mathematics

Review

A Review of Tracking and Trajectory Prediction Methods for Autonomous Driving

Florin Leon and Marius Gavrilescu *

Faculty of Automatic Control and Computer Engineering, "Gheorghe Asachi" Technical University of Iaşi, Bd. Mangeron 27, 700050 Iaşi, Romania; florin.leon@academic.tuiasi.ro
* Correspondence: marius.gavrilescu@academic.tuiasi.ro

Abstract: This paper provides a literature review of some of the most important concepts, techniques, and methodologies used within autonomous car systems. Specifically, we focus on two aspects extensively explored in the related literature: tracking, i.e., identifying pedestrians, cars or obstacles from images, observations or sensor data, and prediction, i.e., anticipating the future trajectories and motion of other vehicles in order to facilitate navigating through various traffic conditions. Approaches based on deep neural networks and others, especially stochastic techniques, are reported.

Keywords: autonomous driving; object tracking; trajectory prediction; deep neural networks; stochastic methods

check for updates

Citation: Leon, F.; Gavrilescu, M. A Review of Tracking and Trajectory Prediction Methods for Autonomous Driving. *Mathematics* **2021**, *9*, 660. https://doi.org/10.3390/math9060660

Academic Editor: Denis N. Sidorov

Received: 28 January 2021
Accepted: 17 March 2021
Published: 19 March 2021

1. Introduction

Autonomous car technology is already being developed by many companies on different types of vehicles. Complete driverless systems are still at an advanced testing phase, but partially automated systems have been around in the automotive industry for the last few years. Autonomous driving technology has been the focus of multiple research and development efforts by various car manufacturers, universities, and research centers, since the middle 1980s.

A famous competition was the DARPA Urban Challenge in 2007. Other examples include the European Land-Robot Trial, which has been held since 2006, the Intelligent Vehicle Future Challenge, between 2009 and 2013, as well as the Autonomous Vehicle Competition, held between 2009 and 2017. Since the early stages of autonomous driving technology development, research in the related fields has been garnering significant interest in universities and industry worldwide.

In this review, we focus on two aspects of an autonomous car system:

- *Tracking:* identifying traffic participants, i.e., cars, pedestrians, and obstacles from sequences of images, sensor data, or observations. It is assumed that some preprocessing of sensor data and/or input images has already been done;
- *Prediction:* assessing the future motion of surrounding vehicles in order to navigate through various traffic scenarios. Beside the prediction of the simple physical behavior of the agents based on a set of past observations, an important issue is to take into account their possible interactions.

The paper is composed of two main parts that focus on these topics.

Section 2 deals with tracking problems as addressed in the related literature. We cover aspects concerning the extraction and use of various features for the detection of pedestrians, vehicles, and obstacles across sequences of images and sensor data. Also, we address the various ways in which authors tackle the problems of ensuring detection consistency, temporal coherence, or occlusion handling. We present methods using deep neural networks, but also alternative, conventional approaches.

Section 3 addresses the problem of motion and behavior prediction in traffic scenarios. We discuss various solutions proposed in the related literature for predicting the trajectory of the ego car with respect to the behavior of other traffic participants. We address methods based on deep neural networks and stochastic models, as well as various mixed approaches.

Section 4 contains some conclusions with regard to the aspects discussed throughout the paper.

2. Tracking Methods

Object tracking is an important part of ensuring accurate and efficient autonomous driving. The identification of objects such as pedestrians, cars, and various obstacles from images and vehicle sensor data is a significant and complex interdisciplinary domain. It involves contributions from computer vision, signal processing, and/or machine learning. Object tracking is an essential part of ensuring safe autonomous driving, since it can aid in obstacle avoidance, motion estimation, the prediction of the intentions of pedestrians and other vehicles, as well as path planning. Most sensor data that have to be processed take the form of point clouds, images, or a combination of the two. Point cloud data may be handled in a multitude of ways, the most common of which is some form of 3D grid, where a voxel engine is used to traverse the point space. Some situations call for a reconstruction of the environment from the point cloud which involves various means of resampling and filtering. In some instances, stereo visual information is available and disparities must be computed from the left-right images. Stereo matching is not a trivial task and has the drawback that the computations required for reasonable accuracy usually have a significant impact on performance. In other cases, multiple types of sensor data are available, thereby requiring registration, point matching, and image/point cloud fusion. The problem is further complicated by the necessity to account for temporal cues and to estimate motion from time-based frames.

The scenes involved in autonomous driving scenarios rarely feature a single individual target. Most commonly, multiple objects must be identified and tracked concurrently, some of which may be in motion relative to the vehicle and to each other. As such, most approaches in the related literature handle more than one object and are therefore aimed at solving multiple object tracking problems (MOT).

The tracking problem can be summarized as follows: a sequence of sensor data is available from one or multiple vehicle-mounted acquisitions devices. Considering that several observations are identified in all or some of the frames from the sequence, how can the observations from each frame be associated with a set of objects (pedestrians, vehicles, and various obstacles) and how can the trajectories of each such object be reconstructed and predicted as accurately as possible?

Most related methods involve assigning an ID or identifying a response for all objects detected within a frame, and then attempting to match the IDs across subsequent frames. This is often a complex task, considering that the tracked objects may enter and leave the frame at different timestamps. They may also be occluded by the environment or may occlude each other. Additional problems may be caused by defects in the acquired images: noise, sampling or compression artifacts, aliasing, or acquisition errors.

Object tracking for automated driving most commonly has to operate on real-time video. As such, the objective is to correlate tracked objects across multiple video frames, in addition to individual object identification. Accounting for variations in motion comes with an additional set of pitfalls, such as when objects are affected by rotation or scaling transformations, or when the movement speed of the objects is high relative to the frame rate.

In the majority of cases, images are the primary modality for perceiving the scene. As such, a lot of efforts from the related literature are in the direction of 2D MOT. These methods are based on a succession of detection and tracking steps: consecutive detections that are similarly classified are linked together to determine trajectories. A significant challenge comes from the inevitable presence of noise in the acquired images, which may adversely change the features of similar objects across multiple frames. Consequently, the computation of robust features is an important aspect of object detection. Features are representative of a wide array of object properties: color, frequency and distribution, shape, geometry, contours, or correlations within segmented objects. Nowadays, the most popular feature detection methods involve supervised learning. Features start out as groups of random values and are progressively refined using machine learning algorithms. Such approaches require appropriate training data and a careful selection of hyperparameters, often through trial-and-error. However, many results from the related literature show that supervised classification and regression methods offer the best results both in terms of accuracy and robustness to affine transformations, occlusion, and noise.

2.1. Methods Using Neural Networks

In terms of classifying objects from images, neural networks have seen a steady rise in popularity in recent years, particularly the more elaborate and complex convolutional and recurrent networks from the field of deep learning. Neural networks have the advantage of being able to learn important and robust features given training data that is relevant and in sufficient quantity. Considering that a significant percentage of automotive sensor data consists of images, convolutional neural networks (CNNs) are seeing widespread use in the related literature, for both classification and tracking problems. The advantage of CNNs over more conventional classifiers lies in the convolutional layers, where various filters and feature maps are obtained during training. CNNs are capable of learning object features by means of multiple complex operations and optimizations. The appropriate choice of network parameters and architecture can ensure that these features contain the most useful correlations that are needed for the robust identification of the targeted objects. While this choice is most often an empirical process, a wide assortment of network configurations exist in the related literature that are aimed at solving classification and tracking problems, with high accuracies claimed by the authors. Where object identification is concerned, in some cases the output of the fully-connected component of the CNN is used, whereas in other situations the values of the convolutional layers are exploited in conjunction with other filtering and refining methods.

2.1.1. Learning Features from Convolutional Layers

Many results from the related literature systematically demonstrate that convolutional features are more useful for tracking than other explicitly computed ones (Haar, Fused Histogram of Oriented Gradients (FHOG), color labeling). An example in this sense is [1], which handles MOT using combinations of values from convolutional layers located at multiple levels. The method is based on the notion that lower-level layers account for a larger portion of the input image and therefore contain more details from the identified objects. This makes them useful, for instance, for handling occlusion. Conversely, top-level layers are more representative of semantics and are useful in distinguishing objects from the background. The proposed CNN architecture uses dual fully-connected components, for higher and lower-level features, which handle instance-level and category-level classification (Figure 2 in [1]). The proper identification of objects, particularly where occlusion events occur, involves the generation of appearance models of the tracked objects. These often result from the appropriate processing of the features learned within convolutional layers.

In [2], the authors note that the output of the fully-connected component of a CNN is not suitable for handling infrared images. Their attempt to directly transfer CNNs pretrained with traditional images for use with infrared sensor data is unsuccessful, since only the information from the convolutional layers seem to be useful for this purpose. Furthermore, the layer data itself require some level of adaptation to the specifics of infrared images. Typically, infrared data offer much less spatial information than visual images. It is much more suited, for example, in depth sensors for gathering distances to objects, albeit at a significantly lower resolution compared to regular image acquisition. As such, convolutional layers from infrared images are used in conjunction with correlation filters to generate a set of weak trackers. This process provides response maps with regard to the targets' locations. The weak trackers are then combined in ensembles which form stronger response maps with a much greater tracking accuracy. The response map of an image is, generally, an intensity image where higher values indicate a change or a desired feature/shape/structure, as the original image is processed by an operator or correlation filter of some kind. By matching or fusing responses from multiple images within a video sequence, one could identify similar objects (i.e., the same pedestrian) across the sequence and subsequently construct their trajectories.

The potential of correlation filters is also exploitable for regular images. These have the potential to boost the information extracted from the activations of convolutional layers. In [3] the authors find that by applying appropriate filters to information drawn from shallow convolutional layers, a level of robustness similar to using deeper layers or a combination of multiple layers can be achieved. In [4], the authors also note the added robustness obtainable by post-filtering convolutional layers. By using particle and correlation filters, basic geometric and spatial features can be deduced for the tracked objects, which, together with a means of adaptively generating variable models, can be made to handle both simple and complex scenes.

An alternative approach can be found in [5], where discriminative correlation filters are used to generate an appearance model from a small number of samples. The overall approach involves feature extraction, post-processing, and the generation of response maps for carrying out better model updates within the neural network. Contrary to other similar results, the correlation filters used throughout the system are learned within a one-layer CNN, which eventually can be used to make predictions based on the response maps. Furthermore, residual learning is employed in order to avoid model degradation, instead of the much more frequently-used method of stacking multiple layers. Other tracking methods learn a similar kind of mapping from samples in the vicinity of the target object using deep regression [6,7], or by estimating and learning depth information [8].

The authors of [9] note that correlation filters have limitations imposed by the feature map resolution. They propose a novel solution where features are learned in a continuous domain, using an appropriate interpolation model. This allows for the more effective resolution-independent compositing of multiple feature maps, resulting in superior classification results.

Methods based on discriminative correlation filters are notoriously prone to excessive complexity and overfitting, and various means are available for optimizing the more traditional methods. The most noteworthy in this sense is [10], who employs efficient convolution operators, a training sample distribution scheme and an optimal update strategy in an attempt to boost performance and reduce the number of parameters. A promising result that demonstrates significant robustness and accuracy is [11], who use a CNN where the first set of layers are shared, as in a standard CNN. These layers then branch into multiple domain-specific ones. This approach has the benefit of splitting the tracking problem into subproblems which are solved separately in their respective layer sets. Each domain has its own training sequences and can be customized to address a specific issue, such as distinguishing a target with specific shape parameters from the background. A similar concept is exploited by [12], i.e., a network with components distinctly trained for a specific problem. In this case, multiple recurrent layers are used to model different structural properties of the tracked objects, which are incorporated into a parent CNN with the same purpose of improving accuracy and robustness. The Recurrent Neural Network (RNN) layers generate what the authors refer to as "structurally-aware feature maps" which, when combined with pooled versions of their non-structurally aware counterparts, significantly improve the classification results.

2.1.2. High-Level Features, Occlusion Handling, and Feature Fusion

Appearance models offer high-level features that are also used to account for occlusion in much simpler and efficient systems. In [13], appearance descriptors are compounded to form an appearance space. With properly-determined metrics, observations having a similar appearance are identified using a nearest-neighbor approach. Switching from image-space to an appearance space seems to effectively handle occlusions, reducing their negative impact at a negligible performance cost.

A possible alternative to appearance-based classification is the use of template-based metrics. Such an approach uses a reference region of interest (ROI) drawn from one or multiple frames and attempts to match it in subsequent frames using an appropriately-constructed metric. Template-based methods perform well for partial detections, thereby accounting for occlusion and/or noise. This is because the template need not be perfectly or completely matched for a successful detection to occur. An example of a template-based method is provided by [14], which involves three CNNs, one for template generation, one dedicated to region searching and one for handling background areas. The method is somewhat similar to what could be achieved by a generative adversarial network (GAN). A "searcher" network attempts to fit multiple subimages within the positive detections provided by the template component while simultaneously attempting to maximize the distance to the negative background component. The candidate subimages generated by the three components are fed through a loss function that is designed to favor candidates closer to template regions than to background ones. Performance-wise, such an approach is claimed to provide impressive framerates and care should be taken when using template or reference-based methods. These are generally suited for situations where there is no significant variation in the overall tone of the frames. Such methods have a much higher failure rate when, for instance, the lighting conditions change during tracking. An example of this phenomenon is when the tracked object moves from a brightly-lit area to a shaded one.

An improvement on the use of appearance and shared tracking information is provided by [15] in the form of a CNN-based single object tracker that generates and adapts the appearance models for multi-frame detection (Figure 3 in [15]). The use of pooling layers and shared features accounts for drift effects caused by occlusion and inter-object dependency. A spatial and temporal attention mechanism is responsible for dynamically discriminating between training candidates based on the level of occlusion. Training samples are weighted based on their occlusion status, which optimizes the training process where both classification accuracy and performance are concerned. Generally speaking, pooling operations have two important effects: on the one hand, the subimage of the feature map is increased, since a pooled feature map contains information from a larger area of the originating image; on the other hand, the reduced size of a pooled map means fewer computational resources are required to process it, which improves performance. The major downside of pooling is that spatial positioning is further diluted with each additional layer. Multiple related papers exploit the so called "ROI pooling", which commonly refers to a pooling operation being applied to the bounding box of an identified object. The resulting reduced representation will hopefully be more robust to noise and geometric variations across multiple frames. ROI pooling is successfully used by [16] to improve the performance of their CNN-based classifier. The authors observe that positioning cues are adversely affected by pooling. A potential solution is to reposition the misaligned ROIs via bilinear interpolation. This reinterpretation of pooling is referred to as "ROI align". The gain in performance is significant, while the authors demonstrate that the positioning of the ROIs is stabilized.

Tracking stabilization is fundamental in automotive application, where effects such as jittering, camera shaking, and spatial/temporal noise commonly occur. Occlusion handling plays an important role in ensuring ROI stability and accuracy. Some authors handle this topic extensively, such as [17], who propose a deep neural network for tracking occluded body parts, by processing features extracted from a VGG19 network. Some authors use different interpretations of the feature concept, adapted to the specifics of autonomous driving. Reference [18] creates custom feature maps by encoding various properties of the detections in raster images (bounding boxes, positions, velocities, accelerations). These images are sent through a CNN that generates raster features that the authors demonstrate to provide more reliable correlations and more accurate trajectories than using features derived directly from raw data.

