Unsupervised Learning of Object Categories from Video

Vorgelegt von
Jonas F. Luft, B.Sc.

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Master of Science
- M.Sc. -
genehmigte Abschlussarbeit.

Gutachter :	Prof. Dr.-Ing. Olaf Hellwich
	Prof. Dr. Manfred Opper
Betreuer :	Dr.-Ing. Ronny Hänsch

Eidesstattliche Versicherung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 27. August 2018 .....

Jonas Luft

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Subject . . . 2
1.3 Objective . . . 3
1.4 Approach . . . 4
1.5 Used methods . . . 5
1.6 Related work . . . 6
1.7 Outline . . . 7

2 Methods **9**
2.1 General . . . 9
2.2 Motion segmentation . . . 10
2.2.1 Long-term trajectories . . . 11
2.2.2 Clustering . . . 13
2.3 Object categorisation and detection . . . 16
2.3.1 Shape representation . . . 17
2.3.2 Detection . . . 21
2.4 Category learning . . . 22
2.4.1 Defining categories . . . 22
2.4.2 Processing chain . . . 23
2.4.3 Similarity . . . 25
2.4.4 Merging . . . 27
2.4.5 Problems and pitfalls . . . 27

3 Evaluation **31**
3.1 Experimental set-up . . . 31
3.1.1 Testing targets . . . 32
3.1.2 Datasets . . . 33
3.2 Experiments . . . 34
3.2.1 Appraisal of tracking method . . . 34
3.2.2 Appraisal of detection method . . . 35
3.2.3 Detailed discussion of results . . . 37

4 Conclusion **47**
4.1 Discussion . . . 47
4.2 Future work . . . 48

A Experimental Data **51**

Bibliography **55**

Chapter 1

Introduction

Contents

1.1 Motivation 1
1.2 Subject 2
1.3 Objective 3
1.4 Approach 4
1.5 Used methods 5
1.6 Related work 6
1.7 Outline 7

1.1 Motivation

With increasing automation of our everyday life, computer vision becomes paramount to technical solutions of many tasks which humans can solve intuitively. While specialised techniques already solve reduced problems like optical character recognition nearly without room for improvement, more general systems, capable of handling many output classes are needed for powerful artificial intelligence systems, mimicking the ability of humans to adapt to and process diverse inputs. A basis for many tasks of digital systems is finding relevant objects in an unstructured environment. One can imagine the great diversity of situations e.g. an autonomously driving vehicle needs to be able to make sense of for safe operation. Most notably, in every given moment the system has to decide which objects in view of it's sensors might pose danger or require any reaction. While specialised laser range finders easily enable manoeuvring around obstacles, having definite information about what these objects close to us are enables predicting their behaviour and thus allows the vehicle to plan it's movements ahead for some period of time. For example, detecting a pedestrian in front of our vehicle almost always requires an immediate stop of the vehicle, while we can expect a car in front of us to move itself, allowing to continue driving while keeping a safe distance.

The problem with any machine learning task, including modern computer vision, is the requirement of good training data. By extracting structural information spanning all of the training data, the computer vision algorithm learns to find and exploit relevant information in images. When the trained algorithm is then confronted with images it has not seen before, it can use the prior information gained during training,

to quickly analyse the given image and make probabilistic predictions e.g. about it's content. It becomes apparent, the training data has to encompass so many possible situations, so the algorithm can build a model general enough to process anything it might encounter in the wild. As a consequence, careful and exhaustive selection of training images is required beforehand.

Furthermore, most machine learning methods require input data which is annotated to some extent. This might be a mark highlighting e.g. the object of interest or just a brief description of the contents of the image. Only then can the algorithm extrapolate image data to the output it is requested to provide. This method is called *supervised learning* and provides the most precise results. The downside is the need for annotating or labelling each image of the possibly very large training data set. Because this can only reliably achieved by humans, this results in a great cost of manual labour.

Another approach is using *unsupervised learning* methods. In this category are all methods, which don't use annotations in the training data set. Since the algorithm is not supplied with class labels it has to assign to objects, it will split them into more or less consistent groups, depending on a learned underlying structure. Because we don't have these labels to guide us, generally a lot more training images are needed to autonomously find the common structure among them. While this eliminates the need for human labour for annotating the training set, a beneficial selection of training images remains. To further reduce the time and care needed during preparation of datasets, we need to find a way to extract more information out of our input data, without putting more human labour in.

1.2 Subject

The special task we are interested in is called *object classification* or *object categorisation*, i.e. determining the *category* or *class* of an object seen by a digital camera. A system can be trained to classify objects of only a single class (for example decide whether a given image contains a cat) up to being able to distinguish hundreds of different classes (compare the ImageNet recognition challenge [Russakovsky *et al.* 2015], with over 10,000 depicted object categories).

In real-world scenarios, because we have to find the relevant objects in cluttered scenes, classification usually requires *object detection* first. This subject concerns localising objects, since there might be many more relevant objects in a single image, then called *scene*. Their position is relevant not only to treat them as separate points of interest, but since it is not known beforehand, finding the position is necessary to know where to look for identifying features. Spatial occurrence (e.g. planes being in the sky) and relation between objects might even give contextual information of the scene and thus enable more reliable identification of objects and object parts. Localisation is conventionally done by firstly finding salient image regions, where objects are to be expected. For better results in more recent approaches, detection

Figure 1.1: An example for a complex localisation task. What objects are in the scene and where? Left: shot from video, right: ground-truth semantic segmentation. Taken from [Brostow *et al.* 2009].

is done simultaneously with the classification by looking for and matching previously learned object-like features in the image.

Generally, the principle of these combined methods is as follows: A definition of objects has to be found and modelled, which in turn allows to store identifying and differentiating information about different object classes. This object model is then separately trained for every class from a sufficient amount of training data to result in discernible representations. For complicated objects, especially with high intra-class variance, and inconsistent datasets (e.g. changing lighting conditions and viewpoints) simple texture matching is insufficient to find object features. Therefore a more sophisticated representation of local appearance is employed. This usually requires higher level features for expressiveness and efficiency. These high dimensional features store e.g. local gradients and orientation, even scale, and are only extracted from image areas with a lot of information. The desired result is a class label and position for every object that has been found in the scene, provided by bounding boxes or even pixel-wise figure-ground segmentation. The latter can be seen in figure 1.1, with the image to the right coloured according to the class labels of present objects.

1.3 Objective

Detecting objects in images and classifying them into distinct categories has made great progress in the recent years and works increasingly well even for large sets of categories. Unfortunately, this performance can only be obtained by providing the algorithms with a sufficiently large amount of hand-labelled training data. This is especially problematic, when objects should be detectable from different viewpoints, which in turn requires even more additional training data showing each object from various angles.

Some popular data sets were created by sourcing data from the internet with queries in image search engines. This at least provides a weakly labelled dataset, using the search query as a class label for the image. While this approach quickly provides a wide variety of class exemplars of all styles and configurations, it also introduces a great amount of labelling noise (wrongly labelled specimen) into the dataset.

We argue that using video data firstly is a quick way to obtain ample training material because it will show objects of interest for multiple frames. Secondly, videos contain more information, which we can automatically exploit to enhance the quality of our detection algorithms in a way that would otherwise have needed more or better annotations. Thirdly, motion can resolve some ambiguities in deciding wether certain information is relevant. Consider again figure 1.1: is it important to distinguish houses from trees in the background? Just as human attention immediately snaps to sudden movements, motion might give clues about the importance of certain objects.

Today video data can be sourced in large quantities because of the abundance of footage from handheld and surveillance cameras on platforms like Youtube. With this data, we can train a detector on any kind of rigid or even articulated objects, as long as they can be observed during motion. This helps a wide variety of CV applications where creating training data by hand for many categories is not feasible or not economical, such as e.g. autonomous navigation in unknown environments.

1.4 Approach

To reduce the amount of human labour needed, we enable the use of unannotated video files to train an object detector. When videos are not chosen at random but by selecting only videos containing certain object classes, the dataset is considered weakly labelled. An interesting way to increase detection quality, is to both use (during training) and provide (during detection) a segmentation of the scene. Pure bottom-up segmentation and detection must be considered ambiguous when using only static cues from single images. On the other hand, videos allow us to observe coherently moving regions. Our processing chain will handle this by implementing long-term tracking (*large displacement optical flow, LDOF*, [Brox & Malik 2011]) of image features. We can subsequently infer which pixels of multiple frames depict a common object. This provides us with a sufficing number of observations with which we will build an *implicit shape model* (*ISM*, [Leibe & Schiele 2003]) codebook, which enables detection of said object in unknown images.

The main challenge of this approach is learning the extent of an unknown category from it's observed exemplars. In this weakly or even unsupervised approach, we don't provide any prior knowledge of class separation. Clues about class-bleonging are entirely captured from local shape, texture and motion information. The general idea is, that objects can be considered of being in the same class, if they share enough similarity. According to this assumption, intra-class variance is captured

by an intentionally not robust merging of shape codebooks. A great difficulty lies in discriminative learning not being possible, since it is not known during training which objects belong to the same category—not even the background can be reliably determined.

1.5 Used methods

Human perception, where generalisation from a very reduced information set is apparent, lead Koffka to the development of gestalt psychology (English publication in [Koffka 1935]), to explain how even weak stimuli can carry a lot of information. Most relevant is the so called *principle of common fate* for inferring shared identity from motion. The observation of coherence of motion over multiple points results in the perception of a single object.

Brox and Malik base a tracker on this principle in [Brox & Malik 2011], to find trajectories which are then used for segmentation in videos ([Brox & Malik 2010]). This can be extended to a per-pixel segmentation ([Brox *et al.* 2014]) of any moving object in the video. We will use this method to generate segmentations from input videos, which gives us the training data needed for building a bag-of-words model of objects.

The basis for our modelling of objects and object classes is the conception of a generative model of local appearance. In general, discriminative models are considered to be achieving a higher performance, whereas discriminative models can reach their best performance with a smaller training set, as is thoroughly discussed and tested in [Ng & Jordan 2001]. Perhaps more important for us is the ability of generative models to better cope with missing data by extrapolation—provided the learned joint distribution of features and class labels is adequate. A more practical advantage is the ability of generative models to create e.g. new class exemplars. We use this for estimating a figure-ground-segmentation of the objects we detect. To exploit the possibilities of local appearance, small-scale texture, gradient and spectral information are put into spatial context by relative spatial location, leading to a description of shape. Local appearance models allow detection of partly occluded objects and capturing high intra-class variance without losing precision, because missing features don't impact the posterior probability. Also, we can consider it the requirement for pure bottom-up approaches, since we can start learning from local features before defining a complete object model.

Leibe and Schiele developed the Implicit Shape Model for simultaneous object categorisation and segmentation [Leibe & Schiele 2003]. It is trained with known instances and segmentations of objects and builds a detector codebook from clustered features. This codebook will then be used for a probabilistic Hough voting procedure. The object hypotheses are refined via Mean-Shift Search and a segmentation of the found object can be determined via backprojection from the activated features in the codebook. This procedure is robust to varying poses and even reliably categorises articulated objects. In the later iterations, Leibe et al. introduce scale-

invariance and successfully resolve ambiguities in conflicting detection hypotheses [Leibe *et al.* 2004]. Their last paper [Leibe *et al.* 2008] thoroughly compares various datasets, region-of-interest detectors (e.g. difference of Gaussians, Hessian, Harris corners) and feature descriptors (e.g. SIFT, patches). We use this as a framework for training object models and running detections.

1.6 Related work

In recent years, object recognition technologies have experienced a dramatic increase in precision. A stepping-stone was the introduction of deep neural networks to image recognition in one way or another. One example of a notable result can be found in [Krizhevsky *et al.* 2012] where deep convolutional neural networks were used for discerning hundreds of different classes. While this method has shown to provide astonishing results, it needed to be trained in a supervised manner on around 15 million images.

By employing part-based models the required amount of training images can be reduced through the incorporation of more prior knowledge of the problem domain. In [Endres *et al.* 2013] the training set is in the range of thousands of objects, without sacrificing detection accuracy. For comparison, in [Felzenszwalb *et al.* 2010] a similar model is used, but it uses iterations of unsupervised learning evaluated with partially labelled data and provides slightly worse detections.

Purely unsupervised methods have to employ some intermediary step to resolve ambiguities. One common approach is using the saliency of the image to locate regions of interest. An example of this method is described in [Zhu *et al.* 2012] and shows mediocre (compared to modern methods) detection and localisation results, e.g. 31% correct localisation (*CorLoc*) in the PASCAL07 dataset [Everingham *et al.*].

Working on videos requires different approaches, because the datasets are often smaller in the number of depicted objects, but contain more contextual information that has to be leveraged. Most closely related to our work is the task of colocalisation. In this problem the goal is to simultaneously localise objects of a common class in a set of input images or videos. While this cannot really be called object classification and no class specific detectors are trained, it can be used to the same effect when providing a sample image instead of a class label. This might indeed make a fine continuation to our proposed method, since as in any unsupervised approach there can be no output of class labels making sense to a human operator.

A prime example is found in [Cho *et al.* 2015] where bottom-up region proposals are formulated in each image, followed by foreground localisation through contrast. With nearest neighbour retrieval of said foreground regions in other images, other class exemplars can be found. Even though they are not exploiting motion information, they reach average 53% CorLoc in a cluttered mixed-class setting.

Related is [Kwak *et al.* 2015], starting with object discovery as well, but then incorporating temporal cues by scoring proposals by their coherence to motion clusters.

Colocalisation follows, training a contextual and temporal neighbourhood network by alternately comparing different videos and frames in a single video. They reach 55.7% average CorLoc on the YoutubeObjects dataset [Prest *et al.* 2012]. With that they outperform Brox and Malik (with 34.8% CorLoc), but provide only bounding boxes instead of precise segmentation masks. It begs the question why objects should be discovered first, if tracking movement alone can define them completely and precisely.

In [Croitoru *et al.* 2017] a method is presented, that uses a convolutional neural network to predict a soft (i.e. probabilistic) segmentation mask. It is trained on every input frame with a principal component analysis method for calculating said masks in a video.

