UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXIX Ciclo

Object Detection and Classification for ADAS and Autonomous Driving

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Supervisor:

Chiar.mo Prof. Alberto Broggi

Tutor:

Chiar.mo Prof. Massimo Bertozzi

PhD Student: *Luca Castangia*

December 2016

A chi mi ha supportato e sopportato

Sommario

List of Figures

List of Tables

Introduction

A reliable and robust perception system of the real world is a necessary for an autonomous vehicle and the Advanced Driver Assistance Systems (ADAS). Obstacles detection and classification are the main pillar for the correct understanding of the dynamic world. Approaches that are based on stereo vision and other 3D reconstruction technologies (e.g. LIDAR,KINECT) have been used for the ADAS first and autonomous ground vehicles, after, providing good results as shown in Fig. 1.

Obstacles detection is a very wide field and in this field the are a lot of works in the last years [2]. In academic research [3, 4], it has been clearly established the essential role of these systems to realize active safety systems, that are systems already introduced by industry [5, 6], industries, show great interest for computer vision systems [1]. These systems need to handle situational criticalities and simultaneously assess awareness of these criticalities by the driver; this two specification requires that the obstacles detection algorithms must be reliable and accurate [7], providing: a real-time output, a stable and robust representation of the scenario and it has to be stable and robust respect in any lighting and weather conditions.

The Initial systems is based on only one sensor (e.g. radar or laser for automatic cruise control and camera for lane departure warning) in addition to proprioceptive sensors such as wheel speed and yaw rate sensors. The current systems, such as automatic cruise control(ACC) operating at the entire speed range or emergency braking assist for collision avoidance, require multiple sensors since the individual measurement cannot meet these requirements. It has led the industry to move towards the use of assortment of them in order to take the advantage of each one. [8, 9].

(a)

(b)

(c)

(d)

Figure 1: Examples of fully autonomous ground vehicle: (a) BRAiVE- VisLab, University of Parma; (b) Bertha - Daimler; (c) KITTI - Karlsruhe Institute of Technology (KIT); (d) Google Car - Stanford Artificial Intelligence Laboratory (SAIL), Stanford University.

The object detection and classification such as vehicle and pedestrian still remaining an active area of research [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. In the ADAS, computer vision technologies are involved for solving problems such as forward collision detection, obstacle detection, lane keeping, traffic sign recognition and pedestrian detection. Assistant Driver Systems are the use case of pedestrians and vehicles detection. The Vehicles need to be equipped with a system able to detect objects and take decision in dangerous situations, where probably the driver is not able to avoid a collision. A full ADAS, with regard to obstacles, would not only include detection but also tracking, it's orientation, it's intent analysis, [20, 21].
The system is able to detect obstacles using a probabilistic occupancy grid built from disparity map. Obstacles classification is performed with SoftCascade trained on Aggregate Channel Features(ACF). The stage of tracking and fusion are able to provide stability and robustness to the result.
The remainder of the dissertation is structured as follows: the chapter 1 provides a broad overview of the state of the art. The system is described, in detail, in the chapter 3 with quantitative and qualitative results reported in the chapter 4. At the end, results, conclusion and future works are presented in the chapter 5.

Chapter 1

State of The Art

This chapter provides an overview of the different approaches presents in the community, divided according to the system steps.

1.1 Monocular Object Detection

Vehicles and pedestrians detection is a very hard challenge. It is complex to find an exhaustive type of feature for the following reasons:

- high variability of obstacles appearance due they change pose, wear different clothes, carry objects and have different size. Erroneous detections can be introduced by shadows, man-made structures, and ubiquitous visual clutter.
- outdoor urban scenarios have clutter background and different illumination and weather conditions, that allow to make only weak assumptions about the scene structure [22].
- obstacles can be occluded, partially or totally. Training classifiers with just full obstacles shapes gives better detection results but also more false positives;
- high dynamic scenes where both the obstacles and camera are in motion. Obstacles also appear at a different viewing angles.

- high performance in detection rate and speed. Complex features are better but, at the same time require more computational resources and then are slow. We need a trade-off between this constraints: speed is fundamental for real-time processing and, detection rate is fundamental for decreasing the number not detected.

It is need to train a classifier for detecting obstacles which can be different in: carried objects, size and clothes. The more general classifier will be, the more obstacles it will detect but, also, will be easier to get false positives such as a tree, poles and guard rails.
A further challenge is to collect an extensive database; a large amount of images that allows to train a classifier that can better interact with the described problems. It is more important, as shown in the next chapters, which kind of subjects are used to train the classifier. The detection can be broken down into the generation of initial object hypotheses (ROI selection) and verification (classification).

1.1.1 ROI selection

ROI selection is one of the main and most important components of detection algorithms. This operation consists of selecting regions of interest where obstacles are present. The impact of this processing step on the computational time is huge; selecting more ROI requires more processing time. The simpler approach is the sliding window technique, where a detection window is shifted at various scales and location over the image. To give a sense of the computational complexity, let us take for example an image with resolution 640x480 pixels. Disregarding the varying window sizes for a moment, if a constant window size of 128x64 is used over the entire image, with a one pixel shift, more than 200.000 detection windows need to be analyzed requiring huge computational resources. Improvements can be achieved by combining the sliding window method with a cascade classifier of increasing complexity. This kind of approach is used in [16], where a combination of haar features and cascade classifier is used to obtain a detection area; also a cascade classifier is applied to understand the image regions where obstacles are more frequently located. An alternative to the use

of a cascade classifier for reducing the search area is to take advantage of the camera calibration knowledge and a priori information on target objects. Other assumptions such as float world objects, common pedestrians geometry, object height, and aspect ratio helps to reduce the number of possible ROIs. Other techniques are derived from the image data and object motion. In surveillance system the background subtraction can be used to obtain the ROI from an image or, also, compute the deviation of the observed optical flow from the expected ego-motion flow field. The use of an interesting point detector would help to extract the regions with high information content based on local discontinuities of the image brightness function. All these techniques allow to reduce the detection window and speed-up the system, which is a key feature for obtaining real-time behavior.

1.1.2 Classification

Once the ROIs of an image are determined, the next step is to understand whether obstacles are present in the ROIs. This process is referred to as classification (or verification), and it involves utilizing pedestrian/vehicle appearance and features. A good starting point is the separation of these models into generative and discriminative models [23]; this approach allows to classify an image subregion as pedestrian/vehicle. Discriminative models differ from generative models in that they do not allow one to generate samples from the joint distribution of x and y, where x is an observed variable and y in an unobserved variable. However, for tasks such as classification and regression that do not require the joint distribution, discriminative models can yield superior performance.

1.1.3 Generative Models

In the generative approach, the empirical observation are explained by a model that describes probabilistically the interaction between the variable quantities. This probabilistic system is specified by two components:

- a list of variables that quantify the status observed and supposed model;

- a joint probability defined over all these variables.

Differences between the generative models are in the nature of these specifics. The a-posteriori probability can be obtained from the a-priori probability and joint probability from the Bayesian approach.

Shape models

Shape cues are very important to reduce variability in pedestrian appearance due to lighting conditions and clothing; there are basically two type of shape representations: discrete and continue.

- Discrete approaches uses representative shapes to simplify complex shapes. This kind of approach requires high specificity for example-based models and consequentially an higher number of example shapes to cover the space of shape models. In this case, trade-off between specificity and compactness in order to use it in the real world must be found.

- In the continuous shape model case, a compact parametric representation for the class conditional density and learning from a set of training shapes are used. Forcing topologically different shapes (such as pedestrian with feet apart and closed) can give many intermediate physically implausible model instantiations. To recover physically plausible regions in the linear model space, conditional density models have been proposed. Nonlinear extensions can also be used jointly with a larger number of training shapes; an alternative solution is breaking the non linear model into piecewise linear patches. Using the continuous model, gaps in the discrete model representation can be filled using interpolation. In previous years, a two-layer statistical field model has been proposed [24] to improve the performance of detectors in presence of occlusions and cluttered background by representing shapes as a distributed connected model. An hidden Markov layer for capturing shapes prior is combined with an observation layer, which associates shape with the likelihood of image observations.

Combined shape and texture models

To create more complex representations, shape and texture information can be combined within a compound of parametric appearance model. These approaches involve separate statistical models for shape and intensity light variations [25]. Model fitting requires joint estimation of shape and texture parameters using iterative error minimization schemes. To reduce the parameter estimation complexity, the relation between fitting errors and the associated model parameters can be learned from examples.

1.1.4 Discriminative models

The discriminative models, on the other hand, directly compare the problem of finding criterias that allow grouping of empirical observations. To do this, usually , it is extracted some intrinsic features such as thickness, shape factors, brightness, from the object and mapped each observation to a point in a multidimensional space. The relationship between similar points of the same object or differences between points of different objects is searched in a second step.

Features

In computer vision and image processing the concept of feature is used to denote a piece of information which is relevant for solving the computational task related to a certain application. More specifically, features can refer to:

- the result of a general neighborhood operation (feature extractor or feature detector) applied to the image;
- specific structures in the image itself, ranging from simple structures such as points or edges to more complex structures such as objects.

Other examples of features are related to motion in image sequences, to shapes defined in terms of curves or boundaries between different image regions, or to properties of such a region. The feature concept is very general and the choice of fea-

tures in a particular computer vision system may be highly dependent on the specific problem at hand. Local filters on pixel intensities are frequently used for feature extraction. One of the most popular features are the non adaptative Haar wavelet features proposed by Papageorgiou and Poggio [26] and then used by many others. Haar features popularity is due to their simplicity and fast evaluation using integral images. Viola and Jones[27] adapted the idea of using Haar wavelets and developed the so called Haar-like features. An Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, summing up the pixel intensities in each region and calculating the difference between these sums. This difference is then used to categorize subsections of an image. For example, let say we have an images database of human faces. It is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore a common haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relatively to a detection window that acts like a bounding box to the target object (the face in this case). Due to overlapping spatial shifts, we have many times redundant representations and we need to select the most appropriate features from a large set of them. Initially, using the geometric configuration of human body, this procedure was done manually [28]; but, are used now automatic procedures of features selection, such as variants of AdaBoost. The automatic extraction can be seen as an optimization for the classification task. We can include particular configuration of spatial features in the optimization, leaving that the features set fits to the underlying data set during training. This type of approach, has been shown to be more effective than the non adaptative Haar wavelets features with regards to pedestrian classification. Other type of features are based on discontinuities in the brightness function in the image in terms of models of local edge structure. The most popular are the Histogram of Oriented Gradients(HOG) descriptors [13], well-normalized image gradient orientation histograms, calculated over local image blocks. They are implemented in dense and sparse representation, where the last one must be preceded by an interest point detector to know the relevant part of the images. Initially, the dense HOG descriptors were computed only at a single fixed scale, to obtain a smaller feature vector

and better performance in terms of speed. But, afterwards, variable size block versions has been implemented with better results than the original HOG descriptor. The local shape filters that explicitly incorporate the spatial configuration of salient edge-like structures have already been investigated by other people: Mikolajczyk et al. [29] introduced multiscale features based on horizontal and vertical co-occurrence groups of dominant gradient orientation. Also sets of edgelets, representing local line or curve segments, have been proposed. An extension has been recently introduced about adapting the local edgelet features to the underlying image data [30]; using the Adaptive Boosting these features are assembled from low-level oriented gradient responses to give us more discriminative local features. This is a very new approach because usually Adaboost is used to select the most discriminative subset of features. Also the outlines, with the extension to spatio-temporal features, have been used to capture the human motion, especially gait. Haar wavelets and local shape filters have been extended to the temporal domain by incorporating intensity differences over time or, also, HOGs have been extended to histograms of differential optical flow [31]. There are different papers comparing the performances of several techniques.

1.1.5 Classifier architectures

The goal of discriminative classification is to found an optimal decision boundary between pattern classes in a features space. Feed-forward multilayer neural networks [32] implement linear discriminant functions in the feature space in which input patterns have been mapped nonlinearly. The optimal boundary is reached minimizing an error criterion with respect to the network parameters, i.e., mean squared error. In the out context, feed-forward multilayer neural networks have been applied particularly in conjunction with adaptive local receptive field features as nonlinearities in the hidden network layer. Support Vector Machines(SVMs)[33] has become a powerful tool to solve pattern classification problem. At opposite of Neural Networks, SVMs do not minimize some error criteria but maximize the margin of a linear decision boundary (hyperplane) to achieve maximum separation between the object classes. In pedestrian/vehicle detection, linear SVM was combined with different type of features [31] [13]. Using non linear SVMs with polynomial or radial kernel showed to

improve considerably the performance but with a significant increase in computational cost and memory requirements. AdaBoost was used as both automatic features selection procedure and as constructor of strong classifiers as weighted linear combinations of the selected weak classifiers, each involving a threshold on a single feature. Viola et al. [27] adopted the boosted cascade detectors to incorporate nonlinearities and speed up the classification process. The main motivation for this type of approach is that the majority of the detection windows in an image are non-pedestrians, so a method is needed to keep those containing pedestrian and remove those containing non-pedestrian in the shortest possible time. AdaBoost performs in each layer using the error of the precedent layer to improve the performance and create a more complex detector. This increase the processing speed requiring just few features in the early cascade layer level to reject non-pedestrians. The ImageNet dataset [34] allows to develop new approach such as deep learning for object classification, localization and segmentation. The approach introduced by [35] in 2012 push the computer vision community to analyze deep learning techniques applied to images. Task such as detection and localization that provide a class and the position of the object over the image, can be performed with[36, 37, 38], that can handle easily the multiclass classification problem. This sliding window approach is of course extremely expensive due to the huge search space. To reduce the amount of times that the classifier (a CNN) needs to run, it could be applied on a coarser grid, but this has the risk of missing the ideal bounding box in some cases. Ideally, one would like to have an initial set of somehow proposed regions that need to be evaluated. It should be minimal while still containing all ground truth bounding boxes of objects in the image. This approach was successfully introduced by Girshick et al. [36] in their work called R-CNN. To generate the region proposals, they employed the Selective Search algorithm [39]

1.2 Tracking

Stable and robust system needs a tracking stage, to re-identify and measure dynamics and motion characteristics and predict and estimate the upcoming position of obstacles on the road. Measurement and sensor uncertainty, data association, and track

management are common issues of objects tracking. Below will be described the monocular and stereo-vision tracking approaches. Even if there are common estimation and filtering methods, depending on available measurements, the estimation parameters differ: often, monocular tracking are based on measurement and estimation in term of pixels, whereas stereo-vision methods estimate dynamics in meters. Combined approach and fusion with other sensing modalities will be also described.

1.2.1 Monocular Object Tracking

Monocular tracking is typically based on image plane. Tracking using monocular vision serves two major purposes:

- facilitate estimation of motion and prediction of obstacles position in the image plane;
- help the temporal stability: filter spurious false positives [40] and maintain awareness of previously detected obstacles that were not detected in a given frame [41].

Measure the motion and predict the position of obstacles in pixel position and velocity is the goal of monocular tracking. The pixels observation space, leads to uniquely vision-based tracking methods, based on the objects appearance in the image plane. Template matching is an example of uniquely vision-based tracking. Object are detected using Haar wavelet coefficients and SVM classification in [42]; frame to frame tracking is performed taking a measurement of the similarity in appearance. Cross-correlation scores is often used in appearance-based tracking. A further step in the tracking process is represented by feature-based tracking [43]. Haar-like features combined with AdaBoost cascade classifier is used in [41], where the tracking is performed through a Kalman filter in the image plane. In order to have a measurement also in case of detector failure, it is exploited a local search over the image patch for similar feature score. In [44] the optical flow is used to track obstacles by directly measuring the new position and the displacement of interest points. Bayesian filter has been largely used in the monocular tracking literature.

