2% reset rate while being much faster at 15.4ms compared to 88.2ms.

When further analysing the evaluation results, we found that MedianFlow provides a more predictable and stable result as long as the appearance of the object does not change too drastically, which is generally the case for traffic participants far away from

Tracker	Mean IoU	Reset rate	Mean Time
KCF [HCMB14]	54.8%	23.5%	17.1ms
SiamRPN [LYW+18]	59.4%	20.5%	364ms
CSRT [LVČZ+17]	70.8%	10.4%	59.5ms
MedianFlow [KMM10]	71.7%	**7.2%**	**15.4ms**
ATOM [DBKF19]	**72.6%**	7.3%	88.2ms

Table 5.1: Tracker Evaluation results, best performing values per column shown in bold.

the camera (see Figure 5.4). ATOM performs better for scenes where appearance changes are more pronounced (see Figure 5.5).

Since the objects in the CVL Project are mostly small and far away (see Section 3.2.2) and the speed performance is of about 15ms, we have chosen to include MedianFlow as the standard tracker in our annotation tool CVLA (see Chapter 4).

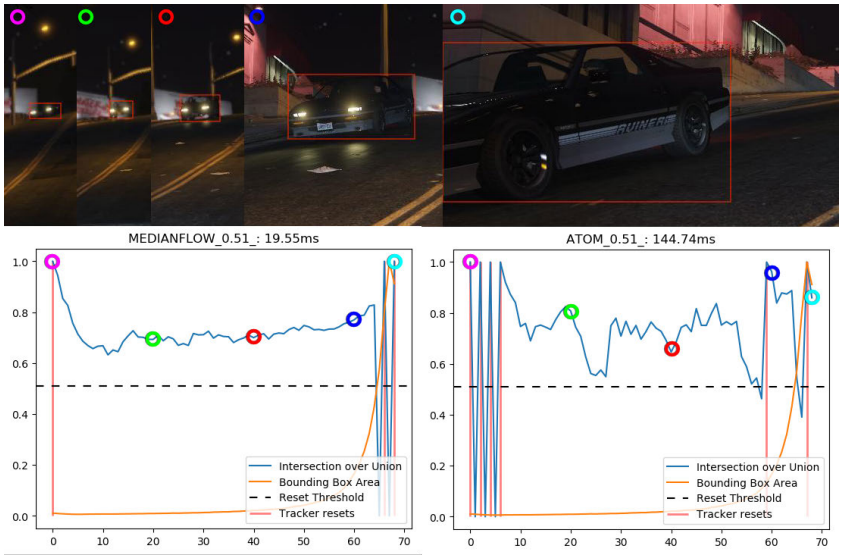

Figure 5.4: Scene (top) with oncoming traffic. ATOM (bottom right) has problems when car is still small and only headlights are visible. MedianFlow (bottom left) delivers more stable tracking results in this situation.

5.2 Preliminary User Study

To compare CVLA to a similar video annotation tool, we performed a preliminary user study with two annotators, annotating 12 videos with 3349 frames and 150 object tracks.

