Utrecht University

MASTER'S THESIS

Automated Privacy-Preserving Video Processing through Anonymized 3D Scene Reconstruction

Author:
Lucía CONDE MORENO

Supervisor:
Ronald W. POPPE

External supervisor:
Joop SNIJDER

Second examiner:
Remco C. VELTKAMP

Degree: MSc Informatica (Computer Science), program: Game and Media Technology

Department of Information and Computing Sciences
Faculty of Science

ICA-6194990

September 11, 2019

Research Center AI
Info Support B.V.

Abstract

We present a novel method for full video anonymization, with the goal of enabling the storage of video recordings in compliance with current data protection regulations. Contrary to the general approach in image and video anonymization techniques, usually focused on processing sensitive image areas, the proposed method aims at inferring the 3D locations of all relevant objects (specifically, people and vehicles) at each frame within the video scene. This way, only the position and motion information is preserved, guaranteeing the anonymity of the recreated scene while also providing sufficient information of the original scene for further analysis.

The method is mainly based on an initial object detection and tracking procedure, plus a depth estimation process, to ultimately compute an approximate location of each object on the ground plane. As anonymity is already ensured thanks to the nature of the method, we focus on evaluating how robust the method is at finding all relevant objects, and how accurately it can locate those objects in the 3D space. We first analyzed the performance of each module individually, with a particular focus on the predictive models for object detection and depth estimation. Ultimately, we evaluated the estimated object locations in terms of matching and proximity to the ground truth, and how the usage of object tracking influenced these results.

We evaluated three well-known object detectors, from which we selected the one able to perform segmentation that also achieved state-of-the-art results for the relevant objects. We compared several models of a monocular depth estimator trained on different subsets, achieving a similar performance with all models for scenes containing our objects of interest. In addition, we implemented a camera self-calibration method that was able to predict the focal length at a small error rate, for varied camera angles. Finally, we analyzed the overall performance of the pipeline by comparing the detected objects and their locations in the 3D space to the true ones: when using tracking, the method achieved an accuracy of 60% at a matching precision of 1.6 meters on the ground plane on our test setting. Although the final output is strongly affected by depth outliers when the input scene significantly differs from that of the training data, the results are promising when there are similarities between the two, especially in terms of viewpoint.

With further improvements, particularly on the depth estimation process, this method will enable the unlimited storage of video feeds that could be analyzed in the long-term for traffic and pedestrians' movement patterns, with the purpose of improving, for instance, crowd and access control logistics in public spaces.

Contents

List of Acronyms

CCTV	**C**losed-**c**ircuit **t**ele**v**ision
mAP	**m**ean **A**verage **P**recision
RGBD	**R**ed-**G**reen-**B**lue-**D**epth
GDPR	**G**eneral **D**ata **P**rotection **R**egulation
ROI	**R**egion **o**f **I**nterest
CNN	**C**onvolutional **N**eural **N**etwork
GMM	**G**aussian **M**ixture **M**odel
HMM	**H**idden **M**arkov **M**odel
SIFT	**S**cale-**I**nvariant **F**eature **T**ransform
HOG	**H**istogram of **O**riented **G**radients
SVM	**S**upport **V**ector **M**achine
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etwork
SPPnet	**S**patial **P**yramid **P**ooling **net**work
YOLO	**Y**ou **O**nly **L**ook **O**nce
SSD	**S**ingle **S**hot multibox **D**etector
FPN	**F**eature **P**yramid **N**etwork
RPN	**R**egion **P**roposal **N**etwork
FCN	**F**ully **C**onvolutional **N**etwork
fps	**f**rames **p**er **s**econd
SURF	**S**peeded **U**p **R**obust **F**eature
LBP	**L**ocal **B**inary **P**attern
SfS	**S**hape **f**rom **S**hading
MRF	**M**arkov **R**andom **F**ield
SORT	**S**imple **O**bject **R**ealtime **T**racking
MOT	**M**ulti-**O**bject **T**racking
SLAM	**S**imultaneous **L**ocalization **a**nd **M**apping
API	**A**pplication **P**rogramming **I**nterface
JSON	**J**ava**S**cript **O**bject **N**otation
RLE	**r**un-**l**ength **e**ncoding
CSV	**c**omma-**s**eparated **v**alue
IoU	**I**ntersection **o**ver **U**nion
AP	**A**verage **P**recision
RMSE	**R**oot-**m**ean-**s**quare **e**rror
MOTP	**M**ulti-**O**bject **T**racking **P**recision
MOTA	**M**ulti-**O**bject **T**racking **A**ccuracy
LSTM	**L**ong **S**hort-**T**erm **M**emory

Chapter 1

Introduction

During the last decades, video surveillance systems have seen a worldwide rise due to a drastic reduction of their cost and an improvement of their image quality. Closed-circuit television (CCTV) cameras have been extensively implemented, particularly in public spaces, with the main purpose of tackling crime but also for the realization of other tasks focused on public safety, such as the enforcement of traffic regulations or the prevention of accidents (Senior et al., 2005). These systems are generally used for either real-time inspection of the video feed or for the storage and subsequent analysis of the recordings (Ribaric, Ariyaeeinia, and Pavesic, 2016).

Although the legitimate purpose of the usage of video surveillance is understandable, the pervasiveness of this kind of systems implies an invasion of the privacy of individuals, even if they are presumably innocent of taking part in any event pertinent to the surveillance task. These individuals involuntarily run the risk of their image and actions being indiscriminately captured. This not only entails the hazard of their personal information being potentially abused for illegitimate intentions, but it can also threaten their autonomy: people are likely to change their behavior and avoid engaging even in licit or beneficial activities when they know there is a possibility that they are being observed, or they are knowingly under surveillance, and they can be identified, due to their fear of prosecution or legal consequences. This is known as the "chilling effect", and it discourages individuals from exercising their freedom of will (Penney, 2016).

As a result, there is a general agreement on the importance of a trade-off between security and privacy in video surveillance or any related public safety application (Cavallaro, 2007; Birnstill et al., 2015). An appropriate balance must be found between both aspects so that the intended task can still be adequately performed without exposing the identity of individuals. This kind of exposure would withhold the individuals' right to intimacy and could even lead to a compromise of their sensitive personal data.

Depending on the context, the term "privacy" has received several different definitions, from its very first one of "the right to be alone" (Warren and Brandeis, 1890), to "intimacy", "secrecy" or "the limited access to the self" (Solove, 2008). Nonetheless, all of its definitions converge in one common aspect: the individual's control over their own personal information.

To provide this control to anyone who could be subject to unnecessary (and thus often unwanted) video surveillance, suitable privacy protection measures should be taken when processing the surveillance video feeds, and primarily if these video files are foreseen to be stored afterwards. This way, the personal, private information of

the exposed individuals can be prevented from becoming publicly available (Padilla-López, Chaaraoui, and Flórez-Revuelta, 2015). Moreover, visual privacy protection would not only benefit the citizens under surveillance but also the entities making use of these kinds of systems, as they can ensure their own capability of performing their desired tasks without risking legal consequences.

It is therefore clear that the preservation of privacy within video surveillance must be enforced. The main concern when defining suitable visual privacy preserving measures, however, is that privacy protection *per se* is an ill-posed problem, due to the lack of a universal meaning of privacy (Senior et al., 2005), whose significance drastically varies among people and also depending on the context.

In consequence, there is no clear agreement yet on how to adequately implement visual privacy protection, nor any standard methodology or approach to tackle this problem. Traditionally, data encryption schemes have been implemented, which indeed limits the access to the original data but does not prevent the misuse by authorized personnel (Cavallaro, 2007) or anyone in possession of the corresponding encryption key. Hence, an irreversible processing would guarantee the privacy of the protected individuals and ultimately be more appropriate regardless of the context of application -as long as a sufficient utility of the resulting video is maintained.

An approach to irreversibly anonymize a video recording would be to manipulate the sensitive areas of the frames in a way that personal data is concealed. In addition, automating this procedure would minimize the overall number of individuals or entities that are able to access and visualize the original, unprotected videos, decreasing the risk of misuse or leakage of the personal data that is portrayed. This is the reason why modern privacy protection solutions mostly rely on computer vision algorithms to automatically detect these sensitive regions on the image, which are later modified in a proper, privacy-preserving manner.

Nevertheless, it is noteworthy that this kind of approach by itself has a considerable shortcoming: to ensure anonymity, or at least to impede personal identification, the detection of sensitive regions would have to be perfect (if the intention is to fully rely on the automation of the process and not to include any manual verification). This is due to the fact that, even if the detection of a certain sensitive region (for instance, a face) fails only for a single frame out of a lengthy video, this frame would allow the identification of the individual, thus making the anonymization procedure ineffective. Unfortunately, this perfect accuracy cannot be absolutely guaranteed with the currently existing computer vision tools, plus the overall accuracy of the detection might also vary depending on the context of the scene (Hoiem, Chodpathumwan, and Dai, 2012). To avoid this dependency on the detection process, an intuitive approach would be to generate an alternative version of the video recording where only the necessary information is transferred, leaving out the personal identifiers. Although this would in principle guarantee privacy protection, depending on the application it could entail the loss of useful non-sensitive information, jeopardizing the utility of the resulting output.

1.1 Scope

Taking all these aspects into consideration, we propose a privacy-preserving video processing method capable of retaining the required information for the task by transferring it without any identifiable properties. As opposed to the main trend

in previous research, our method aims at protecting the privacy not by modifying the original video itself, but by outputting a separate source: a three-dimensional recreation of the recorded scene where the relevant moving elements are presented with anonymous visual representations. In this manner, an isolated detection failure would simply cause an inaccurate scene reconstruction in a per-frame basis, which would have a negligible influence on the scene perception. At the same time, we will avoid the risk of exposing the identity of the recorded individuals or any other sensitive personal information. Likewise, after the application of this method, if the original video is deleted, the anonymization can be considered to be irreversible.

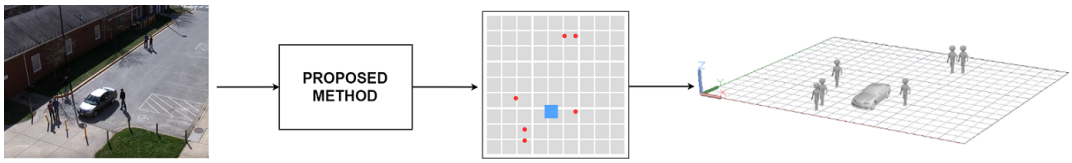

FIGURE 1.1: Representation of the expected input and output of the proposed system, where the 3D locations of the objects of interest are represented on an abstract ground plane; each of the objects' high level representation identifies their class label (person, car, etc.).

1.1.1 Context

The motivation behind the development of this method lies on the current needs of Paaspop, client of Info Support B.V, the company supporting this project along with Utrecht University. The organization behind Paaspop, a Netherlands-based music festival, makes use of a large-scale CCTV system to ensure the security of the visitors during the length of the event. Due to the current legal constraints regarding privacy matters[1], they are not allowed to store the surveillance videos after the festival has ended. Ideally, however, they would like to store the recordings of the surrounding area outside the festival, to be able to analyze traffic and pedestrians patterns in the long-term. This could help them, for example, to improve the coordination of logistics and security bodies, particularly for tasks such as crowd control or the oversight of the festival's entrance and exit points.

1.1.2 Requirements

The implementation of the proposed method must fulfill the following terms:

1. It is capable of detecting, at every frame, the 2D location of the objects of relevance (people and vehicles) within the image
2. It provides an estimation of the location of all the detected objects within the 3D space of the scene, at every frame
3. The generated reconstruction provides anonymized representations of the objects of relevance (which, in their original form, could potentially display visual details revealing personal information)

[1]The regulations that apply in this case are defined in the recently approved European General Data Protection Regulation (European Parliament and Council of the EU, 2018)

Although our main focus is on two specific object classes (people and vehicles), our aim is to build a general approach that can be extended to other classes. For instance, the system will ideally be capable of semantically differentiating the detected vehicles (minimally among motorized and non-motorized vehicles, such as cars and bicycles, respectively; and ideally among different sub-classes of motorized vehicles such as cars, motorcycles, buses or trucks).

Regarding the second requirement, the input video files are expected to be monocular and non-redundant, that is, we have a single view of the scene and no explicit depth information is provided. Moreover, to achieve an end-to-end system that is transparent to the input video, we assume that the video was recorded by an unknown, uncalibrated camera. This way, we would not only tackle the problem posed by Paaspop but we would expand the application range of the system. Nonetheless, it is important to note that the estimation of both the camera parameters and depth from a single-view image is inherently an ill-posed problem and the expected results must be in accordance with this.

Whilst the majority of the privacy protection techniques in research aim at preserving the precise shape and structure of the captured people -in order to retain the intelligibility of their performed activities-, in this case it is not a strict requirement to do so. Paaspop's intended task does not require the preservation of any personal information, as long as the relevant elements -that is, people and vehicles- can be recognized as such.

Therefore, our goal is to be able to preserve the motion patterns of the objects, for which we also assume very few variations in their pose (people will mainly walk and stand over a presumed flat and horizontal ground plane, all objects are considered to be rigid and located at the ground level).

1.1.3 Limitations and challenges

Due to its characteristics, the implemented method is expected to have certain weaknesses:

- Current (even state-of-the-art) computer vision techniques, for both object detection and depth estimation among other applications, do not generally perform well for very crowded scenes (with abundant and significant occlusions, small sizes of certain objects like people in the distance, etc.). This can cause missed or wrong detections, or erroneous locations on the ground plane, and thus loss of information.
- The method is conceptually based on the subsequent combination of several sub-methods or modules that -as seen in the previous point- are not completely accurate; this will mean that the errors from one of them will be accumulated and propagated to the next one.
- As mentioned before, 3D position estimation based on a single, monocular image is an underspecified problem, even more if the image was taken with an uncalibrated camera; this requires additional assumptions that might not be entirely accurate or applicable for every possible scenario.

All of these aspects must be taken into account when considering which kind of input video this method could be used for if we want to guarantee a satisfactory performance.

1.1.4 Contributions

Our contributions can be summarized as follows:

- We introduce a novel method for privacy protection, based on the 3D reconstruction of the scene to:
  - Allow the preservation of the positioning and movement of the relevant anonymized elements
  - Ensure that, regardless of the accuracy of the object detection, the identity and personal information of the relevant elements is never unveiled, thanks to the very nature of the method
- Instead of focusing on the removal of specific personal identifiers, our method aims at representing the relevant elements with only their 3D location, which allows:
  - Full anonymization, as only the objects' class labels (person, type of vehicle) can be discerned
  - Easy integration into any (offline built) 3D recreation of the environment, where the objects can be represented with unidentifiable 3D models at their estimated 3D positions
  - Computational analysis of the scene dynamics (such as walking and driving patterns), as the method can provide numerical input
  - The extraction of data that could be used for crowd analysis systems

1.2 Research questions

Based on the given description of the problem and the proposed solution, we pose the following main research question:

> *How can a 3D reconstruction of a video scene, which recreates the location and movement of the present objects (people and vehicles), ensure that no personal data is provided while preserving the intelligibility of the scene (in terms of traffic and motion patterns)?*

In order to actually obtain the 3D representation that we aim for, we need to first detect, recognize and segment all the relevant objects: those that can potentially reflect any kind of personal data, which can eventually be classified among very few classes (just people and certain vehicles). The current state-of-the-art object detection methods tend to be multi-purpose, and usually focus on a broad range of predefined classes and possible scenarios. Furthermore, as mentioned in Section 1.1.3, computer vision techniques do not guarantee an adequate performance in very crowded scenarios or with large occlusions. This will affect the overall accuracy and thus the reliability of the reconstructed scene. Taking this into account, we pose the following sub-question:

> *SQ1: Can an already existing object detection method be adjusted for a better performance when detecting large and varying numbers of people and vehicles?*

The performance of the implemented object detection method will be evaluated using a well-known dataset for object detection and segmentation benchmarking,

composed of images displaying street scenes similar to our expected ones, containing our object classes of interest. We will use the mean average precision (mAP) as the main performance metric, because of its extended usage in research for evaluating current state-of-the-art object detectors. We will compare its performance to two other state-of-the-art methods that reflect the main trends in current object detection, tested on this same dataset.

After the objects have been detected, the next challenge is to correctly estimate their 3D location, which has to be done based on merely a single 2D image -an ill-posed problem by default. Deep learning-based methods have been proven to give satisfactory results for monocular depth estimation, by learning patterns from RGBD (Red-Green-Blue-Depth) image and video datasets or alternatively pairs of stereo images. However, it is important to note that the available data required for training this kind of networks is relatively limited. This introduces the additional sub-question:

> *SQ2: When used for training a monocular depth estimation network, which available dataset (or combination of them) leads to the best performance in predicting the distance of objects (people and vehicles) to the camera?*

To answer this question, we will compare the performance of several pre-trained versions of our chosen method. We will analyze them by computing the error metrics typically used in depth estimation research, such as the root-mean-square error and its logarithmic version. To see how well each model generalizes to unseen data that is still close to our expected scenario, we will make use of a subset of a well-known stereo dataset. This dataset is similar to the aforementioned one for object detection, in terms of scene viewpoint and content, but it additionally provides ground truth depth maps already computed from the known stereo setting.

The final 3D scene reconstruction will then depend on the performance of three of the main modules in the pipeline: the object detection, the camera self-calibration and the depth estimation, whose performance can be evaluated individually, by comparing their separate outputs to the ground truth. Nevertheless, the most relevant output is the actual expected output of the whole system: the set of estimated 3D locations of the objects, as this is the data that can ensure the preservation of information and thus the utility of the 3D recreation. Particularly, it is important that the 3D representation is a sufficiently faithful recreation of the original scene; this does not always require a highly precise estimation of the position of the objects scale-wise (as long as the general distribution of the objects within the scene is preserved), or the exact head count within a big crowd. To reflect this, we pose the following sub-question:

> *SQ3: How does the combined accuracy of the object detector, the camera self-calibrator and the depth estimator affect the reliability of the scene recreation (in terms of similarity to the ground truth 3D locations)?*

To assess this, we need to compare the estimated 3D locations to the ground truth ones. Although there is a patent lack of datasets that are suitable for this specific task, some benchmark datasets for multi-object tracking do contain the objects' ground plane locations estimated from a multi-view setting, which we can use as our ground truth for evaluation. We will make use of several metrics from a popular multi-object tracking benchmark, adapted for the comparison between the true and predicted ground plane locations.

Finally, we intend to make use of multi-object tracking to gather additional information about the objects with respect to the time domain. We hypothesize that this data would help improve the performance of the pipeline, both in terms of accuracy and robustness, as it can enable the smoothing of the estimated ground plane trajectories (that is, the sequences of 3D locations) and potentially filter out false positives from the object detection output. We therefore pose the final sub-question:

> *SQ4: Does the inclusion of temporal data (obtained by means of object tracking) improve the overall performance of the system?*

We will analyze the influence of tracking by comparing the previously described performance evaluation results between both settings (with and without using multi-object tracking), while maintaining the rest of the evaluation conditions.

1.3 Thesis outline

The remainder of the thesis report will be structured as follows: Chapter 2 presents a literature review of the research related to our problem, by covering what kind of information we need to process or remove from a video to make it anonymous, how we can locate this information within an image, how we can process this information to achieve anonymity, and how we can generate a 3D reconstruction of a 2D scene. Chapter 3 provides a high-level overview of our proposed method, and an individual description of the functioning of each of the modules within the full pipeline. Chapter 4 delves into the evaluation methodology, by first explaining the implementation of the method, including which data is used for training and testing (for both the separate modules and the complete pipeline), how it is annotated and how the training process is performed for each of the modules. This is followed by the definition of the objective metrics to be used for the quantitative analysis. In Chapter 5, the previously described evaluation methodology is applied and the results of the quantitative analysis are presented, which are then discussed in relation to our research questions. Finally, in Chapter 6, we conclude the report by summarizing the main findings and providing some insight into possible future work.

Chapter 2

Literature review

In this chapter, we provide an overview of the most relevant research and the existing methods related to our problem. In Section 2.1, we present a taxonomy of the visual personal identifiers that we can expect to find in the input videos, and that we must consider when assessing the anonymity achieved by our method. Section 2.2 covers the techniques for object detection, which we require for locating and segmenting the relevant objects that must later be anonymized. Section 2.3 introduces the already existing methods for privacy protection in videos, and in Section 2.4 we discuss the concept of 3D reconstruction as a way of protecting privacy, focusing on camera self-calibration and depth estimation methods from single 2D images.

2.1 Visual personal identifiers

For the anonymization process to be successful, we need to ensure that the final 3D scene reconstruction does not contain any personal data. If none of the individuals present in the scene is identifiable anymore by any means, and the anonymization process is irreversible (including the deletion of the original identifiable data, that is, the original video files in this case), the data is no longer considered personal data and the European data protection regulations do not apply anymore (European Parliament and Council of the EU, 2014). This will therefore legally allow the indefinite storage of the resulting 3D reconstruction, or any captures of it, which is the motivation behind this research.

To be able to do so, we need to have a clear image of what constitutes personal data, and in which form or representation we could potentially find it within a video. From a legal standpoint, regarding the legislation that concerns our application, personal data refers to "any information relating to a [...] person [...] who can be identified, directly or indirectly, [...] by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person", according to the definition given in the Article 4.1 of the General Data Protection Regulation (GDPR, European Parliament and Council of the EU (2018)). This means that any information that (1) is attributable to a specific individual ("can be identified"), independently of its nature ("any information"), (2) can lead to the identification of a specific individual ("can be identified") in combination with other pieces of information ("one or more factors"), and (3) can help re-identify a person even if the data itself has been de-identified, encrypted or pseudonymized ("identify [...] indirectly"), constitutes personal data and falls within the scope of the law (European Commission, 2018).

Based on both given definitions, some of the most common and well-known examples of personal data (such as an individual's full name, home address, email address or identification number) are rarely expected to appear in a video recording. On the contrary, there are several personal identifiers that an individual could display in multimedia content and which, individually or in combination with others, could help identify, trace or locate this person (see Figure 2.1). A classification of these diverse personal identifiers is given in Ribaric, Ariyaeeinia, and Pavesic (2016), where three categories are defined: biometric, soft-biometric and non-biometric.

2.1.1 Biometric identifiers

Biometric personal identifiers are biological (and sometimes behavioral), distinctive and verifiable characteristics that can be permanently and (generally) uniquely associated to an individual (Jain and Uludag, 2003; Ribaric, Ariyaeeinia, and Pavesic, 2016).

Biometric data can be easily collected, shared and copied without consent or even actual knowledge of the individual, which is the reason why this kind of data tends to be the main focus of privacy protection measures.

Biometric identifiers can be likewise classified among two types:

- Physiological (face, fingerprints, eye iris, ears)
- Behavioral (voice, gait, gestures, posture)

We can assume that phyisiological biometrics like the iris and the ears are already anonymized if the whole face is, and that fingerprints could not be extracted from regular video recordings due to their nature, small size and the expected image quality of the camera.

Regarding behavioral biometrics, we presuppose the absence of audio in the videos, or the expected destruction of it due to their uselessness for the purpose of the video recordings, hence voice can be discarded from consideration as well. Gait and gestures, however, are complex biometrics that are not trivial to anonymize. In BenAbdelkader, Cutler, and Davis (2002), gait is defined as the combination of two spatio-temporal parameters: stride (or step distance) and cadence (or walking speed), which can definitely be extracted from video recordings. The gait could help to uniquely identify a person, both manually -particularly if there is familiarity with the person (Cutting and Kozlowski, 1977)- or automatically by using computer vision techniques (BenAbdelkader, Cutler, and Davis, 2002).

2.1.2 Soft biometric identifiers

Soft-biometrics are biological, behavioral or adhered personal identifiers, not necessarily permanent, that do not uniquely identify an individual but could do so in combination with other personal identifiers (Jain, Dass, and Nandakumar, 2004). Some examples are the body silhouette, height, weight, gender, age (both specific and non-specific, as in an age range), race or ethnicity, and indelible marks like scars, birthmarks or tattoos.

We consider that, if the skin itself is anonymized, indelible marks like the ones mentioned above could not be extracted, nor racial or ethnic data could be revealed (considering the face has also been anonymized).

2.1.3 Non-biometric identifiers

Non-biometric identifiers are all other personal identifiers which do not belong to the physique or the behavior of an individual, and have a non-permanent character. These mainly include clothes, hairstyle and license plates. Clothes could show not only a clothing style or appearance that is very particular of a specific individual, uniquely identifying them (Gallagher and Chen, 2008), but also reflect the economic status of a person or their job position (in the case of work uniforms).

FIGURE 2.1: Example images from *Google Street View*, where a blurring filter is used to process personal identifiers. Only faces and license plates are processed, while soft biometrics (e.g. height, gender or silhouette of the people) and non-biometric identifiers (e.g. clothing, or car models) remain visible.

A license plate constitutes a non-biometric identifier because it legally binds the ownership of a vehicle to a certain individual's name, hence identifying them. However, vehicles have other additional characteristics that sometimes could also allow the identification of its owner or driver, such as their model (recognizable by the structural appearance of the vehicle) and color, particularly if the combination of both is uncommon.

2.2 Detection of regions of interest

To be able to apply privacy protection techniques to a video, first the regions of interest (ROIs) must be detected; that is, in this case, the areas of an image that can potentially contain personal data. Although some privacy-preserving video processing techniques are sometimes applied to the whole image or frame, the most common approach is to process only the sensitive areas that could potentially display personal identifiers.

In the computer vision field, the process of detecting ROIs is generally called object detection: the precise estimation of the presence and location of objects within an image (Zhao et al., 2019).

Often, object detection implies not only finding if there are objects present in an image -and where-, but also which kind of objects they are. This process is called object recognition, that is, the identification or labelling of a specific object within an image.

Typically, object detection algorithms delimit the area that contains an object in a rectangle shape, or what is called a bounding box. However, a more fine-grained

outline is often required, indicating which parts of the image (generally at a pixel-level) actually correspond to a specific object. This process is called object segmentation.

Depending on the application -particularly in videos where temporal patterns are also of interest-, object detection is sometimes combined with an additional procedure called object tracking: the finding of, given an already detected object, the correspondence of this object in subsequent video frames while building the trajectory of this object (Porikli and Yilmaz, 2012).

Image segmentation, or the combination of object detection and object segmentation, is the process of subdividing an image into its constituent parts or objects that must be identified and analyzed (Zaitoun and Aqel, 2015). Similarly, the sum of object detection, segmentation and recognition is generally called semantic segmentation: the assignment of predefined object or scene labels to each pixel within an image (Yang et al., 2018). Moreover, when each distinct object that is detected within an image is individually delineated, the process is called instance segmentation (Romera-Paredes and Torr, 2016).

It is worth mentioning that the process of object tracking does not necessarily require prior object recognition, as its purpose is to trace the same object within an image sequence regardless of its semantic label. Likewise, depending on the application, object detection does not always connote or require object recognition. Nonetheless, for the purpose of this project, object recognition is of high interest, for instance because knowing the class label of the foreground objects allows us to make assumptions about their three-dimensional properties (regarding depth, shape, height, positioning, among others) for the later 3D reconstruction process. Moreover, among all (foreground) objects that could be detected, we are in principle only interested in those which might portray characteristics that are considered personal data and that must therefore be anonymized (that is, people and vehicles). This means that all these detected objects must hence be subject to object recognition in order to identify their class labels.

Before analyzing the different existing techniques for ROI detection, it is important to make a general division of two categories between which these techniques can be classified. There is a key element that radically changed the general approach followed for object detection: the development of Convolutional Neural Networks (CNNs), introduced by Krizhevsky, Sutskever, and Hinton (2012). Before their appearance, object detection was typically based on either detecting image changes between frames, or explicitly defining and extracting image features to be learnt and recognized by a classifier. After their appearance, object detection techniques drastically shifted to a more joint pipeline based on a CNN (or several of them) where features are implicitly defined and learned.

In Section 2.2.1 we will analyze the traditional pre-CNN approaches, while in Section 2.2.2 we will cover the current and state-of-the-art CNN-based techniques.

2.2.1 Classic object detection

Traditional object detection techniques can generally be divided into two different types: change detectors and classifiers.

Change detectors were the first kind of object detection techniques to be developed, and they are inspired by the idea that any significant variation or motion

within a scene, relative to the original static scene, reflects the presence of an object.

Some of the most well-known change detectors are based on three common approaches: frame differencing, background subtraction and motion segmentation.

Frame differencing basically consists in measuring the intensity dissimilarity between two frames by subtracting the current frame from the previous -or a reference- one. A renowned example of this kind of method is proposed in Bobick and Davis (2001). The main drawbacks of this approach are the fact that objects become undetectable if they remain stationary for a sufficiently long time, and its high sensitivity to illumination changes.

Background subtraction has been one of the most used techniques to detect ROIs before the popularization of classifiers (Yilmaz, Javed, and Shah, 2006). They consist in the building of a representation of the scene -or a background model- and the observation of any deviation from this model for each subsequent frame (Porikli and Yilmaz, 2012). Some relevant examples are based on the usage of multimodal statistical models -such as Gaussian Mixture Models (GMMs) (Stauffer and Grimson, 1999) or Hidden Markov Models (HMMs) (Stenger et al., 2001)- to describe the pixel color or intensity and subsequently match pixels between frames (see Figure 2.2).

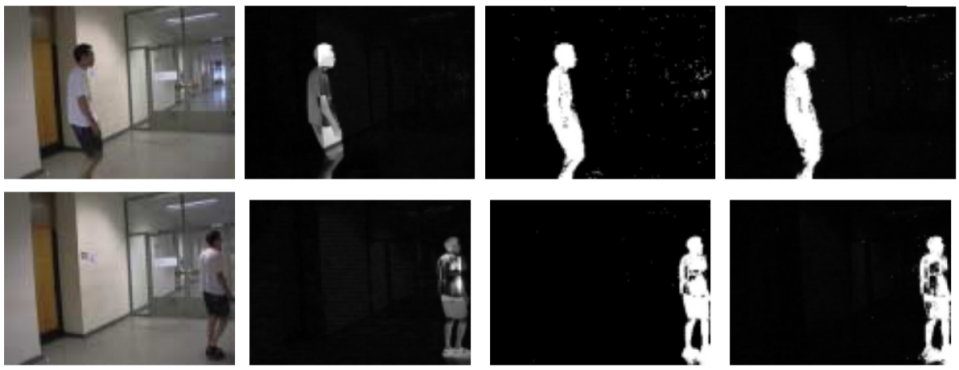

FIGURE 2.2: Detection maps obtained through the background subtraction method proposed in Stenger et al. (2001).

Motion segmentation aims at assigning groups of pixels to several classes based on the speed and direction of their motion. It is often based on the computation of the optical flow, that is, the distribution of apparent motion -or variation and movement of brightness- patterns within an image (Horn and Schunck, 1981).

The most clear shortcoming of these methods is that they tend not to be robust enough to handle background changes (such as camera noise, slow lighting changes, repetitive motion or background objects) or irrelevant motion like wind effects, repetitive movements or sudden lighting changes (Baaziz et al., 2007). This led to the popularity of classifiers: traditional learning techniques specifically applied to the problem of object detection.

This kind of technique aims at learning a classification function that captures the variation in object appearance from a set of labeled training samples. The classifiers that have been traditionally used for object detection take an image region as input -or usually certain features extracted from this region- and perform a binary classification -as in, object/no object-. Some examples of features that are often extracted are Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) or Histogram of Oriented

Gradients (HOG) (Dalal and Triggs, 2005). Some of the most commonly used classifiers are Support Vector Machines (SVMs), which cluster data into two classes by finding the maximum marginal hyperplane that separates them; or boosting, an iterative technique for finding an accurate overall classifier by combining many base classifiers (Freund and Schapire, 1997).

2.2.2 Object detection based on deep learning

The usage of classifiers for object detection eventually led to the application of deep learning, which allowed the extraction of more complex features (without having to manually define them beforehand) and therefore the adaptation to diverse situations and class labels.

Based on the conceptual approach of the most prominent CNN-based object detectors, we can define the following taxonomy:

- Sliding window-based detectors: a rectangular or squared shape is moved across an image to select the area (among all these evenly spaced windows) to use as input for a CNN at every moment
- Region proposal-based detectors: several regions, or so-called boxes, of different sizes and aspect ratios that potentially contain an object, are proposed within an image to later be classified by a CNN
- Single shot detectors: instead of having to run the detection and classification tasks several times (for either the windows or the proposed regions), both region division and classification phases are combined into a single network, and these tasks are hence performed with a single pass through the CNN

Sliding window-based approaches, such as OverFeat (Sermanet et al., 2013), are not particularly popular anymore due to their nature: as they only see local information, they cannot reason about the global context of the image and it is harder to obtain coherent predictions without additional processing.

Generally speaking, region proposal methods tend to be slightly more accurate than the other two, especially if real-time speed is not a requirement, as these techniques come with a large computational cost. The best example of a method based on region proposal is Region-based CNN (R-CNN) (Girshick et al., 2014) and its derivatives, and other relevant methods are Deep Multibox (Erhan et al., 2014) and Spatial Pyramid Pooling net (SPPnet) (He, Zhang, et al., 2015).

Single shot detectors, on the contrary, are meant to be used for applications that require real-time processing -as it can be deduced from their description- but at the expense of accuracy. Some of the most well-known examples are You Only Look Once (YOLO) (Redmon, Divvala, et al., 2016) and its newer versions; Single Shot MultiBox Detector (SSD), (Liu, Anguelov, et al., 2016) and Feature Pyramid Networks (FPNs) applied for object detection (Lin, Dollár, et al., 2017) such as in the RetinaNet method (Lin, Goyal, et al., 2017).

In the following subsections, we will briefly describe the aforementioned methods, and how each of them built upon the previously proposed methods with the purpose of solving their main weaknesses.

Region proposal methods

It was not long after CNNs were proposed that researchers started wondering how this new architecture could be applied for object detection.

One of the first methods to be introduced was **R-CNN** (Girshick et al., 2014): the combination of a region proposal method, a CNN for feature extraction, and an SVM for classification, with the purpose of detecting the location and class label of the objects within an image (see Figure 2.3).

The region proposal method, called Selective Search (Uijlings et al., 2013), consists in looking at the input image through several windows of various sizes in a bottom-up fashion, trying to group together neighbor pixels by a certain feature (such as texture, color or intensity) to identify which of these groups or regions could potentially contain an object. Initially considering each pixel as an independent group, the groups that are the closest feature-wise are then combined to form new larger groups, until all the possible ROIs are merged.

These regions -or boxes- are warped to square size and passed through the CNN. Its output is given to the SVM which classifies if the content of a proposal is in fact an object, and which object -or class label-. The resulting region proposals that do contain an object are then processed through a linear regression function to produce more tight-fitting bounding boxes, outputting the new final bounding box coordinates for the detected objects.

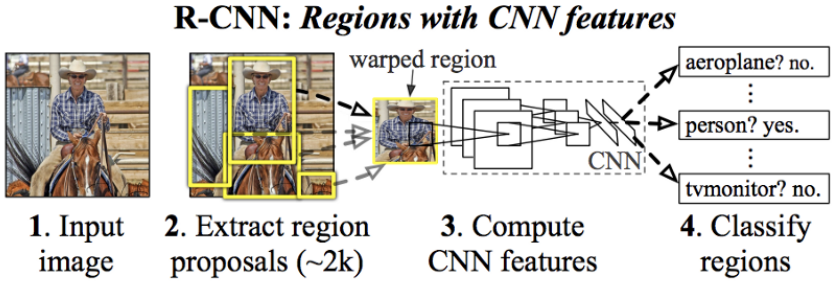

FIGURE 2.3: High-level diagram of the R-CNN object detection method. R-CNN first takes an image as input, computes several region proposals, and extracts the features for each region with a CNN, to then determine the presence (and corresponding label) of an object within the proposed region.