Typically, the state is formed by pixel coordinates of obstacle bounding box and the interframe pixel velocities [45]. In [22], [27], and [33], Kalman filtering was used to estimate the motion of detected vehicles in the image plane. In [46, 47, 48], Kalman filtering was used to estimate the motion of detected vehicles in the image plane. [45, 49, 40, 50, 51, 52] describe the use of particle filtering for monocular tracking. Several studies attempt to estimate longitudinal distance and 3-D information from monocular vision. Typically, it is assumed ground flat [53, 54] or it is used the interest point detection and a robust model fitting step to estimated its parameters [55]. In [56] the estimation of 3-D coordinates from monocular vision is made using a set of constraints and assumption and tracked through a Kalman filter. In [57], ground plane estimation is used to extract the 3-D information. Tracking is based on the interacting multiple models, each one consisting of a Kalman filter. Monocular vision is used in [58, 59] to estimate the ego motion and moving object were tracked using a 3-D Kalman filter.

When the class of the object is not a priori known we cannot use tracking by detection method, such as described before. The model of the target is not fixed, some techniques such as TLD[60], that use a sparse flow and a positive learning method to learn the generic target. Other approach such kernelized correlation filter[61, 62] can achieve better performance than TLD but correlation filter cannot handle the scenario of target occlusion and re-detection of the target over the entire image. Other approach based on convolutional neural network[63, 64] outperform the previous method but, these approach require a huge computational effort that can be achieved using a GPU.

Chapter 2

Machine Leraning Framework

2.1 HOG (Histogram of Oriented Gradient)

The basic idea of HOG is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge direction, even without precise knowledge of the corresponding gradient or edge position. In practice this is implemented by dividing the image window into small spatial regions (âĂIJcellsâĂİ), for each cell accumulating a local 1-D histogram of gradient direction or edge orientation over the pixel of the cell. The combined histogram entries make the representation. For better invariance to illumination and shadowing, it is also useful to contrast-normalize the local response before using them. This can be done by ac- cumulating a measurement of local histogram âĂIJenergyâĂİ over somewhat larger spatial regions (âĂIJblocksâĂİ) and using the results to normalize all of the cells in the block. The HOG representation has several advantages. It captures edges or gradient struc- tures that are very characteristic of local shape, and it does so in a local representation with an easily controllable degree of invariance to local geometric and photometric transformation: translation or rotation make little difference if they are much smaller that the local spatial or orientation bin size. For human detection, rather coarse spatial sampling, fine orientation sampling and strong local photometric normalization turns out to be the best strategy, presumably because

it permits limbs and body segments to change appearance and move from side to side quite a lot provided that they maintain a roughly upright orientation. We can see in the below a algorithm implementation step by step as also shown in figure.

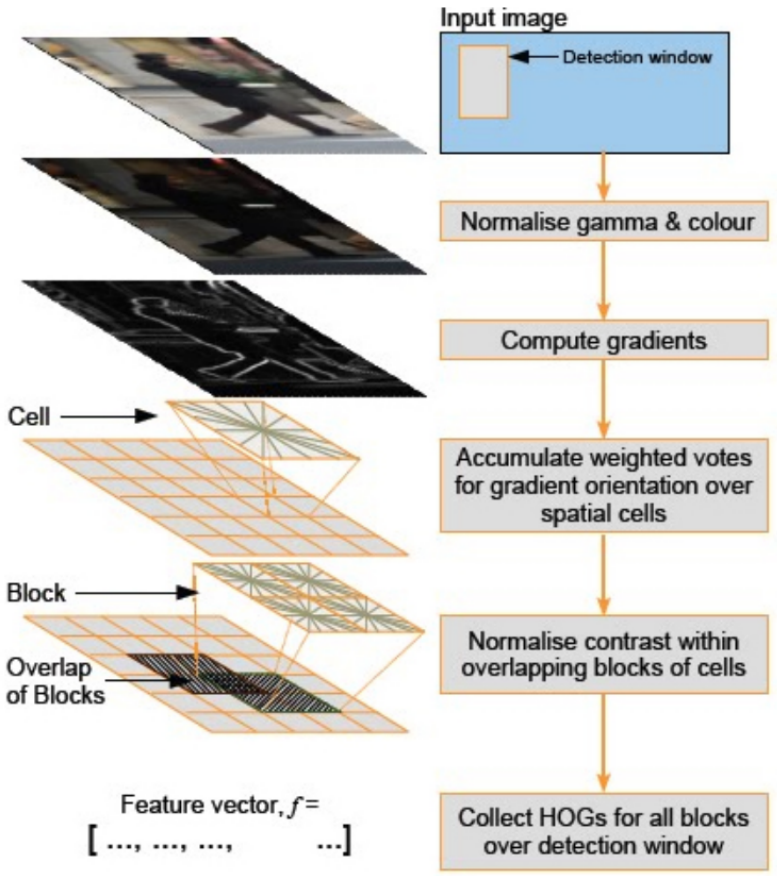


Figure 2.1: main steps for HOG features computation.

2.2 Channel Features

Features extraction for object detection is an import block for building an object detector. This process of feature extraction is a trade-off between the amount of in-

formation features that can be capture and the computational efficiency at which they can be processed. A big feature representations can represent complex patterns in the data but typically require multiple post processing such as normalization steps and a larger local-neighborhood search space. Complex features require more computation effort to be extracted. Motivated by the simple and low computation effort, feature extraction approach of Viola and Jones, Dollar et al.[27][65]. They proposed the idea of using a huge variety of image channels for extracting a richer set of features but keeping an under control the computational efficiency. Now we will give a brief definition of channel features, present various channel types, the feature ranking and how to learn the most important features using the AdaBoost framework. Using the idea of image channels, a family of features can be defined. Let I be the input image and ω a function that manipulate the image using spatial shift-invariant transformation. The application of the function ω on the input image I generate a new processed image called channel image, as its commonly referred as in literature. we use this notation for the channel image C generated by function ω as $C = \omega[I]$.

In the original paper[65] Dollar et al. analyze the feature extraction for pedestrian detection. These results and the experiments performed in [66] shown the best set of channels, using a trade off between information abundance and computation efficiency.

- Gradient orientation channels
- Gradient magnitude channel
- LUV color channels

The Channel features are extracted from the channel's pool. Figure 2.2 displays the features channels, gradient magnitude channel, the gradient orientations channels along quantized orientation 2.3, the LUV color space. Intuitively this set of channels can represents well the essential components of the image: gradient orientations, gradient magnitude, color, texture.

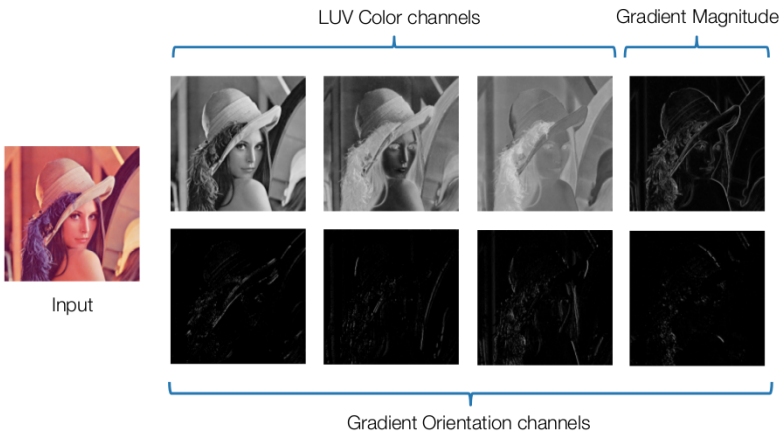


Figure 2.2: Multiple registered image channels $C = \omega[I]$,computed for an example image. From these channels a rich feature set can be extracted, for example Haar features, approximations of HOG features, local sums and histograms.

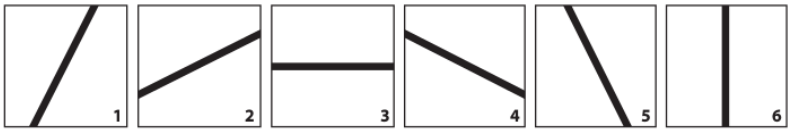


Figure 2.3: Exammple of 6 gradient orientations, indicating the of orientation bins used for the channel computation.

2.3 Adaptative Boosting

The previous section explained the feature representation explained the feature rappresentation of the object detector. This section describe mainly the learning framework. A briefly introduction of AdaBoost framework is shown and explain how it can be used for feature ranking and the construction of an ensamble learning classifier. The goal is to train an object detector of certain model size, and use this detector on image pyramid space.

2.3.1 Feature Selection

A huge candidate feature pool is generated instead carefully designing the features, from these pool AdaBoost will selects the most important feature at each training iteration. In [67] is shown that feature mining vercome the effort for careful feature design and show that feature mining outperform the hand-crafted features. The first step is the generation of a huge feature pool that contain an amount of randomly selected features. First-order channel features are described with:

- random rectangular region (X0,Y0,W,H)
- channel index (Ch)

In the implementation will be extracted an amount of K features, randomly for each channel, thus none of the channels is discriminated. The constraint for the randomly generated features is the minimum size, the rectangular region should have an area at least 20 pixels. Smaller features does not contain useful information and slow down the feature ranking process. For our application of pedestrian and car detection, we train our model at a fixed size, for example 32×64 for pedestrian and 32×32 pixels. After the random generation of the feature pool, Discrete AdaBoost [68] is used for feature ranking and the construction of the strong classifier.

2.3.2 AdaBoost

AdaBoost is a popular boosting framework in machine learning that describes an algorithm of combining many weak classifiers into a linear combination of weak classifier to form a strong classifier. Boosting was proposed originally by Schapire et al. in [68]. A weak classifier has to be better than the random guessing, but the output of the strong classifier should predict the true label.

Let the training data $(x_1, y_1), ..., (x_K, y_K)$ with x_j a feature vector, $y_j \in \{1, +1\}$ the pattern's label. The AdaBoost procedure train a list of weak classifiers $h_n(x)$, giving higher weight to the examples that are not classified correctly [69]. This is performed for a number of training iterations N, and then the final classifier is composed by a linear combination of the weak classifiers obtained from each stage. The

final classification score is denoted by H_N and defined as:

$$\sum_{i=0}^{N} \alpha_i \cdot h_i(x)$$

Where each $h_i(x) \in -1, 1$ and α_i is related to the strength of the weak classifier. The label predicted of an object x is $sign(H_N(x))$. The description of Discrete AdaBoost is shown in Algorithm 0.

Algorithm 1 AdaBoost [68]

The training data $(x_1, y_1), ..., (x_K, y_K)$
Initial weights $w_j = \frac{1}{K}, j \in (1, ..., K)$
for each training iteration j=0 to N **do**
Evaluate weak classifier $h_j(x) \in 1, -1$ using weight w_j
Compute error $\gamma_j = \sum_0^K e_i \cdot w_i$ over all samples and where $e_i = 0$ if the sample is classified correctly by $h_t(x)$ and $e_i = 1$ otherwise
Compute strength coefficent $\alpha_j = log(\frac{1-\gamma}{\gamma})$
Update the weights of the training, set $w_j = e^{\alpha_j \cdot e_i}$ for $i \in (1, .., K)$
Normalize weights $\sum_0^K w_j = 1$
end for
The final classifier is $sign\{\sum_0^N \alpha_i \cdot h_i(x))\}$

In Algorithm 0, the training procedure increases the weights of the pattern that are not classified correctly by $h_i(x)$ using a factor α that depends on the weighted training error. In [68] say that AdaBoost algorithms can be interpreted as stage-wise estimation procedure for fitting an additive logistic regression model. Logistic regression is a popular approach used in statistics, and for a two-class problem has can be formulated as:

$$log\frac{P(y=1|x)}{P(y=-1|x)} = \sum_{i=0}^{N} h_i(x)$$

In AdaBoost training, the function that we wont to minimize is the exponential loss criterion $L(H_N)$, because in [70] is shown that is an upper bound on the misclassifi-

cation error:

$$L(H_N) = \sum_j e^{-y_j H_N(x_j)}$$

2.3.3 Feature Selection

A weak learner $h_i(x)$, is usually a decision tree, in the final object detector is a decision tree with depth= 2. A decision stump is a decision tree with depth= 1, we introduce an explanation for future ranking in case of decision stump, a decision stump is composed by:

- channel feature f_j
- threshold τ
- parity p_j

The definition of the decision stump weak learner is:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \tau_j \\ 0 & otherwise \end{cases}$$

In each training iteration the best weak learner $h_j(x)$ is extracted from the feature pool. AdaBoost select the weak classifier with the lowest weighted error over all training-set. The decision stump extracted is denoted by h_j with associated error γ_j . In each iteration a single features is extracted which minimizes the exponential loss function over the weighted training-set. After the extraction of the best weak classifier over all the features, the weighted error of all weak classifiers is update following the AdaBoost algorithm:

$$\gamma_j = \sum_{i=0}^{K} w_i |h_j(x_i) - y_i|$$

Where K is the total number of patterns in the training-set and w_i is the weight of a pattern at the current iteration. The AdaBoost algorithm will select the decision stump h_j with the best error γ_j . The iteration concludes by updating the weights of all pattern.

The algorithm finish when a number of N weak classifiers are extracted or when the classification error on the training sets is below threshold. When the training procedure is finished and the N most discriminative features are extracted, the strong classifier is built, for every weak classifier a weight is assigned with the following criteria: the weight assigned to a weak classifier is inversely proportional to the training error γ_j. The final strong classifier is composed using N hypotheses0.

Algorithm 2 AdaBoost feature ranking for training decision stump

The training data $(x_1, y_1), ..., (x_K, y_K)$
Initial weights $w_j = \frac{1}{K}, j \in (1, ..., K)$
for each stage j=0 to N **do**
For each feature j, train a decision stump h_j
Choose the weak classifier h_j , with the lowest error γ
Update pattern weights
Normalize weights
Add decision stump h_j to the strong classifier
end for
Output the strong classifier $sign(\sum_0^N \alpha_i \cdot h_i(x))$

Testing the strong classifier for can be very expansive in real world, because for a complex pattern such as pedestrian or vehicle, the number of weak classifier is around 2000 weak classifier. In a sliding windows way we need to test sub-windows of the input windows over all scales is extremely large. The computational effort spent in feature extraction and classification is a significant contribution.

Viola and Jones [71, 72] already pointed out the need to decrease the classification time in order to obtain real time performance. Viola and Jones introduced an attentional cascade as extension of the AdaBoost framework, the cascade can discard the non-promising negative pattern. The strong classifiers can be built for rapidly rejection of a fraction of negative pattern in an image. The other stages of the cascade classifier can focus on achieving high overall precision. By proceeding through the cascade the subsequent classifiers become more difficult to pass. A positive result from a classifier is chained to next one. When a pattern is classified with negative

label, the pattern is immediately rejected and excluded from the cascade. This structure is based on the fact that the majority of patterns is negative, so if we can prune the negative patterns we can avoid computational effort, the VJ cascade design is illustrated in Figure 2.4. The original VJ cascade design has the disadvantage that the evaluation information is not propagated between the stages. Consequently, the decision to accept or reject a pattern at a given stage does not take into account how well the pattern performed in prior stages. [73]. The VJ framework consist in a boosting

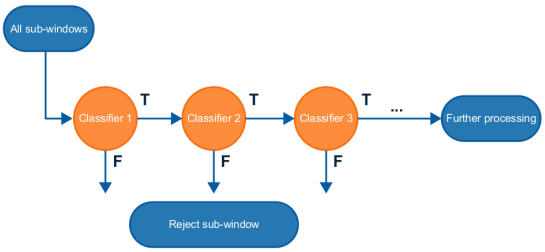


Figure 2.4: Classifier Cascade

cascade of K successive classifiers each with a false positive rate f_i and true positive rate d_i . The false positive rate and the true positive rate of the aggregated classifier is then given by $F = \prod_{k=1}^{K} f_i$ and $D = \prod_{k=1}^{K} f_i$ respectively. Consequently, for each layer of their boosting cascade, the classifier is adjusted to a very high recall to make sure no relevant objects are lost during the classification process. It was shown in [74] that the computational cost involved with evaluating a classifier in the cascade is a function of the false positive rate f_i and the number of weak classifiers N for this classifier. For a given detection task with specified requirements on F and D the problem concerns finding the optimized set of f_i and N-values.