Similar in idea but diametrical in approach to our method is the work by [Wang & Gupta 2015]. They sample patches from videos, track them through every frame and then train a convolutional neural network with the constraint of a loss function ensuring patches of a single track lie close in feature space. While they obtained detection scores close to some supervised methods and seems to capture intra-class variance quite well, their approach has the same drawback as any neural network method: reliance on thousands of videos and thus millions of input samples.

Using a bag-of-words model with local appearance has a great advantage to these state of the art image classification techniques in the greatly reduced number of training images needed.

1.7 Outline

In the following chapter the used algorithms and methods will be presented and their principles described in detail. Our core contribution of a unsupervised strictly bottom-up learning of object categories will be elaborated on in section 2.4.

Chapter 3 introduces our evaluation procedure and the datasets we employ for testing our pipeline. We will provide a detailed analysis of the encountered problems and discuss possible underlying causes. After extensive qualitative assessment, a short overview of our quantitative results follows.

We close by discussing the principal capabilities of our method and suggest further needed refinements and advancements to fully utilise the proposed concept.

Chapter 2

Methods

Contents

2.1 General **9**
2.2 Motion segmentation **10**
2.2.1 Long-term trajectories 11
2.2.2 Clustering 13
2.3 Object categorisation and detection **16**
2.3.1 Shape representation 17
2.3.2 Detection 21
2.4 Category learning **22**
2.4.1 Defining categories 22
2.4.2 Processing chain 23
2.4.3 Similarity 25
2.4.4 Merging 27
2.4.5 Problems and pitfalls 27

2.1 General

Objects in a video sequence move in a consistent way. Feature points belonging to a common object will more or less retain their properties from one frame to the next. Generally speaking, we observe a short-term time invariance within image regions that correspond to a single moving object. We can therefore assume firstly, that features we have found in one frame will be detectable in the next few following frames and secondly, that multiple features belonging to a common object will generally move with the same velocity in the same direction. When considering short-term changes in features and combining them to long-term trajectories, we can track an object through a video sequence, even when the spectral information or the view undergoes considerable variation over time. The first task of using this spatio-temporal information for object detection is extracting moving objects from the sequence, resulting in a segmentation of the scene into all visibly distinct objects for every frame.

Having established which objects are present including their corresponding figure area, we train a separate detector for each. This is done by extracting the image regions for every frame that the tracker identified as belonging to the same object and

Figure 2.1: Exemplary frame from motion segmentation of Hopkins dataset [Tron & Vidal 2007]. Object instances colour-coded.

treat these spatio-temporal tubes as specific single-class training sets. By learning object models with bags-of-words of local appearances, we aim to achieve a representation, that allows establishing similarity between some of these objects, if they would be labelled as belonging to a common category by a human observer. Objects having been found to be similar, are then treated as a single new objects, represented by a newly trained codebook. When no more sufficing similarity is observed, we are left with a set of codebooks disjunctly representing the objects contained in the dataset and allowing to detect similar instances in the wild.

2.2 Motion segmentation

Motion segmentation denotes the process of using motion information to find the area in an image that encompasses an object. When there are multiple objects in the scene which move independently, they can be detected simultaneously, separating them in the process. When calculating segmentation in still images strictly bottom-up, the resulting segments will often not coincide with sensible object boundaries. Providing a decent semantic segmentation usually involves top-down schemes,

e.g. by learning the shape of objects through training and then applying this prior knowledge after detecting those objects. In contrast motion segmentation allows a pure bottom-up approach for at least a partial semantic segmentation by utilising spatio-temporal cues instead of learned context.

Motion segmentation is a two-stage problem: Firstly, discriminative features have to be found in every frame and matched to features in consecutive frames, thus forming trajectories. And secondly, clustering these trajectories into distinct, but consistent groups, conforming to objects.

For use with unsupervised object detection we have certain requirements for the segmentation algorithms to fulfil: First, since the segmentation is our starting point, it may not require any prior information, but work solely with the frames' colour data and temporal order in the video. Second, we require the discovery of any object, even if it is partially occluded or even completely invisible for some frames (which is still difficult for many modern methods). Furthermore we would like a closed figure area with a reasonably large extent, so we can be certain where to extract descriptors of local appearance for the tracked objects.

On a final note, when choosing a trade off, we could rather accept a reduced precision in the object boundaries (resulting in labelling noise) than obtain an over-segmented result, which would lead to problems described in section 2.4.5.

We apply the method presented in [Brox *et al.* 2014] and use their provided binaries[1] for segmenting video data. The result can be seen in figure 2.1, which shows a single frame of a 20 frame video sequence, that was segmented purely from bottom-up cues. In this sequence two cars were tracked and segmented (shown in blue and yellow), whereas all other objects were not moving and thus were inseparable from the background, depicted in red.

2.2.1 Long-term trajectories

The first step in a motion segmentation task is detecting the movement itself. Under the assumption that the change in the scene within two neighbouring frames or even within a certain time interval is not too big, we can apply *optical flow* methods to determine where an object (or rather at this stage, a point belonging to a possible object) has moved to in the subsequent frame. Even though the background might exhibit motion as well because of a moving camera, objects are separable because of an observed discontinuity in the local motion velocity or direction.

The goal is to show where certain points in a frame moved to in the next frame. Given two images of a consecutive sequence $I_a(\vec{x}), I_b(\vec{x}) \in \mathbb{R}^2$ this can be formulated as finding horizontal and vertical displacements $\vec{w}(\vec{x})$ such that

$$I_a(\vec{x}) = I_b(\vec{x} + \vec{w}(\vec{x})) \tag{2.1}$$

[1] Obtained from `https://lmb.informatik.uni-freiburg.de/resources/binaries/pami2013_MoSeg.tar.gz`, February 2018

Figure 2.2: Feature points found in multiple frames are connected by trajectories.

Which is solved by minimizing some total energy function of each pair of frames:

$$E = \int_{\vec{x}} f(|I_b(\vec{x} + \vec{w}(\vec{x})) - I_a(\vec{x})|)\, dx \tag{2.2}$$

with $f(s)$ being some distance norm, in our case a long tailed distribution $f(s) = \sqrt{s^2 + 0.001^2}$, to allow some, but strongly penalise bigger colour deviations, e.g. at occlusion boundaries [Brox & Malik 2011]. Solving this for every pixel in the pair of frames results in a vector field describing the momentary movement speed and direction of every point in the image. The result of these methods is a continuous *dense* flow field. Predominantly, variational methods are used to compute dense optical flow. They work by locally matching grey-values or gradients between neighbouring frames. Because the solution to this matching as sometimes ambiguous (e.g. on contours with constant brightness), some further constraint has to be introduced. This can be done by limiting the difference of velocity between point and the average velocity in the surrounding area [Horn & Schunck 1981] as an addition to the energy term (2.2). A problem of this assumption is the blurring of discontinuities in the flow field. These lines of abrupt velocity change are especially important for us, since they occur at object boundaries. To avoid both of these shortcomings, smoothness can instead be enforced by penalising the total variation on the global optical flow field [Brox & Malik 2011]. Unfortunately, this smoothness requirement still uses local optimisations for finding a global solution and will avoid solutions with large displacements. A coarse-to-fine warping scheme, as proposed by [Mémin & Pérez 1998] can help because the solution will no longer be biased on the initialization of the flow field, but is instead estimated by first considering large movements in the image which will then be successively refined. This improvement solves most problems, but can lead to missing large movements of small structures, e.g. human extremities, when they don't coincide with the movement of larger regions.

A robust approach to capture such movement is finding high dimensional descriptive features in every frame and thereupon matching said features of one frame to the features found in another frame. Of course, not every image point will be able to be matched to the next frame—either because the local change was too high in the observed time interval, or because the point is no longer visible. Latter may result from occlusion or just because this point does not carry enough discriminative

information to be found in the next frame. Therefore we only obtain a *sparse optical flow field* (concerning denseness in space) for each frame. Resulting matches become more robust, when we apply forward-backward matching; i.e. when a feature of the first frame matches a feature in the second frame and the feature of the second frame also matches the same feature in the first frame reasonably well. Figure 2.2 shows possible interest points which can be matched over multiple frames even though the scene underwent considerably change.

Using rich descriptors is advantageous, since they are extremely efficient and robust. They can be found again even after large changes and movements between frames. The choice of features impacts spatial localisation. Some features are modelling an image area with certain properties, hence do not describe exact points in the frame. For precise localisation, more complex descriptors, as e.g. SIFT [Lowe 2004] can be used.

Computing flow fields for neighbouring frames provides accurate and dense velocity estimations. The downside of only considering short-term change is that it can only detect local change. It neglects some information which can be obtained by considering large displacements over greater time spans. The problem with descriptor tracking is, that they are prone to occlusion and provide many outliers which are not trivial to remove. For the best results, we would like to have the advantages of both: Through considering the optical flow field for every frame of a sequence, we can fill in the gaps between tracked feature descriptors.

This allows us to track arbitrary points for the entire sequence. We call this list of points connected over time by matching a *trajectory*. Expectably, only few trajectories will span the entire sequence, when not continuously finding corresponding features because of previously mentioned reasons. It is, however, very important to keep incomplete trajectories, as they allow inference of coherent movement between multiple trajectories and later the affinity to a common object.

In [Brox & Malik 2011] Brox and Malik develop what they call *large displacement optical flow*, where they combine both the variational methods and descriptor matching discussed here for providing an arbitrary density of tracking points with sub-pixel accuracy. Through said density of sampling in space (density in time proves not to be as important, because trajectories can be joined later), they show to provide a tracker that is extremely robust and allows pixel-perfect motion segmentation, while strictly relying on bottom-up clues. We use the methods presented in this work as the starting point in our pipeline.

2.2.2 Clustering

Having established the movement of certain points in the image they have to be grouped according to *common fate*, resulting in areas in the image with coherent movement. To this end a clustering method of the previously illustrated trajectories is developed in [Brox & Malik 2010].

The distance of two point trajectories is the maximum difference of their motion over time. This allows to separate two objects, even when they have shared identical

movement over a long period of time. Consequently, wherever there is a discontinuity in the local flow field, the trajectories belonging to the points on either side will probably end up in different clusters.

Specifically, *spectral clustering* [Shi & Malik 2000] of pairwise affinities is used where trajectories as vertices are connected to a graph by sharing at least one frame. The construction of a full graph allows to connect unrelated or disconnected trajectories automatically by the transitive property. Pairwise affinities of two trajectories A, B can be computed as soon as they share a single frame. Following [Brox & Malik 2010]:

$$d_t^2(A,B) = d_{sp}(A,B)\frac{(u_t^A - u_t^B)^2 + (v_t^A - v_t^B)^2}{5\sigma_t^2} \tag{2.3}$$

where $d_{sp}(A,B)$ is the average Euclidean distance of trajectories A, B; and $u_t = x_{t+5} - x_t, v_t = y_{t+5} - y_t$ describe the 5-frame averaged motion of a point. By consulting the Euclidean distance of two points in a frame, only proximate points are awarded high affinities and distant ones are penalised.

A sensible normalisation enables handling both very little and quite large movements in the same scene with the following equation:

$$\sigma_t = min_{a\in\{A,B\}} \sum_{t'=1}^{5} \sigma(x_{t+t'}^a, y_{t+t'}^a, t+t') \tag{2.4}$$

where $\sigma : \mathbb{R}^3 \to \mathbb{R}$ denotes the local flow field at a certain time. The maximal dissimilarity of two trajectories is then used to calculate an affinity matrix W for clustering with entries

$$w(A,B) = exp(-0.1\, d^2(A,B)) \tag{2.5}$$

The pairwise distances are sufficient only for estimating parameters of a translational motion model. This is an approximation but in most cases acceptable, since a local translation describes long-term scaling and slight rotations (which are sometimes observed with rigid objects and a fixed camera) reasonably well with the right choice of features. More complex motion models would require a higher order of affinities and quickly become computationally intractable.

The tracking and clustering results in a sparse labelling of distinctly moving objects in the scene (see figure 2.3 center). Depending on the density of trajectory points, about 3% of the image area are labelled. Thereafter, we need to refine these labels to per-pixel segmentation masks (consider figure 2.3 right) in each frame to allow training on and with the figure area in the further processing steps. For filling in unlabelled space, colour and texture information are used where motion cues were not reliable. To partition each frame according to the sparse labelling [Brox *et al.* 2014] propose following minimisation:

$$\min_u E(u) = \min_u TV(u) + \sum_{k\in K} \int_\Omega u_k(x) f_k(x)\, dx \tag{2.6}$$

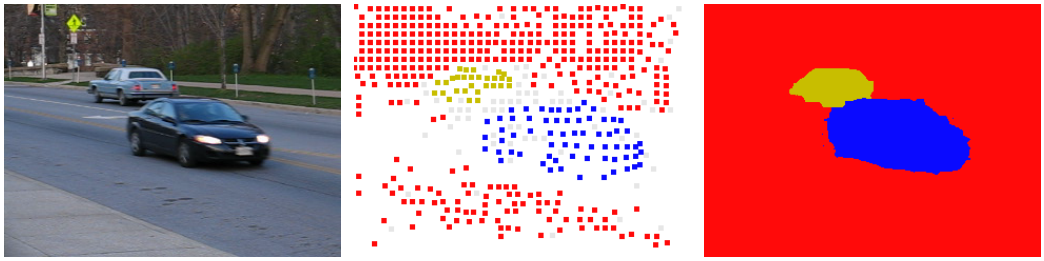

Figure 2.3: Left: image from a video shot, center: clustering of point trajectories indicates similar motion, right: segmentation providing object regions

where u is a label function, projecting the image domain Ω to $[0,1]^K$ and $f_k(x) = 0$ where a label for k is present, otherwise $f_k(x) = \alpha$ with a penalty factor dependant on the confidence of the sparse labelling, i.e. the trajectory spacing.

To provide jumps in the label function where the image gradient $|\nabla I|$ is high, following total variation regularisation is used:

$$TV(u) = \frac{1}{2} \sum_{k \in K} \int_{\Omega} g(x) |\nabla u_k(x)| \, dx \tag{2.7}$$

with $g(x) = 1/\sqrt{|\nabla I|^2 + 0.001^2}$.