One of the main shortcomings of R-CNN is the fact that it requires a forward pass through the CNN for every region proposal for every image, even if some regions overlap. This, plus the fact that its region proposal method generates a very high number of boxes that then need to be passed to the network, dramatically increases the computational time of the process.

Deep MultiBox (Erhan et al., 2014) presented a similar pipeline for scalable object detection (although without the object classification component) with the aim of improving the efficiency of the region proposal phase.

Its region proposal method introduced the concept of prior boxes: bounding boxes of different fixed sizes that aim at approximating the distribution of ground truth boxes. The prior boxes are selected only if they sufficiently overlap with the ground truth ones. Ultimately, only a certain number of boxes are predicted (depending on their confidence scores or on how far from the ground truth boxes they

are) using those prior boxes as starting points.

Trying to improve R-CNN by reducing its computational time, He, Zhang, et al. (2015) introduced **SPPnet**, an enhanced version of R-CNN thanks to two main add-ons: a Spatial Pyramid Pooling (SPP) layer with adaptive size pooling, and the single-time computation of the feature maps.

Initially, in the same fashion as R-CNN, Selective Search is used for obtaining the region proposals. The main difference comes with the subsequent extraction of global features using a pre-trained network, and for each region proposal, the SPP layer matches the corresponding part of the feature map set to a pyramidal set of features (mostly by using max pooling). These features are then used for the object classification and the box regression, instead of inputting the region from the original image like in R-CNN, and having to repeatedly compute the features.

With the purpose of overcoming the known drawbacks of R-CNN, and following the approaches from Deep MultiBox and SPPnet, the first author of the original R-CNN proposed a new version, **Fast R-CNN** (Girshick, 2015).

Similarly to SPPnet, Fast R-CNN extracts the features of the image once, and the input features are selected correspondingly to each region proposal. For this process, they propose a method called RoIPooling (Region of Interest Pooling), which mirrors the SPP layer in SPPnet but using a single layer of features only, instead of a pyramid (see Figure 2.4).

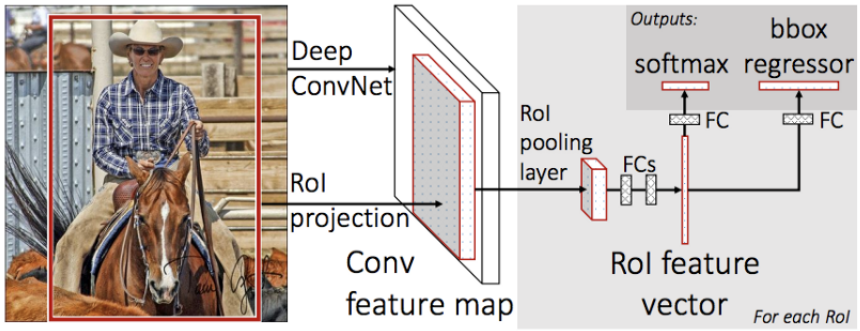

FIGURE 2.4: High-level diagram of the Fast R-CNN object detection method. The region proposals are pooled into a feature map of fixed size and mapped to their corresponding feature vector, through the process called RoIPooling.

An additional drawback of R-CNN that was not tackled in Deep MultiBox and SPPnet is the fact that three separate models must be trained independently: the feature extractor, the classifier and the regression model, making the training of the whole pipeline an arduous task. To solve this, Fast R-CNN jointly trains the three models in a single network: a softmax layer is used to replace the SVM classifier in order to output the class labels of each region, and a linear regression layer is added in parallel to the softmax layer for outputting the final box coordinates. Therefore, Fast R-CNN is fundamentally a variation of SPPnet with only one pyramid level of the feature pooling and with a complete trainable architecture (SPPnet uses a pre-trained feature extractor, hence its parameters cannot then be tuned). In machine learning, a complete trainable architecture is also generally called an end-to-end architecture, where no manually defined algorithms or features are required and the solution to the posed problem is directly learnt from the annotated dataset.

Even after adding these modifications, the authors of Fast R-CNN noticed that there was still a bottleneck within the architecture: the region proposal process significantly decreased the overall speed of the system.

To overcome this, they removed the Selective Search method, and instead of running a separate search algorithm, the same computed feature map set is used for both extracting the features and obtaining the region proposals. With these two processes unified, the new proposed architecture, **Faster R-CNN** (Ren et al., 2017), takes as input the actual full image -instead of the pre-computed region proposals- and outputs the object label and bounding box coordinates.

To achieve this, the proposed network is formed by, first, the convolutional layers that output the feature maps, and second, a Region Proposal Network (RPN) that takes these feature maps and outputs the region proposals. Both the region proposals and the feature maps are then given as input to the classification layer.

The RPN works as follows: it slides a window over the feature maps and, at each area, it outputs k potential bounding boxes along with their confidence scores, with k being the number of different aspect ratios that are commonly found and considered for bounding boxes surrounding the expected objects (named anchor boxes). For each anchor box, the RPN outputs one bounding box and its confidence score per position within the image, and those with a sufficiently high confidence score are then passed on to the classifier.

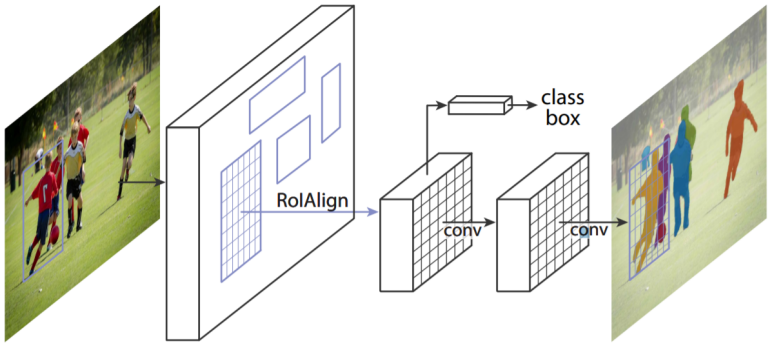

FIGURE 2.5: High-level diagram of the Mask R-CNN architecture. The proposed object regions on the target feature map are aligned to the ones from the input feature map. These regions are then, on one hand, classified and given an object label; and on the other hand, convoluted to find the segmentation mask of the corresponding object.

While all of these methods perform object detection and classification, none of them does object segmentation. **Mask R-CNN** (He, Gkioxari, et al., 2017) extended Faster R-CNN by adding pixel-level classification and segmentation (see Figure 2.5), which is achieved by extending the architecture with an additional branch: a Fully Convolutional Network (FCN) that takes the feature maps as input and outputs a binary map for each object (where the 1-digits represent the pixels belonging to the object and the zeros otherwise).

A problem that arises with this addition is the misalignment of the regions selected by RoIPool relative to the same regions within the original image, which leads to inaccuracies during the segmentation -as pixel-level granularity is required. To

solve this, RoIPool is replaced by an alternative version, RoIAlign. RoIAlign substitutes max pooling with bilinear interpolation to compute the exact values of the input features at each region. In this way, the binary masks (and hence the segmentations) can be correctly computed.

Single shot methods

Although the methods described above offer state-of-the-art accuracy, their relatively slow computational speed turns them unsuitable for real-time applications.

Focusing on achieving a faster performance, the **YOLO** object detector (Redmon, Divvala, et al., 2016) was proposed. Instead of using a sliding window or predicting region proposals (approaches that are both fundamentally based on the re-purposing of a traditional classifier for object detection), YOLO proposes a single neural network in an end-to-end manner: it runs a single pass through the network -that is, the image is 'looked at' only once-, taking the whole image as the input and giving the bounding box parameters as the output.

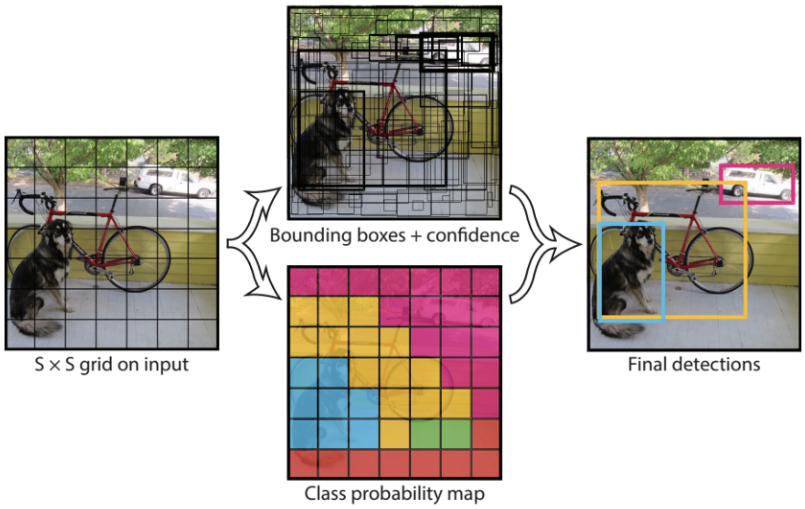

FIGURE 2.6: Overview of the YOLO object detection method.

To do so, the input image is initially divided into a grid, where each cell predicts a certain number of bounding boxes, represented by their relative coordinates and dimensions, plus their confidence score and the class probabilities (see Figure 2.6). The boxes without a high enough confidence score are discarded; likewise, among boxes with a significant overlap, only the one with the highest confidence score is kept (a process that is called non-maximum suppression). This is done under the assumption that each grid cell is 'responsible' for predicting a single object (the one whose center falls inside that cell).

In a similar manner to YOLO, **SSD** (Liu, Anguelov, et al., 2016) aimed at introducing an end-to-end architecture capable of predicting all bounding boxes at once, but following an approach closer to that of Faster R-CNN -by using an RPN to obtain both the confidence score and the class probabilities per box.

Its CNN contains multiple convolutional layers all with differently sized filters (also called multi-scale layers), with the purpose of obtaining feature maps for every region of different size that could potentially be a correct bounding box, improving

the diversity of the proposed boxes. The last layers of the network -called extra feature layers- all use small sized filters meant to output a high number of bounding boxes. The parameters to describe these boxes are similar to those in YOLO with the exception of the replacement of the box coordinates with those of the center point of the box, hence reducing the total number of parameters.

An improved version of YOLO was presented in **YOLOv2** (Redmon and Farhadi, 2016). YOLO9000, as its implementation is named, was trained for more than 9000 categories. YOLOv2 introduced several new characteristics:

Firstly, higher resolution input images are allowed, plus the network architecture is adapted to also include high and low resolution feature maps. This is done to enable the detection of smaller objects, which was one of the weaknesses of YOLO. Secondly, the final fully-connected layer for predicting the boxes coordinates is replaced by the usage of anchor boxes, similarly to Faster R-CNN. Thirdly, regarding the network architecture, batch normalization is used in order to prevent overfitting without requiring the usage of dropout.

Another relevant advancement for research in object detection was the introduction of FPNs. Although an FPN is not an object detector *per se*, it has been used as a feature extractor for several object detection architectures. The idea behind it is that when seeing an image at different scales -in a pyramid-based fashion-, it is easier to detect differently sized objects, especially very small ones.

The flow of data through an FPN goes firstly in a bottom-up pathway: the image goes through the convolutional layers and the spatial resolution decreases, meaning that more abstract and high-level structures can be detected on the image. Secondly, a top-down pathway is followed so that the object locations, now imprecise due to the downsampling processing, can be more accurately predicted afterwards.

One of the main advantages of region proposal-based methods over single shot ones, nonetheless, is that they predict a low number of possible object locations, so the sum of their prediction losses are in principle balanced among classes. On the contrary, single shot detectors such as YOLO or SSD predict a very high number of possible object locations and have therefore an excessive class imbalance, for instance because of classes like "background" having a drastically high incidence in comparison to actual object classes. This imbalance means that the loss values of the most common classes will dominate the gradients and the backpropagation process, while for instance the detected boxes containing objects that might be hard to recognize by the network -that is, with low confidence scores- and are hence relevant for the learning process, would ultimately have an insignificant influence on the backpropagation process.

RetinaNet, a single-shot technique based on an FPN, tries to overcome this issue by scaling the cross-entropy loss so that the most certain cases, for which the network is confident about the predictions, contribute less to its value. This new loss function is called focal loss, and it basically 'sinks' the hill of the original loss function so that for higher confidence scores the loss values are much lower. RetinaNet's architecture is purposely kept simple (see Figure 2.7) to shift the focus of the method to its novel loss function.

In order to try to get the accuracy of single shot detectors closer to that of region proposal-based methods, the authors of YOLOv2 presented a new version of their technique, **YOLOv3** (Redmon and Farhadi, 2018), with several improvements which, although slightly sacrifice speed, do not compromise its real-time application

-guaranteeing 30 frames per second (fps) against YOLOv2's 45fps.

Most of these improvements concern the architecture of the CNN itself, such as the implementation of residual blocks, skip connections and upsampling. Moreover, it stacks a significantly higher number of convolutional layers, 106 instead of the 30 layers in YOLOv2.

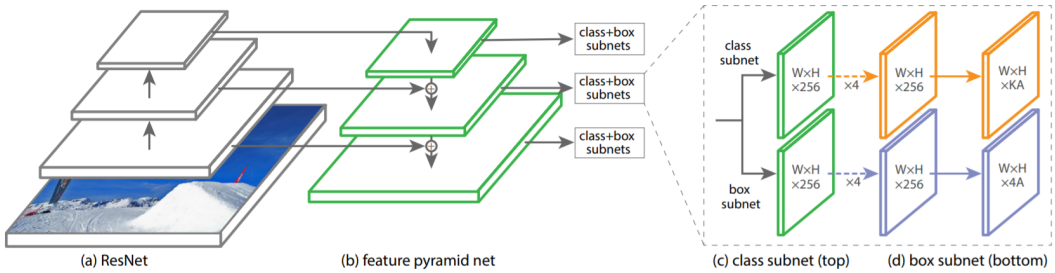

FIGURE 2.7: High-level architecture of the RetinaNet object detection method. The FPN, in charge of producing multi-scale feature maps, is followed by two networks for classifying the boxes and for regressing the boxes' dimensions, respectively.

Another notable add-on is the implementation of multi-label classification: instead of applying a softmax function to the output class probabilities -thus classifying objects with an only label, the one with the higher probability-, certain classes are considered not to be mutually exclusive and objects can therefore be classified with more than one label. To do so, the softmax function is replaced with logistic classifiers to compute the likeliness of the object belonging to a certain class.

2.2.3 Object tracking

Although object tracking is not a requirement whatsoever for object detection, it can often be useful for refining this process: knowing the previous positions of the objects in the scene can help us determine the correctness of the newly detected objects. Furthermore, it could considerably help during the depth estimation and 3D reconstruction phases, as it can be used for smoothing the estimated positions and trajectories of the relevant objects.

There are several challenges that have to be considered when performing visual object tracking, such as: occlusions (between an object and the scene or between objects), illumination and appearance changes, motion and deformation of non-rigid objects, abrupt motion, image noise, background similarity, etc. (Yilmaz, Javed, and Shah, 2006; Lukežic et al., 2017). Nevertheless, there are several assumptions that are typically made when performing object tracking aimed at simplifying the process and removing some of the aforementioned challenges. For instance, it is generally considered that there is no abrupt object motion and that the speed -or at least the acceleration- of the objects is constant. It is also often assumed that the number, size, appearance and shape of the objects are known and invariable (Yilmaz, Javed, and Shah, 2006).

Visual object tracking algorithms typically follow these four steps (Li et al., 2013):

1. Object initialization, performed either automatically (using an object detector) or manually by annotation
2. Appearance modeling, which consists in two sub-steps: visual representation (object descriptor reconstruction based on visual features) and statistical modeling (mathematical model building for object identification through statistical learning)
3. Motion estimation, where future states of the object are predicted based on: the current state, state evolution, and noise of each measurement
4. Object localization by optimization

Delving into these steps, it is noteworthy that visual representations of the objects can be global or local. The former reflect the global characteristics of the object appearance, with some examples being optical flow, histograms, active contours, or the raw pixels themselves. The latter are generally based on interest points or saliency. Some examples of local representations are SIFT or Speeded Up Robust Feature (SURF), a similar but faster version; HOG, Local Binary Pattern (LBP) features for texture encoding, local templates, or corners. On another note, motion estimation is generally performed by using linear regression techniques, Kalman filtering (Beymer and Konolige, 2017) or particle filters.

Most of the recent research on object tracking is based on the concept of **tracking-by-detection**, where statistical modeling is done to precisely support detection. Posing the problem like this, the goal of tracking is therefore to associate the detected objects across the video frames, which is particularly useful in multi-object tracking problems.

Although they are based on the same concept, these kinds of tracking methods can differ in terms of certain parameters, such as the type of target (single-object or multi-object) or the context-awareness, among others. Regarding the algorithmic approach itself, tracking-by-detection techniques are sometimes classified between online (only previous frames are used) and offline (both past and future frames are considered).

Some examples of **online** methods make use of sparse representations (Fagot-Bouquet et al., 2015), tracklets (Bae and Yoon, 2014) or motion (Yoon et al., 2015). Although online techniques are suitable for real-time applications, they can be more prone to errors caused by the lack of future temporal information, such as identity switches and fragmented trajectories.

Offline techniques, on the other hand, can correct these association errors as they have more information available, generally providing a more robust performance than online methods. However, this sometimes comes at the expense of time, which can also vary depending on the kind of offline approach, as these techniques can likewise be classified among global or local. Global offline methods perform the data association simultaneously across all the frames, or across batches of them, whereas local methods perform it across a few frames at each time. Some examples of global offline method include MotiCon (Leal-Taixé, Fenzi, et al., 2014), which makes use of motion information; CEM (Milan, Roth, and Schindler, 2014), which performs data association based on energy minimization; or SMOT (Dicle, Camps, and Sznaier,

2013), based on appearance similarity and motion for optimization among similar objects. While global approaches make use of all the available information, they can carry a significant delay during the processing, which local approaches reduce at the cost of some information loss. Some examples of local techniques are SegTrack (Milan, Leal-Taixé, et al., 2015), which uses low-level information at the pixel level; NOMT (Choi, 2015), which introduces its own visual representation descriptor based on point trajectories; or MHT_DAM (Kim et al., 2015), which revisits a classic object tracking technique by approaching it from the tracking-by-detection perspective.

Similarly to the evolution in research for object detection, deep learning and particularly CNNs have also been used for object tracking methods, as feature extractors and object detectors (Nam and Han, 2016). However, they have not been extensively used for the tracking process itself (Leal-Taixé, Milan, et al., 2017), that is, for the data association among frames, probably due to the large data requirements for training and the scarce availability of adequate datasets for the task.

2.3 Privacy-preserving techniques

Once these potentially sensitive regions have been detected among all the video frames, they can be somehow processed, with the goal of protecting the privacy within the scene.

The techniques for protecting privacy that have been proposed in the past among related research, can be classified following different taxonomies. A thorough one is proposed in Padilla-López, Chaaraoui, and Flórez-Revuelta (2015), where five categories are defined: intervention, blind vision, redaction, secure processing, and data hiding.

Intervention refers to the prevention of private visual data acquisition by directly intervening the camera devices to avoid the image capture. Therefore, this category is not of high interest to us, because we assume that the privacy-sensitive video has already been captured. Likewise, blind vision refers to the anonymization of the processing pipeline itself, by means of secure multi-party computation where no party can see the full process; thus, it is not relevant either in our case, as we are already aware of the intention of the whole procedure. On another hand, data hiding comprises the embedding of the original information inside of a modified version, for instance through steganography or watermarking techniques; although it does involve the processing of video (either pre-recorded or in real-time), it constitutes a reversible process and hence does not meet our requirements.

We will therefore focus on the two remaining categories: redaction and secure processing.

2.3.1 Image processing techniques for privacy protection

Redaction

Redaction refers to the modification or editing of sensitive regions of an image in order to conceal private information. It is the most extensively researched approach for image-based privacy protection, and it is mostly based on the prior detection of the relevant regions through computer vision algorithms, such as the ones we

described earlier in this Chapter. Depending on the way it is performed, redaction can be likewise divided among several sub-types:

Image filtering, the most popular one, consists in the removal or manipulation of the ROIs in an image by applying distorting image filters. Traditionally, the filters that have been applied are:

- **Blurring**: the application of a Gaussian function to modify pixels based on their neighbor ones, with some examples of its application to privacy protection in Frome et al. (2009) and Korshunov and Ebrahimi (2014)
- **Pixelization**: the division of an image into blocks and subsequent average coloring of them, used in Boyle, Edwards, and Greenberg (2000) and Korshunov and Ebrahimi (2014)
- **Masking**: the complete solid coloring of an image

Although masking would intuitively provide the highest level of privacy, it can also hamper the intelligibility of the scene more than the other filters. Pixelization seems to achieve the best privacy-intelligibility balance among these three filters (Korshunov, Araimo, et al., 2012), but it can also be less safe as the original data could be potentially recovered through the integration of pixels along trajectories over time.

Other alternative image filters include:

- **Scrambling**: pseudo-randomized sign flipping of bits during encoding, as it has been applied in Dufaux and Ebrahimi (2006) and Baaziz et al. (2007)
- **Obscuring**: quality reduction, through noise, jitter, color desaturation (Senior et al., 2005), and downsampling (Baaziz et al., 2007)

Another popular redaction method is **encryption**, or the ciphering of ROIs, generally by means of other techniques like image filters themselves, ruled by a certain known key or seed. Although it can indeed be considered a privacy protection method, it can hardly be seen as a way of anonymizing data, as it does not provide a trade-off between privacy and intelligibility but instead focuses on restricting the data analysis for authorized users. In other words, it is a reversible method and thus not adequate for true anonymization.

De-identification is another redaction technique that focuses on preventing direct identification of individuals -for instance, by removing certain sensitive identifiers but not all of them. A widely researched example is the k-same family of algorithms, for face transformation through similarity analysis and image component averaging (Newton, Sweeney, and Malin, 2005; Gross et al., 2009). Other examples are face replacement using a standard face images library (Bitouk et al., 2008) or skin tone substitution for avoiding race-based discrimination (Berger, 2000).

Redaction can also be performed by not just manipulating the appearance of a ROI but also by removing the ROI from the scene, through background reconstruction at its location. This process is called **object removal**. This is not of high interest for our specific case, as we would lose the utility of the final video if applied, but it is nonetheless relevant for the topic of anonymization itself. A valuable example is presented in Wickramasuriya et al. (2005), where they hide the employees in the

surveillance video feeds -as their actions are not needed to perform the surveillance task.

The last redaction method to be covered is the usage of **visual abstractions** (Chinomi et al., 2008), or object replacement with new high-level models. Typically, silhouettes have been used as the substitute model, generated by removing textures and preserving shapes only (Tansuriyavong and Hanaki, 2001), but recently, the idea of using avatars (either 2D or 3D) has been mentioned in research (Sadimon et al., 2010; Padilla-López, Chaaraoui, and Flórez-Revuelta, 2015), although privacy-preserving 3D avatars have not been implemented yet.

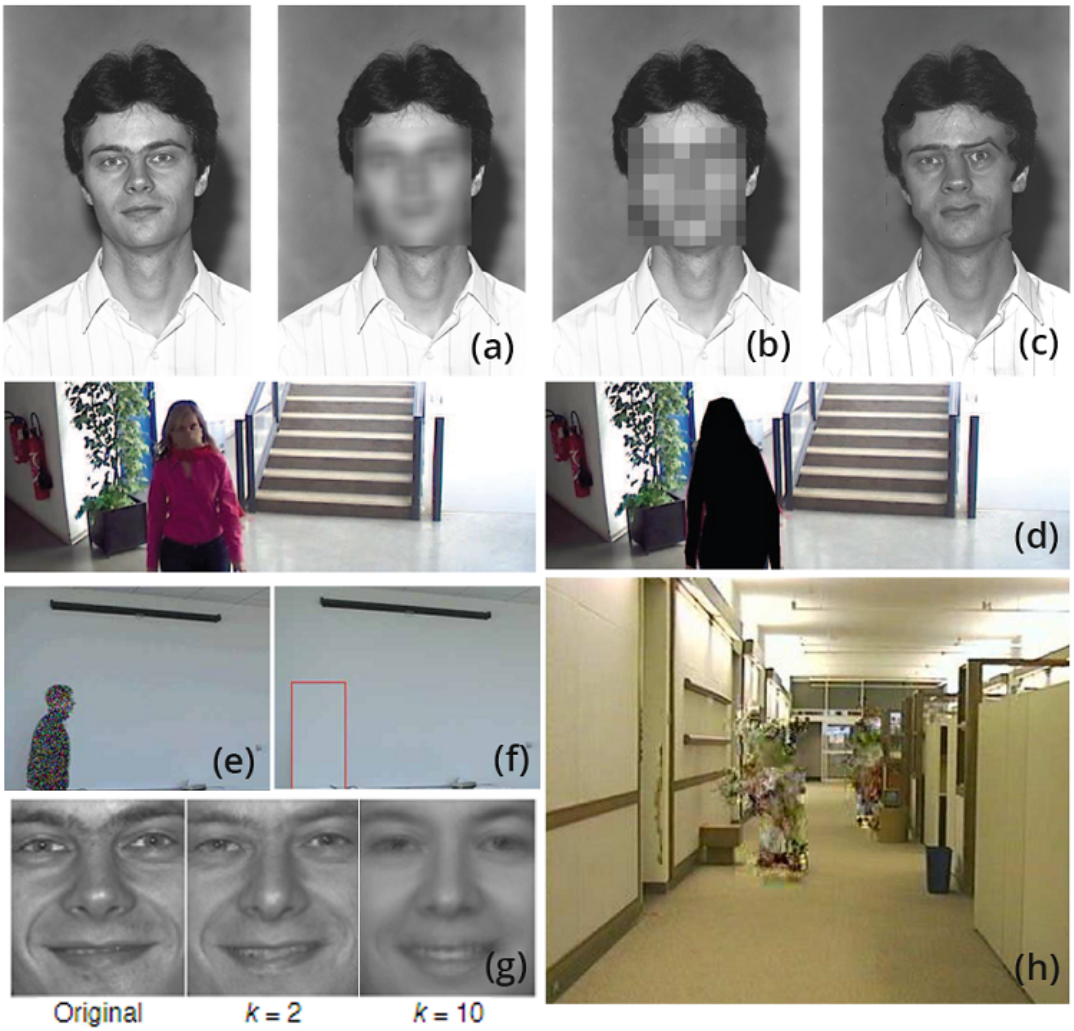

FIGURE 2.8: Examples of privacy protection methods based on redaction applied to ROIs (faces or whole human figures): (a) blurring, (b) pixelization and (c) warping (Korshunov and Ebrahimi, 2014); (d) masking (Korshunov, Araimo, et al., 2012); (e) obscuring and (f) object removal (Wickramasuriya et al., 2005); de-identification through k-same algorithms (Gross et al., 2009); and (h) scrambling (Dufaux and Ebrahimi, 2006).

Secure processing

Secure processing refers to a general privacy-respectful way of processing visual information. Particularly, this can be achieved by rejecting visual information that is not necessary for the task.

A typical way to do so is to only retain the **silhouettes** of the relevant objects (this must not be confused, however, with visual abstraction-based redaction using silhouettes: there, the original ROI is replaced with a silhouette and the rest of the visual information is kept, whereas in secure processing, we remove all the visual information except the silhouettes of the objects). This would still allow, for instance, the intelligibility of certain actions and activities performed by individuals while avoiding their identification (Park and Kautz, 2008; Chaaraoui, Climent-Pérez, and Flórez-Revuelta, 2013).

Another proposed mechanism has been the sole preservation of **depth information** (Zhang, Tian, and Capezuti, 2012). Having removed the information regarding color or structure details, among other aspects, depth visualization prevents face recognition and the direct extraction of visual clues for person identification.

In the same aforementioned paper, they additionally propose the implementation of common 2D RGB cameras for the areas that are unreachable by the RGBD cameras, and the usage of **histograms of width-height ratios** to represent the individuals. This way, the original video files can be deleted while preserving the information relevant to the task, as these histograms are sufficient to distinguish the human actions the authors aim to recognize.

FIGURE 2.9: Visualization of the virtual environment recreation proposed in Fleck and Straßer (2008) for geo-referenced people and vehicle tracking plus activity recognition. On the left, the scene is represented as a 2D ground map, where the locations of the tracked people are depicted, along with their current activity (green: walking, red: falling). On the right, the tracking results are integrated with Google Earth.

In Fleck and Straßer (2008), the authors propose the usage of **metadata** only instead of visual information, by using several different sensors aside from cameras that transmit high-level information, such as the position of an individual on the ground plane, while discarding the actual video feed. This, along with the usage of depth information, is closely related to our proposed methodology.

2.3.2 Anonymization of specific visual personal identifiers

While for many of the methods described in Section 2.3.1 we have assumed that they are applied to the whole ROI, this is not always the case. In this Section 2.3.2, we will briefly classify some image-based privacy protection methods depending on the personal identifier that they aim to protect, that is, the sensitive information that they manipulate or remove.

Regarding non-biometric identifiers, most research lies on the anonymization of license plates, usually by simply applying a blurring filter (Frome et al., 2009). There

is little work done on anonymizing hairstyle and clothing, with an exception being the hair color de-identification presented in Prinosil et al. (2015).

Among biometrics, faces are undoubtedly the most common identifier to be anonymized, either with traditional image filters, warping or morphing (Korshunov and Ebrahimi, 2014), scrambling (Dufaux and Ebrahimi, 2006), replacement (Bitouk et al., 2008) or k-same select algorithms (Gross et al., 2009), among others. Additionally, there are a few studies on gait and gesture anonymization; however, they mainly focus on manipulating the silhouette as a cue for gait, and not other parameters such as the stride or cadence of the walk. Some examples are the downsampling of the ROIs' silhouettes (Baaziz et al., 2007) or the temporal blurring of spatio-temporal boundaries of the detected object (Agrawal and Narayanan, 2011). Another way of anonymizing an individual's gait is to actually replace the whole body with a default one, as proposed in Nodari, Vanetti, and Gallo (2012). Other soft biometric identifiers such as gender, age or race have been extensively researched as an input for automatic person identification but their specific anonymization has not been dealt with yet.

2.4 3D scene reconstruction

Our privacy-preserving approach is, similarly to some of the techniques described above (Zhang, Tian, and Capezuti, 2012; Fleck and Straßer, 2008), to generate an alternative visualization of the scene where the potentially sensitive data is not kept or transferred to. In our case, our aim is to reconstruct, in the 3D space, what is happening in the 2D-captured scene.

In the computer vision field, 3D reconstruction is known as the process of determining the shape, structure and appearance of real objects and recreating them in a three-dimensional environment.

3D reconstruction approaches can be classified among active or passive, depending on the direct interaction, or lack thereof, with the object or scene to be reconstructed (Curless, 1997), by means of physical contact or through the projection of some kind of energy. Active methods do interact with the scene to obtain its depth map, for instance through light sources, lasers, radio waves, microwaves or ultrasounds. Passive methods do not interfere with the scene, and instead they use sensor-based measurements to infer the three-dimensional structures through image understanding -that is, the essential purpose of computer vision. For the latter, a sensor typically refers to a camera, leading to what is known as image-based 3D reconstruction (Remondino, 2007).

Most of the image-based 3D reconstruction approaches focus on stereopsis, or binocular vision, whose goal is to produce a dense depth map from a pair of images. Similarly, multi-view 3D reconstruction aims at reconstructing a complete scene or object from a collection of images taken from several known camera viewpoints (Seitz et al., 2006). These methods rely on geometric cues that allow triangulation for depth estimation, or on the inference and learning of these cues by CNNs. However, there are several other cues containing useful depth information that are actually monocular (Saxena, Chung, and Ng, 2008), such as texture variations, gradients, focus and depth of field, color, or haze, that could potentially enable 3D reconstruction even in a single view setting.

2.4.1 Camera self-calibration

To be able to reconstruct the 3D scene out of a 2D image, we need what is called the camera projection matrix. This matrix provides the conversion between 3D world coordinates and 2D image points, and it is based on certain camera-related values known as the intrinsic and extrinsic camera parameters. The process of extracting these parameters from camera-recorded images is called camera calibration.

Typically, camera calibration is performed by first obtaining several images of a certain object with known geometry, from which it is possible to extract a sufficient number of known 3D world points and their corresponding points on the image plane, to subsequently determine the camera parameters. Nevertheless, this is not always possible, particularly in our expected situation: input videos are directly provided without having any knowledge of the cameras, or any access to them so that calibration could be actively performed.

A solution to this is the procedure of camera self-calibration, which does not require prior knowledge of the recorded scene's inherent structure, and it is usually based on soft assumptions of the appearing objects. Although many self-calibration methods are based on the camera motion, this cannot be applied for a surveillance application such as ours, where there is a single static camera capturing the scene.

Relatively recent camera self-calibration methods can be broadly classified among two types, depending on the kind of object that is used for inferring the locations of the vanishing points (that is, the intersection points of parallel lines in the real world within the recorded image of the scene). These objects can be static structures (such as buildings) and moving objects (such as people or vehicles) (Zhang, Li, et al., 2008). As we cannot establish clear assumptions on our expected background scene, and we do know that at least pedestrians and certain vehicles will be present, we will focus on the second type.

One of the first well-known examples of camera self-calibration based on moving objects was based on the tracking of objects under the assumption of constant linear speed (Bose and Grimson, 2003). Although their method is in principle extensible to different objects, it is mainly focused on vehicles, as the imposed assumptions are harder to guarantee with pedestrians (which are precisely our main expected objects in the scene).

The first method to be based on moving people was proposed by Lv, Zhao, and Nevatia (2002), where they track pedestrians and extract their head and feet locations in several frames to derive three vanishing points (two on the horizon line and the vertical one). Several subsequent self-calibration methods were based on this approach, such as the newer version by the same authors (Lv, Zhao, and Nevatia, 2006), where they apply nonlinear optimization to the calculated parameters. In Krahnstoever and Mendonça (2006), they incorporate temporal information to reduce noise and outliers. In Liu, Collins, and Liu (2011), the authors avoid the need for accurate detection and tracking by performing camera calibration from the foreground mask, assuming that the distribution of relative human heights is previously known.

The main shortcoming of the latter methods is the reliance on the detection and tracking techniques, which can perform poorly, particularly in traditional surveillance videos due to noise, shadows or a low image quality. However, detection and tracking have undergone significant advances in the past years, which should

in principle improve this kind of camera self-calibration method. Other limitations are the dependence on statistical techniques for the removal of outliers, which is problematic in this scenario as the number of outliers is expected to be high; or assumptions on the pedestrians' height, which cannot be guaranteed in a real world setting. Moreover, all the methods assume that the focal length is the only parameter that is completely unknown and no prior approximation is provided.

The more recent method proposed by Tang, Lin, Lee, Hwang, Chuang, and Fang (2016) aims at overcoming these drawbacks, for instance by applying mean shift clustering for minimizing the influence of outliers, or reducing the required assumptions to only the approximate camera height.

2.4.2 Single image-based depth estimation

As it generally happens with image understanding tasks, we as humans are inherently good at depth estimation from a 2D visual source, and we make use of several monocular cues to do so, such as perspective, scaling (relative to the known sizes of familiar objects), object appearance depending on the lighting, shading and occlusion (Howard, 2012). Single image-based depth estimation is, however, a complex task in the computer vision field, because local image features do not unequivocally indicate depth: as an example, *sky* and *water* in a scene could look very similar feature-wise although they are drastically different in terms of geometry (Liu, Gould, and Koller, 2010).

In a similar manner to the evolution of ROI detection seen in Section 2.2, the general approach for depth estimation drastically shifted to the application of deep learning after the appearance of CNNs. Hence, we can classify the different methods between two categories: the pre-CNN (or model-based, in which we also include methods based on traditional machine learning algorithms) and the CNN-based (which we will call data-driven).

Model-based techniques rely on prior cues about the scene that are known or learnt beforehand, or on certain elements that are expected to appear in it. On the contrary, data-driven techniques focus on automatically learning cues or patterns from the given data without previously defining them.