The soft-cascades structure is different, because only one classifier of N weak learners is utilized in stage wise fashion. By determining stage rejection thresholds that control the early rejection of pattern, the same target false positive rates and true positive rates can be achieved. The advantage of this approach is that the difficult optimization process of parameters f_i and N per classifier becomes unnecessary. The approach consists of two main things:

- each stage value need to be changed from a binary-valued decision to a scalar-valued decision, i.e. the decision function is correlated to how well the given pattern passes the stage and the relative importance of the stage.
- the decision function takes into account all previous stages instead only the most recent as with the original VJ cascade [72].

The most important publications regarding soft-cascades are [73] by Bourdev, [75] by Sochman and [76] by Zhang. These works analyze the learning stage rejection thresholds $\theta(n), n \in 1,...,N-1$ for the final softcascade cascade. The chain score $H_n(x)$ of a pattern x is the partial sum up to the current stage n of the strong classifier:

$$H_n(x) = \sum_{i=0}^{n} \alpha_i \cdot h_i(x)$$

The chain scores $H_n(x)$ is non-decreasing, the stage rejection thresholds $\theta(t)$ that these papers attempt to determine are critical to the performance of the cascade classifier. If during classification process, the $H_n(x)$ drops below the stage threshold $\theta(t)$, the pattern is pruned, the window is rejected when $H_n(x) < \theta(n)$. Bourdev et al. [73] introduce the concept of rejection distribution vectors, in this vector is stored the minimum fraction of positive samples v_t that can be missed at each stage, i.e. $v = (v_1,...,v_N)$. The sum of its elements is $1D$ where D is the target detection rate of the classifier. The rejection distribution vectors controls the trade-off between expected evaluation time and false positive rate. The authors propose an algorithm that returns the execution time and false positive rate for a particular rejection distribution vector.

2.4 Convolutional Neural Network

The development of algorithms allowing a machine to learn from very large amounts of data and give a prediction from the data is critically dependent on the process of extracting the most discriminative information from data in raw form. The feature extraction process ca be performed by humans but require a great deal of expertise,

often including trial and error approaches. This process is being replaced by representation learning, this methods achieve the task to automatically learn and discover best representations of the raw data, often thiese methods outperform the traditional handcrafted feature extraction.A representation learning methods known as deep learning will be introduced.

Convolution is often encountered in the context of image processing, where f is the intensity of a given pixel and g is a 2-dimensional weighting function that is called kernel. g is usually non-zero only for a few values in the close neighbour-hood to the central pixel and therefore the sum has to be computed only over those values instead of the whole image. The kernel g is often defined as a small square matrix whose size is called the kernel size k. Depending on the values of the kernel, convolution can be used for several image manipulation operations: If the weights are all positive and for example reflect a Gaussian function with its maximum at the centre, a blurring effect is achieved by convolving the image with the kernel.

The convolutional neural network that Krizhevsky et al. [35] for image classification and object localization had a big impact. This was not only due to the big improvement in classification performance, but it also soon became clear that the convolutional layers of the network learned image features that are applicable for a wide range of vision related tasks like scene recognition and domain adaptation [77]. The network consists of five convolutional layers. Each convolution layer is followed by a Rectified Linear Units layer, that apply the function $f(x) = max(x,0)$ to the pixels of their input. The first and second relu layers are followed by a response normalization step that encourages some small amount of competition between the different output channels of the convolution layers. After each response normalization layer as well as after the fifth convolution layer, overlapping max pooling (pool) is employed. Maximum pooling means that each pixel in the output is set to the maximum value of a small square region (again called kernel) in the input image. In this case, the kernels have dimension 3×3 and are applied at every second pixel, so that the pooling regions overlap. This results in a reduction of the size of the output images by factor 2 with each pooling layer. The first convolution layer also reduces the image size because the convolution is applied with stride 4, i.e. there is one out-

put neuron for every fourth input pixel. Table 2.1 gives an overview over the layers used for feature extraction. While the first part of the layers of the network computes increasingly global feature representations of the input image, the last three layers are only for the classification. Their output is a distribution over the 1000 potential object categories from which the dominant object in the image could be taken.

Convolutional Neural Networks (CNNs) are feedforward neural networks with a layout and architecture specifically designed to handle data arranged in a spatial grid (tensors), such as 2D or 3D images. The networks, like any other ANN, are composed of neurons with weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with an activation function. The architecture is typically composed of several layers, which gives them the characterization of being deep and thus research work on CNNs fall under the domain of deep learning. Essentially, the network computes a mapping function that relates image pixels to a final desired output. In a general CNN, the input is assumed to be an RGB image, i.e. consisting of three channels, corresponding to the red, green and blue color intensity values. Consecutive layers of the CNN may consist of even more channels referred to as feature maps. The number of feature maps typically increase through the layers of a CNN, while the spatial dimension of them decreases until reaching the desired output size. The over-all idea behind this structure is that the representation of the input image is gradually increased in abstraction as it progresses through the layers. Later layers contain more information about the what and how of objects and things in an image, and less of where. Similar to the definition of a feature map, a feature vector at a specific layer of a CNN is defined to be the elements across all feature maps at a given spatial location.

The training of the network was done using an optimization algorithm called stochastic gradient descent (SGD) with mini-batches, momentum and weight decay. For each training iteration i, the image is fed through the network in a forward pass and the loss L of the prediction in comparison to the ground truth label is calculated. Then all gradients of the loss with respect to the weights w of the respective layers are calculated in the backward pass. Finally, the weights of the layers are updated according to

layer	kernel size	stride	out channels	out size	r.f. size	r.f size in %
conv1	11	4	96	56	11	0.24
relu1						
norm1						
pool1	3	2	96	28	19	0.72
conv2	5	1	256	28	51	5.18
relu2						
norm2						
pool2	3	2	256	14	67	8.95
conv3	3	1	384	14	99	19.53
relu3						
conv4	3	1	384	14	131	34.2
relu4						
conv5	3	1	256	14	163	52.95
relu5						
pool5	3	2	256	7	195	75.78

Table 2.1: An overview over the feature extraction layers of the Krizhevsky network. All values are in pixels if not stated otherwise. The convolution kernels and therefore also the receptive fields are square, so only one side length is given. The percental size of the receptive field is given for the expected size of the input images of 224×224 pixels. Note that the input images were downscaled to 256×256 pixels before the 224×224 pixel sized patches are extracted.

$$v_{i+1} = m \cdot v_i - \lambda \cdot \alpha \cdot w_i - \alpha \cdot \left\langle \left. \frac{\partial L}{\partial w} \right|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} = w_i + v_{i+1}$$

Here, $m \cdot v_i$ is the momentum term ($m = 0.9$ was used) that should smooth the updating. $\lambda \cdot \alpha \cdot w_i$ is the weight decay, which is a regularization term that encourages the algorithm to learn small weights. λ was set to 0.0005. α is the learning rate of the algorithm. It started at 0.01 and was reduced by factor 10 two times during the training. The core part of SGD is the last term $\left\langle \left. \frac{\partial L}{\partial w} \right|_{w_i} \right\rangle_{D_i}$, $\alpha \cdot \left. \frac{\partial L}{\partial w} \right|_{w_i}$ is the gradient of the loss with respect to the weights. In normal gradient descend, this derivative would be computed over all training examples. As this is not feasible for big training sets, the gradients are approximated using a number of samples from the training set D_i instead, the D_i are called mini-batches.

2.4.1 Components of CNNs

Convolution

The 2D convolution operation, illustrated in figure 2.5, is defined by a convolution kernel k of size $k \times k$. Given an $W \times H$ input (tensor) X, the convolution kernel is computed over all the pixels of the image. The operator is denoted using $*$ and convolution operation is referred as $Y = K * X$

For each features map layer a convolution ca be defibed as:

$$Y_i = \sigma_i(\sum_{j=1}^{N} k_{i,j} * X_j + b_i)$$

where k is the kernel, b_i the biases, σ_i is an activation function. Another important paraeter of the convolution is the stride, that has the information of which step has to be applied in each convolution.

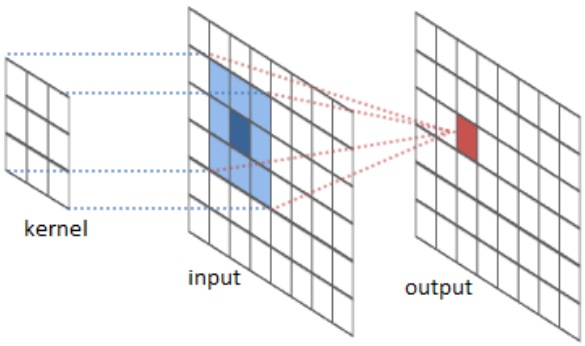


Figure 2.5: example of convolution operator using a 3×3 kernel.

De-Convolution

The inverse operation of convolution is the de-convolution, it can be used for up-sampling, as shown in Fig 2.6. In de-convolution operation, one input activator is associated multiple output activator.

Pooling

The pooling layer is used to reduce the dimension of the feature map, in a pool layer can be associated a kernel size $k \times k$ and stride t, commonly there are max-pooling that perform $max()$ inside the kernel of the feature map, an max-pooling compute the average inside the kernel of the feature maps. The pooling layer provide a way to increase the receptive field.

Unpooling

The inverse operation of pooling is the unpooling, that is used to recover an activator associated to the pooling operation.

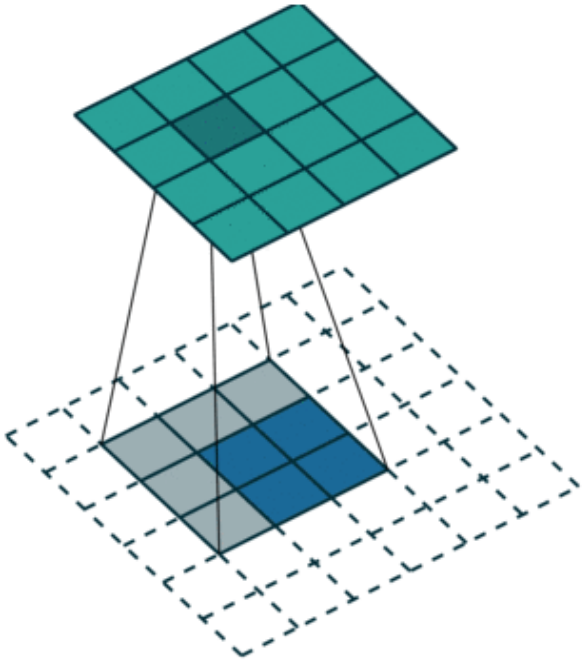


Figure 2.6: Example of the de-convolution operation, one input activator is mapped to multiple output activator

2.4.2 ReLU

The operation of rectifier linear unit is (ReLu) is map an activator input with an activator output, ReLu is defined as $max(0,x)$, an example of ReLU is show in Figure 2.9.

Batch Normalization

In [78] was introduced a technique called mini-batch, a brief explanation of mini-batch: Let a mini-batch of n inputs be $B = \{x_1,..,x_n\}$. Let the normalized mini-batch be $\{\bar{x_1},\bar{x_2},...,\bar{x_n}\}$ and the output of a linear transformation of the batch normalized is represented by $y_1,...,y_n$. The batch normalization transformation is:

$$BN_{\gamma,\beta} : \{x_1,x_2,...,x_n\} \longrightarrow \{y_1,y_2,...,t_n\}$$

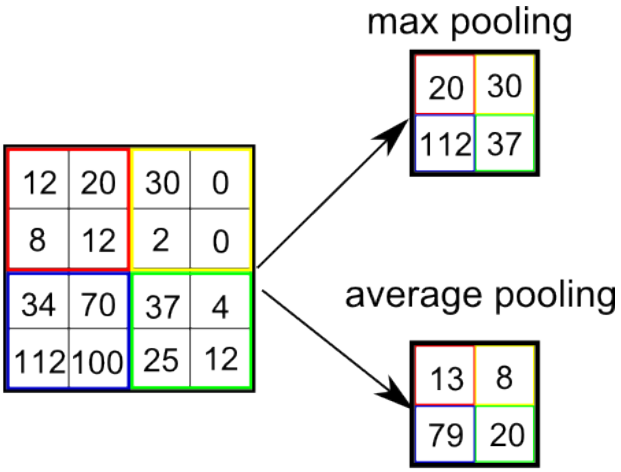


Figure 2.7: Example of the max-pooling operation and average-pooling.

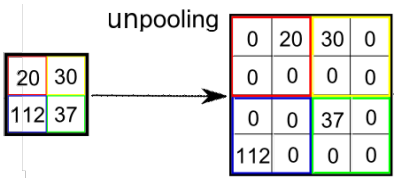


Figure 2.8: example of the max-unpooling operation, the maximum activator is recovered from an associated max-pooling layer.

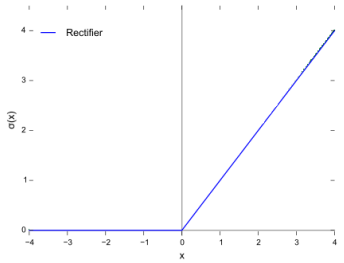


Figure 2.9: aa

where γ and β are parameters that correspond to the scaling and shifting. The mini-batch mean and variance are:

$$\mu_\beta \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i, \sigma_\beta^2 \leftarrow \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_\beta)^2$$

are used to achieve the normalization. Thus, the normalization looks like:

$$\hat{x}_i \leftarrow \frac{x_i-\mu_\beta}{\sqrt{\sigma_\beta^2+\varepsilon}}$$

and the resulting transformation can be given by:

$$y_i \leftarrow \gamma\hat{x}_i+\beta$$

2.4.3 Multiclass Object Detection

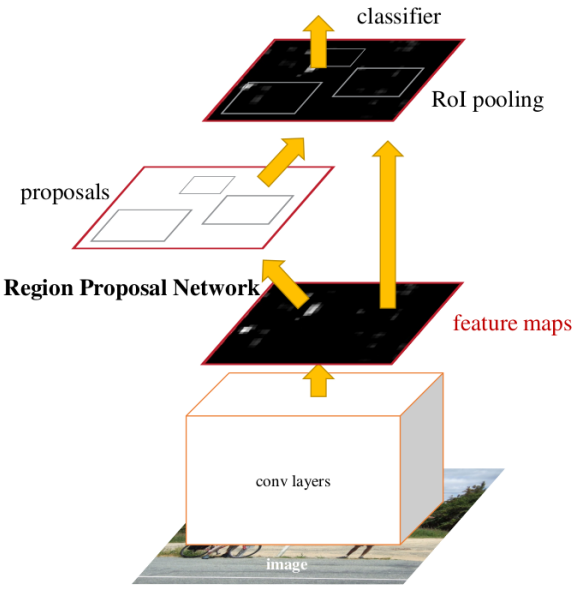


Figure 2.10: Region Propasal Network

Object detection with Faster R-CNN[36], is composed by two modules. The first is a fully convolutional network that find regions proposal, and the second module

is the Fast R-CNN detector that analyze the proposed regions. The entire system is a unified network shwon in Fig. 2.10. A Region Proposal Network takes an input image and will provide a list of rectangular object proposals, each with an objectness score. The goal is to share computation with a Fast R-CNN and the region proposal network [36], both nets will share a common set of convolutional layers[79]. To generate region proposals, a small network is performed via sliding windows over the convolutional feature map output by the last shared convolutional layer. At each sliding-window location, we simultaneously predict multiple region proposals, where the number of maximum possible proposals for each location is denoted as k. The k proposals are parameterized relative to k reference boxes, which we call anchors. An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio2.11.

