MINISTERUL EDUCAŢIEI, CERCETĂRII, TINERETULUI ŞI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

PHD THESIS

Object detection based on candidate generation and classification

Detecţia obiectelor bazată pe generarea şi clasificarea candidaţilor

ing. Robert Varga

Conducător ştiinţific: **prof. dr. ing. Sergiu Nedevschi**

2017

Contents

List of Tables 5

List of Figures 6

Chapter 1 Introduction 7
- 1.1 Definitions and context . . . 7
- 1.2 Motivation . . . 8
- 1.3 Thesis objective . . . 9
- 1.4 Thesis structure . . . 9
- 1.5 Acknowledgements . . . 10

Chapter 2 Related works 11
- 2.1 Object detection overview . . . 11
- 2.2 Relevant feature descriptors . . . 12
  - 2.2.1 Color descriptors and color spaces . . . 12
  - 2.2.2 Texture descriptors . . . 13
  - 2.2.3 Shape descriptors . . . 14
  - 2.2.4 Visual descriptors for object detection . . . 15
- 2.3 Classifiers for object detection . . . 18
  - 2.3.1 Decision stumps . . . 19
  - 2.3.2 Decision trees . . . 19
  - 2.3.3 K-nearest neighbor classifiers . . . 20
  - 2.3.4 Bayes classifiers . . . 21
  - 2.3.5 Linear classifiers . . . 21
  - 2.3.6 Artificial Neural Networks . . . 22
  - 2.3.7 Support Vector Machines . . . 22
  - 2.3.8 Boosted classifiers . . . 25
  - 2.3.9 Classifier evaluation measures . . . 27
  - 2.3.10 Classifier evaluation methods . . . 29
- 2.4 Region of interest selection and candidate generation methods . . . 30
- 2.5 Object detection methods . . . 32
- 2.6 Pallet detection methods . . . 34

2.7 Pedestrian detection methods 36
2.8 Stereo matching and reconstruction 38

Chapter 3 Proposed features and classifiers for object detection **41**
3.1 Normalized Pair Differences and other features 41
3.1.1 Normalized Pair Differences 41
3.1.2 Sparse Local Binary Pattern Histogram 44
3.1.3 Edge density features 44
3.1.4 Experimental Results 45
3.1.5 Conclusions 46
3.2 Boosted ensemble classifiers with decision trees as weak learners 47
3.2.1 Decision tree algorithms 48
3.2.2 Ensemble classifier algorithms 51
3.2.3 Experimental results 52
3.2.4 Conclusions 53

Chapter 4 Proposed candidate generation methods for fast object detection **55**
4.1 Pedestrian candidate generation based on position 55
4.2 Pedestrian candidate generation based on gradient 56
4.3 Bottom-up constructive candidate generation for pallets 58
4.4 Incorporating stereo information for candidate generation 61
4.5 Experimental results 63
4.6 Conclusions 64

Chapter 5 Contributions to object detection and classification methods **65**
5.1 Automatic image annotation by measuring compactness 65
5.1.1 Compactness definition and interpretation 65
5.1.2 Label transfer for image annotation 67
5.1.3 Algorithm overview 68
5.1.4 Experimental results 69
5.1.5 Conclusions 79
5.2 Detection methods for automated logistics operations 80
5.2.1 Pallet detection and position estimation 81
5.2.2 Methods for treating unloading operations 83
5.2.3 Experimental results 86
5.2.4 Conclusions 89
5.3 Pedestrian detection methods 90
5.3.1 Pedestrian detection using reduced number of scales 90
5.3.2 Pedestrian detection without image resize operations 94
5.3.3 Lazy Feature Extraction (LFE) 100
5.3.4 Multimodal Multiresolution Filtered Channels (MM-MRFC) 109
5.3.5 Conclusions 116

Chapter 6 Conclusions **117**

Bibliography **122**

Appendix A Published Papers **132**

A.1 In ISI rated international journals 132

A.2 In ISI indexed conference proceedings 132

A.3 In IEEE Xplore conference proceedings 133

A.4 Independent citations 134

Appendix B Listings of three papers **135**

List of Tables

3.1 Measuring exposure invariance properties of different descriptor types 45
3.2 Classification benchmark statistics 53
3.3 Classification benchmark results 54

4.1 Comparison of different candidate generation schemes. 63

5.1 Comparison using the same feature type 72
5.2 Compactness based annotation results using different feature types on Corel5k . 72
5.3 Comparison with state-of-the-art Corel5k 73
5.4 Sample annotations using color+DCT63+SIFT from Corel5k 73
5.5 The influence of parameters on F1 score 75
5.6 Compactness based annotation results using different feature types on IAPR-TC12 76
5.7 Comparison with state-of-the-art IAPR-TC12 76
5.8 Sample annotations using color+DCT63+SIFT from IAPR-TC12 76
5.9 Compactness based annotation results using different feature types on ESP-game 77
5.10 Comparison with state-of-the-art ESP-game 77
5.11 Sample annotations using SIFT from ESP-game 78
5.12 Composition of datasets acquired from Viano 87
5.13 Detection accuracy for multiple feature configurations 88
5.14 Detection accuracy for decision trees with different depths 88
5.15 Field test results from Viano 89
5.16 Typical execution times for each processing step 89
5.17 Comparison of execution times 92
5.18 RoI parameter tests on the training set 93
5.19 Relevant parameters of the detection algorithm 97
5.20 Area under the DET curve for different parameters on the Cluj Pedestrians dataset 99
5.21 Execution time of different modules 99
5.22 Execution times and log-average miss rate on Caltech. 108
5.23 Results on Caltech in % after introducing each feature channel 113
5.24 Score results on KITTI (in %) after introducing each feature channel 114
5.25 Comparison with state of the art on the KITTI test set - Average Precision in % 115
5.26 Average precision values on the Tsinghua-Daimler 115

List of Figures

2.1 Different visualizations of HOG features 15
2.2 Principle behind integral image sums 17
2.3 Pictorial representation of a decision stump 19
2.4 Illustration of the maximum margin found by a linear SVM 23
2.5 Demonstrating the qualities of SIFT features for object detection and recognition. 33
2.6 Standard euro pallet dimensions 34
2.7 Workflow of the Aggregated Channel Features pedestrian detection system. . . 37
2.8 Projection of a point onto imagers of two stereo cameras 38

3.1 Feature grid of 3 x 16 overlayed on an object (pallet) 43

4.1 The distribution of bounding box centers from the Caltech pedestrian benchmark 56
4.2 Illustrated steps from Algorithm 10. 58
4.3 Visualization of the horizontal guideline detection process. 59
4.4 Stereo-based candidate generation - sample result. 62

5.1 Sample matches and annotation from Corel5k 71
5.2 The influence of bag size - color+DCT63+SIFT 74
5.3 The influence of transfer type 74
5.4 Precision values for each concept and MAP on the NUS-WIDE dataset 78
5.5 Unloading operation for rack - visualization of processing steps 84
5.6 Unloading operation ground level - output for debugging 85
5.7 Unloading operation for block storage - visualization of processing steps 86
5.8 DET curve on the INRIA test set 93
5.9 DET cruves on pedestrian benchmarks 98
5.10 ROC curve and log average miss rate on INRIA 106
5.11 ROC curve and log average miss rate on INRIA 106
5.12 ROC curve and log average miss rate on Caltech-USA 107
5.13 ROC curve and log average miss rate on Caltech-USA 107
5.14 Log-plot of the average number of candidates after each cascade stage 108
5.15 Gathered data points for the mean ratio and fitted polynomials 112
5.16 Caltech comparison with state of the art 113

Chapter 1

Introduction

1.1 Definitions and context

Object detection is a fundamental task in computer vision. Its goal is to determine the position of different objects in input images or videos. This position is indicated by the bounding rectangle of the object or by a tight mask that represents the pixels corresponding to the objects. The difficulty lies in dealing with multiple possible appearances, ignoring irrelevant information such as noise in the image and having a low execution time.

Object detection often relies on or includes object classification and object recognition. **Object classification** aims to determine the correct class of an input object. The classification problem can be binary and in particular one can look for the presence or the absence of a specific object class (e.g. faces, people, bottles, etc.). If this problem is solved, a sliding window approach that checks every valid position can transform the classifier into a detector. This observation lies at the heart of most modern detection approaches which are essentially fast classifiers applied to multiple possible hypotheses.

In **object recognition** specific objects are recognized often from images containing objects from the same class. A typical example here is face recognition based on biometric features used to identify a specific user. In this case the system is required to pair a face to an individual. This task is more specific because it must go beyond simple detection and must perform identification also.

Candidate generation or **region of interest selection** refers to the procedure of providing a smaller input for detection methods than the original search space. Besides the obvious advantage of needing less time to check each position, it also involves a coarse level filtering of negative examples and has a benefic effect on classification. For practical systems such a module is essential in order to ensure fast execution and robust operation.

Object detection can be placed in the context of computer vision and it is related to other domains such as: image processing, pattern recognition, machine learning, robotics and artificial intelligence. In image processing important elements are: interest point detection, corner detection, blob detection, feature descriptors and segmentation. Pattern recognition enables the recognition of higher level features such as: lines, shapes, textures. Machine learning provides knowledge about choosing, training and evaluating powerful classifiers for the task at hand. Artificial intelligence develops classifiers that are based on human physiology such as neural networks.

1.2 Motivation

There is a vast number of applications where object detection is an essential component. All systems that rely on visual input for reasoning about the environment use object detection in some form. Since the current state of the art for most objects is well below human capabilities, the research in the field is active and important.

The automotive industry is focusing on making smarter cars. The improved cars are equipped with different sensors that enable them to reason about the environment. Information gathered can help the driver to avoid accidents. In the long term, many automotive manufacturers aim to develop a fully autonomous car. For such a vision to come true, it is required to have a permanent understanding of the environment.

Besides popular sensors, such as: laser-scanners, radars and other proximity sensors, stereo cameras are being built in many cars (Bosch). Stereo cameras are two cameras placed at a fixed known distance from each other. This configuration enables the estimation of distances and provides 3D reconstructed points of the visible scene. The sensors can help treat several high level tasks such as: lane assist, parking assist, pedestrian detection, obstacles detection, hanging object detection and traffic sign recognition.

Robots and any autonomous entities require object detection to navigate through the environment or to manipulate elements from it. Automated factories work with robots capable of assembling cars, inspecting products, transporting goods, inventory management and moving-line tracking. All of these activities rely on some sort of detection or classification.

Detection and classification are employed in multiple medical imaging applications. Detecting cancerous cells, malformations, illnesses, revealing internal structures hidden by the skin and bones, as well as diagnosing and treating diseases are the main goals. The input can be obtained from different imaging techniques including: microscopic images, X-ray radiography, magnetic resonance imaging, medical ultrasonography or ultrasound, endoscopy, elastography, tactile imaging, thermography, medical photography and nuclear medicine functional imaging techniques as positron emission tomography.

Video surveillance is useful for the following things: increasing security, monitoring and recording activities for various purposes, preventing loss of goods, offering facility protection, enhancing employee safety, deterring vandalism and illegal activities. Persons of interest, license plates, cars or other objects need to be detected and tracked in such applications.

Systems that are introduced in products are required to be robust to different environmental conditions. Vision systems must work in scenarios where the lighting changes quickly, where there is a large variance in illumination conditions, where the illumination conditions are poor and where light artifacts such as glare are present.

Time constraints are also important in detection systems. Performing in real-time is often a requirement that prohibits the development of complex reasoning algorithms. Engineers must find solutions that require a small execution time.

1.3 Thesis objective

The goal of this thesis is to analyze, propose, develop and evaluate approaches for object detection. The main focus is to produce fast and accurate methods by relying on contextual information, candidate generation and descriptive features.

In order to achieve this goal, we rely on the canonical object detection pipeline. However, several steps from the pipeline are customized and new approaches are proposed. The methods developed here should be well-suited for real-world applications. The research should be focused on solving specific problems in order to have direct applicability.

We propose descriptive features for specific object detection tasks. In real-world applications feature descriptors must be robust to changing environmental conditions and varying illumination. Features that are at least partially invariant to such changes are crucial for successful detection. The calculation of features has to be performed for each candidate. So it must be fast. This however, must not limit their descriptiveness.

To obtain a fast detection a good region of interest or candidate generation module is required. This step ensures that irrelevant zones from the image are discarded as fast as possible. More time consuming classification and reasoning can be done on a limited number of difficult cases.

The chosen goal is difficult in itself and by adding time constraints the problem becomes even harder. The qualities of the proposed object detection method can be summarized in the following bullet points:

- robust - provides detections even in suboptimal conditions;
- fast - has a low execution time;
- simple - the algorithms involved are easy to comprehend;
- accurate - the detector has high precision and recall.

1.4 Thesis structure

The present thesis contains six chapters. The introduction has introduced the reader to the domain of object detection. The next chapter describes related works from the technical literature. The presentation is grouped according to relevant topics. We progress from low level features to general object detection and then to complete algorithms for object detection for specific tasks.

Following the previous structure, the chapters three to five present the original theoretical and applicative contributions of this thesis. The topics are treated in a bottom-up manner starting from essential building blocks such as features and classifiers in Chapter 3, followed by proposed candidate generation methods described in Chapter 4. Afterwards, Chapter 5 contains three specific object detection systems. The last chapter contains conclusions regarding the findings from the performed research.

1.5 Acknowledgements

I would like to thank my advisor, Prof. Dr. Sergiu Nedevschi, for his guidance and for the opportunity to work in his research group. I am thankful for the possibility of taking part in relevant research projects. I also thank my colleagues from Lab D02, Lab 6 and Lab 37 for all their help and their support. I would like to thank my brother, Tamas, for proofreading the whole thesis and my family for encouraging me when it was most needed.

Chapter 2

Related works

2.1 Object detection overview

Object detection is one of the fundamental machine vision tasks. The research in this domain has produced numerous approaches. Even though much progress has been made, most detection systems are far from ready for real applications. An important challenge to overcome is providing robust results even when the input image is noisy. It is also difficult to produce fast algorithms that are capable of detection in real-time.

There are several applications where object detection is required. Medical applications such as detecting cancerous cells are of a tremendous importance. Automatic lane marking detection and traffic sign detection have been gradually introduced in modern cars. The scope is to prevent accidental lane changes. Face detection is now applied automatically on photos taken with most cameras to help tag people in the image.

Insights into object detection help us to understand the way our own brains interpret visual information. Biologically inspired system architectures have proven to give good results although at some point it is necessary to break away from imitation. The human visual system is intrinsically multilayered with each layer corresponding to a certain filter. The objects are recognized at the highest level.

Face detection is arguably the most researched area in this field. Common applications include: automatic photo tagging for social media applications and automatic focus for digital cameras. More specific recognition techniques can be used to identify criminals or to verify the identity of individuals automatically at passport checkpoints. The difficulty of the problem lies in covering different face orientations and treating partial occlusion due to glasses, hands or other objects.

Pedestrian detection is another extremely researched topic in this field. Increasing concern for pedestrian safety in the last year has resulted in the flourishing of pedestrian detection algorithms. These are essential in Advanced Driving Assistance Systems for preventing accidents involving pedestrians. Car companies are considering incorporating such systems into their models. For example, Volvo is planning to release cars that come with a pedestrian and cyclist detection module which will be able to stop the car automatically in case of an imminent collision.

Even though the topic of detecting pedestrians was tackled by many researchers, it remains largely unsolved due to several difficulties: the various visual appearance and varied clothing of pedestrians, different possible postures and articulations, crowded scenes where partial occlusion prevents detection and the large range of scales. The problem is still open to research, with systems that meet real-time requirements being especially difficult to develop.

2.2 Relevant feature descriptors

Image features are relevant information extracted from images for a specific task. They are essential for any high level task such as: detection, recognition and tracking. Features can be anything from a simple point to a large descriptor defined on a region. Image features are usually obtained through a process known as feature extraction. The positions where the features are extracted can be determined by using a feature detector which provides a list of interest points or by sampling densely from the whole image. The last approach is practiced with great success in modern machine vision approaches due to advances in computer hardware that makes this demanding approach possible.

Color or intensity at a position from the image is an example of probably the most simple and lowest level feature. More complex ones are calculated on an image region such as Histograms of Oriented Gradients. When an image region is described by a feature vector, we call such a vector a ”feature descriptor” of the region. Feature descriptors can be grouped into several categories. In the following section we provide a taxonomy and exemplify each category with approaches from the literature. Object detection methods often use custom features designed for the specific task.

2.2.1 Color descriptors and color spaces

Color can be viewed as the feature of the lowest level since it incorporates only local information. Even at this level there is a large complexity because of the multitude of available representation. Multiple color models can be used to represent colors. We enumerate some of the most common and useful color-spaces.

The standard RGB color model that is implemented in almost all modern display devices has its limitations including: the luminance and chromatic channels are not separated; it does not possess a channel that is invariant to illumination changes; does not accurately represent distances between real world color; the color channels are highly correlated; perceptual non-uniformity i.e. the distances in RGB do not conform to the perceived difference. Other color models correct some of these drawbacks.

Hue Saturation Intensity (HSI), Hue Saturation Luminance (HSL) and Hue Saturation Value (HSV) Joblove and Greenberg (1978) color-spaces separate color and intensity channels. They are also called phenomenal color-spaces because humans tend to organize colors by hue. HSI and others can be viewed as a cylindrical coordinate representations of the RGB color-space.

All color information is stored in a single channel called hue. This separation is why it is used in many image editing tools for color picking/selection.

The Lab color space is an opponent color space. The color opponent process is a color theory that states that human vision is a result of interpreting signals from rod and cone receptors in an antagonistic manner. The a and b channels represent the two antagonistic color channels or the signals from the two receptors. It was developed by CIE, the International Commission on Illumination - Commission Internationale de l'Eclairage - which is an organization devoted to international cooperation and exchange of information among its member countries on all matters relating to the science and art of lighting.

Essentially, the opponent channels can be obtained by subtracting two channels from the RGB model. A derived space called L*a*b* was designed to represent all perceivable colors because other models such as RGB fail to do so. One other major advantage is that relative perceptual differences between any two colors in L*a*b* can be approximated by treating each color as a point in a three-dimensional space and taking the Euclidean distance between them Jain (1989). Other opponent color models are described and evaluated in van de Sande et al. (2010) for illumination invariance properties designed specifically for object detection applications:

$$\begin{bmatrix} O_1 & O_2 & O_3 \end{bmatrix}^t = \begin{bmatrix} (R-G)/\sqrt{2} & (R+G-2B)/\sqrt{6} & (R+G+B)/\sqrt{3} \end{bmatrix}^t \tag{2.1}$$

Color information is often aggregated to provide better discriminating properties. Such aggregation can be done globally to produce for example a histogram of colors. This approximates the probability distribution of the image but loses relative spatial information between the features. To keep spatial information color descriptors can be extracted from image subregions of fixed size. The final descriptor is a concatenation of multiple values and thus preserves the inherent spatial structure of the image. A specific example would be to find the histogram of colors in all 8x8 regions from the image. An even more general approach is to perform segmentation and retain the mean colors of segments only. For example Selective Search from Uijlings et al. (2013) generates such descriptors in the process of providing object candidates. These segmentation-based descriptors are the most difficult and time consuming to calculate.

2.2.2 Texture descriptors

Texture descriptors describe the repetitive patterns present in the image. In the technical literature there is no formal definition given for what texture is. There are several approaches to extracting texture descriptors: statistical methods, geometrical methods, model-based methods and signal processing - see Penatti et al. (2012) for an enumeration.

Statistical methods estimate the statistics of regions. One can consider first order statistics resulting in a histogram or second order statistics resulting in the co-occurrence histogram. The last method considers pixel pairs and it is one the most popular texture descriptors.

Bag-of-words quantification is a method burrowed from the natural language processing domain (hence the name) - see Sivic and Zisserman (2009). Descriptor centroids are found via a clustering method (usually k-means) and all descriptors are replaced by the closest centroid to it. This is essentially a feature quantization process. Spatial pyramids of histograms of bag of words from Lazebnik et al. (2006) and Dunlop (2010) can be considered examples from this category.

Geometrical methods analyze texture by grouping smaller primitive elements. Several geometrical properties are taken into consideration such as: area, perimeter, aspect ration, thinness ratio and axis of elongation. This approach works well with artificial textures but has difficulty in representing natural textures because these are too chaotic and cannot be represented using primitives.

The idea behind model-based methods is to suppose that the texture is generated by an underlying model. Such a model would be dark and bright spots arranged in a certain configuration. Local binary patterns from Zhu and Wang (2012) , Census described in Zabih and Woodfill (1994) and Rank-transform are examples of such non-linear local transformation based on a texture model.

Texture can also be modeled by applying multiple filters on the image region and concatenating the responses. The filters can be either spatial domain of frequency domain. A collection of Gabor filters with different orientations and scales - see Zheng et al. (2004); the coefficients from the Fourier transform - see Zhou et al. (2001) and the Discrete Cosine Transform coefficients - see Ursani et al. (2007) - are a typical examples.

2.2.3 Shape descriptors

Shape descriptors describe the segments of the image. Image segmentation tries to group pixels together if they belong to the same object. It is a high level task that is easily performed by humans easily but computer vision algorithms still struggle in this area. One can describe the whole region of the shape or only the contour of the object.

The region as a whole can be described by several geometric properties such as: region area; center of mass coordinates; perimeter length; minimal bounding rectangle area; area of enclosing convex hull; aspect ratio; axis of least inertia; various circularity ratios; eccentricity and elongation and Euler number - the number of contiguous parts minus the number of holes.

The boundary of an object can be described by local descriptors. In most situations it is desirable to obtain contour representations that are invariant to translation, rotation, mirroring and articulation changes. Global descriptors consider the whole boundary to generate feature values, some approaches include: Elliptic Fourier descriptors from Kuhl and Giardina (1982), Contour Points Distribution Histogram proposed in Shu and Wu (2011) and invariant moments.

Hierarchical methods build up the object from parts and are learned from multiple examples. Local shape descriptors are the most descriptive because they associate a descriptor to each point from the contour. Shape Context from Belongie et al. (2000) describes the distribution of the other contour points around each point from the contour.