Model-based techniques

We can consider that there are three types of model-based approaches depending on the type of cues that are used, which we have named as passive geometric cues, active geometric cues and semantic cues. The first one refers to the properties expected from certain geometrical structures (normally under specific lighting conditions), whereas the second refers to geometric or depth properties of the scene itself that can be known through camera-based characteristics (such as focus and depth of field). An example of the first one is the brightness pattern over the surface of an object: depending on the known location of the light source and the shades generated on the object's surface, we can infer its three-dimensional structure. The third type of cues refers to the known three-dimensional properties of specific objects - that is, depending on their semantic label. An example would be the assumption that elements like 'ground' or 'road' in a scene can be considered flat and horizontal, constraints that are enforced and applied in Liu, Gould, and Koller (2010). We will

focus on the first and last ones, as active geometric cues require the direct manipulation of the camera settings or the capture of several images in order to estimate depth, which we cannot ensure if we intend to make our pipeline relatively transparent to the input video data.

A notorious method that is based on passive geometric cues -more concretely, on intensity or color gradients- is **Shape from Shading** (SfS). This method aims at resolving the inherent ambiguities of this problem by using prior knowledge of the object's surface photometry and the position of the light source; in other words, it recovers the shape of an object based on the shading in the image (Zhang, Tsai, et al., 1999). Another example is the method proposed in Hoiem, Efros, and Hebert (2007), where they build **appearance-based models of geometric classes** that describe the 3D scene orientation of the different regions within an image. The cues that are used for describing these classes and subsequently allowing the labeling of the surfaces, are related to color (RGB mean, hue, saturation), texture (by using Gaussian- or Laplacian-based filters), location, and perspective (line intersections, vanishing points, gradients). Similarly, Saxena, Chung, and Ng (2008) propose the usage of a Markov Random Field (MRF) to learn features that reflect three local cues ()texture variation, texture gradients and color) at different spatial scales and distances, to be able to capture more global properties of the image.

The main drawback of these previously described methods is that, aside from requiring additional -manually defined- information in the form of image features or camera parameters, the learning algorithm has a significant burden, as it needs to implicitly reason about semantic context in order to learn depth properties. Another model-based approach would therefore be to rely on **prior scene understanding**, allowing the mapping of appearance features to depth separately for each semantic class. An example of this is the technique proposed by Liu, Gould, and Koller (2010) where, after performing semantic segmentation (by using a multi-class MRF with classes such as person, car, road or building), the 3D reconstruction is done based on the predicted semantic labels (using another MRF including prior constraints that are both pixel-based, like smoothness, and region-based, like planarity or connectivity). This way, the depth and geometry constraints can be reinforced more easily at a pixel-level, as they are class-related.

A semantic-aware approach like the latter would be conceptually adequate for our case, as we already require a previous object detection process in order to recognize the objects that must be anonymized. This way, the semantic segmentation would serve not only for the identification of the relevant foreground objects but also for establishing known depth cues and priors for the semantic classes, that would in principle simplify the depth estimation and 3D reconstruction processes. This kind of approach, however, has a notable shortcoming: as the two phases (semantic segmentation and depth estimation) are independent and sequential, if there are errors in the predicted semantic labels it would not be possible to reconstruct the corresponding objects.

Data-driven techniques

Instead of just conditioning depth on the semantic labels, it is also possible to condition the semantic label on the depth, helping the classifier to distinguish between ambiguities caused by the distributions of depths on the training dataset (for instance, the depth of an object could not be estimated if this kind of object was not

seen at a similar distance during training). The technique presented in Ladicky, Shi, and Pollefeys (2014) follows this approach, by training a classifier that predicts only the likelihood of a pixel being at an arbitrary predefined depth (and manipulating the image to obtain the likelihoods for any other depths) instead of training a pixel-wise depth classifier. In other words, the classifier is given examples of the different objects at a certain -canonical- depth and, from this, the different variations of depth can be estimated in relation to the canonical one, based on the scale of the objects within the image.

A method that also makes use of semantic predictions, but without a dependency between depth estimation and image understanding, is presented in Wang, Shen, et al. (2015). Initially based on the end-to-end CNN from Eigen, Puhrsch, and Fergus (2014), which takes the whole image as input and outputs its complete depth map, the proposed framework jointly predicts a global layout of depth values and semantic labels per pixel. Additionally, the image is decomposed into local segments for which depth and semantic labels are predicted, so that global details -like boundaries and structure of objects- can be captured as well.

As it can be seen, depth estimation algorithms are mostly based on learning algorithms, following a supervised approach. Their main issue, nevertheless, is that they require large amounts of depth information for training, like RGB-D image or video data (especially the CNN-based architectures), which is not easily captured or trivial to obtain.

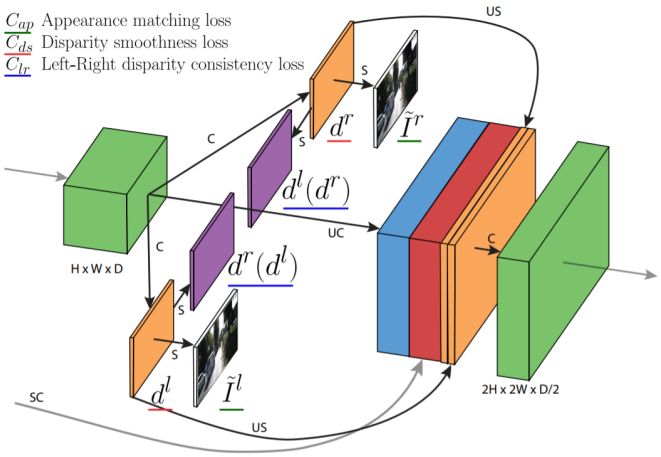

FIGURE 2.10: MonoDepth's deep neural network architecture and training process. Two disparity maps are output (left and right) and compared through the custom loss function. This loss function combines smoothness, reconstruction, and left-right disparity consistency terms. (Image legend: C for Convolution, UC for Up-Convolution, S for Bilinear Sampling, US for Up-Sampling, SC for Skip Connection)

This could be mitigated by replacing this explicit depth data with binocular (stereo) footage, which is easier to obtain. An example of stereo matching-based depth estimation is the technique presented in Godard, Aodha, and Brostow (2017), which is based on a CNN that can generate a disparity map -or in other words, the rules of pixel translation between the left- and right-view image (see Figure 2.10). Furthermore, this network is unsupervised, in a similar fashion to techniques like Deep3D (Xie, Girshick, and Farhadi, 2016) or the CNN architecture from Garg et al. (2016). Instead of training a model by comparing its output depth map to the ground truth one, these stereo view-based approaches follow these steps:

1. The model (typically, a CNN) takes as input a color image (from either the left- or right-view) and outputs a disparity map
2. The opposite side-view image is recreated by shifting the pixels based on the disparity map

During training, the resulting image is compared to the real (opposite side-view) image, and the loss function is applied before iterating. For this step, in the case of Godard, Aodha, and Brostow (2017), they define a custom loss function with three terms reflecting the appearance matching, the disparity smoothness and the disparity consistency, so that the left-right coherence can be verified.

Once a depth map, a disparity map or the opposite side-view image is obtained, and the camera parameters are known, the corresponding estimated scene can be reconstructed in the 3D space.

2.5 Discussion

Achieving full visual anonymization in compliance with the current privacy laws implies the deletion or irreversible manipulation of numerous personal identifiers. Some of these identifiers can sometimes be too subtle, or strongly related to an individual's motion, to be able to anonymize them without compromising the information of interest. An example of this kind of identifier is the human gait, for which an anonymization technique has not yet been proposed.

The majority of the privacy protection techniques that have been proposed in the past are based on the processing of sensitive image regions directly within the source video. Traditional filters such as blurring are still nowadays used for processing faces and license plates, but this approach does not provide full anonymization. Moreover, blurring filters have been seen to provide good intelligibility but inadequate privacy protection compared to other filters, and the opposite happens with more occluding filters like masking (Korshunov, Araimo, et al., 2012). It is noteworthy though that these kinds of filters can offer better privacy preservation when applied to the whole human figure than for instance de-identification filtering. The latter aims at preventing direct identification similarly to face blurring, but the reliability of the results can significantly vary depending on the data used for manipulation. For instance, the algorithms from the k-same family (Gross et al., 2009) generate fake faces based on face images from a default library, but this could be problematic if the library contains faces that are very similar to the ones to be anonymized, ultimately achieving processed faces that could still have some recognizable features.

Although privacy-preserving image processing helps to retain most of the scene information, it has a major risk: if a sensitive region is not correctly detected even in just a single frame, the anonymization process fails and person identification remains possible. This can be solved by creating an alternative video or information feed where only the non-personal information that is relevant for the task is transferred, as we intend to do with our proposed method. A few privacy protection methods based on this concept have been presented before, which tackle this challenge by using different approaches. One proposes the usage of depth information only (Zhang, Tian, and Capezuti, 2012), which can be sufficient for certain applications, particularly for activity recognition in constrained scenarios, but that could be

deemed insufficient for more complex scene analysis. The usage of metadata proposed in Fleck and Straßer (2008) is a good starting point for tasks where tracking or motion pattern analysis are required. Note that, however, this last method requires the geo-location of the people to be tracked, which is unfeasible in an outdoor public setting.

While it is true that this anonymization approach can potentially guarantee full anonymization, detection failures could mean that not all relevant information is retained, and the reliability of the reconstructed scene could be compromised in terms of faithfulness to the original scene.

For both kinds of approaches, a robust object detector is required, either to enforce the privacy of the processed data or the reliability of the scene recreation. The most robust object detection methods that achieve state-of-the-art results nowadays are all based on the usage of CNNs and aim at providing an end-to-end, fully trainable architecture. There are two main trends on how to implement these techniques, which offer a different trade-off between accuracy and real-time application: single shot and region proposal. Single shot techniques provide a very fast execution at the expense of higher accuracy. Region proposal techniques give a better performance, especially when there is a high number of objects in the scene, but are unusable for real-time scenarios.

In the last few years, single shot detectors have proven to give surprisingly good results in terms of performance when evaluated on well-known datasets such as PASCAL VOC (Everingham et al., 2010), even apparently surpassing region proposal-based methods like Faster R-CNN (Ren et al., 2017) when comparing their mean average precision values. Some examples are YOLOv3 (Redmon and Farhadi, 2018), SSD (Liu, Anguelov, et al., 2016) and RetinaNet (Lin, Goyal, et al., 2017). However, most of the times, the results provided by each paper cannot be directly compared to each other because of the differences in, for instance, the quality and format of the input data, optimization functions, equipment or implementation libraries used. As an example, many single shot techniques are evaluated using images of significantly higher resolution than the ones used by the authors of their counterpart methods. Also, due to their nature, single shot methods have certain limitations regarding the objects that can be detected; for instance, very thin objects or small objects cannot be correctly detected, which could be problematic if the objects of interest are pedestrians at a far distance.

When comparing most of these methods re-implemented using the same deep learning framework and evaluated using the same test dataset (Huang et al., 2017), we can see how Faster R-CNN outperforms the other methods in terms of accuracy, although it is significantly slower than even other region proposal methods (see Figure 2.11). Single shot methods are by far the fastest ones, although it is worth mentioning that their accuracy notably varies depending on the CNN architecture used for feature extraction, which can be useful to know when making design choices for a specific application.

Even when obtaining highly accurate results from the object detection phase, we still need to know how the objects are distributed within the three-dimensional space. For this, the depth information, lost in the 2D image capturing process, has to be recovered. Model-based methods make use of known priors about the scene and objects, which can be either semantic properties or general visual characteristics. Semantic priors could work particularly well for rigid objects, and for scenarios where the class labels of all the appearing objects are known or can be extracted.

They could, however, be problematic with non-rigid objects, such as people, whose pose can vary considerably, to the point that depth properties cannot be modeled for every possible situation. General scene priors could perform well but only in scenarios where the lighting has no large variations.

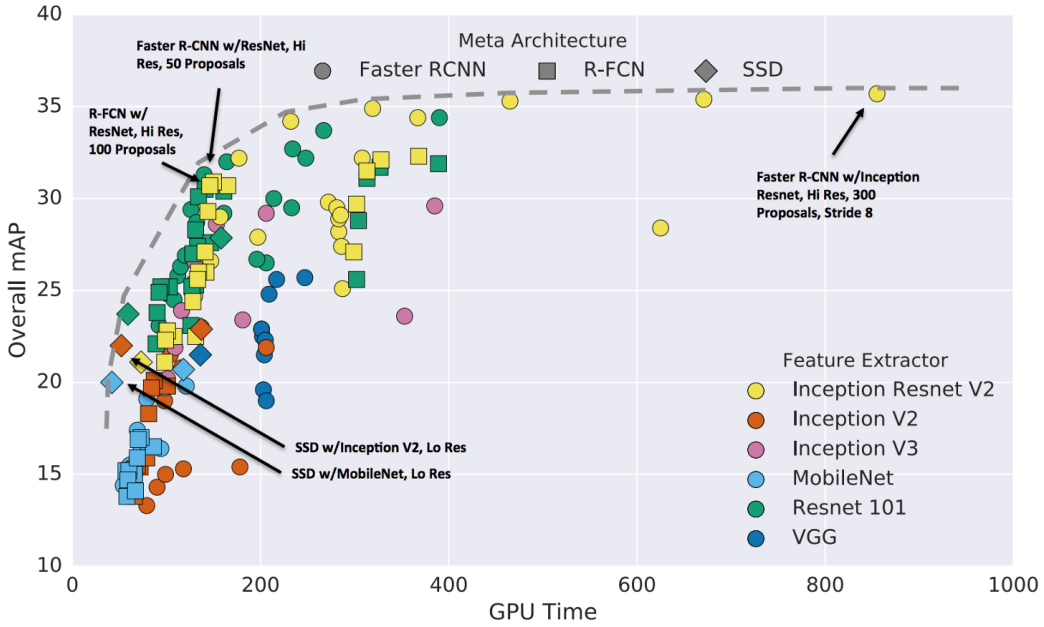

FIGURE 2.11: Comparison of object detection methods (as combinations of approach architecture and type of feature extraction network) in terms of accuracy-time trade-off.

CNN-based methods aim at solving the difficulties of these last techniques, by learning adequate features from the data itself. Many of them follow a supervised learning approach, but the required data for this task is scarce and difficult to obtain. Unsupervised approaches do not require depth data and instead can infer depth from pairs of stereo images. They have proven to be relatively accurate despite its unsupervised character. They also tend to generalize better than supervised techniques, although both of them might not generalize well to every possible case, as there is a high chance that the provided information is insufficient to robustly estimate a depth map of a full scene. Although a shortcoming of unsupervised techniques is that they require additional information, such as the camera focal length, this information is anyways required in our case for the 3D reconstruction task.

Chapter 3

Methodology

In this chapter, we present the methodology we followed, by walking through our approach and the individual modules that it comprises. In Section 3.1, we provide a high-level overview of our method and system architecture, to afterwards delve into each of its modules in the following subsections.

3.1 High-level outline and motivation

As described before, the goal of our method is to estimate the 3D world locations, at each frame, of the relevant objects within a video scene. The architecture of the developed system aims at providing a transparent pipeline for the user, where a video is given as input and the estimated object locations on the ground plane are given as output. These locations could be used afterwards to generate a 3D animation of the scene, using default 3D models for each object class to ensure anonymity. A high-level flow chart of the complete pipeline is given in Figure 3.1.

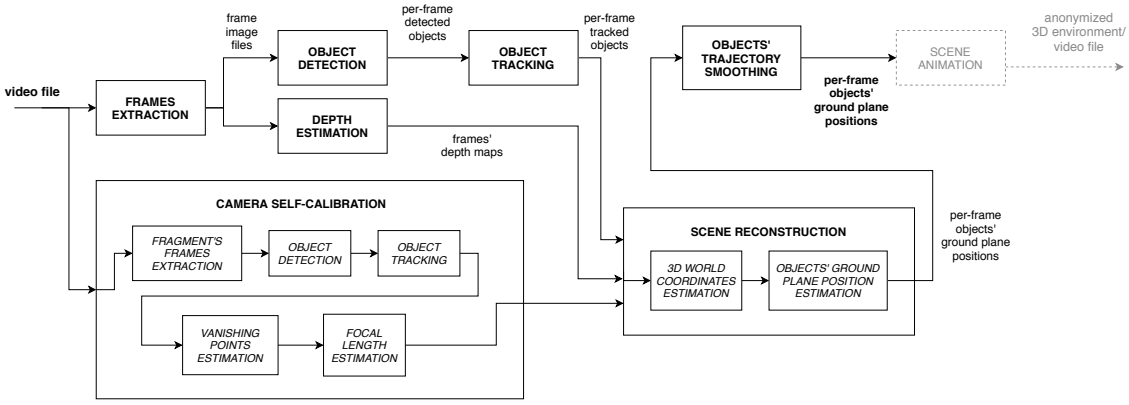

FIGURE 3.1: Diagram of the general system pipeline. It takes as input all the subsequent frames of the video and it outputs, for each of the frames, the estimated locations of the detected objects on the ground plane plus their class label.

Intuitively, we first need to find where the relevant objects are within each 2D frame image. To do so, we must perform object detection focused on the object labels of interest. The selected labels correspond to elements in the scene that are pertinent to traffic patterns and which could potentially display personal identifiers: *person* and road vehicles (*car*, *motorcycle*, *bicycle*, *truck* and *bus*).

To achieve both transparency towards the user and generalization to the input data, we assume that the input video has the following characteristics: (1) it is a

single monocular video, and no redundant visual information is provided or known, and (2) the video comes from an unknown source, and we do not have access to the used camera whatsoever.

These conditions establish two additional requirements for the system: the lost depth information and the camera parameters, respectively, must be somehow recovered or estimated.

For the first aspect, we will perform monocular depth estimation by means of an unsupervised deep learning algorithm, because of its higher applicability to unseen data and the larger availability of training data that resembles our expected scenes. Regarding the second aspect, our objective is to implement a camera self-calibration procedure solely based on the content of the scene that is displayed in the input video.

Once we have obtained this missing information, we can translate every 2D image pixel to its 3D world location, and compute an estimation of the object's location on the world's ground plane by averaging a group of specific pixels within the object area in the image.

Moreover, the fact that only one view of the scene is known, means that the occlusions have a significant influence on the object detection results. To solve this, we will carry out object tracking, in a tracking-by-detection manner, to link the already obtained object detections in the temporal domain. This way, we can smooth the estimated trajectories by, for instance, interpolating the missing ground plane positions corresponding to the frames where the objects temporarily disappear.

3.2 Object detection

Due to its solid performance and the ability to perform segmentation, Mask R-CNN is the chosen method to be used for object detection. The implemented network has been pre-trained on the COCO dataset (Lin, Maire, et al., 2014), and it has been adjusted to only output the desired object classes (*person, car, bicycle, motorcycle, bus* and *truck*). Concerning the input, as Mask R-CNN takes a single image as input, firstly the frames of the video are extracted and stored as separate image files, which are then subsequently processed through the network.

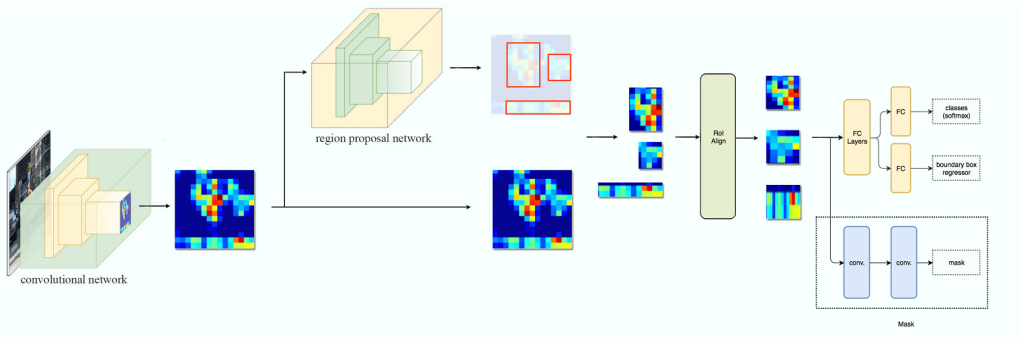

FIGURE 3.2: Overview of the Mask R-CNN method (image taken from Hui (2018)).

Mask R-CNN's state-of-the-art accuracy, in combination with its capability to detect large numbers of objects even under severe occlusion, makes it suitable for the expected street scene. Moreover, its pixel-level segmentation is a useful feature for the overall pipeline, as it encloses the pixels that must be taken into account, for

instance, for locating the head and feet position of the pedestrians, or for estimating the depth and 3D coordinates of their corresponding real world points. On another note, as real-time processing is not a requirement, the lack of a single shot approach for the bounding boxes proposal is not a significant drawback.

As we mentioned at the beginning of Chapter 1, the main risk of our anonymization approach is the potential loss of relevant non-sensitive information. Ideally, we would like to be able to find all the relevant objects, hence prioritizing the network's recall. However, this comes at the cost of a probably higher number of false positives. To give a higher priority to recall, we will reduce Mask R-CNN's confidence threshold, setting it at a default value of 0.75. Afterwards, to filter out potentially wrong detections, we will make use of the chosen tracking algorithm. The rationale behind this is that false positives are likely to appear in just one frame. A tracking algorithm will only output objects that appear several times within the video, hence effectively discarding those isolated detections that have a higher chance of being false positives.

3.3 Object tracking

With the purpose of both improving the performance of the general pipeline and finding points of interest for performing camera self-calibration, we make use of the DeepSORT method (Wojke, Bewley, and Paulus, 2017).

Using tracking for the general pipeline allows us to improve the overall performance of the pipeline in two ways:

- First, as explained in Section 3.2 above, it can help filter out false positives from the detection results. For the subsequent steps in the pipeline, only those objects that the tracking algorithm outputs are considered (which also reduces the computation time during the estimation of the ground plane locations, as the total number of objects is lower).
- Associating objects through subsequent frames in the input video can help us determine when an object was occluded (for instance, by another object passing in front of it), and thus not detected on a certain frame or sequence of frames. With the tracking results, we are able to later interpolate missing ground plane positions for an object corresponding to the frames where it was occluded (see Section 3.8).

The DeepSORT tracker is based on the Simple Online and Realtime Tracking (SORT) algorithm, previously proposed by the same authors (Bewley et al. (2016)). SORT is conceptually based on prior object detection, and is transparent to the object detection method that is used. This means that we can integrate it with our previously chosen object detector, Mask R-CNN, without having to overload the pipeline with repeated detection phases. Moreover, the combination of SORT and Faster R-CNN has ranked higher or on similar ranges of other state-of-the-art detectors in the *MOT Challenge* benchmark (Leal-Taixé et al., 2015) during the last years, and was rated as the best performing open source tracker in 2015.

The main addition of DeepSORT over the original SORT is the fact that it additionally comprises a deep neural network: a CNN that extracts and learns both appearance and motion descriptors from the bounding boxes, in order to improve

the association of detections across frames. It is based on a wide residual network composed of two convolutional layers, six residual blocks and a final dense layer. This CNN enables a more accurate tracking of pedestrians even under complete occlusions, as more robust profiles of them can be built and traced, drastically reducing the number of identity switches.

A characteristic of this additional network that we need to consider, however, is that it was originally trained on a large-scale person re-identification dataset. Hence, it is specifically prepared for people tracking, and the authors did not evaluate its performance nor the full tracker's on other object classes. Nevertheless, we know that pre-trained feature extracting CNNs, if trained on large-scale datasets, tend to generalize well to any kind of displayed objects, regardless of their class; likewise, we can expect this network to be able to learn appearance metrics and descriptors that are applicable to other kinds of objects.

3.4 Depth estimation

For estimating the depth component of the pixels, we implement the unsupervised monocular depth estimation method proposed in Godard, Aodha, and Brostow (2017). This method offers state-of-the-art accuracy, in some cases outperforming supervised approaches. A noted characteristic of this method, that differentiates it from other depth estimators, is that it does not try to predict depth *per se* but it aims at predicting disparity. A disparity map indicates the difference in coordinates of the same pixel, or a similar feature, between two stereo images, due to the physical separation of the cameras. In other words, this method estimates how the paired image of the input image would look like in a stereo setting. For this depth estimation approach, the pair image to be predicted is assumed to be rectified, meaning that the cameras are considered to be parallel and the disparity is only horizontal. Knowing the disparity map, the baseline (distance between the stereo cameras) and focal length of the cameras used for capturing the training data, we can calculate the depth component of each pixel in an input image by means of triangulation.

In Figure 3.3, we can intuitively understand how the disparity and depth values relate. In this top-view diagram of a stereo camera setting, f is the focal length of each of the cameras, the c_x values indicate the locations of the cameras' principal points, and the O points correspond to the optical origins of the cameras. The distance between the origins of the cameras is the baseline, defined as T in the diagram. P is a point from the real world, located at the (X, Y, Z) coordinates. We consider that the origin of each of the image planes is on their upper left corner, and thus x^{l} and x^{r} are the x-coordinates of the projections of P in the images captured by the left and right cameras, respectively. Therefore, the disparity d in this case is the difference between these two values.

If we consider the property of triangle similarity, we know that the triangle $P - O_l - O_r$ is similar to the triangle $P - x_l - x_r$ and their sides are therefore proportional:

$$\frac{T}{T-d} = \frac{Z}{Z-f} \tag{3.1}$$

Which we can decompose and obtain:

$$\frac{f}{Z} = \frac{d}{T} \Rightarrow Z = \frac{f \cdot T}{d} \tag{3.2}$$

Using this equation, we can hence calculate the depth values from the disparity map and the known camera parameters.

As the availability of RGBD or stereo-video datasets that closely match our expected scenario is scarce, we do not intend to train a custom model. Instead, we will focus on the pre-trained models provided by the authors. These models have been trained on large amounts of data, taken from some of the most well-known datasets for autonomous driving systems: *KITTI* (Geiger, Lenz, and Urtasun, 2012), *Cityscapes* (Cordts et al., 2016), and several subsets combining data from both. Although these datasets were recorded with moving cameras (placed at the top of a car) instead of static cameras at higher locations, the images do match the content of our expected scenes (fundamentally outdoors road or street scenes with presence of pedestrians and diverse vehicles).

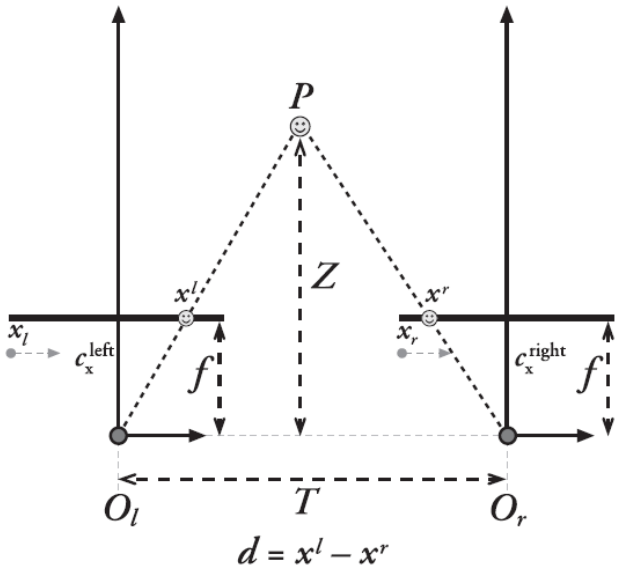

FIGURE 3.3: Simplified two-dimensional diagram of an undistorted, rectified stereo vision system. Picture taken from Bradski and Kaehler (2013).

Due to the ill-posed nature of the monocular depth estimation problem, the authors made some compromises between the accuracy of this method and its general applicability. The model was evaluated on a test split taken from the *KITTI* dataset, and despite comprising data that was not seen during the training, it is still expected to share some resemblance to the training data. Therefore, the evaluation results must be taken with reservations. In the words of the authors, the model is expected to give a better performance if the input data looks similar to the data that was last seen during training (be it the training dataset or the data used for tuning the model). Moreover, they acknowledge the uncertainty on how the disparity can be converted into depth for images with aspect ratios and focal length values that do not match those of the training data[1].

[1] Based on the comments provided by the first author of the paper on the official code repository: https://github.com/mrharicot/monodepth/issues/118

Our approach is to consider that the predictions are precisely based on the stereo camera setting used to record the training images. We convert the disparity values into depth values using the Formula 3.2 we obtained before. As a reminder, and substituting the parameters with their regular names for a more intuitive formulation:

$$depth = \frac{baseline \cdot focal}{disparity} \tag{3.3}$$

Where *focal* and *baseline* correspond to the focal length and baseline distance values of the training data, respectively.

Due to inaccuracies of the method, sometimes a *disparity* value of zero is predicted, theoretically meaning that the pixel corresponds to a point that is very far from the camera, at the optical infinite. In this case, as the formula would yield an infinite value, we first modify the *disparity* value with a pre-defined minimum value of 10^{-4}.

Another limitation of the method itself, which is typically seen in recent monocular depth estimators, is that the estimations are done in a per-frame basis. This means that there is no temporal consistency among the predictions for consecutive video frame images. We intend, nevertheless, to overcome the shortcomings of this aspect by making use of the tracking results.

3.5 Camera self-calibration

The video camera self-calibration procedure is based on the approaches in Lv, Zhao, and Nevatia (2006), Tang, Lin, Lee, Hwang, Chuang, and Fang (2016), and Tang, Lin, Lee, Hwang, and Chuang (2019), among others, which are essentially based on the estimation of vanishing points from the shapes of tracked pedestrians.

In Figure 3.4, we can see an overview on how this process works. Its primary step is the computation of poles (vertical lines representing pedestrians in the image, which go from the location of the head to the location of the feet). The rationale behind this method is that, assuming that a tracked pedestrian is standing straight and perpendicularly to the ground plane:

- All poles should intersect at the vertical vanishing point
- The line passing through the upper ends of two poles (head locations) and the line passing through the corresponding bottom ends of the poles (feet locations) computed from the same tracked pedestrian at different times, are parallel in the real world and should intersect in the horizon line (that is, in a horizontal vanishing point)

By estimating enough poles, we can ultimately compute three vanishing points (the vertical one and two on the horizon line, see Figure 3.5) which can be used to estimate the camera parameters. This approach works in our case because the camera is supposed to be static and have constant focal length and extrinsics.

Although theoretically, three poles would be enough to compute the vertical vanishing point and the horizon line, in practice the object detection and tracking phases are not expected to be sufficiently precise to ensure this. To avoid a significant influence of these noisy pole estimates, we will compute a higher number of poles.

Moreover, to improve the overall estimation of the vanishing lines, we would focus on tracking only one person among the different frames. This is in any case required for estimating the horizontal vanishing points, as the poles must have the same height.

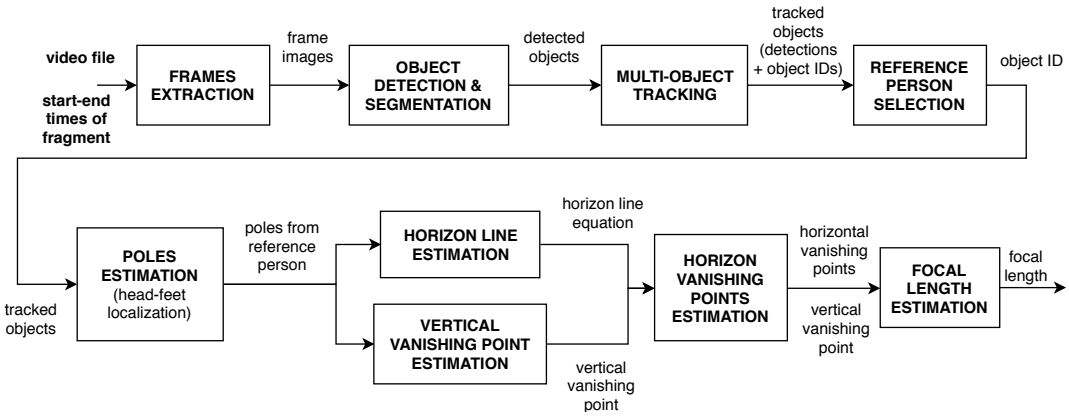

FIGURE 3.4: Diagram of the high-level functioning of the camera self-calibration method. It takes as input the video frames, or a selection of them, where at least one pedestrian is present and walking. This module outputs the estimated focal length of the camera, which we require in order to convert 2D image points into 3D world coordinates.

To calculate the locations of the head and feet, in a similar fashion to Tang, Lin, Lee, Hwang, Chuang, and Fang (2016), each mask area corresponding to a person object is fitted into an ellipse. The intersection of its major axis with the upper edge of the bounding box defines the head location, and likewise, its intersection with the bottom edge determines the feet location. This way, the estimated poles are consistent with each other.

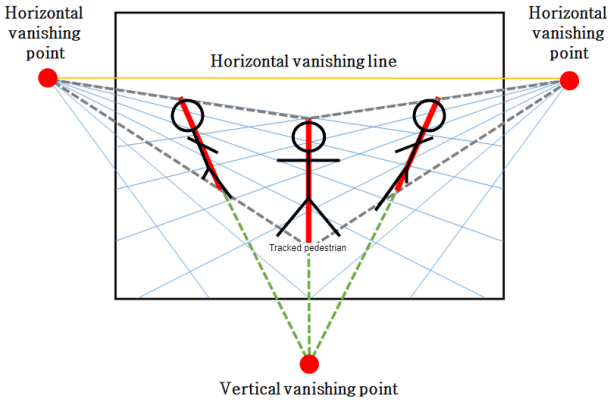

FIGURE 3.5: Visualization of the rotationale behind the camera self-calibration method. The lines corresponding to the estimated poles are expected to intersect in the vertical vanishing point, whereas the parallel lines crossing pairs of heads and feet locations, respectively, are expected to intersect on the horizon line (image modified from Jaehoon, Yoon, and Paik (2016)).

Once we have computed all the potential horizontal vanishing points, we can find an estimate of the horizon line by using linear regression. In the case of the vertical ones, to avoid a major influence of noise and outliers (which are expected to be

abundant), we apply mean-shift clustering. The center of the cluster with the highest number of candidate points is used as our estimation for the vertical vanishing point V_y, reducing the influence of smaller clusters of outliers.

To find the two required horizontal vanishing points for calibrating the camera, we follow the same procedure as in Tang, Lin, Lee, Hwang, Chuang, and Fang (2016). For the first horizontal vanishing point V_x, we select a random point on the horizon line; in our case, for ease of computations, we choose the one passing through $x = 0$ (meaning that its y-coordinate will have the same value as the horizon line's intercept). In a three-point perspective, the camera's principal point has the property of being the orthocenter of the triangle composed by the three vanishing points. This means that, if we consider an imaginary line passing through the principal point which is perpendicular to the $V_x - V_y$ line, this line intersects with the horizon line at the missing vanishing point V_z.

Eventually, our goal is to calculate the focal length based on these estimations. We do so by following the general approach in other self-calibration methods (such as Lv, Zhao, and Nevatia (2006), Junejo and Foroosh (2006), Liu, Collins, and Liu (2011), and again Tang, Lin, Lee, Hwang, Chuang, and Fang (2016)). First, we calculate the roll angle of the camera based on the coordinates of the horizontal vanishing points:

$$roll = tan^{-1}\left(\frac{v_{v_z} - v_{v_x}}{u_{v_x} - u_{v_z}}\right) \tag{3.4}$$

Where u and v represent the (x, y)-coordinates of each corresponding horizontal vanishing point. The focal length can be calculated as:

$$f = \sqrt{-(v^{rot}_{v_x} \cdot v^{rot}_{v_z} + u^{rot}_{v_x} \cdot u^{rot}_{v_x})} \tag{3.5}$$

With the inner terms being defined as:

$$\begin{aligned} v^{rot}_{v_x} &= cos(roll)(v_p - v_{v_x}) - sin(roll)(u_{v_x} - u_p) \\ v^{rot}_{v_z} &= cos(roll)(v_p - v_{v_z}) - sin(roll)(u_{v_z} - u_p) \\ u^{rot}_{v_x} &= cos(roll)(u_{v_x} - u_p) - sin(roll)(v_p - v_{v_x}) \\ u^{rot}_{v_z} &= cos(roll)(u_{v_z} - u_p) - sin(roll)(v_p - v_{v_z}) \end{aligned} \tag{3.6}$$

Where the u_p and v_p correspond to the coordinates of the principal point.