In conclusion, Brox et al. show that with sufficient density of tracking points in space, their method works well for both rigid and articulated objects (such as animals and humans). The latter poses additional difficulties, because of relatively fast movement of extremities, which they overcome by using a scale adaptive local normalisation for tracking. Still, articulated objects are regularly over-segmented, especially when they are close to the camera i.e. they show much more structure and movement within the object, than compared to the background. In some cases this over-segmentation might not be considered erroneous, when the object is segmented into reasonable body parts.

Generally, they outperform several state-of-the-art trackers when considering large displacements and (partial) occlusion of objects. Of course, there are typical limitations arising under certain conditions: first, when there is little structure in parts of the scene, the description of flow becomes arbitrary; second, partially transparent objects are often missed entirely; furthermore, sudden changes of appearance are hard to capture; and moving shadows are often attributed to an object.

Since the clustering of trajectories exhibits quadratic time complexity, it might be sensible to cut long sequences at points in time, where there is no movement. This can reduce the number of evaluated tracks at every point in time, since the tracker also has to examine tracks which showed no movement for many frames, since objects could just be temporarily occluded. This would also allow to speed up the process of training detectors, since sequences could be skipped entirely if they contain no figure area.

In our own experiments we encountered additional problems with the results of the provided implementation, on which subsection 3.2.1 will elaborate.

2.3 Object categorisation and detection

Object categorisation is the task of recognising previously unknown instances of various categories and thence assigning a category or class label to these found instances. To that end, a description of class characteristics of object categories has to be established. To allow generalisation to new domains and modelling a wide variety of categories, an approach that learns a general but descriptive model of objects (and object categories) is required. Common approaches extract local image features and build an appearance model for categories by finding the properties multiple instances of a common category in the training data share. This results in a category model which can be used in reverse to find new instances of this category in previously unseen images.

Usually, an a priori figure-ground segmentation is not available, because it is difficult to obtain in a pure bottom-up fashion from image data. Contrary to that preconception we can rely on this additional prior information. The figure-ground-segmentation extracted from motion information allows us to select features only from the figure area of objects and thus require much fewer training examples, since no discrimination between object and background features has to be learned by extrapolating from many examples.

The obtained segmentation also implies that every category in fact describes one singular object in a scene, but is trained with multiple instances showing different views or lighting conditions. This requires to establish a process of determining whether two object categories (in our case, at first, objects) are similar enough to be considered part of a greater common category and therefore warrant merging their models. This problem remains the core of our work and will be described in detail in section 2.4.

When choosing a method for learning and detecting object categories, a few requirements have to be taken into account. To begin with, we need a method that can exploit the information provided by the prior figure-ground-segmentation—otherwise we will not have gained any advantage by using information compiled from motion cues. Even after this additional information we will have to make do with a very limited number of training exemplars. Typically, video sequences in the used data sets consist of about 30–50 frames, not every one showing the object completely. The used method will need to be able to build a model of the object with this limited training data and still be able to generalise to other objects of the same category, to allow merging later. The training data size gets reduced further by partial occlusion and viewpoint changes, when the object rotates in relation to the camera. Changing view of the object in the video sequence on one hand hides some features from view, but on the other hand leads to interesting opportunities to generalise the detector to multiple viewing angles. Therefore, having trained on such objects, makes it much easier to detect other objects when those are only present in a few frames with quickly varying poses. On the downside, many approaches fail to build a consistent model when confronted with training images from different

camera perspectives, especially if they penalise detection scores when some features are not visible.

A drawback of the bottom-up strategy is the inability to use discriminative learning. Since we cannot know beforehand which segmented objects belong to the same category, we cannot learn features of an object, which have been found to lower the occurrence probability of other categories. This extends to the detection task (at least during training), where we cannot maximise for *a posteriori* probability to determine which category is present when confronted with conflicting hypotheses (compare section 2.4.5).

We apply the method presented in [Leibe *et al.* 2008][2] for detecting and localising objects. They show to be providing a robust detector working well on cluttered scenes and even on articulated objects (such as moving animals which show changing shapes and texture). This method works well without relying on discriminative learning. According to [Leibe *et al.* 2008], datasets with objects visible in 30 to 50 frames should be sufficient to obtain good detection results.

Their core concept involves combining the recognition and segmentation of objects as concurrent reciprocally supporting tasks. This use of collaborating information significantly improves detection results, by refining detection hypothesis through resampling in the figure area. The detection method itself is an implementation of a generalised Hough transform using features of the local object appearance.

While their detection framework is not designed for robustness against changing viewpoints, we can show it provides good results, when rotational movement happens in video sequences during training. This is made possible, because all features from different viewpoints are gathered during training and are accumulated in the codebooks. Features from other viewpoints subsequently might be missing during detection but are not lowering the detection score.

2.3.1 Shape representation

We model objects as a bag-of-words. An arbitrary feature detector finds structure-rich points in the object area (red feature points on white area in figure 2.4), around which we extract an descriptor. Different feature/descriptor combinations are evaluated in [Leibe *et al.* 2008] for various detection tasks.

We chose to use *Difference of Gaussians, DoG* [Lowe 2004] as interest points, in conjunction with either 25x25 pixel patches or *shape context* descriptors [Belongie *et al.* 2000] for coding local appearance feature vectors. DoG interest points are extracted by convolving the image with Gaussian kernels of differing standard deviation and don't have an orientation. This pronounces edges and structure-rich points without being sensible to high-frequency noise.

Pixel patches simply represent gray values in their surroundings as a fixed size vector. This allows matching of complex texture information instead of relying solely on gradient information. The downside is sensitivity to environmental changes (e.g.

[2] We use their source code provided at `http://www.mmp.rwth-aachen.de/software/ism` with some modifications to enable automated processing. Obtained February 2018.

Figure 2.4: Left: image from a video shot, right: interest points in the figure area

specularity, light). Scale invariance is ensured by resizing the patches to a fixed size (here 25x25 pixels, thus resulting in a descriptor length of 625) when storing them in a codebook. Patches are compared to each other by normalised correlation.

Local shape context descriptors on the other hand describe shape while disregarding colour information. They are histograms of gradient orientations in their vicinity, described in a log-polar grid. This makes them robust against geometric transformations and lighting changes. Here we use them with 9 location and 4 orientation bins and thus 36 dimensions. These descriptors are compared using Euclidean distances. We intentionally do not use their rotation-invariant implementation, because we consider this additional information important in natural scenes.

Patches might not be the best choice in terms of efficiency, precision or overall quality of detection results (DoG+SIFT performs considerably better with 87% equal-error-rate (EER) versus 78% EER with DoG+patches, depending on dataset [Leibe *et al.* 2008]). But intuitively, they seem to tolerate affine transformations better since they do not represent orientation of features as strictly, as e.g. shape context descriptors, which sample a histogram of gradients at interest points. This is important for objects moving away from and turning in front of the camera, because shearing can be observed following perspective distortion. Therefore, we expect better generalisation to slightly different viewing angles, but not the optimal detection quality. The biggest problem of patches is scale: small (i.e. distant) objects don't get covered properly, while extreme difficulty arises with datasets showing objects of large range of object scales. A single detector with fixed parameter settings will not detect large and small objects equally well.

Use of pixel patches requires gauging how close to the object boundary we want to allow sampling of interest points. It makes sense only sampling patches containing mostly figure area, so we don't accidentally train on background features, which would introduce significant labelling noise. On the downside, descriptive features on object boundaries will not be covered as well.

As a consequence of this minimal area requirement, objects with a very small figure area won't provide many features or might not be modelled at all, if no patches can be extracted. While this results in a loss of many possible detectors, this also helps avoiding training detectors on erroneously tracked objects, occurring with noisy segmentations.

Using these feature descriptors as a set of visual words, yields an appearance model of objects and are combined into a *codebook*. Object-specific features stored in codebook entries are the basis for a two-stage codebook-building process: First, we need to collect and store descriptive features for each object, clustering and averaging them in the process, forming a codebook to both capture intra-class variance and enable efficient lookup. Second, the spatial occurrence distribution (under 2.3.1.2) of those earlier established local features is modelled, allowing localisation of the object during detection.

2.3.1.1 Codebooks

To begin with training, all exemplars of an object are collected from the training set. Determining the frames containing this object is done simply by using all and only those images, which were labelled with said object by the tracker. With the provided figure-ground segmentation from the tracker we can then make sure to only collect feature descriptions from the figure area, to minimise labelling noise. This reduces the number of training examples needed, which would usually be required to compensate for features wrongly extracted from the background. Without this step, highly textured regions in the background can lead to random peaks in the voting space and consequently false positive detections. Having available this prior information provided by figure segmentation masks, makes codebook building computationally feasible on many long video sequences of hundreds of frames, such as the sequences in the VIRAT dataset (refer to section 3.1.2). It allows us to definitely determine any frames not containing the object and hence not considering the great majority of input images during training.

With the aforementioned descriptive visual features we generate a codebook of prototypical local appearances by grouping visually similar features to a common representative. These newly found clusters are represented by their mean. See figure 2.5 for exemplary codebook entries with their center and some patches. It becomes evident that the employed clustering scheme should not result in a minimal number of clusters but instead provide a grouping which results in a visual compactness of cluster members, that is, only group visually similar features. Thus it will not lose the defining structure of the represented visual features. Since the number of needed clusters is not determinable in advance and tends to be in a similar order of magnitude as the number of input points, common clustering methods such as *k-means* fall short. Instead, Leibe et al. implemented an *agglomerative clustering* scheme, utilising *reciprocal nearest neighbour* pairs, which does not require a stored similarity matrix for thousands of points and thus makes the clustering task

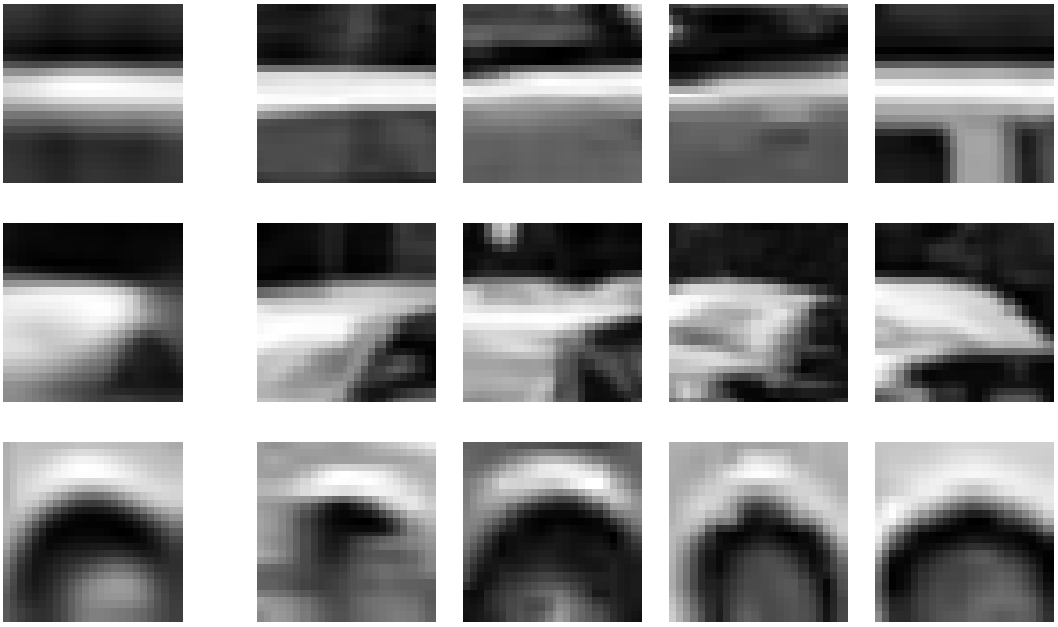

Figure 2.5: Exemplary codebook clusters of shape context detector cars2_06_1-cars5_1-cars2_2-cars7_1-carsbus_1-cars3_2-cars6_1. The first column shows averaged cluster centers, the remaining constituting patches.

extremely efficient. Counting the number of features, that are encompassed by the clusters ensures that features which are found in many exemplars and thus occur repeatedly are later assigned a higher weight during voting.

When sampling features from the figure area only, with segmentation masks extracted from the previous processing steps, we can skip handling of the majority of points found in a frame. During training this significantly reduces computation time, because of the quadratic time complexity of codebook clustering. In the Hopkins155 dataset (refer to section 3.1.2) this reduces the number of extracted features per frame per object to about 50–100 and the number of codebook clusters after processing the whole sequence from around 30,000 to about ten times less.

A sensible clustering of codebook entries proves to be of extreme importance for the later use of the codebooks in detection tasks. Since detection requires a lookup in the codebooks to find the most similar feature (see 2.3.2 below for details), adding more clusters especially when merging object categories (see 2.4.4) leads to immensely prolonged detection times. A safe way to reduce the number of clusters without loosing descriptivity is removing all clusters, that contain only a single feature. In our experiments this reduced the codebook size by about a third of it's clusters, depending on the sequence. For example, the Hopkins cars4_3 object was trained with a 831 cluster codebook, of which 347 only contained one patch. Since features which occur only once over dozens of instances of an object can not be considered descriptive or common for the whole class, removing them from the model makes it more robust. The effect is a more efficient lookup in the cluster,

less atypical votes during detection and in the end reduced noise and false positives during detection.

2.3.1.2 Implicit shape model

The second stage in the codebook-building process is the training of an *implicit shape model (ISM)*. In short, this is a representation of the spatial probability distribution where a certain visual feature can occur on an object. This extends the bag-of-words model to a more descriptive generative model. Specifically, a set of possible displacement vectors from the object centre to the local features is stored in every codebook entry. To this end, all training images are processed a second time to find features, which fall into the previously established clusters of the codebooks by a certain similarity threshold. Then, every found occurrence of this visual word is stored with it's displacement vector at it's cluster mean in the codebook.