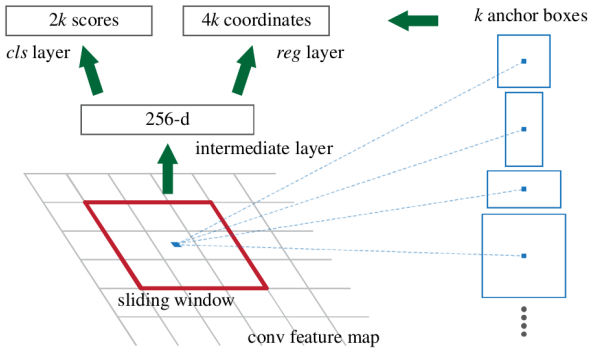


Figure 2.11: Region Propasal Network

The region proposal network is trained image-centric sampling strategy from [36], each mini-batch arises from a single image that contains many positive and negative example anchors.

Chapter 3

System Overview

As mentioned in the previous chapter, obstacles detection and classification is a field that attracts much attention from the research community. Even when narrowed to applications in connection with cars and ADAS, a large body of work exist. ADAS is a challenging domain to work within. Braking system take a short while to be apply, and reaction times must be fast for driving, where fraction of second can be the deciding factor between a collision and a near-miss. At the same time, the system must be robust, so the braking system is not deployed mistakenly (due to a false positives detection), which could itself lead to accidents, or worse, not employ at all (due to a missed detection). In this chapter will be described each part of the system:

- Object Detection and Classification.
- Tracking Monocamera and Multicamera.

The scheme in Fig. 3.2 represents the main steps of the proposed approach:

- Image un-warping:
  the fisheye images are un-warped in order to both correct the lens distortion and obtain a wide-angle view without strong aberrations.
- Classification:

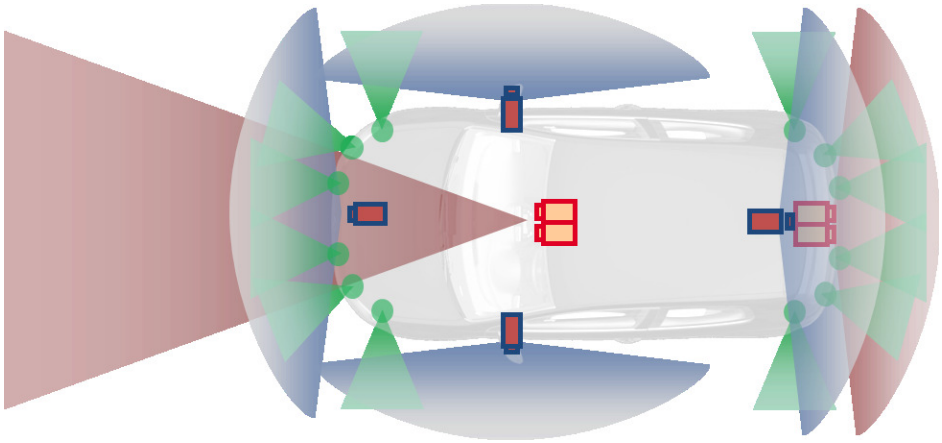


Figure 3.1: V-Charge sensors layout: two stereo cameras, on the front and on the back, for long range narrow angle detection; 4 fish-eye monocular cameras for all-round view obstacle detection; short range collision avoidance sonars (in green).

a classifier is performed on each camera image to detect vehicles and pedestrians on all the surrounding space.

- Multi-camera tracking:
  in order to perform the 360° tracking, firstly an inter-camera association algorithm is performed then, an Unscented Kalman Filter is used to exploit those information to track and generate an all around description, as an unique high-level sensor with 360° field of view.

3.1 Fisheye Images

Fisheye lenses provide very large wide-angle views (theoretically the entire frontal hemispheric view of 180°), but the produced images suffer from severe distortion since the hemispherical scene gets projected onto a flat surface. A procedure to correct the fisheye lens distortion is necessary unless you use an algorithm that takes into account this effect. A common approach involves the distortion correction by

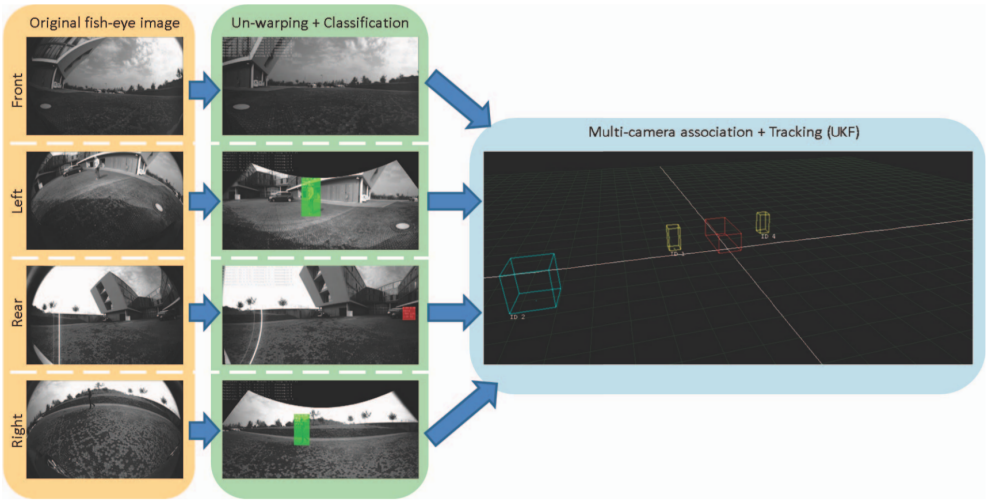


Figure 3.2: Overall system design: from the left to the right it is shown the pipeline of the proposed approach. Firstly, each fisheye image is un-warped; the chosen model allows to have overlapping field of views between adjacent cameras. Secondly, the Soft-Cascade+ACF classifier is used to detect both vehicles (in red) and pedestrians (in green). The outputs from all views are associated each other and finally the Unscented Kalman Filter track all obstacles, allowing to follow an object moving around the vehicle through different field of views.

re-projecting the image on a virtual plane in order to obtain a pinhole image but, the wider the resulting pinhole field of view, the stronger aberration you get on image edges.

In [80] a virtual views layout has been presented in order to correct the lens distortion and exploit the large fisheye lenses field of view. However that approach has some drawbacks: to find all obstacles virtual views need to overlap; this means more pixels to process and conflicted areas where it is hard to determine what is the right detection. To overcome these shortcomings, it has been decided to increase the number of virtual views and merge them together in a single wide image: the final result is equivalent to re-project the original fisheye image onto a semicylindrical surface, as it is done in [81].

Fisheye cameras manufactured to follow the equidistance mapping function, are designed such that the distance between a projected point and the image optical center is proportional to the projected ray incident angle, scaled only by the equidistance parameter. Exploiting this relationship, the resulted cylindrical model is conveniently described by the following equations:

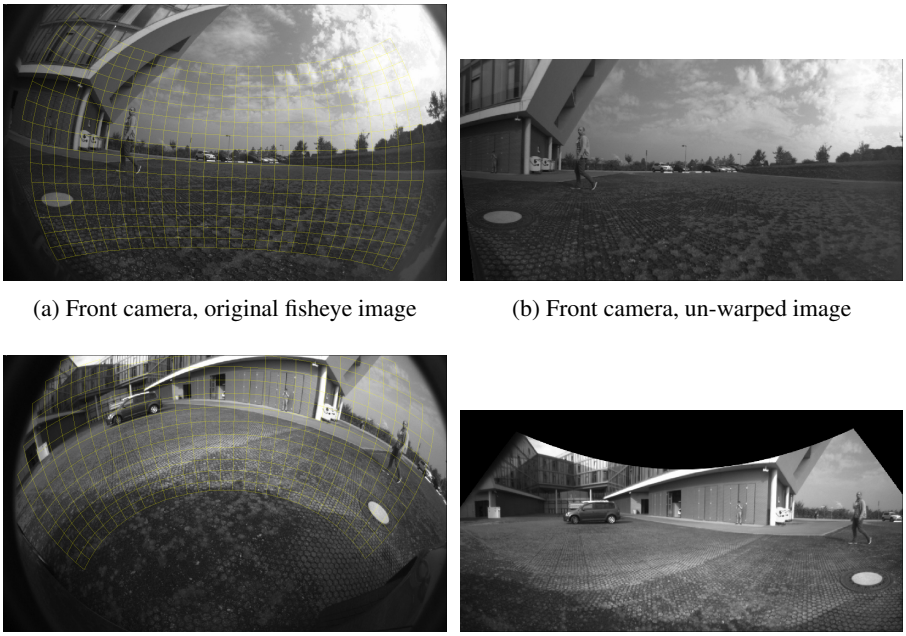
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c_u + \text{atan2}\,(x,z) \cdot f_\theta \\ c_v + k_v \cdot \frac{y}{\sqrt{x^2+z^2}} \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin \frac{u-c_u}{f_\theta} \\ \frac{v-c_v}{k_v} \\ \cos \frac{u-c_u}{f_\theta} \end{bmatrix} \tag{3.2}$$

where (c_u, c_v) is the un-warped image virtual optical center, k_v is the vertical focal length, f_θ is the equidistance parameter, (x,y,z) is the 3D point being projected to the image and (u,v) is the related point in image domain. The Equation 3.1 is the projection of a (x,y,z) 3D point from camera reference frame to cylindrical image reference frame, while the Equation 3.2 is the relation from a (u,v) 2D point from cylindrical image reference frame onto a semicylindrical surface in the camera reference frame. To preserve the original aspect ratio it is convenient to impose $k_v \equiv f_\theta$.

Exploiting the extrinsic, intrinsic and distortion cameras parameters [82] a proper semicylindrical surface has been defined for each fisheye camera; to facilitate the subsequent algorithms, such the classification phase, those semicylinders have been placed just in front of each camera, keeping their axis perpendicular to the vehicle reference z plane. The Fig. 3.3 shows some fisheye images and the resulted un-warped images.

These choices imply that each image row correspond to a circumference arc in vehicle reference z plane; this means that objects on the same row in the same image are at the same distance from the related camera: this property is useful for many optimizations in the classification phase. Noteworthy is also that the whole un-warped image does not respect the pinhole camera model but, locally, the difference between a virtual pinhole camera and the un-warped image is hardly visible; a detailed analysis is covered in section 4.1.1.



(a) Front camera, original fisheye image

(b) Front camera, un-warped image

(c) Left camera, original fisheye image

(d) Left camera, un-warped image

Figure 3.3: Examples of fisheye images and the related un-warped images; into the fisheye images is depicted the corresponding semicylindrical surface. The extrinsic parameters have been also used in the un-warping phase and this is considerable especially on side cameras. Note also the overlapping field of view between the two un-warped images.

3.2 Obstacle Detection and classification

Stereo-vision based obstacles detection is a wide and complex topic, specially referred to automotive applications. In this section, will be described the obstacles detection algorithm starting from a disparity map; as shown in Fig.2.3 and Fig.2.4 in more detail. A dense disparity map is built using a a pair of dedistorted and rectified images, as described in [57]. According the camera calibration parameters, a 3-D point cloud is obtained into world reference system. The $3D$ world points are collected in a $2.5-D$ occupancy grid, called Digital Elevation Map (DEM), with user

defined dimensions. The density of each cell is represented by the number of contained points. In order to discriminate between obstacle and road cell, it is exploited the concept that a cell containing a vertical obstacle has a higher density respect a cell containing an horizontal surface. In this way, cells are classified as obstacle or road only evaluting their density. A more detailed description of the DEM can be found in [83]. This phase permits to assign a classification to each point with a valid disparity and discretize them in a grid. But, a discretization process, if on one hand allows to reduce the problem complexity, on the other hand introduce an error factor depending on the discretization. Then an operation called clustering or segmentation and, according to the data distribution and their origin, can be carry out in several way. In this case, the clustering must be performed on $3D$ point coming from a disparity map and grouped in a cartesian-grid. The disparity computation results in an error of about 0.2 pixel, increasing with the distance, according to the formula 3.2, used to calculate the pixel depth knowing its disparity:

$$z = \frac{Bf}{d}$$

where B represent the baseline and f the pixel focal length of the stereo rig. The depth is represented by the z instead of x since this formula is used to obtain the depth of a pixel in the camera coordinate system, rigid to the stereo rig. This reference system differ to the world reference system for the origin position and different orientation, as shown in Fig.2.6. The other two coordinates are computed using the following equations:

$$x = \frac{B(u-u_0)}{d}, y = \frac{B(v-v_0)}{d}$$

A roto-translation, according to the camera calibration parameters, permits to transform the camera coordinate in world coordinate. Defining with $m_k = (u,v,t)^T$ a generic point in the image plane, it is possible to consider this point as the projection of the world point $p_k = (x,y,z)^T$ on the image plane, so that $m_k = P(p_k)^T$

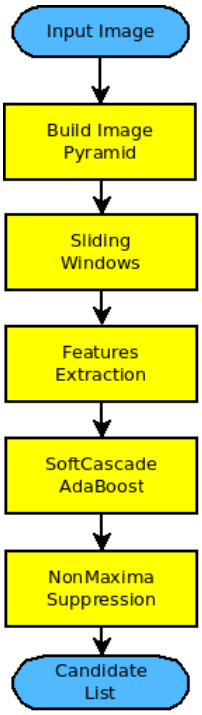


Figure 3.4

3.3 Object Detection and Segmentation

Pedestrians and vehicles detection is based on a Soft-Cascade+ACF classifier. The first phase, regarding the computation of the integral channel, is shared between pedestrians and vehicles detector in order to speed-up the system. Several classifiers with different pattern sizes have been trained reducing the processing time required in the scaled images computation. Five classifiers are used for pedestrians, with dimensions 32×64, 48×96, 58×116, 68×136 and 81×162; two, instead, for vehicles, with dimension 38×38 and 45×45. Two scales for each classifier are computed. The pedestrian classifier with pattern size 32×64 is used to detect only far pedestrian: differently from the other classifier, it is run just at the smallest octave for each scale.

Moreover, the detection is not performed in regions that do not respect the typical range of vehicle/pedestrian size: for each candidate window, through the world-to-image coordinates transformation, its theoretical area is calculated and is checked if it respects the suitable dimensions for a vehicle or pedestrian. Candidates at the same scale-octave are filtered with a 3×3 non maxima-suppression. The final training set is composed by 48000 cars images from a private dataset, 4000 pedestrians images from KITTI+INRIA datasets and 5000 negatives examples, randomly extracted from the initial negatives images. Positive samples are enlarged by 8% to include also the information contained in the border. 5 bootstrapping cycle are performed to improve the detection performance and reduce the false positive rate. At each cycle, the detector is trained on an augmented set composed by the initial negative samples and the hard negative ones obtained by the previous cycle. In Fig 3.4 is show the pipeline for the objects detection with sliding windows over the entire image, this process can be speed up by using geometrical constraints (pedestrian, car) and regions of interest provided from the stereo to prune the number of windows that need to analyzed. Another way to prune the search windows is to use the constraint about the object that we are looking for 3.3.1.

The object detection in the scene can be performed using as initial guess the region of interests provided from the obstacle detector due to avoid a sliding windows over the entire image. When we need to detect multi-class of object the good solution is the convolutional neural network, the region of interest provided from stereo are not used, because the faster R-CNN there are the initial layer that can extract the regions of interest for classification as explained in faster-rcnn[36]. The network is VGG-16 pre-trained over COCO dataset and then fine-tuned over kitti dataset. The region proposal is the main bottleneck in object detection. In R-CNN, the region proposal is performed in two methods:

- selective search: the regions are proposed by network that provide the region of interest in the image, but the network does not share information with RCNN, this slow down the entire process.

- region proposal network (RPN): the network for object detection share layer

with the region proposal network based on convolutional features. The region proposal method is shown in Fig 3.5.