Due to the absence of any open source implementation of this method, and considering the time limitations of the project, we have implemented the method with certain simplifications in relation to Tang, Lin, Lee, Hwang, Chuang, and Fang (2016), such as the suppression of the optimization phase based on the estimation of distribution algorithm. In the aforementioned paper, they make use of a Laplacian linear regression model to estimate the horizon line. Their objective is to model the outliers and noise with a distribution that has heavy tails, in this case, a Laplace distribution, instead of a more typical one like the Gaussian distribution. This kind of approach can be classified within the family of robust linear regressors, which aim at fitting a linear model in the presence of a significant number of outliers. This is an

appropriate kind of linear regression technique for the problem at hand, as the instability of the pole estimation is expected to deliver many outliers when computing candidate vanishing points. Furthermore, one of the main drawbacks of robust regressors is that they do not work on high-dimensional settings, which does not apply in our case, as we have a two-dimensional scenario of spatial coordinates. Nonetheless, the Laplace linear regressor is a relatively uncommon regression method and has not been implemented as part of any standard machine learning library. Due to this, we opted for implementing a Huber linear regressor with its *epsilon* parameter set at the maximum robustness. We considered this technique because it is the closest conceptually to a Laplace linear regression as, contrary to other robust regressors like RANSAC or Theil-Sen, it does not ignore the outliers but rather weights their effect.

The need to implement this method from scratch, however, has also allowed us to include certain improvements over the original approach. The method proposed in Tang, Lin, Lee, Hwang, Chuang, and Fang (2016) and its derivatives by the original authors (e.g. Tang, Lin, Lee, Hwang, and Chuang (2019)), make use of a tracking algorithm (Tang, Hwang, et al. (2016)) that is based on background subtraction and shadow removal for detecting the pedestrian objects. As seen in Section 2.2, the classic object detection methods have mostly been outperformed by the ones based on deep learning. With the purpose of improving the accuracy of this step, and for the sake of optimizing the whole pipeline, we can substitute the background subtractor and shadow remover with our Mask R-CNN module. This is especially useful because Mask R-CNN already performs segmentation and provides the polygonal masks of each detected object, which we can convert to binary masks and use for the ellipse fitting process. Likewise, for the tracking process DeepSORT can be reused. As mentioned before, DeepSORT's appearance feature network has been trained on people re-identification data, making it particularly adequate for this task.

It is noteworthy that the target transparency of the system allows the input video to be, in principle, of any length. To ensure this end-to-end character of the system, the self-calibration module processes the whole input video by default. However, this means that, if the input video has a relatively long duration, the number of computed poles and hence candidate vanishing points could be excessively high, leading to drastic increments in the computation times. To avoid this kind of situation, the user is provided with the option of defining the starting and ending times of a fragment within the input video to be used for self-calibration. Although this introduces a new input parameter, it improves the overall performance in two ways: the computation times can be reduced, and the user can actively choose an optimal video sequence (for instance, where the pedestrians are more clearly visible or display more movement around the whole scene).

As the reference object for self-calibration, we will select the tracked object labeled as *person* that appears in the highest number of frames within the self-calibration fragment. This way, we can obtain a high number of poles from a pedestrian that is expected to be clearly visible throughout the video, while also maintaining the automatism of the method.

This kind of technique is conceptually suitable for our application for several reasons. First, it requires a similar scenario to the one expected in our input videos: there are pedestrians present, and their height is assumed to be constant as they will be either standing or walking, thus always perpendicular to the ground plane. Secondly, the camera setup matches the one assumed in these approaches, where the

camera is static and located at a higher position than the recorded pedestrians. In fact, these are roughly the only assumptions made for these approaches, aside from the camera model assumptions. Otherwise, these self-calibration techniques can be considered relatively flexible, assuming that the object tracking process performs sufficiently well.

3.6 3D world coordinates estimation

Once we have estimated the locations of the vanishing points through the self-calibration process and the value of the focal length, we can compute the remaining camera parameters that are required for the 2D-to-3D conversion. Particularly for our case, we are only interested in the spatial relationships among the objects in the scene and their motion patterns, and we do not require strictly accurate distances in terms of metric units of length. For this reason, we can define some constraints on the expected camera parameters that simplify the process of estimating them and reduce the complexity of the calibration problem. To do so, there are certain assumptions that are typically made for modern cameras, mainly the following:

- We consider a pinhole camera model for our input video camera
- We assume that the recorded images have very low to no distortion (as it is one of the assumptions made for the depth estimation method)
- The images have been rectified beforehand (as the disparity map that the depth estimation outputs is based on predicting a rectified stereo pair image)
- We expect the camera to have zero skew, and square pixels

A pinhole camera model basically considers that the lens is a barrier between the real world scene and the virtual image, containing a single point that acts as an opening and lets the light rays in. This point would correspond to the aperture of the modeled camera. Another typical assumption for the simplification of the calibration problem is to consider that this point, or camera origin, is located at the center of the image plane.

Considering this, the 3D world coordinates of a certain (pixel) point in the 2D image can be calculated from:

$$\overline{\mathbf{X}_{image}} = \mathbf{P} \cdot \overline{\mathbf{X}_{world}} \tag{3.7}$$

Where $\mathbf{P}$ is the camera projection matrix:

$$\mathbf{P} = \mathbf{K} \cdot [\mathbf{R}\ \mathbf{T}] = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{3.8}$$

With c_x and c_y being the camera center coordinates, s being the skew coefficient, the r elements corresponding to the rotation matrix, and the t elements corresponding to the translation vector. The left-side matrix actually represents the camera intrinsic parameters, while the right-side matrix represents the extrinsics. If we take into account the previously defined assumptions, we know that:

- There is no skew, thus the image axes are perpendicular and $s = tan(90°) = 0$
- The pixels are squared, therefore $f_x = f_y = f$
- The principal point of the camera is located at the center of the image plane

Based on this, **P** can be expressed as:

$$\mathbf{P} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{3.9}$$

Solving for the world coordinates, we end up with the following system:

$$\overline{\mathbf{X}_{world}} = \mathbf{P}^{-1} \cdot \overline{\mathbf{X}_{image}} \tag{3.10}$$

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \left[\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \right]^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.11}$$

Where u and v are the 2D image coordinates, and x, y and z are the 3D world coordinates (from which z is known, as it is obtained during the depth estimation phase). Note that, however, this is applicable to a single camera from which we know both the intrinsic and extrinsic parameters, whereas in our case, we have recreated a stereo setting. In this scenario, there is additional information as we have two projections of each 3D point, corresponding to the two cameras. The geometry describing the relationships among two (or more) cameras, the 3D points and their projections, is called epipolar geometry (see Figure 3.6).

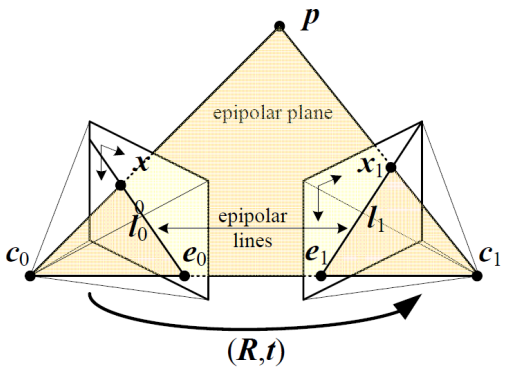

FIGURE 3.6: Typical representation of the epipolar geometry properties. The vector $t = c_1 - c_2$ represents the baseline, or distance between the cameras. The vectors l_0 and l_1 are referred to as the epipolar lines, while the e_0 and e_1 points are the epipoles (image taken from Szeliski (2010)).

If we consider the world reference system to be associated to the first camera (Figure 3.6, left-side), the camera projection matrix for this camera is defined as:

$$\mathbf{P}_0 = \mathbf{K}_0 \cdot [\mathbf{I}\ \mathbf{0}] \tag{3.12}$$

Where I is the identity matrix and K the matrix containing the camera intrinsics. The projection matrix of the second camera will hence be:

$$\mathbf{P}_1 = \mathbf{K}_1 \cdot [\mathbf{R}\ \mathbf{T}] \tag{3.13}$$

Where R and T represent the offset (rotation and translation) of the second camera's origin relative to the first one's. Nevertheless, one of our assumptions is that there is no distortion and the images have been previously rectified. When modelling this with epipolar geometry, the image planes of the cameras are considered to be parallel and aligned, obtaining the scheme shown in Figure 3.8. The offset of the second camera's origin is therefore only a translation in the u horizontal axis (a value we have previously referred to as the baseline), meaning that the projection matrix of this camera can be represented as:

$$\mathbf{P}_1 = \mathbf{K}_1 \cdot [\mathbf{I}[\mathbf{T}_x]] \tag{3.14}$$

Where T_x is the translation on the horizontal axis, that is, the baseline in our rectified scenario. Knowing that the images are aligned and that the cameras share the same intrinsic parameters ($K_1 = K_2 = K$), the projection matrices take the following shape:

$$\mathbf{P}_0 = \mathbf{K} \cdot [\mathbf{I}\ \mathbf{0}] = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.15}$$

$$\mathbf{P}_1 = \mathbf{K} \cdot [\mathbf{I}[\mathbf{T}_x]] = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & f \cdot T_x \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.16}$$

As seen before for a single camera, these projection matrices define the correspondence of 3D points to 2D points within an image. Nevertheless, we are interested in combining both matrices in a way that we can directly transform the 2D coordinates to 3D. Likewise to what we explained in Section 3.4 for the Equation 3.2, we can use triangle similarity to find the relationships between all these variables.

In Figure 3.7, we provide a three-dimensional overview of the camera model that we already showed in Figure 3.3 in 2D. Firstly, let's consider that there is a line parallel to $O_l - O_r$ that crosses P, and that intersects the left camera's principal ray at a point r_c. In this setting, we know that the triangle formed by O_l, P and r_c is similar to the triangle formed by O_l, the left camera's principal point (c_x, c_y), and the left projection of P, namely p_l. If we see this scenario from above, that is, from the same perspective as in Figure 3.3 by considering the x- and z-coordinates only, we know the following:

- The distance on the x-axis between r_c and P is the value of P's coordinate X
- That same line segment, between r_c and P, is proportional to the segment between (c_x, c_y) and p_l
- Likewise, the segment between r_c and O_l equals the Z depth value, and it is proportional to the segment between the principal point (c_x, c_y) and O_l, which

equals the focal length value f

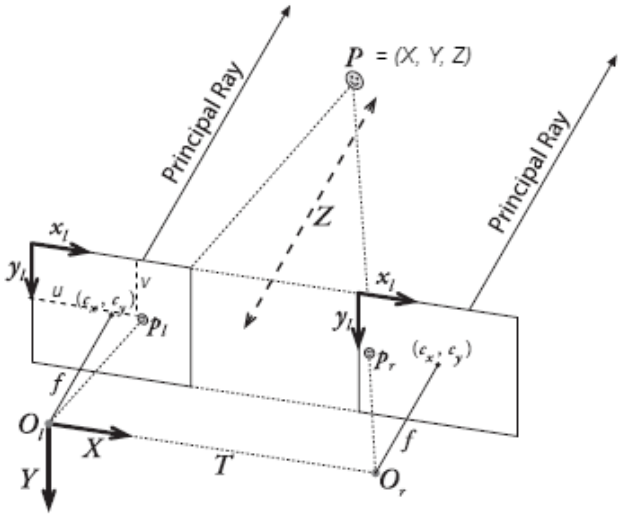

FIGURE 3.7: Three-dimensional representation of an undistorted, rectified stereo camera setting. Image taken from Bradski and Kaehler (2013) and modified to show all the relevant parameters.

If we express this in an equation, defining u as the x-coordinate of p_l, we get:

$$\frac{u - c_x}{X} = \frac{f}{Z} \tag{3.17}$$

As we already know that $Z = \frac{f \cdot T}{d}$, if we substitute Z:

$$\frac{u - c_x}{X} = \frac{f \cdot d}{f \cdot T} = \frac{d}{T} \tag{3.18}$$

Which solving for X:

$$X = T \cdot \frac{u - c_x}{d} \tag{3.19}$$

Likewise, we can apply the same logic for calculating the Y component:

$$\frac{c_y - v}{Y} = \frac{f}{Z} = \frac{d}{T} \Rightarrow Y = T \cdot \frac{c_y - v}{d} \tag{3.20}$$

To have a similar shape to the expression of X, we can change the negative sign:

$$Y = T \cdot \frac{v - c_y}{(-d)} \tag{3.21}$$

We can express P's (X, Y, Z) coordinates in homogeneous coordinates. This is typically done in 3D reconstruction to allow transformations to be expressed using matrices, and also enable the representation of points at the infinite. This way, if we define $(X, Y, Z) = (\frac{X'}{W}, \frac{Y'}{W}, \frac{Z'}{W})$, we can group the equations described above in matrix form:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ W \end{bmatrix} = \begin{bmatrix} u - c_x \\ v - c_y \\ f \\ \frac{-d}{T} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T} & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{3.22}$$

Where the $4x4$ matrix is called the reprojection matrix.

We assumed that the principal point (c_x, c_y) of the camera was located at the center of the image plane, and hence we can calculate it based on the height and width of the images. The focal length value f is previously estimated through the self-calibration process. The baseline T_x is, as described in Section 3.4, already determined by the baseline value of the training dataset images; this is due to the fact that the depth estimation method works under the assumption that the predicted stereo setting is that of the training data.

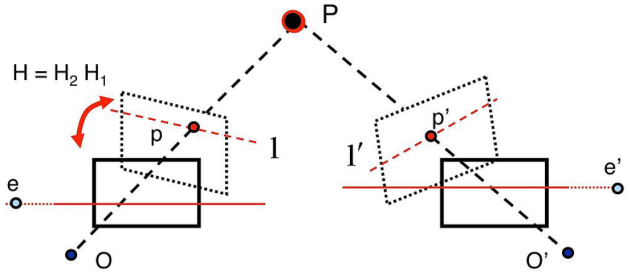

FIGURE 3.8: Rectification of images represented in epipolar geometry, where the image planes become parallel. The epipoles are located at the infinite, and the epipolar lines are parallel to the common horizontal axis of the image planes. Image taken from K. Hata (2017).

In the aforementioned Section 3.4, we explained how we obtain the depth map from the rectified disparity map by using the focal length value of the training dataset. For the 3D world coordinates estimation, we recalculate the disparity map from this depth map using the same conversion formula, but this time using the estimated focal length f of the real camera.

3.7 Estimation of ground plane locations

Once the 3D world coordinates can be computed for each of the 2D image points within a frame image, we can do so for those pixels belonging to the tracked objects. Nevertheless, our interest does not lie on finding the correspondence of every detected point belonging to an object, but rather on knowing the object's approximate location on the 3D world.

Firstly, as commonly done in 3D scene reconstruction, we assume that the ground floor is practically horizontal, which we can generally expect on a regular street or road scene. Under this condition, we will not take the vertical y-axis coordinate into account, as we do not want to retain height information which could be potentially considered personal data. Therefore, our aim is to calculate the (x, z) location of the objects on the ground plane.

In a multi-view setting for multiple object tracking, where every angle of an object is covered and it can be fully reconstructed in the 3D space, calculating the ground plane position is relatively trivial. An intuitive way would be to average the horizontal x- and z- coordinates of every point of the object; this way, the ground position is represented by the centroid of the object structure, which is expected to be preserved even if the object rotates at the same position (assuming the 3D reconstruction is sufficiently robust). This location estimation can be generalized for any kind of object regardless of their shape.

In a single monocular view, however, this problem is not trivial. Not only the depth estimation is expected to perform significantly worse, but only one view of the object is provided, meaning that the centroid of the estimated 3D points of the object will vary if the object changes its orientation.

A general approach is to use a certain point in the bottom part of the bounding box indicating a detected object. As explained before in Section 3.5, an example would be to approximate the object to an ellipse (particularly if the detected object is a person) and consider that the location of their feet is determined by the ellipse's major axis intersecting with the bottom edge of the bounding box (Tang, Lin, Lee, Hwang, Chuang, and Fang, 2016). In a similar manner, Sochor, Juránek, and Herout (2017) compute the convex hull of the object (in this case, a vehicle) and they consider its ground plane position to be the center of the edge corresponding to its bottom front edge. While these methods are appropriate for each object class, they do not seem adequate for generalizing to any kind of object. In particular, the approach in Sochor, Juránek, and Herout (2017) implicitly assumes that a car will not turn around, because the new location point would swap to the edge on the opposite side of the vehicle. Moreover, as the depth estimation from a single self-calibrated camera is not expected to provide highly accurate results, relying on a single point is unsafe.

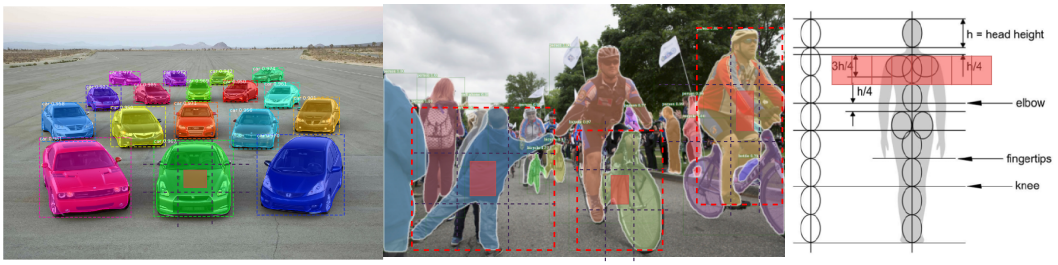

FIGURE 3.9: Examples of object detection and segmentation with Mask R-CNN (left and middle images) particularly of *cars* and *bicycles* from different viewpoints. In the right image, scheme of human body proportions based on the height of the head; if the person's arms were extended horizontally, the length of the bounding box surrounding it will roughly take the space of 8 x width of the head. (Original left, center and right images by Brad Sliz, Facebook Research and De Silva (2008) respectively; the left and center images have been modified to display the area from which the points are selected and averaged to estimate the location of the object).

Taking this into account, we propose an approach that could be considered as a trade-off between the general procedure in a multi-view setting, and the criteria defined in Tang, Lin, Lee, Hwang, Chuang, and Fang (2016) and Sochor, Juránek, and Herout (2017). The main idea is to find an average point among all the estimated 3D points from a central area of the segmented object that we want to locate. The

rationale behind this is that we can expect this area of an object to have a more consistently estimated depth than the boundaries of the object, especially for *person* and vehicle objects such as a *bicycle* or a *motorcycle*, where the segmented area is notably non-convex and severely holed in the case of the vehicles.

This centered area is defined as follows:

First, we divide the bounding box of the object in a $3x3$ grid, from which we focus on the central cell. This cell will encompass the final area, with which it will share its center point. We define two threshold values: a horizontal one and a vertical one, determining the width and height of the area, respectively. By default, we mark the height as $1/5$-th of the total height of the bounding box. This value is set because we can expect this part of an object to be mostly free of holes: regarding vehicles, it will comprise part of its body or frame, and it will belong to the torso and abdomen in the case of people. For the width, we consider the utmost case of a person extending their arms horizontally, where a drastically large part of the bounding box contains points not belonging to the object itself (and belonging, for instance, to the scene background). In this situation, based on the scheme of the human body proportions (De Silva, 2008), the distance between the tips of the fingers can be roughly approximated to 8 times the width of the head. As the widths of the torso, waist and hips are always expected to be larger than that of the head, we can safely assume that the center area will not surpass the sides of the body if its width is equal to $1/8$-th of the bounding box's width.

As we expect the depth estimation to be relatively noisy, in order to avoid a major influence of the outliers, we will calculate the median (instead of the mean) of the selected points' coordinates. The resulting point will define the estimated ground plane location of the corresponding object.

3.8 Trajectory smoothing

As we have seen, the purpose of the whole system is to perform the above explained 2D-to-3D conversion for each of the subsequent frames in the input video. However, this approach does not consider the temporal aspect of a video. To cover this domain, we make use of the aforementioned multi-object tracking.

Multi-object tracking will not only filter out potential false positive detections, but it can also help us relocate objects that disappeared or were occluded at a certain moment along the video. If an object disappears at a frame, it will not be detected during the object detection phase and no bounding box location will be provided, hence the object's position on the ground plane cannot be estimated. However, if the object reappears and the object tracking correctly identifies it, we can associate its new bounding box with the former one. Once we have estimated the ground plane positions for these two detections, we compute the in-between lost positions by using interpolation. The temporal gaps where an object is occluded often take just a few seconds, and we can therefore presuppose that the object's trajectory can be approximated to a linear path. We interpolate n ground plane positions between the two known positions, with n being the number of frames where the object went missing.

Ultimately, the system will output a sequence of per-frame locations of the objects on the ground plane, along with their associated object label. These location points are plotted on a two-dimensional grid representing the ground plane from

the perspective of the camera. Each of the points is color-coded depending on its class label, for which a graph legend is provided.

Although the visualization of these locations in a grid is sufficient for the desired application of the proposed method, ideally it would also be of interest to recreate these positions in a 3D scenario, for ease of visualization by the user. In this 3D recreation, each object would be represented with a default 3D model depending on its class label.

Chapter 4

Evaluation

In this chapter, we introduce our evaluation procedure for assessing the performance of our proposed system. In Section 4.1, in addition to covering the main characteristics of the implementation, we discuss which data was used for training the deep learning models, and which data we will use for testing purposes. Our evaluation approach is based on, first, evaluating each of the models individually, to ultimately assess how the whole pipeline performs. To do so, we selected several metrics for the quantitative analysis of the performance, which we describe in Section 4.2.

4.1 Implementation

Within the proposed anonymization pipeline, we make use of three different architectures based on convolutional neural networks: Mask R-CNN for object detection, MonoDepth with its fully convolutional neural network, and DeepSORT's CNN for object discrimination during tracking. For the three of them, we implement pre-trained models provided by the authors or other third-party entities. The reasons behind this are that training custom networks would be too computationally expensive, even more considering the time constraints of the project and the small performance improvement that we could expect to achieve; and that these pre-trained versions already give state-of-the-art results in similar settings to ours.

4.1.1 Training data

The three abovementioned default models were already trained using well-known datasets that are relevant for their purpose, which already match the content of our expected input videos, in terms of the objects present and the outdoors background scene. Finding and pre-processing other relevant data for them would also be time-consuming and again potentially not yielding an improved performance. Particularly for depth estimation, there is a significant lack of datasets that are large enough or with the sufficient quality for training purposes. This is especially true for RGBD datasets -precisely one of the reasons why we chose an unsupervised method-, as most of them focus only on displaying inanimate rigid objects in indoor spaces, such as the popular *NYU-Depth* dataset (Silberman et al., 2012) and its newer versions.

For training Mask R-CNN, the *MS COCO* dataset (Lin, Maire, et al., 2014) was used. This dataset contains 328.000 images with 2.5 million instances of 80 different object classes to be used for object detection, localization and segmentation systems. Among those 80 object classes, our labels of interest are present: *person*, *car*, *bicycle*, *motorcycle*, *bus* and *truck*. Instead of portraying the objects in an isolated manner, the

images within *MS COCO* aim at reflecting everyday scenes with the complexity that they can entail: cluttered environments, objects placed far on the background, partial occlusions, variety of viewpoints, etc. This way, models trained on this dataset tend to be more robust, offering a better generalization to unseen data.

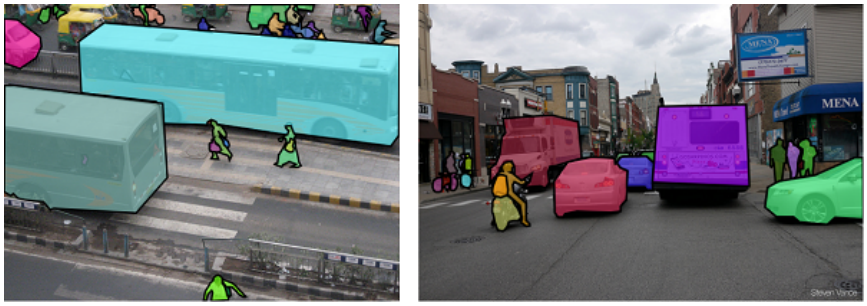

FIGURE 4.1: Example images from the *MS COCO* dataset, portraying a higher (left image) and lower (right image) camera height, respectively. Both contain several instances of our objects of interest, overlaid here by their ground truth segmentation masks.

In the case of the monocular depth estimator, the authors provided several different models, each of them trained on a different subset of stereo images. The two main datasets that are used are *KITTI* (Geiger, Lenz, and Urtasun, 2012) and *Cityscapes* (Cordts et al., 2016), two well-known datasets for training computer vision algorithms applied to autonomous driving. Both of these datasets were recorded from the top of a moving car, and provide rectified image pairs which display real urban street scenes. Although the camera placement implies that the viewpoint might be lower than that of the expected input, the content of the recordings closely matches the one we expect in our particular input setting (see Figure 4.3): both *Cityscapes* and *KITTI* contain the object labels *person* (and derivatives such as *rider* or *person sitting*), *car* and *truck*, matching those of the *MS COCO* dataset. Moreover, *Cityscapes* also contains the labels *bicycle*, *motorcycle* and *bus*, the three remaining labels from *MS COCO* of our selection.

FIGURE 4.2: Example images from the *MARS* dataset, where each rows corresponds to an individual person identity, and each column corresponds to a different camera.

To analyze how the performance varied depending on the kind of training data, five different models of this depth estimator were trained, on five different subsets: the *Cityscapes* training set, the *KITTI* training set, the *Cityscapes* training set plus the *KITTI* training set for tuning, the *Eigen* split, and the *Cityscapes* training set plus the *Eigen* split (Eigen, Puhrsch, and Fergus, 2014) for tuning. The *Eigen* split is a customized dataset containing 20.000 images from the *KITTI* training set and 120.000 images from the *NYU-Depth* dataset mentioned before.

Lastly, the wide residual network from the DeepSORT tracker was trained on the *MARS* dataset (Zheng et al., 2016), a large-scale dataset focused on person re-identification. It contains around 1.1*M* images displaying pedestrians in real scenes (see Figure 4.2). As noted before in Section 3.3, this dataset only contains images of pedestrians, while our aim is to use it for both people and vehicles. Nonetheless, feature extracting networks for object re-association tend to learn patterns based on characteristics that can be found in different kinds of objects, such as color distributions or shapes. An indication of this is the well extended usage of initial weights pre-trained on the ImageNet dataset (Krizhevsky, Sutskever, and Hinton, 2012) for CNNs in general, regardless of their final purpose.

4.1.2 Evaluation data

To assess how well the object detector generalizes to unseen data, the trained model will be tested using the validation subset of the *Cityscapes* dataset (Cordts et al., 2016). For ease of use, we will convert its annotations to *MS COCO*'s format by mapping the IDs of the relevant classes and adjusting the formatting of the boxes and segmentation masks. This dataset has been chosen because of its closeness to the expected street scenes content-wise: it displays outdoor street settings annotated with the same classes as the *MS COCO* ones that are of interest for the project, plus the label *rider* of either a bicycle or a motorcycle, which we will map to *person* to maintain the same assumptions as in *MS COCO* (and therefore as in the pre-trained Mask R-CNN model). The video cameras used to record the data are also relatively similar to surveillance cameras in terms of optical specifications, such as its short focal length. Moreover, as opposed to other similar datasets such as *KITTI*, *Cityscapes* also contains annotations for instance segmentation masks in addition to the bounding boxes, which makes it useful to fully evaluate the performance of pixel-level segmentation methods like Mask R-CNN.

FIGURE 4.3: Example images from the *KITTI* dataset, portraying the paired frames recorded with the left and right cameras of the stereo setting. Both contain several instances of our objects of interest, and have been recorded from a low camera height.

For depth estimation, we will make use of the *KITTI 2015 Stereo* split, which was used by the authors of the chosen method (Godard, Aodha, and Brostow, 2017) for their evaluation. This way, we can make sure that the test split does not include data that was seen during training regardless of the pre-trained model that we use, which we could not ensure if we used our own custom dataset (if it was based on *KITTI*, *Cityscapes* or *NYU-Depth*). By using this evaluation set, we will be able to compare the results with those published by the authors. Furthermore, we can also evaluate the performance of the depth estimator on the areas of interest (based on the segmentation masks from the detected objects) to compare them to the global results for the whole frames.

A dataset that we will repeatedly use to evaluate the rest of the modules is *3DMOT*, a subset from the *MOT (Multi-Object Tracking) Challenge* benchmark (Leal-Taixé et al., 2015) focused on multi-view tracking of people in the 3D world. It provides simultaneous recordings of diverse outdoors scenes from several cameras, which means that we can use it for evaluating the final estimated locations of the objects, as it provides ground plane coordinates of the tracked pedestrians.

3DMOT is a very popular dataset for evaluating 3D tracking algorithms. Nevertheless, these kinds of multi-object trackers are typically based on a multiple camera setting, and therefore the depth estimation and 3D reconstruction processes are more straightforward and reliable. Therefore, we are not able to provide a fair comparison between our method and other multi-object trackers evaluated on this same dataset. On another note, we will make use of the training subset from *3DMOT* instead of the test subset. This is due to the fact that no annotations are provided for it, as it is used as part of the *MOT Challenge* online benchmark, for which only the metrics results are given to avoid deliberate overfitting of the presented trackers. By using the training set, we are able to run our evaluation procedure offline and with the metrics of our choice. Moreover, most of the other existing datasets for evaluating 3D tracking algorithms have been recorded from moving cars, with the purpose of implementing SLAM (Simultaneous Localization and Mapping) algorithms. Some examples are the *Honda Research Institute 3D Dataset (H3D)*, the *Caltech Pedestrian Detection Benchmark*, the *Berkeley Segmentation Dataset*, or the *Middlebury Stereo Dataset*. The *3DMOT* dataset, on the contrary, provides recordings taken with cameras with static viewpoints, exactly as required for our method.

FIGURE 4.4: Example frames from the selected *3DMOT* and *MOT17 Challenge* videos. From left to right: *TUD-Stadtmitte*, *PETS09*, *MOT17-02*, and *TUD-Crossing*.

We will test both the ground plane locations estimation and the implemented camera self-calibration procedure, on the two annotated videos from the *3DMOT* training set: *TUD-Stadtmitte* and *PETS09* (see Figure 4.4). The former displays a pedestrian walkway recorded from a low viewpoint, where several people pass by in both directions. The latter displays a road intersection, with a car and a van parked at the further visible street, where a sparse group of people comes from several directions and passes by the area.

Additionally, we will also test these methods on the *MOT17-02* and *TUD-Crossing* videos from the more recent *MOT17 Challenge* benchmark (see Figure 4.4). While *TUD-Stadtmitte* and *TUD-Crossing* provide a perspective from a lower camera height, *MOT17-02* gives a higher viewing angle, with *PETS09* giving an even higher one. These combination of heterogeneous camera settings can provide us with a more general overview of the performance of the algorithms. Furthermore, although all of these videos are focused on pedestrian multi-tracking, *TUD-Crossing* displays a scene on a road crossing, where both pedestrians and different types of vehicles are shown. This video also provides a challenging setting, with very high levels of occlusion due to the low camera angle. Overall, this set of videos will give us a better overview of the performance.

The main disadvantage of these videos is that they do not provide camera calibration files nor, in the case of *MOT17-02* and *TUD-Crossing*, ground truth 3D locations annotations. Hence, we will stick to a qualitative analysis of this subset for evaluating both processes of self-calibration and estimation of ground plane locations.

In addition, to quantitatively evaluate the influence of the multi-object tracking algorithm in the ground plane locations estimation, we will again make use of the *TUD-Stadtmitte* sequence. For a further qualitative analysis of the tracking algorithm's performance, we will process and study a video sequence from the *Urban Tracker* dataset (Jodoin, Bilodeau, and Saunier, 2014), named *Sherbrooke*. This recording is particularly useful because of its relative closeness to our expected input video feeds: it consists of street surveillance video, hence taken from static cameras located at high locations, similarly to the *PETS09* sequence. Moreover, it displays a road scene with several cars passing by, which will help evaluate the generalization ability of the tracker's appearance-extracting network to other objects aside from people. While *TUD-Crossing* also provides a scene with vehicles, *Sherbrooke* presents less occlusions thanks to its viewpoint, making the evaluation of the tracking algorithm possible in a more standard setting, while also providing other challenges such as objects further from camera or a low camera quality.

For a quantitative evaluation of the self-calibration algorithm, we will make use of the *WILDTRACK* dataset (Chavdarova et al., 2018) for multi-camera-based pedestrian tracking. This dataset contains recordings from several cameras (with relatively similar intrinsics) of the same scene, each of them with a different viewpoint. Therefore, they enable us to see how the self-calibration process works for different camera viewpoints and with different motion patterns of the people in a single scene.

4.1.3 Annotations

For the ground truth annotations and the predictions, we will make use of two different formats: the *MS COCO* and the *MOT Challenge* annotation formats.

Firstly, we will maintain the *MS COCO* annotations format for the object detection phase. The reason behind this is that the usage of this format will enable us to use the COCO API (Application Programming Interface), which provides us with several options for performance evaluation, annotations and data loading, parsing, and visualization[1].

The *MS COCO* annotations are stored as JSON (JavaScript Object Notation) files. These annotation files contain a header with certain image metadata, which is followed by one data structure per object instance (true or detected) in the image data file(s). Each of the annotated objects has the following composition:

```
annotation{
    "id" : int,
    "image_id" : int,
    "category_id" : int,
    "segmentation" : RLE,
    "area" : float,
    "bbox" : [x,y,width,height],
    "iscrowd" : 0 or 1,
}
```

[1]The official repository of the COCO API can be found here: https://github.com/cocodataset/cocoapi

Where *id* and *image_id* indicate the object and image (or frame) identifiers, respectively. The *category_id* represents the object label as a number, ranging from 1 to 80 corresponding to the *MS COCO* classes, from which we are interested in:

```
"categories": [
    {"supercategory": "person","id": 1,"name": "person"},
    {"supercategory": "vehicle","id": 2,"name": "bicycle"},
    {"supercategory": "vehicle","id": 3,"name": "car"},
    {"supercategory": "vehicle","id": 4,"name": "motorcycle"},
    ...
    {"supercategory": "vehicle","id": 6,"name": "bus"},
    ...
    {"supercategory": "vehicle","id": 8,"name": "truck"}
]
```

Among the rest of the object instance parameters, we are interested in the *segmentation* mask and the bounding box (*bbox*) parameters. The mask is represented as a polygon, defined by the coordinates of each of its corners, which are then compressed using an RLE (run length encoding) algorithm. The bounding box is represented by the *x,y*-coordinates of its upper left corner, and the width and height of the box itself.

In the case of the resulting predictions, each of the predicted objects has the following format:

```
{
    "image_id" : int,
    "category_id" : int,
    "segmentation" : RLE,
    "score" : float,
}
```

Where *score* indicates the confidence level of the prediction.

We implemented the object detection module so that the resulting predictions of the model are formatted following the *MS COCO* standard before storing them. For the sake of evaluation, we implemented a script to automatically convert annotations in the *Cityscapes* annotations format to the *MS COCO* format. This script mainly maps the object label identifiers and converts the binary map representing the masks to the polygonal format in *MS COCO*.