Inner Distance Shape Context proposed in Ling and Jacobs (2007) extends the previous approach by substituting the distance function with inner distance. This is defined to be the

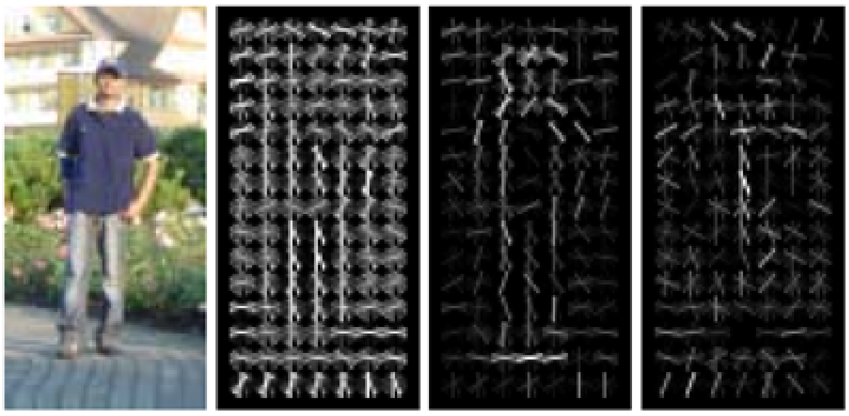


Figure 2.1: Different visualizations of HOG features

distance between two points while traveling only inside the object. Once the descriptors are extracted, different matching techniques can be applied. These range from simple distances such as: Euclidean or Manhattan distance to more complex matching costs such as: assignment cost minimization using dynamic programming; dynamic time warping and earth mover's distance. Multiple fast implementations are provided in Varga et al. (2012), and the methods are also tested on the MPEG7 benchmark.

2.2.4 Visual descriptors for object detection

For practical applications features must go beyond the gray intensity level and color components to provide more relevant information about the objects we are trying to detect. Good features are discriminative (they can help discern between objects and the background), reproducible (they are the same for the same objects even in the presence of noise), possess invariance properties (position, scale, rotation, affine transformation) and are easy to calculate (needed for fast detection). In the following we enumerate some of the more important features for general object detection.

One of the most useful feature types for pedestrian detection is the Histogram of Oriented Gradients (HOG) proposed by Dalal in the papers Dalal and Triggs (2005); Dalal et al. (2006). These features are constructed from histograms where each bin corresponds to an orientation and each pixel contributes to the bin of the gradient angle with a value proportional to the gradient magnitude. The histograms from cells are grouped in blocks and normalized - see Figure 2.1. This grouping in blocks preserves spatial distribution. Finally, all responses within the detection window are concatenated to form the full descriptor that will be fed to the classifier. Many of the best performing methods use this feature in conjunction with other information. HOG features have been extended to enable real-time computation of these features in Zhu et al. (2006) using the integral image optimization trick of Porikli (2005).

Integral images are cummulative sums of the input image in 2D. Their usage transforms intractable algorithms into simple lookups. Since we consider this a key element in implementing fast feature calculation, we provide algorithms descriptions required for cummulative sums. We describe two different methods for calculating integral images. Both work with images indexed from 0. We present the version where padding is performed after computation. In practice the output image J can be allocated and indexed with this padding operation in mind from the start. Algorithm 1 calculates cummulative sums along two directions and Algorithm 2 uses a 2D recursive update formula. The second algorithm leverages the fact that all previous values to the left and towards the top are already calculated. Both approaches can be performed in place. In practice it is often easier to work with a padded version of the integral image, where a stripe of zeros is added both to the bottom and to the left of the output. This avoids checking for boundary conditions and simplifies later calculations.

Algorithm 1 Calculate integral image - $integral(I)$

```
Input: an input image $I$
Output: an integral image $J$, with each entry $J(i,j) = \sum_{y=0}^{i} \sum_{x=0}^{j} I(y,x)$
1: $J = I$
2: for $i = 0 : I.height - 1$ do
3:     for $j = 1 : I.width - 1$ do
4:         $J(i,j) = J(i,j) + J(i,j-1)$
5:     end for
6: end for
7: for $i = 1 : I.height - 1$ do
8:     for $j = 0 : I.width - 1$ do
9:         $J(i,j) = J(i,j) + J(i-1,j)$
10:    end for
11: end for
12: performPadding(J)
13: return $J$
```

Algorithm 2 Calculate integral image - $integral(I)$

```
Input: an input image $I$
Output: an integral image $J$, with each entry $J(i,j) = \sum_{y=0}^{i} \sum_{x=0}^{j} I(y,x)$
1: $J = I$
2: for $i = 1 : I.height - 1$ do
3:     for $j = 1 : I.width - 1$ do
4:         $J(i,j) = J(i,j) + J(i-1,j) + J(i,j-1) - J(i-1,j-1)$
5:     end for
6: end for
7: performPadding(J)
8: return $J$
```

Having calculated the integral image, the sum of elements over a rectangular region can be evaluated using 4 lookups from the integral image. This is illustrated in Figure 2.2. The sum of the values in the shaded region is S and it is equal to the expression $A + C - B - D$, the values are retrieved from the indicated positions from the integral image.

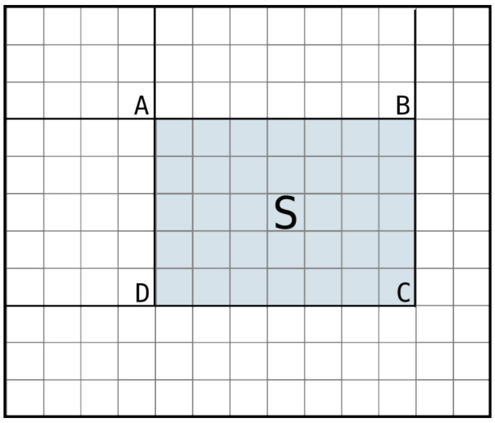


Figure 2.2: Principle behind integral image sums

Haar wavelets were introduced by Viola and Jones in seminal work Viola and Jones (2001a) for fast detection. These are weighted sums of rectangular areas from within the detection window. The authors employ integral images for fast calculation of the features, in fact it is one of the two key ingredients for their real-time method. Even though one can predefine such features by specifying manually a set of regions to calculate them, it is recommended to generate the regions randomly. By generating a large number of features one can apply an AdaBoost to select to most discriminating features automatically. This saves the developer the effort to find the best features and also ensures that none of the relevant feature configurations are missed if we generate a large number of configurations.

Integral Channel Features proposed in Dollar et al. (2009a) can be viewed as a generalization of the concept of Haar wavelets. They are defined on a general image channel. This channel can be an intensity image; a color channel; gradient magnitude; channel corresponding to a histogram orientation bin etc. First order integral channel features are simply sums of rectangular areas from these channels. The optimization with integral images enables extremely fast calculation of theses features in constant time. (Integral images are cumulative sums along both the dimensions of the original image intensity). Despite their simplicity, these features can be used to achieve state-of-the-art results as shown in Dollar et al. (2009b). In Dollár et al. (2010); Dollár et al. (2014) the authors present a fast detection method using these features and a scale correction method. The method relies on estimating the energy of the ratio of image channels at a slightly different scales:

$$E[f(I, s + ds)/f(I, s)] = e^{-\lambda ds} \tag{2.2}$$

Other features used to complement the previous ones are presented next. Even though simple color is not helpful for classification, relative color similarity between areas within the bounding box is a helpful feature. Color self-similarity described in Walk et al. (2010) features involve calculating histograms that encode second order statistics of colors. Motion cues are very helpful for detection when they are available. Works in this direction are: Viola et al. (2005a); Park et al. (2013).

2.3 Classifiers for object detection

Classifiers are explained in the simplest manner in the context of Supervised Learning. Supervised learning is a Machine Learning framework where learning is performed on a labeled dataset. In essence, the learning process is treated as showing a child examples from different classes. Several problems can be formulated as a supervised learning problem.

A labeled dataset consists of multiple examples or instances. The examples from the dataset are usually characterized by feature vectors. The feature vector resides in a multidimensional vector space $X \in R^d$. Each example has an associated class label that is referred to as the ground-truth. If C is the set of the possible classes and there are N samples then the labels form a set $Y = \{y_i \in C | i = 1 : N\}$.

The role of a classifier is to determine the class of an arbitrary input sample. The input sample is usually represented as a feature vector X. If C is the set of the possible classes then we can formally define a classifier as a function $f : R^d \rightarrow C$ that maps each input sample to a class. In the following we will present binary classifiers for which the number of output classes is 2, i.e. $|C| = 2$. Multiclass classifiers can be obtained by training multiple binary classifiers.

A good classifier must possess several properties: generalization property - the ability to learn the underlying pattern in the provided data and to make accurate predictions on previously unseen data; compact model - an intuitive and simple underlying model; the ability to make fast predictions.

The classifier can be viewed as a separation of the feature space in subspaces. The surface that separates the feature space into sets corresponding to the two classes is called the decision boundary. The simplest case is a binary decision boundary for a 2 dimensional feature space, which is a line dividing the plane in two parts.

In the learning framework the dataset is often split into to disjoint sets called the training and the set. The training set is used for learning purposes, the classifier model is obtained on this data. The test set is reserved for evaluation.

The learning algorithm for a classifier most often relies on writing the learning problem as an optimization problem. A cost function or error function is defined, and optimization techniques are applied to find the minimum.

Two main problems can occur during training phase, they are called high bias (overfitting) and high variance (underfitting) situations. A high bias classifier learns the training set very well but fails to generalize and produces a high test set error. The less formal term overfitting illustrates this by suggesting that classifier finds a decision boundary that is complicated and is fitted closely to the training data. High variance classifiers fail to learn the training data mostly because its model is too simple (for example, a line cannot always separate points into two classes).

The Vapnik-Chervonenkis dimension defined in Vapnik (2000) measures the power of a classifier. It is defined to be cardinality of the largest set of arbitrary points that the classifier can shatter. This essentially means that the classifier can learn to separate the classes in any possible configuration. The VC dimension of the perceptron is $D + 1$, where D is the dimension of the feature vector. This means, for example, that a line can separate any three points into any two classes.

Regularization is a technique applied for preventing overfitting and it entails penalizing the size of the parameters. Powerful classifiers have more parameters and by applying a penalty that is added to the cost function that is optimized we can enforce the learning algorithm to find smaller parameters. This is equivalent to eliminating some unnecessary parameters and thus reducing the discriminative power of the classifier but maintaining all options open in the beginning.

2.3.1 Decision stumps

Decision stumps can be viewed as the simplest type of classifiers. The decision rule is based on a condition imposed on a single feature. Regardless of the dimension of the feature vector a decision can be made in constant time $O(1)$. An example of such a classifier would be: if feature i is less than 5 return C_1 else return C_2. Simple classifiers such as this one play a crucial role in constructing powerful ensemble classifiers. In figure 2.3 the set of all examples is split into two disjoint subsets based on a single feature (hair length) and a fixed threshold *T*.

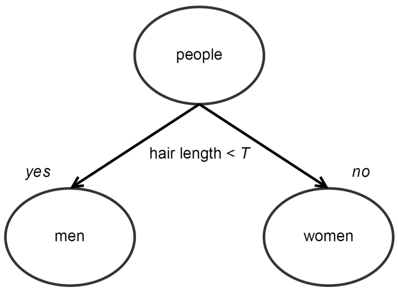


Figure 2.3: Pictorial representation of a decision stump

2.3.2 Decision trees

Decision trees (or classification trees) are a generalization of decision stumps - see Iba and Langley (1992). To make a decision multiple conditions are verified. This can be regarded as applying a series of rules. If one envisions the tree nodes containing the sets that correspond to the samples, then we have the following analogy: the root contains the set of all input samples; each time a rule is applied the set is divided into two child nodes; if the child node contains instances from a single class it becomes a leaf and the decision is readily made; otherwise the splitting is continued. Another analogy would be to consider leaves as class decisions, intermediary nodes as rules imposed on certain attributes and branches (tree edges) as the result of applying the rules. It is useful in practical implementations to limit the height of decision trees even though the leaf nodes do not contain instances from a single class.

The construction of a decision tree usually follows a top-down approach that is denoted Top-Down Induction of Decision Trees - see Quinlan (1986). Given a training set there can be multiple decision trees that correctly classify all instances. In the ID3 method from Quinlan (1986) it is assumed that the tree with the simplest form possesses the greatest generalization capability and thus it will work the best on a more general test set. The ID3 is an iterative construction process where a subsample (window) of the training set is selected and the simplest tree is found. If the tree correctly classifies all instances then the process finishes, otherwise the misclassified instances are added to the subset. The ideal attribute on which to perform a split is determined by maximizing the information gain. Unfortunately, this approach requires attributes to have only a finite number of possible values and so it is not applicable for unbounded real values, which is typically the case for image feature descriptors. Other approaches that rely on information gain include: C4.5 and C5 from Quinlan (1992).

An alternative measure to decide the attribute on which to perform the first split is to measure the Gini impurity. It is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. The CART (Classification and Regresstion Trees) method of Breiman et al. (1984) relies on Gini.

An important remark about the aforementioned methods is that they aim to find a decision tree that fully describes the dataset. Also, they aim to minimize the number of decisions to make in order to classify all the training instances correctly. In practice, recent detection methods often employ shallow binary decision trees in an ensemble classifiers. In this case other criteria are more useful for splitting the nodes. The optimal split is chosen to be the one with minimal weighted classification error.

2.3.3 K-nearest neighbor classifiers

Nearest neighbor classifiers - see Altman (1992) - make a decision about the class of the input sample by comparing it to the training set samples. The distance to each sample is evaluated and each training sample is ranked based on proximity. The first K neighbors (that are closest to the input sample) are then used to determine the output class. A k-NN classifier can be easily transformed into a regressor by returning the mean of a certain feature value instead of the class.

It is considered one of the simplest forms of classifiers and often used as baseline for other results. It is a type of instance-based learning or lazy learning since no model is generated during training but instead test instances are compared to the training instances at prediction time. Extensions to the standard algorithm have been developed, a typical example would be: to assign weights to each instance based on the distance; to use more complicated distance measures or to employ efficient data structures - such as k-d trees - to find the closest neighbors more quickly.

k-NN has some strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data) - see Cover and Hart (1967). For some values of k the method is guaranteed to approach the Bayes error rate.

2.3.4 Bayes classifiers

Bayes classifiers estimate the posterior probability density function of each class given the input features. It was introduced under a different name into the text retrieval community in the early 1960s in Russell and Norvig (1995). It can achieve competitive results if the input data is suitably preprocessed.

The posterior is obtained using the Bayes rule from the prior of each class and the likelihood. The likelihood is the conditional probability density function of the features given the class. During training we choose the classifier that minimizes the probability of misclassification (or risk). The equation for the posterior (the probability of class k given the features X) is given by:

$$p(C_k|X) = \frac{p(X|C_k)p(C_k)}{p(X)} \tag{2.3}$$

The Naive Bayes Classifier assumes that features are independent. This assumption enables the efficient calculation of the posterior as log sums of separate log likelihoods. The result is that such classifiers are highly scalable, requiring a number of parameters linear in the number of features. The simplification leads to the following equation:

$$p(C_k|X) \propto \prod_{i=1}^{d} p(X_i|C_k)p(C_k) \tag{2.4}$$

Choosing the class of an instance at prediction times involves calculating the posteriors and returning the index of the class with the highest posterior:

$$k^* = argmax_k p(C_k, X) \tag{2.5}$$

2.3.5 Linear classifiers

Weighing each feature with a different number and summing up the terms provides a score value. Thresholding the resulting score value can be a meaningful classifier. More formally, binary linear classifiers operate using the following decision rule: if $\theta^t X > \theta_0$ return C_1

else return C_2. If θ_0 is included in the parameter vector, and X is augmented with 1, this simplifies to the dot product condition: $\Theta^t\overline{X} > 0$. Learning can be performed using the variants of the perceptron algorithm. Overfitting can be avoided by employing regularization which penalizes large unnecessary coefficients from the weight vector θ.

Even if the examples are not linearly separable in the original feature space, progress can be made via feature transformation methods. In this case more features are generated such as powers of the original features. An equivalent and more efficient formulation is the kernel trick which does not require us to guess the necessary transformation. Instead, we only need to provide a kernel function which compares two instances.

2.3.6 Artificial Neural Networks

Artificial Neural Networks were proposed in 1943 by Warren McCulloch and Walter Pitts in McCulloch (1943). The authors proposed to mimic the neural network from the brain by using a simplified mathematical model they called ”threshold logic”. The network is built from simple units called the perceptron which have multiple inputs and a single output. The output is activated when a linear combination of the input surpasses a certain threshold. The power of the network comes from the cooperation of many interlinked perceptrons. A key development in this field was the introduction of the backpropagation algorithm by Werbos (1974) which allowed the training of multilayered networks that were able to learn more complex problems (such as the *xor* function).

2.3.7 Support Vector Machines

Support Vector Machines, proposed in Cortes and Vapnik (1995), are one of the most powerful classifiers. They have three main improvements over the simple linear classifiers. Firstly, during training the decision boundary with the maximum margin is selected. This ensures the clear separation of the examples and reduces the classification error when the classes are separable. Secondly, kernel methods employ kernel functions that enable comparison between features in a higher dimensional (possibly infinite dimensional) feature space. This eliminates the need for linear separability which does not always hold. Thirdly, the concept of soft margin permits some outliers by introducing a slackness term in the target function. In the following we guide the reader through each of these strengths by deriving the minimization problem that lies at the heart of SVM.

To illustrate the concept of margin Figure 2.4 is analyzed. Points from two classes are depicted, where class membership is equivalent to color. The classifier needs to separate the two classes by defining a separating line (or hyperplane in the general multidimensional case). This is defined by a linear equation:

$$w^T X = b \tag{2.6}$$

The weight vector w is the normal to the line and b is the intercept term, it is equal to the signed distance of the line from the origin. The margin is defined as the distance between the closest two points from separate classes along the direction perpendicular to the line. More technically, the margin can be defined using a dot product (projection along the normal vector):

$$m = \min_{i,j,y_i>0,y_j<0} w^T \cdot (X_i - X_j)/||w||. \tag{2.7}$$

The points lying on the edge of the margin are called support vectors. The goal of the classifier is to have a large margin since this ensures a robust classification. There is a ”neutral” zone is where classification is ambiguous. However, for the training set all examples are cleanly separated and are as far from the decision boundary as possible. Having a large margin is equivalent to the previous condition.

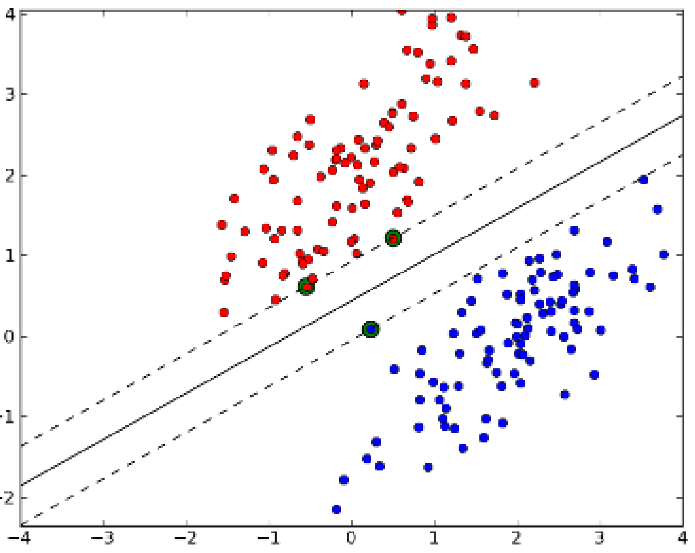


Figure 2.4: Illustration of the maximum margin found by a linear SVM

In order to have a large margin for the classifier, we need to ensure that all positive points are classified as larger than 1 and all negative points are classified as smaller than -1. The convention of choosing ± 1 simplifies calculations and does not affect the size of the margin as we shall see later. The conditions can be formulated compactly in the following equation:

$$y_i(w^T X_i - b) \geq 1 \tag{2.8}$$

where X_i is the i-th feature vector, w is the linear weight vector needed to be determined, b is the bias term (a scalar constant) and y_i is the class label. Class labels are either +1 or -1. In this configuration the size of the margin is $2/||w||$. The problem of having the largest margin can be formulated as a minimization problem:

Minimize $||w||$ with respect to w and b (i.e. maximize the margin $2/||w||$) subject to constraints from 2.8.

An equivalent form considers the square of the norm, thus transforming the problem into a quadratic optimization problem. Using Karush-Kuhn-Tucker multipliers, which generalize Lagrange multipliers, to incorporate the constraints into a single optimization function, we can formulate the following form (primal form):

$$w^* = \arg\min_{w,b} \max_{\alpha \geq 0} \left(\frac{1}{2} w^T w + \sum_i \alpha_i (y_i (w^T X_i - b) - 1) \right) \tag{2.9}$$

The dual form can be obtained by writing the weight vector as a linear combination of the training instances $w = \sum_i \alpha_i X_i y_i$. This is implied by the stationary Karush-Kuhn-Tucker condition. After substitution and arrangements we have:

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j X_i^T X_j \tag{2.10}$$

subject to constraints $a_i \geq 0$ and $\sum_i \alpha_i y_i = 0$.

In 1995, Corinna Cortes and Vladimir N. Vapnik introduced the soft-margin formulation for SVMs. They have received the 2008 ACM Paris Kanellakis Award for their contribution. The soft margin formulation allows for a few misclassified examples by introducing slack variables ξ_i and penalizing such mistakes. The tolerance to such outliers is controlled by the cost parameters C in a linear fashion. In this case the optimization problem has the following form:

$$w^* = \arg\min_{w,\xi,b} \left(\frac{1}{2} w^T w + C \sum_i \xi_i \right) \tag{2.11}$$

subject to constraints:

$$y_i (w^T X_i - b) \geq 1 - \xi_i \tag{2.12}$$

Note that the constraints have been modified to allow for certain instances to have a classification score that violates the original hard margin constraints. In such cases $\xi_i > 0$. If we penalize mistakes by letting $C \to \infty$ then this forces all $\xi_i = 0$ and we obtain the original optimization problem with hard margin. Choosing $C = 0$ does not make sense, since it implies that ξ_i can take on any value to satisfy the constraints and no progress is made.