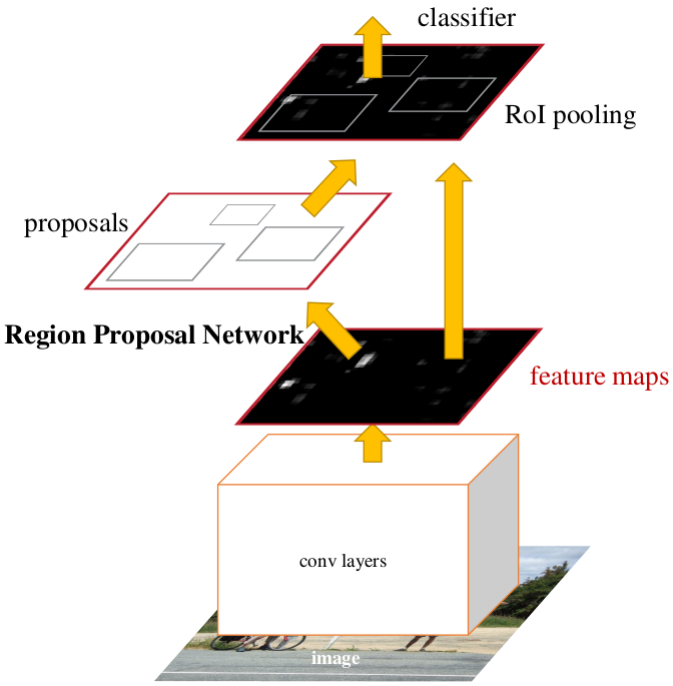


Figure 3.5: Region Proposal

To crossvalidate the obstacle provided from the obstacle detector, can be performed a segmentation with an architecture encoder-decoder trained over cityscapes dataset, an example of segmentation is provided in Fig. 3.7, but using these techniques it is not possible to achieve real time performance without a GPU. The segmentation is performed using a deconvolution approach. The approach uses a feature extractor that learn how to perform dense pixel labeling prediction from a feature vector followed by repeated up-convolutions and max-unpooling operator. The architecture of the network is a the pre-trained VGG 16-layer model [79], followed by a mirrored version of the down-sampling network, which start from feature vectors as inputs and reconstructs the dense pixel labeling.

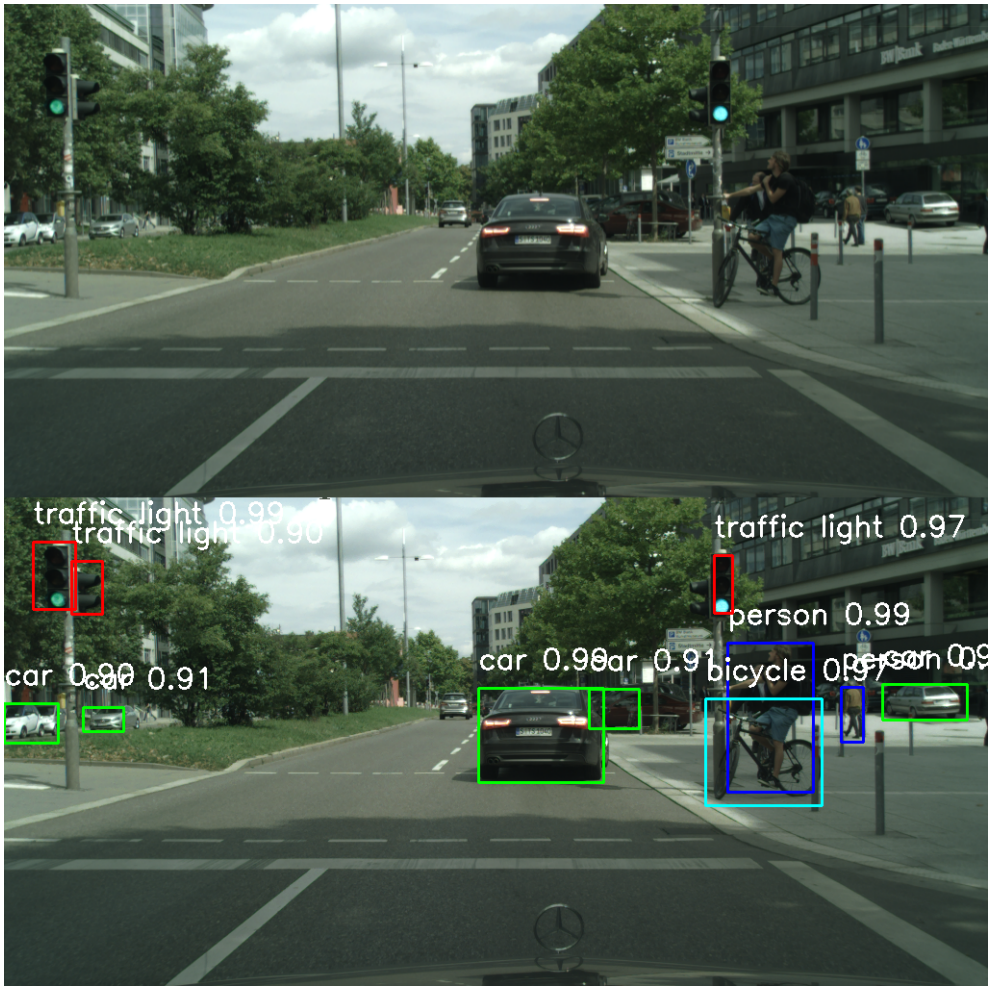


Figure 3.6: Multiclass Object Detection using FasterRCNN trained over 4 classes(vehicle, pedestrian, traffic light, bicycle)

3.3.1 Search Ranges

If we know where the horizon of the image is, it is possible to discard many hypotheses based on size object size. This not only allows us to speed-up computation, but also to decrease false positives. Using perspective information as an additional

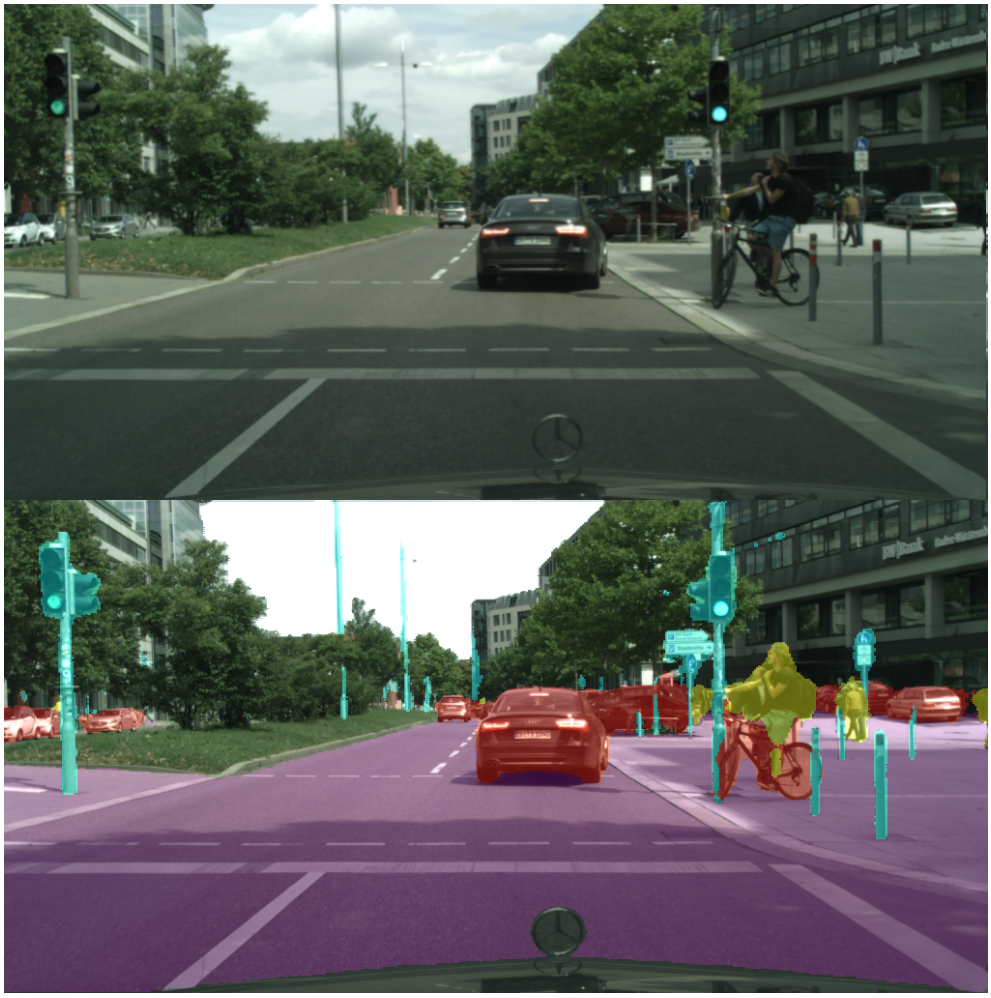


Figure 3.7: Encoder-Decoder trained for segmentation over Cityscapes

constraint is useful for any multi-scale, sliding-window approach, and it can reduce execution time considerably, depending on how strict the constraint is. Our objective is to avoid calculating the filter convolution in areas of the image where the filter size is not congruent with perspective information. In our case, different considerations must be made, as we calculate convolutions in the frequency domain, and we use dif-

ferent resolutions for the root and part features. We use the details of the camera lens and its position to learn the perspective center, and use two parameters, $W0$ and $W1$, defining the minimum and maximum pedestrian width in meters. From these parameters, for each line of the source image, we can compute the minimum and maximum plausible width of a pedestrian laying on that specific line, as shown in figure 3.8. Search ranges can greatly speed-up computation and prune false positive detections, but it is important to choose them carefully to avoid pruning useful detections.

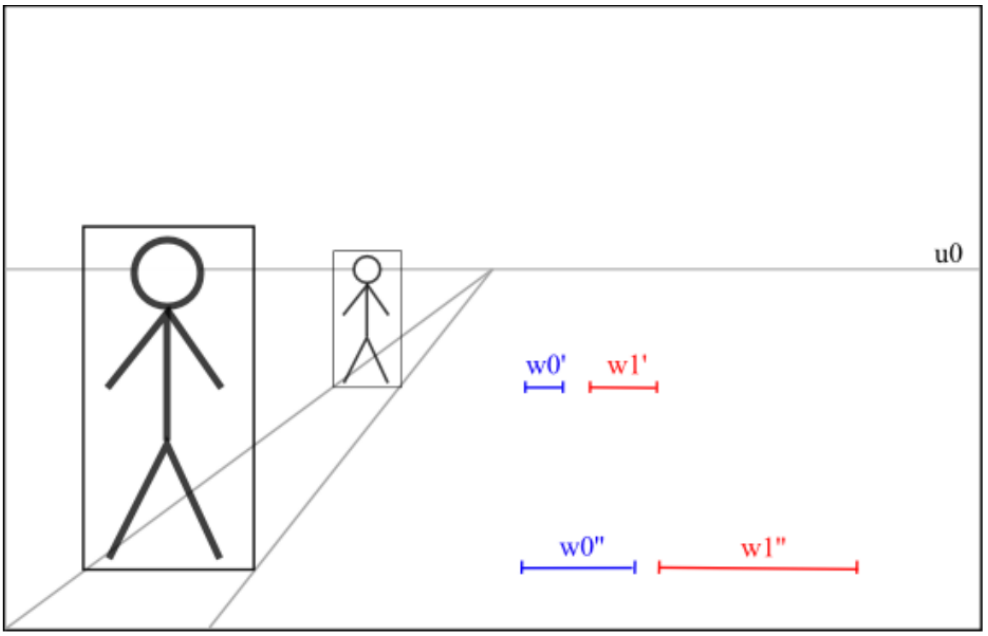


Figure 3.8: Given the information about the horizon position $(u0)$, for each scanline of the image we can estimate a minimum (W_0^1) and maximum (W_1^1) plausible width of a pedestrian laying on that line, based on absolute parameters $W0$ and $W1$ representing the minimum and maximum width of pedestrians in meters.

3.4 Motion Estimation

The object pose in the space is defined by six parameters, degrees of freedom, which represent rotation and translation respect to a defined reference system.

$$x = [\theta, \phi, \psi, t_x, t_y, t_z]$$

The object motion is represented by the first order derivative, respect to the time, of the equation

$$\dot{x} \approx \Delta x = x_t - x_{t-1}$$

The motion is estimated using the movement of the single image pixels. Two phase are required: the ego-motion estimation and then the object motion estimation. The ego-motion estimation is required since using the pixel movement between two frames it is possible only to obtain the relative motion between the host vehicle and the obstacles. Combining the ego-motion with the relative obstacle motion we can obtain the absolute obstacle motion. The pixel motion is obtained using the image pair of two subsequent frames.

The optical flow is the most common technique, in the state of the art, to obtain motion information from subsequent images. The algorithm consists in the extraction of the image points with the most informative content, keypoints. The extraction is based on the algorithm described in [47], while a brute force search is used in the matching phase. An example is shown in Fig.2.15, where the motion is represented by a bidimensional vector. The image prove as the motion of the host vehicle affects also the motion of the obstacles.

Only the points belonging to the static obstacles can be used to estimate the motion of the host vehicle, since their movement is affects only by the movement of the vehicle. The obstacles detection algorithm, is used to separate points belonging to the obstacles: not being able to distinguish between static or moving obstacles, it has been considered all moving obstacles. Using a stereo-pair images, each keypoint has associated with a 3-D world position. In this way it is possible, basing on keypoint position, to discriminate if the points belong to an obstacle or not.

Figure 3.9: Sparse optical flow

3.5 Monocular tracking

The tracking process is based on an Unscented Kalman Filter (UKF). The Kalman state is represented by: the projection of the vehicle rear axis center points on the ground (tracking point), the vehicle width and its longitudinal and lateral speed. Position and speed are relative to the host vehicle, so at each step the predicted position of the tracked element is obtained from the composition of the host vehicle movement (provided by inertial sensors) and the target vehicle movement. Tracking point and vehicle width in the image coordinates form the observation: assuming that the point lies on the ground, it is possible to obtain these image coordinates through the pinhole camera model. Knowing the 2-D tracking point position and the vehicle width

in meters (Kalman state), the boundary position is calculated in image coordinates. The observation point is represented as the center along the horizontal axis of these two points, and the vehicle width in pixels, as the difference along the same axis.
The associations between new and tracked candidates are performed by building an overlap matrix and using the Hungarian algorithm to find the best matching that maximize the total overlap. Two elements are definitively associated if the overlap is higher than a certain threshold 3.10. The overlap is calculated as follow:

$$overlap = \frac{area(BB_{candidateA} \cap BB_{candidateB})}{area(BB_{candidateA} \cup BB_{candidateB})} \tag{3.3}$$

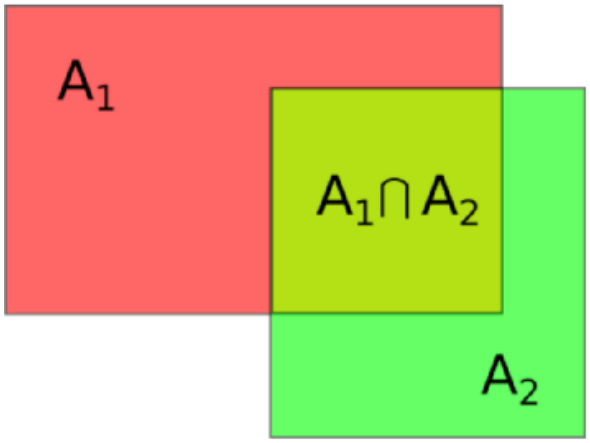


Figure 3.10: Regions overlapping.

where *BB* is the candidate bounding box. As shown in Fig. 3.9, detection and prediction are used as sources to generate new candidates. The first list of candidates is the detection process output, the second list, instead, is obtained from the features matching process.
Features are searched based on a multiple local convolution, key point and descriptors, extracted from two different hash images as described in [84].
Features are matched between previous and current frame and, to each features pair, it is assigned an object membership if their position is inside a tracked vehicle bounding box. If there are enough features belonging to a vehicle, they are used to compute the

optical flow and obtain the “features” candidates. In the association phase, to avoid drift due to the features matching process, candidates coming from the detection step have an higher priority: if there is a matching with a detection candidate, it is used to make the Kalman observation. Otherwise, overlap with features candidate is checked and, in case of matching, it is used to make the update the filter.