In the case of the depth estimation, we also preserve the annotation format used by the authors of the method. The disparity maps are stored as a .npy file, which is the binary file format used by Python's *NumPy* library for storing and accessing multidimensional arrays. Here, the disparity values of each of the pixels are stored in a matrix matching the original dimensions of the frame images.

For the remainder of the pipeline, the *MOT Challenge* annotation format is used. The *MOT Challenge* annotations consist in a CSV (comma-separated value) type of file, where each row corresponds to a certain object instance within a specific image. This choice of format is justified in principle because our 3D locations evaluation subset contains annotated videos from the *3DMOT* dataset, but also because the DeepSORT object tracker is natively implemented to support detections and predictions on this format. Therefore, the usage of the *MOT Challenge* format enables a more transparent implementation and flow of the rest of the pipeline processes (mainly tracking and ground plane position estimation).

It is nevertheless noteworthy that the *MOT Challenge* benchmark focuses, as its acronym indicates, on multi-object tracking evaluation. It does not focus on multi-object detection itself (even though a detection phase might be required for specific object tracking algorithms). This implies that its annotation format does not include

any fields containing information such as an object's category or its segmentation masks (only bounding boxes are used). Due to this aspect, we will extend the original annotation format with two additional fields per row, storing the object label and its segmentation mask, respectively. Taking this into account, the modified annotation format for each of the rows (corresponding to a certain object instance) looks like this:

```
<frame_number>, <object_id>,
<bbox_left>, <bbox_top>, <bbox_width>, <bbox_height>,
<confidence>, <x>, <y>, <z>,
<object_category>, <segmentation_mask>
```

Where <*frame_number*> and <*object_id*> store the frame and object identifiers, respectively. The parameters <*bbox_left*> and <*bbox_top*> contain the *x*,*y*-coordinates of the upper left corner of the bounding box, whose shape is defined by the width and height values, fields that are followed by the confidence level of the prediction. The fields <*x*>, <*y*> and <*z*> are placeholders for the 3D world coordinates of the object location in the real scene.

To implement this annotation formatting, firstly the *MS COCO* predictions obtained with Mask R-CNN are converted from the *MS COCO* format to the *MOT Challenge* one and stored as a CSV file. At this stage, no <*object_category*> and <*segmentation_mask*> fields exist, and the fields <*object_id*>, <*x*>, <*y*> and <*z*> contain dummy values, as the tracking and ground plane locations estimation processes have not yet been performed. This data is then input to the tracking algorithm, which will output an identically formatted file with the corresponding predicted object ID's. For each of the object instances within the tracking results, its ground plane location will be estimated and stored in its corresponding row, on the <*x*>, <*y*> and <*z*> fields.

4.1.4 Framework

Although some of the pipeline processes could have been jointly run to partially reduce the computation time (such as the object detection and depth estimation, as they are independent), we decided to implement the pipeline in a fully modular fashion. This way, not only the pipeline flow can be more easily abstracted by the user, but it is possible to run each of the modules separately, guaranteeing that a module's running time or possible failure rate does not affect that of the other modules. Moreover, these modules can be more easily modified, improved or substituted with newer versions or alternative methods, providing a better overall scalability of the full method.

The complete pipeline has been developed using Python 3.6. Each of the modules has been implemented as a separate script, which are then called sequentially from the main script.

As there are several modules that correspond to processes based on convolutional neural networks (namely object detection, depth estimation and object tracking), our desired approach is to use a common framework for their implementation, and ideally unify them under the same high-level interface. To achieve this, we make use of the TensorFlow framework (Abadi et al., 2016), a dataflow programming library widely used for scalable deployment of deep learning solutions. Additionally, for a more intuitive and higher level abstraction of the implemented models, we use

Keras (Chollet, 2015), an API for fast experimentation with neural networks capable of running on top of TensorFlow.

For the object detection process, we make use of Matterport's implementation of Mask R-CNN[2]. Although the original authors of the Mask R-CNN method (He, Gkioxari, et al., 2017) actually open sourced their implementation's code[3], Matterport's version is based on TensorFlow (as opposed to Detectron being based on PyTorch) and it is provided as a Python package, hence enabling a more seamless integration with the pipeline. Additionally, it offers a more extensive documentation and flexibility of options (such as integration with the COCO API), making it easier to adapt it to our task.

In the case of depth estimation, we make use of the implementation provided by the original authors of the method[4], as it is already implemented in TensorFlow and it provides a comprehensive documentation for its functions that are of most interest to our pipeline. Likewise, we also integrate the official implementation of DeepSORT published by its authors[5], based on TensorFlow as well. In order to achieve more flexibility, we perform minor changes to their source code, for instance to allow the processing of several images in a way that is transparent to the naming conventions of the files.

Furthermore, to provide a fair comparison of Mask R-CNN against other object detectors, we implemented an alternative version of the object detection module where two other models, in addition to Mask R-CNN, can be loaded: YOLOv3 (Redmon and Farhadi, 2018) and RetinaNet (Lin, Goyal, et al., 2017). As mentioned before, to achieve transparency towards the evaluation script, the versions of the models to be used are all implemented using the same frameworks, concretely Keras as the frontend and TensorFlow as the backend. The original implementation of YOLOv3 was done on a custom C-based framework, called DarkNet, developed by its first author[6], and RetinaNet has no official implementation that is publicly available. Due to this, we choose to use a conversion tool from YOLOv3's DarkNet-based pre-trained weights to Keras with a TensorFlow backend[7], and to integrate Fyzir's implementation of RetinaNet[8] based on Keras and TensorFlow. On another note, we also implement the COCO API library mentioned before, both for the usage of performance evaluation functions and also for other data manipulation functions, such as the compression and uncompression of the binary segmentation masks.

Regarding the camera self-calibration, there is neither an official nor an unofficial implementation of the method -or any similar architectures- publicly available. Therefore, we implemented our modified self-calibration architecture from scratch, and likewise for our scene reconstruction module (comprising the 3D world coordinates estimation and ground plane locations estimation processes).

Within these modules, for all the operations regarding image loading and manipulation, we make use of OpenCV (Bradski and Kaehler, 2013), an open source library for computer vision and machine learning applications. For all numerical

[2]The code repository can be found here: https://github.com/matterport/Mask_RCNN/

[3]The official implementation of Mask R-CNN, named Detectron, was published by its authors and it is available here: https://github.com/facebookresearch/Detectron/

[4]The official code repository can be found here: https://github.com/mrharicot/monodepth

[5]https://github.com/nwojke/deep_sort

[6]The official repository for this framework can be found here: https://github.com/ pjreddie/-darknet, where all the different versions of YOLO can be implemented and run.

[7]This tool can be found at : https://github.com/qqwweee/keras-yolo3

[8]The code repository is available at: https://github.com/fizyr/keras-retinanet

computations involving multidimensional arrays and matrices, the Python libraries NumPy (Van Der Walt, Colbert, and Varoquaux, 2011) and SciPy (Jones, Oliphant, Peterson, et al., 2001) are used. For other machine learning algorithms that are required (such as the Huber linear regression and the mean-shift clustering within the camera self-calibration process), we use Scikit-Learn (Pedregosa et al., 2011), a library for statistical learning, data mining and data analysis.

4.2 Metrics

Due to the nature of the proposed method, privacy can be in principle guaranteed, as only positional information is maintained in the 3D reconstruction and no personal data is transferred to it. Therefore, our main focus during the evaluation process is the assessment of the preserved utility, or intelligibility, of the recorded scene.

4.2.1 Object detection

There are several metrics that are commonly used to evaluate the performance of classification algorithms and particularly object detection:

A multi-class classification problem is fundamentally based on receiving a certain input and predicting a certain category for this input, which could match the ground truth one or not. The cases where the predicted category, or class, matches the true one are called true positives of that specific class. The cases where the input was not predicted as belonging to a certain class and it indeed does not correspond to this class, are called true negatives of the class. On the contrary, a false positive is the case when the input is tagged with this class but this prediction is wrong; likewise, a false negative happens when the input was not tagged with this class but it was actually its true class. In other words, *true* and *false* indicate if the prediction was correct or incorrect, whereas *positive* and *negative* indicate if the class at hand was predicted for the current input or not, respectively.

There are two popular metrics that make use of these concepts. **Precision** aims at assessing how good a model is at identifying only the true cases of a certain class within the dataset (see Equation 4.1). **Recall**, on another hand, aims at measuring how good a model is at finding all the true cases of a certain class within the dataset (see Equation 4.2).

$$Precision = \frac{True\ positives}{True\ positives + false\ positives} \tag{4.1}$$

$$Recall = \frac{True\ positives}{True\ positives + false\ negatives} \tag{4.2}$$

Based solely on their definitions, we can intuitively know that precision and recall are inversely related. In a classification problem, an input typically gets a probability value of belonging to each of the possible classes, and depending on a defined threshold, it will be labeled with a class (or several of them) or not. If this prediction confidence threshold is lowered, the recall will likely increase because there will be

more positives overall; however, this also implies that the number of false positives increases, hence decreasing the precision.

By varying the value of this confidence threshold, we can prioritize either precision or recall, and see how the performance results change for the same model. The different precision-recall pairs can be represented in what is commonly known as the **precision-recall curve**. This curve displays the recall and precision values on the horizontal and vertical axes, respectively, and it allows us to visualize the trade-off between these two metrics. Intuitively, the precision-recall curve measures the performance of a classifier at different threshold settings. The area that this curve occupies, or area under the curve (AUC) of precision-recall, is a popular measure of the performance of a model: it measures how well the model can separate or distinguish between classes, which a higher value indicating a better performance.

An alternative metric to AUC, that is usually considered conceptually equivalent to it, is the **average precision** (AP). To calculate it, the recall axis is divided into several evenly distributed segments. For each of these segments, the maximum precision value is chosen. Average precision is simply the average of all these maximum precision values at evenly spaced recall ranges:

$$AP = \frac{1}{11} \sum_{Recall_i} Max_{Prec}(Recall_i) \tag{4.3}$$

In a multi-class classification problem, the average precision can be calculated for each of the classes, similarly to precision and recall. Nevertheless, the problem of object detection comprises not only a classification task but also a location task. To measure how well the predicted bounding boxes enclose the actual objects, a common metric to use is the **Intersection over Union** (IoU), which indicates how close a predicted box (or alternatively, a segmentation mask) is to the corresponding ground truth one:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{4.4}$$

The performance of modern object detectors is typically measured using the **mean average precision** (mAP), which is computed as the mean average precision value over all the classes, and generally also over different IoU thresholds (by calculating separate precision-recall values for each of them, and obtaining their average precision). The popularity of this metric is due to the fact that it gives a general overview of the method's accuracy, for both detection and localization processes, with a single numerical value. This value can then be easily compared to that of other detectors, facilitating the performance evaluations.

Due to their generalized use in recent research papers, we will compute the mAP metrics defined in the COCO dataset's evaluation server for their benchmark competition. These metrics are the following:

- Average Precision (AP) at multiple IoUs, concretely at 10 IoU threshold values between 0.50 and 0.90 at a step of 0.05; this metric is usually referred to as simply ***AP***

- AP at IoU = 0.50 (referred to as $\boldsymbol{AP_{50}}$), a metric that is typically used in other similar datasets such as PASCAL VOC
- AP at IoU = 0.75 ($\boldsymbol{AP_{75}}$), as a stricter metric for performance evaluation
- AP across small-sized objects ($area < 32px^2$), or $\boldsymbol{AP_S}$
- AP across medium-sized objects ($32px^2 < area < 96px^2$), or $\boldsymbol{AP_M}$
- AP across large-sized objects ($area < 96px^2$), or $\boldsymbol{AP_L}$

These metrics are calculated over the bounding boxes for all the evaluated object detectors. We will calculate the mean average precision metrics over the detected regions and over the 6 object categories of interest described before, applying the same IoU thresholds mentioned above.

We will compare these evaluation results among the pre-trained versions of Mask R-CNN, and YOLOv3 and RetinaNet by using the same testing dataset. These methods have been chosen for the benchmark because they are the best reflectors of the current trends in state-of-the-art object detection: Mask R-CNN follows a region proposal approach and focuses on accuracy, whereas YOLOv3 follows a single shot approach and focuses on speed. While RetinaNet is also a single shot method, its purpose is to find a better balance between speed and accuracy without highly compromising its real-time application. These three approaches are therefore representative of the main ways of tackling object detection that exist nowadays.

4.2.2 Depth estimation and camera self-calibration

As the depth (or disparity) maps are composed of continuous values, the metrics to evaluate them must focus on measuring the numerical differences among the true and predicted depth values. These metrics do not only need to provide the absolute distances between each true-predicted pair, but also how these differences relate to the whole problem, in terms of error rates, throughout the sequence of images to be processed.

The metrics that are generally used to compare this kind of algorithms in recent publications (Eigen, Puhrsch, and Fergus, 2014; Eigen and Fergus, 2015; Laina et al., 2016; Godard, Aodha, and Brostow, 2017), along with qualitative visual inspections of the resulting depth maps, are four: **absolute relative difference**, **squared relative difference**, RMSE (**root-mean-square error**) and RMSElog (**logarithmic root-mean-square error**), which are defined as follows:

$$AbsRel = \frac{1}{T}\sum_{i,j} |y_{i,j} - y^*_{i,j}| / y^*_{i,j} \tag{4.5}$$

$$SqRel = \frac{1}{T}\sum_{i,j} |y_{i,j} - y^*_{i,j}|^2 / y^*_{i,j} \tag{4.6}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{i,j} |y_{i,j} - y^*_{i,j}|^2} \tag{4.7}$$

$$RMSE_{log} = \sqrt{\frac{1}{T}\sum_{i,j} |log\ y_{i,j} - log\ y^*_{i,j}|^2} \tag{4.8}$$

Where y is the predicted depth map of the input image and y^* is its ground truth depth map. The i, j indices refer to the position of each pixel within the depth maps, and T is the total number of pixels. In our case, as our interest lies solely on estimating the depth of the detected objects, we will compute these metrics for, on one hand, the full image; and on the other hand, only for the pixels contained within the predicted segmentation masks of the relevant objects (as the *KITTI* dataset does not include ground truth segmentations).

In addition, we will calculate three **threshold error** metrics that are often used when evaluating depth estimators, $\delta < thr$, where $thr = \{1.25, 1.25^2, 1.25^3\}$ and $\delta = \%$ of y_i such that $max(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}) < thr$. In other words, these metrics reflect the percentage of pixels for which the error ratio when estimating their depth surpasses a certain threshold. Intuitively, the higher the values of these metrics are, the better performance the depth estimator is displaying; whereas for the first four metrics above explained, as they reflect incorrectness, the smaller their value the better. As an extra metric, we will calculate **D1-all**, commonly used for evaluations on the *KITTI* dataset. Its value reflects the percentage of 'bad pixels', or pixels for which the depth estimation was severely far from the true depth. It can be considered as the opposite metric of the threshold errors, as it accounts for the pixels for which the depth error ratio surpasses a certain threshold; particularly for *KITTI*, this threshold is set at a disparity of either 3 pixels or 5% of the real depth in pixels.

The purpose of all these metrics is to provide a statistical overview of the differences between the true and predicted depth values, and therefore to show how accurate the method is at estimating the depth component of the pixels within the images.

For the evaluation of the camera self-calibration process, as we are only aiming at estimating one of the camera parameters (the focal length), we will make use of the same first four error metrics as in depth estimation (**absolute relative difference, squared relative difference, RMSE** and **RMSElog**). Similarly to the purpose of depth estimation, in the case of camera self-calibration we intend to estimate the numerical value of the focal length, hence the four aforementioned error metrics are appropriate for its evaluation.

4.2.3 Ground plane locations estimation and object tracking

Ultimately, the most important part of the pipeline is its final output: the estimated ground plane locations of the objects. The main issue with evaluating these locations, and hence the performance of the full pipeline, is that the 3D reconstruction of a scene and multi-object 3D tracking from a single monocular 2D video is rather uncommon, and no standard metrics or benchmark exist for this challenge.

Taking this into account, our approach is to make use of metrics that are typically used in multi-camera object tracking. Some of the most popular are the CLEAR MOT metrics (Bernardin and Stiefelhagen, 2008), which are for instance used as part of the *MOT Challenge* benchmark (Leal-Taixé et al., 2015).

The CLEAR MOT metrics try to evaluate how well the predicted objects and their locations (either in the 2D image or in the 3D world) match the true ones, plus the

relative levels of wrong predictions (or false positives) and true objects that were not found (or misses). These metrics, therefore, can help us provide a balanced overview of the performance of our pipeline based on two aspects: how precise or accurate the system is at correctly locating the objects on the ground plane, and how robust it is at finding these objects in the first place.

The CLEAR MOT benchmark first defines three categories where the true or predicted objects can be placed: **matches** indicate the predicted objects that do match true ones, **misses** reflect the number of true objects that were not matched, and **false positives** the number of predicted objects that were not matched.

Knowing this, we will make use of four main metrics from CLEAR MOT: **MOTP (Multi-Object Tracking Precision), MOTA (Multi-Object Tracking Accuracy), miss rate** and **false positive rate**, which are defined as follows:

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{4.9}$$

Where d_t^i are the distances, per object per frame, of each matched object to the true one, and c_t is the total number of matches made across all frames of the video. Its unit is therefore a length unit (typically meters), and it essentially indicates the spatial precision that is achieved with an accuracy, or *MOTA*, of:

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t))}{\sum_t g_t} \tag{4.10}$$

Where m_t are the number of misses, fp_t the number of false positives, and mme_t the number of mismatches. The value mme_t considers the situations where there are identity switches. However, as the coherence of the assigned identities to the objects is not of high relevance to this project, we will take this parameter out of consideration. The parameter g_t indicates the total number of true objects across all frames. It is important to point out that *MOTA* is not a usual accuracy metric, and due to its definition it can actually take negative values, which is something to consider during the evaluations.

The **miss rate** is defined as:

$$\bar{m} = \frac{\sum_t m_t}{\sum_t g_t} \tag{4.11}$$

Which represents the ratio of misses m_t over the total number of true objects across all frames. Similarly, the **false positive ratio** is:

$$\bar{fp} = \frac{\sum_t fp_t}{\sum_t g_t} \tag{4.12}$$

The distance metric to calculate d_t^i in *MOTP* (see Equation 4.9) can differ depending on the scenario or application. For the most common 2D multi-object tracking algorithms, the overlap of the true and predicted bounding boxes is generally used, or in other words, the IoU (see Equation 4.4). A pair is typically considered to be

a match if the IoU is higher than 0.5. In 3D tracking, however, this distance metric would not be as adequate. The general approach is to actually calculate the Euclidean distance, on the ground plane, between the estimated 3D location of the object and its annotated 3D location. A common threshold that is used in multi-camera object tracking is 1 meter. As we have repeatedly discussed throughout this report, 3D tracking from a single monocular video is expected to perform with a significantly lower accuracy than its multi-camera-based counterpart. Thus, we will consider a higher initial threshold of 2 meters.

It is important to note that *MOTP* is considered to be a rough performance metric, because of the high influence of the annotations which, in the case of datasets for multi-object tracking, are often ambiguous or inaccurate. An example of this is, for instance, the *TUD-Stadtmitte* video from *3DMOT* (see Section 4.1.2), which has several annotations available that significantly differ among them (with the original ground truth containing no annotations for partially occluded objects). Concerning *MOTA*, its value tends to be significantly affected in crowded scenarios, or when there are several objects that enter or leave the frame: as the number of errors is cumulative throughout the video, this prevents any tracking algorithm (including multi-camera based ones) from reaching an accuracy of 100% or close to it (Milan, Schindler, and Roth, 2013). Knowing these aspects, during the evaluation discussion (see Section 5.4), we consider that it will be more relevant to focus on the evolution of the metric results rather than on their absolute values.

Another metric that is often used to evaluate object trackers is the **F1-score**, the harmonic average of the **precision** and **recall** metrics. In the case of object tracking, precision and recall are conceptually similar to the metrics used in classification systems (like we saw in Equations 4.1 and 4.2), but the definitions of the involved terms match the ones described before. The matches correspond to the true positives, and the misses are the false negatives. The false positives are, as already mentioned, the total number of predicted objects that were not matched.

The F1-score metric is defined as:

$$F1\text{-}score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.13}$$

To evaluate the influence of tracking within the proposed system, we will compare the performance of the full pipeline when using tracking and when not using it. For this, we will use the corresponding aforementioned metrics for evaluating the ground plane locations estimation in both scenarios.

Chapter 5

Results and discussion

In the following chapter, we present and discuss the results of our evaluation processes. We first evaluate each of the predictive modules individually, that is: the object detection, the depth estimation, and the camera self-calibration. We afterwards evaluate the performance of the full pipeline jointly, by analyzing the output ground plane locations relative to both the visual scene itself and the ground truth locations.

5.1 Object detection

We evaluate the object detection architectures by measuring their performance on the same test data: the validation subset of the *Cityscapes* dataset, consisting of 500 images from outdoor steet-like scenes taken from a moving car, which relatively match our expected real application data in terms of content.

With the purpose of evaluating our chosen detection method relative to the state-of-the-art, three pre-trained models are evaluated. These models correspond to three of the most representative object detection architectures in recent research: Mask R-CNN, YOLOv3 and RetinaNet. All of them were trained on the *MS COCO* dataset's *train* subset and all of its 80 object categories. In the case of YOLOv3, we evaluate two pre-trained models made available by its authors: the YOLOv3 originally described in the paper, and YOLOv3 with an SPP layer, an incremental improvement provided by its first author after the publication of the paper.

The resulting values for the evaluation metrics defined in Section 4.2 can be found in Table 5.1. One of the most noticeable characteristics of these results is the apparent superiority of the RetinaNet model, which outperforms all the other models in every metric. This is even more praiseworthy because of the usage of a model trained with a ResNet-50 backbone, a reduced version of the ResNet-101 network described by the authors in the original architecture.

TABLE 5.1: Evaluation results for the tested models on the *Cityscapes* validation subset over the predicted bounding boxes. The mean average precision values are calculated over the corresponding IoUs and over the six class labels of interest (*person* and the selected vehicles).

	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN	ResNet-101-FPN	20.1	32.3	20.9	1.2	16.4	49.2
YOLOv3-608	Darknet-53	14.8	26.3	15.4	0.8	10.5	39.1
YOLOv3-SPP	Darknet-53	14.2	24.6	14.7	1.0	9.0	38.3
RetinaNet	ResNet-50-FPN	22.5	38.0	22.6	2.3	21.2	50.9

While the original YOLOv3 performs better overall than its SPP-based counterpart, it does perform slightly worse for smaller objects. This can be considered as an indicator that an SPP layer could potentially improve the performance of single shot models for detecting smaller or elongated objects, which has typically been one of these models' main weaknesses. In any case, the performance of all the evaluated models for small-sized objects is notably poor, a property that must certainly be taken into account when evaluating the whole system pipeline.

TABLE 5.2: Evaluation results for the tested models on the *MS COCO* test subset over the predicted bounding boxes and over all 80 *MS COCO* classes. These results are taken from the original research papers of Mask R-CNN (He, Gkioxari, et al., 2017), YOLOv3 (Redmon and Farhadi, 2018) and RetinaNet (Lin, Goyal, et al., 2017) and were computed by the *MS COCO* evaluation server for their benchmark contest.

	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
YOLOv3	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9
RetinaNet	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2

Another aspect that is noteworthy is the large deviation of these results in respect to the ones provided in their corresponding research papers, which can be found in Table 5.2. A reasonable cause could be the fact that, as all of the performance evaluations run by the authors were done over the *MS COCO* test subset, the pre-trained models could have actually been tuned for an optimized performance over this specific dataset, consequently producing lower average precision values when tested on other datasets that might differ from *MS COCO*. An example of this is the imbalance of the number of instances per class in the *MS COCO*: according to the authors' paper, the number of *person* instances is close to 1 million, while for most of the vehicles labels, this number is closer to 10,000 (with the *car* class having more instances overall, of around 100,000, which is still far from the *person* class). The *Cityscapes* dataset does contain a high number of vehicle objects, as it mostly portrays on-road scenes, and the lower number of examples of these objects seen by Mask R-CNN during training could be causing the accuracy decrease. Furthermore, *Cityscapes* is considered to be a challenging dataset which, in the words of its authors, is "biased towards busy and cluttered scenes, where many, often highly occluded, objects occur at various scales" (Cordts et al., 2016). The performance over this dataset is therefore expected to be affected by this.

FIGURE 5.1: Example image from *Cityscapes* (left) displaying several people walking; visualization of its ground truth annotations (center), and visualization of the Mask R-CNN output (right).

In spite of the significant variations in value, the actual relative differences among the methods do not considerably diverge from those noticed in our evaluation results. RetinaNet still outperforms the other methods according to the majority of the metrics, except for the average precision at an $IoU = 0.50$: its value is slightly lower than the one indicated for Mask R-CNN, whereas in our evaluation results it was particularly higher in comparison.

FIGURE 5.2: Example image from *Cityscapes* (left) displaying a sidewalk in poor lighting conditions; visualization of its ground truth annotations (center), and visualization of the Mask R-CNN output (right).

It is important to note, nevertheless, that despite the seemingly improved performance of RetinaNet over Mask R-CNN, this single shot architecture does not perform object segmentation, a valuable feature for the full anonymization pipeline proposed in this research. Mask R-CNN offers both this feature and an adequate performance for the task, in fact significantly better than the one achieved by the YOLOv3 single shot models.

FIGURE 5.3: Example image from *Cityscapes* (top) displaying a road scene; visualization of its ground truth annotations (center), and visualization of the Mask R-CNN output (bottom).

Qualitatively speaking, Mask R-CNN trained on *MS COCO* provides a remarkable performance for both people and vehicles. People are detected with notably high levels of confidence, and are meticulously segmented even under sharp occlusions (see Figure 5.1). Likewise, our selected vehicle labels are also correctly identified and distinguished, displaying a methodical distinction between bicycles and

motorcycles, two labels that were intuitively expected to be problematic. An example of this can be seen in Figure 5.2, where the bicycle and motorcycle present are correctly detected and told apart with surprisingly high confidence levels.

Furthermore, in this same Figure 5.2, Mask R-CNN shows a surprisingly good performance under poor lighting conditions. This is even more remarkable since the aforementioned motorcycles are not even annotated in the ground truth, possibly because the authors did not expect detectors to perform well on this area of the image. Similarly, two people who were not annotated were correctly detected and segmented by Mask R-CNN: one over the dark, right area of the picture, and a barely visible one at the background, on the left side.

Likewise, cars are located and segmented with high accuracy and confidence, even under severe occlusions such as in a road stop or traffic jam (see Figure 5.3). With the goal of testing the limits of the method, we provide a scene with extreme conditions, such as critical occlusions among cars. An example is given in Figure 5.4, showing a row of parked cars that is almost perpendicular to the camera. Here, we can see how certain vehicles are not detected, which is nevertheless expected: for instance, only a tire is visible from one of them. The next car in the row might have remained undetected because of its dark and uniform coloring, which has a very low contrast with its already shadowy background.

FIGURE 5.4: Example image from *Cityscapes* (top) displaying a road scene with severe occlusions; visualization of its ground truth annotations (center), and visualization of the Mask R-CNN output (bottom).

Overall, Mask R-CNN provides a state-of-the-art object detection accuracy that complies with the requirements of our proposed method, giving an impressive performance on our object labels of interest in all three aspects of detection, localization and segmentation. Moreover, it displays a fair performance on extreme settings (such as with high levels of occlusion or poor lighting conditions, among others). It is also noteworthy how it is able to detect objects that were not originally annotated,

most probably due to these extreme conditions. This means that the quantitative results might be negatively influenced by predictions that could, in fact, be potentially outperforming the defined ground truth.

5.2 Depth estimation

We evaluate several models of the chosen monocular depth estimation method (from here on, MonoDepth) which were trained and tuned on different well-known depth datasets. The implementation of MonoDepth provided by its original authors provides five of these pre-trained models, which were trained on: *KITTI* (Geiger, Lenz, and Urtasun, 2012), *Cityscapes* (Cordts et al., 2016), the Eigen split (Eigen, Puhrsch, and Fergus, 2014), *Cityscapes* tuned on *KITTI*, and *Cityscapes* tuned on Eigen.

The evaluation is performed on the *KITTI Stereo 2015* subset, which contains 200 stereo image pairs (from which the left images were used) and their true depth maps.

First, we evaluate the performance taking into account all the pixels in the image and their respective depth. The results for this evaluation can be found in Table 5.3. Notably, the obtained results are close to those presented in the original paper, even slightly better in most cases. This can be seen on Table 5.4, which shows the evaluation results obtained by the authors and published in their original paper. The values shown in blue indicate those metrics for which our evaluation results were better than those on the paper, and the ones in red indicate a worsening in our results in respect to the authors' results. The values in black exactly match those in the paper.

TABLE 5.3: Comparison of the different pre-trained models provided by the authors of MonoDepth, tested on the left images from the *KITTI 2015 Stereo 200* training set. All the pixels were taken into account when calculating these values.

Model/Train. data	**AbsRel**	**SqRel**	**RMSE**	**RMSElog**	**D1-all**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
KITTI	0.1171	1.1819	5.815	0.206	30.003	0.848	0.943	0.977
Cityscapes	0.6992	10.0554	14.427	0.542	94.772	0.053	0.327	0.862
Eigen	0.0975	1.1050	5.084	0.173	20.696	0.893	0.962	0.985
City2KITTI	0.0995	0.9299	5.136	0.177	25.046	0.878	0.961	0.986
City2Eigen	0.0806	0.8025	4.581	0.150	16.562	0.921	0.974	0.990

It is noticeable how the authors did not provide evaluation results for the models trained or tuned on the Eigen split, because they precisely used this dataset as the evaluation set. This data subset contains data from both *KITTI* and *NYUDepth*. We decided not to evaluate on the Eigen split as the *NYUDepth* data, mostly containing images of indoor spaces and objects, is relatively far from our expected data, particularly in terms of content. It is nonetheless remarkable that the performance evaluation results generalize in a similar manner for a different dataset.

Furthermore, when the same metrics are calculated considering the pixels belonging to relevant objects only, the majority of the metrics display a slight improvement (see Table 5.5). This shows how the performance of the method is apparently more satisfactory specifically for the areas of the image we are interested in, that is, for the relevant objects such as people and vehicles.

The exception to this is the model trained on *Cityscapes*, which has the worst evaluation results in both scenarios, and whose performance worsened when considering the objects' pixels only. A possible reason for its general performance, which was already noted by the authors of MonoDepth, is that the *Cityscapes* data was recorded on a camera setting that is substantially different from the other datasets in terms of camera parameters, such as the focal length or the distance between the pairs of stereo cameras. This could make the model not generalize well to the unseen test data, which precisely belongs to the *KITTI* dataset hence sharing similar parameters with the other training datasets. This effect seems, nevertheless, to be overcome when the *Cityscapes* model is tuned with any of the other training sets, as these tuned models (named *City2KITTI* and *City2Eigen* in the tables) are exactly the ones displaying the best performance results.

TABLE 5.4: Comparison of the different pre-trained models of MonoDepth, tested on the left images from the *KITTI 2015 Stereo 200* training set. These results are provided by the authors in their original paper (Godard, Aodha, and Brostow, 2017), and were calculated taking all the pixels and their depths into account.

Model/Train. data	**AbsRel**	**SqRel**	**RMSE**	**RMSElog**	**D1-all**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
KITTI	0.124	1.388	6.125	0.217	30.272	0.841	0.936	0.975
Cityscapes	**0.699**	10.060	14.445	**0.542**	94.772	**0.053**	0.327	**0.862**
Eigen	-	-	-	-	-	-	-	-
City2KITTI	0.100	0.934	5.141	0.178	25.077	**0.878**	**0.961**	**0.986**
City2Eigen	-	-	-	-	-	-	-	-

The model that can apparently be considered to be optimal is the one trained on *Cityscapes* and tuned on the Eigen split. It achieves the best performance results for every metric in both comparisons, except for the square relative difference for which *City2KITTI* obtains a slightly better value. Overall, this would make *City2Eigen* the seemingly most appropriate technique to be used within the project pipeline.

TABLE 5.5: Comparison of the different pre-trained models provided by the authors of MonoDepth, tested on the left images from the *KITTI 2015 Stereo 200* training set. Only the pixels belonging to the image areas that Mask R-CNN predicted as relevant objects (from the 6 classes previously defined) were considered for the calculation of these metrics.

Model/Train. data	**AbsRel**	**SqRel**	**RMSE**	**RMSElog**	**D1-all**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
KITTI	0.1082	0.9176	4.039	0.151	23.893	0.891	0.963	0.987
Cityscapes	0.8372	14.3411	16.620	0.608	98.224	0.013	0.122	0.754
Eigen	0.0912	0.7230	3.662	0.129	18.726	0.919	0.978	0.993
City2KITTI	0.0823	0.5110	3.227	0.119	16.404	0.925	0.982	0.995
City2Eigen	0.0759	0.5245	3.151	0.111	13.724	0.946	0.984	0.994

Qualitatively speaking, however, the apparent accuracy of *City2Eigen* strongly depends on the kind of image and camera setting, specifically in terms of the depicted scene and angle of recording. Two of the training datasets used for MonoDepth, *KITTI* and *Cityscapes*, have both been recorded from a driving car, meaning that the height of the camera is relatively low and the objects can be considered to be approximately parallel to the camera plane. The Eigen split contains mainly data from the *NYUDepth* dataset, whose images generally depict day-to-day objects in indoor scenarios, but it also contains data from *KITTI*. In scenes where the camera

height is similar to that one in *KITTI* and *Cityscapes*, MonoDepth -and particularly *City2Eigen*- seems to perform adequately.

Likewise, the depth estimation generally seems to be more accurate with objects that are closer to the camera than those further away. An example of both aspects can be seen when MonoDepth is tested on a frame from one of the *MOT Challenge* videos, concretely the *TUD-Stadtmitte* sequence (see Figure 5.5), for which the model provides a competent depth estimation, especially for the regions displaying people. Surprisingly, the model trained only on *Cityscapes* provides an apparently better depth estimation of the people's shapes, which could be caused by the high incidence of people within the images from the *Cityscapes* training set.

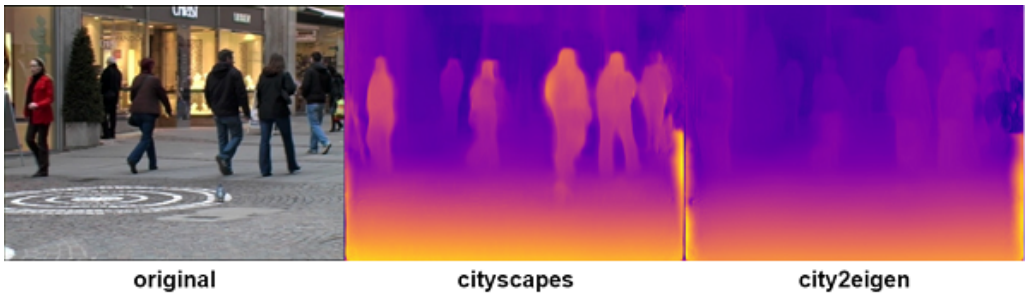

FIGURE 5.5: 2D visualization of the depth map output by MonoDepth's models trained on *Cityscapes* (middle) and *Cityscapes* tuned on Eigen (right), for a frame taken from the *MOT Challenge* benchmark dataset.

These results unfortunately worsen when the input image has been taken from a more elevated location. An example is a second frame from the *MOT Challenge* dataset -this time from a different source video file, named *AVG-TownCentre*- shown in Figure 5.6, which portrays a street scene recorded from what could be a CCTV camera. Moreover, most of the objects of interest are relatively far from the camera, a situation where MonoDepth tends to underperform, as we noted before. Not only the distance of the people to the camera seems to be incorrect in relation to other people, but the predicted depth for pixels belonging to the same person seems to drastically vary.