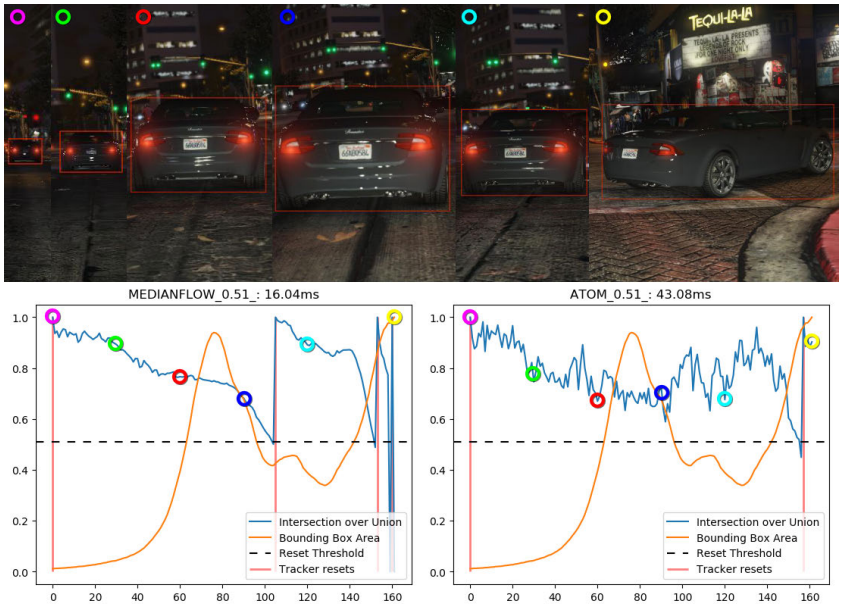

Figure 5.5: Scene (top) with preceding traffic. MedianFlow (bottom left) can follow the car for about 100 frames before re-initialisation, whereas ATOM does not lose track until the very last moments, when the car's appearance changes drastically due to turning.

We again used the VIPER dataset as our ground truth data and chose nighttime scenes on non-urban roads. As a comparison tool, we picked Scalabel [YXC+18] as it was successfully used to annotate over 100,000 images in the BDD100k dataset. We focused on comparing the annotation process with regards to time, keystrokes, mouse movement, clicks and annotation accuracy; the latter is represented by mean IoU. In order to ensure a fair comparison of these values, we had to make sure that the annotators were focusing on the same 150 object tracks regardless of the tool used. This was accomplished by displaying a visual anchor (ground truth downsized to 40% of actual size) over the objects of interest (see Figure 5.6). We expect this overlay to introduce a bias towards more accurate annotations and higher IoU values. However, as the overlay was shown in both annotation tools, we do not expect this bias to affect the relative differences in comparing the annotation accuracy of both tools.

Since we focused on relative improvements between tools, we made sure that the individual annotators worked on the same test sequences in both tools (i.e. annotator A did sequence 1-4 and annotator B did sequence 5-12 in both tools). To track annotation times and clicks, we used a mouse tracking application called "Mousotron" [1], which enabled us to keep score of the number of clicks, keystrokes, travelled mouse distance, and scroll wheel invocations.

[1] http://www.blacksunsoftware.com/mousotron.html

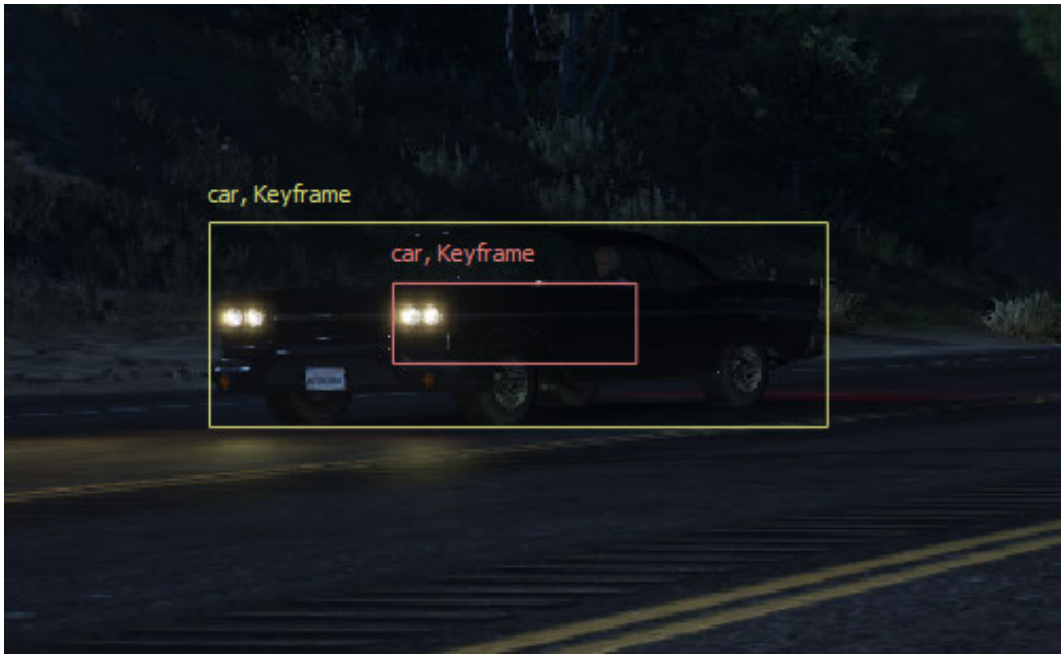

Figure 5.6: Visual anchor in red, user annotation in yellow.