3.5.1 Tracking and coarse-to-fine classification

Stability and robustness are fundamental to implement a practical system. To cover these requirements, a specific stage of classification and tracking has been added. Tracked vehicles with low uncertainty are marked as 'triggered' and removed from the 'normal' detection pipeline. They are inserted in a new stage of classification/tracking where a more detailed classifier will be run in the neighborhood of previous detections area, enlarged according to Kalman covariance. A reduced set of detection scales will be analyzed, obtained from the tracked object uncertainty: by knowing the range of variation in the obstacle size, it is possible to compute the minimum and maximum scale of detection. This leads to an improvement in terms of detection: by increasing the number of scale per octave (that is not possible on the entire image because the computational time would be too high), the approximation error, due to the scale factor, decreases significantly. Triggered objects classification is performed in parallel: a new classifier is allocated for each object, sharing the same precomputed integral image with different search areas and scale/octave limits.

3.6 Multicamera tracking

Tracking is performed jointly through the four triggered cameras allowing to follow an object moving around the vehicle.

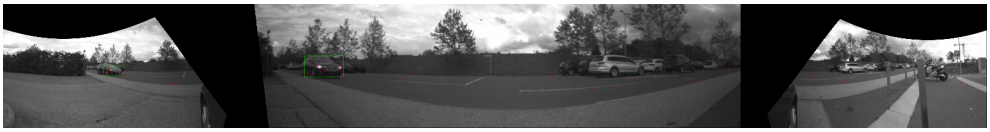
3.6.1 Association

The association process is responsible to create a relationship from the new detected obstacles and the previous tracked ones. Considering a case with N new obstacles and

M tracked ones. The first step needed is to associate the new obstacles to the tracked ones. Often, $N \neq M$; this means both that not all the observations can be associated with a tracked object and both that not all the tracked objects can be associated with an observation. In the first case, the obstacle has appeared in the scene and it was observed for the first time, so it is inserted in the tracked list. In the second case, the tracked object has not been observed in this frame; if this event is persist in several frame, the object is removed from the tracked list since it means that it disappeared from the scene. The association value between two objects, is a value ranging from -1 to 1 and it is estimated using a classifier[73]. The all-around tracking requires that the association must be performed not only between objects of the same class in subsequent frames, but also between objects of the same class but coming from different cameras. The matching in subsequent frames is performed using the bounding boxes overlap. Several comparison metrics, instead, have been analyzed for the multi-cameras one:

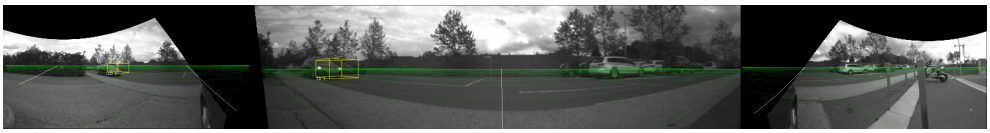
- Euclidean distance: objects coming from all the cameras are projected in the world coordinates space and the euclidean distance represents the matching cost.
- Polar distance: objects are still projected in the world coordinates but, in this case, the polar distance represents the matching cost.
- Image distance: objects on each camera are projected on other cameras (if the projection is possible) and the matching cost is represented by the bounding boxes overlap.

The Image distance metric has been selected for the multi-camera matching process because achieves better results and its metric is coherent with the subsequent tracking algorithm. An association based on features matching between different cameras has been also tested but discarded due to the high computational cost for the features extraction and matching. Once defined a comparison metric, the Hungarian algorithm is used to find the best association and maximize the matching cost. After this step, each tracked object may have been associated with one or two elements, depending if it has been seen in one or more cameras.



(a) Feature extraction and clustering

(b) Tracking and association



(c) Multi-view exploitation

Figure 3.11: Example of obstacle tracking using features, when the vehicle stops at a crossroad. (a) The features are extracted and grouped. (b) The clusters are re-projected into the other views and associated. (c) The obstacle position is computed in the world reference coordinates.

Dynamic objects generate moving features, it is possible to cluster similar features in the image and track the cluster on behalf of the actual object. The clustering is executed exploiting position and movement information of the features, refining at a later stage the result by grouping small similar clusters that share the same region of the image.

The clusters obtained are tracked in the image domain and are also used to generate the obstacle in the vehicle reference frame. Since the fish-eye cameras present overlapped field of views, an obstacle may be visible in multiple cameras, hence each obstacle derived from a cluster tracked in an image is then re-projected into the other views and associated with potential nearby clusters.

3.6.2 Tracking

An Unscented Kalman Filter (UKF) is used for the tracking process; the filter state is composed by the tracking point, the object width and its relative speed. The predicted position is obtained combining the host vehicle movement information (from the inertial sensor) and the object position and speed. While the filter state is in the vehicle reference frame, the filter observations are in the images reference frame.

The bounding box extraction process requires the conversion of the top-left and bottom-right points of the object in the world coordinates and, then, the projection of these points in the image coordinates. In case of an object in the front camera, these points can be calculated subtracting and adding on the x axis the width value to the tracking point. This procedure, instead, would not be valid for objects in different cameras. Then, a different method has been used for the bounding box extraction: firstly, the tracking point and the top center point of the object are projected in the image coordinates; from these two points it is possible to extract the bounding box height in pixels and use it to find the width and, then, the top-left and bottom-right points in image coordinates starting from the projected tracking point.

As described in the 3.6.1 section, one or two elements can be associated to a tracked object after the matching process. In case of a “single” association, the observation is represented by the tracking point and object width in image coordinates: through the Equations 3.1, 3.2 is possible to switch between image and world coordinates. In case of a “double" association, instead, the observation is represented by a couple of tracking points and widths in pixels from the objects in the two different cameras. In this case, the predicted observation are the projections of the object state in the two image coordinates. Extra care is required to manage the two different obstacles types, pedestrians and vehicles. This involves different filter parameters, regarding the object movement and, the different extraction of the object bounding box from its state: while vehicles have the same ratio between height and width, the pedestrians height is twice their width.

Mono obstacles

$$
\begin{aligned}
y_m(t) &= \begin{bmatrix} u_{pixel} \\ v_{pixel} \\ h_{pixel} \\ w_{pixel} \\ d_{pixel} \\ V^b(t) \\ \omega_\psi(t) \end{bmatrix} = h_m(x(t)) + \nu_m(t) \\
h_m(x(t)) &= \begin{bmatrix} t\left(p^b(t)\right)_x \\ t\left(p^b(t)\right)_y \\ \left| t\left(p^b(t)\right)_y - t\left(p^b(t) - \begin{bmatrix} 0 & 0 & H(t) \end{bmatrix}^T\right)_y \right| \\ \frac{K_u(l_x(t)-r_x(t))}{(R_b^c g_{ctr}^{idx}(p^b(t)))_z} \\ \frac{K_u(f_x(t)-b_x(t))}{(R_b^c g_{ctr}^{idx}(p^b(t)))_z} \\ R_n^b V^n(t) \\ \omega_\psi(t) \end{bmatrix} \\
\nu_m &= \begin{bmatrix} \nu_u(t) & \nu_v(t) & \nu_h(t) & \nu_w(t) & \nu_d(t) & \nu_v(t)^T & \nu_{\omega_\psi}(t) \end{bmatrix}^T
\end{aligned}
\tag{3.4}
$$

Chapter 4

Results

The system evaluation has been divided in two parts: the object detection and the obstacles tracking. A qualitative and quantitative analysis has been carried out for single stereo head camera using TME Motorway dataset, quantitative results will be shown for multiclass object detection and semantic segmentation.

4.1 Dataset

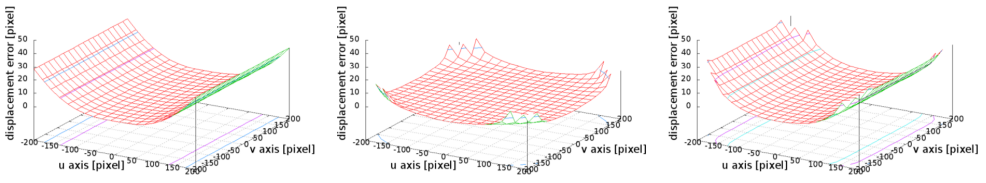
4.1.1 Fisheye object classification

The basic assumption of our approach was the classificator compatibility with cylindrical images; since no object detection evaluation dataset with fisheye images was available, we chose to evaluate "differences" between un-warping fisheye images using pinhole model and cylindrical model. For this purpose we defined "difference" d as follow:

$$d(u,v) = \left\| (u,v) - (u',v') \right\| \tag{4.1}$$

where (u,v) is a point in image domain and (u',v') is the point obtained by re-projecting (u,v) from pinhole model to cylindrical model.

This analysis helps to understand the expected object distortion given its size in the equivalent pinhole model, and consequently it is possible to estimate the max-



(a) Displacement errors along u axis

(b) Displacement errors along v axis

(c) Euclidean norm of displacement errors

Figure 4.1: Displacement errors between pinhole model and cylindrical model. In (a) and (b) are shown the displacement errors along u and v axes, in (c) is depicted the Euclidean norm of displacement errors as described by Equation 4.1. The greater the error is, the higher the displacement between pinhole model and cylindrical model is at that coordinate.

imum acceptable size. In Fig. 4.1 the results[1] of this analysis are shown: firstly we evaluated the per-axis re-projection displacement errors, then we computed Euclidean norm as described by Equation 4.1.

Predictably, the main displacement error occurs along u axis: points further than 150 pixels have displacement error u component of approximately 20 pixels. With a similar analysis of "Euclidean norm of displacement errors" graph (Fig. 4.1 c) we can overestimate the error of objects as wide as 200 pixels in less than 10 pixels, and less than 20 pixels for object as wide as 300 pixels; along the v axis we have less constraints.

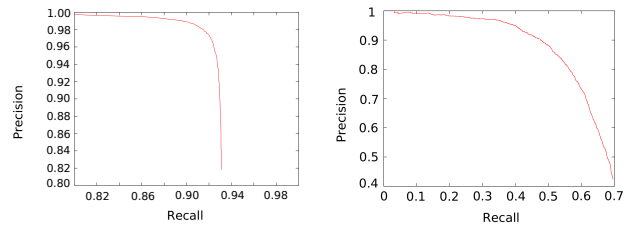
Two types of metrics have been selected to test the classifier behavior:

- precision/recall
- miss-rate

The precision/recall results are depicted in Fig. 4.2: the left plot is related to the vehicle classification, while the right one to the pedestrian classifier.

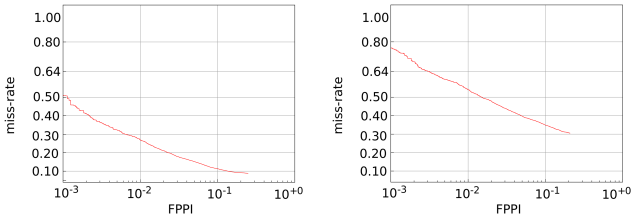
The miss-rate results are shown in Fig. 4.3, as before the left plot is relative to vehicle while the right to pedestrian.

[1] These results are computed using the real intrinsic camera parameter utilized in our system: $f_\theta = k_v \simeq 266.67$ [pixel].



(a) (b)

Figure 4.2: Precision/recall metric results for the vehicle (a) and pedestrian (b) classifiers.



(a) (b)

Figure 4.3: Miss-rate metric results for the vehicle (a) and pedestrian (b) classifiers.

The reported results are in line with the state-of-the-art.

For the evaluation of the 360° tracking the classifier performance has been analyzed over images unwarped using the current cylindrical model. Since there are no public datasets available online with annotated fish-eye images, annotated pinhole images have been warped according to the cylindrical model; then the impact of the cylindrical model on the classifier performance has been studied by comparing results on original images and cylindrical images.

Three classifiers have been used, a vehicle classifier with dimension 38x38 and two pedestrian classifiers with dimensions 32x64 and 48x96. Based on the results shown in Table 4.1 the results presented previously can be still considered valid since the usage of the cylindrical model has a minor influence on the classifier performance.

Classifier	Sample #	Model	Correct Detection
Pedestrian 32x64	130017	Pinhole Cylindric	96.7127% 96.7150%
Pedestrian 48x96	67158	Pinhole Cylindric	98.1148% 98.1044%
Vehicle 38x38	58120	Pinhole Cylindric	82.4260% 82.3623%

Table 4.1: Correct detection comparison between pinhole and cylindrical model

Exploiting the overlapping field of view the 360° tracking improve the reliability and accuracy of the whole system allowing to perform well even in challenging scenario like objects occlusion or crossing objects. This type of situation can occurs frequently in parking lot environments where a single camera could struggle to handle them; an example is shown in Fig. 4.4 where the algorithm manages to handle an occluded pedestrian. Table 4.2 respectively reports the correct and false detection rates for both pedestrian and vehicle candidates.

Class	Correct detection	False detection
Pedestrian	1179 (95.9%)	51 (4.1%)
Vehicle	2920 (98.1%)	57 (1.9%)

Table 4.2: Correct and false detection for pedestrian and vehicle 360° tracking

4.1.2 TME Motorway Dataset

TME Motorway Dataset [1], used to benchmark the system has been selected from the acquisition made in Northern Italy in December 2011 using the BRAiVE test vehicle [85]. In the Fig.4.6 is shown a comparison between the baseline detector (it does not use the coarse-to-fine classification in tracking phase), the improved detector (it use coarse-to-fine classification in the tracking phase) and the results of TME Motorway Dataset. The system has the best performance from 0 m to 60 m, after this distance it exhibit a worse performance because the pattern has a lowest size of 32×32. From 0 m to 50 m it is possible to see the improvement of coarse-to-fine classification. The classification is performed on 32×32 pixels patterns. The training set at the final bootstrapping cycle is composed approximately by 75000 rear images of cars and 11000 negative samples. To generate negative examples from 640×480 images, several windows are sampled randomly from 10000 negative training images provided from the initial negative training set. The first detector obtained from this list of negatives is applied to all negative training images to find hard negative examples, then a new detector is trained using this augmented set (initial negative with hard negatives) to produce the second detector. This procedure was repeated up to 10 times. The test-set used to generate the benchmarks shown in this section is composed with 2000 annotated frames. The AdaBoost classifier is transformed into a Soft-Cascade classifier to improve the speed of the detector. The algorithm performance has been tested in the acquired images, evaluating with the defined validation set, both the AdaBoost classifier and its Soft-Cascade version. By comparing the correct detection rates obtained through the application of the two classifiers, it has been proved that, from a detection performance point of view, they provide equivalent results: measuring the classification capabilities for the validation set designed, either the traditional AdaBoost and its Soft-Cascade version provide the same correct detection rate on the validation set. Experimentally, it has been also demonstrated that the Soft-Cascade design allows to reduce 5 times the computational cost: the AdaBoost classifier trained with 400 weak classifiers and its SoftCascade version with a rejection rate of 0.04 has been tested; the traditional AdaBoost classifier needs all the weak classifiers (400) to correctly discard a negative sample, while with the Soft-

Cascade scheme the same negative pattern is correctly classified in average after only 23 weak classifiers. The final system utilizes a detector dedicated that use the search ranges and does not analyze the region already tracked to improve the detection's time for each target. In Fig. 4.6 and Fig. 4.7 are show the results of the system on the TME Motorway Dataset. The entire system run on Intel core i7 4 cores at 2.66 GHz with an average time of $\sim$ 35.6 ms.