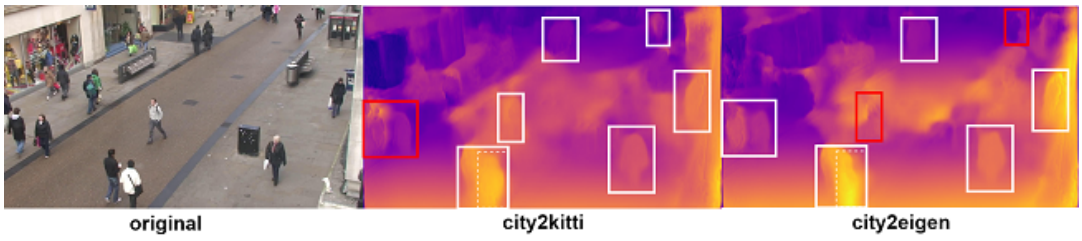

FIGURE 5.6: Depth map visualization of a frame from the *MOT Challenge* benchmark, output by MonoDepth's models trained on *Cityscapes* tuned with *KITTI* (middle) and *Cityscapes* tuned on the Eigen split (right). The locations of the people appearing in the image are indicated with squares, from which the red squares indicate an apparent incorrect depth estimation of the people's shapes.

MonoDepth is, like Mask R-CNN and deepSORT, based on a deep neural network, hence a black box method by nature. Although we are not able to know the specific patterns that these trained models learned, we can speculate on them based on the obtained output. One of the reasons behind these artifacts might be, for example, the contrast and hue of the colors of these objects. In the same Figure 5.6, we

can see how people dressing clothes with more uniform colors in terms of tone and lightness have a more homogeneous depth map. On the contrary, people with highly contrasted clothes display large depth artifacts. An example is the man on the left side of the frame, whose hood (of white color) and black coat have considerable different estimated depths. Dark areas of an image could be potentially influencing the performance of the depth estimation because they can be confused with shadows, which also tend to be problematic with these kinds of methods.

TABLE 5.6: Comparison of the evaluation results for the two main object classes (*person* and *car*), tested on the left images from the *KITTI 2015 Stereo 200* training set using the *City2Eigen* model. Only the pixels belonging to the image areas that Mask R-CNN predicted as each specific object class were considered for the calculation of these metrics.

Model/Train. data	**AbsRel**	**SqRel**	**RMSE**	**RMSElog**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
person	0.1753	2.6646	6.344	0.182	0.771	0.908	0.985
car	0.0775	0.5826	3.190	0.114	0.945	0.981	0.991

In fact, while the evaluation results for the areas belonging to relevant objects seem to improve the general results, if we perform the same evaluations per object class, we can see significant differences depending on the evaluated label. In particular, *person*-labelled objects seem to give dramatically worsened results (see Table 5.6).

FIGURE 5.7: Depth map visualization of a frame from *MOT Challenge*'s *PETS09* video, output by MonoDepth's *City2Eigen* pre-trained model. The people in the frame present a depth map that is notably incorrect and not coherent with the depth estimation of the ground floor at the same location. The ground floor also presents visible artifacts over its whole area.

On another note, the area that seems to give some of the largest artifacts and visibly wrong depth estimations is the ground floor in scenes where the camera is placed higher. We can again see that in Figure 5.6, especially on the center area of the frame. Another example is the output for the *PETS09* video from the *MOT Challenge* dataset, where the whole scene presents significant artifacts, both for the objects and the background, including the ground floor (see Figure 5.7).

Furthermore, one of the main limitations of this depth estimation method, already covered in Section 3.4, is the lack of consideration of the time domain, meaning that the depth maps are predicted individually for each of the frames. This sometimes causes some significant differences in the predicted disparity for the same 2D point in subsequent frames (see Figure 5.8). This unreliability of the depth prediction, plus its variability along time, implies that the depth component of the subsequently computed 3D ground plane location will not only be incorrect but also

unstable along the z-axis, ultimately causing a visible flickering of the 3D ground plane locations computed for a certain object over time.

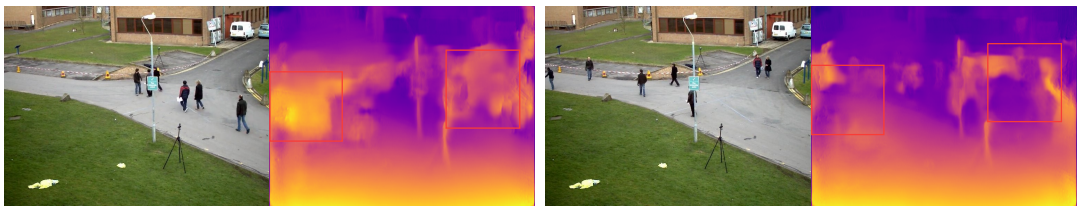

FIGURE 5.8: Depth map visualization of two different frames from 3DMOT Challenge's *PETS09* video, output by MonoDepth's *City2Eigen* pre-trained model. The depth estimation artifacts are not only significant but they also present a high variation over time.

5.3 Camera self-calibration

We execute the camera self-calibration method on the selected *MOT Challenge* videos. As most of these videos do not provide ground truth calibration files (and the few that do, use undefined units), we evaluate them in a qualitative manner. In addition, we provide a quantitative evaluation of the focal length estimation for the *WILDTRACK* sequence, by running the self-calibration procedure on two of its videos, recorded from different cameras (in terms of intrinsics, angle and height).

FIGURE 5.9: Pole estimation process for camera self-calibration: ellipse fitting of reference pedestrian's segmentation mask in *TUD-Stadtmitte* (left), estimated poles during the self-calibration fragment (center), and example of head-to-head and feet-to-feet lines intersection for the estimation of an horizon vanishing point candidate (right).

The first part of the self-calibration process consists in finding a reference object, locating its head and feet within each frame, and calculating the head-feet poles. We provide an example of the visualization of these steps in Figure 5.9 (left and center images), for the *TUD-Stadtmitte* recording.

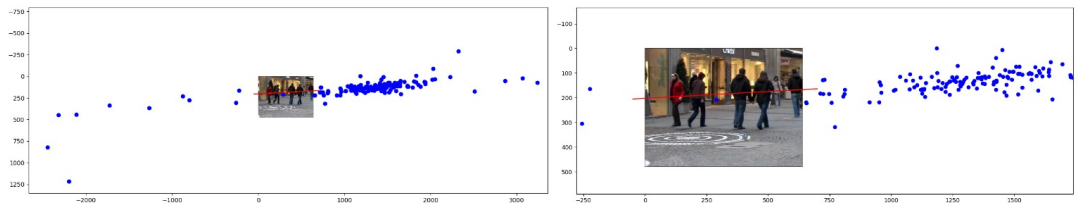

FIGURE 5.10: Horizon line (in red) for *TUD-Stadtmitte* estimated through linear regression applied on the candidate horizontal vanishing points (left; zoomed-in version on the right).

The lines between the heads and feet locations, respectively, are calculated for each pair of poles to find a candidate horizontal vanishing point. An example of the computed lines in *TUD-Stadtmitte* can be seen in Figure 5.9 (right image).

Eventually, linear regression is applied to the computed candidate vanishing points in order to find an estimate of the horizon line. A visualization of the estimated horizon for *TUD-Stadtmitte* is given in Figure 5.10.

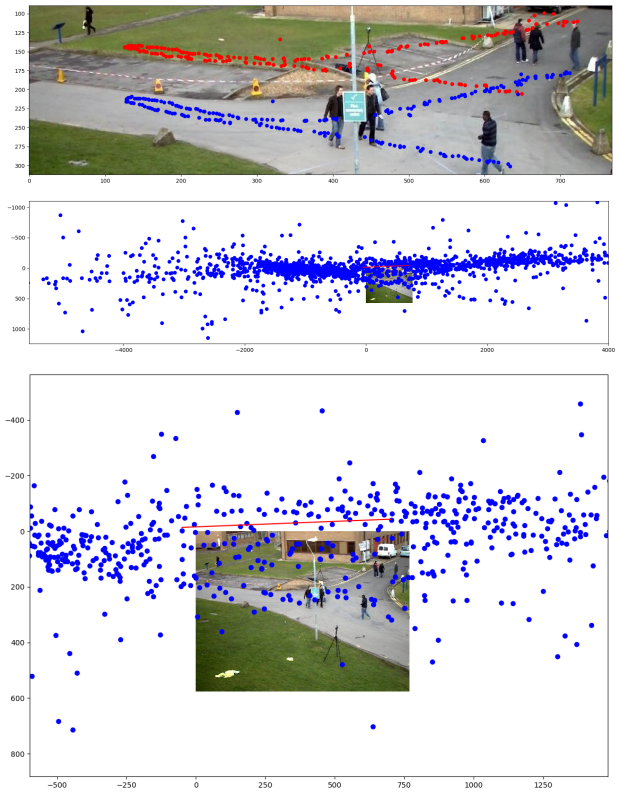

FIGURE 5.11: Horizon line (in red) for *PETS09* estimated through linear regression applied on the candidate horizontal vanishing points (left; zoomed-in version on the right).

In this case of the *TUD-Stadtmitte* video, the horizon line appears as a realistic estimation, and relatively robust to large outliers (such as the ones on the bottom left corner, in the abovementioned Figure 5.10). Nonetheless, it is interesting to see that it is tilted to some extent. This imprecision is to be foreseen in the case of *TUD-Stadtmitte*, because of the nature of the reference object's motion: the person moves from left to right, almost in parallel to the image plane. This means that the computed poles do not provide a thorough view on the three-dimensionality of the scene, producing somewhat less accurate candidate vanishing points.

On the contrary, the reference object within the *PETS09* sequence does move in several directions throughout the video, providing varied candidate points (see top image in Figure 5.11). The number of computed candidate points is also higher than in *TUD-Stadtmitte* because of the longer duration of the video fragment, and their linear-like trend is more visually recognizable (see Figure 5.11, center image). The final horizon line appears to be tilted again, in a similar manner to *TUD-Stadtmitte*. This time, however, it is likely caused by a characteristic of the scene background: the path leading to the parked vehicles is on a slope. This implies that the assumption of a flat ground plane does not fully apply, and the poles will not always reflect parallelism in the real world. This turns out into a possibly higher number of outliers which, in the case of the horizon line, will generate a bias on its inclination.

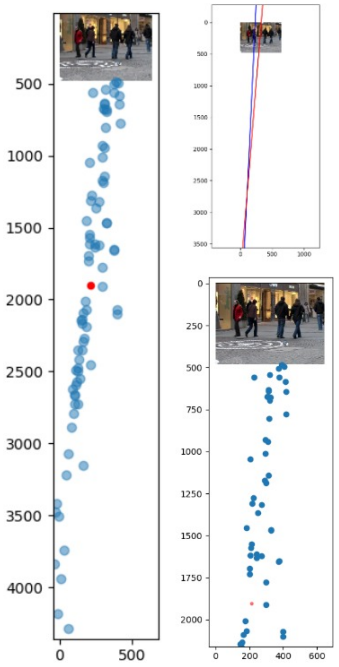

FIGURE 5.12: Vertical vanishing point (in red) for *TUD-Stadtmitte*, estimated through mean shift clustering applied on the candidate vertical vanishing points (left; zoomed-in on the bottom-right). Example of candidate vanishing point obtained by intersecting two elongated poles (top-right).

After the horizon line is estimated, the vertical vanishing point is computed. Several outliers are again to be expected, which is why a technique like mean shift clustering is used to find the final estimate. However, to reduce the number of outliers, we also filter out all those candidate points that are contained within the image area or above, as we already know they are erroneous.

It is interesting to see that in *TUD-Stadtmitte*, the candidate vertical vanishing points gather in a linearly-shaped group (see Figure 5.12). This pattern seems to indicate that the line perpendicular to the ground plane where the vanishing point is located is approximately known, but the lack of knowledge about the camera height introduces uncertainty. Hence, the mean shift cluster will tend to locate the vertical vanishing point around the middle area of this group of points.

The case of *PETS09* is slightly different, as most of the candidate points seem to be concentrated on the upper part, relatively closer to the image area. This ultimately leads to a cluster center (that is, the vertical vanishing point) closer to the area with the highest concentration of candidate points.

It is noteworthy, however, that the difference in the y-coordinate between *TUD-Stadtmitte*'s and *PETS09*'s vertical vanishing points intuitively reflects the field of view of each of the utilized cameras. The *PETS09* sequence was recorded from a camera height typically associated with surveillance cameras, which also tend to have wide-angle lenses, to be able to capture a broader scope of the scene. Therefore, its vertical vanishing point is expected to be closer to the scene's ground plane than in *TUD-Stadtmitte*, where the lens is seemingly closer to a standard one (namely, with a higher focal length value). We encounter this exact situation when comparing the estimated vertical vanishing points for these two videos (see Figures 5.12 and 5.13).

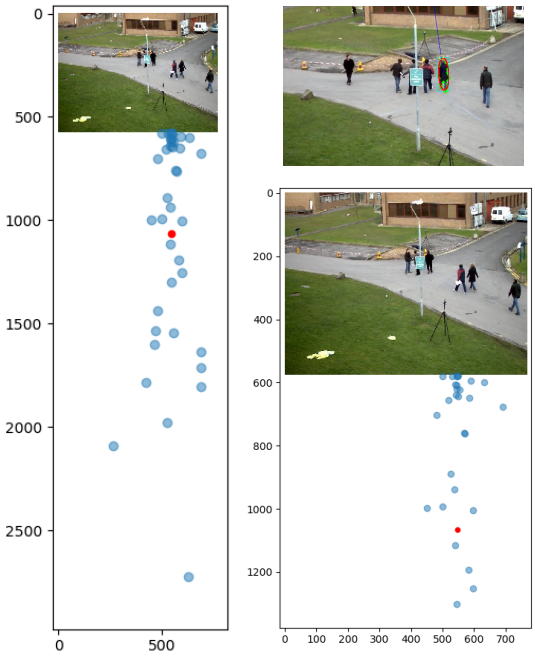

FIGURE 5.13: Vertical vanishing point (in red) for *PETS09*, estimated from the candidate vertical vanishing points (left, zoomed-in on the bottom-right). Example of ellipse fitting and head-feet locations estimation for reference object (top-right).

These sequences from the *3DMOT* pose several challenges in terms of the background scene or the pedestrians as potential reference objects. As an additional example, we also provide a visualization of the calibration results for the two additional *MOT Challenge* sequences: *MOT17-06* and *TUD-Crossing*. In both cases, the results are subjectively satisfactory (see Figures 5.15 and 5.16), with an estimated horizon line that closely matches what a viewer would manually define.

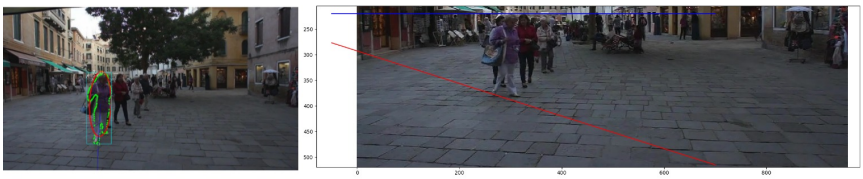

FIGURE 5.14: Pole estimation process for camera self-calibration: ellipse fitting of reference pedestrian's segmentation mask in *MOT17-06* (left), and example of head-to-head and feet-to-feet lines intersection for the estimation of an horizon vanishing point candidate (right).

Particularly for the case of *MOT17-06*, there are several pedestrians that move around the 3D space without a linear path, which is especially useful for their usage as reference objects. The selected reference pedestrian in this case moves towards the camera during the first half of the video, and to the right side of the frame in the second half, almost in parallel to the image plane (see Figure 5.14). This provides us with several relevant poles to calculate the candidate vanishing points, ultimately yielding a precise estimate of the horizon location (see Figure 5.15).

As these video sequences are not provided with camera calibration files, we focus on a different set of videos for evaluating the estimated focal length values. We

process two recordings from the *WILDTRACK* dataset (Chavdarova et al., 2018). An interesting aspect of these videos is that they provide different viewpoints of the same scene, hence enabling the analysis of the performance of the self-calibration process in different settings, and with different viewing angles of the same potential reference objects. Moreover, the chosen recordings were captured with different camera models (a GoPro Hero 3 and a Hero 4), which can also reflect the performance variability depending on the original camera parameters.

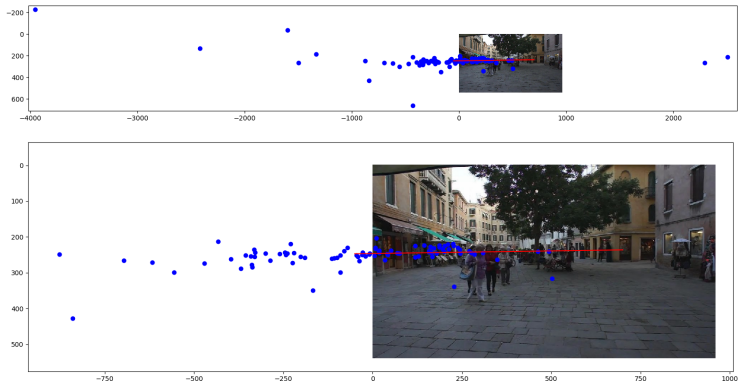

FIGURE 5.15: Horizon line (in red) for *MOT17-06* estimated through linear regression applied on the candidate horizontal vanishing points (top; zoomed-in version on the bottom).

It is important to note, however, that these videos come with a disadvantage: their image frames are significantly distorted, while one of our assumptions is that they are expected to have low to no distortion. Although this would be an unsafe choice of evaluation data regarding the ground plane locations estimation, it offers an additional challenging scenario to test the self-calibration process.

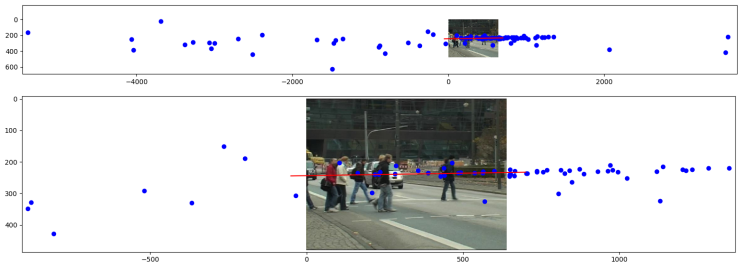

FIGURE 5.16: Horizon line (in red) for *TUD-Crossing*, estimated through linear regression applied on the candidate horizontal vanishing points (top; zoomed-in version on the bottom).

We show the estimated horizon line for the video recorded from camera 5 in Figure 5.17. It is visibly tilted, which is expected considering the wide angle distortion of the image. On this aspect, it is interesting to see that it actually aligns with part of the stairway in the background, which is severely curved due to the distortion; yet, it approximates its expected location, considering the direction of the drawn lines on the scene's floor and their expected points of intersection.

The estimated horizon line for the recording taken with camera 1 is shown in Figure 5.18. This video displays a particular case where the camera is already not fully parallel to the ground plane. The slant of the horizon line therefore correctly

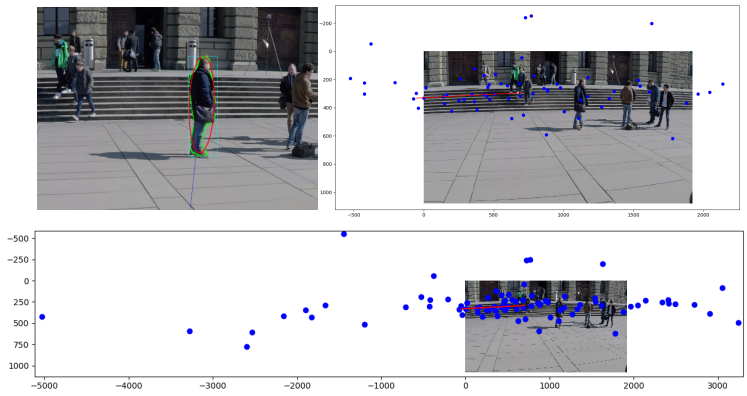

FIGURE 5.17: Horizon line (in red) for the *WILDTRACK* sequence recorded from camera 5, estimated through linear regression applied on the candidate horizontal vanishing points (top-right; zoomed-in version on the bottom). Example of ellipse-fitted reference object for head-feet localization (top-left).

corresponds to the perceived camera one, which can be seen through its similarity to the drawn lines on the floor (that is, the ones that are perpendicular to the stairs).

The error rates for the analyzed *WILDTRACK* videos are given in Table 5.7. The resulting error distances are overall on the low end, especially considering the scale of the true focal length values, which are close to $2000px$. An RMSE in the range of 0 to 20 can be seen as negligible when compared to that scale, because the consequent errors during the estimation of the 3D world coordinates and ground plane locations are expected to be within the defined matching threshold ranges (see Section 4.2).

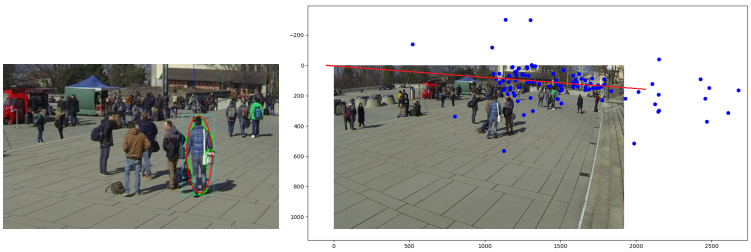

FIGURE 5.18: Horizon line (in red) for the *WILDTRACK* sequence recorded from camera 1, estimated through linear regression applied on the candidate horizontal vanishing points (right). Example of ellipse-fitted reference object for head-feet localization (left).

On another note, it is interesting to see that the error rates are slightly lower for camera 1 than for camera 5. This could be influenced by their difference in viewpoints. Camera 1 displays a low camera height, thus the occlusions among the pedestrians are expected to be larger than for camera 5, which is located at a higher angle. The occlusions can prevent the algorithm from obtaining useful poles, for instance by hiding head and feet locations, which could also eventually provoke noise and outliers among the estimated poles.

These results are particularly promising, considering the limitations of the self-calibration method caused by its simplification, plus the method's dependence on the content of the scene. It is important to note that the selection of an appropriate video fragment is critical for the performance of the self-calibration process, as seen

TABLE 5.7: Performance evaluation results of the camera self-calibration process, for two recordings of the *WILDTRACK* sequence.

WILDTRACK sequence	**true f**	**pred. f**	**AbsRel**	**SqRel**	**RMSE**	**RMSElog**
cam1	1743.448	1807.773	0.037	2.373	8.020	0.036
cam5	1708.657	1586.381	0.072	8.750	11.058	0.074
combined			0.054	5.562	9.884	0.058

before when studying the *MOT Challenge* videos. Due to its nature, this technique heavily relies on the calculation of head-feet poles that are well distributed across the 3D scene, and that is made possible by a reference object which displays a varied motion path throughout the video sequence.

5.4 Ground plane locations estimation

The main and last task of the full pipeline is to output and store the estimated ground plane locations of the relevant objects within the video scene. These locations, along with their object label, can be visualized in an animated sequence of grids representing the ground plane, which already gives us an idea of what is taking place in the scene (see Figure 5.19). Ultimately, these locations could also be used to recreate the scene in a 3D environment, where the object can be represented with default models (see Figure 5.20).

As explained in Section 4.1.2, we make use of the *3DMOT* training subset from the *MOT Challenge* benchmark as the evaluation data. The video sequences contained in this subset, *TUD-Stadtmitte* and *PETS09*, are paired with annotations containing the 3D locations of the recorded objects, which we can compare our estimates to.

The *3DMOT* subset is, in fact, one of the very few existing datasets containing 3D locations of the objects. These locations were originally computed from a multi-camera setting with overlapping camera views, and both *TUD-Stadtmitte* and *PETS09* constitute some of the most extensively used ground truth datasets with this kind of setup (Chavdarova et al., 2018).

Nevertheless, both sequences possess certain limitations that must be taken into account when evaluating the performance results. Firstly, as mentioned above, the objects' 3D locations are not directly known but they are implicitly calculated from the multi-camera arrangement. This is a noteworthy aspect, as the final annotations for both videos could have therefore been influenced by noise and imprecision during their computation.

Regarding each of the videos individually, there are certain characteristics of the scene and its contents that add new challenges, not only to the original purpose of the *MOT Challenge* benchmark (that is, multi-object tracking) but also to our pipeline. In the case of *PETS09*, as mentioned before in Section 5.3, part of the ground plane is composed of a slope, which makes even the ground truth calibration produce inconsistent mappings of the 3D points across all 2D images (Chavdarova et al., 2018).

The *TUD-Stadtmitte* video, on the contrary, does provide a scene with a ground plane that can be approximated to a flat surface. It provides, however, a very low

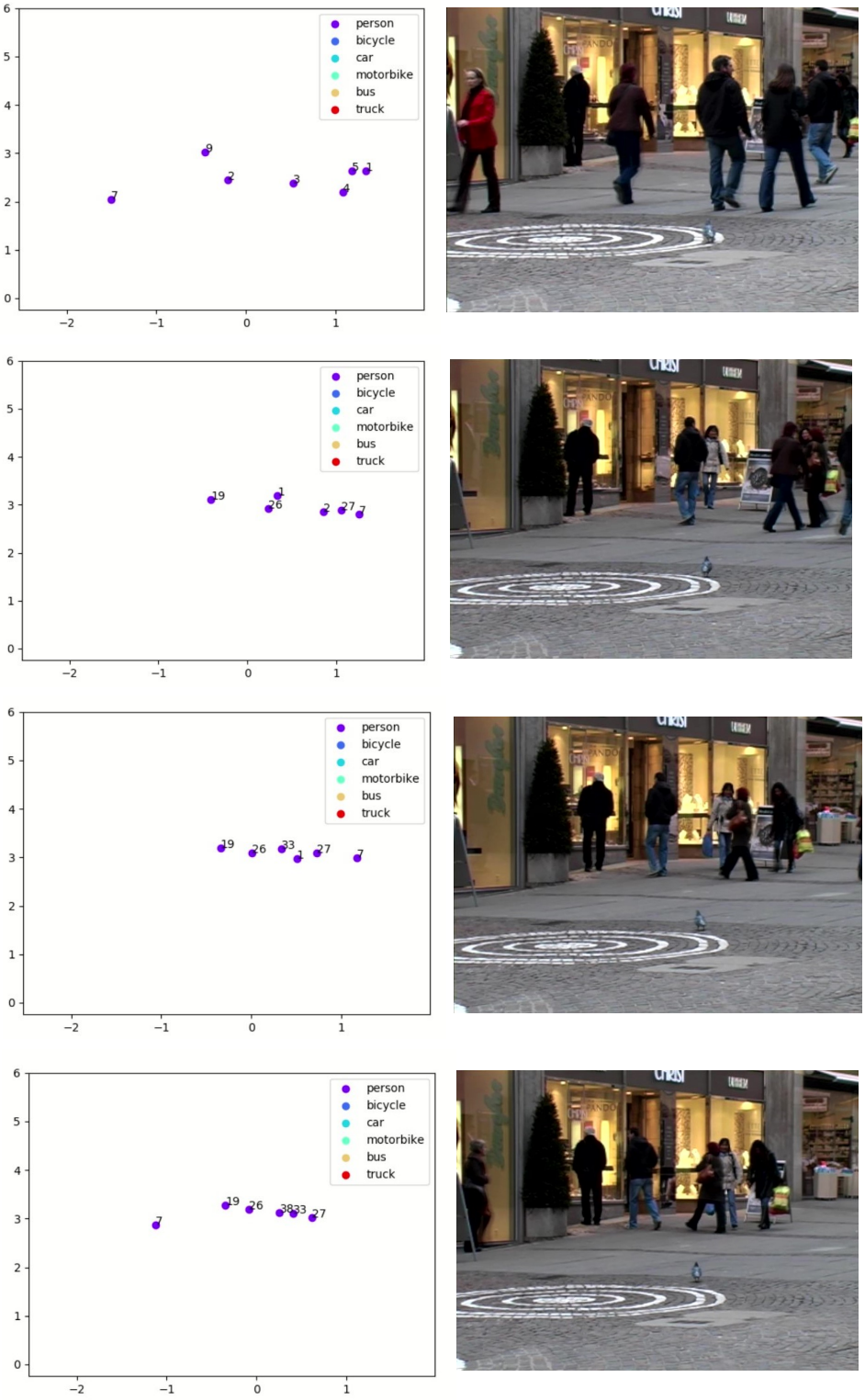

FIGURE 5.19: Examples of the estimated locations of the pedestrians in *TUD-Stadtmitte*, at different frames of the video sequence. These plots represent the ground plane as a 2D grid, where the camera is located at $x = 0$.

camera angle, and it displays severe occlusions among objects. These aspects make the estimation of the objects' ground plane positions substantially difficult (Andriyenko and Schindler, 2011). Consequently, this video sequence is often used for

FIGURE 5.20: Examples of the estimated locations of the pedestrians in *TUD-Stadtmitte*, at a certain frame of the video sequence, in a 3D environment. All the objects with the same label (in this case, *person*) are represented with the same default 3D model, ensuring anonymity.

object tracking evaluation on the 2D image space only, instead of on the 3D space (Andriyenko, Schindler, and Roth, 2012). These are, again, limitations that we need to take into account when analyzing the performance of our pipeline.

During the 3D world coordinates calculation, the camera model determines that the camera's optical axis is located at $x = 0$. Hence, the ground plane locations output by our pipeline are also mapped this way. Conversely, the ground truth locations on the ground plane provided for *3DMOT* do not follow this precept, as they are computed from a multi-camera setting. To allow a fair comparison, both sets of 3D locations are scaled so that their ground plane areas match each other. We do so by rotating, translating and resizing the (x, y)-coordinates so as to locate the camera

axis at $x = 0$ and preserve the distance units from the ground truth (in meters). Moreover, as the *MOT Challenge* benchmark only considers people tracking, only the predicted objects labelled as *person* are considered for the evaluation (effectively discarding, for instance, the predictions related to the two vehicles that appear in the *PETS09* sequence). The scaled true and predicted 3D locations of the objects in *TUD-Stadtmitte* can be seen in Figure 5.21, and likewise for *PETS09* in Figure 5.22.

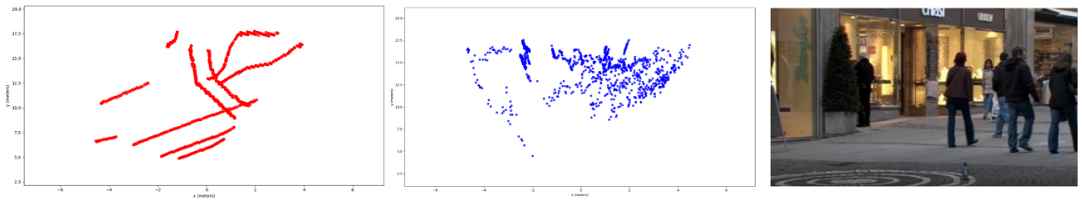

FIGURE 5.21: Ground truth locations/trajectories of pedestrians in *TUD-Stadtmitte* (left), estimated locations/trajectories (center), and example frame from the *TUD-Stadtmitte* video (right). These plots represent the ground plane as a 2D grid, where the camera is located at $x = 0$ and all the positions over the full duration of the video are displayed. All the estimated positions throughout the whole video, for all the tracked objects, are shown.

As a first step, following the common practice when evaluating multi-object trackers, we measure the performance by setting a single specific matching threshold. As explained in Section 4.2, while the most commonly used threshold is $1m$, this is done in multi-camera settings, or settings when more precise measurements can be achieved (such as by using extra sensors). Therefore, for this situation, we choose a threshold of $2m$, for which we provide the results in Table 5.8. Although this is still a restrictive threshold for the proposed pipeline, particularly considering the performance bottleneck caused by MonoDepth, it will give an initial overview on the obtained estimations.

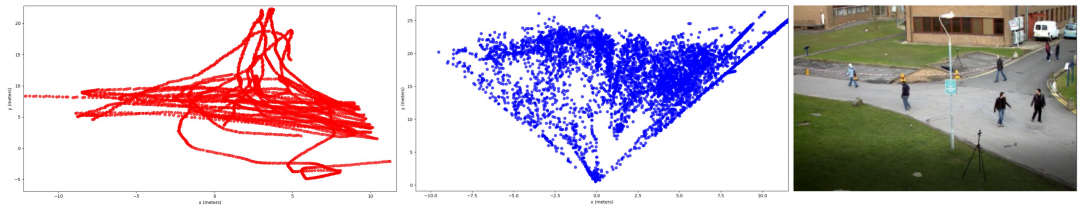

FIGURE 5.22: Ground truth locations/trajectories of pedestrians in *PETS09* (left), estimated locations/trajectories (center), and example frame from the *PETS09* video (right). These plots represent the ground plane as a 2D grid, where the camera is located at $x = 0$ and all the positions over the full duration of the video are displayed. All the estimated positions throughout the whole video, for all the tracked objects, are shown.

We can see that there are sharp differences in performance between the two test videos. While the results for *TUD-Stadtmitte* are definitely promising, those of *PETS09* are drastically weaker. This is, in fact, something to be expected for several reasons, mainly involving the already known limited performance of the depth estimation method:

Firstly, although *TUD-Stadtmitte* provides an overall challenging scene for evaluation, its low camera angle is a commonality shared with the training data used for the monocular depth estimation. As seen in Section 5.2, this depth estimation method can only ensure an adequate performance when the testing data is close to

the one seen during training. Furthermore, in this same Section 5.2, we could see how the estimated depth maps for *TUD-Stadtmitte* were significantly more reliable than those of *PETS09*, which displayed several artifacts with high variability. This influences, therefore, the accuracy of the ground plane locations: with the high number of outliers, less matches are obtained and the number of false positives increases.

TABLE 5.8: Performance evaluation results of the ground plane locations estimations, for the two *3DMOT* training subset videos (*TUD-Stadtmitte* and *PETS09*).

th = 2.0	**MOTP**	**MOTA**	**miss ratio**	**FP ratio**	**precision**	**recall**	**F1-score**	**match/miss**
TUD-St.	1.116	0.177	0.541	0.282	0.696	0.545	0.611	1.197
PETS09	1.990	-0.711	0.867	0.844	0.183	0.180	0.181	0.220

In the case of *TUD-Stadtmitte*, we can see that the results are encouraging, even with a restrictive threshold. An F1-score value of 0.61 indicates a relevant accuracy, considering that both the object detection among 6 classes and the object tracking processes are evaluated. The match/miss score of 1.196 indicates a higher incidence of matches over misses, although the miss ratio (0.54) shows that around half of the true objects have not been matched over the ground plane. The combination of the MOTP and MOTA values point out that objects can be matched approximately 18% of the time with a precision of 1.12 meters. Again, although this accuracy is relatively low, it is confined to a relatively high precision, considering the wide dimensions of the 3D scene.

The *PETS09* sequence offers a different situation, where the incidence of outliers drastically worsens the results. The number of matches is much lower than the number of misses and false positives, causing the ratios of these last two to reach exceedingly high values. The main aspect that these results point out is that the system cannot be accurate with the required precision determined by a threshold of 2 meters (note that, due to the definition of the MOTA metric, it can reach negative values if the number of misses and false positives is high).

TABLE 5.9: Performance evaluation results of the ground plane locations estimations, for the two *3DMOT* training subset videos (*TUD-Stadtmitte* and *PETS09*) considering the x-axis only.

th = 2.0	**MOTP**	**MOTA**	**miss ratio**	**FP ratio**	**precision**	**recall**	**F1-score**	**Match/miss**
TUD-St.	0.430	0.661	0.332	0.007	0.993	0.735	0.845	2.780
PETS09	0.677	0.382	0.453	0.165	0.840	0.657	0.737	1.192

Nonetheless, to measure how the depth estimation is negatively affecting these results, we run the same evaluations, with the same match threshold, but this time considering the x-coordinate only. In this scenario, there is a remarkable performance improvement for both *3DMOT* video sequences (see Table 5.9). In the case of *TUD-Stadtmitte*, a multi-object tracking accuracy of 66% can be achieved with a precision of 0.43 meters in respect to the ground truth. Moreover, not only the number of matches is almost 3 times the number of misses, but the ratio of false positives is low. This indicates how, although a number of true objects were missed in the predictions, the majority of the predicted objects were correctly matched to true ones. This same output is reflected in the notably high precision value and the high recall.