5.2.1 Results

Our evaluation of the annotation time and accuracy suggests that using CVLA for video annotation results in faster annotation speeds as well as more accurate data. Table 5.2 contains a summary of our evaluation. The total time needed to annotate the 3349 chosen frames in two different tools divided between two annotators was 18 hours and 41 minutes. 6 hours and 55 minutes were spent in CVLA, and 11 hours and 46 minutes in Scalabel, thus resulting in a speed increase of about 1.69. Additionally, using our tool, the mean IoU increased by about 1.06. Mouse (click, scroll) and keyboard invocations could be significantly reduced (2.28), whereas the distance the mouse moved over the screen was only decreased by a factor of 1.04.

	Time	IoU	Invocations	Dist.
Scalabel [YXC+18]	11h 46m	78.74%	63965	1.27km
CVL [Ours]	6h 55m	83.46%	28001	1.22km
Improvement factor	**1.69**	**1.06**	**2.28**	**1.04**

Table 5.2: Preliminary User Study results. CVLA performs better in all measured categories.

Figures 5.7, 5.8 and 5.9 show the measured values per video sequence and tool, while

also indicating which values refer to which annotator. It can be seen that there are no inherent differences between the two annotators, and that the improvements rather vary depending on the underlying video data. In Figure 5.7 we show the average annotation time per bounding box, where we count each bounding box per frame separately (e.g. 3 objects of interest each visible on 5 frames result in 15 bounding boxes). This graph shows that CVLA had a shorter time per box on all but one video of our test set with speedup factors ranging from 0.9 to 2.6 (video 7 and 12 respectively).

Mouse and Keyboard invocations can be seen in Figure 5.7. We observed that invocations varied a bit less when using CVLA compared to Scalabel (relative standard deviation of 31% vs. 43%), while CVLA always needed far fewer invocations (improvement factor between 1.3 and 3.9). This consistent improvement regarding mouse and keyboard invocations can most likely be explained by the fact that we keep zoom and pan information across frames, while Scalabel loses this information on frame changes. It resets the zoom and shows the whole frame resized to the dimensions of the view port.

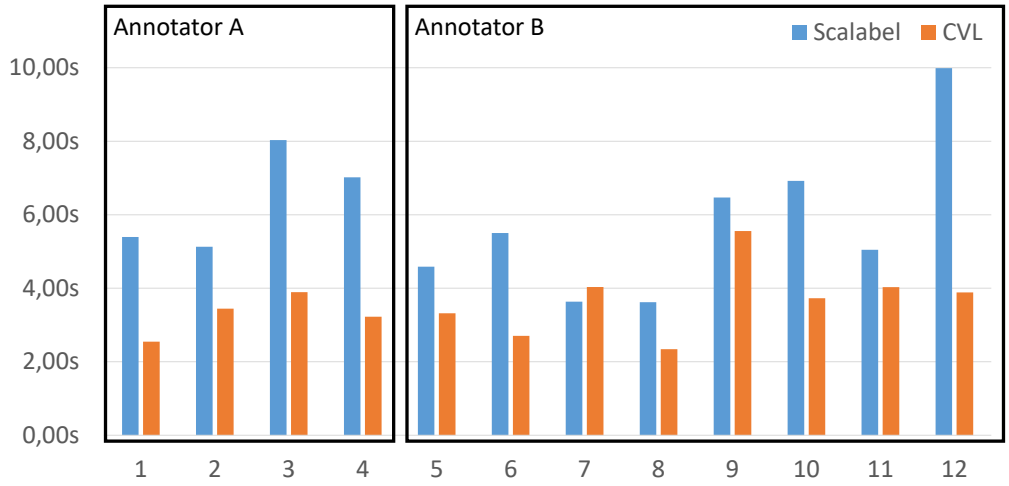

Figure 5.7: Annotation time per bounding box per video and annotation tool.