The idea of tracking robustness and stability is sometimes solvable using image and object fusion. The related methods are referred to as being “instance-aware”. This concept means that a targeted object is matched across the image space and across multiple frames by fusing identified objects with similar characteristics. Reference [19] proposes a fusion-based method that uses single-object tracking to identify multiple candidate instances. Subsequently, it builds target models for potential objects by fusing information from detection and background cues. The models are updated using a CNN, which ensures robustness to noise, scaling, and minor variations of the targets’ appearance. As with many other related approaches, an online implementation offloads most of the processing to an external server leaving the embedded device from the vehicle to carry out only minor, frequent tasks. Since quick reactions of the system are crucial for safe vehicle operation, performance and a rapid response of the underlying software is essential, which is why the online approach is popular in this field. Fusion methods are also applied for multimodal inputs, such as in [20], who propose a model based on a convolutional autoencoder to obtain features from a combination of multiple sensor sources, in order to account for improved environment perception.

Also in the context of ensuring robustness and stability, some authors apply fusion techniques to information extracted from convolutional layers. It has been previously mentioned that important correlations can be drawn from deep and shallow layers that can be exploited together for identifying robust features in the data. This principle is used for instance in [21]. In order to ensure robustness and performance, various features extracted from layers in different parts of a CNN are fused to form stronger characteristics that are affected to a lesser degree by noise, spatial variations, and perturbations in the acquired images. The identified relationships between CNN layers are exploited in order to account for lost spatial information that occurs in deeper layers. The method is claimed to have improved accuracy over the state-of-the-art of the time, which is consistent with the idea of ensuring robustness and low failure rates. Deeper features are more consistent and allow for stronger classification, while shallow features compensate for the detrimental effects of filtering and pooling. This allows for deep features to be better integrated into the spatial context of the images. On a similar note, in [22] features from multiple layers that individually constitute weak trackers are combined to form a stronger one, by means of a hedging algorithm. The practice of using multiple weak methods into a more effective one has significant potential and is based on the principle that each individual weak component contains some piece of meaningful information on the tracked object, while also having useless data mostly found in the form of noise. By appropriately combining the contributions of each weak component, a stronger one can be generated. As such, methods that exploit compound classifiers typically show robustness to variances of illumination, affine transforms, or camera shaking. The downside of such methods is that multiple groups of weak features are needed, which causes penalties in real-time response. Additionally, the fusion algorithm has its own performance-impacting overhead.

Alternative approaches exist which mitigate this to some extent. For example, the use of multiple sensors directly supplies the necessary data, as opposed to relying on multiple features computed from the same camera or pair of cameras. An example in this direction is provided in [23], where an image gallery from a multi-camera system is fed into a CNN in an attempt to solve multi-target multi-camera tracking and target re-identification problems. For correct and consistent re-identification, an observation in a specific image is matched against several ones from other cameras using correlations as part of a similarity metric. Such correlation among images from multiple cameras are learned during training and subsequently clustered to provide a unified agreement between them. Eventually, after a training process that exploits a custom triplet loss function, features are obtained to be further used in the identification process. In terms of performance, the method boasts substantial accuracy considering the multi-camera setup. The idea of compositing robust features from a multi-faceted architecture is further exploited in works such as [24]. A triple-net setup is used to generate features that account for appearance, spatial cues, and temporal consistency.

2.1.3. Ensuring Temporal Coherence

One of the most significant challenges for autonomous driving is accounting for temporal coherence in tracking. Nearly all automotive scenarios involve video and motion across multiple frames. Consequently, handling image sequence data and accounting for temporal consistency are key factors in ensuring successful predictions, accuracy, and reliability. Essentially, solving temporal tracking is a compound problem. On the one hand, it involves tracking objects in single images considering all the problems induced by noise, geometry and the lack of spatial information. On the other hand, it should ensure that the tracking is consistent across multiple frames. That is, assigning correct IDs to the same objects in a continuous video sequence.

This presents a lot of challenges, for instance when objects become occluded in some frames and are exposed in others. In some cases, the tracked objects suffer affine transformations across frames, of which rotation and shearing are notoriously difficult to handle. Additionally, the objects may change shape due to noise, aliasing and other acquisition-related artifacts that may be present in the images. Video is rarely if ever acquired at "high enough" resolution and is in many cases in some lossy compressed format. As such, the challenge is to identify features that are robust enough to handle proper classification and to ensure temporal consistency considering all pitfalls associated with processing video data. This often involves a "focus and context" approach: key targets are identified based on features learned from current frames and from the context of the tracked object. Processing a key frame in a video sequence provides the focus, while the information from previous frames form the context.

For this type of problem, one popular approach is to integrate recurrent components into the classifier, which inherently account for the context provided by a set of elements from a sequence. Neural networks with recurrent layers, such as long short-term memory (LSTM) and gated recurrent units (GRU), are commonly employed in the related literature for the processing of temporal data. When training and exploiting recurrent layers to classify sequences, the results from one frame carry over to the computations that take place for subsequent frames. As such, when processing the current frame, resulting detections also account for what was found in previous frames. For automotive applications, one advantage of neural networks is that they can be trained off-site, while the resulting model can be ported to the embedded device in the vehicle where predictions and tracking can occur at usable speeds. While training a recurrent network or multiple collaborating networks can be a lengthy process, forward-propagating new data can happen quite fast, making these algorithms a good choice for real-time tracking.

Another concept that consistently appears in the related literature is "historical matching". The idea is to carry over part of the characteristics of tracked objects across multiple frames, by building an affinity model from shape, appearance, positional, and motion cues. This is achieved in [25] using dual CNNs with multistep training, which handle appearance matching using various filtering operations and linearly composing the resulting features across multiple timestamps. The notion of determining and preserving affinity is also exploited in [26] where data consisting of frame pairs several timestamps apart are fed into dual VGG networks (models based on convolutional neural networks with an architecture designed for image recognition tasks). The resulting features are permuted and incorporated into association matrices that are further used to compute object affinities. This approach has the benefit of partially accounting for occlusion using only a limited number of frames, since the affinity of an object that is partially occluded in one frame may be preserved if it appears fully in the pair frame.

Ensuring the continuity of high-level features such as appearance models is not a trivial task, and multiple solutions exist. For example [27] uses a CNN modified with a discriminative component intended to correct for temporal errors that may accumulate in the appearance of tracked objects across multiple frames. Discriminative network behavior is also exploited in [28] where selectively trained dual networks are used to generate and correlate appearance with a motion stream. Also, decomposing the tracking problem into localization and motion using multiple component networks is a frequently-encountered solution, further exploited in works such as [29,30]. As such, using two networks that work in tandem is a popular approach and seems to provide accurate results throughout the available literature (Figure 2 in [30]).

In this context, siamese convolutional networks have the ability to learn similarities by comparing features from dual-stream convolutional layers. One example is provided by [31], where appearance and motion are handled by a combination of CNNs that work together within a unified framework. The motion component uses spotlight filtering over feature maps that result from subtracting features drawn from dual CNNs. A space-invariant feature map is then generated using pooling and fusion operations. The other component handles appearance by filtering and fusing features from a different arrangement of convolutional layers. Data from ROIs in the acquired images are passed to both components. Motion responses from one component are correlated with appearance responses from the other. Both components produce feature maps that are composed together to form space- and motion-invariant characteristics to be further used for target identification. As such, a common functionality of such models is to feed the similarities learned among different inputs to subsequent network components that carry out the classification/detection task [32,33].

Some authors take this concept further by employing several network components [34], each of which contributes features exhibiting specific and limited correlations. When joined together, the features form a complete appearance model of the tracked objects. Other approaches map network components to flow graphs, the traversal of which enables optimal cost-function and feature learning [35]. It is worthy of noting that the more complicated the architecture of the classifier, the more elaborate the training process and the poorer the performance. A careful balance should therefore be reached between the complexity of the classifier, the completeness of the resulting features, and the amount of processing and training data needed to produce high-accuracy results. All this should involve a computational cost consistent with the needs of automotive applications. For instance, in [36] the authors propose a lightweight solution where the feature extractor consists in only two convolutional layers, while a careful selection of motion patterns solves the data association problem.

In [37], the idea of object matching from frame pairs is further explored using a three-component setup: a siamese network configuration handles single object tracking and generates short-term cues in the form of tracklet images, while a modified version of GoogleNet Inception-v4 generates re-identification features from multiple tracklets. The third component is based on the idea that there may be a large overlap in the previously-computed features, which are consequently treated as switcher candidates. As a result, a switcher-aware logic handles the situation where IDs of different objects may be interchanged during frame sequences mainly as a result of partial occlusion.

As the difficulty of the tracking problem increases, so does the need to design systems capable of learning increasingly useful and robust features. In this sense, many solutions consist in models that extract features expressing increasingly abstract concepts, which have the potential for greater generalization. Therefore, a lot of effort is directed toward identifying object features that are higher-level, more abstract representations of how the object fits within the overall context of the acquired video sequence. Examples of such concept are the previously-mentioned “affinity”; another is “attention”, where some authors propose neural-network-based solutions for estimating attention and generating attention maps. Reference [15] computes attention features that are spatially and temporally sound using an arrangement of ROI identification and pooling operations. Reference [38] uses attention cues to handle the inherent noise from conventional detection methods, as well as to compensate for frequent interactions and overlaps among tracked targets. A two-component system handles noise and occlusion, and produces spatial attention maps by matching similar regions from pair frames. Temporal coherence is achieved by weighing observations across the trajectory differently, thereby assigning them different levels of attention. This process results is filtering criteria used to successfully account for similar observations while eliminating dissimilar ones. Another noteworthy contribution is [39], where attention maps are generated using reciprocative learning. The input frame is sent back-and-forth through several convolutional layers: in the forward propagation phase classification scores are generated, while the back-propagation produces attention maps from the gradients of the previously-obtained scores. The computed maps are further used as regularization terms within a classifier. The advantage of this approach is its simplicity compared to other similar ones. The authors claim that their method for generating attention features ensures long-term robustness. Other methods that use frame pairs and no recurrent components do not seem to work as well for very long-term sequences. Recently, attention mechanisms have been gaining significant ground for solving temporal consistency problems, since they allow the underlying model the freedom to weigh selective portions of a time-based sequence. Other noteworthy examples of works where attention mechanisms are incorporated into a CNN-based detector are [40,41].

2.1.4. LSTM-Based Methods

Generally, methods that are based on non-recurrent CNN-only approaches are best suited to handle short scenes where quick reactions are required in a brief situation that can be captured in a limited number of frames. Various literature studies show that LSTM-based methods have more potential to ensure the proper handling of long-term dependencies while avoiding various mathematical pitfalls. One example in this sense is the “vanishing gradient” problem, which in practice manifests as a mis-trained network resulting in drift effects and false positives. Furthermore, handling long-term dependencies means having to deal with occlusions to a greater extent than in shorter term scenarios.

Most approaches combine various classifiers that handle spatial and shape-based classification with LSTM components that deal with temporal coherence. An early example of an RNN implementation is [42], which uses an LSTM-based classifier to track objects in time, across multiple frames (Figure 1 in [42]). The authors demonstrate that an LSTM-based approach is better suited to removing and reinserting candidate observations to account for objects that leave/reenter the visible area of the scene. This provides a solution to the track initiation and termination problem based on data associations found in features obtained from the LSTM layers. This concept is exploited further by [43] where various cues are determined to assess long-term dependencies using a dual LSTM network. One LSTM component tracks motion, while the other handles interactions, and the two are combined to compute similarity scores between frames. The results show that using recurrent components to handle lengthy sequences produces more reliable results than other methods based on frame pairs. Some implementations using LSTM layers focus on tracking-while-driving problems, which pose additional challenges compared to most established benchmarks using static cameras. As an alternative to solutions that involve creating models of vehicle behavior, Reference [44] circumvent the need for vehicle modeling by directly inputting sensor measurements into an LSTM network to predict future vehicle positions and to analyze temporal behavior. A more elaborate attempt is [45] where instead of raw sensor data, the authors establish several maneuver classes and feed maneuver sequences to LSTM layers in order to generate probabilities for the occurrence of future maneuver instances. Eventually, multiple such maneuvers can be used to construct the trajectory and/or anticipate the intentions of the vehicles.

Furthermore, increasing the length of the sequence increases accuracy and stability over time, up to a certain limit where the network saturates and no longer improves. A solution to this problem would be to split the features into multiple sub-features, followed by reconnecting them to form more coherent long-term trajectories. This is achieved in [46] where a combined CNN and RNN-based feature extractor generates tracklets over lengthy sequences. The tracklets are split on frames that contain occlusions. A recombination mechanism based on gated recurrent units (GRUs) recombines the tracklet pieces according to their similarities, followed by the reconstruction of the complete trajectory using polynomial curve fitting.

Some authors do further modifications to LSTM layers to produce classifiers that generate abstract high-level features, such as those found in appearance models. A good example in this sense is [47] where LSTM layers are modified to do multiplication operations and use customized gating schemes between the recurrent hidden state and the derived features. The newly-obtained LSTM layers are better at producing appearance-related features than conventional LSTMs, which excel at motion prediction. Where trajectory estimation is concerned, LSTM-based methods exploit the gating that takes place in the recurrent layers, as opposed to regular RNNs, which pass candidate features into the next recurrent iteration without discriminating between them. The filters inherently present in gated LSTMs have the potential to eliminate unwanted feature candidates which may represent unwanted trajectory paths. Candidates which eventually lead to correctly-estimated motion cues are maintained. Furthermore, LSTMs demonstrate an inherent capability to predict trajectories that are interrupted by occlusion events or by reduced acquisition capabilities. This idea is exploited in order to find solutions to the problem of estimating the layout of a full environment from limited sensor data, a concept referred to in the related literature as "seeing beyond seeing" [48]. Given a set of sensors with limited capability, the idea is to perform end-to-end tracking using raw sensor data without the need to explicitly identify high-level features or to have a pre-existing detailed model of the environment. In this sense, recurrent architectures have the potential to predict and reconstruct occluded parts of a particular scene from incomplete or partial raw sensor output. The network is trained with partial data and it is updated through a mapping mechanism that makes associations with an unoccluded scene. Subsequently, the recurrent layers make their own internal associations and become capable of filling in the missing gaps that the sensors have been unable to acquire. Specifically, given a hidden state of the world that is not directly captured by any sensor, an RNN is trained using sequences of partial observations in an attempt to update its belief concerning the hidden parts of the world. The resulting information is used to "unocclude" the scene that was initially only partially perceived through limited sensor data. Upon training, the network is capable of defining its own interpretation of the hidden state of the scene. The previously-mentioned result is elaborated upon by a group that includes the same authors [49]. A similar approach previously applied in basic robot guidance is extended for use in assisted driving. In this case, more complex information can be inferred from raw sensor input, in the form of occupancy maps. Together with a deep network-based architecture, these allow for predicting the probabilities of obstacle presence even in occluded portions within the field of view. In [50], the idea of using LSTM layers to process sensor data is depicted by modeling actor trajectories and activities based on the output of an arrangement on inertial sensors. The proposed neural network learns correlations among sensor outputs and consequently forms an inertial odometry model.

In more recent studies, authors tend to add supplementary processing stages to their LSTM-based models. This additional effort seems to stem from the need to generate and incorporate an increasingly-refined and abstract array of features into the tracking process. As tracking scenarios increase in complexity, the resulting problem space increases in size and dimensionality. This motivates the need for extending an LSTM-centered model by incorporating it into a broader system. An example of such an approach is [51], where sequential dependencies are handled by LSTM layers as in commonly the case in such works. While relying on convolutional feature maps in the initial phases of the tracking pipeline, there is an additional mechanism for preparing selection proposals for the LSTM layers to process. Additionally, the common problem of feature inadequacy and class imbalance in the learning phase is handled by a GAN-based stage where the candidate samples are augmented. It is worth mentioning that, as more and more layers of different types are added to such a system, the reliability of the selected features may increase together with robustness to potential biases in the training data. However, at the same time, there is the risk that training and validating such a system may become a tedious, time consuming task. As the complexity increases, so does the need for extending the training data set and supplement the required computational resources. Other efforts in the direction of producing more usable features involve determining pedestrian intention as suggested by [52]. In this scenario, an LSTM model is used in conjunction with an intention filter to select suitable trajectory offset hypotheses so as to add to the reliability of the predicted result. The use of intention as a defining concept for features is also explored by [53], who enhance LSTM cells by introducing additional speed and correlation components. These components serve to model the more complex interactions required to define intention. In [54], instead of refining feature candidates using additional mechanisms, the authors choose to change the representation of the respective features. Specifically, LSTM layers are reconfigured and repurposed to handle multidimensional hidden states as opposed to the 1D vectors used traditionally. This increases the ability of such layers to accurately model spatial and temporal interactions among pedestrians. Conversely, in [55] the authors adapt an arrangement of LSTM layers to process sparse 3D data structures as opposed to changing internal data representation. The authors choose to model the interactions among pedestrians using graphs and, consequently, graph convolutional networks. Such systems still rely on LSTM layers to encode temporal dependencies. The spatial and sequence-related relationships among the tracked actors (represented as nodes) is modeled by determining connections in the form of graph edges [56,57].