The dual form of the previous problem can be stated as:

$$L(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j X_i^T X_j \tag{2.13}$$

subject to constraints $0 \leq a_i \leq C$ and $\sum_i \alpha_i y_i = 0$. Because of the linear penalty function, the slack variables vanish and only the constraints on the α_i change.

The last issue that needs to be addressed is linear separability. What happens if the training data cannot be separated with a simple line? The normal approach would be to extend the feature space by forming other features. A typical example would include higher order polynomials of the features. As this can lead to a large number of features and possibly does not solve the problem, a better solution is needed. Kernel methods resolve this by changing the dot product from the dual form to a general kernel function:

$$L_{kernel}(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(X_i, X_j) \tag{2.14}$$

This avoids the introduction of higher order features and essentially makes the trip to the other feature space and back without requiring the explicit transformation. Also, this opens the possibility of having an infinite dimensional feature space if we use the Gaussian kernel (Radial Basis Function):

$$k(X_i, X_j) = exp(-\gamma ||X_i - X_j||^2) \tag{2.15}$$

We end the description by providing a short list of implementations for SVM. Probably the most used and recommended is the LIBSVM by Chang and Lin (2011) free library available in multiple programming languages. SVM is included also in Matlab, Weka, SVMlight and other environments/libraries. The most common optimization technique used to solve the quadratic programming problem is the Sequential Minimal Optimization algorithm (SMO) shown in Platt et al. (1999). The algorithm breaks this problem into a series of the smallest possible sub-problems, which are then solved analytically.

2.3.8 Boosted classifiers

Ensemble classifiers discussed in Schapire (1990) rely on combining multiple simple classifiers to obtain a robust decision. The simple classifiers are called weak learners and are typically fast and cheap to evaluate. The outputs of the weak learners are combined to make a final classification. AdaBoost is an example of a training method that helps to find multiple weak learners so that their combination results in a low classification error.

AdaBoost stands for Adaptive Boosting. It is a training algorithm for ensemble classifiers. It was shown by Feund and Shapire in Schapire (1990) that several weak learners can be just as powerful as a strong classifier when combined together. The authors have received the Godel Prize in 2003 for their work, indicating the importance of their discovery.

In the following we provide an intuitive description which will be followed by a formal presentation of the theory and pseudocode. Intuitively, AdaBoost weighs each training example and changes the distribution of these weights according to which example was successfully classified. The error measure will be the weighted training error which takes into account these value. If an example was misclassified, its weight will be increased in the next stage to guide the classifier into learning that specific example.

We now turn to a more formal description. Let the combined score of the ensemble classifier at stage T be the linear combination of the scores of the weak learners:

$$H_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{2.16}$$

Each weak learner can return the values $+1$ or -1. Our goal is to learn the classifiers h_t and to find the coefficients α_t that minimize the error function:

$$E_t = \sum_{i=1}^{N} e^{-y_i H_t(x_i)} \tag{2.17}$$

This exponential loss function has a high value in the cases of mismatches between the sign of the prediction given by $H_t(x_i)$ and the true class label y_i. By adopting the weighting function $w_i^{(t)}$ to be 1 if $t = 1$ and $e^{-y_i H_{t-1}(x_i)}$ for other values of t the error can be expressed as:

$$E_t = \sum_{i=1}^{N} e^{-y_i H_{t-1}(x_i)} e^{-y_i \alpha_t h_t(x_i)} = \sum_{i=1}^{N} w_i^{(t)} e^{-y_i \alpha_t h_t(x_i)} \tag{2.18}$$

which results from the original definition of the error and is obtained by cutting off the sum from the expression of H_t at $t-1$ and separating the first part which is equal to $w_i^{(t)}$. The error can be rearranged to obtain:

$$E_t = \sum_{i=1}^{N} w_i^{(t)} e^{-\alpha_t} + \sum_{y(i) \neq h_t(x_i)} w_i^{(t)} (e^{\alpha_t} - e^{-\alpha_t}) \tag{2.19}$$

We wish to choose the best weak classifier at step t to minimize this error. The expression attains the minimal value when the second term is minimal since the first term is independent of our choice for h_t. The term $e^{\alpha_t} - e^{-\alpha_t}$ is dependent only on α_t and not on h_t. So the optimal weak classifier at step t must minimize the weighted training error $\sum_{y(i) \neq h_t(x_i)} w_i^{(t)}$. We now turn to determining the weight coefficient α_t. For this we use another form for the error function where we group incorrect and correct predictions in two terms:

$$E_t = \sum_{y(i) \neq h_t(x_i)} w_i^{(t)} e^{\alpha_t} + \sum_{y(i) = h_t(x_i)} w_i^{(t)} e^{-\alpha_t} \tag{2.20}$$

The minimum can be found analytically by taking the partial derivative and equating it with 0:

$$\frac{\partial E}{\partial \alpha_t} = \sum_{y(i) \neq h_t(x_i)} w_i^{(t)} e^{\alpha_t} - \sum_{y(i) = h_t(x_i)} w_i^{(t)} e^{-\alpha_t} = \epsilon_t e^{\alpha_t} - (1 - \epsilon_t) e^{-\alpha_t} = 0 \tag{2.21}$$

Solving for α_t by multiplying by e^{α_t} and taking the logarithm we obtain:

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t} \tag{2.22}$$

where ϵ_t is the weighted training error: $\sum_{y_i \neq h_t(x_i)} w_i^{(t)}$. So at each iteration we pick the classifier with the lowest weighted training error, calculate its weight α_t and add it to the ensemble classifier.

2.3.9 Classifier evaluation measures

We define a dataset as a set containing N samples, each sample consists of a feature vector X_i, which is vector from a D dimensional real vector space and sample label y_i which is from the set of admissible class labels C. The classifier that is tested is denoted by $f(\cdot)$.

In the following definitions we use the Iverson bracket convention for summations which entails that logical expressions surrounded by $[expr]$ evaluate to 1 if true and to 0 if false. This enables the succinct and exact representation of the sums. The training error is defined as (Note, all sums are taken over the all the possible values of i):

$$E = \frac{1}{N} \sum_i [f(X_i) \neq y_i] \tag{2.23}$$

The accuracy is its complementary measure:

$$A = \frac{1}{N} \sum_i [f(X_i) = y_i] = 1 - E \tag{2.24}$$

For cases when the distribution of the instances in the classes is skewed, the standard error measure does not provide a meaningful measure of the error. For example, on a dataset with instances 10000 instances from c_1 and 10 instances from c_2 a classifier that always returns c_1 achieves a low error rate of $\frac{10}{10010} \approx 1e-3$. More relevant measures in this case are the weighted error rate, precision and recall.

The weighted training error considers weights for each training instance. Classes with less instances can be given larger weights, or difficult examples can also receive a larger importance. If the weights are stored in a vector w, the weighted error has the following form:

$$E_w = \frac{1}{\sum_i (w_i)} \sum_i [f(X_i) \neq y_i] w_i \tag{2.25}$$

Precision for class c is defined as the percentage of correctly classified instances from the class c:

$$P_c = \frac{\sum_i [f(X_i) = y_i][f(X_i) = c]}{\sum_i [f(X_i) = c]} \tag{2.26}$$

This is the same as the number of true positives divided by the sum of true positives and false positives.

Recall for class c is defined as the percentage of correctly recalled instances from the class c:

$$R_c = \frac{\sum_i [f(X_i) = y_i][f(X_i) = c]}{\sum_i [y_i = c]} \tag{2.27}$$

This is the same as the number of true positives divided by the sum of true positives and false negatives.

It can be easily observed that good performance is ensured by both a high recall and precision. In practice, there is a trade-off between these two values. Opting for a more precise classifier often results in missing some instances. Requiring the recall of as many examples as possible will likely lead to classification mistakes.

The F_1 measure or score is the harmonic mean of the precision and recall. It is a special case of the general F_β measure:

$$F_\beta = (1 + \beta^2)\frac{P_c \cdot R_c}{\beta^2 P_c + R_c} \tag{2.28}$$

It was derived so that F_β would measure ”the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision” - from Rijsbergen (1979). It is useful because it provides a single scalar value for given precision and recall values. This simplifies the comparison and ranking of classifiers.

Cohen’s kappa coefficient is a statistical measure that can also be applied for evaluating classifier performance. It measures the agreement of the ground-truth labeling and that of the classifier. It is generally thought to be a more robust measure than simple percent agreement calculation since κ takes into account the agreement occurring by chance. The value of the coefficient is given by:

$$\kappa = \frac{Q(a) - Q(e)}{1 - Q(e)} \tag{2.29}$$

where $Q(a)$ is the probability of agreement and is equal to the accuracy value, and $Q(e)$ is the probability of chance agreement. For the case of a classifier and ground-truth data it is given by:

$$Q(e) = \frac{1}{N^2}\sum_c \sum_i [f(X_i) = c] \sum_i [y_i = c] \tag{2.30}$$

The Reciever Operating Characteristics (ROC) curve is the curve traced by plotting the true positive rate (sensitivity or recall) versus the false positive rate (fallout or 1-specificity). Different values for precision and recall are obtained when the discrimination threshold of a binary classifier is varied. Picking a point from the curve is equivalent to selecting a threshold for discrimination. This can be done by considering the requirements of the application, in some situations it may be useful to have a high recall while in other cases a high precision is desired. The area under the curve (AUC) can be used as an effective measure of the classifier’s performance. Several similar curves exist, such as the Detection Error Tradeoff (DET) curve which plots the error rate versus the miss rate.

The confusion matrix (also known as contingency table or error matrix) is a straightforward and useful way to organize the mistakes committed by a classifier. It is a table of $|C|^2$ entries, each row corresponding to a predicted class and each column to the true class. Each position is essentially a bin from a histogram, which numbers the times each class c_1 was predicted while having the true class label c_2 (possibly $c_1 = c_2$). Of course the larger the values on the main diagonal the matrix are, the higher the accuracy is. Precision and recall values can be readily calculated for each class from the confusion matrix by performing row- or column-wise summation and scaling.

2.3.10 Classifier evaluation methods

It is easy to see that evaluating classifiers is not so straightforward. Typically, classifier minimize an error measure on a training set but simply looking at this number is often misleading, unless the training set is sufficiently large. The main reason is that classifiers should possess the ability to generalize and should not memorize the input data. Several methods have been devised to perform a more accurate performance evaluation.

Resubstitution is a technique used for testing where the classifier is trained and tested on the whole data. If the training data is sufficiently large, and the classifier is properly regularized, this can be a first alternative to evaluate the classifier. However, it is not recommended in practice because the classifier can simply memorize the correct class labels with no generalization taking place.

One of the most objective evaluation forms is to test the classifier on a new dataset that contains instances not found in the training dataset. Of course the test dataset should be similar to the training set and should be a considerable subsample from the universe of all possible instances on which the classifier should work. The situation of having a separate test set and training set is often not an option because of the lack of data.

Data Partitioning (or holdout) is a process by which starting from a dataset we obtain two disjoint sets for training and testing respectively. Usually, the portion for learning is larger but this also means that the evaluation will not be so accurate on the small number of test samples. It is recommended to train a model on the full set after the performance is evaluated in order to use the available data in its integrity.

Cross-validation is one of the most effective evaluation methods. It combines multiple data partitionings to get a better estimate on the measure of generalization of the classifier. Define an n-fold cross-validation by dividing the available data in n equal parts. After this n separate training sessions are performed by leaving out each of the n parts respectively. The learned model is evaluated on the part left out. The resulting score measures are averaged or aggregated in some other manner to find a global score. Optionally the full data is fed into the training procedure as the last step after evaluation.

By performing a cross-validation with the number of folds equal to the number of available examples we obtain a leave-one-out evaluation strategy (or Jackknife). Since this process entails training the classifier a large number of times, if no model update strategy exists, it is very time consuming. However, the advantage is that all samples are tested and there is no selection bias (which samples should be used for learning and which for evaluation).

2.4 Region of interest selection and candidate generation methods

A region of interest signifies a zone from the image that will be processed. It is distinguished as a separate step in detection methods in the survey by Gerónimo et al. (2010) showing its importance. The rest of the image is considered to have no relevant information for the current task. Many applications have a well defined region of interest such as a simple rectangular area of the image. In other situations we may consider to establish the region of interest dynamically. This has many advantages but also proposes several challenges. The main advantage is that by reducing the area that is processed we obtain a lower execution time. In case of object detection we also eliminate false positives that would have been detected in the zones outside the region of interest. In this section and later on we will use the terms region of interest selection, candidate generation and object proposal to refer to the same process.

The challenge is to design an algorithm that is able to perform a region of interest selection dynamically. Also, the shape of the region must be general and not limited to a rectangular region. Sliding window object detectors work on a search space that consists of multiple positions and bounding box heights and widths. This is a 4 dimensional space. One common way to reduce the size of this space is the consider a fixed aspect ratio (0.43-0.5 for pedestrians). In this case the search space is reduced to 3 dimensions: position x,y and scale (box size). Region of interest selection should be applied directly on this space instead of image space. This is essentially equivalent to reducing the total of candidates by quickly classifying several of them as negatives.

In order to understand the importance of reducing the search space, we provide a quantitative analysis of the expected number of candidates for an input image. Consider an image of size A, we aim to search at every position on a fixed grid with step size of d, and we employ m scales each time resizing the image by a factor of $\lambda < 1$. The resulting number of candidates for this baseline setup is given by:

$$C = \sum_{k=0}^{m-1} A \frac{1}{d^2} \lambda^k = \frac{A}{d^2} \frac{1-\lambda^m}{1-\lambda} \tag{2.31}$$

Typical applications use at least VGA (640 x 480) resolution input image. Plugging in typical values for the parameters $A = 480 \cdot 640 = 307200$, $d = 2$, $m = 32$, $\lambda = 2^{-\frac{1}{8}} \approx 0.91700$ results in a number of 867512 candidates. A modern computer can execute roughly 10^8 operations per second. This means that even if every candidate is classified by performing a single operation it would require 8 milliseconds for the whole image. Of course, feature extraction and classification needs to be performed for each candidate so this time is a very low underestimate.

Based on the discussion presented before, we highlight several important characteristics of a good region of interest selection method:

- it must have a high recall - It must retain all the important regions in the image. Eliminating any region in this step will prevent detection afterwards.
- it must have a high rejection rate - It must reject the majority of the candidates. The property provides the key advantage of speeding up later processing steps.
- it must be sufficiently fast - Its insertion in the processing pipeline must not increase the running time. Otherwise, it is not worth adding a supplementary processing module.
- it should be robust to slight variations - This entails producing the same results in the presence of slight noise.

Good region of interest (RoI) selection methods can reduce the execution time of detection methods significantly because they eliminate unnecessary calls to the classifier. A survey by Gerónimo et al. (2010) presents several approaches under the paragraph of Foreground segmentation. Most of the methods make use of stereo information to detect good candidate regions such as: Benenson et al. (2012); Labayrade et al. (2002); Nedevschi et al. (2009). Monocular approaches are fewer in number and include: biologically inspired attentional algorithms from Itti et al. (1998); Serre and Poggio (2010), vertical symmetry detection from infrared images of Broggi et al. (2005) and segmentation algorithms. Simple and efficient region of interest selection methods using only monocular information are hard to find or nonexistent.

Another important category of methods provides object candidates or object proposals. These methods work on general objects since it has been shown that reliable features exist that can discern general objects from the background, and that these features are independent to object classes. The survey written by Hosang et al. (2014) presents several methods from this category. Each of the presented approaches is evaluated in terms of recall (the percentage of relevant objects successfully recalled), repeatability (same results when noise is applied to images) and detection rate when used with a detector.

The objectness measure proposed by Endres et al. in Endres and Hoiem (2010, 2014) relies on multiple cues to provide good object proposals as fast as possible. They employ color, texture histogram intersection, boundary strength and several layout agreement features. They formulate the problem as a structured learning problem and adopt a slack-rescaled method with loss penalty to solve it.

The authors in Cheng et al. (2014) develop a fast method to provide object candidates based on the gradient signature of the resized image. Their system is trained on features obtained from resized images to a fixed 8x8. The gradient magnitude is binarized and an SVM is trained. For faster prediction the SVM prediction function is replaced by a binary approximation which transforms the linear combination into additions and subtractions.

Selective Search from Uijlings et al. (2013) tackles the problem by performing a hierarchical clustering to obtain image segments that could represent objects. In Arbeláez et al.

(2014) graph cut is applied to the segmentation problem. Another promising line of research is considering boxes that have few lines intersecting their boundaries, Edge Boxes of Zitnick and Dollár (2014). It was shown that minimizing the lines that cross the object boundary is a relevant measure for candidate generation. Also, this approach is constructive, meaning that it build up candidate bounding boxes.

2.5 Object detection methods

Object detection in computer vision refers to the task of detecting the correct position of a target object in a given input image. The result of the detection process is information about the object pose. The simplest form of the pose is the position of the object but it can incorporate other information such as: orientation, scale, minimal enclosing bounding box and object parts.

Object detection is difficult because the appearance of the objects in the images is highly variable. This stems from multiple sources. Firstly, the image process itself introduces variability due to illumination changes, digitization artifacts and camera position changes. Secondly, the object itself may have several different appearances especially for general objects such as persons. Successful detectors must learn the properties of the objects that are present most of the time regardless of pose (invariants).

Two main categories of object detection can be distinguished: generative methods and discriminative methods. Generative methods (Amit (2002); Felzenszwalb (2001); Fergus et al. (2003)) construct a probability model for object pose and background. They rely on estimating the conditional probability of the objects being present given certain features. On the other hand, discriminative methods (Rowley et al. (1998); Viola and Jones (2001a,b); Felzenszwalb et al. (2010)) learn to discern between positive and negative examples and consequently require negative examples to perform training.

Detection can be posed as a *binary classification* problem by defining a positive class for the presence of the object and a negative class for the absence of it (background). One can perform detection by checking every position and every scale for positive instances. This approach is typical and it is referred to as sliding window detection.

Object detection methods have a wide range of applications in a variety of areas including robotics, medical image analysis, surveillance and human computer interaction. Current methods work reasonably well in constrained domains but are quite sensitive to clutter and occlusion.

A popular benchmark for object detection is the PASCAL VOC object detection challenge, presented inEveringham et al. (2010). More recently, large scale datasets have appeared such as LabelMe - Russell et al. (2008) and ImageNet - Deng et al. (2009). The goal of object detection benchmarks is to fuel the competition for best object detectors for common categories such as people, cars, horses and tables in photographs. The challenge has attracted significant attention in the computer vision community over the last few years, and the performance of the best systems have been steadily increasing by a significant amount on a yearly basis.

A closely related concept is that of *object recognition* where the detected object is compared to existing templates to find a match (such as in a biometric security system). Two main types of approaches can be discerned.

Template-based recognition methods compare the content of a region to a template or a family of templates. This treats objects as a whole and can also be called holistic. We enumerate a few approaches from this category. Some methods from this category are: Chamfer distance matching performed on edges and distance transform of Enzweiler and Gavrila (2009); Divide-and-conquer methods divide the search space recursively in order to eliminate zones that are not promising, as shown in Lampert (2010); The method presented in Torralba et al. (2007) compares intensity values directly to templates using a flexible distance function; Methods based on histogram representations from Linde and Lindeberg (2004); Varga and Nedevschi (2013b); Bag-of-words transformation methods proposed in Li and Perona (2005); Lazebnik et al. (2006).

A more bottom-up approach would be to construct objects from parts by first detecting important features in the image and combining them. Some of the most important techniques for such detecting methods are: Scale Invariant Feature Transform of Lowe (1999) (SIFT) ensure scale and rotation invariance and are suitable for matching objects even under large viewpoint changes (see Figure 2.5 - the features of the train are recognized even when the object is rotated and partially occluded, figure from Lowe (1999)); SURF features, proposed by Bay et al. (2006), are a variant of SIFT features where the keypoint positions are found much faster; Hough transform from Duda and Hart (1972) recognizes shapes and curves that have a parametric form by accumulating votes for each parameter configuration; Methods that rely on geometric consistency verification impose restrictions on the relative positions of features to enforce correct object structure; Random Sample Consensus (RANSAC) from Bolles and Fischler (1980) is a robust way of fitting a model onto data points that contain outliers.

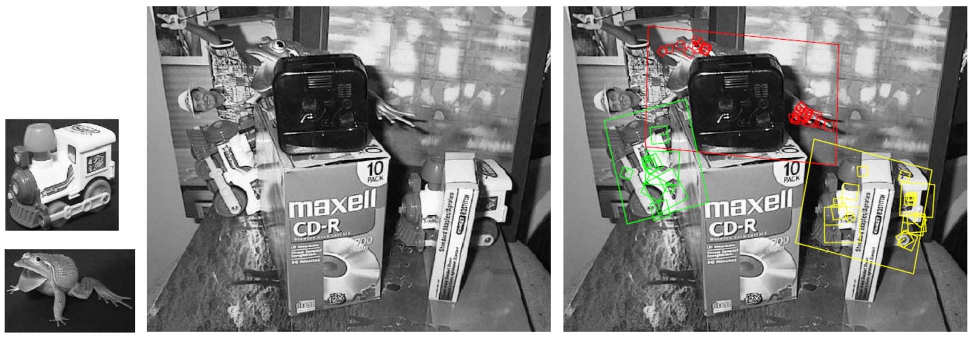


Figure 2.5: Demonstrating the qualities of SIFT features for object detection and recognition.

2.6 Pallet detection methods

Pallet detection is a special case of object detection. In this section we present existing approaches to this problem because a part of our work is in this field. In this case the detection needs to be reliable and precise.

Automated Guided Vehicles (AGV) perform logistic operations without requiring human intervention. This necessitates the existence of a sensor capable of estimating the position of the pallet that needs to be loaded by the machine. A machine vision-based detection system for pallets can handle multiple tasks and help the AGV reason about its environment.

Stereo cameras offer a good solution for 3D sensing applications. The cost of such systems is lower compared to laser scanners. Also camera systems offer a larger field of view and the possibility of high level reasoning on data. The main drawback of such systems is the difficulty of working in poor and rapidly changing illumination conditions.

Pallets are wooden supports designed to hold goods and are easily graspable by the forklift because of its pockets (see Figure 2.6). Pallets are standardized and for our purposes they are handled from only one side. We desire a flexible detection module that can identify the relative position of the pallet from any image under various lighting conditions.