Mean IoU per video is shown in Figure 5.9. Here we see more consistent improvements, ranging from 1.02 to 1.10. The main contribution of this consistent IoU increase can be attributed to bounding boxes with relatively small areas (up to 400 pixels) as shown in Figure 5.10, where the mean IoU is 72.16% in CVLA vs. 57.68% in Scalabel. There are two likely explanations for this rather large increase for small bounding boxes: (i) unlimited zoom in CVLA and (ii) fairly consistent visual appearance for smaller objects, which means that they are easier to track. Figure 5.11 shows a box plot of the recorded IoU values in our preliminary study, and we can see that mean values and variance for most bounding box sizes are comparable, but for the smallest areas (less than 400 pixels)

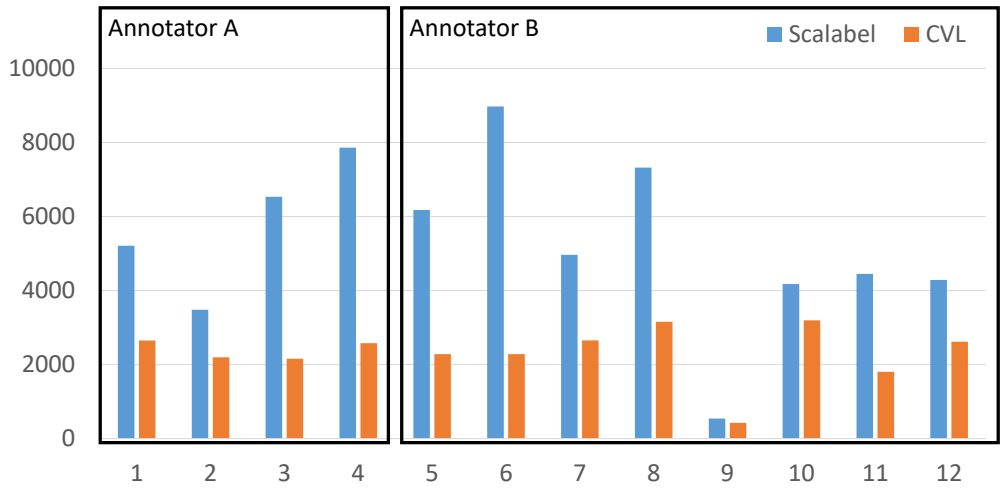

Figure 5.8: Mouse and keyboard invocation count (summed up) per video and annotation tool

we can see a clear improvement in mean IoU (64% to 73%) with less variance or more consistency (standard deviation 14.5% to 9.9%). Another interesting observation we can make from Figure 5.11, is that the number of annotated bounding boxes per tool does vary a little bit even though we showed the visual anchor seen in Figure 5.6 as a helper to identify all of the AOI. Luckily this difference is less than 3% 7368 (Scalabel) vs. 7580 (CVLA), and it does not change the broader meaning of the results.

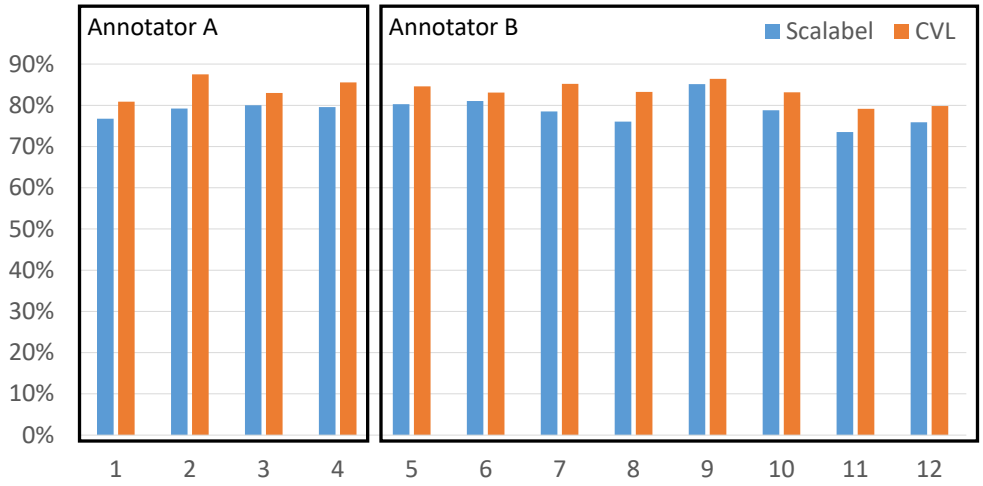

Figure 5.9: Mean IoU value comparison per video and annotation tool.