Additionally the figure-ground segmentation of the pixels surrounding the feature point is extracted from the training image and stored in the codebook entry as well. An averaged patch of the occurred figure segmentations results in a local map of figure probability around a visual feature, which can help the detection task by determining where it is safe to resample further feature points and also to provide a probabilistic segmentation mask, when detecting the object in a still image. Since we are mainly interested in training on videos in the present work and thus rather use the more reliable segmentation of the tracking in 2.2, this process will not be discussed further.

When using shape context or other descriptors, patches are still being extracted from the training set and stored in codebooks, to visualise the supporting evidence and provide probabilistic segmentation masks in the detection tasks.

2.3.2 Detection

ISM codebooks are used for detection tasks by providing a shape template to a *generalised Hough transform* [Ballard 1981]. Firstly, all relevant descriptors are collected from a test image. Next the extracted descriptors are checked against the codebook, to determine whether they are evidence for a known object. Therefore, we have to iterate through the entire codebook and check the descriptor in question against every codebook entry. When a fixed similarity threshold is reached, we suppose the descriptor is an instance of said codeword. It is important to note, a descriptor can in this way "activate" multiple different codewords and consequently provide evidence for multiple object categories.

Accordingly, we collect all activated codebook entries and let any of those cast votes for their represented object categories. Here the *ISM* comes into play: in every codebook entry, we stored possible occurrence locations for the represented descriptor in relation to the object centre. Conversely, when having matched a codebook entry, it casts multiple votes for possible object centres. The Hough parameter space is three-dimensional along horizontal, vertical and scale axes. Thus

the voting space is filled with object categories, centres and scales, only leaving local maxima to be determined. At every maximum exceeding a certain signal strength, we have established a possible object hypothesis. Finally, Leibe et al. establish multiple refinements to improve the results. Central is a *minimum description length (MDL)*-based criterion to resolve ambiguities between overlapping hypotheses. [Leibe *et al.* 2008] shows a significant improvement of detection results in comparison to other recent (to their time of writing) methods. On the downside, computation time sometimes becomes an issue: clustering codebooks generally takes a long time (though this is only relevant for training) and more importantly, even though features are scale invariant, Hough voting and finding Hough maxima in 3D (position and scale) Hough space quickly becomes a complexity issue, thus biasing the choice of datasets in favour of manageable object scale ranges. We have found enhancing the parameter space to include rotation not to be necessary. Further evaluation can be found in section 3.2.2.

2.4 Category learning

With segmentations and trained detectors in place, the remaining challenge is deciding whether two separate detectors in fact detect the same object (or rather object category). Of course, this question cannot be definitively answered, because a separation of objects into categories is completely subjective and somewhat arbitrary. But for a computer vision algorithm the answer is rather close at hand: two codebooks describe the same object, if two detectors trained on one codebook each detect the same identical objects in any given image or video sequence.

Considering the different training exemplars used for training but also general shortcomings of detection methods, we will not see completely identical detection results but rather have to anticipate a certain degree of uncertainty. Thus a similarity metric is needed, which decides whether the detection results of some detectors are sufficiently similar to reasonably assume a commonly described object category.

After determining which codebooks indeed share a common category, they should be combined into a single codebook from which a new detector is trained. This newly found detector will be less specific than it's constituents and thus allows a greater generalisation and possibly detection of further objects belonging to the same *category*. As a side effect, detection time and space needed for the codebook will typically improve after an iteration of optimisation during codebook-building.

2.4.1 Defining categories

Merging of a lot of codebooks again raises the question of where to draw the line between categories. As growing codebooks become more general, they will detect a higher number of specimen but usually at risk of an increasing uncertainty, because every merging step risks incorporating flawed training data, such as labelling noise or ambiguous descriptors.

Figure 2.6: Cut-outs of segmentations of exemplary frames from sequences cars2_2 (left) and cars8_1 (right).

Ultimately, the answer to this question boils down to the used dataset. Consider figure 2.6, where we can see two segmented object instances from the Hopkins dataset. Are these two objects constituents of a common object class? Classes can be considered hierarchical, as e.g. the ImageNet [Russakovsky *et al.* 2015] database is structured. Accordingly, both objects here can be considered "vehicles" as well as "cars", leading to a commonly used detector. We could go further down the category tree and consider them members of the disjunct classes "sedan" and "pickup truck", in consequence expecting separate detectors. In a dataset depicting many different air and ground vehicles, it might make sense to group these objects into the category "car", in a dataset consisting only of cars we might want to split them. But when splitting them further, we might run into the risk of our algorithm to separate classes by orientation (e.g. back views and front views, as depicted here), which is contrary to our declared goal.

2.4.2 Processing chain

Henceforth we will elaborate on the determination of class-belonging and training of the resulting, combined class models.

Figure 2.7 shows an overview of our processing pipeline. Supplementary, the pseudo-code in algorithm 1 shows the program flow of our pipeline. First, the tracker is run each video of the training dataset (fig. 2.7 (a)). By seperating the figure masks from the obtained dense motion segmentation, object *sequences* are created. We call a sequence the colour information of the frames in conjunction with the figure-ground-segmentation of a single moving object in the scene. Multiple sequences may be created from a single video, when several moving objects can be seen.

Then, a detector is trained for each sequence, as described in section 2.3.1, with descriptors only extracted from the figure-area (fig. 2.7 (b)). This results in a set of very specific, over-fitted codebooks, which in theory only can be used to detect

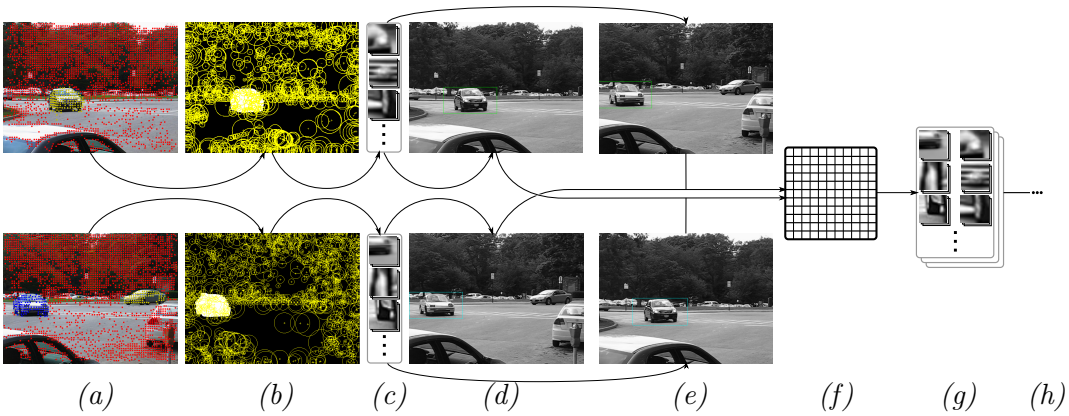

Figure 2.7: Schematic overview of the processing chain: (a) tracking results, (b) figure-ground segmentation mask and interest points, (c) codebooks with clusters represented by averaged centres, (d) baseline detection, (e) detection on unknown sequence, (f) similarity matrix, (g) merged codebooks, (h) detection and merging process reiterates with combined codebooks.

an explicit example of an object category. Since these codebooks are created only from a single sequence, we call them *1-codebooks* (fig. 2.7 (c)).

The training procedure begins with running detection tasks (see section 2.3.2) with every 1-codebook. Detectors using said codebooks are applied to every sequence, considering only figure area, to reduce computation time and avoid training on false positives which would introduce labelling noise (fig. 2.7 (e)). Of special importance is the sequence the detector was trained on (fig. 2.7 (d)), which will serve as a baseline, showing what the best achievable result is for the given sequence is. For detections during the training phase, we set a reduced detection threshold, thus allowing detections with higher uncertainty. This leads to "false-positive" detections, which show that other objects in fact share features with the training sequence. With a sensible choice of thresholds, this allows us to infer common object categories when two sequences in question share enough features to exhibit a multitude of detections.

The number of detections on an unfamiliar sequence informs a similarity measure of the two sequences and hence of the codebooks (fig. 2.7 (f)). When reaching a specified similarity threshold the two 1-codebooks are merged to a 2-codebook, trained on both sequences and thus detecting instances of their common object category (fig. 2.7 (g)). This clustering of codebooks is elaborated on in section 2.4.4.

After running detection with the newly found 2-codebooks on every sequence, we compute similarities to every 1-codebook and 2-codebook to find further merge-candidates. This step is continued in multiple iterations until no further merge is warranted by the similarity threshold. The end result is a set of codebooks of differing size (*n-codebooks*), modelling the shape and appearance of multiple object

Figure 2.8: Detector trained on cars5_1 with results on sequence cars3_2 in blue, baseline in green. Frames 1 to 3, from Hopkins155 dataset.

classes. For illustration, consider the resulting similarities to all sequences of the dataset in tables A.1 to A.6, where codebooks had been combined until the mean similarity fell short of 15%.

2.4.3 Similarity

A detector that is trained on a set of images and then used to detect object instances in that same sequence is expected to deliver near perfect results. When the codebook building process does not introduce too much abstraction, there will be a detected object in every frame of a sequence in which the object is detectable. It could be undetectable because of a small size or occlusion, both of which reduce the detection certainty below the detection thresholds.

This allows us to use the detection results of a detector d_i on it's own training data as a "baseline", determining what the ideal achievable detection performance looks like for a given sequence s_i of instances of object o_i. We will compare those results with the detections of another detector d_j, trained on a sequence s_j containing a different object o_j, to evaluate the second detectors performance when detecting object o_i. Those detection results (see figure 2.8) and the results of detector d_i detecting object o_j will be used to calculate the reciprocal similarity of detectors d_i and d_j.

The maximal similarity is being achieved, when detector d_i detects every instance of o_j that was also detected by d_j and vice versa. This would mean the objects o_i and o_j are probably the same exact one. Even though one detector might not be able to generalise to the other sequence, many matching detections with the second detector still allow for the assumption of similar objects o_i and o_j. Similar enough in fact, to warrant the hypotheses that o_i and o_j belong to the same object category and should be used to build a new codebook to train a common detector $d_{i \cup j}$.

Specifically we match detections of d_j in every frame f of sequence s_i with the detections of the baseline detector d_i, to compute their similarity:

$$sim'(d_i, d_j) = \frac{1}{|d_i(s_i)|} \sum_{f \in s_i} J\langle d_j(s_i^f), d_i(s_i^f)\rangle \tag{2.8}$$

Algorithm 1 Processing Chain

Require: set of videos V

```
S ← ∅  #set of sequences
D ← ∅  #set of detectors
for all v_i ∈ V do
    s_i ← motionSegmentation(v_i)
    S ← S ∪ s_i
end for
for all s_i ∈ S do
    d_i ← trainDetector(s_i)
    D ← D ∪ d_i
end for

R ← ∅  #set of detection results
Q ← ∅  #list of candidates for merging
repeat
    for all s ∈ S do
        for all d ∈ D do
            r_d^s ← detect(d, s)
            R ← R ∪ r_d^s
        end for
    end for
    for all detection results (r_n, r_m) ∈ R × R do
        sim ← calcSim(r_n, r_m)  #compare detector pair
        if sim > threshold then
            Q ← Q ∪ (d_n, d_m, sim)
        end if
    end for
    cluster(Q, sim)
    for all merge candidates d_n, d_m ∈ Q do
        D ← D \ {d_n, d_m}
        d_{n∪m} ← trainDetector(s_n ∪ s_m)  #retrain merged detector
        D ← D ∪ d_{n∪m}
    end for
until Q = ∅
```

and reciprocally

$$sim(d_i, d_j) = \frac{sim'(d_i, d_j) + sim'(d_j, d_i)}{2} \tag{2.9}$$

The number of frames with matching detections is normalised by the number of baseline detections $|d_i(s_i)|$ so the similarity is not biased by the length of the sequence nor the a priori occurrence probability of the object in the input video. We assume detections to be matching where their Jaccard index is above

$$J\langle d_j(s_i^f), d_i(s_i^f)\rangle = \frac{|d_j(s_i^f) \cap d_i(s_i^f)|}{|d_j(s_i^f) \cup d_i(s_i^f)|} = 0.5 \tag{2.10}$$

2.4.4 Merging

Merging of codebooks defines the relation of intra-class variance to inter-class variance. We use a simple and naive average-linkage clustering scheme, in which we merge the most similar codebook pairs of every iteration, until no candidate pairs above a given similarity threshold are left. This threshold controls how much intra-class variance should be allowed and thus in effect, the size of the final codebooks. Average linkage clustering defines the distance between two clusters A, B as

$$dist(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} dist(a, b) \tag{2.11}$$

We use as the distance of codebooks $dist(a, b) = 1 - sim(a, b)$, where pairwise similarities between codebooks are calculated with (2.10). The distance to a higher order (i.e. clustered) codebook trained on a set of sequences K therefore is defined as the normalised mean of distances to constituent codebooks. This way, reciprocal similarity and thus the likelihood of merging some codebooks is identical before and after a merging step.

While merging codebooks generally increases the probability of detecting further objects in unknown sequences, the method chosen for clustering can affect the resulting categories. We repeatedly observe merged categories which show a relatively high similarity (sometimes even from early on) to additional sequences, but are not merged, because the other sequence was previously merged into a second category, with the remaining sequences in the second category showing low similarity scores (see section 3.2.3.3 for full discussion). A method which tends to create chains (such as single-link clustering) might alleviate this to the detriment of compactness of clusters. Further clustering methods have to be evaluated to find computational solutions, where the similarity of resulting categories compares more closely to the subjective similarity apparent to a human user.

2.4.5 Problems and pitfalls

Dealing with the background in training sequences is especially problematic. It is hard to define what constitutes “background” in computer vision tasks. Usually the

background (or *ground*) is distinguished from foreground objects (or *figure*) by having a much higher variance of appearance between its instances. Since we neither have a distinct set of object classes at the beginning, which we can use to define what a figure should look like, and we cannot discern background before training. Therefore we can't easily decide which of the tracked objects constitutes the background. In other words, the motion segmentation procedure does not provide a *figure-ground segmentation* but rather a *figure-figure segmentation*. It is important to note, there might be multiple segments constituting what a human might call the background (when dealing with over-segmentation) or even none (when a moving objects fills the entire camera viewing range).