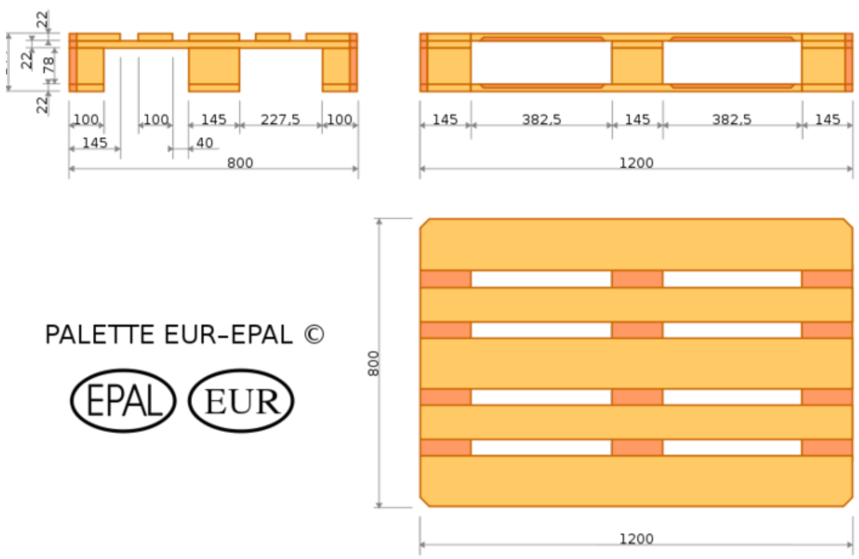


Figure 2.6: Standard euro pallet dimensions

Approaches for autonomous load handling from scientific and technical community use different types of sensors to obtain an understanding about the environment. We will group these approaches into two main categories: vision-based (monocular or stereo cameras) and 2D or 3D time-of-flight Laser Range Finder (LRF). We start by describing the approaches from the second category.

Walter et al. (2010) presents a method for autonomous manipulation of a priori unknown palletized cargo with a robotic lift truck. The sensor involved in detection is a horizontal LIDAR.

Operating on the noisy points from the sensor, an algorithm is applied to perform closest edge detection.

Sky-Trax System offers a solution to detect the presence of pallets on the forklift. The system uses an ultrasonic sensor that uses sound waves to detect objects in its range. It has a broad sensing area, is compact, durable, accurate and inexpensive. Pepplerl-Fuchs also offers ultrasonic sensors for solving the same task.

SICK industries manufacture laser scanners for multiple purposes. A paper from the National Institute of Standards and Technology - Bostelman et al. (2006) - presents a pallet detection method based on LADAR (laser detection and ranging) using SICK S3000 laser scanners. This has the advantage over cameras that it is able to operate in complete darkness and invariant lighting. They also tackle unloading operation for trucks using the same sensor. Hough transform from Hough (1962); Duda and Hart (1972) is applied to detect lines that represent walls and other boundaries from the gathered laser points.

The system designed in Baglivo et al. (2011) combines two sensors, a laser scanner and a camera to localize the pallet given some prior knowledge with large uncertainty. The pallet is detected from the color image acquired from the camera and the points provided by the laser. The vision part uses edge template matching and distance transform. Both sources must agree on the detection in order for it to be considered valid. This approach suffers from the following disadvantages: the calibration between the laser scanner and camera; edge information is not reliable; laser scanner only offers information along scan lines. The authors have evaluated their system on 300 examples with results indicating a good localization precision. They have found difficulties due to lighting conditions in 5 cases.

The paper from Pradalier et al. (2008) describes a vision-based system functioning outdoors consisting of an autonomous hot metal carrier. The system from Seelinger and Yoder (2006) uses easily identifiable features (landmarks, fiducials) of the form of concentric circles for easy registration. This however, requires the labeling of all pallets with such features. The success rate they obtained from 100 operations is 98%. The work of Cucchiara et al. (2000) makes use of corner features, region growing and decision trees. Least squares line fitting and a single camera is employed in Byun and Kim (2008). Other approaches include: Kim et al. (2001) line-based model matching is used; Pages et al. (2011) Colour-based segmentation; Nygårds et al. (2000) sheet-of-light range camera.

Detection approaches for pallets can draw inspiration from other specific object detection methods such as pedestrian detectors. We turn our attention to such approaches in the next section.

2.7 Pedestrian detection methods

This section provides references to important contributions for pedestrian detection. For a comprehensive review on the subject, the reader is advised to consult reviews and surveys such as Dollár et al. (2012), Enzweiler and Gavrila (2009), Gerónimo et al. (2010).

The availability of benchmarks is crucial for any field in order to enable an objective comparison of existing algorithms. Pedestrian detection is no exception. Luckily, there are several high quality benchmarks available for research purposes. The INRIA dataset, presented in Dalal and Triggs (2005), is one of the first pedestrian detection benchmarks. The Caltech dataset from Dollar et al. (2009b) contains approximately 1000 images with thoroughly annotated pedestrian bounding boxes. Other dasaets for reporting results include: TUD Wojek et al. (2009); ETH Ess et al. (2008); Daimler Enzweiler and Gavrila (2009).

Pioneering work in pedestrian detection was done by Dalal and Triggs Dalal and Triggs (2005) which presented Histograms of Oriented Gradients as features. The features were specifically tailored for pedestrian detection. They are one of the first features to be used for pedestrian detection. Its importance is demonstrated by the fact that almost every modern method uses gradient histogram features in some form.

Viola and Jones in Viola and Jones (2001a) introduced a lightning fast approach for object detection by combining Haar features and cascaded classifiers. Although the method presented there is applied for face detection, it is universal and can be applied to any form of object detection. Haar features are differences of sums calculated on rectangular areas from the intensity image. Cascading enables the fast rejection of many candidates and it is the main reason for the high speed of the method. The approach from this seminal work is extended and perfected in many state-of-the-art algorithms.

Dollar et al. have produced numerous detection algorithms based on integral channel features - see Dollar et al. (2009a); Dollár et al. (2010), which are related to the concept of Haar features. In this case features are sums of pixel values of custom image channels on a rectangular area. In Dollár et al. (2012) the authors speed up detection by suppressing and enhancing detection scores based on spatial and scale proximity. In Dollár et al. (2010) the authors present a method for approximating feature values at different scales without recomputing the actual features. Since feature calculation at different scales is a bottleneck for most algorithms, this improves the speed considerably. Integral channel features are employed on randomly generated rectangles inside the pedestrian bounding box. In a more recent work from Dollár et al. (2014) the authors show that simple square features with a 4x4 support region are enough to achieve state-of-the-art performance and this also enables fast detection at 32 frames per second (see Figure 2.7 from Dollár et al. (2014)).

Benenson et al. have developed top performing detection methods both in regard to speed and detection accuracy by modifying the thresholds for their classifiers to account for scale differences - see Benenson et al. (2012). In Benenson et al. (2013) they perform a comprehensive analysis of the effect of different parameters such as: feature pool size, feature normalization, number of weak learners etc. on detection performance. Franken classifiers are suggested in Mathias et al. (2013) for a better treatment of occlusion. More recent research suggests filtered

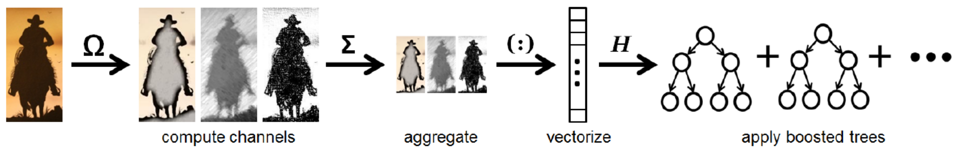


Figure 2.7: Workflow of the Aggregated Channel Features pedestrian detection system.

channel features with certain patterns such as checkerboard or square improve performance, see Zhang et al. (2015).

Non-holistic approaches of Felzenszwalb et al. (2010); Wojek and Schiele (2008) use deformable part models where the part locations are considered latent variables and the optimization problem can be solved by Latent-SVM. Deep neural networks, used in Luo et al. (2014), can automatically learn hierarchical feature representations that correspond to body parts and their configuration. These approaches provide state-of-the-art performance on recent object detection challenges. Their key advantage is that they are capable of learning feature representations and thus do not require manual design in this manner.

Other methods that rely on new discriminative features have been developed for the pedestrian detection task. Sketch tokens are introduced as contour-based features which are learned from human sketches Lim et al. (2013). In Costea and Nedevschi (2014) Costea et al. show that bag-of-words features are descriptive enough for the pedestrian detection task. Informed Haar features proposed in Zhang et al. (2014) make use of common sense knowledge to manually design a pool of rectangular features. Spatially pooled features from covariance matrix features and Local Binary Patterns were successfully utilized in Paisitkriangkrai et al. (2014b,a) along with a classifier that is specifically targeted at optimizing the area under the ROC curve in the desired interval.

The employment of additional information such as stereo depth Marín et al. (2013), optical flow Walk et al. (2010); Park et al. (2013), bounding box context Yan et al. (2013), person-to-person patterns Ouyang and Wang (2013) and extended training set Nam et al. (2014) can further improve the detection rate. Benenson et al. in Benenson et al. (2014) combine many of the previous techniques to form a powerful detector.

Recently, more and more object detection approaches rely on deep convolutional neural networks. In pedestrian detection improvements have been registered through this method. In Cai et al. (2015) the authors use multiple features ranging from simple to complex but push back the calculation of the more time consuming features to later cascade stages where only a handful of candidates are left. A fast implementation for detection is possible even with neural networks as shown in Angelova et al. (2015).

2.8 Stereo matching and reconstruction

Stereo vision refers to a 3D understanding of the environment that is usually achieved through two cameras. Human vision is of stereo type due to the fact that it employs input from the two eyes separated by a lateral displacement. In a similar manner, two cameras placed in the same configuration can provide sufficient information to deduce the depth of the objects. Points from the real world that lie on the same line passing through the optical center get projected to the same pixel position. Because of this, one camera is not sufficient enough to infer depth from the image.

By adopting the simplification that all light rays pass through a single point (called pinhole camera model), the depth can be calculated if the camera parameters are known. The required intrinsic parameters are the focal distance f, camera principal point and distortion coefficients. The extrinsic parameters come in the form of the translation vector and the rotation matrix for each camera or a relative translation and rotation between the two cameras. In a canonical configuration the relative translation vector between the two cameras has non-zero component only along the x axis, and the cameras are aligned to face the same direction (the rotation matrix is close to the identity).

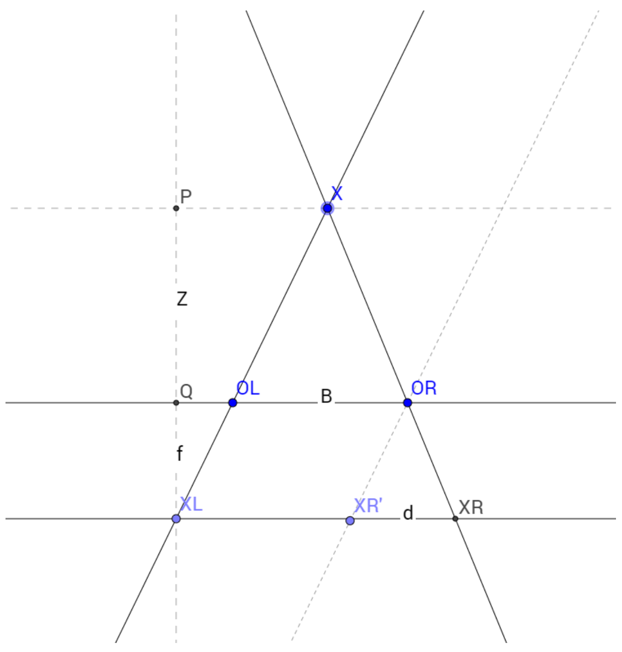


Figure 2.8: Projection of a point onto imagers of two stereo cameras

Figure 2.8 shows an illustration of two cameras, and a point projected from the real world unto the imaging surfaces of each camera. We follow standard conventions for defining axis and notation. Here we discuss a planar case where the height component (y-axis) is ignored. The same reasoning can be generalized along that axis also. The difference along the x axis between the coordinates of the same point projected onto the two images is denoted as the disparity

($d = x_R - x'_R$). We will show that the disparity value is inversely proportional to the distance to the object.

An exact formula can be obtained by studying the similar triangles XO_RO_R and $O_RX'_LX_R$, where X is the point that is being projected; O_L and O_R are the optical centers of the left and right cameras, respectively; X_L and X_R are the projections onto the left and right image, respectively; X'_L is the intersection of the line passing through O_R that is parallel to XO_L with the image of the right camera. We write a similarity relation between the ratios of the heights and bases of the two triangles:

$$\frac{Z}{f} = \frac{B}{x_R - x_L} = \frac{B}{d} \tag{2.32}$$

where Z is the distance from the cameras along the z axis, f is the focal length and B denotes the baseline (horizontal displacement between the two cameras); x_L and x_R are the coordinates in pixels of the projected point. From here we can express the distance based on the other known quantities and arrive at a very important formula:

$$Z = \frac{B \cdot f}{d} \tag{2.33}$$

Note that in the previous formula we have supposed that the disparity value d is known to us. However, in order to determine the correct value we need to find pixel correspondences between the left and right image pixels. This is known as the correspondence problem and it is difficult to solve because of: some points may not be visible in the other image (occlusion); textureless or repetitive areas can be easily mismatched and searching can be very time consuming.

The canonical configuration along with the process of image undistortion provides two input images where the epipolar lines corresponding to the same point are horizontal. Epipolar lines are the intersection of the plane passing through the point and the two optic centers with the imager. If the cameras are not in canonical configuration, image rectification can be performed which reprojects the point to obtain two images that correspond to a canonical configuration. This is achieved by knowing the precise relative rotation and translation between the two cameras.

The extrinsic parameters of the camera are the rotation matrix R and the translation vector T. If these have been obtained from a calibration procedure for the left and right camera in R_l, T_l, and R_r, T_r respectively, the rectification procedure can be performed. First, the baseline vector is found as:

$$\vec{B} = \vec{T_r} - \vec{T_l} \tag{2.34}$$

Then a possible form for the rectification rotation matrix is given by:

$$R_{rect} = [B, R_l(:,3) \times B, B \times (R_l(:,3) \times B)] \tag{2.35}$$

where we have calculated the columns of the matrix by taking the baseline vector, the cross product between the third column of the left rotation matrix (z-axis), and the third orthogonal component which is the cross product of the previous vectors. This aligns the cameras so that the x-axis becomes parallel to the baseline and the new y-axis becomes normal to the old z-axis and the new x-axis.

We have established that in order to obtain depth information for a point we have to determine its disparity value. Stereo matching algorithms produce a disparity map that contains disparity values for each position (dense) or for certain points or zones only (sparse). According to an important work from Scharstein and Szeliski (2002), the major steps of each stereo algorithm can be given by: matching cost computation; cost (support) aggregation; disparity computation/optimization and disparity refinement.

Matching cost computation is the first step and it is the operation at the lowest level. Its aim is to calculate the initial cost volume (disparity space) for each possible position and valid disparity value. The approaches from the literature rely on different dissimilarity measures between pixels such as: absolute difference, squared difference, sampling insensitive absolute difference Birchfield and Tomasi (1998), normalized cross-correlation, binary features defined on edges, sign of the Laplacian, gradient-based measures and non-parametric measures (Census or Rank Transform) Zabih and Woodfill (1994).

Since matching a single pixel is unreliable, cost aggregation is necessary. By assuming that neighboring positions have similar disparity values, local costs can be replaced by aggregated versions. In the simplest form aggregation can be performed by summing or averaging over a neighborhood of the original cost. More complicated forms include convolution with a filter (such as a 2D or 3D Gaussian); choosing the aggregation window size adaptively; iterative diffusion.

Disparity calculation can take into account only raw cost values. In this case the method is local. Such approaches are prone to noise but can be implemented in situations where execution time is critical. In contrast, global methods try to minimize a certain global cost function defined on the whole disparity map. Most methods from this category rely on the techniques of belief propagation such as Felzenszwalb and Huttenlocher (2006) and graph cuts of Kolmogorov and Zabih (2001). In the middle ground, constraints can be enforced on stripes or only along certain directions. In this case we are talking about semi-global methods which lie between the local and global ones both in terms of speed and smoothness. Notable examples in this category are the ones that rely scanline disparity optimization such as Kim et al. (2005) and Semi-Global Matching introduced by Hirschmuller (2005).

The last stage aims at correcting small mistakes such as: occluded pixels, non-confident reconstruction and small salient patches elimination (which are probably errors). In this stage subpixel interpolation can also be performed which reestimates disparity values with higher precision, a step that is necessary for certain applications where small disparity changes can lead to large errors.

Chapter 3

Proposed features and classifiers for object detection

Starting from the current chapter, we present the original theoretical and applicative contributions of this thesis. The topics are treated in a bottom-up manner, beginning with the essential building blocks like the underlying features and the classifier. These are a common parts for the detection systems that follow. Candidate generation approaches are presented afterwards. We end the description with three different detection systems. Each section contains relevant experimental validation as well as conclusions specific to the topic that is treated.

3.1 Normalized Pair Differences and other features

3.1.1 Normalized Pair Differences

The paper from Varga and Nedevschi (2016) describes our proposed features for object detection. Our aim was to develop a feature vector that is defined on grayscale images which is sufficiently descriptive and also robust to illumination changes. This is a requirement for systems working in conditions where frequent exposure changes are made during image acquisition. This may be due to changing lightning conditions or dynamic view changes.

We consider here a simplified image formation model that assumes pixel values are proportional to the exposure time. We intend to define features that are invariant to exposure changes. According to our crude model for changing the exposure image intensity values are modified linearly, i.e. the resulting image *J* is equivalent to the original one *I*, multiplied by a constant:

$$J = \alpha \cdot I \tag{3.1}$$

This approximation holds also for gain changes if we ignore saturating effects. Weber's law states that the human response is proportional to the relative change stimuli and not to the absolute magnitude. Inspired by this we develop a feature that reflects the importance of relative change. If differences between intensity features are normalized by reference values, then the importance of changes is determined according to their relative magnitude to the reference values.

Let $I = \{I_i, i = 1, n\}$ be a set of n grayscale intensity values from an image region, i.e. the intensity values in row-major order from the image matrix. This region is cropped from the whole image and usually represents a possible position of an object. Since the area of the region can be large in practice, we wish to perform a resizing operation in order to subsample the values. Another important point is that most classifiers work with fixed size feature vectors. So all bounding boxes should be resized to a same standard size. We define the normalized pair difference between features I_i and I_j as:

$$npd(i,j) = \frac{I_i - I_j}{I_i} \tag{3.2}$$

This feature contains relevant information about the relative change between the intensities I_i and I_j. Firstly, the sign shows which of them is higher. Secondly, the normalization ensures a response that is relative to the feature magnitude. The possible number of features is of order $O(n^2)$ if all possible pairs are calculated. In most cases some region of the input rectangle can be discarded.

We can easily verify that a property of this definition is invariance under multiplication with a constant:

$$\frac{\alpha \cdot I_i - \alpha \cdot I_j}{\alpha \cdot I_i} = \frac{I_i - I_j}{I_i} \tag{3.3}$$

This property will ensure the robustness of the descriptor against camera exposure time changes. In practice it is recommended to limit the values of the features. Since the fraction can have large values, we apply a sigmoid type function to confine the features in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

$$npd(i,j) = tan^{-1}\left(\frac{I_i - I_j}{I_i + \epsilon}\right) \tag{3.4}$$

The constant ϵ is a small number that helps avoid division by zero. This formulation is practical because it avoids additional checks for the magnitude of I_i. The inverse tangent function can be replaced by any sigmoid-type function but should be included because it has a beneficial effect.

To describe the whole region we select all possible intensity pairs I_i and I_j and calculate npd for each of them. The concatenation of the obtained values will be considered as the descriptor of the region. This results in a $n(n-1)/2$ dimensional feature vector which is large even for small size patches.

We have also considered an alternative version of this descriptor. We can choose an intensity value as a reference and compare all other ones to it. This results in a descriptor that is of size $n - 1$ only but is less descriptive and prone to noise in the reference intensity value. The reference value may be changed to be the mean of the intensity of the region.

We illustrate the concepts defined above in the context of pallet detection. Figure 3.1 shows a pallet with an overlayed grid. Each cell in the grid will become one single intensity value I_i. This essentially means that the crop from the image will be resized to the given grid size. Note that certain zones from the image may not be of interest. These cells are marked with red in the figure. We eliminate them by providing a binary mask and do not consider values from those regions.

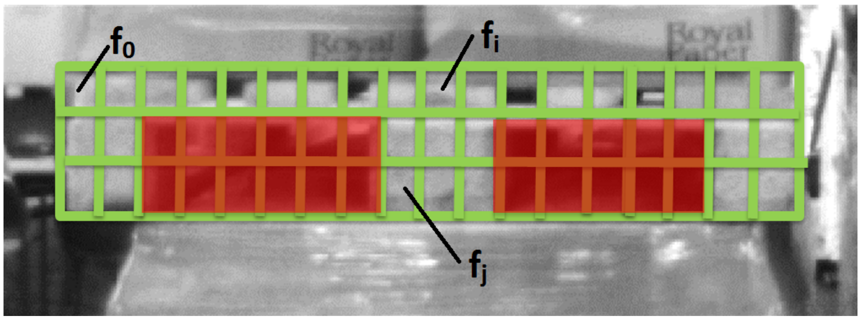


Figure 3.1: Feature grid of 3 x 16 overlayed on an object (pallet)

Algorithm 3 Calculate Normalized Pair Differences features - $npd(I)$

Input: an input image I (or a subregion of it), mask for eliminating regions
Output: an array with feature values F

```
 1: $I = resize(I, w, h)$
 2: allocate $F$, $k = 0$
 3: for $i = 1 : w \cdot h$ do
 4:     if $mask(i) = true$ then
 5:         for $j = i + 1 : w \cdot h$ do
 6:             if $mask(j) = true$ then
 7:                 $k = k + 1$
 8:                 $F(k) = atan(\frac{I(i)-I(j)}{I(i)+\epsilon})$
 9:             end if
10:         end for
11:     end if
12: end for
13: return $F$
```

Algorithm 3 describes the exact procedure for calculating **npd** features. The algorithm depends on the target window width and height, denoted by w and h respectively. First, the image is resized to the target dimension. Then each pair that has both elements in the valid zone is considered. The time complexity of this algorithm is $O((wh)^2)$, i.e. quadratic in the number of elements considered. If the mask eliminates an α proportion of the regions from the image this can be lowered to $O((\alpha wh)^2)$.