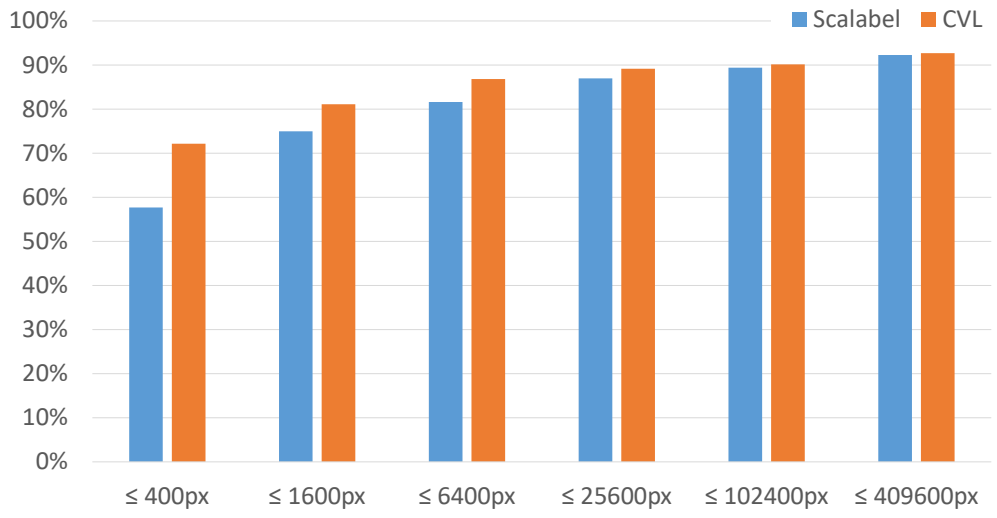

Figure 5.10: Mean IoU value comparison grouped by maximum pixel area.

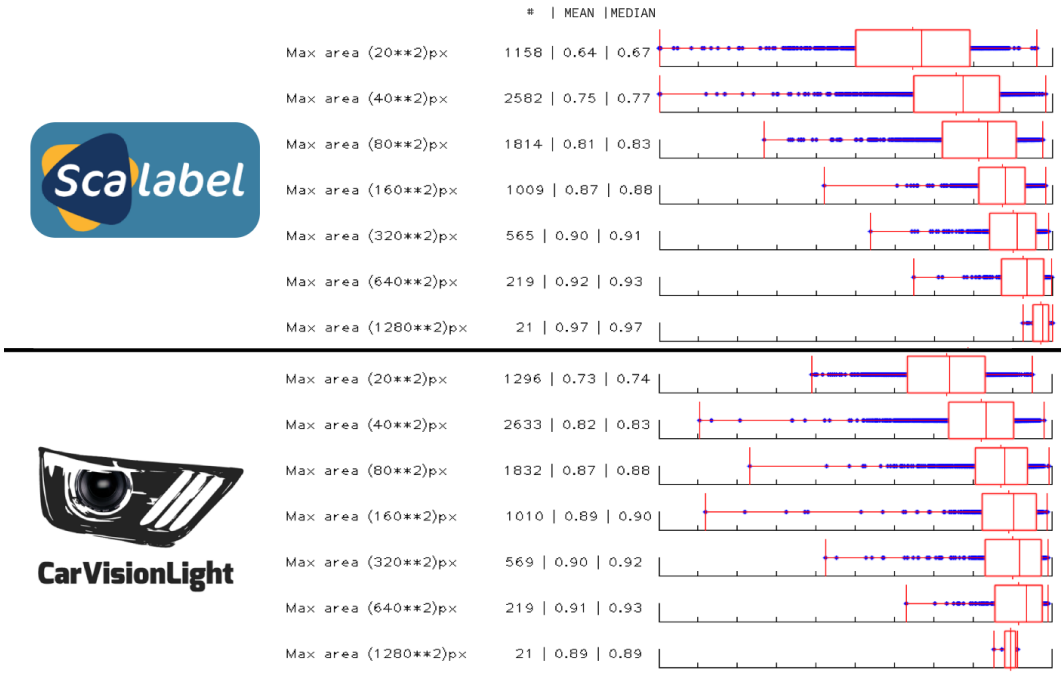

Figure 5.11: IoU values as box plot graph from Scalabel (top) and CVLA (bottom).