2.1.5. Miscellaneous Neural Network-Based Methods

An interesting alternative to conventional deep learning architectures is the use of GANs, as demonstrated in [58]. GANs train generative models and filter their results using a discriminative component. GANs are notoriously difficult to train, which is one of the reasons why they see seldom use in the related literature. In terms of tracking, GANs alleviate the need to compute expensive appearance features and minimize the fragmentation that typically occurs in more conventional trajectory prediction models. A generative component produces and updates candidate observations, of which the least updated are eliminated. The generative-discriminative model is used in conjunction with an LSTM component to process and classify candidate sequences. This approach has the potential to produce high-accuracy models of human behavior, especially group behavior. At the same time, it is significantly more lightweight than previously-considered CNN-based solutions.

Another "outlier" solution in the related literature is [59], one of the few efforts involving reinforcement learning for MOT applications. The proposed model is split into two parts: a predictive component based on a CNN, which treats pedestrian detections as agents and determines the displacement of a target agent from its initial location; a decision network, which uses the resulting predictions and detections within a deep reinforcement learning network where the actions among the agents and their environment are rewarded so as to maximize their shared utility. Consequently, the collaborative interactions of multiple agents are exploited in order to simultaneously detect and track them more effectively. Other driver-centric reinforcement learning-based solutions determine driving rules for collision avoidance by weighing vehicle paths against potential pedestrian trajectories [60].

2.2. Other Techniques

While the current state-of-the art methods for MOT are mostly neural network-based, there also exist a multitude of other approaches which exploit more traditional, unsupervised means of providing reliable tracking. Neural networks gained popularity in recent years due in no small part to the availability of more powerful hardware, particularly GPUs, which allowed for training models capable of handling realistic scenarios in a reasonable amount of time. Neural networks however have the downside of needing vast amounts of reliable training data. Also, they require a lot of experimentation and trial-and-error before the right design and hyperparameter set is found for a particular scenario. There are, however, situations where training data may not be readily available in sufficient quantity and variety. Such cases call for a more straightforward design and a more intuitive model that can provide reliable tracking without necessarily requiring supervised learning. Neural network models are harder to understand in terms of how they function, and, while as deterministic as their non-neural network-based counterparts, are less intuitive and meant for use in a "black-box" manner. This is where other, more transparent methods come into place.

The tracking problem can be formulated similarly to the neural-network case: given a set of observations/appearances/segmented objects in multiple video frames, the task is to develop a means of determining relationships among these elements across the frames and to come up with a means of predicting their path. Various authors formulate this problem differently, for instance some methods involve determining tracklets in each frame and then assembling object trajectories in a full video sequence by combining tracklets from all or some of the frames [61]. Traditional, non-NN-based approaches, especially non-supervised ones, generally formulate much more straightforward models. Some are based on a graph or flow-oriented interpretation of the tracked scene. Others rely on emitting hypotheses as to the potential trajectories of the tracked targets, or otherwise formulating some probabilistic approach to predicting the evolution of objects in time. It is worth noting that many of the more conventional, unsupervised algorithms from the related literature do not generalize the solution as well as a NN-based method. Consequently, they are usable in a limited number of scenarios, by comparison. Some works attempt to circumvent this problem using evolutionary algorithms as multicriteria optimization methods [62]. However, while capable of covering a significant portion of the problem space, such methods have the downside that the optimal trajectory needs to be periodically recalculated, which can hinder performance especially for on-board-only systems. Also, methods that attempt to account for temporal consistency do not handle time sequences as lengthy as, for instance, an LSTM network. The likely explanation is that an unsupervised method requires far more processing capabilities the more frame elements it is fed. In the case of a properly-trained neural network, the amount of computational resources required does not increase as much with the length of the associated sequence. However, in practice, especially on an embedded device as required in automotive tracking, porting a more conventional method may be more convenient in terms of implementation and platform compatibility than running a pre-trained NN model.

Another important aspect worth mentioning is that conventional methods are much more varied in terms of their underlying algorithms, as opposed to an NN-based architecture which features various arrangements of the same two or three neural network types, with additional processing of layer activations or outputs as the case may be. For this reason, we do not attempt to cover all the approaches ever developed for object tracking, but we rather focus on representative works featuring various successful attempts at MOT.

2.2.1. Traditional Algorithms and Methods Focusing on High-Performance

The Kalman filter is a popular method with many applications in navigation and control, particularly with regard to predicting the future path of an object, associating multiple objects with their trajectories, while demonstrating significant robustness to noise. Generally, Kalman-based methods are used for simpler tracking, particularly in online scenarios where the tracker only accesses a limited number of frames at a time, possibly only the current and previous ones. An example of the use of the Kalman filter is [63], where a combination of the aforementioned filter and the Munkres algorithm as the min-cost estimator is used in a simple setup focusing on performance. The method requires designing a dynamic model of the tracked objects' motion, and is much more sensitive to the type of detector. However the proper parameters are established, the simplicity of the method allows for significant real-time performance.

Similar methods are frequently used in simple scenarios where a limited number of frames are available and the detections are accurate. In such situations, the simplicity of the implementations allows for quick response times even on low-spec embedded client devices. In the same spirit of providing an easy, straightforward method that works well for simple scenarios, Reference [64] provides an approach based on bounding-box regression. Given multiple object bounding boxes from a set of ordered frames, the authors use a regression model to predict the positions of the objects' bounding boxes in following frames. An important restriction of such an approach is that it only successfully detects targets that move only slightly across consecutive frames, making it reliable in scenarios where the frame rate is high enough and relatively stable. Furthermore, a reliable detector is a must in such situations, and crowded scenes with frequent occlusion events are not handled properly. As with the previous approach, this is well suited for easy cases where robust image acquisition is available and performance and implementation simplicity are a priority. Unfortunately, noisy images are fairly common in automotive scenarios where, for efficiency and cost reasons, a compromise may be made in terms of the quality and performance of the cameras and sensors. It is often desirable that the software be robust to noise so as to minimize the hardware costs.

In Reference [65], tracking is done by a particle filter for each track. The authors use the Munkres assignment for bounding boxes within consecutive images for each track. A cost matrix is then generated based on the associations made among bounding boxes from current and previous images. Specifically, the cost of associating two bounding boxes is determined from the Euclidean distance between the centers of the boxes, as well as their size variation. This approach is simple to implement, but the assignment algorithm has an $O(n^3)$ complexity, which is likely too high for real-time tracking.

Various attempts exist for improving noise robustness while maintaining performance, for example in [66]. In this case, the lifetime of tracked objects is modeled using a Markov Decision Process (MDP). The policy of the MDP is determined using reinforcement learning, whose objective is to learn a similarity function for associating tracked objects. The positions and lifetimes of the objects are modeled using transitions between MDP states. Reference [67] also use MDPs in a more generalized scheme, involving multiple sensors and cameras and fusing the results from multiple MDP formulations. Note that Markov models can be limiting when it comes to automotive tracking, since a typical scene with multiple interacting targets does not exhibit the Markov property where the current state only depends on the previous one. In this regard, the related literature features multiple attempts to improve reliability. Reference [68] propose an elaborate pipeline featuring multiview tracking, ground plane projection, maneuver recognition, and trajectory prediction. The method involves an assortment of approaches, which include Hidden Markov Models and Variational Gaussian mixture models. Such efforts show that an improvement over traditional algorithms involves sequencing together multiple different methods, each with its own role. As such, there is the risk that the overall resulting approach may be too fragmented and too cumbersome to implement, interpret, and improve properly.

Works such as [69] attempt to circumvent such limitations by proposing alternatives to tried-and-tested Markov models, in this case in the form of a system that determines behavioral patterns in an effort to ensure global consistency for tracking results. There are multiple ways to exploit behavior in order to guide the tracking process. For instance, a possible solution would rely on learning and minimizing/maximizing an energy function that associates behavioral patterns with potential trajectory candidates. This concept is exemplified by [70], who propose a method based on minimizing a continuous energy function aimed at handling the very large space of potential trajectory solutions. A limited, discrete set of behavior patterns impose limitations on the energy function. While such a limitation offers better guarantees that a global optimum will eventually be reached, it may not allow for a complete representation of the system.

An alternative approach which is also designed to handle occlusions is [71], where the divide-and-conquer paradigm is used to partition the solution space into smaller subsets, thereby optimizing the search for the optimal variant. The authors note that while detections and their respective trajectories can be extracted rather efficiently from crowded scenes, the presence of ambiguities induced by occlusion events may raise significant detection errors. The proposed solution involves subdividing the object assignment problem into subproblems, followed by a selective combination of the best features found within the subdivisions (Figure 3 in [71]). The number and types of the features are variable, thereby accounting for some level of flexibility for this approach. One particular downside is that once the scene changes, the problem itself also changes and the subdivisions need to reoccur and update, therefore making this method unsuitable for scenes acquired from moving cameras.

A similar problem is posed in [61], where it is also noted that complex scenes pose tracking difficulties due to occlusion events and similarities among different objects. This issue is handled by subdividing object trajectories into multiple tracklets and subsequently determining a confidence level for each such tracklet, based on its detectability and continuity. Actual trajectories are then formed from tracklets connected based on their confidence values. One advantage of this method in terms of performance is that tracklets can be added to already-determined trajectories in real-time as they become available without requiring complex processing or additional associations. Additionally, linear discriminant analysis is used to differentiate objects based on appearance criteria. The concept of appearance is more extensively exploited by [72], who use motion dynamics to distinguish between targets with similar features. They approach the problem by determining a dynamics-based similarity between tracklets using generalized linear assignment. As such, targets are identified using motion cues, which are complementary to more well established appearance models. While demonstrating adequate performance and accuracy, it is worth mentioning that motion-based features are sensitive to camera movement and are considerably more difficult to use in automotive situations. Motion assessment metrics that work well for static cameras may be less reliable when the cameras are in motion and image jittering and shaking occur.

The idea of generating appearance models using traditional means is exemplified in [73], who use a combination appearance models learned using a regularized least squares framework and a system for generating potential solution candidates in the form of a set of track hypotheses for each successful detection. The hypotheses are arranged in trees, each of which are scored and selected according to the best fit in terms of providing usable trajectories. An alternative to constructing an elaborate appearance model is proposed by [74], who directly involve the shape and geometry of the detections within the tracking process, therefore using shape-based cost functions instead of ones based on pixel clusters. Furthermore, results focusing on tracking-while-driving problems may opt for a vehicle behavior model, or a kinematic model, as opposed to one that is based on appearance criteria. Examples of such approaches are [75–77], where the authors build models of vehicle behavior from parameters such as steering angles, headings, offset distances, and relative positions. Note that kinematic and motion models are generally more suited to situations where the input consists in data from radar, Light Detection and Ranging (LiDAR) or Global Positioning Systems (GPS), as opposed to image sequences. In particular, attempting to reconstruct visual information from LiDAR point clouds is not a trivial task and may involve elaborate reconstruction, segmentation and registration preprocessing before a suitable detection and tracking pipeline can be designed [78].

Another class of results from related literature follows a different paradigm. Instead of employing complex energy minimization functions and/or statistical modeling, other authors opt for a simpler, faster approach that works with a limited amount of information drawn from the video frames. The motivation is that in some cases the scenarios may be simple enough that a straightforward method that alleviates the need for extended processing may prove just as effective as more complex and elaborate counterparts. An example in this direction is [79] whose method is based on scoring detections by determining overlaps between their bounding boxes across multiple consecutive frames. A scoring system is then developed based on these overlaps and, depending on the resulting scores, trajectories are formed from sets of successive overlaps of the same bounding boxes. Such a method does not directly handle crowded scenes, occlusions or fast moving objects whose positions are far apart in consecutive frames, however it may present a suitable compromise in terms of accuracy in scenarios where performance is detrimental and the embedded hardware may not allow for more complex processing. This is in contrast to high-performance methods that use on-board hardware to provide a lot of the information required for tracking, therefore reducing the reaction time of the underlying system in high-speed scenarios [80]. An additional important consideration for this type of problem is how the tracking method is evaluated.

Most authors use a common, established set of benchmarks which, while having a certain degree of generality, cannot cover every situation that a vehicle might be found in. As such, some authors such as [81] devote their work to developing performance and evaluation metrics and data sets, which allow for covering a wide range of potential problems that may arise in MOT scenarios. As such, the choice in the method used for tracking is as much a consequence of the diversity of situations and events claimed to be covered by the method, as it results from the evaluation performed by the authors. For example, as was the case for NN-based methods, most evaluations are done for scenes with static cameras, which are only partly relevant for automotive applications. The advantage of the methods presented thus far lies in the fact that they generally outperform their counterparts in terms of the required processing power and computational resources, which is a plus for vehicle-based tracking where the client device is usually a low-power solution. Furthermore, some methods can be extended rather easily, as the need may be, for instance by incorporating additional features or criteria when assembling trajectories from individual detections, by finding an optimizer that ensures additional robustness, or, as is already the case with some of the previously-mentioned papers, by incorporating a light-weight supervised classifier in order to boost detection and tracking accuracy. Additionally, the problem of false or malicious information from other traffic participants (for example in a multi-vehicle situation) has the potential to affect the accuracy of such methods. One proposed solution to this issue is to cluster the observations drawn from cooperative tracking according to their reliability and their potential to adversely affect the tracking results [82].

2.2.2. Methods Based on Graphs and Flow Models

A significant number of results from the related literature present the tracking solution as a graph search problem or otherwise model the tracking scene using a dependency graph or flow model. There are multiple advantages to using such an approach: graph-based models tailor well to the multi-tracking problem since, like a graph, it is formed from inter-related nodes each with a distinct set of parameter values. The relationships that can be determined among tracked objects or a set of trajectory candidates can be modeled using edges with edge costs. Graph theory is well understood and graph traversal and search algorithms can be widely found, with implementations readily available on most platforms. Likewise, flow models can be seen as an alternative interpretation of graphs, with node dependencies modeled through operators and dependency functions, forming an interconnected system. Unlike a traditional graph, data from a flow model progresses in an established direction that starts from initial components where acquired data are handled as input; the data then traverse intermediate nodes where they are processed in some manner and end up at terminal nodes where the results are obtained and exploited. Like graphs, flow models allow for loops that implement refinement techniques and in-depth processing via multiple local iterations.

Most methods that exploit graphs and flow models attempt to solve the tracking problem using a minimum path or minimum cost-type approach. An example in this sense is [83], where multi-object tracking is modeled using a network flow model subjected to min-cost optimization. Each path through the flow model represents a potential trajectory, formed by concatenating individual detections from each frame. Occlusion events are modeled as multiple potential directions arising from the occlusion node and the proposed solution handles the resulting ambiguities by incorporating pairwise costs into the flow network.

A more straightforward solution is presented by [84], who solve multi-tracking using dynamic programming and formulate the scenario as a linear program. They subsequently handle the large number of resulting variables and constraints using k-shortest paths. One advantage of this method seems to be that it allows for reliable tracking from only four overlapping low resolution low fps video streams, which is in line with the cost-effectiveness required by automotive applications.