3.1.2 Sparse Local Binary Pattern Histogram

Local Binary Pattern (**lbp**) features have been successfully used as features for texture description. For a given region the histogram of **lbp** values offers a descriptor that is invariant to the illumination changes. In our detection methods we employ such a histogram and calculate it using a slightly modified version of the original algorithm. We provide this in the following for completeness of description.

Algorithm 4 Calculate sparse LBP histogram - $lpbHist(I)$

```
Input: an input image $I$ (or a subregion of it), sparsity parameter $s$
Output: an array with feature values $H$
1: $H = [0] * 256$
2: for $i = 1 : s : I.height$ do
3:     for $j = 1 : s : I.width$ do
4:         $descr = lbp(I, i, j)$
5:         $H[descr] + +$
6:     end for
7: end for
8: $H = H/sum(H)$
9: return $H$
```

Algorithm 3 describes a slightly modified **lbp** histogram calculation procedure. We introduce a parameter s that controls the density at which we sample the **lbp** values. This small trick reduces the time complexity from $O(I.width \cdot I.height)$ to $O(I.width \cdot I.height/s^2)$, i.e. provides a quadratic speedup factor. This is important to consider since the input region can have a large size. The **lbp** values for each pixel position are precalculated once for the entire image. We have found that this approach with on-demand histogram calculation is faster than constructing integral images for each possible **lbp** value. This would require 256 integral images which results in high memory consumption. In our experiments we have found that the subsampling does not affect the form of the histogram in a significant manner. Because of this we set $s = 5$ for typical cases.

3.1.3 Edge density features

Another important feature we use to measure edge strength is the density of the gradient inside a rectangular area. Calculating density entails normalising by the size of the region. This is helpful because it creates a descriptor that does not depend on the size of the input image.

Algorithm 5 Calculate edge density along boundary regions - $edgeDensity(I)$

```
Input: an input image $I$ (or a subregion of it), regions $R_k$, precalculated integral image for the
    gradient values $IE$
Output: an array with feature values $E$
1: for $k = 1 : nrregions$ do
2:     $E(k) = IE(R_k.bottom, R_k.right) + IE(R_k.top, R_k.left)$
3:     $E(k) = E(k) - IE(R_k.bottom, R_k.left) - IE(R_k.top, R_k.right)$
4:     $E(k) = E(k)/area(R_k)$
5: end for
6: return $E$
```

Algorithm 5 presents the processing steps for the edge density features. The two key ingredients are calculating regional sums using an integral image called IE and the normalization by the size of the area. In most cases these features are evaluated at multiple bounding box locations relative to the central object. The time complexity is $O(K)$, where K is the number of regions to consider.

3.1.4 Experimental Results

We have performed tests in order to evaluate the invariance properties of the features we use. A sequence containing 317 measurements is recorded of a static pallet with varying exposure time. The change in exposure time modifies the appearance of the pallet from barely visible to saturated white. Descriptors are extracted from the same region. We evaluate the mean and the maximum of the standard deviations of each component. The Euclidean distance is also calculated between each descriptor pair and the mean and the maximum is found. We divide by the feature dimension for a fair comparison. All feature values are normalized to be in the range [-1, 1]. Table 3.1 shows the results, entries are ordered from top to bottom from least invariant to most invariant (we show only values for differences).

Feature	**dim.**	**mean std. dev**	**max std. dev.**	**mean diff.**	**max diff.**
intensity	53	2.29e-01	3.18e-01	3.78e-02	1.09e-01
edge	53	6.25e-02	3.38e-01	2.44e-02	4.23e-02
npd	1327	5.34e-02	4.08e-01	2.74e-03	6.11e-03
lbp	257	1.18e-03	5.86e-02	2.93e-04	8.43e-04

Table 3.1: Measuring exposure invariance properties of different descriptor types

The table includes intensity features as a baseline. The **npd** features have similar properties as the **lbp** histogram but they are more descriptive and structure information is maintained.

By looking at the mean and maximum differences we can state that **npd** features change less under the tested conditions compared to the intensity and edge features.

3.1.5 Conclusions

The current chapter described features which can be efficiently computed. Such features that are required for successful and fast detection procedure. All of the presented proposals and improvements can be calculated efficiently. Normalized Pair Differences create a signature for the object that is robust to illumination changes. The Sparse LBP Histogram constructs the traditional LBP Histogram on a subset of data points for faster execution. Edge density features defined on custom regions can be evaluated efficiently and provide relevant information for detecting boundaries. The impact on the classification/detection performance will be studied in later sections. The work from this part has been published in Varga and Nedevschi (2016).

3.2 Boosted ensemble classifiers with decision trees as weak learners

This section presents contributions for implementing boosted classifiers with decision trees efficiently. We highlight how our proposals are different from existing implementations. We also derive the computational complexity for each function and evaluate the classifier on machine learning benchmarks. The true test to the correctness of the implementation results from the success of the detection methods presented later on.

Research in object detection has shown that ensemble classifiers are one of the most a powerful classifiers. In most cases they are implemented with decision trees as weak learners. The depth of the trees is limited to 2 in most pedestrian detection methods. Training is performed with Discrete AdaBoost or other similar variants. Several state-of-the-art pedestrian detection methods employ such a classifier, e.g. the works from Benenson et al. (2014); Dollár et al. (2014). The main advantage is that it has a very fast prediction function at almost no cost for detection accuracy.

There are several implementations for AdaBoost available: Weka (Java), jBoost (Java), scikit-learn (python), mlpack (C++), MultiBoost (C++), Piotr's toolbox Dollár (2006) (Matlab/C++). Of these the closest to our needs is the one by Dollar. It was employed for training a pedestrian detection classifier and it is self-contained. We highlight the important differences between our proposal and their implementation: ours is a pure C++ implementation; we perform no feature quantization and obtain the same speed; no feature subsampling is done. The other libraries are either implemented in a slow interpreted language or do not have functions for training/predicting with the powerful combination of AdaBoost + Decision Trees. For instance, MultiBoost has only decision stumps implemented. In general it is hard to find an implementation that is includable in a C++ project and is lightweight.

Even though the prediction phase is simple and efficient, a naive implementation of training several decision trees on a large dataset is unfeasible. Each decision tree has to select the best error threshold for each feature. In the following we describe our proposed implementation that optimized running speed and memory consumption. We describe our proposed modifications to the standard algorithm that yields a faster code.

Let us denote the number of features by D and the number of training instances by N. One could try using each feature value as a threshold and then check the training error in each case. This would lead to a $O(D \cdot N^2)$ algorithm which is highly inefficient and is already intractable for a dataset of order $1e5$. By first performing a sorting and then selecting thresholds between two consecutive feature values, we achieve an $O(D \cdot N \cdot log(N))$ time algorithm. This can be further improved by sorting only once at the startup and saving the ordering. Some approaches (such as Marín et al. (2013)) optimize the execution time by randomly subsampling the features and thus reducing the value of D.

To summarize, our proposed speed-up consists in remembering the rank of each instance for each feature. At the splitting phase these ranks are propagated to the children. To show that this is correct we rely on the fact that reweighting the instances does not affect the order of the feature values. This leads to an $O(D \cdot N)$ algorithm after an initial sorting step, which is done only once in the beginning.

The fast implementation by Dollar in Dollár (2006) in Matlab/C++ achieves a low execution time by employing feature quantization and then by calculating the cummulative distribution function of the histogram to achieve the same time complexity as our proposed method. In contrast, we do not quantize the feature values and obtain the same speed. For some classification tasks, preserving the original unquantized feature values is proven to beneficial.

3.2.1 Decision tree algorithms

We start a more detailed description of the classifier functions by showing the prediction function for decision trees in Algorithm 6. The other more complex parts can be built around this basic function. The required input is an array of features denoted by X. The decision tree is traversed from root towards a leaf based on decisions at each node. If the feature value stored at the node's index is smaller than the threshold value stored in the node, then we descend to the left child, otherwise to the right child. When a leaf node is reached, the label of the node gives the prediction result.

The time complexity of this algorithm is given by $O(maxdepth)$ since it requires the traversal of the decision tree of maximal depth $maxdepth$, which is independent of the size of the feature vector. Only comparisons and assignment operations are required. If the node label value is set later, even multiplication operations are avoided. The is possible because labels are of the form $\pm\alpha_t$ for each tree. The space complexity of the prediction algorithm is $O(1)$ because it is independent of the size of the feature vector or the depth of the tree.

The binary tree is stored as an array by indexing the nodes in preorder. No linking through pointers is required because the left and right child of each node i is located at positions $2i + 1$ and $2i + 2$ respectively. All index values are required to be less than the feature dimension D.

Algorithm 6 Decision tree predict - $predictTree(X)$

```
Input: decision tree model and input vector $X$
Output: decision tree prediction
1: $node = root$
2: while node has children do
3:     if $X[node.index] < node.threshold$ then
4:         $node = node.leftchild$
5:     else
6:         $node = node.rightchild$
7:     end if
8: end while
9: return $node.label$
```

Given this prediction mechanism we can now describe the training procedure to obtain such a tree structure. The objective that is minimized during the training phase is the weighted classification training error. This is the sum of the weights of the misclassified instances from the training set:

$$E_w = \sum_{predictTree(X_i)Y_i<0} w_i \tag{3.5}$$

Since the weights of the instances will be changed, the objective function will be minimized by different trees at each stage. The training procedure is described in the following Algorithm 7. It is recursive and it starts from the root of the tree. The parameters $indexes$ contains all indexes at first and $depth = 0$. The procedure stops when the maximum depth of the tree is reached. Another terminating conditions is if the distribution of the instances is disproportionate. In this case always predicting the same class label would result in a low error rate and this is chosen.

In preparation we find the sum of the weights of the positive examples s^+ and sum of the weight of the negative examples s^-. Next, each feature is checked to find the error for splitting at some threshold (lines 10-35). Since the order of the indexes when the instances are sorted based on each feature is available in the matrix oi, we need to perform a linear pass only. During each pass we can find the error for classifying all instances smaller than the current feature value as negative instances in err^-. Since the instances are ordered, the misclassified examples up the current position have the value v^+ and the other mistakes above the threshold add up to $s^- - v^-$. A similar reasoning holds for err^+. The minimal error is retained along with the label and threshold for which it was obtained. The global error and field values are chosen based on each feature.

Lines 36-43 construct the list of instances that are placed in each subtree. Based on these lists and the oi ordered indexes matrix, the new oi_{left} matrix is generated by preserving the order but only using the indexes from the $left$ list. A similar procedure is used for the right subtree. Recursive calls are made using the constructed data and reusing the training set X and Y. This is important because for a large training size of the memory is consumed by X and Y and we must avoid unnecessary copying. The label of the child nodes is set by the parent based on the best error found.

The execution time of the algorithm is dominated by the error minimization procedure which is clearly $O(D \cdot N)$ in time. Since at each depth the same number of operations or less are performed, we have the total time complexity of $O(md \cdot D \cdot N)$, where md is maximum allowed depth of the decision tree. The space complexity is given by $O(N \cdot D)$ required for storing the ordered indexes for the subtrees after splitting. For training sets having less than $2^{16} = 65536$ - which is common - two bytes (short datatype) are sufficient for each index which saves space.

Parallelization of the training procedure is possible since the outer loop for finding the best threshold value can be performed independently. In this case the optimal error values and configurations are saved in an array of size $O(D)$. Executing calls on the subtrees on different threads is not efficient since tree sizes can be unbalanced.

Algorithm 7 Train Decision Tree - $trainTree(X, Y, w, oi, indexes, depth)$

```
Input: feature vectors $X$, label vector $Y$, weight for each instance $w$, ordered indexes $oi$, selected/active instances, depth in the tree
Output: decision tree as a binary tree with each node containing pointers to children, index for splitting, threshold for splitting and label
 1: $n^+ = \sum_{Y(i)>0} 1$
 2: $n^- = \sum_{Y(i)<0} 1$
 3: if $depth == maxdepth$ OR $n^+ < th$ OR $n^- < th$ then
 4:     $node.threshold = \infty$, $node.index = 0$,
 5:     $node.leftchild = node.rightchild = 0$, $node.label = sgn(\sum w)$
 6:     return
 7: end if
 8: $s^+ = \sum_{Y(i)>0} w(i)$
 9: $s^- = \sum_{Y(i)<0} w(i)$
10: for $i = 1 : D$ do
11:     $v^+ = v^- = 0$
12:     for $j = 1 : N - 1$ do
13:         if $Y(oi(i,j)) > 0$ then
14:             $v^+ = v^+ + w(oi(i,j))$
15:         else
16:             $v^- = v^- + w(oi(i,j))$
17:         end if
18:         $err^- = v^+ + s^- - v^-$
19:         $err^+ = v^- + s^+ - v^+$
20:         $err = min(err^+, err^-)$
21:         if $err < err*$ then
22:             $err* = err$
23:             if $err^- < err^+$ then
24:                 $label = -1$
25:             else
26:                 $label = 1$
27:             end if
28:             $threshold = (X(oi(i,j),j) + X(oi(i,j+1),j))/2$
29:         end if
30:     end for
31:     if $err* < global$ then
32:         $global = err*$
33:         update node fields with saved $label$, $index = j$ and $threshold$
34:     end if
```

```
35: end for
36: $left = [], right = []$
37: for $j = 1 : N$ do
38:     if $X(j, index) < threshold$ then
39:         $left.append(j)$
40:     else
41:         $right.append(j)$
42:     end if
43: end for
44: split $oi$ in $oi_{left}$ and $oi_{right}$ based on $left$ and $right$
45: call $trainTree(X, Y, w, oi_{left}, left, depth + 1)$
46: $node.leftchild.label = label$
47: call $trainTree(X, Y, w, oi_{right}, right, depth + 1)$
48: $node.rightchild.label = -label$
```

3.2.2 Ensemble classifier algorithms

Training of the ensemble classifier using AdaBoost follows the standard algorithm with a few adjustments (Algorithm 8). We begin by sorting each feature column and retaining the permutation required for ordering. This data structure is needed for the subsequent training phase. As stated before, this step requires $O(D \cdot N \cdot log(N))$ but it is done only once and it provides a large speedup for later steps. For each stage we find the optimal decision tree using Algorithm 7. The optimal tree depends on the weight distribution which is uniform in the beginning but changes in function of the misclassified examples. The weights are modified with a multiplicative factor, scaling up the weights of the examples classified wrongly.

Another addition to the standard method is that we change the label of the decision tree. This avoids multiplying at prediction time with c_m by setting the labels of the decision tree leaves from ± 1 to $\pm c_m$. Such small changes impact the speed because the prediction operations are performed millions of times. Obtaining an error larger than 0.5 should not happen and it signals an unusual situation.

Algorithm 8 Train ensemble classifier (Discrete AdaBoost) - $train(X, Y)$

```
Input: feature values for all training instances as $X \in R_{NxD}$ matrix and class label for each
    instance as $Y \in \{-1, 1\}^N$ vector
Output: boosted classifier with lowest error rate
1: for all feature index i do
2:     order column vector $X(:, i)$
3:     save permutation required for ordering in $oi(i, :)$
4: end for
5: initialize weight vector $w(:) = 1/N$
6: for t=1:$nrweak$ do
```

```
 7:     $err_t = trainTree(X, w, oi)$
 8:     **if** $err_t > 0.5$ **then**
 9:         break
10:     **end if**
11:     $c_m = 0.5log(\frac{1-err_t}{err_t})$
12:     $tree_t.setLabel(c_m)$
13:     **for** i=1:N **do**
14:         $w_i = w_i \cdot exp(-c_m \cdot y_i \cdot predict(X_i))$
15:     **end for**
16:     $w = w/sum(w)$
17: **end for**
```

The prediction function for the ensemble classifier is given in Algorithm 9. It is a rejection cascade with a fixed threshold encoded in the variable $rej < 0$. If the partial score of the classifier is less than rej, then the result is returned. This produces a speed boost because the majority of examples are usually negative. The upper bound for the execution time of this module is $O(nrweak \cdot maxdepth)$ but in practice not all stages are evaluated and the classifier returns prematurely.

Algorithm 9 Ensemble classifier predict - $predict(X)$

```
**Input:** $nrweak$ decision trees, negative rejection threshold $rej$, input feature vector $X$
**Output:** prediction result $res$
1: $res = -rej$
2: **for** $i = 1 : nrweak$ **do**
3:     $res = res + predictTree(X)$
4:     **if** $res < 0$ **then**
5:         **return** $res$
6:     **end if**
7: **end for**
8: **return** $res$
```

The classifier presented in this section is capable of handling large datasets and is able to exploit multiple cores for execution by paralellizing at the feature selection level. This module can be used as a solid building block for general object detection.

3.2.3 Experimental results

To test the classifier, we apply it to several classification benchmarks. The datasets represent supervised learning problems posed as binary classification tasks. We work with numerical data and several features. Table 3.2 provides statistics about the datasets such as: number of features, training set size and test set size. The source of these is the Machine Learning Repository [1]. Where no train/test split exists, the whole set is used and crossvalidation is performed for evaluation.

name	**nr. features**	**training set size**	**test set size**
adult	14	32561	16281
pima-indians-diabetes	8	768	768
wdbc	31	569	569
skin-nonskin	3	245057	245057

Table 3.2: Classification benchmark statistics

Table 3.3 shows the error rate obtained for different classifiers. We include the best results reported on the site for the benchmarks as well as from [2], there are probably other publications with better results.

3.2.4 Conclusions

The presented algorithms for training and predicting are essential to the detection system that are going to be presented. We have shown how to implement decision tree training efficiently for large datasets. By performing a sorting operation in the beginning of the training phase, the time complexity of the execution time of the following steps can be reduced from $O(D \cdot N^2)$ to $O(D \cdot N)$, where N is the number of training examples and D is the dimension of each feature vector.

Other relevant changes to the standard high level AdaBoost algorithm are required for further speed gains. In this regard the important proposed steps were: preserving the sorted structure at training during the splitting phase, implementing decision trees as arrays that store the nodes in preorder and premultiplying by the weight factor to avoid multiplications entirely at prediction.

The experimental section shows that the classifier has both fast execution time at training and at testing and also provides excellent accuracy values on several classification benchmarks. The library source code is available publicly at the repository at [3].

[1]Dataset repository
[2]Classification benchmark results
[3]Classification library repository

adult

classifier	parameters	error rate	training time	test time
Boosting + Decision trees	depth = 3, 15 trees	14.06 %	0.27 sec	0.005 sec
K-Nearest Neighbor	K = 31	19.74 %	0	70.26 sec
Naive Bayes Gaussian	-	31.35 %	0.002 sec	0.006 sec
FSS Naive Bayes	-	14.05 %	-	-

pima-indians-diabetes

classifier	parameters	error rate	training time	test time
Boosting + Decision trees	depth = 20, 100 trees	2.34 %	2.98	0.008 sec
K-Nearest Neighbor	K = 127	32.42 %	0	0.051 sec
Naive Bayes Gaussian	-	29.81 %	0	0.001 sec
Logdisc	-	22.3 %	-	-

wdbc

classifier	parameters	error rate	training time	test time
Boosting + Decision trees	depth = 5, 5 trees	4.01 %	0.005 sec	0.001 sec
K-Nearest Neighbor	K = 81	33.55 %	-	0.047 sec
Naive Bayes Gaussian	-	8.43 %	-	0.001 sec
Naive MFT	-	2.9 %	-	-

skin-nonskin

classifier	parameters	error rate	training time	test time
Boosting + Decision trees	depth = 5, 100 trees	0.05 %	5.97 sec	0.13 sec
K-Nearest Neighbor	-	-	-	∞
Naive Bayes Gaussian	-	9.22 %	-	0.084 sec
FSS Naive Bayes	-	2.9 %	-	-

Table 3.3: Classification benchmark results

Chapter 4

Proposed candidate generation methods for fast object detection

This section presents proposed methods for candidate generation. We use the terms candidate generation, object proposal generation and region of interest selection to refer to the same procedure whereby the search space for the sliding window classifier is reduced to a smaller one either in a constructive way or by eliminating certain regions.

Existing object proposal methods were described in the previous chapter. We have demonstrated the need for a RoI selection module in applications where speed is critical. For a survey in this domain the reader should consult Hosang et al. (2014). Some best performing object proposal methods that are relevant to the current topic are: Endres and Hoiem (2010, 2014); Uijlings et al. (2013); Cheng et al. (2014); Zitnick and Dollár (2014). However, these approaches from the literature are for general objects. In this thesis we provide algorithms for specific object categories such as pedestrians and pallets. Later, we also provide a description of the system in which they can be integrated.

4.1 Pedestrian candidate generation based on position

In usual applications we are interested in detecting objects that appear only in the center of the image. An elementary approach would be to disregard the margins of the image to reduce the search space. However a more appropriate indicator is the position of center of the bounding box. We propose to admit only the rectangles whose centers lie in the central horizontal stripe of the image. This is a heuristic which is easy to implement and it can be verified by performing statistics on an extensive object detection dataset.

In Figure 4.1 we show the spatial distribution of the centers of bounding boxes for pedestrians from the Caltech pedestrian detection benchmark. It is easy to see that almost all box centers lie in a horizontal stripe. More quantitatively, 99.77 $\%$ of the boxes have their centers in the 200:300 stripe. Adopting this approach results in a $\frac{100}{480} \approx 0.2$ reduction of the number of candidates to consider. This condition can also be easily checked.