CHAPTER 6

Conclusion and Future Work

In this chapter, we first give a short summary of the individual chapters of the diploma thesis. We provide some conclusions on our findings from the tracker evaluation and our preliminary user study. Finally, we give an outlook onto possible future topics of research.

6.1 Summary

Chapter 1: In the first Chapter we gave a short introduction and problem statement to explain the motivation behind our need for creating a nighttime road-scene dataset and the need to create our own annotation tool CVLA to speed up the dataset creation process.

Chapter 2: In this Chapter we documented our literature review on the current state of the art in the scientific community regarding semi-automatic annotation algorithms ranging from semantic segmentation to bounding box trackers and took a look at publicly available video annotation tools. Regarding the annotation tools, we focused on platform choice and data propagation capabilities.

Chapter 3: This Chapter documents our literature review of publicly available datasets, looking at their respective sizes, scene variety and temporal density. We also shared details of our own CVL dataset with statistics about bounding box sizes and class distributions.

Chapter 4: Here we talk about the implementation of our Annotation Tool and the UI decisions we made to optimize the work flow of annotating videos. We provide some details about the first version of our tool, which was based on web technologies, and describe why we pivoted to a locally installable program based on the python programming language. Furthermore, features like our tracker inclusion and timeline overview are presented.

Chapter 5: This Chapter talks about our tracker evaluation, in which we tested state-of-the-art tracking algorithms on synthetic nighttime road scenes to assess their applicability for our annotation tool. Additionally we share the promising results of a preliminary user study we conducted to measure annotation times and speed using our tool compared to Scalabel [YXC+18].

6.2 Synopsis of our Contributions

We introduced a semi-automatic video annotation tool (CVLA) with a focus on nighttime traffic scenes. Our tool includes state-of-the-art tracking algorithms, which were selected based on an evaluation analysis of the VIPER dataset [RHK17]. Furthermore, it features a user interface that focuses on minimizing the number of clicks and keystrokes needed to annotate video data. We have conducted a preliminary user study based on two users, which has shown promising results regarding the speed and accuracy increase of CVLA – using tracking algorithms – compared to an existing tool (Scalabel [YXC+18]) – using linear interpolation as its data propagation mechanism. On average, the annotations created with our tool have been 1.06 times more accurate in terms of mean IoU value, while taking 1.69 times less time to create. The average number of mouse and keyboard invocations was reduced by a factor of 2.28.

6.3 Future Work

To confirm the results from our preliminary user study, a more thorough study with a larger group of representative annotators is a possible next step. We plan to perform a larger user study with at least 10 participants with varying levels of experience in annotating videos. Another interesting question, which we did not cover in our preliminary study, is the discoverability of AOI in the scene. In our preliminary study we displayed an overlay to help the study subjects identify the objects in need of annotation. It would be interesting to study whether our gamma correction or other visibility enhancing techniques like using a wavelet transform [ŁBHA13] or illumination map estimation [GLL16] would help with identifying AOI.

List of Figures

List of Tables

Acronyms

ABS Anti-lock Braking System. 1

AFS Advanced Front-lighting System. 1

AI Artificial Intelligence. 40

AOI Areas of Interest. 1, 9, 59, 64

API Application Programming Interfaces. 38

ATOM Accurate Tracking by Overlap Maximization. 38, 51, 54–56, 67

CGI Computer-Generated Imagery. 19

CNN Convolutional Neural Network. 7

CPU Central Processing Unit. 38

CSRT Channel and Spatial Reliability Tracker. 52

CV Computer Vision. 19

CVAT Computer Vision Annotation Tool. 14, 16, 23, 44

CVF Cost Volume Filtering. 5–9, 65

CVL CarVisionLight. 1–3, 9, 19, 21, 24, 27, 28, 31–34, 45, 50, 55, 63, 66

CVLA CarVisionLight Annotator. 3, 19, 37, 38, 40, 43, 44, 49, 55, 57–59, 61, 63, 64, 66, 67, 69