Another related solution is [85], where a cost function is developed from estimating the number of potential trajectories as well as their origins and end frames. Then, the scenario is handled as a shortest-path problem in a graph, which the authors solve using a greedy algorithm. This approach has the advantage that it uses well-established methods, therefore affording some level of simplicity to understanding and implementing the algorithms.

In [86], a similar graph-based solution divides the problem into multiple subproblems by exploring several graph partitioning mechanisms and uses greedy search based on Adaptive Label Iterative Conditional Modes. Partitioning allows for successful disassociation of object identities in circumstances where said identities might be confused with one another. Also, methods based on solution space partitioning have the advantage of being highly scalable, therefore allowing fine tuning of their parameters in order to achieve a trade-off between accuracy and performance. Multiple extensions of the graph-based problem exist in the related literature, for instance, when multiple other criteria are incorporated into the search method. Reference [87] incorporate appearance and motion-based cues into their data association mechanism, which is modeled using a global graph representation and makes use of generalized minimum clique graphs to locate representative tracklets in each frame. Among other advantages, this allows for a longer time span to be handled, albeit for each object individually.

Another related approach is provided in [88], where the solution consists in a collaborative model which makes use of a detector and multiple individual trackers, whose interdependencies are determined by finding associations with key samples from each detected region in the processed frames. These interdependencies are further exploited via a sample selection method to generate and update appearance models for each tracker.

As extensions of the more traditional graph-based models that use greedy algorithms to search for suitable candidate solutions and update the resulting models in subsequent processing steps, some authors handle the problem using hypergraphs. These extend the concept of classical graphs by generalizing the role of graph edges. In a conventional graph an edge joins two nodes, while in a hypergraph edges are sets of arbitrary combinations of nodes. Therefore an edge in a hypergraph connects to multiple nodes, instead of just two as in the traditional case. This structure has the potential to form more extensive and complete models using a singular unified concept and to alleviate the need for costly solution space partitioning or subdivision mechanisms. Another use of the hypergraph concept is provided by [89], who build a hypergraph-based model to generate meaningful data associations capable of handling the problem of targets with similar appearance and in close proximity to one-another, a situation frequently encountered in crowded scenes. The hypergraph model allows for the formulation of higher-order relationships among various detections, which, as mentioned in previous sections, have the potential to ensure robustness against simple transformations, noise, and various other spatial and temporal inaccuracies. The method is based on grouping dense neighborhoods of tracklets hierarchically, forming multiple layers which enable more fine-grained descriptions of the relationships that exists in each such neighborhood. A related but much more recent result [90] is also based on the notion that hypergraphs allow for determining higher order dependencies among tracklets, but in this case the parameters of the hypergraph edges are learned using a structural support vector machine (SSVM), as opposed to being determined empirically. Trajectories are established as a result of determining higher order dependencies by rearranging the edges of the hypergraph so as to conform to several constraints and affinity criteria (Figure 1 in [90]). While demonstrating robustness to affine transforms and noise, such methods still cannot handle complex crowded scenes with multiple occlusions and, compared to previously-mentioned methods, suffer some penalties in terms of performance, since updating the various parameters of hypergraph edges can be computationally costly.

2.3. Discussion

Most of the results from the available literature focus on generating abstract, high-level features of the observations found in the processed images, since, generally, the more abstract the feature, the more robust it should be to transformations, noise, drift, and other undesired artifacts and effects. Most authors rely on an arrangement of CNNs where each component has a distinct role in the system, such as learning appearance models, geometric and spatial patterns, or learning temporal dependencies. It is worth noting that a strictly CNN-based method needs substantial tweaking and careful parameter adjustment before it can accomplish the complex task of consistent detection in space and across multiple frames.

LSTM-based architectures seem to show more promising results for ensuring long-term temporal coherence, since this is what they were designed for, while also being simpler to implement and train. For the purposes of autonomous driving, an LSTM-based method shows promise, considering that training should happen offline and that a heavily-optimized solution is needed to achieve a real-time response. Designing such a system also requires a fair amount of trial-and-error since currently there is no well established manner to predict which network architecture is suited to a particular purpose.

One particularly promising direction for automotive tracking are solutions that make use of limited sensor data and that are able to efficiently predict the surrounding environment without requiring a full representation or reconstruction of the scene. These approaches circumvent the need for lengthy video sequences, heavy image processing and the computation of complicated object features while being especially designed to handle occlusion and objects outside of the immediate field of view. As such, where automotive tracking is concerned, the available results from the state-of-the art seem to suggest that an effective solution would make use of partial data while being able to handle temporal correlations across lengthy sequences using an LSTM component.

Other, unsupervised approaches not reliant on neural networks offer a more straightforward model with an easier implementation. The downside often consists in the lack of generalization that such systems are capable of. The features required for detection and stability often have to be manually established, as opposed to supervised methods that can learn meaningful features on their own. The choice in terms of the most useful and reliant tracking model ultimately rests on many factors, among which we mention: the size and complexity of the problem, the availability of training data, the available computational resources, and, ultimately, the requirements in terms of accuracy, coherence, and stability.

We summarize our findings in Table 1, where we classify the works referenced in this Section according to the main method and the general approach used throughout. We choose to feature distinctive categories for solutions relying mainly on convolutional layers and on recurrent ones. Other methods are grouped into their own category. This choice is motivated by the fact that, as of yet, deep neural networks consistently show the most promise for the problems described throughout this Section. Many authors have found inventive and effective solutions to tracking problems using neural network-based models, since they offer the most robust features while being natively designed to solve focus-and-context problems in data sequences.

Table 1. Classification of tracking solutions from existing works.

Category	Strenghts/Weaknesses	Overall Approach	Contributions
Methods based on convolutional neural networks	Strengths: - good at learning spatial, shape and geometric features - reduced computational load compared to regular neural networks - translation invariance Weaknesses: - cannot determine temporal dependencies in sequence data without additional mechanisms	Tracking from features directly learned by simple single-stream convolutional layers	[8,16,18]
		Dual-stream CNNs with data associations performed by additional model components	[6,19,28,37]
		Tracking using responses from convolutional features processed through correlation filters	[2–5,22,23]
		Multistream CNNs that determine similarities between multiple ROIs and target templates	[14]
		Models that determine appearance descriptors or that generate appearance representations from convolutional features	[1,13,25,27,32]
		Convolutional models that account for temporal coherence using a multi-network pipeline	[24]
		Tracking from features learned by fusing responses from dual stream convolutional layers (Siamese CNNs)	[26,30,31,38]
		Models that generate features from convolutional layers and use attention mechanisms for temporal coherence and matching	[15,39–41]
		Multi-stream convolutional layers used for detecting pedestrian poses	[17]
		Siamese networks combining convolutional features with complementary features from image processing	[33]
Methods based on recurrent neural networks	Strengths: - good for processing data series - good for learning temporal features and dependencies, and for ensuring temporal coherence Weaknesses: - cannot deduce interactions without additional mechanisms - generally more difficult to train	Models based on LSTM cell configurations that directly predict vehicle/obstacle occupancy	[44,49]
		Model for motion prediction based on vehicle maneuvers	[45]
		LSTM-based architectures that generate appearance and motion models and learn interaction information over extended sequences	[43,47,51]
		Multi-layer GRU-based architecture which splits and reconnects tracklets generated from convolutional features	[46]
		Basic RNN that encodes information from multiple frame sequences	[42,48]
		LSTM layers that focus on learning and interpreting actor intentions	[52,53]
		LSTM-based object detection and tracking adapted for sequences of higher-dimensional data	[55]
		LSTM models that encode relationships between actors using graph representations	[56,57]
		LSTM model that uses multidimensional internal representations of data sequences	[54]
Methods not relying on neural networks	Strengths: - designing a working model is more straightforward compared to neural networks - most are not training data-dependent Weaknesses: - traditional, classic methods do not model sequence dependencies as effectively as many RNN-based solutions	Models that represent and predict actor relationships using flow-networks and graphs	[83,85,89,90]
		Models relying on geometric representations, kinematics and pose estimations	[74,75,77,91]
		Models that ensure detection coherence using adaptive partitioning of the problem space	[71,86]
		Methods relying on Markov models and Markov decision processes	[66–68]
		Methods that build appearance models and/or use appearance similarity metrics	[72,73,87,88]
		Methods using a multi-stage tracking pipeline incorporating filtering, segmentation, clustering and/or data association	[76,78]
		Methods relying on lightweight filtering and optimization for high-speed high-performance applications	[63,64,80,84]

3. Trajectory Prediction Methods

In order to navigate through complex traffic scenarios safely and efficiently, autonomous cars should be able to predict the way in which the other traffic participants will behave in the near future with a sufficient degree of accuracy. The prediction of their motion is especially difficult because there are usually multiple interacting agents in a scene. Also, driver behavior is multi-modal, e.g., in different situations, from a certain common past trajectory, several different future trajectories may emerge. An autonomous car must also find a balance between the safety of people involved (its own passengers and other human drivers, or pedestrians) and choosing an efficient speed to reach its destination, without any perturbations to existing traffic. Predicting the future state of its environment is particularly important when the autonomous vehicle should act proactively, e.g., when changing lanes, overtaking other traffic participants and managing intersections [45].

Other difficulties come from the requirement that such a system must be prepared to handle rare, exceptional situations. However, because of the great number of possibilities involved, it should take into account only a reasonable subset of possible future scene evolutions [92] and often, it is important to identify the most probable solution [93].

Reasoning about the intentions of other drivers is a particularly helpful ability. Trajectory prediction can be treated on two different levels of abstraction. On the higher level, one can identify the overall intentions regarding a discrete set of possible actions, e.g., changing a lane or moving left or right in an intersection. On the lower level, one can predict the actual continuous trajectories of the road users [94].

Trajectory prediction needs to be precise but also computationally efficient [95]. The latter requirement can be satisfied by recognizing some constraints that reduce the size of the problem space. For example, the current speed of a vehicle affects its stopping time or the allowed curvature of its future trajectory so as to maintain the stability of the vehicle. Even if each driver has his/her own driving style, it is assumed that traffic rules will be obeyed, at least to some extent, and this will constrain the set of possible future trajectories [93].

A recent white paper [96] states that a solution for the prediction and planning tasks of an autonomous car may consider a combination of the following properties:

- *Predicting only a short time into the future.* Given the probabilistic nature of trajectory prediction, the farther one predicts into the future, the less certain the results become. Moreover, the probability distribution of the predicted trajectories disperses and thus becomes less useful altogether;
- *Relying on physics where possible.* Machine learning models, e.g., deep networks, can be used to predict trajectories, but they can suffer from approximation errors. Especially in simple, non-interactive scenarios, e.g., when the vehicles have constant speeds or accelerate/decelerate in a foreseeable manner, using physics-based extrapolations can provide more precise results. Also, each type of vehicle may have its own dynamics, so the identification of the vehicle class before prediction is a necessary initial step;
- *Considering whether road users obey traffic rules.* The autonomous car may plan as if the other traffic participants observed the imposed traffic rules, e.g., cars stopping at red lights or pedestrians not crossing the street in forbidden areas. However, defensive safety measures must be in place to prevent accidents with the so-called "vulnerable" road users;
- *Recognizing particular traffic situations.* For example, the behavior of traffic participants caught in a traffic jam differs from their behavior in flowing traffic.

Further, Reference [96] asserts that the self-driving car system should be prepared not only for the worst-case illegal behavior of the other traffic participants, but also for their worst-case legal behavior. The prediction system should be able to learn what the "reasonable" conduct of the other drivers may be in various circumstances. This may also depend on local conditions, such as different "driving cultures" in different countries.

3.1. Problem Description

To tackle the trajectory prediction task, one needs to have access to real-time data from sensors such as LiDAR, radar, or camera, and to a functioning system that allows detection and tracking of traffic participants in real-time. Examples of pieces of information that describe a traffic participant are: the bounding box, position, velocity, acceleration, heading, and yaw rate, i.e., the change in the heading angle. It may also be needed to have mapping data of the area where the ego car is driving, i.e., road and crosswalk locations, lane directions, and other relevant map-related information. Past and future positions are represented in an ego car-centric coordinate system. Also, one needs to model the static context with road and crosswalk polygons, as well as lane directions and boundaries [97]. An example of available information on which the prediction module can operate is presented in Figure 1.3 in [98].

More formally, prediction can be defined as reasoning about probable outcomes based on past observations [99]. Let X_t^i be a vector with the spatial coordinates of agent i at observation time t, with $t \in \{1, 2, ..., T_{obs}\}$, where T_{obs} is the present time step in the series of observations. The past trajectory of agent i is a sequence $X^i = \{X_1^i, X_2^i, ..., X_{T_{obs}}^i\}$. Based on the past trajectories of all agents, one needs to estimate the future trajectories of all agents, i.e., $\hat{Y}^i = \{\hat{Y}_{T_{obs}+1}^i, \hat{Y}_{T_{obs}+2}^i, ..., \hat{Y}_{T_{pred}}^i\}$.

It is also possible to first generate the trajectories in the Frenet coordinate system along the current lane of the ego vehicle, and then convert it to the Cartesian coordinate system [93]. The Frenet coordinate system is useful to simplify the motion equations when cars travel on curved roads. It consists of longitudinal and lateral axes, denoted as s and d, respectively. The curve that goes through the center of the road determines the s axis and indicates how far along the car is on the road. The d axis indicates the lateral displacement of the car. d is 0 on the center of the road and its absolute value increases with the distance from the center. Also, it can be positive or negative, depending on the side of the road.

3.2. Classification of Methods

There are several classification approaches presented in the literature regarding trajectory planning methods.

An online tutorial [100] distinguishes the following categories:

1. *Model-based approaches.* They identify common behaviors of the vehicle, e.g., changing lane, turning left, turning right, determining the maximum turning speed, etc. A model is created for each possible trajectory the vehicle can follow and then probabilities are computed for all these models. One of the simplest approaches to compute the probabilities is the autonomous multiple modal (AMM) algorithm. First, the states of the vehicle at times $t - 1$ and t are observed. Then the process model is computed at time $t - 1$ resulting in the expected states for time t. Then the likelihood of the expected state with the observed state is compared, and the probability of the model at time t is computed. Finally, the model with the highest probability is selected;
2. *Data-driven approaches.* In these approaches a black box model (usually a neural network) is trained using a large quantity of training data. After training, the model is applied to the observed behavior to make the prediction. The training of the model is usually computationally expensive and is made offline. On the other hand, the prediction of the trajectories, once the model is trained, is quite fast and can be made online, i.e., in real-time. Some of these methods also employ unsupervised clustering of trajectories using, e.g., spectral clustering or agglomerative clustering, and define a trajectory pattern for each cluster. In the prediction stage, the partial trajectory of the vehicle is observed, it is compared with the prototype trajectories, and the trajectory most similar to a prototype is predicted.