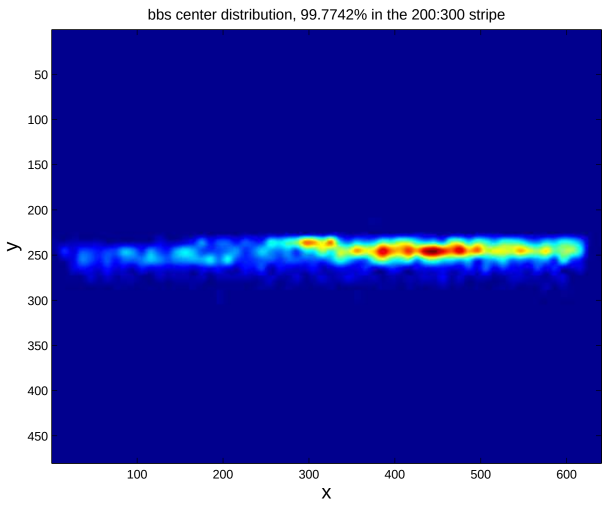


Figure 4.1: The distribution of bounding box centers from the Caltech pedestrian benchmark

We apply this selection module for pedestrian detection in Varga et al. (2014). The previous module was slow compared to the optimized feature extraction and classification module. This was the reason for adopting a simple and effective selection module. The tendency for objects to occupy central position is a general phenomenon and this fact can be reused in other applications also. In Hosang et al. (2014) one of the baseline object proposal methods is based on generating random boxes with center position and area drawn from a normal distributions. For a large number of samples this approach performs well indicating its usefulness.

4.2 Pedestrian candidate generation based on gradient

We have proposed a region of interest selection method for pedestrian detection that is based on edge strength. Our work has been published in Varga and Nedevschi (2013a). Here we provide details about the design of the algorithm. We have opted for a gradient-based method because it is fast to calculate and because object boundaries are likely to be present where the gradient magnitude is high. One can treat the ROI selection method as a simple classifier or candidate generator. This classifier must be fast enough to enable speeding up the system as a whole.

To successfully delimit objects we search for the top and bottom boundary. We opt for vertical boundaries since pedestrian width has a lot a variance and there are many other objects with vertical structure. Our approach can be viewed as constructive since it searches for possible positions for the top and bottom of candidates and constructs a list of candidates from these rather then eliminating candidates. This contrasts other methods from the literature where all possible regions are examined. We present our approach as pseudocode in Algorithm 10. The different steps of the algorithm are also visualized in Figure 4.2.

Algorithm 10 Candidate generation for pedestrians based on the gradient map

```
Input: Input image.
Output: Regions of interest as an array of rectangles.
 1: Filter the image with a Gaussian filter
 2: Obtain the edge image using a filter for the y direction (vertical).
    Name the filtered image top.
 3: Suppress small values using a fixed or dynamic threshold $t_1$.
 4: Filter the image top with a horizontal box filter of dimension $d$.
    Name the filtered image bottom.
 5: Set RoIs = $\emptyset$
 6: for all possible rectangles with top center point $(x, y)$ and height $h$ do
 7:     if $top(x, y) > t_1$ and $bottom(x, y + h) > t_2$ then
 8:         Add the rectangle $(x - h/2, y) - (x + h/2, y + h)$ to RoIs.
 9:     end if
10: end for
```

Step 1 helps to reduce noise, especially in images with high edge density (eg. dense foliage). In step 2, we apply a gradient edge filter in y direction to obtain the top and bottom boundaries. We can use different methods for obtaining edge image such as: filtering with Sobel, Prewitt or Scharr kernels, or applying the Canny edge detection algorithm. From hereafter we shall refer to the result of either of these operations as the *top* image because it determines to positions of bounding box tops.

We proceed by searching for locations where the gradient has a high value. For this, in step 3 we threshold to zero all pixels under a given value t_1. All non-zero locations will be considered as the middle point of the top of a potential bounding box. All that is left is to find the matching bottom, and the width is determined by the fixed aspect ratio. We observe that the bottom of a bounding box for a pedestrian will touch the feet, but it may touch it roughly at a single point (in the case of standing pedestrians when viewed from side) or in multiple points (for walking pedestrians). This suggests that it is not enough to search for the bottom of the bounding box under the first initial top point. We propose to sum up gradient values along the horizontal direction and to check these sums for possible bottom delimitators. To save time, the sums are precalculated using a horizontal cumulative sums resulting in a 1-dimensional box filter. This corresponds to step 4.

The region of interest selector will then consider all possible rectangles and will decide it is a region of interest if the gradient at the top has a value larger than a threshold t_1, and also if the sum of gradients along the horizontal at the bottom is above a second threshold t_2 (steps 5-10). The parameters for this classifier are: the type of edge detection (Sobel, Scharr, Prewitt, Canny), threshold value for the top image t_1, the dimension of the box-filter d, threshold value for the bottom image t_2, the heights of the admissible bounding boxes and the standard deviation of the Gaussian smoothing σ applied before processing (0 for no smoothing).

Figure 4.2 shows the results of the processing steps. You can see from left to right:

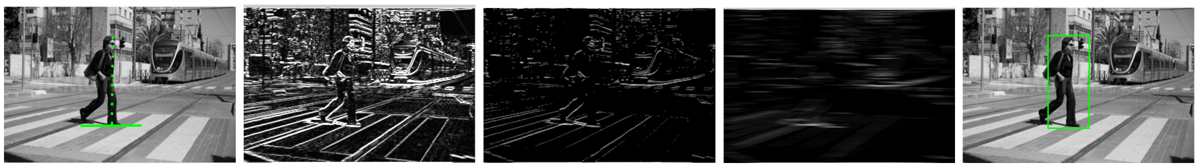


Figure 4.2: Illustrated steps from Algorithm 10.

possible top position and bottom line; Sobel filtered image; thresholded edge image; horizontal box filtered image and final resulting candidate.

4.3 Bottom-up constructive candidate generation for pallets

A constructive approach for region of interest selection entails building object candidates from bottom-up. Several top performing candidate generation algorithms operate this way such as: Selective Search in van de Sande et al. (2010) and Edge Boxes from Zitnick and Dollár (2014). The advantage is that it avoids the verification of every possible position for rejection. Instead, we generate possible boxes by finding their boundaries and combining these.

It is an acceptable assumption that most object boundaries are located at image edges. The constructive candidate generation algorithm detects the four edges of the bounding rectangle by inspecting the gradient image. In the related works section we have presented such an approach but that required checking every position. Here we propose improvements and extensions.

An essential step is the way in which we obtain the gradient image. The gradient image is needed for edge and line detection. Normalized gradient has been proposed and employed in calculating HOG Dalal and Triggs (2005) features and also in modern pedestrian detection algorithms Dollár et al. (2010). Normalized gradient values are obtained by box-filtering the gradient magnitude and dividing the original gradient magnitude and other channels by the filtered values. This ensures a successful edge detection even in dark regions.

In the following we provide the exact steps for calculating the normalized gradient maps. The gradient components along the x and y axis are obtained in a standard way by convolution with Sobel filters:

$$G_x = I * S_x \tag{4.1}$$

$$G_y = I * S_y \tag{4.2}$$

The gradient magnitude is defined as the L_1 norm of the two components:

$$M = |G_x| + |G_y| \tag{4.3}$$

The box filtered magnitude will act as a normalization factor:

$$\hat{M} = M * B \tag{4.4}$$

where B is a square box-filter of dimension w x w. Typical values for w are odd numbers from the interval $[5, 25]$. It is important to note that this filtering can be performed in $O(1)$ time per pixel for any filter size w. The normalized magnitude and the normalized gradient components are obtained by dividing the original values with the box filtered gradient magnitude (pixel by pixel):

$$\overline{M} = M/(\hat{M} + \varepsilon) \tag{4.5}$$

$$\overline{G_x} = \lambda \cdot G_x/(\hat{M} + \varepsilon) \tag{4.6}$$

$$\overline{G_y} = \lambda \cdot G_y/(\hat{M} + \varepsilon) \tag{4.7}$$

All division and summation operations in the previous definitions are carried out element by element. The small constant $\varepsilon = 5e - 3$ avoids division by zero. The multiplier λ is required to convert the normalized values into the $[0, 255]$ interval.

Intuitively, this operation produces strong responses where the relative change in intensity is large compared to the average intensity change in the neighboring region. This improves edge detection in poorly illuminated regions.

Once the gradient images are available we can proceed and perform edge and line detection. Candidate rectangles will be built from the detected lines. We first employ the normalized gradient in the y direction as in eq. (4.7) to detect important horizontal lines called horizontal guide lines. A histogram that accumulates gradient values along each line is used to find local maxima. In other words we perform a projection along the horizontal direction. The neighborhood size for the local maximum is fixed and it controls the density of lines that will be detected. The threshold for marking local maximums is chosen adaptively based on histogram values. More precisely, only values higher that the mean and a constant times the standard deviation are admitted. This approach is appropriate when the structure of the image contains strong horizontal lines and we can rely on the extracted guidelines later on. See Figure 4.3 for an illustration of the previous concepts. The projection of the gradient is shown on the left side of the image as a histogram. Detected lines correspond to the positions of local maximums in the histogram.

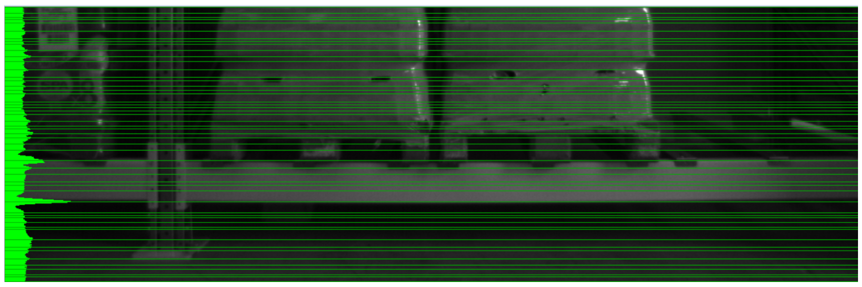


Figure 4.3: Visualization of the horizontal guideline detection process.

Vertical lines are detected only between guideline pairs that fit the dimension constraints. These lines are detected where the sum of gradient along x direction exceeds a certain percentage

(10 %). Summation can be performed in constant time by calculating integral images first. The resulting candidate rectangles arise from combining vertical edges that fit the dimension constraints regarding width, height and aspect ratio.

Algorithm 11 describes the candidate generation algorithm. First, we accumulate the gradient values along the y axis find local maximums in the $vlines$ list (line 6). The cumulative sum along the y direction is calculated to faster edge density calculation. Next, we consider each pair of vertical lines. If the difference between them satisfies the dimensional constraints, we proceed and calculate the edge strength between the two horizontal lines by making use of the cumulative sum E. The position of the local maximums are found in line 22. If both the left boundary and the right boundary (with a certain amount of slack to allow for variation) is among these positions a new candidate is added to the list.

The time complexity of this algorithm is $O((I.height/v)^2 \cdot I.width)$ where v is the size for the vertical local maximums. This may be large but all operations are fast since we only require additions and conditional statements. Edges are marked as strong in an indicator array if their edge strength is above a certain threshold.

Algorithm 11 Generate candidates based on gradient - $candidatesEdge(I)$

```
Input: an input image $I$
Output: an array with rectangles $candidates$
 1: Calculate $\overline{G_x}$, $\overline{G_y}$ and $\overline{M}$
 2: $hist = [0] * I.height$
 3: for $i = 1 : I.height$ do
 4:     $hist[i] = \sum_j G_y(i, j)$
 5: end for
 6: $vlines = nms(hist, v)$
 7: $E = cumsum(G_x)$
 8: $candidates = []$
 9: for $i = 1 : size(vlines)$ do
10:     for $j = i + 1 : size(vlines)$ do
11:         $height = vlines(j) - vlines(i)$
12:         $width = height/apect\_ratio$
13:         if $height < min\_height$ OR $width < min\_width$ then
14:             $continue$
15:         end if
16:         if $height > max\_height$ OR $width > max\_width$ then
17:             $break$
18:         end if
19:         for $k = 1 : I.width$ do
20:             $edges(k) = E(j, k) - E(j, i)$
21:         end for
22:         $hlines_{i,j} = nms(edge, h)$
23:         for $k = 1 : I.width$ do
```

```
            if $k \in hlines_{i,j}$ AND $k + width \in hlines_{i,j}$ then
                $candidates.add(rect(i, k, width, height))$
            end if
        end for
    end for
end for
return $candidates$
```

The region of interest selection method presented here is applicable mainly for objects that have a near rectangular shape such as cars, buildings, poles etc. In the papers Varga and Nedevschi (2014, 2016) we have applied it for pallet detection. The frontal view of pallets is rectangular so this approach is well suited for such applications.

4.4 Incorporating stereo information for candidate generation

In applications where we have access to stereo depth information, we can limit the region of interest for processing by considering only the objects with fronto-parallel surfaces. The reason for this is that the axis of the stereo system is roughly perpendicular to the target objects. Also, relevant objects tend to occupy a central position and a large percentage of the image.

Objects with fronto-parallel surfaces appear as a line in the u-disparity map. Also, they lie on the disparity plane with high appearance frequency. We define the principal disparity as the disparity value that corresponds to the highest local maximum from the disparity histogram. The highest local maximum is considered because this corresponds to the largest obstacle in front of the camera. We call principal disparity plane the plane obtained by selecting only points that are close to the principal disparity. This is equivalent to highlighting only the objects that are closest from the visual scene.

Once the principal disparity value is determined, the region of interest can be limited to the zone where such disparity values are frequent. We do this by starting from the extremities (left, right and bottom) and shrinking the boundary of the original region of interest until the frequency of the preponderant disparity exceeds a limit (see Figure 4.4). The limit is chosen in an adaptive manner based on the mean and standard deviation of the histogram. Principal disparity also gives us information about the approximate and expected dimensions of the objects in the image plane. This also reduces the number of possible candidates. We apply normal edge-based candidate generation on the reduced region of interest and apply the new constraints found regarding the size of the object.

In the paper Varga and Nedevschi (2016) we have presented a system that incorporates the presented approach successfully. Target pallets are detected only on a limited region that corresponds to the most preponderant fronto-parallel obstacle from the scene. Figure 4.4 illustrates a sample result and describes the processing steps required. The image contains: top-left -

disparity histogram; top-right - original image with reduced region of interest marked with white lines; bottom-left - v-disparity map; bottom-right - disparity map with values highlighted that are close to the principal disparity; the projections along the two axis are visualized that help determine the region boundaries.

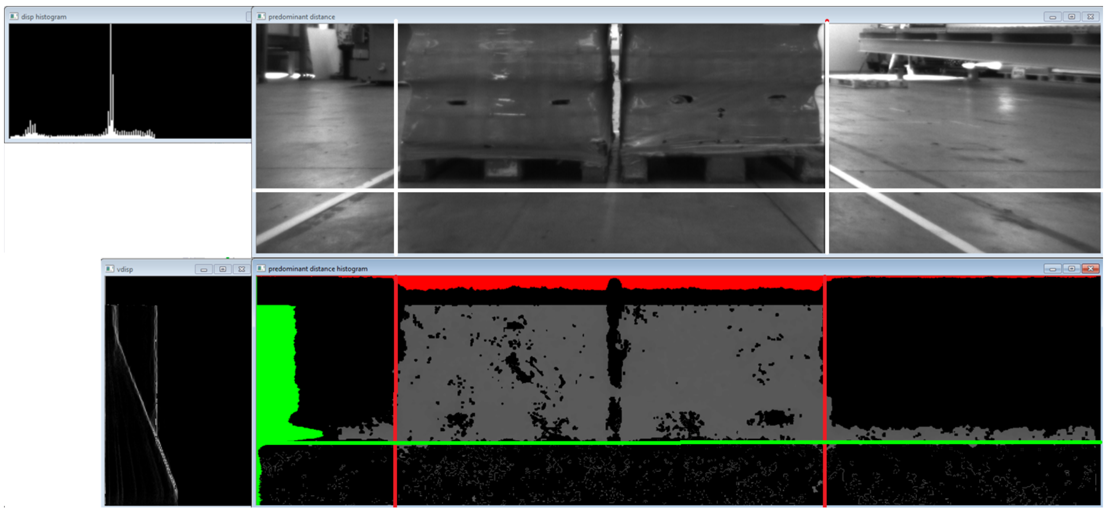


Figure 4.4: Stereo-based candidate generation - sample result.

Boosted decision trees offer both high classification accuracy and fast prediction time. Since prediction is made by comparing individual features against threshold, the time taken does not depend on the dimension of the feature vector. This enables us to use longer feature vectors and pick the best features at training time.

The classifier is trained using the positive examples available from the manual annotations. Negative samples are generated automatically from each training image from regions that surely do not contain any of the target objects. A bootstrapping process can be applied which trains a new classifier by including the mistakes made by the old one.

State-of-the-art pedestrian detection systems employ decision trees that are limited to height 2. It has been shown that this is a sweet spot that ensures the most accurate detection. In Varga et al. (2015) we test whether or not such an observation holds for pallet detection. Aside from the fact that we detect different types of objects, our feature vector also has other characteristics.

We operate with more descriptive features called normalized pair differences (**npd**). These features were proposed by us in a different work specifically for the task of pallet detection. They represent intensity pair differences normalized in such a way as to ensure certain illumination invariance properties.

Decision trees with limited height/depth are usually a compromise. It is often the case that leaf nodes contain a mixture of positive and negative instances and thus the tree must misclassify a certain percentage of the instances. By enabling a larger depth one can expect to have a more powerful classifier because of the increased possibility of branching further. Care must be taken because this runs the risk of overfitting on the training data since the classifier becomes stronger. However, increasing the depth of the tree requires storing more data and as a consequence the memory requirement grows exponentially.

4.5 Experimental results

Candidate generation algorithms are evaluated by checking the overlap between the candidates provided by the method and the ground truth object bounding boxes. The percentage of recalled bounding boxes are defined as the **coverage**. A box is recalled if it overlaps sufficiently with the ground-truth box. In general, relative overlap is considered but for more precise localization an absolute overlap criteria is desired. We have considered an absolute overlap threshold: the absolute positioning error should be less than 15 px along both axes. The results with different methods on our training dataset is presented in Table 4.1. Even though we do not achieve full coverage, rectangles near the ground truth are obtained, and using adjustments and validations we can come closer to the actual location of the object.

The method $All(x, y)$ signifies selecting all candidates that meet the dimensional constraints with a stride of x along the x axis and y along the y axis. This is the baseline and it generates a large number of candidates but offers 100 % coverage. $Grid(x, y)$ is an approach that finds both global horizontal and vertical guidelines in the same manner as presented in the previous section. The two parameters indicate the neighborhood size for local maxima detection from histograms. $Edge(x, y)$ refers to the presented edge-based candidate generation with non-smoothed gradient obtained from Sobel filters. *Edge normalized(x,y,z)* uses normalized (smoothed) gradient with a box filter of dimension z x z. The presented results are obtained by evaluating on the Viano2 training set. The reader can find more details about the dataset in the experimental results section for logistics operations detection methods. Only optimal parameter configurations are shown.

The approach from the last row offers an acceptable coverage while drastically reducing the number of candidates generated per image. The numbers in the parentheses indicate the step size in horizontal and vertical direction, and the filter size (where applicable).

Method	**Coverage**	**Avg. nr. candidates**
All(5,5)	100 %	1370k
Grid(7,7)	99.40 %	508k
Edge(5,3)	99.52 %	374k
Edge normalized (3,3,15)	98.81 %	35k

Table 4.1: Comparison of different candidate generation schemes.

4.6 Conclusions

This section has presented several approaches for candidate generation. Each method is directly applicable to one of the two specific tasks: pedestrian detection or pallet detection. We have found that gradient-based candidate generation helps in eliminating most of the irrelevant candidates. Essential steps for this method are: edge detection from a normalized gradient map; detecting dominant horizontal and vertical lines; grouping these lines into rectangles the respect several constraints. These provide a list of good potential candidates. This approach has high coverage (recall) and is also small compared to other approaches and can be generated rapidly.

Chapter 5

Contributions to object detection and classification methods

5.1 Automatic image annotation by measuring compactness

In its simplest form, automatic annotation of images aims at labeling images with keywords from a dictionary. It is strongly related to object detection but in this case the position of the object is not required. In this section we present a proposed approach for performing label transfer that is similar to k-nearest neighbor classification.

5.1.1 Compactness definition and interpretation

The novel part in our method is the way similarity is measured between two images. We employ a measure called compactness. Compactness is defined on two point sets: *X* - data points and *C* - cluster centers. It is a measure that indicates how close data points are to a set of centers. The set of centers will represent cluster centers from training image feature descriptors. So we distinguish between a set of data points X and the center points C and define the compactness in the following way:

$$\mathbf{c}(X, C) = \frac{1}{|X|} \sum_{i=1}^{|X|} \min_j d(x_i, c_j) \tag{5.1}$$

where $j \in \overline{1, K}$, $|X|$ denotes the cardinality of the set X and $d(x, y)$ is a metric defined on the D dimensional space. The previous definition states that compactness is the sum of the distances of each point from X to the closest point from C. Here X refers to any points in general, and in particular it can be the same as the set of features extracted from a test image. In our experiments we have found that the L^1 distance performs best in this context compared to the L^2 or the Chi-Square metric.

A less restrictive definition uses L^p norms instead of the distance function. This is useful in practice because it avoids extracting roots and has interesting properties.

$$\mathbf{c}(X, C) = \frac{1}{|X|} \sum_{i=1}^{|X|} \min_j ||x_i - c_j||_p^p \tag{5.2}$$

Note, when applying k-means on a set of points X, the objective function to minimize is exactly the compactness of the centers and the point set X. So the following are equivalent to the L^2 norm compactness applied to the same points from which the clusters centers were obtained: within-cluster sum of squares; the minimum sum of squares; distortion function; potential function (the literature uses a multitude of terms referring to this value).

The interpretation for such a measure involves decomposing the sum in two terms. Suppose we are given a set of points X, and we want to find their compactness relative to some set of centers C. Consider the following partitioning of X around each $c_k \in C$ (Voronoi partitioning):

$$X_k = \{x \in X | k = argmin_j\{||x - c_j||\}\}, k = \overline{1, |C|} \tag{5.3}$$

This states that the sets X_k contain all the points that have center c_k as the closest center to them. Clearly the sets X_k are mutually disjoint sets and $\cup X_k = X$. Then the following identity is true for compactness that uses the L^p norm:

$$\begin{aligned} \mathbf{c}(X, C) &= \frac{1}{|X|} \sum_{k=1}^{|C|} |X_k| \mathbf{c}(X_k, c_k) \\ &= \frac{1}{|X|} \sum_{k=1}^{|C|} \sum_{x \in X_k} ||x - c_k||_p^p \end{aligned} \tag{5.4}$$

$$= \frac{1}{|X|} \sum_{k=1}^{|C|} (\sum_{x \in X_k} ||x - \overline{x_k}||_p^p + |X_k| \cdot ||\overline{x_k} - c_k||_p^p)$$

Where $\overline{x_k}$ are the centers of mass for the points X_k. The decomposition is true because the partitions contain only the closest elements to c_k.