DAVIS Densely Annotated VIdeo Segmentation. 8

dcf Discriminative Correlation Filter. 52

FAVOS Fast and Accurate Online Video Object Segmentation. 8

FCN Fully Convolutional Network. 7

GPU Graphics Processing Unit. 38, 39

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 7

IoU Intersection over Union. 53

KCF Kernelized Correlation Filters. 38, 52

LIDAR Light Detection and Ranging. 20, 21, 65

LKA Lane Keeping Assist. 1

Lucid Lucid Data Dreaming for Video Object Segmentation. 8

MOT Multiple Object Tracking. 9

OSVOS One-Shot Video Object Segmentation. 8, 9

R-CNN Region Convolutional Neural Network. 7

RADAR Radio Detection and Ranging. 21

SiamRPN Siamese Region Proposal Network. 38, 52, 54

SIFT Scale-Invariant Feature Transform. 2

UI User Interface. 4, 5, 10–14, 16, 37, 40–44, 63, 66

UID Unique Identifier. 40

URL Uniform Resource Locator. 38

VATIC Video Annotation Tool from Irvine California. 10–12, 14, 44, 65

VIPER VIsual PERception. 24, 33, 37, 49, 50, 54

ViTBAT Video Tracking and Behavior Annotation Tool. 11, 12, 16, 65

VOT Visual Object Tracking. 9

WebGL Web Graphics Library. 38

Bibliography

[AT13] Alexander Andreopoulos and John K Tsotsos. 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8):827–891, 2013.

[BBDL10] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2544–2550. IEEE, 2010.

[BFC09] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[BHRG12] Nicole Brosch, Asmaa Hosni, Christoph Rhemann, and Margrit Gelautz. Spatio-temporally coherent interactive video object segmentation via efficient filtering. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 418–427. Springer, 2012.

[BNFD16] Tewodros A Biresaw, Tahir Nawaz, James Ferryman, and Anthony I Dell. Vitbat: Video tracking and behavior annotation tool. In *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 295–301. IEEE, 2016.

[Bro16] Nicole Brosch. *Spatio-temporal Video Analysis for Semi-automatic 2D-to-3D Conversion*. PhD thesis, Technische Universität Wien, 2016.

[BSG16] Nicole Brosch, Tanja Schausberger, and Margrit Gelautz. Towards perceptually coherent depth maps in 2d-to-3d conversion. *Electronic Imaging*, 2016(5):1–11, 2016.

[CHX+17] Long Chen, Xuemin Hu, Tong Xu, Hulin Kuang, and Qingquan Li. Turn signal detection during nighttime by cnn detector and perceptual hashing tracking. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3303–3314, 2017.

[ČLK16] Luka Čehovin, Aleš Leonardis, and Matej Kristan. Visual object tracking performance measures revisited. *IEEE Transactions on Image Processing*, 25(3):1261–1274, 2016.

[CLL+19] Zhengping Che, Max Guangyu Li, Tracy Li, Bo Jiang, Xuefeng Shi, Xinsheng Zhang, Ying Lu, Guobin Wu, Yan Liu, and Jieping Ye. D^2-city: A large-scale dashcam video dataset of diverse traffic scenarios. *Computing Research Repository (CoRR)*, abs/1904.01975, 2019.

[CLS+19] 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. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8740–8749. IEEE, 2019.

[CMPT+17] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 221–230. IEEE, 2017.

[COR+16] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223. IEEE, 2016.

[CTH+18] J. Cheng, Y.-H. Tsai, W.-C. Hung, S. Wang, and M.-H. Yang. Fast and accurate online video object segmentation via tracking parts. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7415–7424. IEEE, 2018.

[DBKF19] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4660–4669. IEEE, 2019.

[DRM+19] Patrick Dendorfer, Seyed Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian D. Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. CVPR19 tracking and detection challenge: How crowded can it get? *Computing Research Repository (CoRR)*, abs/1906.04567, 2019.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE, 2005.