A survey [101] proposes a different classification based on three increasingly abstract levels:

1. *Physics-based motion models.* They apply the laws of physics to estimate the trajectory of a vehicle, by considering inputs such as steering, acceleration, weight, and even the coefficient of friction of the pavement in order to predict outputs such as position, speed, and heading. *Challenges* are related to noisy sensors and sensitivity to initial conditions. *Methods* include Kalman filters and Monte Carlo sampling. *Advantages.* Such models are very often used in the context of safety, as classic fail-safe methods when more sophisticated approaches, such as those using machine learning, are utilized for prediction. They can also be employed in situations that lack intricate interactions between road users. These models do not have to be very simple, as they can include detailed representations of vehicle kinematics, road geometry, etc. *Disadvantages.* They are usually appropriate for short-term predictions, e.g., less than a second, because they cannot predict maneuvers that aim to accomplish higher level goals, e.g., slowing down to prepare to turn in an intersection or because the vehicle in front is expecting a pedestrian to cross the street;
2. *Maneuver-based motion models.* They try to estimate the series of maneuvers that the cars perform on their way, but consider each vehicle to be deciding independently from the other traffic participants. These models attempt to identify such maneuvers as early as possible, and then assume that the maneuvers continue into the near future and estimate the corresponding trajectories. They use either prototype trajectories or maneuver intention estimation. *Challenges* are related to occlusions and the complexity of intentions. *Methods* include clustering, hidden Markov models, and reinforcement learning. *Advantages.* The identified maneuvers serve as a priori information or evidence that conditions future motion. Therefore, they tend to be more reliable than the physics-based ones and their predictions remain relevant for longer periods of time. *Disadvantages.* Because of the independence assumption, these models cannot handle the ways in which the maneuvers of a car influence the behavior of its neighbors. The interactions between traffic participants can be strong in scenarios with a high density of agents, e.g., intersections, possibly with specific priority rules. By ignoring the inter-agent interactions, these models tend to provide less accurate interpretations of such situations;
3. *Interaction-aware motion models.* This is the most general class of models, where the maneuvers of the vehicles are considered to be influenced by those of their neighboring road users. These models use prototype trajectories or dynamic Bayesian networks. *Challenges* refer to the ability to detect interactions and to a possible combinatorial explosion. *Methods* include coupled hidden Markov models, dynamically-linked hidden Markov models, and even rule-based systems. *Advantages.* The inclusion of inter-agent dependencies contributes to a better understanding of the situation. On the one hand, they facilitate longer-term predictions compared to physics-based models. On the other hand, they can be more reliable than maneuver-based models. *Disadvantages.* Because they often have to compute all possible trajectories, they can be inefficient from the computational point of view. Therefore, they may not be appropriate for real-time use cases.

The higher the level of abstraction of a prediction model, the more computationally expensive the model tends to become. Therefore, algorithms have been proposed that focus only on the most plausible trajectories. Also, the performance of the prediction methods are highly coupled with risk estimation possibilities. Therefore, the authors of [101] consider that successful approaches in this area should consider both vehicle motion modeling and risk estimation.

A classification somewhat similar with the previous two is mentioned in [102], which distinguishes the following motion prediction categories of methods:

1. *Learning-based motion prediction*: learning from the observation of the past movements of vehicles in order to predict the future motion;
2. *Model-based motion prediction*: using motion models;
3. *Motion prediction with a cognitive architecture*: trying to reproduce human behavior.

In the rest of this section, we present some specific approaches classified by their main prediction "paradigm", namely neural networks and other methods, most of which use some kind of stochastic representation of the agents' behavior in the environment. This is especially useful since some works use the same model to address different abstraction levels of the trajectory prediction task.

3.3. Methods Using Neural Networks

Many of the approaches presented in the literature that are based on neural networks use either recurrent neural networks (RNNs), which explicitly take into account a history composed of the past states of the agents, or simpler convolutional neural networks (CNNs). Other authors use conditional variational autoencoders (CVAEs) or more recent methods such as generative adversarial networks (GANs) and attention mechanisms.

A generative system is DESIRE [99], which has the goal of predicting the future locations of multiple interacting agents in dynamic (driving) scenes. It can handle the multi-modal nature of the prediction, i.e., for the same set of inputs, the predicted outputs may have several distinct values (a one-to-many mapping). It also takes into account the scene context and the interactions between traffic participants. It uses a single end-to-end neural network model, which the authors report to be computationally efficient. Using a deep learning framework, DESIRE can rank and refine the set of generated trajectories by considering the long-term future values, i.e., the sum of discounted rewards.

The corresponding optimization problem tries to maximize the potential future reward of the prediction, using the following mechanisms (Figure 2 in [99]):

1. *Diverse sample generation:* A conditional variational auto-encoder (CVAE) is used to capture the multi-modal nature of future trajectories. It uses stochastic latent variables which can be sampled to generate multiple possible future hypotheses for a single set of past information. The CVAE is combined with an RNN that encodes the past trajectories, to generate hypotheses using another RNN;
2. *Inverse optimal control (IOC)-based ranking and refinement:* After including the context and the interactions, the most likely trajectories are identified using potential future rewards, similar to inverse optimal control (IOC) or inverse reinforcement learning (IRL). The agents maximize long-term values. The authors believe that in this way the generalization capabilities of the model are improved and the model is more reliable for longer-term predictions. Since a reward function is difficult to design for general traffic scenarios, it is learned by means of IOC. The RNN model assigns rewards to each prediction hypothesis and assesses its quality based on the accumulated long-term rewards. In the testing phase, there are multiple iterations in order to obtain more accurate refinements of the future prediction;
3. *Scene context fusion:* This module aggregates the agent interactions and the context encoded by a CNN. Then this information is passed to an RNN scoring module which computes the rewards.

In [103], a method to predict the trajectories of the neighboring traffic participants is proposed using a long short-term memory (LSTM) network, with the goal of taking into account the relationship between the ego car and surrounding vehicles. The LSTM is a type of recurrent neural network (RNN) capable of learning long-term dependencies. Generally, an RNN has a vanishing gradient problem. An LSTM is able to deal with this through a forget gate, designed to control the information between the memory cells in order to store the most relevant previous data. The proposed method considers the ego car and four surrounding vehicles. It is assumed that drivers generally pay attention to the relative distance and speed with respect to the other cars when they intend to change a lane. Based on this assumption, the relative amounts between the target and the four surrounding vehicles are used as the input of the LSTM network. The feature vector $\mathbf{x}_t$ at time step t is defined by twelve features: lateral position of target vehicle, longitudinal position of target vehicle, lateral speed of target vehicle, longitudinal speed of target vehicle, relative distance between target and preceding vehicle, relative speed between target and preceding vehicle, relative distance between target and following vehicle, relative speed between target and following vehicle, relative distance between target and lead vehicle, relative speed between target and lead vehicle, relative distance between target and ego vehicle, and relative speed between target and ego vehicle. The input vector of the LSTM network is sequence data with $\mathbf{x}_t$'s for past time steps. The output is the feature vector at the next time step $t + 1$. A trajectory is predicted by iteratively using the output result of the network as the input vector for the subsequent time step.

In [44], an efficient trajectory prediction framework is proposed, which is also based on an LSTM. This approach is data-driven and learns complex behaviors of the vehicles from a massive amount of trajectory data. The LSTM receives the coordinates and velocities of the surrounding vehicles as inputs and produces probabilistic information about the future positions of the traffic participants on an occupancy grid map (Figure 1 in [44]). The proposed method is reported to have better prediction accuracy than Kalman filtering.

The occupancy grid map is widely adopted for probabilistic localization and mapping. It reflects the uncertainty of the predicted trajectories. In [44], the occupancy grid map is constructed by partitioning the range under consideration into several grid cells. The grid size is determined such that a grid cell approximately covers a quarter of a lane to recognize the movement of the vehicles on the same lane, as well as the lengths of the vehicles (Figure 3 in [44]).

When predictions are needed for different time ranges, e.g., Δ = 0.5, 1, 2 s, the LSTM is trained independently for each time range. The LSTM produces the probability of occupancy for each grid cell. Let (x, y) be the identifier of a cell in the occupancy grid. Then the softmax layer in LSTM i computes the probability $P_o(i_x, i_y)$ for the grid element (i_x, i_y). Finally, the outputs of the n LSTMs are combined using $P_o(i_x, i_y) = 1 - \prod_{i=1}^{n} \left(1 - P_o^{(i)}(i_x, i_y)\right)$. The probability of occupancy $P_o(i_x, i_y)$ summarizes the prediction of the future trajectory for all n vehicles in the single map.

Alternatively, the same LSTM architecture can be used to directly predict the coordinates of a vehicle as a regression task. Instead of using the softmax layer to compute probabilities, the system can produce two real coordinate values x and y.

In [45], another LSTM model is described for interaction-aware motion prediction. Confidence values are assigned to the maneuvers that are performed by vehicles. Based on them, a multi-modal distribution over future motions is computed. More specifically, the model computes probabilities for each type of maneuver, based on six maneuver classes. The input to the LSTM is represented by the past positions of the ego car and its neighbors, and the geometry of the road lanes.

Social LSTM [104], used for predicting the trajectory of pedestrians, uses LSTM with a social pooling layer which allows neighbors, up to a certain distance, to exchange information. The hidden states of their corresponding LSTMs are pooled together and used as an input for the following prediction step.

Taking into account the time constraints of a real-time system, Reference [97] uses a simple feed-forward CNN architecture for the prediction task. The authors use an RGB image to represent the scene context. However, a vector of velocity, acceleration, and yaw rate can also be included. In this case, this vector is concatenated with the flattened output of the CNN. Then, these aggregated features are sent to a fully connected layer.

A similar approach is used in [18], which predicts multiple possible trajectories together with their probabilities. The context is also encoded as an image that is passed to a CNN. Given the raster image and the state estimates of agents at a time step, the CNN is used to predict a multitude of possible future state sequences, as well as the probability of each sequence.

As part of a complete software stack for autonomous driving, NVIDIA created a system based on a CNN, called PilotNet [105], which outputs steering angles given images of the road ahead. This system is trained using road images paired with the steering angles generated by a human driving a car that collects data. The authors identified the elements of the road image that have the greatest effect on the steering decision. It seems that in addition to learning the obvious features such as lane markings, edges of roads and other cars, the system learns more subtle features that would be hard to anticipate and program by engineers, e.g., bushes lining the edge of the road and atypical vehicle classes, while ignoring structures in the camera images that are not relevant to driving. This capability is derived from data without the need of hand-crafted rules.

In [94], the authors propose a learnable end-to-end model with a deep neural network that reasons about both high level behavior and long-term trajectories. Inspired by how humans perform this task, the network exploits motion and prior knowledge about the road topology in the form of maps containing semantic elements such as lanes, intersections, and traffic lights. The so-called IntentNet is a CNN that outputs three types of variables in a single forward pass: the detection scores for vehicle and background classes, the action probabilities corresponding to the discrete intentions, and bounding box regressions in the current and future time steps representing the intended trajectory. This design enables the system to propagate uncertainty through the different components and is reported to be computationally efficient.

A sequence-to-sequence CNN architecture is also used in [95] for an end-to-end trajectory prediction model. The authors say that the results are comparable to those of other, more complex approaches using LSTMs. Trajectory histories are embedded by means of a fully connected layer. Stacked convolutional layers are used to learn temporal dependencies in a consistent manner. Then, the features from the final convolutional layer are passed through a fully-connected layer to simultaneously generate all predicted positions. The authors report that the results when only one time step at a time is predicted are worse than the results when all future times are predicted at the same time.

The CoverNet model [106] uses a CNN in combination with a trajectory set generated from the input state containing, e.g., speed, acceleration, and yaw rate. The image features pass though some fully-connected layers and produce probabilities for each mode using softmax.

The Y-net model [107] uses the U-Net architecture [108] for the semantic segmentation of the input image. It also computes a distribution for the future trajectories, where the sampled points are clustered using k-means [109].

The TraPHic model [110] uses a hybrid LSTM-CNN network. The trajectory information is passed through LSTMs to construct three maps for the horizon, neighbors, and ego vehicle. The first two are further passed through different CNNs and concatenated with the ego car tensor. The resulting latent representations are passed through another LSTM to predict the ego trajectory.

The EvolveGraph approach [111] uses graphs to model the behavior of heterogeneous agents. It proposes an observation graph, fully connected, to represent the agents in the scene, and an interaction graph for the agent–agent and agent–context interactions. It also employs an encoder–decoder technique, with the encoder using softmax for edge classification and the decoder generating a Gaussian mixture distribution for prediction.

The TNT model [112] uses VectorNet [113], a hierarchical graph neural network, to encode the context of a scene, including road lanes and the position of the traffic signs, beside the trajectories of the agents. The generated set of trajectories is finally filtered to reject similar instances.

Generative approaches are also used. For example, PRECOG [114] follows the idea of identifying high-level goals and condition the predictions based on those. It employs CNNs and RNNs, and also a generative model where the latent variables stand for plausible behavior of the agents in a scene. Reference [115] uses a so-called "conditional flow" variational autoencoder (CF-VAE) that can handle multi-modal conditional distributions. PECNet [116] conditions the predicted trajectories on their endpoints with a conditional variational autoencoder (CVAE) and proposes the "truncation trick", i.e., truncating the sampling distribution with a smaller standard deviation for cases with a few samples to increase the diversity for multi-modal prediction.

Several trajectory prediction models employ Generative Adversarial Networks (GANs) [117]. This architecture has two components: a generator and a discriminator. Instead of training the generator model to directly match the desired data distribution, in this case the generator is trained so that it increases the error rate of the discriminator. In turn, the discriminator tries to distinguish whether a given sample belongs to the true data distribution or is generated by the generator. Both components are engaged in a competition to outsmart the other one, and from this process the generator learns to generate data that resemble the true data distribution. In the domain of trajectory prediction, Social GAN [118] uses a GAN where the generator is composed of an LSTM-based encoder, a context-pooling module, and an LSTM-based decoder. The discriminator uses LSTMs as well.

Other models employ GANs in conjunction with attention mechanisms. AEE-GAN [119] uses attention in order to alleviate the issues given by the complexity of a scene with many heterogeneous interacting agents. For trajectory encoding, it also uses LSTMs. A characteristic feature is the enhanced attention module containing two components: one for recurrent visual attention enforcement (RVAE) and one for social enforcement (SE). The results of the RVAE are visualized with the Grad-Cam method [120], which creates a heatmap with the attention weights of the image pixels.

Another GAN-based architecture is Social Ways [121], which uses three types of losses: discrimination loss for the discriminator, adversarial loss for the generator, and information loss for both. SoPhie [122] uses a GAN module together with a feature extractor module composed of a CNN and several LSTMs encoders, and an attention module with two components: physical attention and social attention.

Attention mechanisms are also used with techniques other than GANs. MHA-JAM [123] uses a CNN for the transformation of the input image and LSTMs for trajectory encoding, whose outputs then pass to several attention heads that provide the data for the LSTM decoders.

Other authors rely on methods to handle graphs explicitly. For example, DAG-NET [124] uses an attention-enhanced graph neural network (GNN) together with a recurrent variational encoder (RVAE) composed of a variational autoencoder (VAE) and a recurrent neural network (RNN). Multiverse [125] is also based on a graph attention network that is used by a convolutional recurrent neural network (ConvRNN). Unlike other approaches, it uses an occupancy grid for a coarse-grained prediction, which is further refined by a fine-grained prediction. Graphs are also employed by Trajectron++ [126], where a scene is represented as a spatio-temporal graph in which nodes denote the agents and edges denote their interactions. A local map is processed by a CNN, trajectories are encoded with LSTMs, multi-modal solutions are handled by means of a CVAE, and the trajectory decoders are based on gated recurrent units (GRUs).

$P2T_{IRL}$ [127] uses similar techniques, i.e., attention, GRU, CNN, but it conditions trajectories by means of a policy learned with inverse reinforcement learning (IRL) on a grid that represents the scene.

3.4. Methods Using Stochastic Techniques

The authors of [128] use Partially Observable Markov Decision Processes (POMDPs) for behavior prediction and nonlinear receding horizon control, or model predictive control, for trajectory planning. The POMDP models the interactions between the ego vehicle and the obstacles. The action space is discretized into: acceleration, deceleration, and maintaining the current speed. For each of the obstacle vehicles, three types of intentions are considered: going straight, turning, and stopping. The reward function is chosen so that the agents make the maximum progress on the road while avoiding collisions. A particle filter is implemented to update the belief of each motion intention for each obstacle vehicle. For the ego car, the bicycle kinematic model is used to update the state.