In the case of *PETS09*, the results display an even higher increment. A multi-object tracking accuracy of 38% is achieved at a precision of 0.68 meters, falling not far behind from the performance on *TUD-Stadtmitte*. While the number of misses

is still proportionally high, the ratio of false positives is again positively low. Both sets of results illustrate the performance bottleneck that the depth estimation causes in the pipeline, particularly for recorded scenes that strongly differ from those seen during the training of the aforementioned method. These differences can occur, for instance, in terms of viewpoint or camera parameters: while *TUD-Stadtmitte* shares a similar, low viewpoint with that of datasets like *Cityscapes* or *KITTI*, the *PETS09* sequence was recorded from a very high camera angle. Moreover, the people displayed are located much closer to the camera in *TUD-Stadtmitte*, which could be facilitating its distinction from the background for the depth estimation network.

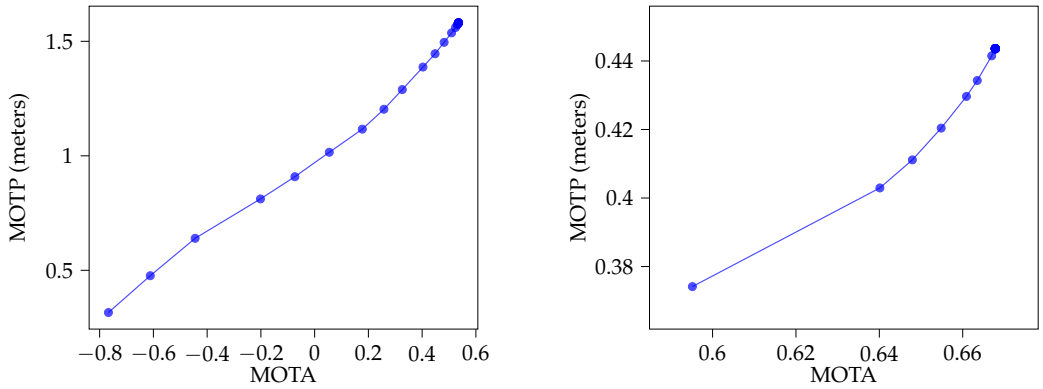

FIGURE 5.23: Multi-object tracking accuracy-precision curve for *TUD-Stadtmitte*, considering both (x, y)-coordinates (left) and x-coordinate only (right).

A more insightful way of evaluating the pipeline can be to see how it performs at several different threshold values. We show these results in Figures 5.23 and 5.24, for *TUD-Stadtmitte* and *PETS09*, respectively, for the two scenarios above studied (considering both coordinates and the x-coordinate only).

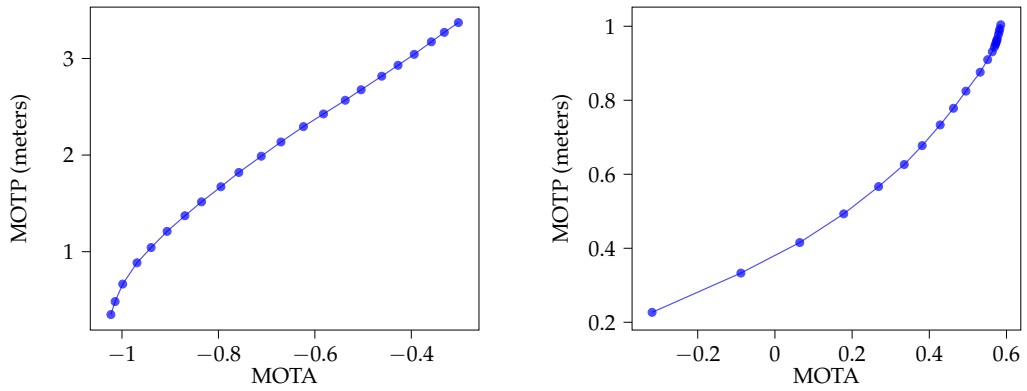

FIGURE 5.24: Multi-object tracking accuracy-precision curve for *PETS09*, considering both (x, y)-coordinates (left) and x-coordinate only (right).

In the case of *TUD-Stadtmitte*, the performance starts becoming acceptable with an increasing precision of $1m$, reaching satisfactory levels of accuracy (around 60%) with a multi-object tracking precision of approximately $1.5m$ (see Figure 5.23). These results are, again, notably improved when only the x-coordinate is considered, already displaying a good accuracy for strict precision values. The pipeline achieves a minimum accuracy of almost 60% with an approximate precision of $0.35m$, which raises until almost 70% still with a solid precision of $0.45m$.

As expected, the performance on *PETS09* sharply deteriorates in relation to *TUD-Stadtmitte*. This is particularly true when both (x, y)-coordinates are considered, as the accuracy is zero or negative within the range of acceptable MOTP values. On the contrary, these results radically improve when only the x-coordinate is taken into account, reaching an accuracy of 60% with a robust precision of 1 meter. This extreme difference between the two scenarios, in addition to an also significant difference for *TUD-Stadtmitte*, again demonstrates the overall high error rate of the depth estimation process. As mentioned before, this is aggravated when the recorded scene highly differs from the ones seen during the training process of the depth estimator.

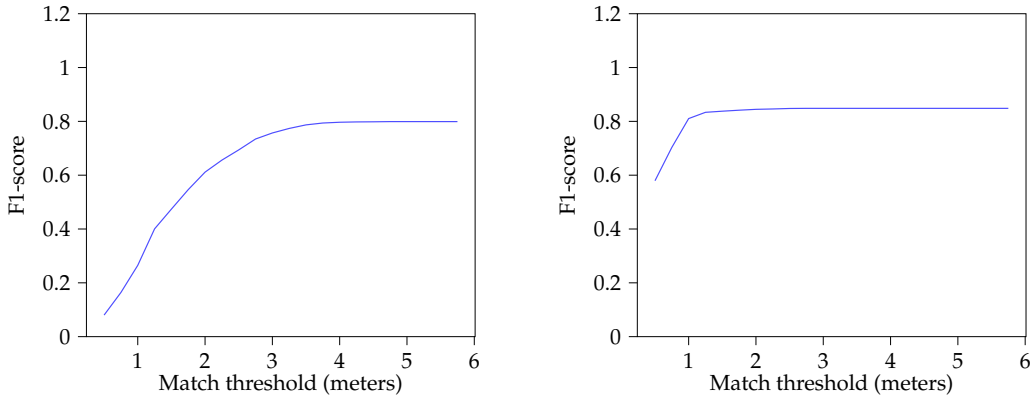

FIGURE 5.25: F1-score curve for the *TUD-Stadtmitte* sequences, with varying 3D location match threshold, considering both (x, y)-coordinates (left) and x-coordinate only (right).

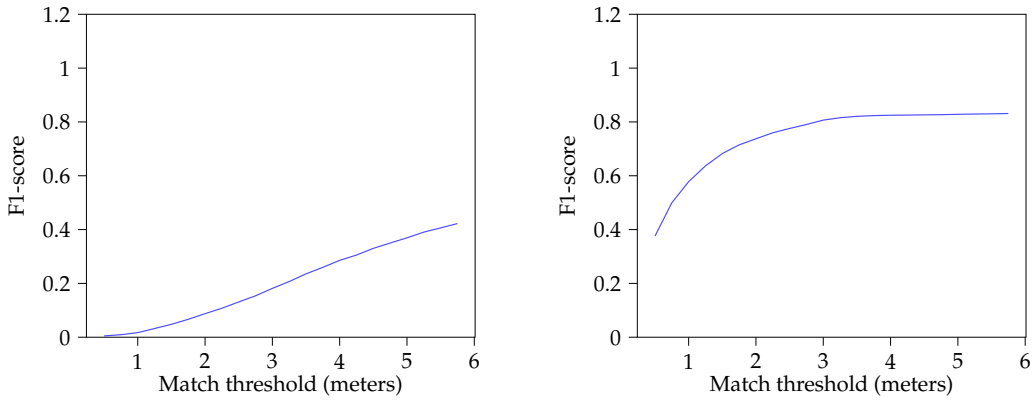

FIGURE 5.26: F1-score curve for the *PETS09* sequences, with varying 3D location match threshold, considering both (x, y)-coordinates (left) and x-coordinate only (right).

To provide an additional overview of the pipeline's accuracy, we make use of the F1-score metric. Similarly to the MOTA-MOTP curve, we also analyze its value relative to the matching threshold. We study the evolution of the F1-score within a threshold range going from 0.5 to 6 meters, for both *3DMOT* videos and for the two scenarios above considered (both coordinates and x-coordinate only).

In *TUD-Stadtmitte*, an already significant F1-score of 0.6 is already achieved at a threshold of $2m$, with its value converging to 0.8 when the threshold approximates $3m$ (see Figure 5.25). This high F1-score is reached much sooner when only the x-coordinate is considered, at a threshold of just $1m$.

A revealing aspect of the analysis of the F1-score is that, surprisingly, this metric shows a similar trend for the *PETS09* sequence when considering the x-coordinate only (see Figure 5.26). Not only the curve converges at an F1-score of 0.8, but this value is also reached at a threshold close to $3m$. This is, however, not the case when both coordinates are considered: as expected, a tolerable F1-score value is only achieved at very high threshold ranges.

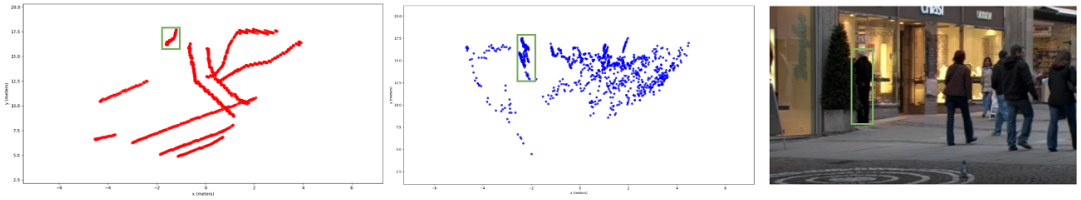

FIGURE 5.27: Ground truth (left) and estimated (center) locations/trajectories of pedestrians in *TUD-Stadtmitte*. The green squares correspond to the image and locations of the only pedestrian that stays still during the whole video.

While these metrics give us a general overview of the performance of the pipeline, it is also interesting to see how the resulting 3D locations relate to the real scene from a qualitative perspective. Moreover, we know the ground truth annotations of the *3DMOT* videos are not entirely reliable, as they were likewise indirectly computed from other data sources. Due to this, we deem it appropriate to additionally study and compare the annotated ground plane locations, against both the recorded scene and the predicted locations. To do so, we analyze the resulting locations on the *TUD-Stadtmitte* sequence. Following this approach, we also intend to focus on those particular cases within the video scenes that yield special challenges to the pipeline and that could be the main causes of a decay in performance.

The *TUD-Stadtmitte* sequence displays an outdoors scene of people walking through a street. These people appear at different distance ranges from the camera, and most of them enter or leave the frame at some point during the recording. Opposite to this, there is a person who stays still throughout the full duration of the video. His location on the ground plane is therefore supposed to be constant, ideally, or at least restricted to a small region on the estimated ground plane grid, during the whole sequence. While its predicted locations display visible outliers in the y-axis, varying along a range of roughly 5 meters, they are all still delimited within a precise range in the x-axis of approximately 0.5 meters (see Figure 5.27, center image). Nonetheless, a noteworthy aspect is that there is also a strong flickering in the ground truth annotated locations corresponding to this person, along both (x, y)-axes (see Figure 5.27, left image). Even if these locations present less outliers depth-wise, their range of variation on this axis is still notable, approximately reaching 2 meters. More surprisingly, their variation on the x-axis is similar to that of the predicted locations.

One of the challenges that these videos offer is the appearance and disappearance of relevant objects in the scene, by either entering or leaving the frame. As we saw before, depth estimation maps often present large outliers on the sides of the image, especially when there are moving objects or partially occluded elements on those areas. Objects coming in or out are thus expected to cause considerable outliers among their predicted 3D locations. This is the case in *TUD-Stadtmitte*, particularly for a large object (that is, close to the camera) leaving the frame: when the object is half-visible, its estimated location starts to flicker along the limit of the camera's field of view, until the object leaves completely (see Figure 5.28). It is also noticeable

FIGURE 5.28: Ground truth (left) and estimated (center) locations/trajectories of pedestrians in *TUD-Stadtmitte*. The green and yellow squares correspond to the image and locations of a pedestrian that enters the frame, and a pedestrian that leaves the frame, respectively.

that, for instance, the outliers are not as severe for a different object, which this time enters the frame (same Figure 5.28). Although the results are similar in terms of visible outliers along the camera's delimited field of view, the variation in depth is not as intense. The estimated 3D locations of this object soon reflect the object's motion pattern and direction in the real scene, without relatively strong variations (roughly along a range of $2m$).

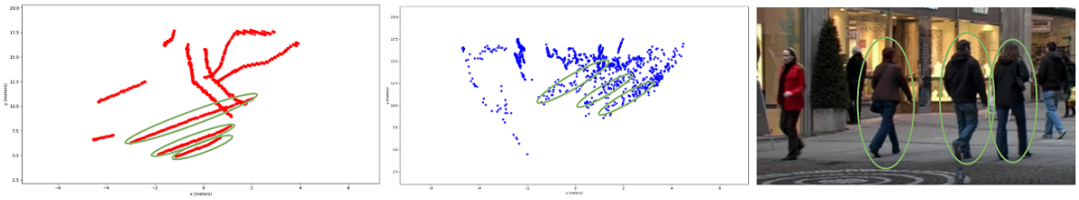

FIGURE 5.29: Ground truth (left) and estimated (center) locations/trajectories of pedestrians in *TUD-Stadtmitte*. The three ellipses indicate the trajectories of three people that move from the centre of the frame and leave the frame on the right side.

In addition to these particular situations, we are also interested in seeing how the objects' trajectories are estimated overall, when these objects walk through the scene within the frame. An example is given by the three people that are the most present in the video, who walk together from one side of the frame to the other, and diagonally to the image plane in the 3D space (see Figure 5.29). In this scenario, despite the depth outliers, their trajectories are still recognizable among the predicted locations. The estimated trajectories display a realistic representation of the real world trajectories in terms of direction and spatial relationships among the three objects. Note that, nevertheless, the estimated distances tend to be apparently more compressed the further the objects are from the camera. As an example, if we focus on the predicted locations, these three objects seem to be closer to the still person on the background than they actually are in the real scene. This is, in fact, an expected aftermath of depth estimation from stereo video -which also applies in this case, as MonoDepth tries to predict the stereo paired right-image of the input image. Depending on the baseline distance between the cameras, for very distant objects

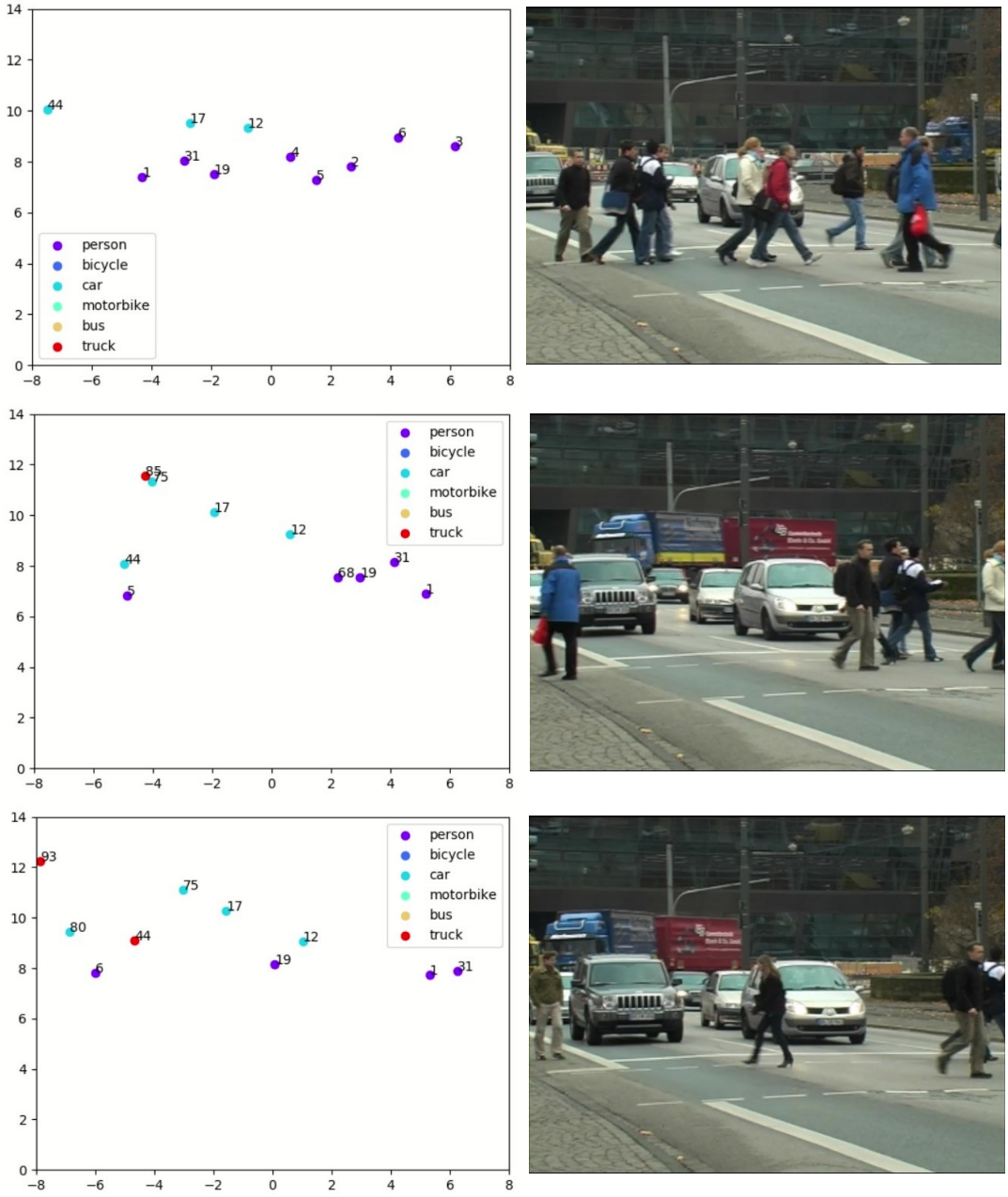

FIGURE 5.30: Examples of the estimated locations of the relevant objects (pedestrians and vehicles) in *TUD-Crossing*, at different frames of the video sequence. These plots represent the ground plane as a 2D grid, where the camera is located at $x = 0$. The frame on the bottom displays a moment where the label of the object identified as 44 (which could subjectively be considered a van-like vehicle) flickers between *car* and *truck*, as it shares visual characteristics from both.

there might not be a discernible disparity, even if the objects are well separated in the real world. In this situation, the accuracy is expected to significantly decrease as the depth values increases, because small errors in disparity lead to large depth errors. This is why, past a certain distance from the camera, depth estimation has been proven to be generally unreliable (Saxena, Schulte, and Ng, 2007).

One of the drawbacks of using the *3DMOT* videos for the evaluation of the

pipeline, is that they are limited to the tracking of pedestrians. They do not include any kind of moving vehicle, which is also a center of interest in our proposed pipeline. As no ground truth data seems to be available for vehicle tracking in 3D, we focus on a qualitative analysis of the *TUD-Crossing* sequence, as it displays both pedestrians and several vehicles (one truck and several cars).

Some examples of the estimated objects' locations and labels for *TUD-Crossing* are given in Figure 5.30. The presence and labels of the objects within the scene are correctly detected and identified. As it was mentioned before, the depth estimation seems to be more compressed on the layers that are the furthest from the camera, which can be seen on the frame shown at the top: the cars are estimated to be closer to each other than they are in the real world. A similar thing happens on the frame in the center, where the truck and the furthest car are estimated to be separated by less than a meter. In this same frame, it is noticeable that the car on the left is wrongly predicted to be closer to the camera than the one on the right. This could be due to the nature of this left car: as it is a jeep-style vehicle, it can appear to have a larger size than the SUV car on the right, hence producing the illusion of proximity.

This same vehicle also displays a striking behavior on the example frame at the bottom: its label actually switches between *car* and *truck* for a few frames. This is provoked by the delimitation of the object classes, as this vehicle shares visual characteristics between cars, trucks, and other non-included vehicle labels such as vans. This flickering is, in any case, easily recognizable (also thanks to the consistent object identification performed by the tracking algorithm) and could be manually fixed in a quick manner. Therefore, it does not strongly affect the purpose of the pipeline. It is, nonetheless, an example of the restrictions caused by a delimited pool of possible object labels, in terms of generalization to newer or unknown scenes.

5.4.1 Usage of object tracking

Finally, one of our goals was to see how the integration of multi-object tracking ultimately influences the estimation of the final 3D locations, in comparison to estimating these locations directly from the object detection results. We therefore run the same evaluation by removing the deepSORT module, and calculating the 3D world coordinates and ground plane positions from the results output by Mask R-CNN. We run this evaluation process only on the *TUD-Stadtmitte* video, as the performance on *PETS09* is too severely affected by the depth outliers to be able to provide a clear and fair comparison.

The results are given in Table 5.10, where the original results using tracking are also provided for comparison. Generally speaking, all the metrics display a visible deterioration when no tracking is used. While the F1-score (and likewise, the corresponding precision and recall values) presents a smaller decay, there is a clear worsening in the multi-object tracking accuracy, combined with a weaker tracking precision. The fact that the ratio of false positives has raised significantly but the miss ratio has not, actually corresponds with the fact that pure detections are used instead of tracked ones: we expect the object detection results to include a higher number of objects, potentially wrong detections, which the tracking algorithm will filter afterwards. This implies that Mask R-CNN's output can possibly contain more false positives, but it also means that there could be more options of estimates to be matched with true objects, hence influencing the miss ratio slightly less. In spite of

this, it is apparent that an object tracking algorithm, and its usage for missing locations interpolation, helps the overall pipeline with its ultimate goal of locating all the relevant objects in the 3D space.

TABLE 5.10: Performance evaluation results of the ground plane locations estimations, for the *3DMOT* training subset's *TUD-Stadtmitte* video, with and without object tracking integration (top and bottom, respectively).

th = 2.0	**MOTP**	**MOTA**	**miss ratio**	**FP ratio**	**precision**	**recall**	**F1-score**	**match/miss**
w/ tracking	1.116	0.177	0.541	0.282	0.696	0.545	0.611	1.197
w/o tracking	1.218	0.027	0.597	0.377	0.590	0.477	0.528	0.911

As explained in Section 3.8, the usage of object tracking not only helps filtering out potentially wrong detections, but it also enables the interpolation of missing locations due to full occlusions. For this aspect, however, a general consistency of the object identification during tracking is necessary. Although this consistency is not strictly necessary for our initially intended application (traffic pattern analysis on an anonymized video), if the object IDs are not generally congruent, there can be certain settings where the interpolation process could even be detrimental to the overall performance of the system. An example of this could be identity switches between two objects right before or after one of these objects is fully occluded, resulting in a possible wrongly interpolated location for this object. Moreover, interpolation would not be possible if an object does not preserve its ID after a full occlusion, meaning that the tracking process is eventually not taken full advantage of.

To evaluate the overall consistency of the tracking algorithm, we display all the estimated ground plane locations throughout the *TUD-Stadtmitte* sequence by assigning a color to each of the output object IDs (see Figure 5.31). Here, we can see the same exact patterns to the ones we analyzed in Figures 5.27, 5.28 and 5.29, but this time with the identification data added.

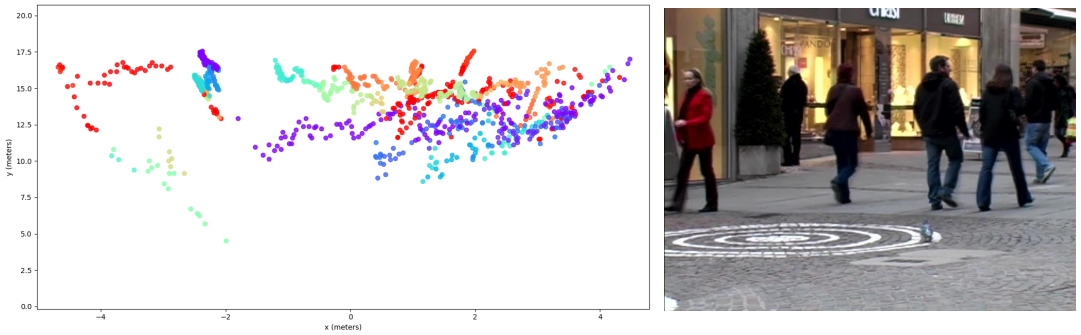

FIGURE 5.31: Estimated locations/trajectories of pedestrians in *TUD-Stadtmitte* (left), and example frame from the *TUD-Stadtmitte* video (right). All the estimated positions throughout the whole video for all the tracked objects are shown, and each location is represented with a different color depending on the corresponding tracked object's ID.

Picking up from the same object examples studied in Section 5.4, we can see how the objects entering or leaving the frame pose similar challenges for the object identification. The person entering the frame during the second half of the video not only presents coherent estimations of her ground plane locations, as we discussed before, but it also displays the same tracked object ID throughout her whole trajectory. The person leaving the frame at the beginning of the video, on the other hand, presents

identity switches that also match the timing of the heavy depth outliers: we can see that its object ID changes twice when she reaches the side of the frame.

In the case of the person standing still, it is noticeable how his object ID varies several times throughout the sequence. Considering the color changes, this could be due to its occlusion by other objects, or them entering and leaving the frame: this person's corresponding ground plane locations match the ID color-code of a few of the other objects, which signals the incidence of several identity switches with them.

Among those people with whom the identity switches happen, we can notice the three people whose trajectories we analyzed as well in Section 5.4. These three people walk by in front of the man standing still, a probable cause of the identity switches during those frames. Their later estimated locations seem, however, to correctly preserve their original object IDs, which can be due to these objects not being occluded throughout the sequence. Again, the objects that suffer the heaviest occlusions are the most affected by identity switches, such as the people walking on the background, which are all fully occluded (by the three aforementioned passersby) at some point during the video.

As pointed out before, a limitation of the *3DMOT* videos for the evaluation of the proposed pipeline is the absence of (moving) vehicles. Similarly to what has been previously done with the *TUD-Crossing* sequence, we perform a qualitative evaluation of the tracking results obtained for a video displaying several vehicles on a road scene: the *Urban Tracker*'s *Sherbrook* sequence. This is particularly interesting to evaluate how well deepSORT generalizes to other types of objects, considering that its appearance feature extraction network was trained on images of people only.

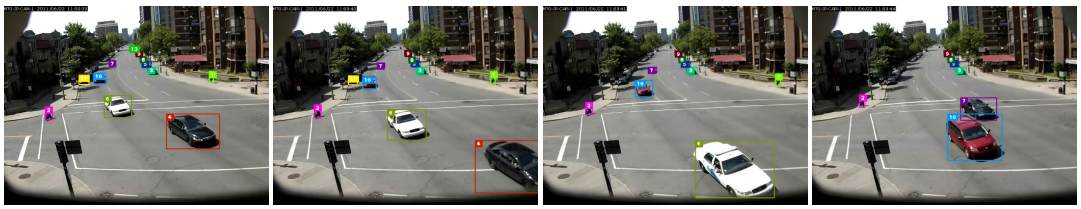

FIGURE 5.32: Example frames of the *Sherbrook* sequence displaying the object tracking results obtained with deepSORT.

In Figure 5.32, several frames from the *Sherbrook* video are shown, containing the tracking results (bounding boxes and object IDs). It is interesting to see how the cars are correctly tracked throughout the video, with no identity switches during their movement. Likewise, the parked cars that are further in the scene also preserve their initial object ID despite the large distance to the camera, which could difficult both the detection and tracking processes.

There are, nevertheless, certain situations where the tracking seems to perform weakly. Two examples are shown in Figure 5.33, on the images tagged as (1) and (3), where the black and white cars, respectively, suffer an identity switch when reaching the bottom of the frame. This shows the difficulties for the tracker to correctly identify semi-occluded or half-shown objects with a label different than *person*, a potential consequence from the training of deepSORT's network based only on people images.

Another noticeable trait of the tracking results in this sequence, as opposed to sequences displaying pedestrians only, is that the tracking results show some flickering for the vehicles, with the bounding boxes disappearing on some of the frames.

An example of this can be seen in Figure 5.33, on the frame tagged as (2), where the car with the ID 7 is lost during tracking.

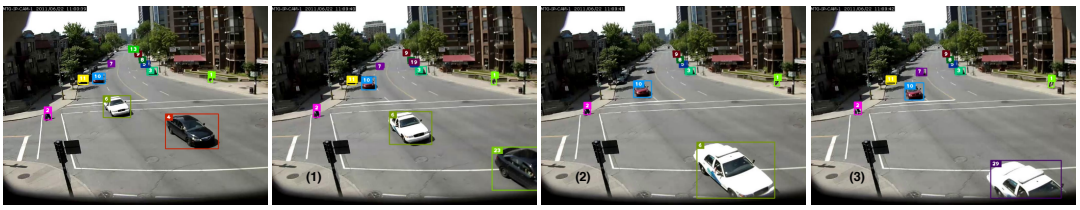

FIGURE 5.33: Example frames of the *Sherbrook* sequence displaying errors within the object tracking results obtained with deepSORT.

In spite of this, the vehicles are correctly re-identified after the flickering. This flickering does therefore not constitute a strong issue for the purpose of the proposed pipeline, as the missing locations can be interpolated considering that the object IDs are consistent.

5.5 Overall discussion

We have first evaluated how the chosen object detection method performs for images displaying the expected scenario: an outdoors scene where both people and vehicles pass by. To compare its performance to other existing object detectors for this particular scene setting, we have analyzed three additional models under identical evaluation conditions, and which were implemented using the same framework. One of the most noticeable traits of the performance results is that, without any kind of modification from the original methods, they display an acute decay in comparison to the ones provided by the authors. This can be reflective of a strong overfitting of the models to the original testing data used by the authors, which hinders their usage as a performance reference point. However, the relative differences among the evaluated object detection methods were still preserved within our evaluation results, indicating that RetinaNet surpassed Mask R-CNN for all average precision metrics, with the YOLOv3 models giving the lowest results for these same metrics.

To answer the **research sub-question 1**, we could see that the object detector that gives the best apparent performance for our objects of interest is the RetinaNet architecture. However, the second best performing detector, Mask R-CNN, is the only method that additionally performed object segmentation, a characteristic that is required for the rest of the pipeline. Moreover, when pre-trained on the *MS COCO* dataset, Mask R-CNN's performance already reaches state-of-the-art levels that sufficiently meet the requirements of our pipeline, even surpassing in some cases what is defined as the ground truth.

Regarding the depth estimation process, we evaluated several models of our chosen estimator trained with different data subsets. The model trained on the *Cityscapes* dataset and tuned on the Eigen split gave the best performance results for all the utilized metrics. This was the case for both evaluation scenarios: when considering the full images, and when considering the pixels belonging to relevant objects only. Nonetheless, although the metrics display a seemingly tolerable average error rate, the qualitative evaluations showed the reality of the output depth maps: the artifacts are abundant, and the estimated depth is not temporally consistent. Moreover, the results for the evaluation metrics when considering *person*-labelled objects only, presented significant deterioration in relation to the general results. Therefore, to answer **research sub-question 2**, the *City2Eigen* model offered the best evaluation results overall, but its performance is still deficient, rendering MonoDepth as a sub-par method for the purpose and requirements of the proposed pipeline.

As we have seen, while the object detection gives highly accurate predictions in terms of object localization and recognition of our objects of interest, the depth estimation procedure entails a severe performance bottleneck. The high presence of outliers in the depth component of the ground plane locations implies that these estimated locations are often not matched to the corresponding true ones. This results in numerous estimates being considered false positives, and likewise, several true objects being accounted as misses.

To demonstrate this, we executed the same tests for evaluating the final output of the system (that is, the estimated object locations on the ground plane), but considering the x-coordinate only. This was done to assess, on one hand, the influence of the erroneous depth predictions on the estimated ground plane locations, and on the

other hand, the performance of the remaining pipeline modules combined (including the 3D world coordinates computation) relative to the ground plane locations estimation. On these tests, the evaluation results dramatically improved, obtaining a much higher number of matches and thus reducing the overall error rates.

On another note, the computed focal length determines the scale of the predicted ground plane area, and if it matches the real one in terms of the absolute units of length. Note that, nonetheless, its estimated value does not affect the relative distances among objects and their spatial relationships. Hence, we can consider that, although it does not affect the subjective intelligibility of the scene, it does indicate the apparent absolute distances within the scene, which are otherwise lost if the original video is no longer available. The recreated ground plane can, however, be scaled to the corresponding real world distances based on the video scene, as a manual correction of the estimations before the deletion of the original video.

While the estimated focal length directly affects the predicted scale of the ground plane, this can be easily corrected based on the original video source, and it does not affect the intelligibility of the scene itself. The modules that most heavily influence the intelligibility of the scene are the object detection and the depth estimation. The former determines the presence of the objects within the scene at each instant throughout the video, while the latter outlines these objects' locations on the 3D space. To answer **research sub-question 3**, considering the large depth outliers given by MonoDepth, this depth estimation method has the most substantial influence on the reliability of the recreated scene. For the two *3DMOT* videos previously evaluated, the multi-object tracking accuracy showed an average improvement of 78.9% when only the x-coordinate is considered as opposed to both coordinates, with an average precision improvement of exactly $1m$. The continuous flickering of the ground plane points and the numerous outliers also imply that, qualitatively, the intelligibility of the reconstructed scene is compromised if the current depth estimation method is in use.

Finally, we integrated a multi-object tracking algorithm to help filter out potential false positive detections, and to smooth the estimated trajectories of the objects within the scene. To see how this process influences the estimation of the ground plane locations, we evaluated the performance of the pipeline when removing the tracking module. The performance results drastically deteriorate in this setting, especially because of the increment in the number of false positives, possibly caused by the absence of the tracking algorithm acting as a filter. Knowing this, to answer **research sub-question 4**, the achieved accuracy is 15% higher with an improved precision in distance of around $0.1m$, for the same evaluation scenario, when using tracking. This does not only show that the object tracking algorithm improves the overall performance, but also that it plays a crucial role in the performance of the proposed pipeline.

Chapter 6

Conclusion

The aim of this research was, firstly, to introduce a new approach to video anonymization, based on recreating the video scene in the 3D space; and secondly, to analyze to what extent this new approach can ensure both the anonymity and intelligibility of the scene. The focus of the research was on a specific case study where the per-frame locations of people and vehicles in an outdoors setting must be preserved, for future analysis of the traffic and motion patterns throughout time.

Contrary to the general approach in video anonymization, where the original video is processed by manipulating the potentially sensitive areas, our method is based on the generation of an alternative feed, where only the relevant information is transferred. This way, we avoid the preservation of any kind of personal data, hence anonymity can be guaranteed.

The proposed method is based on an initial detection phase, where the objects of interest are localized at each video frame, on the 2D image space. These objects are then associated across the frames using a multi-object tracking algorithm, with two main purposes: filtering out false positives from the detected objects, and estimating the trajectories of the objects, to afterwards facilitate their smoothing and the removal of outliers. Parallel to this, a depth map is predicted for each of the video frames. Using this depth information, the 3D world coordinates are calculated for the 2D points at each of the frame images, from which an approximate location on the ground plane of each of the previously detected objects can be computed, by averaging a selection of points within each object's image area. In addition, to obtain a system capable of processing any kind of input video, a camera self-calibration procedure is implemented, so that the camera parameters required for the aforementioned 2D-to-3D point coordinates conversion can be estimated from the recorded scene itself.