Component	Average Time
Features + Matching	4.8 ms
Object Detector	28 ms
Tracking + Re-classification	1.5 ms
Prediction	1.6 ms
Total	35.6 ms $\sim$ 28 fps

Table 4.3: Computational analysis of each modules of the system performed on the entire TME Motorway Daylight dataset.

4.1.3 Cityscapes

The Cityscapes data set is a large-scale data set with images captured with a stereo camera positioned on the front of a car. Cityscapes dataset contains a stereo images sequences recorded in street scenes from 50 different cities. The images are annotated pixel-wise with different classes 4.4. The dataset contains 5000 finely annotated images, an example is show in 4.8 and about 20000 coarsely annotated images. The coarse annotations were generated by polygonal labeling and thus do not clearly distinguish between objects and instances in a scene, while the fine annotations were fine-tuned to pixel-level. The images were acquired and provided at a resolution of 2048×1024 pixels. The finely annotated data is divided into training, validation and testing sets, consisting of 2975, 500 and 1525 images.

Category	Classes
Flat	Road, Sidewalk, Parking , Rail track
Human	Person , Rider
Vehicle	Car, Truck, Bus ,On rails, Motorcycle, Bicycle, Caravan Trailer
Construction	Building, Wall, Fence, Guard rail, Bridge, Tunnel
Object	Pole, Pole group, Traffic sign, Traffic light
Nature	Vegetation, Terrain
Sky	Sky

Table 4.4: Cityscapes dataset.

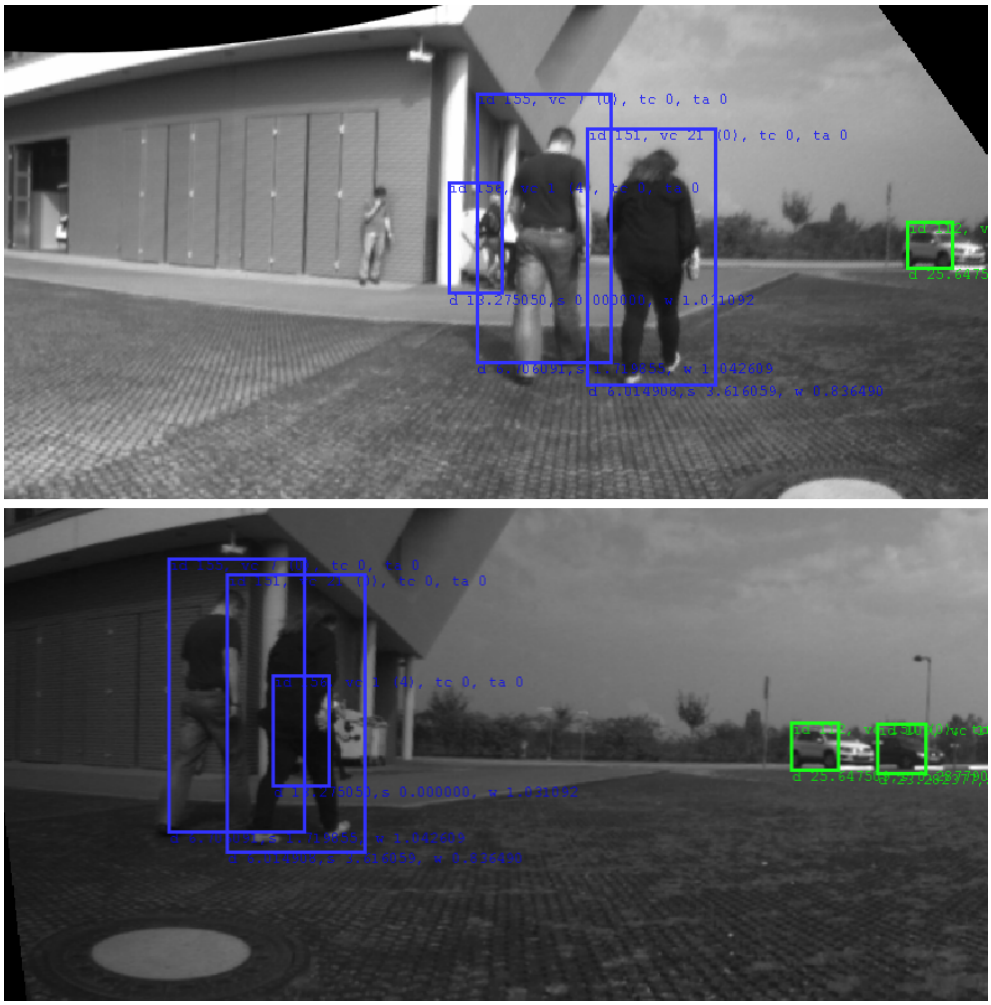


Figure 4.4: Situation where 360° tracking gives benefits: the multiple points of view allow to overcome an occlusion (the far pedestrian is visible only in the top image)

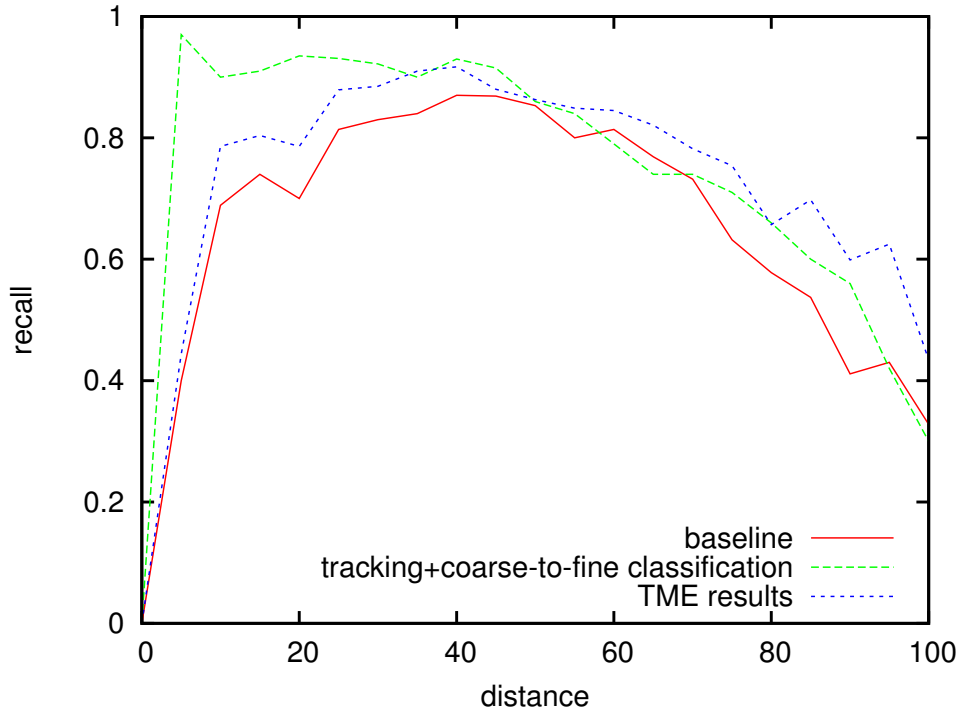


Figure 4.5: Recall in function of distance in TME DayLight Dataset. The comparison involves the baseline detector (without coarse-to-fine classification in tracking phase), the detector that use tracking+coarse-to-fine classification and the results of TME [1]

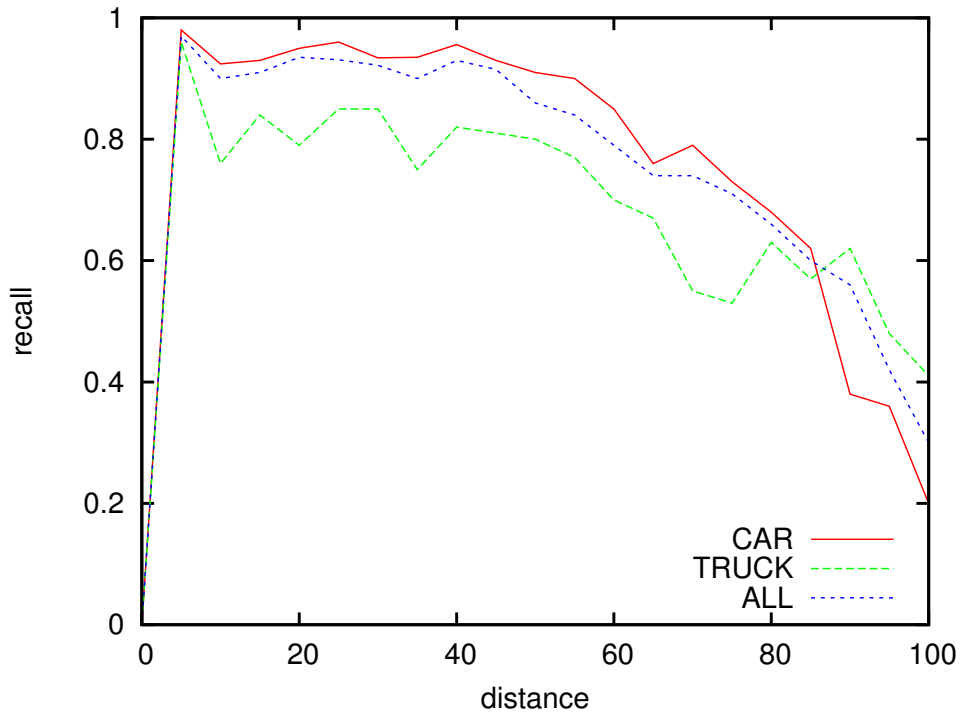
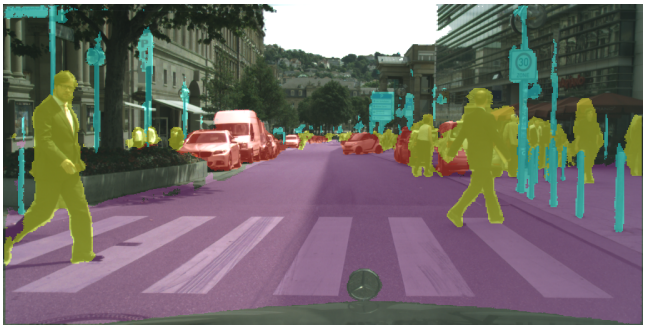


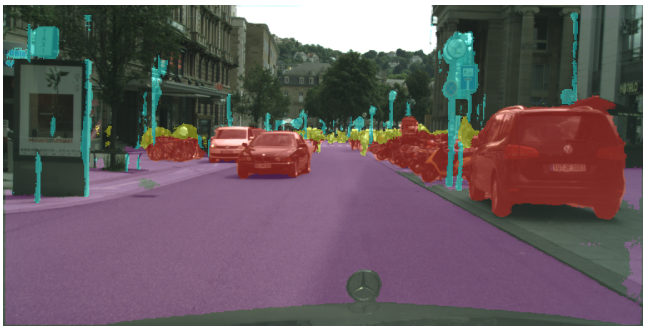
Figure 4.6: Recall rate in function of ground truth distance. [1]

Figure 4.7: In these images is shown the output of the system, each vehicle detected is represented with different bounding boxes: green for vehicles belonging the same lane of host vehicle, yellow for vehicles in the same direction of host vehicle but different lane, red for vehicles in opposite direction. (*a*) and (*b*) are referred to TME Motorway Dataset, (*c*) and (*d*) are referred to our dataset.

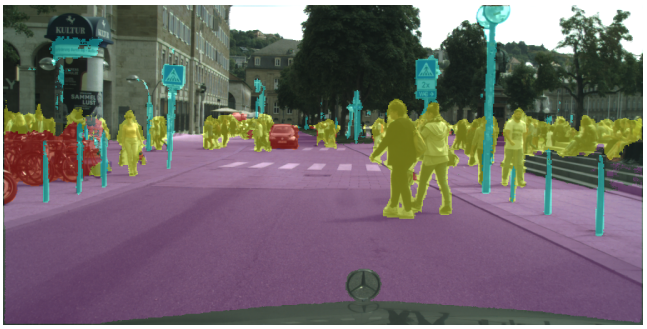
Figure 4.8: Example of pixel-wise annotation of different classes.



(a)



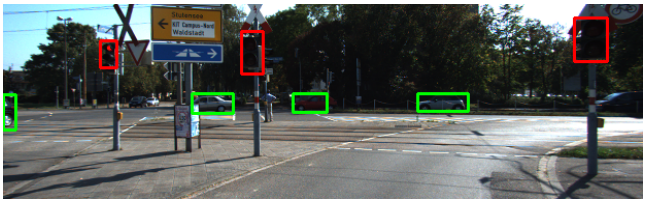
(b)



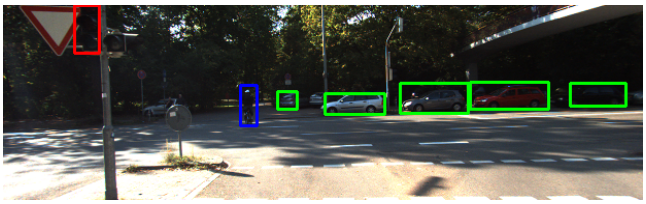
(c)

(d)

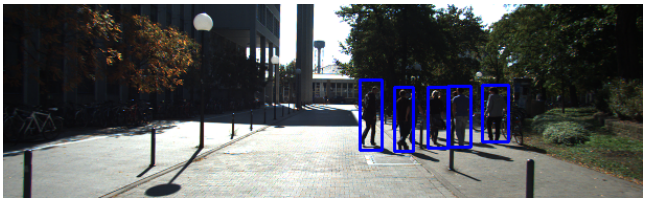
Figure 4.9: Examples segmentation with encoder-decoder network over cityscapes.



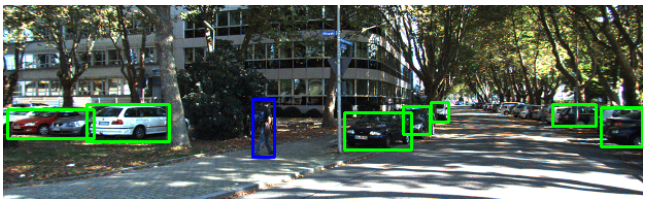
(a)



(b)



(c)



(d)

Figure 4.10: Examples multiclass object detection on Kitti.

Chapter 5

Conclusions and Future directions

The dissertation closing arguments are presented in this chapter. A critical assessment with contributions and conclusions, future works and important directions will be presented.

5.1 Summary

The described system is able to detect object of certain classes in a sorround view of 360°. The objects are, then, tracked and fused in the multi-camera tracking. Several automotive application can be derived by the proposed approach: When obstacles is covered by th front camera, the depth information of the disparity map is used, to estimate the distance of teh object. The visual odometry on points not belonging to an obstacles has been used to extract motion information of the host vehicle. The visual odometry on obstacles points, instead, has been used to obtain the relative objects motion information. Their combination has allowed to obtain the absolute motion information for the obstacles. The segmentation could be improved, especially in complex environment as the urban scenario with group of pedestrian, using also the semantic segmentation rovided from pixel labeling. After the detection, the obstacles are classified as pedestrians or vehicles using a SofCascade Adaboost with ACF or Fast-RCNN. An UKF based tracking step is performed after the classification. The

approach has been compared with the state of art. Moreover, the system works at higher frame rate, making it appropriate for automotive applications.

5.2 Conclusions

The system can provide a reconstruction of the dynamic world surrounding the vehicle, proving to be able to help the driver in the assessment of critical situations.
In particular, the developed algorithm provides a stable, robust and reliable detection, classification and tracking of the multiple targets coming from different cameras. Moreover, the proposed approaches were seen to outperform the state of the art approaches on a public dataset.
A combination of monocular and stereo classification in combination with an hard time constraints inserted in the classifier pipeline, has led to significantly speed up the system. This is fundamental for automotive applications, where real-time processing is a strong constraint; especially given that the classification step is, often, the most demanding in terms of time.
A fault tolerant and reliable system requires sensors redundancy and complementarity. Common approaches rely on object level fusion where only high-level information are used. This leads to a fast processing time but, at the same time, produces poor results being unable to exploit the specific sensor data.