Article [129] presents a method to predict trajectories in dense city environments. The authors recorded the trajectories of cars comprising over 1000 h of driving in San Francisco and New York. By relating the current position of an observed car to this large dataset of previously exhibited motion in the same area, the prediction of its future position can be directly performed. Under the hypothesis that the car follows the same trajectory pattern as one of the cars in the past at the same location had followed. This non-parametric method improves over time as the amount of samples increases and avoids the need for more complex models.

Paper [93] presents a trajectory prediction method that combines the constant yaw rate and acceleration (CYRA) motion model with maneuver recognition. The maneuver recognition module selects the current maneuver from a predefined set (e.g., keep lane, change lane to the right or to the left, and turn at an intersection) by comparing the center lines of the road lanes to a local curvilinear model of the path of the vehicle. The proposed method combines the short-term accuracy of the former technique and the longer-term accuracy of the latter. The authors use mathematical models that take into account the position, speed, and acceleration of vehicles.

In [130], a method is presented that evaluates the probabilistic prediction of real traffic scenes with varying start conditions. The prediction is based on a particle filter, which estimates the behavior-describing parameters of a microscopic traffic model, i.e., the driving style as a distribution of behavior parameters. This method seems to be applicable for long-term trajectory planning. The driving style parameters of the intelligent driving model (IDM) are continuously estimated, together with the relative motion between objects. By measuring vehicle accelerations, a driving style estimation can be provided from the first detection without the need of a long observation time before performing the prediction. By using a particle filter, it is possible to handle continuous behavior changes with arbitrarily shaped parameter distributions. Forward propagation using Monte Carlo simulation provides an approximate probability density function of the future scene.

In first-order Markov models, a state prediction depends only on the previous observed state, therefore, if the set of past trajectories has common subsequences, the quality of future predictions may be poor. An additional problem is that the data obtained from sensors can be affected by occlusions. The approaches based on Gaussian processes (GPs) overcome this problem by modeling motion patterns as velocity flow fields and provide good performance in the presence of noise. Another advantage is that the predictions have a simple analytical form, and this can be used to assess the risk in traffic scenarios.

As the traffic participants have a mutual influence on one another, their interaction is explicitly considered in [102], which is inspired by an optimization problem. For motion prediction, the collision probability of a vehicle performing a certain maneuver is computed. The prediction is performed based on the safety evaluation and the assumption that drivers avoid collisions. This combination of the intention of each driver and the driver's local risk assessment to perform a maneuver leads to an interaction-aware motion prediction. The authors compute the probability that a collision will occur anywhere in the whole scene, considering that the number of different maneuvers is limited (e.g., lane changes, acceleration, maintaining the speed, deceleration, and combinations), and then the proposed system assesses the danger of possible future trajectories.

The same concept of considering risk is used in [92], which applies a Bayesian approach combined with maneuver-based trajectory prediction. First, a collection of high-level driving maneuvers is assessed for each vehicle with inference in the Bayesian network that models the traffic scene. Then, maneuver-based probabilistic trajectory prediction models are employed to predict the configuration of each vehicle forward in time. The proposed system has three main parts: the maneuver detection, the prediction, and the criticality assessment. In the last part, the individual joint distributions are used together with a parametric free space map-based representation of the environment with probability distribution functions to estimate the probability of a collision between the ego car and any of its neighbors within the prediction horizon via Monte Carlo simulation.

The authors of [68] propose a framework with three interacting modules: a trajectory prediction module based on a motion-based interaction model combined with maneuver-specific variational Gaussian mixture models, a maneuver recognition module based on hidden Markov models (HMMs) for assigning confidence values for maneuvers being performed by surrounding vehicles, and a vehicle interaction module that handles the context of the scene and assigns final predictions by minimizing an energy function based on outputs of the other two modules. The paper defines ten maneuver classes defined by combinations of lane passes, overtakes, cut-ins, and drifts into the ego lane. A corresponding energy minimization problem is set so that the predictions where cars come too close to one another are penalized.

3.5. Mixed Methods

The authors of [131] use a model-based approach relying on vehicle kinematics and an assumption that drivers plan trajectories in such a way as to minimize an unknown cost function. They introduce an IRL algorithm to learn the cost functions of other vehicles in an energy-based generative model. Langevin sampling, a Monte Carlo-based sampling algorithm, is used to directly sample the control sequence. Langevin sampling is shown to generate better predictions with higher stability. It seems that this algorithm is more flexible than standard IRL methods, and can learn higher-level, non-Markovian cost functions defined over entire trajectories. The cost functions are extended with neural networks in order to combine the advantages of both model-based and model-free learning. The study uses both environment structure, in the form of kinematic vehicular constraints, which can be modeled very accurately, and the assumption that human drivers optimize their trajectories according to a subjective cost function.

Multiple deep neural network architectures are designed to learn the cost functions, some of which augment a set of hand-crafted features. The human-crafted cost functions are defined as ten components: the distance to the goal, the distance to the center of the lane, the penalty of collision to other vehicles (inversely proportional to the distance to other vehicles), the L2-norm of acceleration and steering, the L2-norm for the difference of acceleration and steering between two frames, the heading angle to lane, and the difference to the speed limit.

The application of deep learning and mixture models for the prediction of human drivers in traffic is investigated in [132]. The chosen approach is a mixture density network (MDN) where the neural model has LSTM units and the mixture model consists of univariate Gaussian distributions. It applies multi-task learning, in that by sharing the representation between multiple tasks, one enables the model to generalize better. A limitation is that the tasks usually have to be related to some extent. For example, a single neural network can predict both longitudinal and lateral accelerations from the same input, where the first few layers in the network are shared between the two tasks, and then separated into two different layers to produce the final outputs. To capture the intention of the driver, another layer is used in parallel to the motion prediction layer after the LSTM layers. This layer indicates if the driver intends to switch lane and remain there within the next four seconds.

Another algorithm is the Predictron [133]. This architecture is an abstract model based on a Markov reward process, which can be rolled forward for a series of "imagined" planning steps. The predictron is trained end-to-end with the objective that the accumulated values computed in each forward pass should approximate the true value function. It is reported to demonstrate more accurate predictions than conventional deep neural network architectures.

The Monte Carlo Tree Search (MCTS) [134] algorithm can also be used in the context of trajectory planning. It simulates the possible future trajectories starting from the current state, then it evaluates the performance of the leaves using an evaluation function, e.g., a "value network", and finally it uses these evaluations to update the internal values along the trajectory. The architecture presented in [135], called MCTSnet, incorporates the simulation-based search into a neural network, working with vector embeddings. Its advantage is that gradient-based optimization can be used to train the network end-to-end.

However, internal action sequences directing the control flow of the network cannot be differentiated. To address this, an approximate method for credit assignment is proposed that allows to learn this part of the search network from data.

3.6. Discussion

We summarize the works and their specific techniques presented in the trajectory prediction section in Table 2. The table is sorted by publication year in order to give the reader an impression about the overall progress in this field.

The datasets that were used as benchmarks by the papers were also included. We must mention that all authors report experimental results on some kind of datasets, e.g., driving data specifically collected in some areas of the world or synthetic data collected from simulators. However, only the publicly available, real-world datasets were included in the table.

Information about the general capabilities of the methods was included in terms of the ability to provide multi-modal predictions and whether the social context, i.e., the other agents in the scene, was taken into account. Here, we only mention the approaches that handle the interactions and the context explicitly, e.g., with some kind of pooling mechanism or graph representation, not those that just consider an image as the input, which implicitly contains graphical depictions of all agents.

Some of the works also predict the trajectories of pedestrians, not only vehicles. However, we do not distinguish between these case studies, but only mention the main methods which can be used in both situations.

Table 2. Overview of trajectory prediction solutions.

Contribution	Year	Datasets	Multi-Modal	Social Context	Methods
[93]	2013				physics model (CYRA), maneuver recognition
[102]	2013			Yes	interaction-based, risk estimation, discrete maneauvers
[92]	2016		Yes		Bayesian networks, maneuver-based, risk estimation, Monte Carlo simulation
Social LSTM [104]	2016	ETH, UCY		Yes	LSTM
DESIRE [99]	2017	KITTY, Stanford Drone	Yes	Yes	CVAE, GRU, IRL
[44]	2017		Yes (probabilities of occupancy grid cells)	Yes	LSTM, occupancy grid, softmax
PilotNet [105]	2017				CNN, outputs steering angles
[130]	2017		Yes		particle filter, IDM, driving style estimation, Monte Carlo simulation
Predictron [133]	2017				Markov reward process, DNN (fully-connected deep neural network)
[103]	2018	I-80		Yes (only four neighbors)	LSTM
[45]	2018	NGSIM, I-80	Yes	Yes	LSTM, maneuver-based
[97]	2018				CNN
[18]	2018		Yes		CNN

Table 2. *Cont.*

Contribution	Year	Datasets	Multi-Modal	Social Context	Methods
IntentNet [94]	2018		Yes		CNN, intention-based, discrete intention set
[95]	2018	ETH, UCY			CNN
[128]	2018		Yes		POMDP, particle filter, discrete states and actions
[129]	2018				probabilistic sampling
[68]	2018			Yes	Gaussian mixture models, HMM, discrete maneauvers
[132]	2018	NGSIM			DNN, MDN, LSTM
MCTSNet [135]	2018				DNN, vector embeddings, Monte Carlo tree search
Social GAN [118]	2018	ETH, UCY	Yes	Yes	GAN, LSTM
[131]	2019	NGSIM			IRL, Langevin sampling, DNN
PRECOG [114]	2019	nuScenes	Yes	Yes	GRU, CNN, generative model
Social Ways [121]	2019	ETH, UCY	Yes	Yes	GAN, LSTM, generative model, attention
SoPhie [122]	2019	ETH, UCY, Stanford Drone	Yes	Yes	GAN, attention, CNN, LSTM
TraPHic [110]	2019	NGSIM		Yes	LSTM-CNN hybrid network
MHA-JAM [123]	2020	nuScenes	Yes	Yes	multi-head attention, LSTM, CNN
AEE-GAN [119]	2020	Waymo, Stanford Drone, ETH, UCY	Yes	Yes	attention, GAN, LSTM
CF-VAE [115]	2020	Stanford Drone	Yes	Yes	CF-VAE
CoverNet [106]	2020	nuScenes	Yes		softmax for discrete trajectory set
DAG-NET [124]	2020	Stanford Drone, SportVU	Yes	Yes	VAE, RNN, attention, GNN
EvolveGraph [111]	2020	Honda 3D, Stanford Drone, SportVU	Yes	Yes	graphs, GRU, Gaussian mixture
Multiverse [125]	2020	VIRAT/ActEV	Yes		ConvRNN, occupancy grid, graph attention network
P2T$_{IRL}$ [127]	2020	Stanford Drone	Yes		IRL, attention, GRU, CNN
PECNet [116]	2020	Stanford Drone, ETH, UCY	Yes	Yes	CVAE (Endpoint VAE), truncation trick
TNT [112]	2020	Argoverse, Interaction, Stanford Drone	Yes	Yes	VectorNet, MLP (classic multilayer percepton)
Trajectron++ [126]	2020	ETH, UCY, nuScenes	Yes	Yes	LSTM, attention, GRU, CVAE, Gaussian mixture model
Y-Net [107]	2020	Stanford Drone, ETH, UCY	Yes		U-Net, k-means

In general, many authors use CNNs to process the graphical inputs, e.g., camera-based images or maps, and LSTMs for trajectory encoding and decoding. Because of the constraints of real-time requirements, some works also use CNN architectures for prediction [97]. They seem to be able to model complex relations and capture spatial correlations in the data [136]. Some papers state that they are also competitive in modeling temporal data [95], with performance comparable to that of the LSTMs, but with a much simpler internal structure. Multi-modal predictions are often made with some kind of generative models such as CVAE. The methods based on CNNs seem to be more lightweight and fast than those containing LSTM and CVAE components. Still, a large number of approaches combine these techniques in some way.

Other works employ more recent techniques such as GANs and graph representations in conjunction with neural networks. Attention mechanisms also seem promising to distinguish the important features in the context of a complex scene with many interacting agents.

The data themselves may cause difficulties, because a network only learns what is present in the data, and hopefully generalizes well, but there may be situations where the humans do not behave according to previous observations. This is one drawback of using neural networks. However, it seems that the advantages of using data-driven approaches outperform the disadvantages.

Many methods that belong to the stochastic paradigm try to estimate the probabilities of discrete maneuvers. When using, e.g., hidden Markov models, the movement of the traffic participants is evaluated independently, an assumption which is true only for simple scenarios. Gaussian Process regression can quantify uncertainty, but it is also limited in its ability to model complex interactions. For this purpose, other techniques such as Bayesian networks can be used instead, with the disadvantage of an increased computation time and thus a difficulty in handling real-time learning tasks [97].

Although it is possible to do multi-step prediction with a Kalman filter, it cannot be extended far into the future with reasonable accuracy. A multi-step prediction done solely by a Kalman filter was found to be accurate up until 10–15 time steps, after which the predictions diverged and ended up being worse than constant velocity inference [132]. This emphasizes the advantages of data-driven approaches, as it is possible to observe almost an infinite number of variables that may all affect the driver, whereas the Kalman filter relies solely on the physical movement of the vehicle.

Another approach is to learn policies in a supervised way, e.g., imitation learning. The cost function of a human driver can be estimated with inverse reinforcement learning and then a policy can be extracted from the cost function [136]. However, this may again be inefficient for real-time applications [97].

In multi-agent contexts such as those defined by traffic scenarios, since an agent's actions depend on the other agents' actions, uncertainty can propagate to future states with the consequence that an agent completely stops because all possible actions are deemed as unacceptably unsafe. This is known as the "freezing-robot" problem. Deadlock avoidance and multi-objective decision making are very common in practice, e.g., in autonomous robotics [137–139].

Finally, it should be mentioned that in this section, we have addressed the trajectory prediction problem. A related, but distinct problem, is trajectory planning, i.e., finding an optimal path from the current location to a given goal location. Its aim is to produce smooth trajectories with small changes in curvature, so as to minimize both the lateral and the longitudinal acceleration of the ego vehicle. For this purpose, there are several methods reported in the literature, e.g., using cubic spline interpolation, trigonometric spline interpolation, Bézier curves, or clothoids, i.e., curves with a complex mathematical definition, which have a linear relation between the curvature and the arc length, and allow smooth transitions from a straight line to a circle arc or vice versa. Deep reinforcement learning methods [140,141] such as policy gradients [142], deep Q-network [143], actor-critic [144], asynchronous advantage actor-critic [145], proximal policy optimization [146], trust region policy optimization [147], imagination-augmented agents [148], or proximal gradient temporal difference learning [149] can also be used to decide the possible maneuvers that the ego car can make in order to optimize criteria related to risk and efficiency.

4. Conclusions

Learning-based approaches have basically become the norm for autonomous driving problems. Although explicit rule-based methods may have an important advantage in the form of explicit knowledge, hand-crafted rules usually take a considerable amount of effort to devise and validate, and usually do not have satisfactory generalization capabilities because of the great variability of situations that may appear in a driving context. Unfortunately, techniques based on learning typically require large quantities of data in order to cover a sufficiently large part of the space of possible driving behaviors.

Because they capture the generative structure of vehicle trajectories, model-based methods can potentially learn more from fewer data than model-free methods. However, good cost functions are challenging to learn, and simple, hand-crafted representations may not generalize well across tasks and contexts. In general, model-based methods can be less flexible and may underperform model-free methods in the limit of infinite data. Model-free methods take a data-driven approach, aiming to learn predictive distributions over trajectories directly from data. These approaches are more flexible and require less knowledge engineering in terms of the type of vehicles, maneuvers, and scenarios, but the amount of data they require may be very large.