Additionally, a lot of non-moving objects might appear in the background, which constitutes a large amount of labelling noise which first, makes discriminative learning harder and second, might lead to merging of background with object sequences. Thus, background sequences should be removed before clustering, but a sensible method is far from trivial and remains to be established.

A partial solution is working only on figure area during training, by masking the object sequence during both codebook building and detection to calculate similarity of codebooks. Exploiting additional available prior information in this way makes the training procedure more robust, since fewer false detection bring noise into the codebooks. Also computation time for a video sequence gets reduced, because of interest points from the background not being handled. This happens at the cost of an increased number of required iterations, when there are multiple objects in a video, which is not a problem with most datasets.

Conversely, over-segmentation is very problematic. First, some segments might not show enough descriptive features at all to build a codebook and second, each of the segments of an object might show quite different descriptors, so it becomes very difficult to merge them together later. In the end, no consistent object model can be generated and comparison to similar objects becomes impossible.

2.4.5.1 Tracking

Low frame rates lead to segmentation of incoherent objects, resulting in over-segmentation, which greatly increases computation time during codebook building and, ultimately, prohibits building a sensible object model. This problem translates especially to quickly moving repetitive background, where the result will be an arbitrary ground segmentation, leading to a range of problems, already described in section 2.4.5.

On the other hand, too high frame rates lead to not detecting any movement at all when observing slowly moving objects. While the tracker works well on relatively quick movements, the global smoothness requirement in the trajectory clustering regularly leads to disregarding objects showing only little change between frames. As a positive, this effect reduces noise and slightly moving background (e.g. trees in the wind) gets redacted.

If noise indeed is tracked, the result are segments with a size of only a few pixels. Because of their small figure area, they don't provide enough possibilities to sample features, resulting in empty codebooks, thus not affecting detection results. Since even empty detectors will be checked for similarity in the codebook merging process, they are trimmed beforehand by a simple requirement of minimum figure area, to reduce computation time.

2.4.5.2 ISM

The provided detector implementation has a serious limitation in relying on bounding boxes with a fixed, predetermined aspect ratio and size. They thus should be compared to ground-truth data only with reservations. When working on objects of roughly the same dimensions, which are of course influenced by orientation, the flawed bounding boxes are not much of a problem, but when observing a wide range of poses, the criteria for a good match have to be adjusted accordingly. Nevertheless, to examine the quality of a localisation, it is instead advisable to foremost consider the supporting evidence.

Chapter 3

Evaluation

Contents

3.1 Experimental set-up 31

3.1.1 Testing targets 32

3.1.2 Datasets 33

3.2 Experiments 34

3.2.1 Appraisal of tracking method 34

3.2.2 Appraisal of detection method 35

3.2.3 Detailed discussion of results 37

3.1 Experimental set-up

We applied our method to a number of publicly available videos. To test this new approach we only chose rather plain and uncluttered scenes containing rigid objects, mostly vehicles, to examine the performance under constrained conditions. It has to be noted that most datasets are intended for supervised learning methods and show a wide range of different objects and configurations in a given class, each with only few example images. This is unsuitable for us, since we repurpose the datasets for unsupervised learning and can't handle too great intra-class variance.

At the beginning, we train detectors on single sequences, with only a few dozen images of the same object from slightly different viewing angles. This makes generalisation extremely difficult, since the detector is at first not trained by processing features common to all objects of a class. The expectably ambivalent detection quality and necessary selection of datasets makes calculation of common performance indicators (e.g. precision-recall curves) inexpedient.

Missing these customary metrics makes a standardised quantitative comparison of detection results to other methods difficult. Instead we put our focus an a qualitative analysis of our results, thoroughly interpreting discovered faults to further the understanding of the problem domain at hand.

To evaluate the usefulness of our method in practice, we nevertheless have to reach a certain level of quality in the detection results. We test this by looking at different test tasks.

3.1.1 Testing targets

To begin with, we make a simple test whether our detectors were correctly trained, by looking at the quality of detections on the training sequence, where every frame with a visible object should show a sensible detection. A cross-check of merged detectors against their constituent 1-detectors will show whether generalisation impairs the representation accuracy of exemplars in the object model the codebook was created from.

After a few iterations of codebook merging, detection results will improve for a wider selection of sequences, but still fall behind supervised methods which can rely on a great deal of prior knowledge and thus better handle a higher diversity of data points in training.

Accordingly, one of the main concerns in our experiments is the ability of the detectors to generalise to additional sequences, resulting in larger merged codebooks. Of further interest is, whether this allows the generalisation especially to objects seen from different viewpoints than the object in the training sequence. We will elaborate on the evaluation of our hypothesis, that local appearance codebooks trained on moving objects are able to model objects from varying poses.

Of course, merging with objects which are clearly of a separate category should be considered a fatal flaw. This matter will remain to be somewhat subjective, for a pedestrian should not fall into the category "cars", but the distinction of different builds of vehicles depends on the specific training data and probably very narrow parameter margins (compare section 2.4.1).

The actual usability for real world applications will be evaluated by running test detections on sequences not incorporated during the training stage, i.e. on sequences which where not merged into the codebook. This is notably different from the standard evaluation of most methods, where disjunct test and training sets are used. We train very specific models over a complete video sequence or even a set of sequences, where the detectors were only trained on a subset of the frames. Since other objects might be present in those frames and we don't know in advance, which frames are not used for training we can not—but also do not have to—completely separate the test set from the training sequences. Of course, during evaluation detecting the exact object instances the detector was trained on will not be treated as a successful detection of unknown instances.

By running test detection tasks on the whole image instead of on the figure area only, further objects in the background (e.g. parked cars) can be detected. Some datasets only provide ground-truth annotations for moving foreground objects, hence evaluation has to be done manually. For sequences which do not provide ground-truth data for moving objects either (either because the dataset is not (fully) labelled, or actually intended for different computer vision tasks), we can evaluate detection quality at least by comparing against a baseline—i.e. a detector trained exclusively on the given object in said sequence.

Of course, false-positive can only really occur in the background as well, since we don't expect too many classes of moving objects, and even unmerged categories

of vehicles (predominant in the chosen datasets) showing some similarity are not generally to be considered a false detection. Since training is done on a reduced number of exemplars without possible discrimination from domain negatives we do expect a number of high confidence false positives.

3.1.2 Datasets

After initial trials, we found two datasets suiting our needs: The first dataset we consider is the Hopkins155+16 traffic video set [Tron & Vidal 2007]. Leibe and Schiele use it for validating their algorithms, thus we can expect it to produce applicable results for our work. Initial experiments already show, we can't obtain results even close to their reported quality when generalising to unknown class instances, because our detectors are extremely overfitted, since they are trained on multiple views of only one to a few specimen of a given class, whereas [Leibe *et al.* 2008] was able to use the entire dataset for training a single class. There are 14 short video sequences showing only few objects each, often single moving cars and some parked ones. The various models of cars can be seen from different perspectives and sometimes during a turn in an intersection, which helps us to train detectors from multiple viewpoints. Sequences are taken with a handheld, moving camera, show some partial occlusion and are 16–50 frames long.

More challenging is the VIRAT video dataset ([Oh *et al.* 2011], Release 2, ground dataset, video 050000_01) showing moving vehicle and pedestrian traffic. Although intended for event and activity detection in a surveillance situation, the long sequences with fixed perspective are well suited to extract and compare moving objects with our approach. Since most of the provided videos come in high image resolution and framerate, we downsample the data to reduce computation time and allow comparison with the previous dataset. Most traffic scenes show flowing traffic in one image area (sometimes in the distance with small objects) with very consistent movement and another part with few moving objects but a lot of interaction. We chose few exemplary sequences to observe certain conditions: In the VIRAT050000 scene, multiple objects (cars and crossing pedestrians) are visible for about 50 frames each with little occlusion. The cars move in a straight line, visible from a top–back–side view, along with parked vehicles.

Since the tracker does not expose many parameters, choice and preparation of input videos is paramount for obtaining consistent results. We use videos with a display format close to 480p (aspect ratio varying slightly) as a compromise between level of detail and computation time. We had to convert the videos from different datasets to varying frame rates, because the tracker proved to be unreliable under some circumstances.

To be comparable, every detector in a given experiment (i.e. per dataset) is configured with identical parameter settings. Otherwise it would be difficult to decide automatically, what parameters to chose after merging codebooks, and thus merged codebooks might lose the ability to properly detect objects. Since we have to adjust parameters to adequately cover detection for the majority of the objects

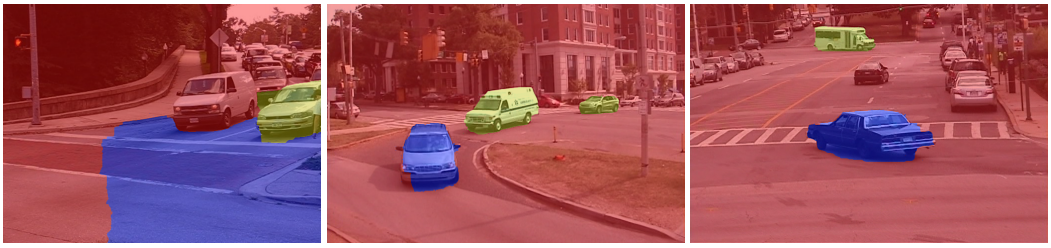

Figure 3.1: Erroneous tracking on the Hopkins dataset. Overlay of dense motion segmentation on sequences cars9 (left), carsTurn (center), carsbus (right).

present in the scene, this can lead to separation of e.g. objects of different scale (e.g. buses/lorries and other cars), meaning comparatively large and small objects will generally not be able to be modelled correctly (because either not enough features can be extracted, or features only visible with sufficient resolution will be predominant) and thus never merge with other codebooks.

As is the case for any computer vision method, which is not supplied with special prior knowledge from camera calibration or a description of the observed scene, choice of parameters will be highly dependant on the datasets and the objects depicted therein. So unfortunately, when transferring these methods to new domains, a complete reevaluation of every selected parameter with multiple experiments is necessary.

Labelling of objects as seen throughout this document is a result of the tracking and motion segmentation procedure and takes the following form: *<video name>_<sequence number>* with the sequence number being an arbitrary numerator of the tracked objects in the video.

3.2 Experiments

In the following sections we will firstly evaluate the performanced of the used tracking and detection framework, before closely considering the experimental results on the basis of our previously stated targets.

3.2.1 Appraisal of tracking method

On the Hopkins traffic dataset, the tracker provides overall good results. Most objects in the sequences are segmented entirely and precisely. Minor flaws can be found firstly, as labelling noise resulting from misinterpreting object boundaries, either where a shadow moves with it, small concavities get attributed to the figure and where the objects move close to the image border and the remaining gap does not provide enough information to be separated from the object. Secondly, we sometimes observe additional tracked objects, appearing only for few frames and showing only a few pixels figure area in their segmentation masks. We couldn't find a definite cause for this, sometimes it might just be noise in the input images, in a few cases it is a result of small or nearly fully occluded objects.

Figure 3.2: Dense segmentation overlaid on frame 485 (left) and 290 (right) of VIRAT sequence. Red: background, green: figure area of object VIRAT_8 (left) and VIRAT_6 (right).

Figure 3.1 shows frames from the only three flawed sequences: the first one being the most problematic, as over-segmentation leads to training an object model on the noisy structure rich background, resulting in an erroneous category merge later (see table 3.2). The second showed two objects moving in such a similar fashion the tracker considered them one; and thirdly, a single car with little apparent motion being missed by the tracker entirely.

In other sequences, as in the scenes of the VIRAT videos, we observe a lot of misses in the tracking of small objects. This can be attributed to the great distance to the camera, allowing too few tracking points on small objects, like pedestrians or the scooter in the left image of figure 3.2. Unfortunately, increasing the sampling density of tracking points did not improve the results enough before the increasing computation time made further experiments impossible. The red car visible in the same image is missed for indeterminable reasons.

Smoothly moving traffic provides difficulties, since cars often move coherently in tight groups (when not stopping or turning separately) and thus get tracked as a single object, consider the right hand side of figure 3.2. This results in further and more dramatic problems, further described in section 3.2.3.

In general, the quality of detected objects is sufficient for our needs, as most errors can be handled by further processing, while missing objects during tracking will of course impede the overall quality of results.

3.2.2 Appraisal of detection method

The biggest observed problem with the use of the ISM for object detection becomes apparent with the occurrence of objects in a great range of scales. It manifests in wrong localisation of objects especially when they are larger than the objects the detector was trained on. We observe unfitting bounding boxes, even though the supporting evidence is good. Position voting becomes difficult when the detector was trained on other sequences and trying to generalise object parts to other, larger objects. For example, wheel wells or the undercarriage of cars seem to be good, discriminative feature along many scales, but lead to position and scale voting far

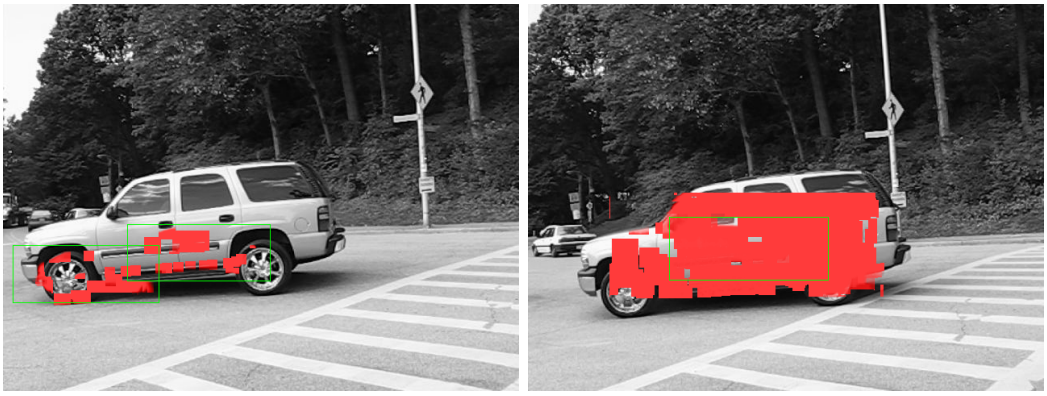

Figure 3.3: Left: detection of merged patch-detector cars3_2-cars5_1-cars2_06_1-cars6_1 on cars1, right: detection of merged patch-detector carsbus1_1-cars1_1 on cars1. Red: supporting evidence, green: detection bounding boxes.