5.3 Direction for Future works

The thesis has covered a large area in the field of detection, tracking and fusion of obstacles, which leaves considerable possible improvements. In the following, interesting directions for future works are sketched.

- Use multiple stereo camera to cover the entire sorround view of the car, this lead to have disparity information of the objects, that is useful to consider during the tracking phase.

- The segmentation with pixel labeling can improve the performance when the obstacles need to be associated.
- The multiclass object detection can be used to improve the number of classes that the system can handle.
- The filter can be extended with a constrained Kalman filter with a kinematic model, in order to force the vehicles to move under kinematic constraints. This requires to have a different filter and evolution for vehicles and pedestrians and other classes that we need to handle.

Bibliography

[1] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas. A system for real-time detection and tracking of vehicles from a single car-mounted camera. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 975–982, Sept 2012. doi:10.1109/ITSC.2012.6338748.

[2] A. Discant, A. Rogozan, C. Rusu, and A. Bensrhair. Sensors for obstacle detection - a survey. In *Electronics Technology, 30th International Spring Seminar on*, pages 100–105, May 2007. doi:10.1109/ISSE.2007.4432828.

[3] M.M. Trivedi, T. Gandhi, and J. McCall. Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety. *Intelligent Transportation Systems, IEEE Transactions on*, 8(1):108–120, March 2007. doi:10.1109/TITS.2006.889442.

[4] U. Nunes, M.M. Trivedi, and C. Laugier. Introducing perception, planning, and navigation for intelligent vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, 10(3):375–379, September 2007.

[5] T. Dang, J. Desens, U. Franke, D. Gavrila, L. Schafers, and W. Ziegler. Steering and evasion assist. *Springer-Verlag*, 2012.

[6] A. Bartels, M. Meinecke, and S. Steinmeyer. Lane change assistance. *Springer-Verlag*, 2012.

[7] A. Talukder, R. Manduchi, A. Rankin, and L. Matthies. Fast and reliable obstacle detection and segmentation for cross-country navigation. In *Intelligent*

Vehicle Symposium, 2002. IEEE, volume 2, pages 610–618 vol.2, June 2002. doi:10.1109/IVS.2002.1188019.

[8] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 18(12):1696–1710, Dec 2006. doi:10.1109/TKDE.2006.183.

[9] S. Matzka and R. Altendorfer. A comparison of track-to-track fusion algorithms for automotive sensor fusion. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 189–194, Aug 2008. doi:10.1109/MFI.2008.4648063.

[10] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, April 2012. doi:10.1109/TPAMI.2011.155.

[11] D. Geronimo, A.M. Lopez, A.D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1239–1258, July 2010. doi:10.1109/TPAMI.2009.122.

[12] A. Prioletti, P. Grisleri, M.M. Trivedi, and A. Broggi. Design and implementation of a high performance pedestrian detection. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1398–1403, June 2013. doi:10.1109/IVS.2013.6629662.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005. doi:10.1109/CVPR.2005.177.

[14] S.J. Krotosky and M.M. Trivedi. On color-, infrared-, and multimodal-stereo approaches to pedestrian detection. *Intelligent Transportation Systems, IEEE*

Transactions on, 8(4):619–629, Dec 2007. doi:10.1109/TITS.2007.908722.

[15] A. Broggi, P. Cerri, S. Ghidoni, P. Grisleri, and Ho Gi Jung. A new approach to urban pedestrian detection for automatic braking. *Intelligent Transportation Systems, IEEE Transactions on*, 10(4):594–605, Dec 2009. doi:10.1109/TITS.2009.2032770.

[16] P. Geismann and G. Schneider. A two-staged approach to vision-based pedestrian recognition using haar and hog features. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 554–559, June 2008. doi:10.1109/IVS.2008.4621148.

[17] T. Gandhi and M.M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. *Intelligent Transportation Systems, IEEE Transactions on*, 8(3):413–430, Sept 2007. doi:10.1109/TITS.2007.903444.

[18] S. Sivaraman and M.M. Trivedi. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *Intelligent Transportation Systems, IEEE Transactions on*, 14(4):1773–1795, Dec 2013. doi:10.1109/TITS.2013.2266661.

[19] Yuqiang Liu, Bin Tian, Songhang Chen, Fenghua Zhu, and Kunfeng Wang. A survey of vision-based vehicle detection and tracking techniques in its. In *Vehicular Electronics and Safety (ICVES), 2013 IEEE International Conference on*, pages 72–77, July 2013. doi:10.1109/ICVES.2013.6619606.

[20] U. Scheunert, H. Cramer, B. Fardi, and G. Wanielik. Multi sensor based tracking of pedestrians: a survey of suitable movement models. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 774–778, June 2004. doi:10.1109/IVS.2004.1336482.

[21] Young-Chul Lim, Chung-Hee Lee, Soon Kwon, and Jong-Hun Lee. Position estimation and multiple obstacles tracking method based on stereo vision sys-

tem. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 72–77, June 2009. doi:10.1109/IVS.2009.5164255.

[22] Zehang Sun, G. Bebis, and R. Miller. On-road vehicle detection: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):694–711, May 2006. doi:10.1109/TPAMI.2006.104.

[23] M. Enzweiler and D.M. Gavrila. Monocular pedestrian detection: Survey and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2179–2195, Dec 2009. doi:10.1109/TPAMI.2008.260.

[24] Ying Wu and Ting Yu. A field model for human detection and tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):753–765, May 2006. doi:10.1109/TPAMI.2006.87.

[25] M. Enzweiler and D.M. Gavrila. A mixed generative-discriminative framework for pedestrian classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. doi:10.1109/CVPR.2008.4587592.

[26] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 4, pages 35–39 vol.4, 1999. doi:10.1109/ICIP.1999.819462.

[27] P. Viola and M. Jones. Robust real-time face detection. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 747–747, 2001. doi:10.1109/ICCV.2001.937709.

[28] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(4):349–361, Apr 2001. doi:10.1109/34.917571.

[29] C. Mikolajczyk and A Zisserman. Human detection based on probabilistic assembly of robust part detectors. *ECCV*, pages 69–82, 2004.

[30] P. Sabzmeydani and G. Mori. Detecting pedestrians by learning shapelet features. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007. doi:10.1109/CVPR.2007.383134.

[31] N. Dalal and C. Schmid. Human detection using oriented histograms of flow and appearance. In *European Conf. Computer Vision (ECCV), 2006. ECCV. IEEE Conference on*, pages 428–441, 2006.

[32] A.K. Jain, R.P.W. Duin, and Jianchang Mao. Statistical pattern recognition: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, Jan 2000. doi:10.1109/34.824819.

[33] V. Cherkassky. The nature of statistical learning theory . *Neural Networks, IEEE Transactions on*, 8(6):1564–1564, Nov 1997. doi:10.1109/TNN.1997.641482.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015. URL: http://dx.doi.org/10.1007/s11263-015-0816-y, doi:10.1007/s11263-015-0816-y.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[37] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu,

Royce Cheng-Yue, Fernando Mujica, Adam Coates, and Andrew Y. Ng. An empirical evaluation of deep learning on highway driving. *CoRR*, abs/1504.01716, 2015. URL: http://dblp.uni-trier.de/db/journals/corr/corr1504.html#HuvalWTKSPARMCM15.

[38] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann Lecun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *http://arxiv.org/abs/1312.6229*.

[39] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. 2013. URL: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf, doi:10.1007/s11263-013-0620-5.

[40] S. Sivaraman and M.M. Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *Intelligent Transportation Systems, IEEE Transactions on*, 11(2):267–276, June 2010. doi:10.1109/TITS.2010.2040177.

[41] A. Haselhoff and A. Kummert. An evolutionary optimized vehicle tracker in collaboration with a detection system. In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–6, Oct 2009. doi:10.1109/ITSC.2009.5309835.

[42] Wei Liu, Xuezhi Wen, Bobo Duan, Huai Yuan, and Nan Wang. Rear vehicle detection and tracking for lane change assist. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 252–257, June 2007. doi:10.1109/IVS.2007.4290123.

[43] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), December 2006. URL: http://doi.acm.org/10.1145/1177352.1177355, doi:10.1145/1177352.1177355.

[44] Ying Zhu, D. Comaniciu, V. Ramesh, M. Pellkofer, and T. Koehler. An integrated framework of vision-based vehicle detection with knowledge fusion. In

Intelligent Vehicles Symposium, 2005. Proceedings. IEEE, pages 199–204, June 2005. doi:10.1109/IVS.2005.1505102.

[45] Yi-Ming Chan, Shih-Shinh Huang, Li-Chen Fu, and Pei-Yung Hsiao. Vehicle detection under various lighting conditions by incorporating particle filter. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 534–539, Sept 2007. doi:10.1109/ITSC.2007.4357745.

[46] Wen-Chung Chang and Chih-Wei Cho. Real-time side vehicle tracking using parts-based boosting. In *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, pages 3370–3375, Oct 2008. doi:10.1109/ICSMC.2008.4811818.

[47] J. Arrospide, L. Salgado, M. Nieto, and F. Jaureguizar. On-board robust vehicle detection and tracking using adaptive quality evaluation. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 2008–2011, Oct 2008. doi:10.1109/ICIP.2008.4712178.

[48] B. Aytekin and E. Altug. Increasing driving safety with a multiple vehicle detection and tracking system using ongoing vehicle shadow information. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 3650–3656, Oct 2010. doi:10.1109/ICSMC.2010.5641879.

[49] C. Idler, R. Schweiger, D. Paulus, M. Mahlisch, and W. Ritter. Realtime vision based multi-target-tracking with particle filters in automotive applications. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 188–193, 2006. doi:10.1109/IVS.2006.1689626.

[50] A. Takeuchi, S. Mita, and D. McAllester. On-road vehicle tracking using deformable object model and particle filter with integrated likelihoods. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 1014–1021, June 2010. doi:10.1109/IVS.2010.5548067.

[51] H.T. Niknejad, A. Takeuchi, S. Mita, and D. McAllester. On-road multivehicle tracking using deformable object model and particle filter with improved like-

lihood estimation. *Intelligent Transportation Systems, IEEE Transactions on*, 13(2):748–758, June 2012. doi:10.1109/TITS.2012.2187894.

[52] Xue Mei and Haibin Ling. Robust visual tracking and vehicle classification via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2259–2272, Nov 2011. doi:10.1109/TPAMI.2011.66.

[53] Jianzhu Cui, Fuqiang Liu, Zhipeng Li, and Zhen Jia. Vehicle localisation using a single camera. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 871–876, June 2010. doi:10.1109/IVS.2010.5548101.

[54] C. Hilario, J.M. Collado, J.M. Armingol, and A. de la Escalera. Visual perception and tracking of vehicles for driver assistance systems. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 94–99, 2006. doi:10.1109/IVS.2006.1689611.

[55] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. URL: http://doi.acm.org/10.1145/358669.358692, doi:10.1145/358669.358692.

[56] J. Nuevo, I. Parra, J. Sjoberg, and L.M. Bergasa. Estimating surrounding vehicles' pose using computer vision. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1863–1868, Sept 2010. doi:10.1109/ITSC.2010.5625000.

[57] C. Hoffmann. Fusing multiple 2d visual features for vehicle detection. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 406–411, 2006. doi:10.1109/IVS.2006.1689662.

[58] K. Yamaguchi, T. Kato, and Y. Ninomiya. Vehicle ego-motion estimation and moving object detection using a monocular camera. In *Pattern Recognition,*

2006. ICPR 2006. 18th International Conference on, volume 4, pages 610–613, 2006. doi:10.1109/ICPR.2006.1165.

[59] K. Yamaguchi, T. Kato, and Y. Ninomiya. Moving obstacle detection using monocular vision. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 288–293, 2006. doi:10.1109/IVS.2006.1689643.

[60] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, July 2012. URL: http://dx.doi.org/10.1109/TPAMI.2011.239, doi:10.1109/TPAMI.2011.239.

[61] Liyang Yu, Chunxiao Fan, and Yue Ming. A visual tracker based on improved kernel correlation filter. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, ICIMCS '15, pages 60:1–60:4, New York, NY, USA, 2015. ACM. URL: http://doi.acm.org/10.1145/2808492.2808552, doi:10.1145/2808492.2808552.

[62] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, pages 2544–2550. IEEE Computer Society, 2010. URL: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#BolmeBDL10.

[63] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip Torr. Fully-convolutional siamese networks for object tracking. *arXiv preprint arXiv:1606.09549*, 2016.

[64] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference Computer Vision (ECCV)*, 2016.

[65] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *PAMI*, 2014.

[66] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, 2009.

[67] P. Dollár, Z. Tu, H. Tao, and S. Belongie. Feature mining for image classification. In *CVPR*, June 2007.

[68] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of online learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997. URL: http://dx.doi.org/10.1006/jcss.1997.1504, doi:10.1006/jcss.1997.1504.

[69] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. volume 28, page 2000, 1998.

[70] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. 1997.

[71] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. pages 511–518, 2001.

[72] Paul Viola, Michael J. Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *Int. J. Comput. Vision*, 63(2):153–161, July 2005. URL: http://dx.doi.org/10.1007/s11263-005-6644-8, doi:10.1007/s11263-005-6644-8.

[73] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 236–243 vol. 2, June 2005. doi:10.1109/CVPR.2005.310.

[74] Rong Xiao, Long Zhu, and Hong-Jiang Zhang. Boosting chain learning for object detection. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 709–, Washington, DC, USA, 2003. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=946247.946591.

[75] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *2005 IEEE Computer Society Conference on Computer Vision and*

Pattern Recognition (CVPR'05), volume 2, pages 150–156 vol. 2, June 2005. doi:10.1109/CVPR.2005.373.

[76] Cha Zhang and Paul A. Viola. Multiple-instance pruning for learning efficient cascade detectors. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1681–1688. Curran Associates, Inc., 2008. URL: http://papers.nips.cc/paper/3265-multiple-instance-pruning-for-learning-efficient-cascade-detecto pdf.

[77] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference in Machine Learning (ICML)*, 2014.

[78] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. volume abs/1502.03167, 2015. URL: http://arxiv.org/abs/1502.03167.

[79] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL: http://arxiv.org/abs/1409.1556.

[80] Alberto Broggi, Elena Cardarelli, Stefano Cattani, Paolo Medici, and Mario Sabbatelli. Vehicle detection for autonomous parking using a Soft-Cascade AdaBoost classifier. In *Procs. IEEE Intelligent Vehicles Symposium 2014*, pages 912–917, Dearbon, MI, USA, June 2014. arXiv:doi:10.1109/IVS.2014.6856490.

[81] Wolfgang Schulz, Markus Enzweiler, and Tobias Ehlgen. Pedestrian recognition from a moving catadioptric camera. In FredA. Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 456–465. Springer Berlin Heidelberg,

2007. URL: http://dx.doi.org/10.1007/978-3-540-74936-3_46, doi:10.1007/978-3-540-74936-3_46.

[82] Lionel Heng, Bo Li, and Marc Pollefeys. Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[83] F. Oniga and S. Nedevschi. Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection. volume 59, pages 1172–1182, March 2010. doi:10.1109/TVT.2009.2039718.

[84] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968, June 2011. doi:10.1109/IVS.2011.5940405.

[85] Alberto Broggi, Stefano Debattisti, Matteo Panciroli, Paolo Grisleri, Elena Cardarelli, Michele Buzzoni, and Pietro Versari. High performance multi-track recording system for automotive applications. *Intl. Journal of Automotive Technology*, 13(1), January 2011.

Ringraziamenti

My thanks go to my family and all the people who helped me in studies. I want to thank also the people of the Vislab and the professors Alberto Broggi and Massimo Bertozzi.