The advantage of using such a measure over brute force comparison are: it is much faster and it avoids noise due to outliers because it operates on distribution centers rather than on raw data points. Another common alternative is distance defined on bag-of-words type histogram descriptors. If the bag of words approach is used, then every image will be characterized by a histogram which reflects the distribution of the closest prototypes associated to each feature. The prototypes are k-means cluster centers and together they form the dictionary. The disadvantage in this case is that some relevant centers for the current image may not be present in the global dictionary. This prohibits the correct comparing of the images since important centers will be mapped to other centers from the dictionary. Even if all the relevant centers of the image are inside the dictionary, if two images have the same histograms, we cannot determine how close or far they are even though there may be significant differences between them.

5.1.2 Label transfer for image annotation

Once the similarity measure is in place, a mechanism is needed to transfer labels from similar images. In Varga and Nedevschi (2013b) we have introduced several method for transferring labels. All methods rely on the construction of a label histogram where the histogram bin values are obtained from weighing the labels from the top-matching images using different strategies.

The result of the matching procedure can be modelled as a function μ which returns an ordered list of indexes based on the compactness between the test image descriptors and each of the training image centers: $\mu(I) =< i_1, i_2, ..., i_N >$, where $\mathbf{c}(X, C^{(i_1)})$ is the minimal compactness, the one with i_2 is the second smallest and so on.

We define a weight function $\omega : \overline{1, N} \rightarrow \mathbb{R}$. Every label from the training image I_{i_n} increments the corresponding bin in the histogram by $\omega(n)$:

$$h = \sum_{n=1}^{N} \sum_{l \in L_n} \omega(n) \delta_l \tag{5.5}$$

L_n represents the labels from n-th match, more precisely from the training image I_{i_n} from the list $\mu(I)$. These labels are available from ground-truth information G. $\delta_l \in \mathbb{R}^M$ is a vector containing zeros on all positions except at the position uniquely associated to label l where it is one. By changing the expression of the weight function ω different types of transfer techniques can be achieved. In the following we present some particular cases.

In this case, every match from the list $\mu(I)$ contributes evenly to the histogram. We have:

$$\omega_0(n) = 1, \forall n = \overline{1, N} \tag{5.6}$$

This is the most elementary type of transfer, it can be viewed as a majority voting scheme. It has the advantage that it eliminates those labels that only appear in a few matches. However, if the matching technique is good, we want to give more importance to the best matches.

Rank based transfer entails weighing the best matches more and decreasing the weight exponentially based on the rank. In this case the weight function has the following form:

$$\omega_a(n) = 2^{a(1-\frac{k-1}{N-1})}, \forall n = \overline{1, N} \tag{5.7}$$

The parameter a can be tuned to obtain the best results. Of course the base of the exponent can be any number $b > 1$ or equivalently we can choose a to be $a' log_2 b$. One can see that $\omega_a(1) = 2^a$ and $\omega_a(N) = 1$. Note that weighing the matches equally (case w_0) is a special case of this function where $a = 0$. This is why at parameter testing, these two functions fall into the same category.

Note that this gives importance to matches according to their position regardless of their distance to the test instance. It may well be that all matches are very close, in this case the rank is not relevant.

The technique presented in Makadia et al. (2008) favours the best match and the rest of the labels are transferred based on their appearance frequencies in the training set. This case corresponds to the following form:

$$\omega_J(1) = 10, \omega_J(n) = 1, \forall n = \overline{2, N} \tag{5.8}$$

The histogram values will be updated on the last step in order to take into account the label frequencies. It is a greedy technique and thus depends on a good first match.

To take into account the compactness values $\mathbf{c}_n$ of each of the matches, the following weight function is defined:

$$\omega_d(n) = 2^{b(1-\mathbf{c}_n/\mathbf{c}_1)}, \forall n = \overline{1, N} \tag{5.9}$$

This is particularly useful where compactness values are relevant, however the first match will always receive the same weight, i.e. because this weight function is relative to $\mathbf{c}_1$, it does not treat the case where even the best match is far away from the test instance.

Histogram construction using weight functions can be easily extended to the case where we intend to use multiple features. We begin by constructing the histogram normally for the first descriptor type. Then we save the histogram instead of resetting the bins to zero and repeat the process for the matches obtained from the other descriptor types. In this way every descriptor contributes to the final histogram which will provide the annotations.

In this paragraph we will refer to an instance of the algorithm that uses a specific kind of local feature simply as "a method" in order to simplify explanation. In this case, the definition for the transfer histogram becomes:

$$h = \sum_m \eta_m \sum_n \sum_{l \in L_{m,n}} \omega(n) \delta_l \tag{5.10}$$

where the index m refers to the method number, η_m is the weight of the method m and $L_{m,n}$ is the set of labels from the n-th match using method m. In our experiments we have set $\eta_m = 1, \forall m$, i.e. we weigh each feature type equally. Note that the order of applying different methods is irrelevant.

5.1.3 Algorithm overview

In this section we provide the high level steps required for the training process and for the image annotation process. Afterwards, the effect of different parameters on the execution time is discussed. The training involves the steps described by Algorithm 12.

Algorithm 12 Compactness training procedure

```
Output: centers from all training images.
1: for all training images $I_n$ do
2:     Extract local features $X^{(n)}$
3:     Apply k-means using $K$ centers to obtain $C^{(n)}$
4:     Save centers
5: end for
```

We have fixed the number of clusters for the k-means algorithm to $K = 20$ for all our experiments based on some preliminary tests. If multiple features will be used for annotation, it is necessary to run the training for each feature type. Note that this way, we form the building blocks for more complex methods that use different feature combinations and the training is done only once for each feature type.

In order to annotate an image, the following operations from Algorithm 13 are to be executed. If multiple features are used, then these operations are performed for each feature type, and the equation (5.10) is used once at the end to form the transfer histogram.

Algorithm 13 Compactness label transfer

```
Input: Training image centers.
Output: Annotations for every test image.
1: for all test images $I$ do
2:     Extract local features $X$
3:     Sample $X$ to get $B$
4:     for all training image $I_n$ do
5:         find $\mathbf{c}(B, C^{(n)})$
6:     end for
7:     Obtain the first $N$ best matches using $\mu(I)$
8:     Transfer ground-truth labels from matches to obtain annotation using (5.5)
9: end for
```

5.1.4 Experimental results

In the following we present our evaluation results from our work Varga and Nedevschi (2013b). Testing is performed on standard datasets for image annotation.

To enable comparison between methods, the protocol for evaluation follows Carneiro et al. (2007). The different datasets are split into two disjoint sets: training set - used for extracting k-means centers; test set - for the evaluation of the method. No information about the ground-truth labels of the test set are used when generating the automatic annotations. The automatically generated annotations are afterwards compared with the human given ones to obtain metric values.

We label each image with exactly five labels. For each keyword from the dictionary that appeared at least once in the test ground-truth, we calculate the precision and recall values. For each label, we define the following numbers:

- l_h - the number of times l appears in the test ground-truth;
- l_a - the number of times l was provided in an annotation by the automatic annotation method;
- l_c - the number of correct annotations with the label l.

In this setting the precision is $\mathbf{p} = l_c/l_a$ and the recall is $\mathbf{r} = l_c/l_h$. In order to obtain a global score, we find the average precision and recall. These are obtained by averaging the precision and recall values of all the keywords which appear at least once in the test ground-truth. Another metric used is the number of non-zero-recalls. This is calculated as: $\mathbf{nzr} = \sum_{l_{ic}>0} 1$.

Additionally, we introduce an indicator rarely used for evaluating annotation methods. The $F1$ score is the harmonic mean of the average precision and the average recall. It enables us to look at a single value for finding the best parameters and makes it easier to compare different annotation methods. By taking the harmonic mean, the score is closer to the lesser value, so a high $F1$ score can only be achieved with both a high precision and high recall.

We performed extensive testing on this database using different underlying features for the matching method. The structure of this database has been described in the previous works (e.g.Carneiro et al. (2007)). We mention only that the training set has 4500 images, and the test set consists of 500 images, the size of the dictionary is 374. The metric values for matching using only one type of feature, as well using multiple features are shown in Table 5.2 . The numbers next to the feature type indicate the dimension of the descriptor vector.

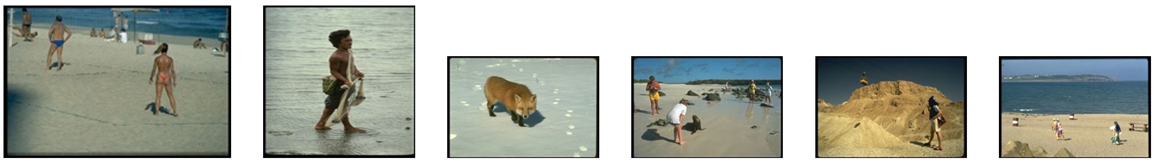
To clearly show the advantage of using compactness over histogram distances we provide test results on the Corel5k using the same features and transfer method w_J as in Makadia et al. (2008). In addition, we provide the best results obtained using one of the proposed transfer methods - w_a refers to the rank based exponential weighing with the subscript parameter a having the optimal value. Table (5.1) shows that in all cases compactness ensures a higher average precision and the same or higher recall. We have used L^1 metric for comparison and not Kullback-Leibler divergence for the Lab colorspace as in Makadia et al. (2008). The proposed weighing further improves score values boosting both precision and recall.

To find the best parameters we have performed a grid search varying several parameters in the ranges given below. Test time can be saved because matches are obtained once for each bag size and afterwards different transfer techniques can be applied. The results with the highest $F1$ score are presented in Table (5.2). As mentioned before, we determined that L^1 distance behaves best in this context for compactness calculation. Parameter ranges used for testing are:

- bag size - $|B| \in \{50k | k \in \overline{1, 10}\}$;
- neighbourhood size - $N \in \{5, 10, 15\}$;
- weight function type - w_a, w_J or w_d;
- weight function w_a parameter - $a \in \overline{0, 5}$;
- weight function w_d parameter - $b = 300$;
- considering frequency or not - $\varphi \in \{0, 10^{-3}, 2 \cdot 10^{-3}\}$;
- number centers per image - $K = 20$.

The Table 5.2 contains metric values using different features on the Corel5k benchmark. Entries are ordered based on $F1$ measure that guided us in deciding which method is better. The last column shows the average execution time in seconds for a single image annotation using a single threaded execution on the machine described in section 7.1. ('-' signifies no data available). Execution time is given for the best parameter combination and it depends on bag size. Simple color descriptors behaved surprisingly well compared to different texture descriptors. Also the low dimensionality of this feature permits a very low execution time. Color variants of the texture descriptors perform better than gray-scale ones. The lower part of the table contains combinations of features. This confirms that the annotation method successfully combines multiple features and produces better results than using individual features.

We now compare our results with previous state-of-the-art results in Table 5.3. Each percentage is taken from the indicated reference. Optimal configuration found by our tests is: using color descriptors along with DCT63 and SIFT, with the parameters set to: $K = 20, B = 200, \varphi = 2 \cdot 10^{-3}, N = 5, a = 3$ (last line from Table (5.2)). We note that Compactness based methods produce similar results to SML when using the same features (see DCT63 and DCT192 in Table 5.2) but using SIFT proves to be better. By efficiently utilizing multiple features, our simple approach outperforms many methods from the literature based on the $F1$ score including MBRM, SML, JEC, ProbSim. MRFA does not provide exactly 5 labels at annotation which helps to achieve higher scores. The better results of TagProp can be explained by the fact that it employs Metric Learning which could also be used in our context to improve results.



(a) Test image (b) Match 1 (c) Match 2 (d) Match 3 (e) Match 4 (f) Match 5

Figure 5.1: Sample matches and annotation from Corel5k

Predicted labels for test image a): **water, beach, tree, people, sand**. Showing only best five matches based on SIFT features. Note that incorrect labels from match 2 (confusion between sand and snow) get filtered out because of the transfer technique.

Feature	Precision	Recall	NZR	F1
JEC+RGB	20	23	110	21.39
JEC+Lab	20	25	118	22.22
JEC+HSV	18	21	110	19.38
Comp+RGB+w_J	21.98	24.38	121	23.12
Comp+Lab+w_J	21.29	24.80	123	22.91
Comp+HSV+w_J	19.33	26.94	128	22.51
Comp+RGB+w_5	21.58	26.85	123	23.93
Comp+Lab+w_5	22.34	25.62	123	23.87
Comp+HSV+w_3	21.95	26.68	124	24.09

Table 5.1: Comparison using the same feature type

Feature	Precision	Recall	NZR	F1	exec
Gabor(12)	7.46	8.67	76	8.02	-
HOG(9)	11.22	11.57	85	11.40	-
Law(9)	13.82	17.56	105	15.47	-
color HOG(36)	14.36	17.57	109	15.80	-
WLD(48)	16.99	18.42	108	17.67	1.83
SIFT(128)	17.00	24.94	122	20.21	-
CSIFT(256)	19.49	24.51	120	21.72	-
color(9)	22.71	27.06	128	24.69	1.39
DCT(63)	22.32	28.27	129	24.95	0.48
DCT(192)	22.82	29.02	129	25.55	5.28
SIFT-OCS(384)	23.75	31.23	140	26.98	22.58
WLD + color(57)	26.45	27.88	120	27.15	4.4
SIFT + WLD + color(441)	30.19	31.99	131	31.06	18.23
SIFT + DCT63 + color(456)	30.15	32.17	133	31.13	20.62

Table 5.2: Compactness based annotation results using different feature types on Corel5k

Some general remarks can be made about the influence of different parameters on the metric values. We analyse results from Corel5k in detail. It is possible that the behaviour on other databases is different. The effect of each parameter is analysed by fixing the other ones to their optimal values. Multiple cases are considered where necessary.

Increasing the bag size B has moderate effect on score values. This can be studied using

Method	Precision	Recall	NZR	F1
MBRM - Feng et al. (2004)	24	25	137	24.48
SML - Carneiro et al. (2007)	23	29	137	25.6
JEC - Makadia et al. (2008)	27	32	139	29.2
ProbSim - Chunsheng Fang (2009)	25.4	36.5	106	29.7
Compactness	30.15	32.17	133	31.13
MRFA-grid - Xiang et al. (2009)	31	36	172	33.31
TagProp - Guillaumin et al. (2009)	32.7	42.3	160	36.8

Table 5.3: Comparison with state-of-the-art Corel5k

Prediction	people swimmers pool water athlete	stone pillar temple people sculpture	people outside museum dance tree	cars tracks formula wall straightaway	sky mountain tree snow sky
Ground-truth	people pool swimmers water	pillar temple sculpture stone	tree people dance outside	cars formula tracks wall	clouds mountain sky snow

Table 5.4: Sample annotations using color+DCT63+SIFT from Corel5k

Figure (5.2). Score values increase and oscillate, and in some cases reach a maximum for fairly low values of B. Because bag size linearly influences the execution time, lower bag size values such as 100 or 200 can be utilized. This is practical because it achieves faster execution while maintaining near-optimal performance. The oscillating behaviour is due to the errors introduced from sampling.

We now study the influence of neighbourhood size N. Better results were obtained using smaller N values. This may be due to the relatively small size of the database, so most of the images have few good matches among the training instances. We have found that $N = 5$ produces best results for individual features and on some occasions $N = 10$ for multiple feature case. Further fine-tuning could involve experiments considering $N \in \{1, 2, 3, 4\}$. This also demonstrates that the matching technique is efficient because the first few matches provide good labels to transfer.

Figure (5.3) contains metric values using different transfer techniques. We have associated $a = -2$ to JEC-type transfer and $a = -1$ to distance-based transfer. We have found that in almost all cases, weighing based on rank performs best. In some cases we obtain better results with ω_d or ω_J, but the general recommendation is ω_a. The scores with $a = 0$ are almost always lower than the optimal scores obtained using $a = 3$ or $a = 4$. Overall tendency here suggests to accord significantly more importance to the best match. To provide a more encompassing overview Table (5.5) shows score values using only RGB features. Every row corresponds to a constant bag size and every column contains a different transfer technique.

If we consider frequency information, it can increase overall performance ($F1$ score). However, this almost always entails an increase in precision and a decrease in recall and NZR values. The reason for this is that favouring the more frequent terms reduces the chance to annotate with rare labels. The influence of the frequency was tested using 3 different values: zero influence, minimal influence setting $\varphi = max f_l$ as suggested, and twice the previous value. The second case gives higher $F1$ score in general.

Computation time varies in accordance with the time complexity formulas derived in section 5. It is linear with respect to feature dimension and also with respect to bag size. These two parameters can control the execution time. Modifying the bag size has only minor negative effects on annotation performance. Even though the IAPR-TC12 and ESP-game datasets are much larger annotation time still remains fairly low due to the optimizations mentioned (halting calculation when distance exceeds the current N-th maximum).

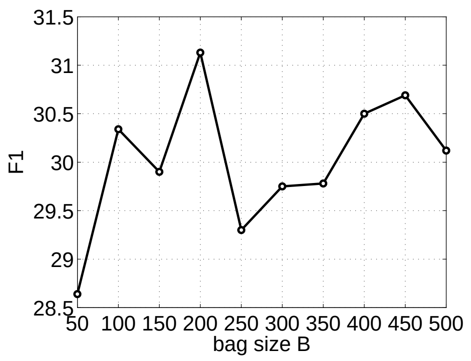
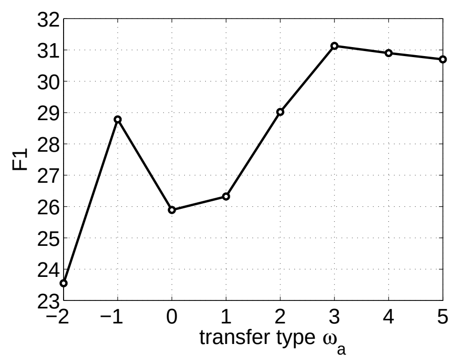
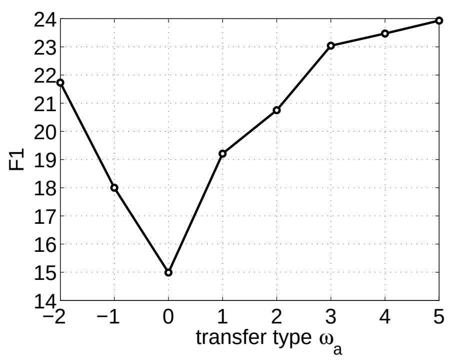


Figure 5.2: The influence of bag size - color+DCT63+SIFT

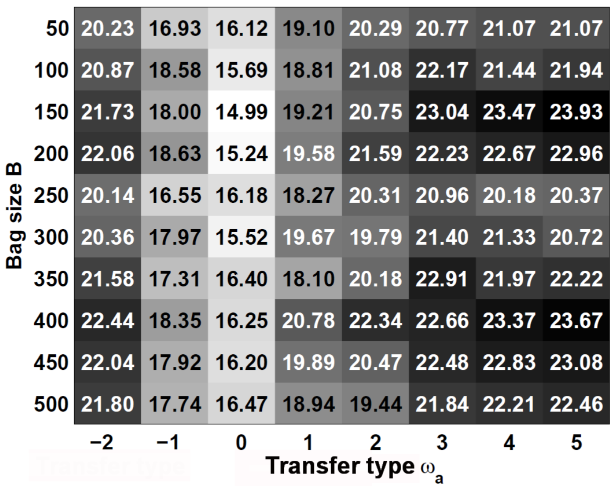


(a) multiple features



(b) RGB only

Figure 5.3: The influence of transfer type



Bag size B \ Transfer type ω_a	−2	−1	0	1	2	3	4	5
50	20.23	16.93	16.12	19.10	20.29	20.77	21.07	21.07
100	20.87	18.58	15.69	18.81	21.08	22.17	21.44	21.94
150	21.73	18.00	14.99	19.21	20.75	23.04	23.47	23.93
200	22.06	18.63	15.24	19.58	21.59	22.23	22.67	22.96
250	20.14	16.55	16.18	18.27	20.31	20.96	20.18	20.37
300	20.36	17.97	15.52	19.67	19.79	21.40	21.33	20.72
350	21.58	17.31	16.40	18.10	20.18	22.91	21.97	22.22
400	22.44	18.35	16.25	20.78	22.34	22.66	23.37	23.67
450	22.04	17.92	16.20	19.89	20.47	22.48	22.83	23.08
500	21.80	17.74	16.47	18.94	19.44	21.84	22.21	22.46

Table 5.5: The influence of parameters on F1 score

The IAPR-TC12 image collection consists of 20,000 still natural images taken from locations around the world and comprising an assorted cross-section of still natural images - see Grubinger et al. (2006). The same images are used from the IAPR-TC12 database as those in Makadia et al. (2008) in order to compare results in a correct manner. This database is larger, the training set numbers 17825 images and the test set contains 1980 images with 291 labels. The image annotations and test/training split is obtained from the files located at the web-page [1].

The metric values are calculated using all the labels from the ground-truth. This is the right way to obtain the number of correct labels however recall values will be lower. This is so because we only provide 5 labels, and in cases where in the ground-truth there are more than 5, we inevitably end up marking some labels as not recalled.

We provide some sample annotations for this dataset in Table 5.8. Notice that a lot of images have much more labels than 5. The results for this database (Table 5.6) again indicate that fairly good results can be obtained using simple color descriptors. However, SIFT features outperform other features mostly by reaching an $F1$ score of 32.13. It can be seen that the combination of different features is more successful on this database. Average precision value increases with 11%. Note that combining features results in lower recall and higher precision values. This is natural since more features provide more "opinions" about the correct label and the consensus tends to reflect the truth.

The comparison made in Table 5.7 shows that Compactness obtains much better precision than MBRM and JEC (by 15%). Recall and NZR values are lower, but we mention here that using JEC-type transfer similar values were obtained as in Makadia et al. (2008).