Figure 3.4: Left: output of interest point detector for the frame 9 of sequence carsTurning. Interest points with scale in yellow. Right: false positive detections of merged patch detector cars3_2-cars5_1-cars2_06_1-cars6_1 on same frame. Red: supporting evidence, green: detection bounding boxes.

off the desired result, due to the occurrences stored in the codebook (see figure 3.3 left, where the detector was trained on much smaller cars). This can be alleviated to a certain degree, by enlarging the Hough space along the scale dimension, to the cost of considerable computation time.

Unfortunately further problems arise with fixed bounding boxes concerning object size. Not only do all detectors have to share the same object size, even though they describe differently sized objects, it seems the detector only refines the detection by sampling more features in the vicinity of the detection hypotheses. This leads to problem at the detection boundaries which might not represent the object boundaries (see figure 3.3 right, showing a detection on training data).

Finally, as expected when training without negative examples, reliance on few high contrast edges is observed in the Hopkins dataset. In some structure-rich image regions, so many interest points are found (compare figure 3.4 left) that some match by pure chance. This leads to the car boundary appearing to be indistinguishable

to the kerb (consider the detection of the parked car to the left in figure 3.10.) This can lead to high confidence false positive detection in some sequences, as seen on the right in figure 3.4. This is the reason for most false positives in the Hopkins dataset, especially with shape context descriptors (compare table 3.2.3.5).

3.2.3 Detailed discussion of results

To begin with, when removing all detectors with empty codebooks or those which do not detect a single occurrence on their training sequence, the remaining detectors show a good representation of their constituent object. Looking at the training sequences, each detector provides perfect localisation and precise backprojected segmentation. Sometimes we see missing detections for single frames because of occlusion, whence not enough descriptive features being found, resulting in falling short of detection thresholds.

3.2.3.1 Generalisation to foreground objects

Because the VIRAT dataset mostly shows objects from a far distance, classification has to rely on a small number of features. This makes textural information more important than shape. The problems arise when the cars in the sequence are too small to show common features like lights or grilles. Instead, the chassis colour becomes descriptive in such a way, that detectors of differently coloured cars don't get merged. VIRAT_4-5 shows this spectral sensitivity and only detects dark cars. See exemplary figure 3.5, where the white car is not detected, even though they have the exact same orientation. Compare this to the Hopkins sequences 4 and 5 (see figure 3.6), where a black and white car seen from the identical camera position show the highest similarity in the experiment. Apparently, the VIRAT sequence does not show enough detail and also an unfortunate viewing angle for extracting common features. Compare the evidence provided by wheel wells, front grilles, and underbody edges provided in close-up shots of cars in the Hopkins sequences.

We observe some problems arising from errors in the tracking results. Segmenting two coherently moving cars is ambiguous even for the best methods, when working solely with bottom-up cues instead of a top-down object model. Thereby, follow-up problems in the detection results occur with detection methods leveraging shape and spatial context.

Firstly, the most apparent problem stems from a good detection, that does not lead to a merged codebook, because only one of two combined objects is detected whereas the baseline trained on the conglomerate object shows a different detection. Consider figure 3.7, where the object 01 consists of both the bus and the trailing sedan, resulting in a baseline bounding box somewhere in the middle (conforming to the underlying occurrence distribution), whereas the detection of VIRAT_5-4 perfectly captures the latter vehicle.

Less conspicuous is a phenomenon encountered because of Hough voting with local shape descriptors. When casting votes for the localisation of an object, the

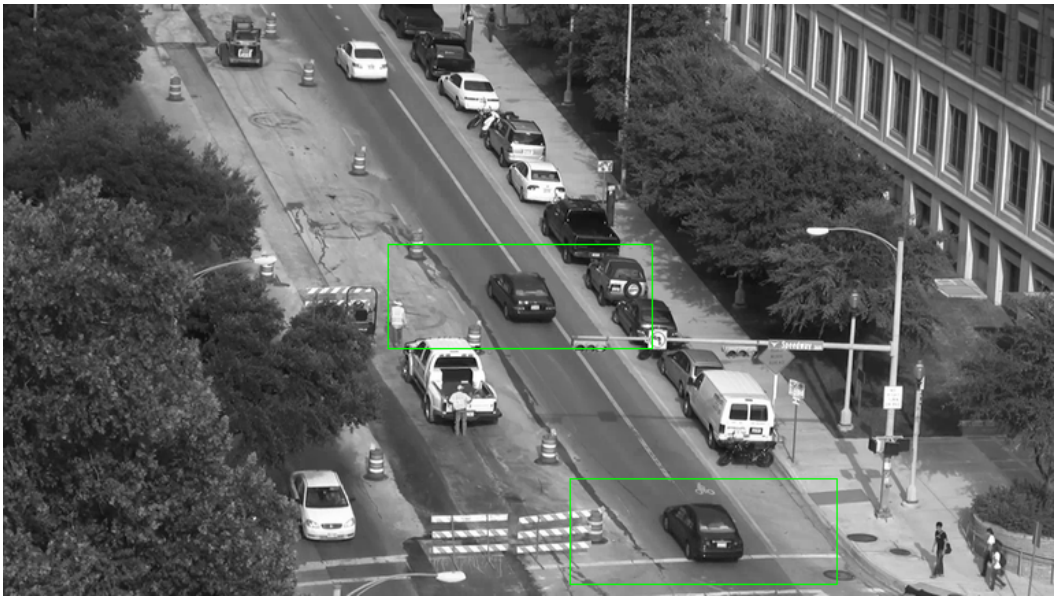

Figure 3.5: Merged patch-detector VIRAT_05-04 with recognition results (green: detection) on sequence from VIRAT dataset. Frame 532.

position of an object hypothesis is defined by the centre point. Since the centre of two objects wrongly combined by the object lies inbetween those objects, the offset vectors describing the relative position of features of these objects might lead to votes cast far from the actual object centre along either direction. When votes from a second object corroborate this hypothesis, this leads to a localisation in the middle of two objects, as can be seen in figure 3.9. When these two objects have indeed been segmented as a single moving object, we do not consider it a false detection, since it does capture the properties of the observed "object".

When there is no second object, but the features of a single found object cast votes for both of the two objects of the trained occurrence, there might be enough confidence to lead to a faulty detection. This can show in one of two ways: First, by a second, additional object detection, where no object can be seen (where one of two detections is considered a false positive) or second, when the evidence for the combined object is stronger than for the singularly detected one, a detection completely off from the real specimen. This is also considered a false detection. Henceforth we call these phenomena *phantom detections*, since the supporting evidence shows a ghost-like shape of a car, which is actually not there (see figure 3.8 for reference). These can be avoided to a certain degree with very conservative detection thresholds, but not without greatly reducing correct detections, hence we will put special scrutiny on them in the experiments.

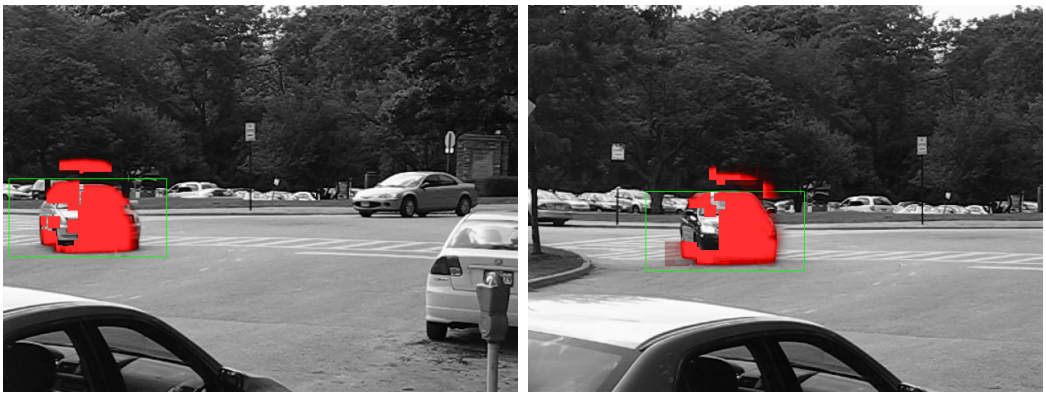

Figure 3.6: Patch-detectors cars4_3 (left) and cars5_2 (right) with recognition results (green: detection, red: supporting evidence) on frame 13 of sequence cars5 (left), and frame 18 of sequence cars4 (right) from Hopkins dataset.

Figure 3.7: Dense segmentation overlay of virat0500_01_01 (left) and detections of merged detector virat0500_01_05-virat0500_01_04 (blue) and baseline (green) on frame 215 of sequence virat0500_01, from VIRAT dataset.

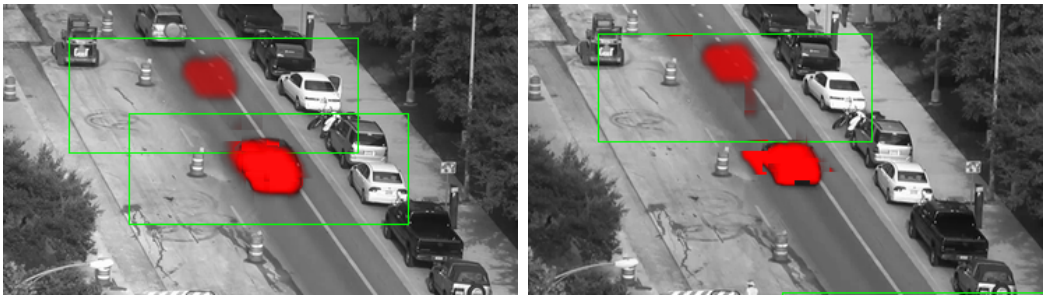

Figure 3.8: Merged patch-detector VIRAT_05-04 with recognition results (green: detection, red: supporting evidence) on sequence from VIRAT dataset. Detail of frames 310 (left) and 542 (right).

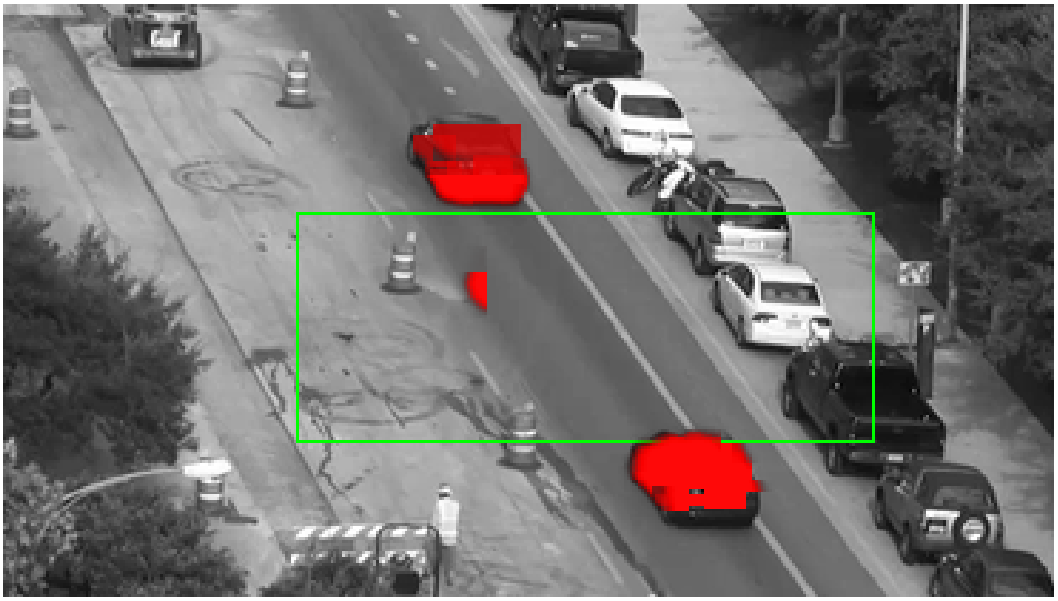

Figure 3.9: Merged patch-detector VIRAT_05-04 with recognition results (green: detection, red: supporting evidence) on sequence from VIRAT dataset. Detail of frame 547.

3.2.3.2 Generalisation to other viewpoints

The Hopkins dataset provides many sequences with cars visible from a variety of viewpoints. Therefore it is well suited to evaluate the generalisation to different poses. Even though ISM was evaluated with this dataset and thus should be able to accomplish good results, we note that Leibe et al. trained separate detectors for multiple viewing angles to achieve the results in [Leibe *et al.* 2008].

During the first merging steps, mostly cars from similar viewpoints get merged, because they show the highest similarity scores. When sequences showing a rotating object get merged, this allows increasing similarity to objects from slightly different viewpoints, since features of additional orientations get incorporated into the codebook. Subsequently, this allows merging of objects seen mostly in a different orientation but either sharing only a couple of frames in a similar one or can exhibit common features. Of course, similarity scores for these merging candidates will be lower. After a few iterations, the most descriptive features become apparent. For example, wheel wells are well visible from cars seen from a 45° angle between side and front and provide a good rotation-invariant feature allowing to generalise to both front and side views of other cars.

Examining the resulting codebooks shown in table 3.2 underpins this observation: Compare the Hopkins codebooks containing cars4_3, cars5_2 and cars2_06_2. The first two objects show cars first from the side, then rotating to a front perspective. After they got merged, the last object, seen from a front–

side view only, gets incorporated. The sequences cars3_2 cars5_1, cars2_06_1 and cars6_1 all show side views, and only generalise to objects with further views when using shape context descriptors.

As we can see, generalisation of codebooks is only possible to the degree of rotation captured in the single sequences. When no object turns full 180°, (mirroring is captured easily, by supplying the detector with mirrored features), then complete generalisation is unlikely, because there will be few or even no features at all matching objects which were seen from a completely different viewpoint.