We managed to achieve a seamless integration of several state-of-the-art models for object detection and segmentation, depth estimation, and multi-object tracking. The object detection provided an outstanding performance on scenes matching the content of our expected scenario, often surpassing the ground truth expectations, while reaching a mean average precision of 20.1%. This precision was notably higher with medium to large-sized objects, while objects that were further from the camera caused some decline in the performance. The integration of multi-object tracking enabled a drastic improvement of the performance of the overall pipeline, thanks to its filtering of false positive detections and the ability to interpolate missing locations during object occlusions. Its usage led to a multi-object tracking accuracy of 17.7% in our test setting, as opposed to a 2.7% accuracy without tracking under the same conditions, with a further improvement of approximately 0.1 meters in the matching precision when using tracking.

As anonymity is already ensured by the nature of the method, the focus during the evaluation was, therefore, on the degree of scene intelligibility that can be achieved. In our main test setting, at a precision of 1.12 meters on the ground plane, the method was able to reach an accuracy of just 18% when matching our predicted objects to the true ones within the 3D space. However, an accuracy of almost 60% was obtained when raising the matching precision to approximately 1.6 meters. It is important to note, nevertheless, that these results significantly differed depending on the kind of setting. The aforementioned test setting provided a scene shot from a low camera angle, containing only *person*-labeled objects who were mostly close to the camera. The performance significantly worsened in our second test setting, which displayed a much higher viewpoint, and the objects were generally far from the camera. In spite of this, considering that only a single monocular view of the scene -recorded with an unknown, uncalibrated camera- is provided as input, these results are moderately promising.

6.1 Limitations and future work

Despite the hopeful prospects of these results, the proposed method holds a severe performance bottleneck: the depth estimation process. The problem of depth estimation in this scenario, where only a single 2D image is available, is inherently difficult due to its ill-posed nature: for a certain image, there are infinite possible solutions for a matching 3D environment. In the case of our chosen depth estimator, an adequate performance could only be achieved when the input images closely matched the training images, in terms of viewpoint and focal length of the camera. On the contrary, when these parameters diverged from the ones of the training data, the depth maps presented several artifacts, added to the lack of temporal consistency. The areas of the image corresponding to the ground plane displayed the most visible artifacts, which caused the error rates to lower when only the areas corresponding to relevant objects were evaluated. These objects, however, also showed incongruities in their depth maps. Particularly, the areas belonging to objects labeled as *person* yielded larger errors, with an RMSE that doubled its value compared to when the areas belonging to all relevant objects were considered. These limitations are thus the expected cause of the previously mentioned differences in performance, depending on the similarity of the test images to the training data.

The large outliers caused by the depth estimation process also prevented the proposed method from integrating several other mechanisms to improve its performance. An example of this is the implementation of a trajectory outlier removal procedure. This kind of procedure was not possible because the resulting depth estimations were too unstable to provide clear trajectories of the objects. If this kind of technique was to be implemented, it would enable a more accurate smoothing of the trajectories output by the tracking algorithm relative to the true ones.

Similarly to this, one of the initial objectives was to set a certain speed threshold for the tracked objects, which was discarded because of the unreliability of the depth estimation. If the depth outliers were less numerous, a speed threshold would enable the correction of large movements between consecutive frames. This speed threshold could be defined based on the maximum distance that an object can be displaced, depending on both the frame rate and the object class. There is one situation, however, where determining the threshold based on the currently considered object labels would be problematic: the bicycle and motorcycle riders would have to share

the speed threshold of their ridden vehicle. Considering that these two types of two-wheelers have significantly different average speeds, a possible solution would be to re-train the object detector after labeling the riders for each of these vehicles as two separate classes from common pedestrians. The *Cityscapes* dataset could be used, as it already included the *rider* class; this would, nevertheless, require a re-annotation of these riders as separate bicycle and motorcycle riders, which would be a time-consuming task.

With regard to the object detection as well, another limitation encountered through this research was the confusion levels among object classes for certain vehicles, such as vans or jeep-styled cars. This could again be solved by re-training the object detector using several more vehicle-related object classes. It is noteworthy, however, that the expected performance improvement might not be proportionate to the added time and complexity that the extra annotation task implies.

The initial problem to be solved by the proposed method focused on the anonymization of an already existing, single video coming from an unknown camera source. This narrowed the achievable accuracy of processes like depth estimation, 3D reconstruction or camera calibration. Nevertheless, if the intention of anonymization is known beforehand prior to the recording of the video, the whole pipeline could be simplified and a significantly better performance could be attained. Mainly, two or more redundant cameras could be used to record the scene. If the distribution of this multi-camera or stereo setting is known, and the cameras are previously calibrated, the 3D reconstruction procedure will be more straightforward and will enable a more reliable computation of the locations on the ground plane. Particularly for the case of a multiple camera setting, occlusions among objects could be tackled more robustly as several angles of the scene are provided. Furthermore, these kinds of settings would facilitate the 3D tracking of the objects, instead of being limited to 2D tracking algorithms.

As an alternative to cameras or camera-only-based settings, other devices could be used, such as RGBD cameras or LiDAR-based systems. This will make it possible to obtain robust measurements of the depth component of each point within a frame image, effectively enabling the estimation of more consistent locations on the ground plane.

On the contrary, if the intention of video anonymization is not known beforehand and the initial problem remains, the same proposed method could be kept and further improved.

Intuitively, the main focus would be to polish the depth estimation process. On one hand, a different depth estimation approach could be followed where, instead of making use of deep learning (typically a black box technique), the depth could be determined by certain established rules or assumptions based on the expected scene. Some examples would be the assumption of smooth object trajectories on the ground plane, or a constant height of the pedestrians. Although the latter would be ineffective for an automated approach, as height can dramatically vary among people (particularly if children are present in the scene at some point), it would allow a manual estimation of depth, and likewise for the localization of vanishing points for calibrating the camera.

On another hand, if the focus remains on deep learning-based methods, several options could be considered:

First, as the method aims at processing video sequences, it would be effective to

take the temporal domain into account. This could be done, for instance, by using recurrent neural networks such as an LSTM (Long Short-Term Memory) network, where the previous states are considered for the depth predictions. Although the usage of this kind of network does not guarantee a better depth estimation, it would help reduce the number of large depth outliers across consecutive frames. On a different note, and relative to the consideration of the time factor, it would be useful to delve into other object tracking techniques; in particular, as real-time execution is not required for the current application of the method, offline tracking algorithms could be used, where all the frames are checked prior to the association of the detected objects. In addition, knowing that Mask R-CNN is used for the object detection phase, a tracking-by-segmentation paradigm (Yeo et al., 2017) could be followed. This type of tracking aims at solving some of the limitations of tracking-by-detection, with an example being the drift problem: the error accumulation caused by occlusions (Gall, Rosenhahn, and Seidel, 2008) and the re-scaling of bounding boxes for non-rigid objects, such as people.

Another possibility, regardless of the type of deep neural network to be used, is to train the depth estimation network with a customized dataset. Mainly due to the limitations imposed by the time constraints of this research, several compromises had to be done concerning the choice of datasets. Pre-trained versions of the predictive models were used, restricting the choice of training data. This was, nevertheless, not a significant drawback, since these versions were mostly trained by the original authors of the methods. This generally meant that the training processes had been done using appropriate training data (such as the extensively used *KITTI* or *Cityscapes* datasets) and computationally powerful resources, ultimately yielding models that already reached optimal results. Despite this, the restricted choice of data delimited the robustness of the models: for instance, all the datasets used for training the depth estimation models had a similar viewpoint, preventing a good performance on other types of settings (such as the one in the *PETS09* sequence).

These limitations not only applied to the training data but also to the evaluation data. The unavailability of relevant depth data, plus the lack of stereo datasets already including disparity or depth maps, compelled the usage of an evaluation subset based on *KITTI*. Although the images from this dataset were not seen during training, they shared several similarities with the training data (again, mostly viewpoint-wise). This caused seemingly encouraging quantitative results, while at the same time, the qualitative evaluation on non-annotated images from different datasets displayed large artifacts and errors.

To solve these limitations, a larger and broader training dataset covering a diverse range of scenarios and viewpoints could be created, with the aim of achieving a more robust network able to generalize to different input images. A dataset like this one could be built upon existing datasets, including the ones used throughout this research.

An alternative would be to do the exact opposite, by training the network with depth information delimited to the specific scene, camera height and viewpoint that are expected in the input video. This way, the network would be overfitted to this kind of scenario, which would nonetheless be adequate for a restricted and known usage of the proposed method, but not for a generalized use.

As mentioned before, stereo image datasets are significantly more common than RGBD datasets, but the depth information is not directly provided and has to be

computed based on the camera parameters. While RGBD data provides more reliable depth information, it can still contain noise, depending on the camera or sensors used for capturing it. To obtain a more trustworthy ground truth data for training, synthetic target data could be used. This is a common approach within the context of domain adaptation (Wang and Deng, 2018), and it consists in the generation of artificial data that resembles the real targeted one. In the case of our expected outdoors street scenario, an adequate approach would be to create 3D environments populated by animated models of people and vehicles. With this kind of data, true depth maps could be automatically generated for each of the frames, as the 3D coordinates of each point in the 3D space are explicitly known. An example of this kind of dataset is *SYNTHIA* (Ros et al., 2016), which provides sensor simulations (RGB cameras and depth sensors) of an outdoors 3D-modeled environment, annotated following the format of the *Cityscapes* dataset.

Having this kind of synthetic dataset also implies that the locations on the ground plane of each present object can be accurately extracted in a straightforward manner. This opens up the possibility to train a fully end-to-end network that could directly predict the ground plane locations of the relevant objects in an image or video. This line of thought is inspired by other end-to-end approaches focusing on predicting, for instance, the 3D human pose of the people in a video (Alp-Güler, Neverova, and Kokkinos, 2018); or on performing combined 3D object detection, tracking and motion forecasting (Luo, Yang, and Urtasun, 2018). An end-to-end CNN that predicts the ground plane locations directly from the input images would unify all the required tasks in a single, fully trainable network. This would simplify the overall pipeline, avoiding the propagation of errors from one module to the next and the need for optimizing each of the modules. Nonetheless, estimating 3D locations from single 2D images is a complex problem as seen throughout this research. The inability to decompose the problem into individual tasks due to this end-to-end nature could hence end up deteriorating the overall performance of the pipeline. In spite of this, it is yet to be seen how future advancements in the development of end-to-end deep neural networks could provide a solution to tackle the problem of 3D locations estimation to, ultimately, achieve a reliable anonymized scene reconstruction.

Bibliography

Abadi, M. et al. (2016). "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283.

Agrawal, P. and P.J. Narayanan (2011). "Person De-identification in Videos". In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.3, pp. 299–310. DOI: 10.1109/TCSVT.2011.2105551.

Alp-Güler, R., N. Neverova, and I. Kokkinos (2018). "DensePose: Dense Human Pose Estimation in the Wild". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 0.1109/CVPR.2018.00762.

Andriyenko, A. and K. Schindler (2011). "Multi-Target Tracking by Continuous Energy Minimization". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 1265–1272. DOI: 10.1109/CVPR.2012.6247893.

Andriyenko, A., K. Schindler, and S. Roth (2012). "Discrete-Continuous Optimization for Multi-Target Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 1926–1933. DOI: 10.1109/CVPR.2012.6247893.

Baaziz, N. et al. (2007). "Security and Privacy Protection for Automated Video Surveillance". In: *IEEE International Symposium on Signal Processing and Information Technology*, pp. 17–22. DOI: 10.1109/ISSPIT.2007.4458044.

Bae, S.-H. and K.-J. Yoon (2014). "Robust Online Multi-Object Tracking based on Tracklet Confidence and Online Discriminative Appearance Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1218–1225. DOI: 10.1109/CVPR.2014.159.

BenAbdelkader, C., R. Cutler, and L. Davis (2002). "Stride and Cadence as a Biometric in Automatic Person Identification and Verification". In: *Proceedings of the 5th IEEE International Conference on Automatic Face Gesture Recognition*. Washington DC, USA: IEEE. DOI: 10.1109/AFGR.2002.1004182.

Berger, A. M. (2000). *Privacy Mode for Acquisition Cameras and Camcorders*. US Patent 6,067,399.

Bernardin, K. and R. Stiefelhagen (2008). "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics". In: *EURASIP Journal on Image and Video Processing* 2008.1, p. 246309. DOI: 10.1155/2008/246309.

Bewley, A. et al. (2016). "Simple Online and Realtime Tracking". In: *CoRR* abs/1602.00763. URL: http://arxiv.org/abs/1602.00763.

Beymer, D. and K. Konolige (2017). "Real-Time Tracking of Multiple People Using Continuous Detection". In: *IEEE International Conference on Computer Vision (ICCV). Frame Rate Workshop*, pp. 1–8. DOI: 10.1109/TPAMI.2018.2844175.

Birnstill, P. et al. (2015). "Privacy-Preserving Surveillance: an Interdisciplinary Approach". In: *International Data Privacy Law* 5.4, pp. 298–308. DOI: 10.1093/idpl/ipv021.

Bitouk, D. et al. (2008). "Face Swapping: Automatically Replacing Faces in Photographs". In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 3. ACM, p. 39. DOI: 10.1145/1399504.1360638.

Bobick, A. and J. Davis (2001). "The Representation of Human Movement Using Temporal Templates". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.3, pp. 257–267. DOI: 10.1109/34.910878.

Bose, B. and W.E.L. Grimson (2003). "Ground Plane Rectification by Tracking Moving Objects". In: *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 94–101.

Boyle, M., C. Edwards, and S. Greenberg (2000). "The Effects of Filtered Video on Awareness and Privacy". In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM, pp. 1–10. DOI: 10.1145/358916.358935.

Bradski, G. and A. Kaehler (2013). *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*. 2nd. O'Reilly Media, Inc.

Cavallaro, A. (2007). "Privacy in Video Surveillance [In the Spotlight]". In: *IEEE Signal Processing Magazine* 24.2, pp. 168–166. DOI: 10.1109/MSP.2007.323270.

Chaaraoui, A.A., P. Climent-Pérez, and F. Flórez-Revuelta (2013). "Silhouette-based Human Action Recognition Using Sequences of Key Poses". In: *Pattern Recognition Letters* 34.15, pp. 1799–1807. DOI: 10.1016/j.patrec.2013.01.021.

Chavdarova, T. et al. (2018). "WILDTRACK: A Multi-Camera HD Dataset for Dense Unscripted Pedestrian Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5030–5039. DOI: 10.1109/CVPR.2018.00528.

Chinomi, K. et al. (2008). "PriSurv: Privacy Protected Video Surveillance System Using Adaptive Visual Abstraction". In: *International Conference on Multimedia Modeling*. Springer, pp. 144–154. DOI: 10.1007/978-3-540-77409-9_14.

Choi, W. (2015). "Near-Online Multi-Target Tracking with Aggregated Local Flow Descriptor". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 3029–3037. DOI: 10.1109/ICCV.2015.347.

Chollet, F. (2015). *Keras*. URL: https://github.com/fchollet/keras.

Cordts, M. et al. (2016). "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 3213–3223. DOI: 10.1109/CVPR.2016.350.

Curless, B.L. (1997). *New Methods for Surface Reconstruction from Range Images*. PhD thesis, Stanford University.

Cutting, J.E. and L.T Kozlowski (1977). "Recognizing Friends By Their Walk: Gait Perception Without Familiarity Cues". In: *Bulletin of the Psychonomic Society* 9.5, pp. 353–356. DOI: 10.3758/BF03337021.

Dalal, N. and B. Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

De Silva, L. (2008). "Audiovisual Sensing of Human Movements for Home-care and Security in a Smart Environment". In: *International Journal on Smart Sensing and Intelligent Systems* 1. DOI: 10.21307/ijssis-2017-288.

Dicle, C., O. I. Camps, and M. Sznaier (2013). "The Way They Move: Tracking Multiple Targets with Similar Appearance". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2304–2311. DOI: 10.1109/ICCV.2013.286.

Dufaux, F. and T. Ebrahimi (2006). "Scrambling for Video Surveillance with Privacy". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*. DOI: 10.1109/CVPRW.2006.184.

Eigen, D. and R. Fergus (2015). "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture". In: *Proceedings*

of the IEEE International Conference on Computer Vision (ICCV), pp. 2650–2658. DOI: 10.1109/ICCV.2015.304.

Eigen, D., C. Puhrsch, and R. Fergus (2014). "Depth Map Prediction from a Single Image Using a Multi-scale Deep Network". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems*. Vol. 2. MIT Press, pp. 2366–2374.

Erhan, D. et al. (2014). "Scalable Object Detection Using Deep Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2147–2154. DOI: 10.1109/CVPR.2014.276.

European Commission (2018). *What is personal data?* URL: https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en (visited on 02/20/2019).

European Parliament and Council of the EU (2014). *EC Data Protection Working Party 216: Article 29 Opinion 05/2014 on Anonymisation Techniques*. URL: https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf (visited on 02/20/2019).

— (2018). *European General Data Protection Regulation*. URL: https://gdpr-info.eu/ (visited on 02/19/2019).

Everingham, M. et al. (2010). "The PASCAL Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2, pp. 303–338. DOI: 10.1007/s11263-009-0275-4.

Fagot-Bouquet, L. et al. (2015). "Online Multi-Person Tracking based on Global Sparse Collaborative Representations". In: *IEEE International Conference on Image Processing (ICIP)*, pp. 2414–2418. DOI: 10.1109/ICIP.2015.7351235.

Fleck, S. and W. Straßer (2008). "Smart Camera-based Monitoring System and its Application to Assisted Living". In: *Proceedings of the IEEE* 96.10, pp. 1698–1714. DOI: 10.1109/JPROC.2008.928765.

Freund, Y. and R.E. Schapire (1997). "A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting". In: *Journal of computer and system sciences* 55.1, pp. 119–139. DOI: 10.1006/jcss.1997.1504.

Frome, A. et al. (2009). "Large-Scale Privacy Protection in Google Street View". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2373–2380. DOI: 10.1109/ICCV.2009.5459413.

Gall, J., B. Rosenhahn, and H.-P. Seidel (2008). "Drift-Free Tracking of Rigid and Articulated Objects". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 1–8. DOI: 10.1109/CVPR.2008.4587558.

Gallagher, A.C. and T. Chen (2008). "Clothing Cosegmentation for Recognizing People". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8. DOI: 10.1109/CVPR.2008.4587481.

Garg, R. et al. (2016). "Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue". In: *European Conference on Computer Vision*. Springer, pp. 740–756. DOI: 10.1007/978-3-319-46484-8_45.

Geiger, A., P. Lenz, and R. Urtasun (2012). "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

Girshick, R. (2015). "Fast R-CNN". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

Girshick, R. et al. (2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

Godard, C., O.M. Aodha, and G.J. Brostow (2017). "Unsupervised Monocular Depth Estimation with Left-Right Consistency". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6602–6611. DOI: 10.1109/CVPR.2017.699.

Gross, R. et al. (2009). "Face De-identification". In: *Protecting Privacy in Video Surveillance*. Springer, pp. 129–146. DOI: 10.1007/978-1-84882-301-3_8.

He, K., G. Gkioxari, et al. (2017). "Mask R-CNN". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2961–2969. DOI: 10.1109/TPAMI.2018.2844175.

He, K., X. Zhang, et al. (2015). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9, pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

Hoiem, D., Y. Chodpathumwan, and Q. Dai (2012). "Diagnosing Error in Object Detectors". In: *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*. Vol. 3. Florence, Italy: Springer-Verlag, pp. 340–353. DOI: 10.1007/978-3-642-33712-3_25.

Hoiem, D., A. A. Efros, and M. Hebert (2007). "Recovering Surface Layout from an Image". In: *International Journal of Computer Vision* 75.1, pp. 151–172. DOI: 10.1007/s11263-006-0031-y.

Horn, B.K.P. and B.G. Schunck (1981). "Determining Optical Flow". In: *Artificial intelligence* 17.1-3, pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2.

Howard, I. (2012). *Perceiving in Depth, Vol. 1: Basic Mechanisms*. Oxford University Press.

Huang, J. et al. (2017). "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3296–3297. DOI: 10.1109/CVPR.2017.351.

Hui, J. (2018). *Image segmentation with Mask R-CNN*. URL: https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272 (visited on 08/19/2019).

Jaehoon, J., I. Yoon, and J. Paik (June 2016). "Object Occlusion Detection Using Automatic Camera Calibration for a Wide-Area Video Surveillance System". In: *Sensors* 16, p. 982. DOI: 10.3390/s16070982.

Jain, A. and U. Uludag (2003). "Hiding Biometric Data". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.11, pp. 1494–1498. DOI: 10.1109/TPAMI.2003.1240122.

Jain, A.K., S.C. Dass, and K. Nandakumar (2004). "Soft Biometric Traits for Personal Recognition Systems". In: *International Conference on Biometric Authentication, Lecture Notes in Computer Science* 3072, pp. 731–738. DOI: 10.1007/978-3-540-25948-0_99.

Jodoin, J., G. Bilodeau, and N. Saunier (2014). "Urban Tracker: Multiple Object Tracking in Urban Mixed Traffic". In: *IEEE Winter Conference on Applications of Computer Vision*, pp. 885–892. DOI: 10.1109/WACV.2014.6836010.

Jones, E., T. Oliphant, P. Peterson, et al. (2001). *SciPy: Open Source Scientific Tools for Python*. URL: https://www.scipy.org.

Junejo, I. and H. Foroosh (2006). "Robust Auto-Calibration from Pedestrians". In: *IEEE International Conference on Video and Signal Based Surveillance*. IEEE, pp. 92–92. DOI: 10.1109/AVSS.2006.99.

K. Hata, S. Savarese (2017). *CS231A Course Notes 3: Epipolar Geometry*. URL: https://web.stanford.edu/class/cs231a/course_notes/03-epipolar-geometry.pdf (visited on 07/03/2019).

Kim, C. et al. (2015). "Multiple Hypothesis Tracking Revisited". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 4696–4704. DOI: 10.1109/ICCV.2015.533.

Korshunov, P., C. Araimo, et al. (2012). "Subjective study of privacy filters in video surveillance". In: *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pp. 378–382. DOI: 10.1109/MMSP.2012.6343472.

Korshunov, P. and T. Ebrahimi (2014). "Towards Optimal Distortion-Based Visual Privacy Filters". In: *IEEE International Conference on Image Processing (ICIP)*, pp. 6051–6055. DOI: 10.1109/ICIP.2014.7026221.

Krahnstoever, N. and P. R. S. Mendonça (2006). "Autocalibration from Tracks of Walking People". In: *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 4–7.

Krizhevsky, A., I. Sutskever, and G.E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* 25, pp. 1097–1105.

Ladicky, L., J. Shi, and M. Pollefeys (2014). "Pulling Things Out of Perspective". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 89–96. DOI: 10.1109/CVPR.2014.19.

Laina, I. et al. (2016). "Deeper Depth Prediction with Fully Convolutional Residual Networks". In: *IEEE 4th International Conference on 3D Vision (3DV)*. IEEE, pp. 239–248. DOI: 10.1109/3DV.2016.32.

Leal-Taixé, L., M. Fenzi, et al. (2014). "Learning an Image-based Motion Context for Multiple People Tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3542–3549. DOI: 10.1109/CVPR.2014.453.

Leal-Taixé, L., A. Milan, et al. (2017). "Tracking the Trackers: an Analysis of the State of the Art in Multiple Object Tracking". In: *arXiv preprint arXiv:1704.02781*.

Leal-Taixé, L. et al. (2015). "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking". In: *arXiv preprint arXiv:1504.01942*.

Li, X. et al. (2013). "A Survey of Appearance Models in Visual Object Tracking". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 4.4, p. 58. DOI: 10.1145/2508037.2508039.

Lin, T.-Y., P. Dollár, et al. (2017). "Feature Pyramid Networks for Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2117–2125. DOI: 10.1109/CVPR.2017.106.

Lin, T.-Y., P. Goyal, et al. (2017). "Focal Loss for Dense Object Detection". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988. DOI: 10.1109/TPAMI.2018.2858826.

Lin, T.-Y., M. Maire, et al. (2014). "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 740–755.

Liu, B., S. Gould, and D. Koller (2010). "Single Image Depth Estimation from Predicted Semantic Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1253–1260. DOI: 10.1109/CVPR.2010.5539823.

Liu, J., R. T. Collins, and Y. Liu (2011). "Surveillance Camera Autocalibration Based on Pedestrian Height Distributions". In: *British Machine Vision Conference (BMVC)*. Vol. 2.

Liu, W., D. Anguelov, et al. (2016). "SSD: Single Shot MultiBox Detector". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2.

Lowe, D.G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2, pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94.

Lukežic, A. et al. (2017). "Discriminative Correlation Filter with Channel and Spatial Reliability". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4847–4856. DOI: 10.1109/CVPR.2017.515.

Luo, W., B. Yang, and R. Urtasun (2018). "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting With a Single Convolutional Net". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2018.00376.

Lv, F., T. Zhao, and R. Nevatia (2002). "Self-Calibration of a Camera from Video of a Walking Human". In: *Object Recognition Supported by User Interaction for Service Robots*. Vol. 1. IEEE, pp. 562–567. DOI: 10.1109/ICPR.2002.1044793.

— (2006). "Camera Calibration from Video of a Walking Human". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.9, pp. 1513–1518. DOI: 10.1109/TPAMI.2006.178.

Milan, A., L. Leal-Taixé, et al. (2015). "Joint Tracking and Segmentation of Multiple Targets". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5397–5406. DOI: 10.1109/CVPR.2015.7299178.

Milan, A., S. Roth, and K. Schindler (2014). "Continuous Energy Minimization for Multitarget Tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.1, pp. 58–72. DOI: 10.1109/TPAMI.2013.103.

Milan, A., K. Schindler, and S. Roth (2013). "Challenges of Ground Truth Evaluation of Multi-Target Tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 735–742. DOI: 10.1109/CVPRW.2013.111.

Nam, H. and B. Han (2016). "Learning Multi-Domain Convolutional Neural Networks for Visual Tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4293–4302. DOI: 10.1109/CVPR.2016.465.

Newton, E., L. Sweeney, and B. Malin (2005). "Preserving Privacy by De-identifying Facial Images". In: *IEEE Transactions on Knowledge and Data Engineering* 17.2. DOI: 10.1109/TKDE.2005.32.

Nodari, A., M. Vanetti, and I. Gallo (2012). "Digital Privacy: Replacing Pedestrians from Google Street View Images". In: *IEEE Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pp. 2889–2893.

Padilla-López, J.R., A.A. Chaaraoui, and F. Flórez-Revuelta (2015). "Visual Privacy Protection Methods: A Survey". In: *Expert Systems with Applications* 42.9, pp. 4177–4195. DOI: 10.1016/j.eswa.2015.01.041.

Park, S. and H. A. Kautz (2008). "Privacy-Preserving Recognition of Activities in Daily Living from Multi-view Silhouettes and RFID-based Training". In: *AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems*.

Pedregosa, F. et al. (2011). "Scikit-Learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.Oct, pp. 2825–2830.

Penney, J.W. (2016). "Chilling Effects: Online Surveillance and Wikipedia Use". In: *Berkeley Technology Law Journal* 31, p. 117. DOI: 10.15779/Z38SS13.

Porikli, F. and A. Yilmaz (2012). "Video Analytics for Business Intelligence. Studies in Computational Intelligence". In: *Object Detection and Tracking* 409, pp. 3–41. DOI: 10.1007/978-3-642-28598-1_1.

Prinosil, J. et al. (2015). "Automatic Hair Color De-identification". In: *IEEE International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 732–736. DOI: 10.1109/ICGCIoT.2015.7380559.

Redmon, J., S. Divvala, et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

Redmon, J. and A. Farhadi (2016). "YOLO9000: Better, Faster, Stronger". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

— (2018). "YOLOv3: An Incremental Improvement". In: vol. abs/1804.02767.

Remondino, F. (2007). "Detailed Image-Based 3D Geometric Reconstruction of Heritage Objects". In: *Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V. (DGPF) Tagungsband* 16.

Ren, S. et al. (2017). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6, pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

Ribaric, S., A.M. Ariyaeeinia, and N. Pavesic (2016). "De-identification for Privacy Protection in Multimedia Content: A Survey". In: *Signal Processing: Image Communications* 47, pp. 131–151. DOI: 10.1016/j.image.2016.05.020.

Romera-Paredes, B. and P.H.S. Torr (2016). "Recurrent Instance Segmentation". In: *European Conference on Computer Vision. Lecture Notes in Computer Science* 9910, pp. 312–329. DOI: 10.1007/978-3-319-46466-4_19.

Ros, G. et al. (2016). "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2016.352.

Sadimon, S. B. et al. (2010). "Computer Generated Caricature: A Survey". In: *2010 International Conference on Cyberworlds*, pp. 383–390. DOI: 10.1109/CW.2010.33.

Saxena, A., S.H. Chung, and A.Y. Ng (2008). "3D Depth Reconstruction from a Single Still Image". In: *International Journal of Computer Vision* 76.1, pp. 53–69. DOI: 10.1007/s11263-007-0071-y.

Saxena, A., J. Schulte, and A. Y. Ng (2007). "Depth Estimation Using Monocular and Stereo Cues". In: *International Joint Conferences on Artificial Intelligence (IJCAI)*. Vol. 7, pp. 2197–2203.

Seitz, S.M. et al. (2006). "A comparison and evaluation of multi-view stereo reconstruction algorithms". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1, pp. 519–528. DOI: 10.1109/CVPR.2006.19.

Senior, A. et al. (2005). "Enabling Video Privacy through Computer Vision". In: *IEEE Security & Privacy* 3.3, pp. 50–57. DOI: 10.1109/MSP.2005.65.

Sermanet, P. et al. (2013). "OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks". In: *CoRR* abs/1312.6229.

Silberman, N. et al. (2012). "Indoor Segmentation and Support Inference from RGBD Images". In: *European Conference on Computer Vision*. Springer, pp. 746–760. DOI: 10.1007/978-3-642-33715-4_54.

Sochor, J., R. Juránek, and A. Herout (2017). "Traffic Surveillance Camera Calibration by 3D Model Bounding Box Alignment for Accurate Vehicle Speed Measurement". In: *CoRR* abs/1702.06451. URL: http://arxiv.org/abs/1702.06451.

Solove, D. (2008). *Understanding Privacy*. Harvard University Press. ISBN: 978-0-674-02772-5.

Stauffer, C. and W.E.L. Grimson (1999). "Adaptive Background Mixture Models for Real-Time Tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. IEEE, pp. 246–252. DOI: 10.1109/CVPR.1999.784637.

Stenger, B. et al. (2001). "Topology Free Hidden Markov Models: Application to Background Modeling". In: *Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV)*. Vol. 1. IEEE, pp. 294–301. DOI: 10.1109/ICCV.2001.937532.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Microsoft Research. DOI: 10.1007/978-1-84882-935-0.

Tang, Z., J. Hwang, et al. (2016). "Multiple-Kernel Adaptive Segmentation and Tracking (MAST) for Robust Object Tracking". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1115–1119. DOI: 10.1109/ICASSP.2016.7471849.

Tang, Z., Y.-S. Lin, K.-H. Lee, J.-N. Hwang, and J.-H. Chuang (2019). "ESTHER: Joint Camera Self-Calibration and Automatic Radial Distortion Correction From Tracking of Walking Humans". In: *IEEE Access* 7, pp. 10754–10766. DOI: 10.1109/ACCESS.2019.2891224.

Tang, Z., Y. Lin, K. Lee, J. Hwang, J. Chuang, and Z. Fang (2016). "Camera Self-Calibration from Tracking of Moving Persons". In: *23rd International Conference on Pattern Recognition (ICPR)*, pp. 265–270. DOI: 10.1109/ICPR.2016.7899644.

Tansuriyavong, S. and S.-i. Hanaki (2001). "Privacy Protection by Concealing Persons in Circumstantial Video Image". In: *Proceedings of the 2001 Workshop on Perceptive User Interfaces*. ACM, pp. 1–4. DOI: 10.1145/971478.971519.

Uijlings, J.R.R. et al. (2013). "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104.2, pp. 154–171. DOI: 10.1007/s11263-013-0620-5.

Van Der Walt, S., S.C. Colbert, and G. Varoquaux (2011). "The NumPy Array: a Structure for Efficient Numerical Computation". In: *Computing in Science & Engineering* 13.2, p. 22. DOI: 10.1109/MCSE.2011.37.

Wang, M. and W. Deng (2018). "Deep Visual Domain Adaptation: A Survey". In: *Neurocomputing* 312, pp. 135–153. DOI: 10.1016/j.neucom.2018.05.083.

Wang, P., X. Shen, et al. (2015). "Towards Unified Depth and Semantic Prediction from a Single Image". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2800–2809. DOI: 10.1109/CVPR.2015.7298897.

Warren, S.D. and L.D. Brandeis (1890). "The Right to Privacy". In: *Harvard Law Review* 4, pp. 193–220. DOI: 10.2307/1341305.

Wickramasuriya, J. et al. (2005). "Privacy-Protecting Video Surveillance". In: *Real-Time Imaging IX*. Vol. 5671, pp. 64–75. DOI: 10.1117/12.587986.

Wojke, N., A. Bewley, and D. Paulus (2017). "Simple Online and Realtime Tracking with a Deep Association Metric". In: *CoRR* abs/1703.07402. URL: http://arxiv.org/abs/1703.07402.

Xie, J., R. Girshick, and A. Farhadi (2016). "Deep3D: Fully Automatic 2D-to-3D Video Conversion with Deep Convolutional Neural Networks". In: *European Conference on Computer Vision*. Springer, pp. 842–857. DOI: 10.1007/978-3-319-46493-0_51.

Yang, L. et al. (2018). "Efficient Video Object Segmentation via Network Modulation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, UT, USA, pp. 6499–6507. DOI: 10.1109/CVPR.2018.00680.

Yeo, D. et al. (2017). "Superpixel-Based Tracking-by-Segmentation Using Markov Chains". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1812–1821. DOI: 10.1109/CVPR.2017.62.

Yilmaz, A., O. Javed, and M. Shah (2006). "Object Tracking: A Survey". In: *ACM Computing Surveys (CSUR)* 38.4, p. 13. DOI: 10.1145/1177352.1177355.

Yoon, J. H. et al. (2015). "Bayesian Multi-Object Tracking Using Motion Context from Multiple Objects". In: *IEEE Winter Conference on Applications of Computer Vision*, pp. 33–40. DOI: 10.1109/WACV.2015.12.

Zaitoun, N.M. and M.J. Aqel (2015). "Survey on Image Segmentation Techniques". In: *Procedia Computer Science* 65, pp. 797–806. DOI: 10.1016/j.procs.2015.09.027.

Zhang, C., Y. Tian, and E. Capezuti (2012). "Privacy Preserving Automatic Fall Detection for Elderly using RGBD Cameras". In: *International Conference on Computers for Handicapped Persons*. Springer, pp. 625–633. DOI: 10.1007/978-3-642-31522-0_95.

Zhang, R., P.-S. Tsai, et al. (1999). "Shape-from-Shading: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.8, pp. 690–706. DOI: 10.1109/34.784284.

Zhang, Z., M. Li, et al. (2008). "Practical Camera Auto-Calibration Based on Object Appearance and Motion for Traffic Scene Visual Surveillance". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8. DOI: 10.1109/CVPR.2008.4587780.

Zhao, Z. et al. (2019). "Object Detection With Deep Learning: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21. DOI: 10.1109/TNNLS.2018.2876865.

Zheng, L. et al. (2016). "MARS: A Video Benchmark for Large-Scale Person Re-Identification". In: *European Conference on Computer Vision*. Springer, pp. 868–884. DOI: 10.1007/978-3-319-46466-4_52.