The past three decades have seen increasingly rapid progress in driverless vehicle technology. In addition to the advances in computing and perception hardware, this rapid progress has been enabled by major theoretical progress in computational aspects. Autonomous cars are complex systems that can be decomposed into a hierarchy of decision making problems, where the solution of one problem is the input to the next. The breakdown into individual decision making problems has enabled the use of well-developed methods and technologies from a variety of research areas.

This literature review has concentrated only on two aspects: tracking and trajectory prediction. It can serve as a reference for assessing the computational tradeoffs between various choices for algorithm design.

Author Contributions: Writing—Sections 1 and 2, M.G., Sections 3 and 4, F.L.; funding acquisition, F.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Continental AG within the *Proreta 5* project.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We kindly thank Continental AG for their great cooperation within *Proreta 5*, which is a joint research project of the Technical University of Darmstadt, University of Bremen, "Gheorghe Asachi" Technical University of Iaşi and Continental AG.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, L.; Ai, H.; Shang, C.; Zhuang, Z.; Bai, B. Online multi-object tracking with convolutional neural networks. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 645–649. [CrossRef]
2. Liu, Q.; Lu, X.; He, Z.; Zhang, C.; Chen, W.S. Deep Convolutional Neural Networks for Thermal Infrared Object Tracking. *Knowl.-Based Syst.* **2017**. [CrossRef]
3. Danelljan, M.; Häger, G.; Khan, F.S.; Felsberg, M. Convolutional Features for Correlation Filter Based Visual Tracking. In Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), Santiago, Chile, 7–13 December 2015; pp. 621–629. [CrossRef]

4. Mozhdehi, R.J.; Reznichenko, Y.; Siddique, A.; Medeiros, H. Deep Convolutional Particle Filter with Adaptive Correlation Maps for Visual Tracking. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018; pp. 798–802. [CrossRef]
5. Song, Y.; Ma, C.; Gong, L.; Zhang, J.; Lau, R.W.H.; Yang, M.H. CREST: Convolutional Residual Learning for Visual Tracking. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2574–2583.
6. Wang, C.; Galoogahi, H.K.; Lin, C.H.; Lucey, S. Deep-LK for Efficient Adaptive Object Tracking. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 627–634.
7. Du, M.; Ding, Y.; Meng, X.; Wei, H.L.; Zhao, Y. Distractor-Aware Deep Regression for Visual Tracking. *Sensors* **2019**, *19*, 387. [CrossRef] [PubMed]
8. Zhou, H.; Ummenhofer, B.; Brox, T. DeepTAM: Deep Tracking and Mapping. In *Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 851–868.
9. Danelljan, M.; Robinson, A.; Khan, F.S.; Felsberg, M. Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In Proceedings of the Computer Vision—ECCV 2016—14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; pp. 472–488. [CrossRef]
10. Danelljan, M.; Bhat, G.; Khan, F.S.; Felsberg, M. ECO: Efficient Convolution Operators for Tracking. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 6931–6939. [CrossRef]
11. Nam, H.; Han, B. Learning Multi-domain Convolutional Neural Networks for Visual Tracking. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4293–4302.
12. Fan, H.; Ling, H. SANet: Structure-Aware Network for Visual Tracking. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 2217–2224.
13. Wojke, N.; Bewley, A.; Paulus, D. Simple online and realtime tracking with a deep association metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649. [CrossRef]
14. Li, K.; Kong, Y.; Fu, Y. Multi-stream Deep Similarity Learning Networks for Visual Tracking. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, Melbourne, Australia, 19–25 August 2017; pp. 2166–2172.
15. Chu, Q.; Ouyang, W.; Li, H.; Wang, X.; Liu, B.; Yu, N. Online Multi-object Tracking Using CNN-Based Single Object Tracker with Spatial-Temporal Attention Mechanism. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 4846–4855.
16. Cui, Z.; Lu, N.; Jing, X.; Shi, X. Fast Dynamic Convolutional Neural Networks for Visual Tracking. In Proceedings of the 10th Asian Conference on Machine Learning, Beijing, China, 14–16 November 2018.
17. Fabbri, M.; Lanzi, F.; Calderara, S.; Palazzi, A.; Vezzani, R.; Cucchiara, R. Learning to Detect and Track Visible and Occluded Body Joints in a Virtual World. In Proceedings of the Computer Vision—ECCV 2018—15th European Conference, Munich, Germany, 8–14 September 2018; pp. 450–466. [CrossRef]
18. Cui, H.; Radosavljevic, V.; Chou, F.; Lin, T.; Nguyen, T.; Huang, T.; Schneider, J.; Djuric, N. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 2090-2096. [CrossRef]
19. Chu, P.; Fan, H.; Tan, C.C.; Ling, H. Online Multi-Object Tracking With Instance-Aware Tracker and Dynamic Model Refreshment. In Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 7–11 January 2019; pp. 161–170. [CrossRef]
20. Shahian Jahromi, B.; Tulabandhula, T.; Cetin, S. Real-Time Hybrid Multi-Sensor Fusion Framework for Perception in Autonomous Vehicles. *Sensors* **2019**, *19*, 4357. [CrossRef] [PubMed]
21. Bhat, G.; Johnander, J.; Danelljan, M.; Khan, F.S.; Felsberg, M. Unveiling the Power of Deep Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
22. Qi, Y.; Zhang, S.; Qin, L.; Yao, H.; Huang, Q.; Lim, J.; Yang, M. Hedged Deep Tracking. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4303–4311. [CrossRef]
23. Ristani, E.; Tomasi, C. Features for Multi-target Multi-camera Tracking and Re-identification. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6036–6046. [CrossRef]
24. Teng, Z.; Xing, J.; Wang, Q.; Lang, C.; Feng, S.; Jin, Y. Robust Object Tracking Based on Temporal and Spatial Deep Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1153–1162. [CrossRef]
25. Yoon, Y.C.; Boragule, A.; Yoon, K.; Jeon, M. Online Multi-Object Tracking with Historical Appearance Matching and Scene Adaptive Detection Filtering. In Proceedings of the 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 27–30 November 2018; pp. 1–6.
26. Sun, S.; Akhtar, N.; Song, H.; Mian, A.S.; Shah, M. Deep Affinity Network for Multiple Object Tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 104–119. [CrossRef] [PubMed]
27. Li, S.; Ma, B.; Chang, H.; Shan, S.; Chen, X. Continuity-Discrimination Convolutional Neural Network for Visual Object Tracking. In Proceedings of the 2018 IEEE International Conference on Multimedia and Expo (ICME), San Diego, CA, USA, 23–27 July 2018.

28. Hahn, M.; Chen, S.; Dehghan, A. Deep Tracking: Visual Tracking Using Deep Convolutional Networks. *CoRR* **2015**, arXiv:1512.03993v1 [cs.CV].
29. Yang, L.; Liu, R.; Zhang, D.; Zhang, L. Deep Location-Specific Tracking. In Proceedings of the 25th ACM International Conference on Multimedia, MM '17, Mountain View, CA, USA, 23–27 October 2017; ACM: New York, NY, USA, 2017; pp. 1309–1317. [CrossRef]
30. Feichtenhofer, C.; Pinz, A.; Zisserman, A. Detect to Track and Track to Detect. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 3057–3065.
31. Jiang, X.; Zhen, X.; Zhang, B.; Yang, J.; Cao, X. Deep Collaborative Tracking Networks. In Proceedings of the 29th The British Machine Vision Conference, Newcastle, UK, 3–6 September 2018.
32. Chen, L.; Lou, J.; Xu, F.; Ren, M. Grid-based multi-object tracking with Siamese CNN based appearance edge and access region mechanism. *Multimed. Tools Appl.* **2020**, *79*. [CrossRef]
33. Zhang, W.; Du, Y.; Chen, Z.; Deng, J.; Liu, P. Robust adaptive learning with Siamese network architecture for visual tracking. *Vis. Comput.* **2020**. [CrossRef]
34. Son, J.; Baek, M.; Cho, M.; Han, B. Multi-object Tracking with Quadruplet Convolutional Neural Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 3786–3795. [CrossRef]
35. Schulter, S.; Vernaza, P.; Choi, W.; Chandraker, M.K. Deep Network Flow for Multi-object Tracking. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2730–2739.
36. Wang, N.; Zou, Q.; Ma, Q.; Huang, Y.; Luan, D. A light tracker for online multiple pedestrian tracking. *J. Real-Time Image Process.* **2021**, *18*, 1–17. [CrossRef]
37. Feng, W.; Hu, Z.; Wu, W.; Yan, J.; Ouyang, W. Multi-Object Tracking with Multiple Cues and Switcher-Aware Classification. *CoRR* **2019**, arXiv:1901.06129v1 [cs.CV] .
38. Zhu, J.; Yang, H.; Liu, N.; Kim, M.; Zhang, W.; Yang, M.H. Online Multi-Object Tracking with Dual Matching Attention Networks. In Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; Springer International Publishing: Cham, Switzerland, 2018; pp. 379–396.
39. Pu, S.; Song, Y.; Ma, C.; Zhang, H.; Yang, M.H. Deep Attentive Tracking via Reciprocative Learning. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Montréal, QC, Canada, 3–8 December 2018.
40. Wang, Y.; Zhang, Z.; Zhang, N.; Zeng, D. Attention Modulated Multiple Object Tracking with Motion Enhancement and Dual Correlation. *Symmetry* **2021**, *13*, 266. [CrossRef]
41. Meng, F.; Wang, X.; Wang, D.; Shao, F.; Fu, L. Spatial–Semantic and Temporal Attention Mechanism-Based Online Multi-Object Tracking. *Sensors* **2020**, *20*, 1653. [CrossRef] [PubMed]
42. Milan, A.; Rezatofighi, S.H.; Dick, A.; Reid, I.; Schindler, K. Online Multi-Target Tracking using Recurrent Neural Networks. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
43. Sadeghian, A.; Alahi, A.; Savarese, S. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 300–311.
44. Kim, B.; Kang, C.M.; Kim, J.; Lee, S.H.; Chung, C.C.; Choi, J.W. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 399–404.
45. Deo, N.; Trivedi, M.M. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1179–1184.
46. Ma, C.; Yang, C.; Yang, F.; Zhuang, Y.; Zhang, Z.; Jia, H.; Xie, X. Trajectory Factory: Tracklet Cleaving and Re-Connection by Deep Siamese Bi-GRU for Multiple Object Tracking. In Proceedings of the 2018 IEEE International Conference on Multimedia and Expo (ICME), San Diego, CA, USA, 23–27 July 2018. [CrossRef]
47. Kim, C.; Li, F.; Rehg, J.M. Multi-object Tracking with Neural Gating Using Bilinear LSTM. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
48. Ondrúška, P.; Posner, I. Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, Phoenix, AZ, USA, 12–17 February 2016.
49. Dequaire, J.; Ondruska, P.; Rao, D.; Wang, D.Z.; Posner, I. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *Int. J. Robot. Res.* **2018**, *37*, 492–512. [CrossRef]
50. Chen, C.; Zhao, P.; Lu, C.; Wang, W.; Markham, A.; Trigoni, N. Deep-Learning-Based Pedestrian Inertial Navigation: Methods, Data Set, and On-Device Inference. *IEEE Internet Things J.* **2020**, *7*, 4431–4441. [CrossRef]
51. Du, Y.; Yan, Y.; Chen, S.; Hua, Y. Object-adaptive LSTM network for real-time visual tracking with adversarial data augmentation. *Neurocomputing* **2020**, *384*, 67–83. [CrossRef]
52. Huang, Z.; Hasan, A.; Shin, K.; Li, R.; Driggs-Campbell, K. Long-Term Pedestrian Trajectory Prediction Using Mutable Intention Filter and Warp LSTM. *IEEE Robot. Autom. Lett.* **2021**, *6*, 542–549. [CrossRef]
53. Quan, R.; Zhu, L.; Wu, Y.; Yang, Y. Holistic LSTM for Pedestrian Trajectory Prediction. *IEEE Trans. Image Process.* **2021**, *30*, 3229–3239. [CrossRef] [PubMed]

54. Song, X.; Chen, K.; Li, X.; Sun, J.; Hou, B.; Cui, Y.; Zhang, B.; Xiong, G.; Wang, Z. Pedestrian Trajectory Prediction Based on Deep Convolutional LSTM Network. *IEEE Trans. Intell. Transp. Syst.* **2020**, 1–18. [CrossRef]
55. Huang, R.; Zhang, W.; Kundu, A.; Pantofaru, C.; Ross, D.A.; Funkhouser, T.; Fathi, A. An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds. In *Computer Vision—ECCV 2020*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 266–282.
56. Dan, X. Spatial-Temporal Block and LSTM Network for Pedestrian Trajectories Prediction. *arXiv* **2020**, arXiv:2009.10468.
57. Zhou, Y.; Wu, H.; Cheng, H.; Qi, K.; Hu, K.; Kang, C.; Zheng, J. Social graph convolutional LSTM for pedestrian trajectory prediction. *IET Intell. Transp. Syst.* **2021**, *15*, 396–405. [CrossRef]
58. Fernando, T.; Denman, S.; Sridharan, S.; Fookes, C. Tracking by prediction: A deep generative model for multi-person localisation and tracking. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV 2018), Lake Tahoe, NV, USA, 12–15 March 2018; IEEE: Lake Tahoe, NV, USA, 2018; pp. 1122–1132. [CrossRef]
59. Ren, L.; Lu, J.; Wang, Z.; Tian, Q.; Zhou, J. Collaborative Deep Reinforcement Learning for Multi-Object Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
60. Li, J.; Yao, L.; Xu, X.; Cheng, B.; Ren, J. Deep reinforcement learning for pedestrian collision avoidance and human-machine cooperative driving. *Inf. Sci.* **2020**, *532*, 110–124. [CrossRef]
61. Bae, S.; Yoon, K. Robust Online Multi-object Tracking Based on Tracklet Confidence and Online Discriminative Appearance Learning. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1218–1225. [CrossRef]
62. Receveur, J.B.; Victor, S.; Melchior, P. Autonomous car decision making and trajectory tracking based on genetic algorithms and fractional potential fields. *Intell. Serv. Robot.* **2020**, *13*. [CrossRef]
63. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468. [CrossRef]
64. Bergmann, P.; Meinhardt, T.; Leal-Taixé, L. Tracking without bells and whistles. *arXiv* **2019**, arXiv:1903.05625.
65. Mogelmose, A.; Trivedi, M.M.; Moeslund, T.B. Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations. In Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea, 28 June–1 July 2015; pp. 330–335. [CrossRef]
66. Xiang, Y.; Alahi, A.; Savarese, S. Learning to Track: Online Multi-object Tracking by Decision Making. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4705–4713. [CrossRef]
67. Rangesh, A.; Trivedi, M.M. No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles using Cameras and LiDARs. *arXiv* **2019**, arXiv:1802.08755.
68. Deo, N.; Rangesh, A.; Trivedi, M.M. How Would Surround Vehicles Move? A Unified Framework for Maneuver Classification and Motion Prediction. *IEEE Trans. Intell. Veh.* **2018**, *3*, 129–140. [CrossRef]
69. Maksai, A.; Wang, X.; Fleuret, F.; Fua, P. Non-Markovian Globally Consistent Multi-object Tracking. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2563–2573. [CrossRef]
70. Milan, A.; Roth, S.; Schindler, K. Continuous Energy Minimization for Multitarget Tracking. *IEEE TPAMI* **2014**, *36*, 58–72. [CrossRef] [PubMed]
71. Solera, F.; Calderara, S.; Cucchiara, R. Learning to Divide and Conquer for Online Multi-target Tracking. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, Santiago, Chile, 7–13 December 2015; IEEE Computer Society: Washington, DC, USA, 2015; pp. 4373–4381. [CrossRef]
72. Dicle, C.; Camps, O.I.; Sznaier, M. The Way They Move: Tracking Multiple Targets with Similar Appearance. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013; pp. 2304–2311. [CrossRef]
73. Kim, C.; Li, F.; Ciptadi, A.; Rehg, J.M. Multiple Hypothesis Tracking Revisited. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4696–4704. [CrossRef]
74. Sharma, S.; Ansari, J.A.; Murthy, J.K.; Krishna, K.M. Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 3508–3515.
75. Andersen, H.; Chong, Z.J.; Eng, Y.H.; Pendleton, S.; Ang, M.H. Geometric path tracking algorithm for autonomous driving in pedestrian environment. In Proceedings of the 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Banff, AB, Canada, 12–15 July 2016; pp. 1669–1674. [CrossRef]
76. Manjunath, A.; Liu, Y.; Henriques, B.; Engstle, A. Radar Based Object Detection and Tracking for Autonomous Driving. In Proceedings of the 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), Munich, Germany, 15–17 April 2018; pp. 1–4. [CrossRef]
77. Cao, J.; Song, C.; Peng, S.; Song, S.; Zhang, X.; Xiao, F. Trajectory Tracking Control Algorithm for Autonomous Vehicle Considering Cornering Characteristics. *IEEE Access* **2020**, *8*, 59470–59484. [CrossRef]
78. Kampker, A.; Sefati, M.; Rachman, A.S.A.; Kreisköther, K.; Campoy, P. Towards Multi-Object Detection and Tracking in Urban Scenario under Uncertainties. *arXiv* **2018**, arXiv:1801.02686.