In general, a local appearance codebook makes localisation difficult, because features produce a lot of different, contradictory votes when fetched from objects with slightly different rotations. With a following hypothesis selection as employed here, this problem will be mitigated to some degree, by returning a single refined detection after employing the MDL selection but the backprojected segmentation will remain imprecise.

One more notable effect stems from the merging of detectors for objects of different scales. As can be seen in figure 3.10 on the left, detections for both cars are supported by evidence clearly outside the extent of the object. Apparently, since the detector has to rely on only few codewords, features provide conflicting evidence and Hough votes for detections of different sizes. The following extraction of Hough maxima can incorporate this to a certain extent to refine the hypothesis and find a suitable detection.

As mentioned before, greater scale differences are problematic: cars1_1 and cars10_1 are large objects (over four times the area of others) and never merge because the detection bounding boxes don't even fit the object well in the baseline.

3.2.3.3 Generalisation resulting in larger merged codebooks

Generally, we observe, merging of codebooks does not reduce their expressiveness. The quality of detections on the training sequences does not worsen after merging codebooks. The clustering of features and representing them by their centres, allows codebooks to grow in size without losing information. The exception is codebooks merged from sequences with wrongly combined objects. When the single-object features of a merged detector detects only a part of a combined object, the bounding box will be wrong. Similarity scores to the training data might reduce, e.g. in the case of VIRAT_05-04 to 95% (see table 3.1).

As we have shown in the previous paragraphs (under 3.2.3.1 and 3.2.3.2), the overfitted reduced codebooks trained from single objects we started with can indeed generalise to other objects—under certain conditions.

On the other hand, especially detectors from the VIRAT sequence show a lot of detections in the test sequence (see 3.2.3.5 for details), but merging of codebooks is not happening for different reasons.

There are three main causes of insufficient results in number and size of codebooks. First, some objects do not get tracked at all, which is the main problem in the VIRAT050000 scene. Even when detectors would be able to correctly localise

Figure 3.10: Merged patch-detector cars5_1-cars3_2-cars2_06_1-cars6_1 (left) and VIRAT_8-7 (right) with recognition results (green: detection, red: supporting evidence) on frame 36 of sequence cars4, from Hopkins155 dataset (left) and frame 378 of VIRAT sequence (right).

these objects, since they have no defined figure area, we cannot build codebooks from them. Second, the tracker provides bad results, resulting in multiple objects being tracked as one. Through the problems described under 3.2.3.1, this results in bad matches and hence no merging of codebooks. But even when they are merged, detection quality reduces further and further merging becomes unlikely. Last, many objects which do show sufficient similarity don't get merged because they are not similar enough to the whole cluster. The objects might have not been clustered in early iterations, because other sequences showed higher similarities, but later cannot be merged, because the average-linkage clustering scheme penalises when the objects show low similarity to a subset of members of a target cluster. This could be addressed by choice of a different clustering scheme, e.g. single-linkage clustering, but more evaluation is required to determine to what extent formation of chains is desirable and acceptable.

This is aggravated by similarity often observably being only unidirectional. I.e. detector A shows a lot of detections on sequence B, but B none on sequence A. Hypothetically, this implies that some sequence are just better suited for building descriptive codebooks, either because they have more frames and show the object from a better angle or are just less noisy or show more consistent lighting.

3.2.3.4 Further objects in the background

After having only detected on figure area during the training procedure, afterwards we can run detections on the whole image area. This allows us to detect cars that are either not captured by the tracker, or static during the whole sequence, e.g. parked cars. Figure 3.10 examples for background detections in both datasets. As we can see from the number of detections in table 3.1, the detector VIRAT_7-8 with shape context descriptors is the only one to regularly detect parked cars in the VIRAT sequence. It detects all three white parked cars (the one on the centre strip

of the street is actually moving very slowly and is missed by the tracker entirely) in almost every frame.

Looking at the right hand side image in figure 3.10, explains why: apparently the detectors rely heavily on the bottom edge and high contrast features (such as rear bumpers and dark lights on a white car) to detect vehicles. Observing cars entering and leaving the image, confirms this assumption, showing no detections, when cars have not fully entered the image, but still being detected when having left the viewing range half-way.

3.2.3.5 Quantitative analysis

Table 3.2 lists the experiments that went into this evaluation, with their used datasets (see section 3.1.2), applied feature descriptor and the number of separate objects detected by the tracker. The last column depicts the result of the training procedure: a list of the remaining codebooks. 1-codebooks were omitted because they can be deemed insufficiently descriptive to allow detection of unknown objects.

The resulting codebooks are picked up in table 3.1 for comparing their detection results. The first two columns, again, show the used dataset and descriptor. These are followed by the detector in question, defined by it's set of used sequences. Next is the total number of detections that was obtained over all sequences with said detector. Since the used processing chain makes a strict separation of test- and training-data difficult, the number of detections on sequences the detector was trained on are listed. The sixth column represents the valid and true positive detections that could be determined by comparing against the baseline detectors of each sequence, followed by the number of false positive or wrongly localised detections. In the last column the calculated CorLoc measure is stated as a percentage. Note, that for calculating this measure, the detections on the training sequences were subtracted.

Merging with objects, which are clearly of a separate category should be considered a fatal flaw. Because of our choice of datasets, the only possible false categorisation of moving objects could be detecting pedestrians with one of our car detectors. We found no instance of this happening. Instead, we observe a number of false positives in both structure-rich and textureless areas of the background.

Specifically, the merged detector VIRAT_05-04 showed a few false positives on an empty street (see figure 3.11, right) both for patches and shape context descriptors. The detections show high confidence scores of about 1.6 (compared to scores of 1.1 to 1.8 of correctly detected objects outside of the training set) and are not consistent (i.e. five false positives at a similar position in the background, visible over 772 frames). Presumably, the road markings contribute to the evidence, because they were incorporated during training and adverse lighting conditions worsen the problem.

Even though one Hopkins sequence is not properly tracked and has a single figure segment for two separate cars, middle and phantom detections don't occur here. Presumably because detection of objects rely on more features and thus spurious ev-

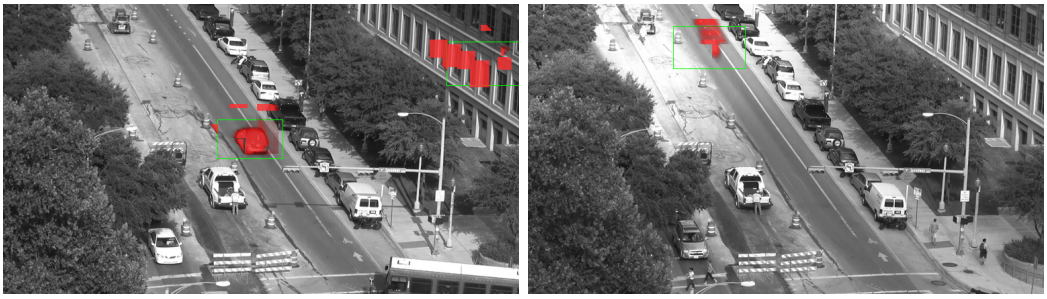

Figure 3.11: Shape context detectors VIRAT_2-1 (left) and VIRAT_5-4 (right) with recognition results (green: detection, red: supporting evidence) on frame 176 (left), and frame 676 (right) of VIRAT sequence.

idence is not strong enough to reach detection thresholds. For the VIRAT sequence we observe problems resulting from false tracking only with pixel patch descriptors. Shape context descriptors make the detectors way less susceptible to problems with objects wrongly combined by the tracker. Instead, it might detect the objects of the other sequences but with a low similarity score, because the bounding boxes in the baseline detection are different for combined objects. For example the detector VIRAT_7-8 shows 30 detections on 45 frames of one part of object 3, whereas the bounding box evaluation only results in two matches. It is worse with shape context detectors, because they don't exhibit middle detections which might match better with the baseline.

When looking at the numbers (see table 3.1), shape context descriptors seem to allow better generalisation, leading to more codebook merges and ultimately, more detections than with patches. The higher sensitivity, evident through the higher number of total detections, has the downside of resulting in many more false positive detections. Even though some of those false detections lie on an suitable object (dominant in sequence cars10 of the Hopkins dataset), their size and extent did not warrant terming them correct. Nonetheless, there are a lot of false positives in the background area in single sequences (mostly cars9 and carsTurning of the Hopkins dataset), reducing the usability of affected detectors.

As can be seen from table 3.1 and the calculated *Correct Localisation (CorLoc)*[1] measure, the resulting detectors of the single experiments show very inconsistent results. We can not provide precision measures, because the input videos are not annotated for background objects.

[1]CorLoc is calculated as $CorLoc = \frac{\#totaldetections - \#falsepositives}{\#totaldetections}$ subtracting all detections which match the objects the detector was trained on.

input	feature	detector	total	train	other	FP	CorLoc*
VIRAT	PP	{5, 4}	233	71	12	14	91.4
VIRAT	SC	{5, 4}	131	55	2	5	93.4
	SC	{2, 1}	195	111	2	84	2.3
	SC	{7, 8}	1090	36	30	0	100
Hopkins	PP	{bus_1, 1_1}	171	74	38	77	20.6
	PP	{2_06_2, 5_2, 4_3}	133	104	12	12	58.6
	PP	{bus_2, turn_1, 2_2, 10_2}	170	159	8	22	26.7
	PP	{3_2, 5_1, 2_06_1, 6_1}	291	101	26	103	45.8
Hopkins	SC	{3_1, 2_1}	252	48	43	1	99.5
	SC	{2_06_2, 4_3, 5_2}	420	104	115	119	62.3
	SC	{9_1, bus_2, 10_2 }	238	81	37	39	75.2
	SC	{2_07_1, 2_07_2, turn_1}	664	102	76	403	28.4
	SC	{2_06_1, 5_1, 2_2, 7_1, bus_1, 3_2, 6_1}	602	148	112	205	54.8

Table 3.1: Detections of 2+-detectors. CorLoc* percentage calculated excluding detections on training instances.

Exp.	dataset	feature	#objects	2+-detectors
11	VIRAT0500_01	PP	8	{5, 4}
14	VIRAT0500_01	SC	8	{5, 4}, {7, 8}, {1, 2}
12	Hopkins cars	PP	19	{bus_1, 1_1}, {2_06_2, 5_2, 4_3}, {bus_2, turn_1, 2_2, 10_2}, {3_2, 5_1, 2_06_1, 6_1}
13	Hopkins cars	SC	22	{3_1, 2_1}, {2_06_2, 4_3, 5_2}, {9_1, bus_2, 10_2}, {2_07_1, 2_07_2, turn_1}, {2_06_1, 5_1, 2_2, 7_1, bus_1, 3_2, 6_1}

Table 3.2: Number of moving objects in dataset and resulting codebooks using either pixel patch or shape context descriptors.

Chapter 4

Conclusion

Contents

4.1 Discussion 47
4.2 Future work 48

4.1 Discussion

In this work we investigated a novel approach to use unsupervised learning on video data to train object detectors. The task was separated into a three step process: first, a figure-ground segmentation is obtained solely from bottom-up cues to provide noise free subsets of training data containing separate objects; then, this object-specific training data is used to generate codebooks of local object appearance; last, similarity of objects is incorporated into a clustering of object codebooks, resulting in detectors of combined object classes.

We have selected and discussed state of the art methods for figure-ground segmentation from video as well as codebook building and detection.

While either of the chosen methods might not provide competitive results in modern computer vision challenges, they were successfully integrated into a pipeline establishing a proof-of-concept on select video datasets. To transfer the results to seminal unstructured datasets (e.g. Youtube-Objects, [Prest *et al.* 2012]), allowing comparison to related algorithm, further evaluation is needed. In particular, to build exhaustive models of object shape, training requires videos showing objects in full dimensions, from multiple viewpoints (preferably complete rotations) over many frames in a single sequence. Because training is begun by training detectors on a single object first, before generalising to other sequences, a sufficient number of visually similar objects has to be present in the dataset, to allow gradual growth of codebooks. Full and exhaustive evaluation of the parameter space remains to be done, to determine whether the used algorithms can be applied to more general tasks, or if parts of our process need to be completely revamped.

Section 3 showed a number of shortcomings in the usability of said methods. In particular, even though motion segmentation works quite well with specific datasets, it either misses too many objects or does not correctly discern two objects with similar movement. Better performance might have been achievable in theory, but performance considerations did not allow full exploitation of arbitrary space density of tracking points.

The detection itself did not perform too well either with the given implementation. Detections proved to be inconsistent over the course of a video sequence. That is, one would expect an object that is detected once, be reliably detected again, if it's orientation, shape and texture does not change. Instead, some objects were only detected for a fraction of their visible frames. Occurrence of false positives did not really show a clear behaviour either–in some frames they were at identical locations, and in others they got no evidence. Presumably, the detection ecosystem is quite sensitive to subtle change in the lighting conditions. More parameter evaluation is needed to establish a concise appraisal of the overall applicability of the used detector.

Without doubt, the advocated scale invariance of the ISM framework is not evident. Usable results can only be acquired be either obscure fine-tuning of parameter settings or choosing homogeneous datasets. It comes as no surprise that generalisation to a common object category is impossible when the observed objects are too small to extract common features. A further problem, but no news, is the need for choosing specific feature descriptors for different objects and datasets. Overall, scale and feature selection problems make the preparation of workable datasets paramount. Many datasets are hardly usable and thus make standardised comparison to state-of-the-art methods impossible.

Considering both tracking and detection, it seems a wealth of information present in the data remains unused. Further development might make good progress, employing a more holistic approach, incorporating more contextual information.

4.2 Future work

To improve the motion segmentation entails two parts. First, to reduce missing objects entirely during tracking, clustering of motion trajectories has to be reviewed. Often, tracking points are found correctly on the objects, but apparently, since flow is computed solely between neighbouring frames they get attributed to the background or other objects. This results in objects not being tracked, even though they show a clear and distinct displacement. On the other hand, to reduce combining objects which share the same movement, one has to introduce more top-down appearance information into tracking, since strict bottom-up cues cannot solve this problem. Tackling both of these defects is necessary to reduce false positives, and more importantly, wrongly placed matches which hamper codebook merges, to get better overall results. Only then can we make an admissible appraisal of the generalisation performance.