[Gam95] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pages 157-159. Pearson Education India, 1995.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. IEEE, 2014.

[GLL16] Xiaojie Guo, Yu Li, and Haibin Ling. Lime: Low-light image enhancement via illumination map estimation. *IEEE Transactions on Image Processing*, 26(2):982–993, 2016.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[Han95] Hermann O Handwerker. Allgemeine sinnesphysiologie. In *Physiologie des Menschen*, pages 195–215. Springer, 1995.

[HCMB14] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2014.

[HRB+12] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, 2012.

[HST10] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *European Conference on Computer Vision (ECCV)*, pages 1–14. Springer, 2010.

[JLM+18] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *European Conference on Computer Vision (ECCV)*, pages 784–799. Springer, September 2018.

[KBI+17] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. In *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

[KML+16] Matej Kristan, Jiří Matas, Aleš Leonardis, Tomáš Vojíř, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, 2016.

[KMM10] Zdenek Kalal, Krystian Mikolajczyk, and Jiří Matas. Forward-backward error: Automatic detection of tracking failures. In *International Conference on Pattern Recognition*, pages 2756–2759. IEEE, 2010.

[KPL+13] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiří Matas, Fatih Porikli, Luka Čehovin Zajc, Georg Nebehay, Gustavo Fernandez, Tomáš Vojíř, Adam Gatt, Ahmad Khajenezhad, Ahmed Salahledin, Ali Soltani-Farani, Ali Zarezade, Alfredo Petrosino, Anthony Milton, Behzad Bozorgtabar, Bo Li, Chee Seng Chan, Cherkeng Heng, Dale Ward, David Kearney, Dorothy Monekosso, Hakki Can Karaimer, Hamid R. Rabiee, Jianke Zhu, Jin Gao, Jingjing Xiao, Junge Zhang, Junliang Xing, Kaiqi Huang, Karel Lebeda, Lijun Cao, Mario Edoardo Maresca, Mei Kuan Lim, Mohamed El Helw, Michael Felsberg, Paolo Remagnino, Richard Bowden, Roland Goecke, Rustam Stolkin, Samantha Yueying Lim, Sara Maher, Sebastien Poullot, Sebastien Wong, Shin'Ichi Satoh, Weihua Chen, Weiming Hu, Xiaoqin Zhang, Yang Li, and Zhiheng Niu. The visual object tracking vot2013 challenge results. In *International Conference on Computer Vision (ICCV) Workshops*, pages 98–111, 2013.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[ŁBHA13] Artur Łoza, David R Bull, Paul R Hill, and Alin M Achim. Automatic contrast enhancement of low-light images based on local statistics of wavelet coefficients. *Digital Signal Processing*, 23(6):1856–1866, 2013.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. IEEE, 1999.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440. IEEE, 2015.

[LVČZ+17] Alan Lukežič, Tomáš Vojíř, Luka Čehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4847–4856. IEEE, 2017.

[LYW+18] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8971–8980. IEEE, 2018.

[NORBK17] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, pages 4990–4999. IEEE, 2017.

[PPTM+16] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732. IEEE, 2016.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[RHK17] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *International Conference on Computer Vision (ICCV)*, pages 2213–2222. IEEE, 2017.

[RSM+16] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243. IEEE, 2016.

[RVRK16] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, pages 102–118. Springer, 2016.

[Sek18] Sekachev, Boris and Manovich, Nikita. Computer vision annotation tool, 2018. [Online; https://github.com/opencv/cvat/, accessed 26-March-2020].

[She16] Anting Shen. Beaverdam: Video annotation tool for computer vision training labels. *EECS Department, University of California, Berkeley, Master Thesis*, 2016.

[VDWSVL09] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.

[VPR13] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013.

[VR11] Carl Vondrick and Deva Ramanan. Video annotation and tracking with active learning. In *Advances in Neural Information Processing Systems*, pages 28–36, 2011.

[XKSG16] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3d to 2d label transfer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3688–3697. IEEE, 2016.

[YXC+18] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *Computing Research Repository (CoRR)*, abs/1805.04687, 2018.