79. Bochinski, E.; Eiselein, V.; Sikora, T. High-Speed tracking-by-detection without using image information. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; pp. 1–6. [CrossRef]
80. Shan, Y.; Zheng, B.; Chen, L.; Chen, L.; Chen, D. A Reinforcement Learning-Based Adaptive Path Tracking Approach for Autonomous Driving. *IEEE Trans. Veh. Technol.* **2020**, *69*, 10581–10595. [CrossRef]
81. Ristani, E.; Solera, F.; Zou, R.S.; Cucchiara, R.; Tomasi, C. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. *arXiv* **2016**, arXiv:1609.01775.
82. Pi, W.; Yang, P.; Duan, D.; Chen, C.; Cheng, X.; Yang, L.; Li, H. Malicious User Detection for Cooperative Mobility Tracking in Autonomous Driving. *IEEE Internet Things J.* **2020**, *7*, 4922–4936. [CrossRef]
83. Chari, V.; Lacoste-Julien, S.; Laptev, I.; Sivic, J. On pairwise costs for network flow multi-object tracking. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 5537–5545.
84. Berclaz, J.; Fleuret, F.; Turetken, E.; Fua, P. Multiple Object Tracking Using K-Shortest Paths Optimization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 1806–1819. [CrossRef] [PubMed]
85. Pirsiavash, H.; Ramanan, D.; Fowlkes, C.C. Globally-optimal greedy algorithms for tracking a variable number of objects. In Proceedings of the CVPR 2011, Colorado Springs, CO, USA, 20–25 June 2011; pp. 1201–1208. [CrossRef]
86. Ristani, E.; Tomasi, C. Tracking Multiple People Online and in Real Time. In Proceedings of the 12th Asian Conference on Computer Vision, Singapore, 1–5 November 2014.
87. Roshan Zamir, A.; Dehghan, A.; Shah, M. GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs. In *Computer Vision—ECCV 2012*; Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C., Eds.; Springer: Heidelberg/Berlin, Germany, 2012; pp. 343–356.
88. Naiel, M.A.; Ahmad, M.O.; Swamy, M.; Lim, J.; Yang, M.H. Online multi-object tracking via robust collaborative model and sample selection. *Comput. Vis. Image Underst.* **2017**, *154*, 94–107. [CrossRef]
89. Wen, L.; Li, W.; Yan, J.; Lei, Z.; Yi, D.; Li, S.Z. Multiple Target Tracking Based on Undirected Hierarchical Relation Hypergraph. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1282–1289. [CrossRef]
90. Wen, L.; Du, D.; Li, S.; Bian, X.; Lyu, S. Learning Non-Uniform Hypergraph for Multi-Object Tracking. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 8981–8988. [CrossRef]
91. Dolatabadi, M.; Elfring, J.; van de Molengraft, R. Multiple-Joint Pedestrian Tracking Using Periodic Models. *Sensors* **2020**, *20*, 6917. [CrossRef] [PubMed]
92. Schreier, M.; Willert, V.; Adamy, J. An Integrated Approach to Maneuver-Based Trajectory Prediction and Criticality Assessment in Arbitrary Road Environments. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 2751–2766. [CrossRef]
93. Houenou, A.; Bonnifait, P.; Cherfaoui, V.; Yao, W. Vehicle trajectory prediction based on motion model and maneuver recognition. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 4363–4369. [CrossRef]
94. Casas, S.; Luo, W.; Urtasun, R. IntentNet: Learning to Predict Intention from Raw Sensor Data. In Proceedings of the 2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29–31 October 2018; pp. 947–956.
95. Nikhil, N.; Morris, B.T. Convolutional Neural Network for Trajectory Prediction. In Proceedings of the Computer Vision—ECCV 2018 Workshops, Munich, Germany, 8–14 September 2018; pp. 186–196. . [CrossRef]
96. Aptiv.; Audi.; Baidu.; BMW.; Continental.; Daimler.; Fiat.; Chrysler Automobiles.; HERE.; Infineon.; Intel.; Volkswagen. Safety First for Automated Driving. Available online: https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf (accessed on 2 July 2019).
97. Djuric, N.; Radosavljevic, V.; Cui, H.; Nguyen, T.; Chou, F.; Lin, T.; Schneider, J. Motion Prediction of Traffic Actors for Autonomous Driving using Deep Convolutional Networks. *CoRR* **2018**, arXiv:1808.05819v3 [cs.LG].
98. Ward, E. Models Supporting Trajectory Planning in Autonomous Vehicles. Ph.D. Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2018.
99. Lee, N.; Choi, W.; Vernaza, P.; Choy, C.B.; Torr, P.H.S.; Chandraker, M.K. DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2165–2174. [CrossRef]
100. Singh, A. Prediction in Autonomous Vehicle–All You Need to Know. Available online: https://medium.com/m/global-identity?redirectUrl=https%3A%2F%2Ftowardsdatascience.com%2Fprediction-in-autonomous-vehicle-all-you-need-to-know-d8811795fcdc (accessed on 28 January 2021).
101. Lefèvre, S.; Vasquez, D.; Laugier, C. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech J.* **2014**, *2014*, 1–14. [CrossRef]
102. Lawitzky, A.; Althoff, D.; Passenberg, C.F.; Tanzmeister, G.; Wollherr, D.; Buss, M. Interactive scene prediction for automotive applications. In Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, QLD, Australia, 23–26 June 2013; pp. 1028–1033. [CrossRef]
103. Woo, H.; Sugimoto, M.; Wu, J.; Tamura, Y.; Yamashita, A.; Asama, H. Trajectory Prediction of Surrounding Vehicles Using LSTM Network. In Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, QLD, Australia, 26–28 June 2013.

104. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; Savarese, S. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 961–971. [CrossRef]
105. Bojarski, M.; Yeres, P.; Choromanska, A.; Choromanski, K.; Firner, B.; Jackel, L.D.; Muller, U. Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. *CoRR* **2017**, arXiv:1704.07911v1 [cs.CV].
106. Phan-Minh, T.; Grigore, E.C.; Boulton, F.A.; Beijbom, O.; Wolff, E.M. CoverNet: Multimodal Behavior Prediction using Trajectory Sets. *arXiv* **2020**, arXiv:1911.10298.
107. Mangalam, K.; An, Y.; Girase, H.; Malik, J. From Goals, Waypoints and Paths to Long Term Human Trajectory Forecasting. *arXiv* **2020**, arXiv:2012.01526.
108. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 234–241.
109. MacQueen, J.B. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*; University of California Press: Berkeley, CA, USA, 1967; pp. 281–297.
110. Chandra, R.; Bhattacharya, U.; Bera, A.; Manocha, D. TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 8475–8484. [CrossRef]
111. Li, J.; Yang, F.; Tomizuka, M.; Choi, C. EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning. *arXiv* **2020**, arXiv:2003.13924.
112. Zhao, H.; Gao, J.; Lan, T.; Sun, C.; Sapp, B.; Varadarajan, B.; Shen, Y.; Shen, Y.; Chai, Y.; Schmid, C.; et al. TNT: Target-driveN Trajectory Prediction. *arXiv* **2020**, arXiv:2008.08294.
113. Gao, J.; Sun, C.; Zhao, H.; Shen, Y.; Anguelov, D.; Li, C.; Schmid, C. VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. *arXiv* **2020**, arXiv:2005.04259.
114. Rhinehart, N.; McAllister, R.; Kitani, K.; Levine, S. PRECOG: PRediction Conditioned On Goals in Visual Multi-Agent Settings. In Proceedings of the International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
115. Bhattacharyya, A.; Hanselmann, M.; Fritz, M.; Schiele, B.; Straehle, C.N. Conditional Flow Variational Autoencoders for Structured Sequence Prediction. *arXiv* **2020**, arXiv:1908.09008.
116. Mangalam, K.; Girase, H.; Agarwal, S.; Lee, K.H.; Adeli, E.; Malik, J.; Gaidon, A. It Is Not the Journey But the Destination: Endpoint Conditioned Trajectory Prediction. In *Computer Vision—ECCV 2020*; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 759–776.
117. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27.
118. Gupta, A.; Johnson, J.; Li, F.F.; Savarese, S.; Alahi, A. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018), Salt Lake City, UT, USA, 18–23 June 2018.
119. Lai, W.C.; Xia, Z.X.; Lin, H.S.; Hsu, L.F.; Shuai, H.H.; Jhuo, I.H.; Cheng, W.H. Trajectory Prediction in Heterogeneous Environment via Attended Ecology Embedding. In Proceedings of the 28th ACM International Conference on Multimedia, MM '20, Seattle, WA, USA, 12–16 October 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 202–210. [CrossRef]
120. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 618–626. [CrossRef]
121. Amirian, J.; Hayet, J.B.; Pettre, J. Social Ways: Learning Multi-Modal Distributions of Pedestrian Trajectories with GANs. In Proceedings of the CVPR Workshops, Long Beach, CA, USA, 16–20 June 2019.
122. Sadeghian, A.; Kosaraju, V.; Sadeghian, A.; Hirose, N.; Rezatofighi, H.; Savarese, S. SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 1349–1358. [CrossRef]
123. Messaoud, K.; Deo, N.; Trivedi, M.M.; Nashashibi, F. Trajectory Prediction for Autonomous Driving based on Multi-Head Attention with Joint Agent-Map Representation. *arXiv* **2020**, arXiv:2005.02545.
124. Monti, A.; Bertugli, A.; Calderara, S.; Cucchiara, R. DAG-Net: Double Attentive Graph Neural Network for Trajectory Forecasting. *arXiv* **2020**, arXiv:2005.12661.
125. Liang, J.; Jiang, L.; Murphy, K.; Yu, T.; Hauptmann, A. The Garden of Forking Paths: Towards Multi-Future Trajectory Prediction. *arXiv* **2020**, arXiv:1912.06445.
126. Salzmann, T.; Ivanovic, B.; Chakravarty, P.; Pavone, M. Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020.
127. Deo, N.; Trivedi, M.M. Trajectory Forecasts in Unknown Environments Conditioned on Grid-Based Plans. *arXiv* **2020**, arXiv:2001.00735.

128. Zhou, B.; Schwarting, W.; Rus, D.; Alonso-Mora, J. Joint Multi-Policy Behavior Estimation and Receding-Horizon Trajectory Planning for Automated Urban Driving. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 2388–2394. [CrossRef]
129. Suraj, M.S.; Grimmett, H.; Platinský, L.; Ondrúška, P. Predicting trajectories of vehicles using large-scale motion priors. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1639–1644. [CrossRef]
130. Hoermann, S.; Stumper, D.; Dietmayer, K. Probabilistic long-term prediction for autonomous vehicles. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 237–243. [CrossRef]
131. Xu, Y.; Zhao, T.; Baker, C.; Zhao, Y.; Wu, Y.N. Learning Trajectory Prediction with Continuous Inverse Optimal Control via Langevin Sampling of Energy-Based Models. *CoRR* **2019**, arXiv:1904.05453v1 [cs.LG].
132. Andersson, J. Predicting Vehicle Motion and Driver Intent Using Deep Learning. Master's Thesis, Chalmers University of Technology, Göteborg, Sweden, 2018.
133. Silver, D.; van Hasselt, H.; Hessel, M.; Schaul, T.; Guez, A.; Harley, T.; Dulac-Arnold, G.; Reichert, D.; Rabinowitz, N.; Barreto, A.; et al. The Predictron: End-To-End Learning and Planning. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australi, 6–11 August 2017; pp. 3191–3199.
134. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* **2016**, *529*, 484–489. [CrossRef]
135. Guez, A.; Weber, T.; Antonoglou, I.; Simonyan, K.; Vinyals, O.; Wierstra, D.; Munos, R.; Silver, D. Learning to Search with MCTSnets. *CoRR* **2018**, arXiv:1802.04697v2 [cs.AI].
136. Schwarting, W.; Alonso-Mora, J.; Rus, D. Planning and Decision-Making for Autonomous Vehicles. *Annu. Rev. Control Robot. Auton. Syst.* **2018**, *1*, 187–210. [CrossRef]
137. Zhou, Y.; Hu, H.; Liu, Y.; Lin, S.W.; Ding, Z. A distributed method to avoid higher-order deadlocks in multi-robot systems. *Automatica* **2020**, *112*, 108706. [CrossRef]
138. Foumani, M.; Moeini, A.; Haythorpe, M.; Smith-Miles, K. A cross-entropy method for optimising robotic automated storage and retrieval systems. *Int. J. Prod. Res.* **2018**, *56*, 6450–6472. [CrossRef]
139. Foumani, M.; Gunawan, I.; Smith-Miles, K. Resolution of deadlocks in a robotic cell scheduling problem with post-process inspection system: Avoidance and recovery scenarios. In Proceedings of the 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 6–9 December 2015; pp. 1107–1111. [CrossRef]
140. Sutton, R.S.; Barto, A.G. *Reinforcement Learning*; MIT Press: Cambridge, MA, USA, 2018.
141. Lapan, M. *Deep Reinforcement Learning Hands-On*; Packt Publishing: Birmingham, UK, 2018.
142. Grondman, I.; Busoniu, L.; Lopes, G.A.D.; Babuska, R. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Trans. Syst. Man Cybern. Part (Appl. Rev.)* **2012**, *42*, 1291–1307. [CrossRef]
143. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
144. Konda, V.R.; Tsitsiklis, J.N. Actor-Critic Algorithms. *SIAM* **2000**, *42*, 1008–1014.
145. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1602.01783.
146. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *CoRR* **2017**, arXiv:1707.06347v2 [cs.LG].
147. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.I.; Abbeel, P. Trust Region Policy Optimization. *CoRR* **2015**, arXiv:1502.05477v5 [cs.LG].
148. Weber, T.; Racanière, S.; Reichert, D.P.; Buesing, L.; Guez, A.; Rezende, D.J.; Badia, A.P.; Vinyals, O.; Heess, N.; Li, Y.; et al. Imagination-Augmented Agents for Deep Reinforcement Learning. In Proceedings of the 31st International Conference on Neural Information Processing, Long Beach, CA, USA, 4–9 December 2017; pp. 5694–5705. [CrossRef]
149. Liu, B.; Ghavamzadeh, M.; Gemp, I.; Liu, J.; Mahadevan, S.; Petrik, M. Proximal Gradient Temporal Difference Learning: Stable Reinforcement Learning with Polynomial Sample Complexity. *J. Artif. Intell. Res.* **2018**, *63*, 461–494. [CrossRef]