Having addressed correctable performance losses in tracking and detection on their own, we have to examine the clustering of detectors in detail. Testing and evaluating different similarity measures, thresholds and clustering schemes could fill another paper. Our choice of clustering method was based on the intuitive need for compact clusters. While this assumption might hold when looking at multiple categories with some similarity between them, we need something else when cate-

gorising very diverse object in the wild. Modern datasets incorporate a lot of strictly separable categories ranging from birds to trains. Here the inter-class variance is hardly a problem. Instead, we would need to find a way to properly model and tolerate the intra-class variance (which is especially high in not hand picked datasets like Youtube Objects) to come even remotely close to the performance of weakly supervised learning methods. Conversely to our experiments, in these cases a highly likely formation of chains during clustering might be desirable to exploit the little similarity that can be observed to learn categories. We can already observe the need for this effect when looking at our experimental results. Some sequences share object orientation, which results in the first merges. After only a few iterations, they don't generalise further, even though the combined detectors show excellent results on other sequences, because the average similarity with the remaining objects in the merged codebook is too low.

We expect great improvements not only in performance but also in detection results, by designing a more holistic processing chain. At it's simplest, descriptors calculated for codebook building should not be recalculated every frame, when they were already used by the tracker. If runtime is more of an issue than space requirement, those features should be calculated once per frame and then reused for codebook building, detection and merged detectors. While this does not improve overall complexity, shaving off a few seconds of the pervasive feature extraction is a good idea.

What is more important and should improve performance significantly, is finding some way of integrating parameter estimation from the scene. The greatest challenge with our training framework is fine tuning the parameters in a way that, first, there are good results for each scene, and second, detectors can generalise to each others sequences. Choosing common parameters for all the different objects and detectors in a dataset is at least a compromise if not the reason for bad overall performance. While finding certain abstract thresholds always requires some trial-and-error and provides the basis for selecting a precision-recall target, other parameters can and should be informed by information from the image data. A straightforward example is evident through the problematic behaviour of ISM with differing object sizes. Those could easily extrapolated from the segmentation masks output by the tracker, to resize features and occurrences to a standard value, establishing a real scale invariance in object detection without exploding computational cost through an additional dimension in Hough space.

Using this simplified more robust object model would improve generalisation to other objects in different scenes, but also allow comparison to other datasets. Because we could then use specific parameters per object we enable using very different datasets in conjunction, where objects might be seen in a wider variety of orientations and poses.

Generally a lot of problems stemming from the training uncertainties—e.g. object sizes and parameter estimation, could be handled in an intermediary step with information obtained from the tracker. A specific method to incorporate this additional "meta" information into a training framework needs to be developed—special

difficulties arise because of our use of video: object sizes and other parameters are prone to change within a single sequence, so we need to find a detector which can accommodate it.

In our evaluation we treated the resulting merged codebooks as separate detectors, instead of using them in unison to find the most likely solutions. Considering ensemble learning methods, we should use all detectors simultaneously on each sequence and choose a classification by considering results of every detector. In the simplest case, this would constitute in finding the maximum a posteriori class probability when observing conflicting class hypotheses. Observing high uncertainty in multiple detectors (e.g. with typical false positives in the background) would allow us to deem certain detections incorrect and greatly reducing recall rate.

Finally, analogous to classic unsupervised learning, after initial detections we could train on objects detected in the background as well. While this might help to get more object information in small datasets, it also introduces a lot of uncertainty which we don't have when only training on moving objects with defined figure areas.

Appendix A

Experimental Data

	2_1	2_2	2_06_1	2_06_2	2_07_1	3_1	3_2	4_2	4_3	5_1	5_2	6_1	9_1	10_1	10_2	bus_1	bus_2	turn_1	turn_2
1	-	.94	.47	-	-	-	.90	.07	.45	.95	.21	.94	1	-	.11	.96	.33	1	.33
2	.16	.06	.18	-	.81	.19	-	.71	.05	.24	.11	.25	-	.02	.09	.04	.24	1	.21
3	-	.13	-	.06	-	.05	-	-	.02	.05	.08	-	-	-	.97	-	.96	.44	.02
4	-	.55	-	.88	-	-	-	-	.96	.26	.95	-	-	-	.66	-	.31	.58	.38
5	.94	-	-	-	-	.90	-	.21	.02	.21	-	-	-	.02	.03	-	.02	.38	.15

Table A.1: Similarity matrix for shape context 2+-codebooks and all used sequences of the Hopkins dataset. Empty columns omitted.

1	{2_06_1, 5_1, 2_2, 7_1, bus_1, 3_2, 6_1}
2	{2_07_1, 2_07_2, turn_1}
3	{9_1, bus_2, 10_2}
4	{2_06_2, 4_3, 5_2}
5	{3_1, 2_1}

Table A.2: Detector keys to table A.1

	1_1	2_2	2_06_1	2_06_2	3_2	4_3	5_1	5_2	6_1	10_1	10_2	bus_1	bus_2	turn_1	turn_2
1	.96	.06	.5	-	.90	.11	.95	-	1	.05	.09	-	.08	.13	.05
2	-	.94	-	-	-	-	-	.08	-	-	.86	-	.96	1	.13
3	-	-	.06	.88	-	.96	.05	.95	-	-	.03	-	-	.15	.05
4	.87	.03	-	-	-	-	-	-	-	.02	.06	.98	-	.65	.21

Table A.3: Similarity matrix for pixel patch 2+-codebooks and all used sequences of the Hopkins dataset. Empty columns omitted.

1	{3_2, 5_1, 2_06_1, 6_1}
2	{bus_2, turn_1, 2_2, 10_2}
3	{2_06_2, 5_2, 4_3}
4	{bus_1, 1_1}

Table A.4: Detector keys to table A.3

detector	1	2	3	4	5	6	7	8
{1, 2}	.96	.82	-	-	-	.07	-	-
{5, 4}	.03	-	-	.83	.95	-	-	-
{7, 8}	.1	.1	.2	.26	.13	.18	.68	.95

Table A.5: Similarity matrix for shape context 2-codebooks and all used sequences of the VIRAT dataset.

detector	1	2	3	4	5	6	7	8
{5, 4}	.05	.16	-	.9	.93	-	-	-

Table A.6: Similarity matrix for pixel patch 2-codebook and all used sequences of the VIRAT dataset.

Bibliography

[Ballard 1981] Dana H. Ballard. *Generalizing the Hough Transform to Detect Arbitrary Shapes.* In Pattern Recognition Society, volume 13, pages 111–122, 1981. 21

[Belongie *et al.* 2000] Serge Belongie, Jitendra Malik and Jan Puzicha. *Shape Context: A new descriptor for shape matching and object recognition.* In Advances in Neural Information Processing System (NIPS), pages 831–837, 2000. 17

[Brostow *et al.* 2009] Gabriel J. Brostow, Julien Fauqueur and Roberto Cipolla. *Semantic object classes in video: A high-definition ground truth database.* Pattern Recognition Letters, vol. 30, no. 2, pages 88 – 97, 2009. 3

[Brox & Malik 2010] Thomas Brox and Jitendra Malik. *Object segmentation by long term analysis of point trajectories.* In European Conference on Computer Vision (ECCV), Lecture Notes in Computer Science. Springer, Sept. 2010. 5, 13, 14

[Brox & Malik 2011] Thomas Brox and Jitendra Malik. *Large displacement optical flow: descriptor matching in variational motion estimation.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 33, no. 3, pages 500–513, 2011. 4, 5, 12, 13

[Brox *et al.* 2014] Thomas Brox, Peter Ochs and Jitendra Malik. *Segmentation of moving objects by long term video analysis.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 36, no. 6, pages 1187 – 1200, Jun 2014. Preprint. 5, 11, 14

[Cho *et al.* 2015] Minsu Cho, Suha Kwak, Cordelia Schmid and Jean Ponce. *Unsupervised Object Discovery and Localization in the Wild: Part-based Matching with Bottom-up Region Proposals.* 2015. 6

[Croitoru *et al.* 2017] Ioana Croitoru, Simion-Vlad Bogolin and Marius Leordeanu. *Unsupervised learning from video to detect foreground objects in single images.* Oct 2017. 7

[Endres *et al.* 2013] Ian Endres, Kevin J. Shih, Johnston Jiaa and Derek Hoiem. *Learning Collections of Part Models for Object Recognition.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 939–946, 2013. 6

[Everingham *et al.*] Mark Everingham, Luc Van Gool, Chris Williams, John Winn and Andrew Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.* 6

[Felzenszwalb *et al.* 2010] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester and Deva Ramanan. *Object Detection with Discriminatively Trained Part-Based Models.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 32, no. 9, pages 1627–1645, September 2010. 6

[Horn & Schunck 1981] Berthold K. P. Horn and Brian G. Schunck. *Determining Optical Flow.* ARTIFICAL INTELLIGENCE, vol. 17, pages 185–203, 1981. 12

[Koffka 1935] Kurt Koffka. Principles of gestalt psychology. Harcourt, Brace and Company, New York, 1935. 5

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks.* In F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, editeurs, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. 6

[Kwak *et al.* 2015] Suha Kwak, Minsu Cho, Ivan Laptev, Jean Ponce and Cordelia Schmid. *Unsupervised Object Discovery and Tracking in Video Collections.* In IEEE International Conference on Computer Vision (ICCV), 2015. 6

[Leibe & Schiele 2003] Bastian Leibe and Bernt Schiele. *Interleaved object categorization and segmentation.* In proceedings of the British Machine Vision Conference, BMVC, pages 759–768, 2003. 4, 5

[Leibe *et al.* 2004] Bastian Leibe, Aleš Leonardis and Bernt Schiele. *Combined Object Categorization and Segmentation With An Implicit Shape Model.* In ECCV workshop on statistical learning in computer vision, pages 17–32, 2004. 6

[Leibe *et al.* 2008] Bastian Leibe, Aleš Leonardis and Bernt Schiele. *Robust Object Detection with Interleaved Categorization and Segmentation.* International Journal of Computer Vision (IJCV), vol. 77, no. 1, pages 259–289, May 2008. 6, 17, 18, 22, 33, 40

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision (IJCV), vol. 60, pages 91–110, 2004. 13, 17

[Mémin & Pérez 1998] Etienne Mémin and Patrick Pérez. *Dense estimation and object-based segmentation of the optical flow with robust techniques.* IEEE Transactions on Image Processing, vol. 7, pages 703–19, 02 1998. 12

[Ng & Jordan 2001] Andrew Y. Ng and Michael I. Jordan. *On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes*, 2001. 5

[Oh *et al.* 2011] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia chih Chen, Jong Taek Lee, Saurajit Mukherjee, J. K. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xioyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick and Hamed Pirsiavash. *A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011. 33

[Prest *et al.* 2012] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid and Vittorio Ferrari. *Learning object class detectors from weakly annotated video.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3282–3289, 2012. 7, 47

[Russakovsky *et al.* 2015] 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.* International Journal of Computer Vision (IJCV), vol. 115, no. 3, pages 211–252, 2015. 2, 23

[Shi & Malik 2000] Jianbo Shi and Jitendra Malik. *Normalized cuts and image segmentation.* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 22, no. 8, pages 888–905, Aug 2000. 14

[Tron & Vidal 2007] Roberto Tron and René Vidal. *A benchmark for the comparison of 3D motion segmentation algorithms.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007. 10, 33

[Wang & Gupta 2015] Xiaolong Wang and Abhinav Gupta. *Unsupervised Learning of Visual Representations using Videos.* In IEEE International Conference on Computer Vision (ICCV), 2015. 7

[Zhu *et al.* 2012] Jun-yan Zhu, Jiajun Wu, Yichen Wei, Eric Chang and Zhuowen Tu. *Unsupervised Object Class Discovery via Saliency-Guided Multiple Class Learning*, 2012. 6

Unsupervised Learning of Object Categories from Video

Abstract: Modern computer vision applications have shown astonishing results in recent years. Most of it was only made possible be increasing availability of labelled data used for training various methods. Especially object detection requires thousands of images of diverse objects to provide quality results. Sourcing annotated training data seems to be the bottleneck in computer vision research.

We develop a proof-of-concept method to obviate the need for time consuming labelling of images by hand, by leveraging spatio-temporal cues from video data to inform object recognition.

We show that pure bottom-up tracking provides a sufficing data basis to train representative object models. Furthermore, by using a local appearance bag-of-words model, it is possible to generalise from one class instance seen over multiple frame to other objects, thus establishing a category relation without prior knowledge nor supervision.

Keywords: unsupervised learning, object classification, shape models, motion segmentation

Unüberwachtes lernen von Objektklassen aus Videos

Zusammenfassung: Die jüngsten Erfolge automatischer Bilderkennung beruhen auf der Verfügbarkeit einer großen Zahl an bekannten und annotierten Trainingsdaten. Insbesondere der Bereich der Objektklassifizierung hat einen hohen Bedarf an verschiedenen Bildern von unterschiedlichsten Objekten, um gute Ergebnisse zu erzielen. Annotierte Datensätze in ausreichender Größe scheinen der begrenzende Faktor bei der weiteren Entwicklung des Feldes zu sein.

Wir stellen einen neuartigen Ansatz vor, zeit- und arbeitsaufwändige Annotierung von Trainingsbildern zu umgehen. Dazu werden räumlich und zeitlich abhängige Informationen aus Videodaten für die Objekterkennung nutzbar gemacht.

Wir können zeigen, dass mit einer Segmentierung bewegter Objekte reine Aufwärtsverfahren ausreichend sind, um deskriptive und repräsentative Objektmodelle zu lernen. Durch die generative Repräsentation einzelner Objekte, die aber für mehrere Einzelbilder im Video sichtbar sind, als Verteilung lokaler Teilformen ist es möglich, diese zu vergleichen und dadurch ohne weiteres Wissen oder Überwachung übergreifende Objektklassen zu finden.

Keywords: Unüberwachtes Lernen, Objektklassifizierung, bewegungsorientierte Segmentierung