[1]Makadia annotation files

Feature	**Precision**	**Recall**	**NZR**	**F1**	**exec**
color(9)	23.89	23.63	216	23.76	2.0
DCT(63)	25.24	24.64	225	24.94	6.3
SIFT(384)	31.82	32.45	245	32.13	17.0
SIFT+DCT+color	42.9	22.6	228	29.6	44.0

Table 5.6: Compactness based annotation results using different feature types on IAPR-TC12

Method	**Precision**	**Recall**	**NZR**	**F1**
MBRM - Feng et al. (2004)	24	23	223	23.48
JEC - Makadia et al. (2008)	28	29	250	28.49
Compactness	42.9	22.6	228	29.6
TagProp - Guillaumin et al. (2009)	46.0	35.2	266	39.88

Table 5.7: Comparison with state-of-the-art IAPR-TC12

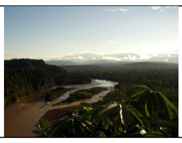
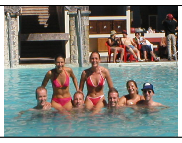
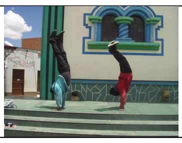
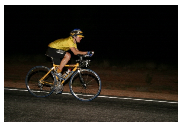
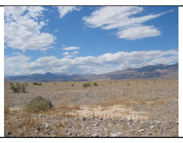
					
Prediction	view river jungle middle cloud	pool people woman tree man	building front ornament trouser jacket	bike country helmet side short	sky mountain cloud desert bush
Ground-truth	cloud hill jungle middle palm range river view	chair man people pool woman	building column front jacket ornament person trouser	bike cap country cycling cyclist hand helmet jersey racing road short side	cloud desert mountain shrub sky

Table 5.8: Sample annotations using color+DCT63+SIFT from IAPR-TC12

The ESP dataset is the result of an experiment involving collaborative human annotation. The subset of pictures used is the same as in Makadia et al. (2008). More exactly: 19659 training images, 2185 test images, annotated with 269 different labels. An advantage of this set is that it is a result of an agreement between multiple annotators, so annotations are not biased by individual preference.

Table 5.9 contains results using a limited set of features and their combination. Five sample annotations are provided in Table 5.11. In this case WLD texture descriptor and color descriptors collaborate well. This may be so because in this set, texture can discriminate instances better than in previous datasets. To enable comparison with the existing methods, we summarize other results in Table 5.10.

Feature	**Precision**	**Recall**	**NZR**	**F1**	**exec**
Law-color(30)	16.36	16.07	217	16.22	2.32
WLD(48)	19.65	17.33	228	18.42	3.61
color(9)	19.73	19.28	230	19.50	1.56
DCT(63)	21.49	20.50	236	20.99	7.92
SIFT(384)	22.75	20.42	230	21.52	35.13
WLD+color	31.07	19.78	227	24.17	11.63
SIFT+DCT+color	34.67	21.29	233	26.38	39.45

Table 5.9: Compactness based annotation results using different feature types on ESP-game

Method	**Precision**	**Recall**	**NZR**	**F1**
JEC - Makadia et al. (2008)	22	25	224	23.4
Compactness	34.67	21.29	233	26.38
TagProp - Guillaumin et al. (2009)	39.2	27.4	239	32.25

Table 5.10: Comparison with state-of-the-art ESP-game

NUS-WIDE from Chua et al. (2009) is a large image dataset consisting of 269,648 images and associated tags from Flickr. This was created by a research team from the National University of Singapore, who also provide tags for 81 concepts. It is suitable for testing label transfer annotation algorithms. We have obtained this dataset by downloading the images using the provided URLs, however 36515 images are either missing or are blank which can be detrimental for annotation precision.

We have carried out experiments using the proposed color descriptor and we have compared the obtained results with the NUS-WIDE Lab histogram based k-NN annotation baseline from Chua et al. (2009). The only difference between the two methods is the distance calculation. In the first case we have used compactness and in the second case the L1 distance between global Lab histograms as in Chua et al. (2009). We could not directly use the feature vectors provided with the dataset because they are global feature vectors and compactness operates on local features, but the underlying feature type is the same. For every test image, we generate 5 labels. If the ground truth information specifies n labels, we evaluate the performance on the

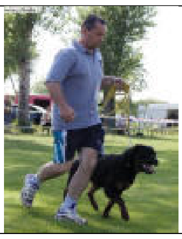
					
Prediction	people sky crowd tree blue	man black dog grass tree	coin gold round circle money	sky blue people tower building	old man shirt glasses book
Ground-truth	crowd man people pole sky tree	black dog grass green guy man run shoes white	circle coin gold old round square	blue building people sky tower	book glasses green hand man old shirt

Table 5.11: Sample annotations using SIFT from ESP-game

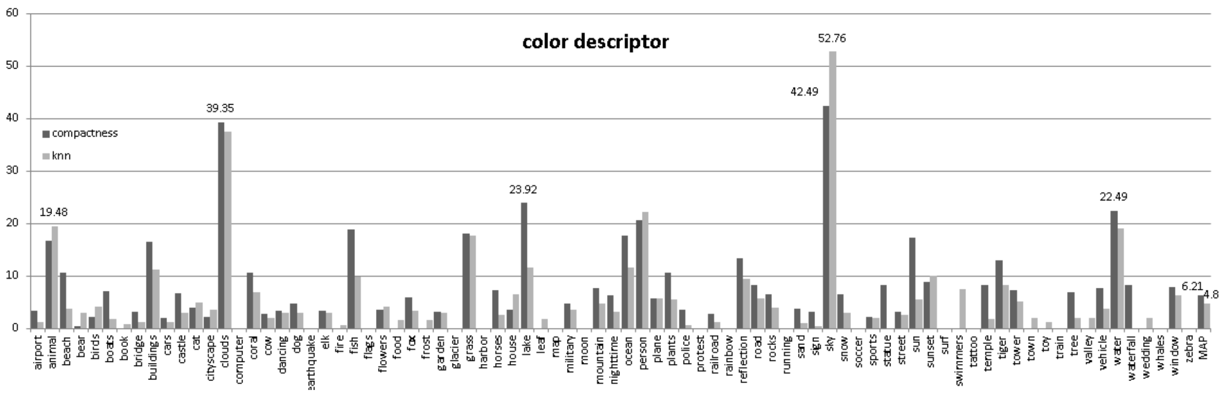


Figure 5.4: Precision values for each concept and MAP on the NUS-WIDE dataset

first m labels, where m = min(5; n). We present the annotation performance in Figure 5.4. It is given in terms of precision for each concept and in terms of mean average precision (MAP). Concepts with more training examples - such as clouds, person, sky - have a significantly higher precision value for both methods. The k-NN based method has more concepts with non-zero precision and performs better for some concepts with more training examples. However, for most concepts compactness provides a higher precision. The MAP obtained with compactness is of 6.21 in comparison with 4.8 corresponding to the k-NN based classification algorithm.

5.1.5 Conclusions

We have shown that compactness is a relevant measure to find similar images from a training dataset to a given query image. We can transfer labels from the training images to annotate the input image. The label transfer mechanisms combined with the compactness measure were compared to best-performing approaches and achieved competitive results on several annotation benchmarks. This work has been published and recognized in Varga and Nedevschi (2013b).

5.2 Detection methods for automated logistics operations

The work presented here is part of the PAN-Robots project described in PAN-Robots (2013) whose aim is to create an automated logistics environment. Installation and maintenance of such an environment is costly and time consuming. Thus, one of the main goals of the project is to ensure this with low installation time and costs.

Throughout this part we will use the following terms: AGV - Automated Guided Vehicle - refers to automated forklifts for logistic operations; operation point - 3D position of the center of the frontal view of the pallet or the future position of the unloaded pallet; load handling - operations pertaining to loading or unloading of palletized goods by the AGV.

The automated load handling system is required to perform accurate pallet detection and operation point estimation. It must also provide the orientation of the pallet. The input of the system consists of a pair of images, an operation point position request, information about operation type (the number of pallets, the level, number of reference points, storage type), pallet dimensions, 3D static map. During loading and unloading operations, the AGV travels to the operation point and stops at a distance of approximately 2.5 meters. At this position the system must provide the position of the pallets to enable corrections to AGV path. As the AGV approaches the palletss the positions of the pallets are updated online up to a certain distance of approximately 1.8 meters.

For loading and unloading operations, the system must detect and provide the 3D position of the pallet or pallets with an accuracy of: 5 cm (at 1 std. dev.) and 1 deg (at 1 std. dev.) at a distance of 2.5 m from the pallets; 1 cm (at 1 std. dev.) and 1 deg (at 1 std. dev.) at a distance of 2 m from the pallets.

The two main components of the system are the sensors and the processing unit. We employ two Manta G-223 NIR cameras mounted in canonical horizontal configuration (displaced horizontally and facing the same way). The resolution of the cameras is 2048px by 2048px. The cameras are equipped with Schneider Cinegon lenses with a focal length of 4.8mm and F number 1.8. An auxiliary light source comprised of several LEDs is positioned between the cameras. The cameras are mounted on the AGV behind the forks and are lowered on demand to grant view of scene in front of the forks.

The processing unit consists of an industrial PC ADLINK MXC-6301 which is a high-performance fanless embedded computer integrating a 3rd generation Intel Core i7 processor and QM77 chipset to provide powerful computing and superior graphic performance. The AGV provides a constant frame rate trigger to acquire images from the cameras. When this was not an option - as in our laboratory - we have used an Arduino Uno microcontroller for synchronous triggering of the two cameras. In the following we demonstrate that our current camera setup can provide the accuracy needed.

5.2.1 Pallet detection and position estimation

In this section we describe an original method for detecting pallets for automated logistics operations. This is clearly a specific case of object detection where the visual appearance of the object is relatively simple, although variation and lighting conditions make this task difficult.

The other challenge is that the scenario requires precise 3D localization which means that the detection must be precise. There is also a need for depth information. Our proposed system performs detection and depth estimation separately.

Precise pallet detection and position estimation is required for automating logistic operations. This enables accident free loading and unloading operations of pallets. Pallets are wooden supports with standardized dimensions. They are grippable from each side although usually the shorter side is employed. Vision-based approach is possible for such a task.

A pallet detection system is required to produce detections in an industrial environment. So it must be robust to lighting condition changes, and it must be able to operate in dark environments. In order to help the loading procedure, it must be accurate. Fast detection is required for increased productivity.

Our proposal is to use several relevant features for pallet detection. The first step is to adjust the exposure time of the cameras by measuring the average intensity in the input images. This value must be in a desired interval, and the camera exposure is changed based on a feedback loop for this purpose.

The second step is to select a list of candidates. Here we employ Algorithm 11 from the previous sections to generate candidates based on edge information. This quickly reduces the number of possible positions to check for pallets but still does not provide us with final detections since there are many false positives present in the list.

Features are calculated for each candidate and the resulting descriptor will be the concatenation of the different types of features: edge density along the defined border regions, normalized pair differences signature and the LBP histogram. An ensemble classifier is applied on the calculated features and the final detections are obtained by performing a non-maximum suppression.

Algorithm 14 Pallet Detection $palletDetect(I)$

```
Input: input image $I$, classifier model $M$
Output: detected pallet bounding boxes
1: Preprocessing
2: $candidates = candidatesEdge(I)$ with 11
3: $X = [size(candidates)][D]$
4: for $i = 1 : size(candidates$ do
5:     crop $I_2 = I(candidates_i)$
6:     calculate edge features in $E = edgeDensity(I_2)$ using 5
7:     calculate npd features in $F = npd(I_2)$ using 3
8:     calculate lpd features in $H = lbpHist(I_2)$ using 4
9:     $X(i, :) = [E, F, H]$
```

```
10: end for
11: $detections = []$
12: for $i = 1 : size(candidates$ do
13:     $s_i = predict(X(i,:))$ using 9
14:     if $s_i > \theta$ then
15:         $detections.add(candidates_i)$
16:     end if
17: end for
18: $NMS(detections)$
19: return $detections$
```

The non-maximum suppression module is responsible for eliminating overlapping detections with low confidence. It is common for detection methods to return multiple overlapping detections around the true position of the object. Our policy is to have low threshold for accepting detections as positives i.e. having θ form the previous algorithm set to a negative value, e.g. $\theta = -10$. We also tolerate slight deviations from expected positions, size and aspect ratio for pallets.

For each bounding box pair that overlap more than θ_x along the x axis and θ_y along the y axis the box with the lowest confidence is eliminated. Small overlaps are tolerated since close pallets can appear to superimpose near the edges due to small positioning errors. A low accepting threshold is set for the classifier, in order ensure that even detections with low confidence are found. Since the number of pallets that are needed to be detected is known a priori, this information can be used to retain only the most confident detections.

Several other enhancements can be made to the detection windows. These options are enabled in most cases. Corrections include: enhancing the scores of rectangles close to the center of the image that lie in specific regions where the pallets are expected to be; enhancing the scores of pallet pairs that lie next to each other along the x axis; enhancing the scores of rectangles that have correct aspect ratio.

The rectangle that indicates the position of the pallet in the left image is the starting point for the plane fitting procedure. All pixels corresponding to the pallet's wooden part and excluding the pockets are reconstructed by the stereo module to obtain a set of 3D points. A RANSAC approach is applied by fitting multiple plains on only a subset of the points. Each plane is scored based on the number of inliers, the close points to the plane. Parameters of the method are: size of the subset: 5 points; number of trials: 100; threshold for inliers: 2 (disparity values). The distance to the central point situated on the pallet is reconstructed with the disparity value obtained from the fitted plane. This provides a more robust estimation and also gives subpixel disparity values.

Pallet orientation is estimated by averaging distances along a vertical stripe and left and right extremities of the pallet. It is more precise and stable than using the normal vector of the plane. Orientation of the pallet is needed only along a single axis.

5.2.2 Methods for treating unloading operations

Unloading operations require the target position for the place of one or two pallets. We define the **unloading cuboid** as the cuboid in 3D space which represents the empty area in front of the AGV bounded by obstacles on the left and right and the floor on the bottom. Our aim is to detect this unloading cuboid. An error is signaled if for some reason the dimension of the cuboid is smaller than the dimension of the pallets that are to be unloaded.

Rack storage is comprised of poles holding up shelves at different heights. We start by detecting the structure of the rack. This can be obtained from the disparity map. The poles and the support for the pallets must be at roughly the same distance from camera because the approach of the AGV is almost perpendicular to the rack. We extract the disparity corresponding to the main fronto-parallel object from the scene. We call this the **principal disparity** d^*.

The principal disparity is taken to be the largest local maximum from the disparity histogram above a certain minimum threshold value.

$$d^* = \arg \max_{d > d_{min}} \{h(d), d \in N(d)\} \tag{5.11}$$

where h is the disparity histogram. This definition takes into account the following considerations. It is a local maximum because it must appear frequently in the image. It is the largest local maximum because we want to consider the closest fronto-parallel object. We must ensure that the disparity is higher than a limit because we want to eliminate cases where the background walls have a larger appearance frequency. This limit is calculated from the maximal admissible distance to the rack (around 3 meters).

Once the principal disparity is determined, we filter the disparity image and retain only the disparity values that are close to the principal disparity and ignore the rest of the depth map. We then construct vertical and horizontal projections of the non-zero elements.

$$D^*(x, y) = abs(D(x, y) - d^*) < 5 \tag{5.12}$$

$$D^*_x = \sum_y D^*(x, y) \tag{5.13}$$

$$D^*_y = \sum_x D^*(x, y) \tag{5.14}$$

These projections provide us with the necessary information to delimit the free zone in front of the camera. Columns and rows with high projection values in D^*_x and D^*_y respectively mean the presence of the rack. We start from a point located in the middle of the region of interest of coordinates (x_0, y_0) and travel in three directions to find the left, right and bottom limit to the open space available.

The coordinates of the left limit in pixel coordinates is x_{left} and it is first value x left to x_0 for which the vertical projection $D^*_y(x)$ is above $max(h) * 0.8$. The value of x_{right} is determined in the same manner in the other direction. For y_{bottom} we use the horizontal projection $D^*_x(y)$.

The points (x_{left}, y_{bottom}) and (x_{right}, y_{bottom}) can be reconstructed using the principal disparity d^* to obtain the real world coordinates of the corners of the unloading cuboid: (X_{left}, Y_{left}, Z_{left}) and (X_{right}, Y_{right}, Z_{right}). The pallets must be placed inside the cuboid on the level of Y_{left}. The value of Y_{right} should be very close to Y_{left} otherwise we signal an error. The pallets must be placed next to the closest pole. This is the left pole if $|X_{left}| < |X_{right}|$ otherwise the right pole. All the presented operations are illustrated in Figure 5.5 for a sample case. The image contains in the top: disparity histogram and the limits of the unloading cuboid drawn on the input image; bottom: disparity image with principal disparity highlighted, the projections and the limits of the unloading cuboid are overlayed.

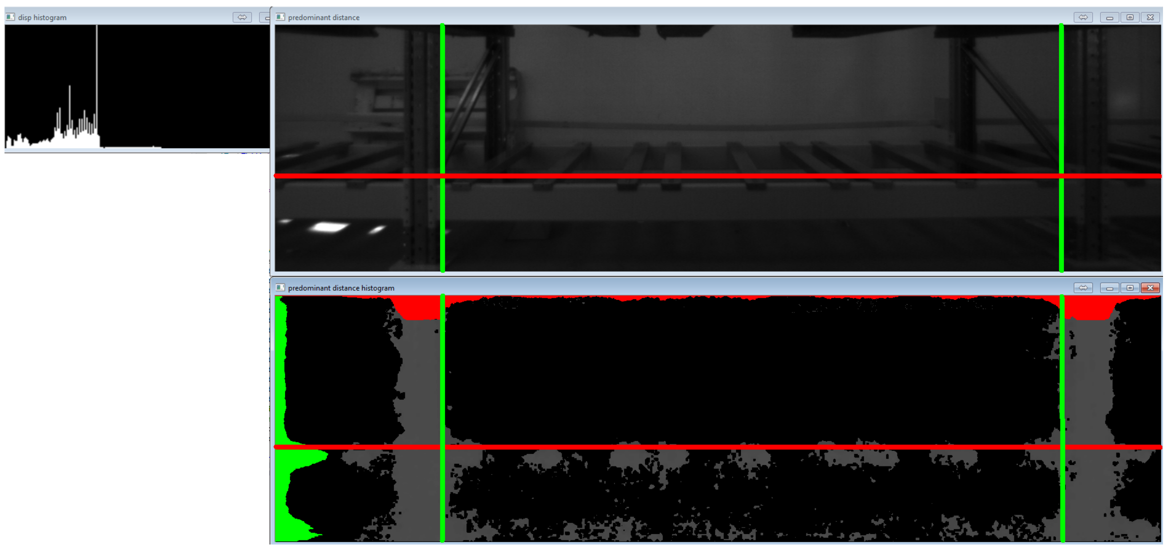


Figure 5.5: Unloading operation for rack - visualization of processing steps

Block storage - ground level

It is required for scenarios where the unloading operation takes place in a block storage on ground level. This helps to determine the target position of the unloaded pallets based on markings on the ground. We have developed a module that tackles this specific task. The main tool employed in this process is the Hough transform Hough (1962); Duda and Hart (1972) for finding the important lines in the input image.

Since stereo reconstruction is not very reliable on the floor region, both images need to be processed in order to have stereo information. We start by performing a standard Hough transform for line detection. The input image is a linear combination of the edge image and the original intensity image. The importance given to the edge image is 0.95 while for the intensity image it is 0.05. The reason for this is that we want to detect bright lines only. The bin size for angles is set to 1 degree. We locate the local maximum in the Hough accumulator using a neighborhood of 5. The left line from the floor marking is obtained by finding the line with the largest angle $\in [50, 70]$ degrees. We find the middle horizontal line through the region of interest. This line must also lie close to the middle of the region of interest. The position from the top of the region of interest must be $[90, 150]$. The right line from the floor marking is obtained by finding the line with the largest angle $\in [290, 310]$ degrees.

After having the three main lines for the floor marking, we can extract the intersections. Name the intersection between the left line and the middle line point A_l for the left image and the other intersection as point B_l. For the right image we define A_r and B_r analogously. We can reconstruct the 3D position of these points by using the differences $A_l - A_r$ and $B_l - B_r$. Alternatively, we can operate with the disparity values from the stereo matching algorithm.

Figure 5.6 illustrates edge map and the 50 most important lines obtained via Hough transform in green for a sample case. In blue are the lines of the actual floor markings. Red crosses indicate the intersection points. The edge image is noisy because of light reflections from the floor.

The position of the pallets can be found by knowing their dimensions and placing them at a certain distance from each floor marking. Note, that currently we are assuming that the AGV is roughly facing the floor markings and that the middle line is approximately horizontal.

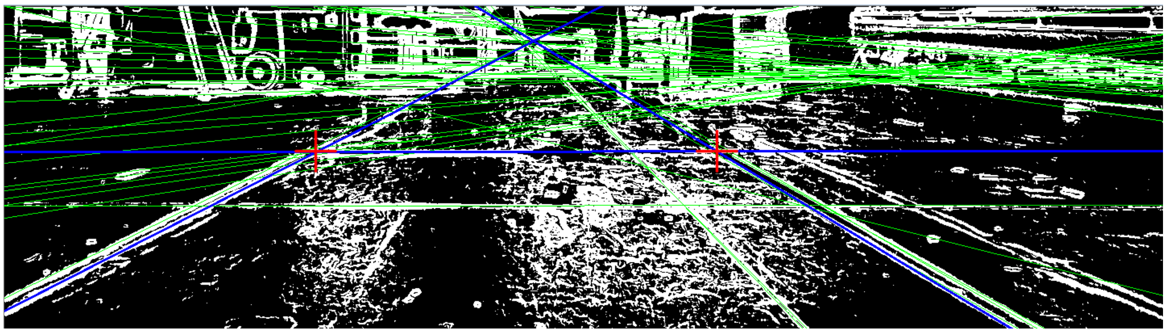


Figure 5.6: Unloading operation ground level - output for debugging

Block storage - stacking

For this case we use most of the ideas from the rack case. However, the unloading cuboid must be on top of the block storage. In this case we pick x_{left} and x_{right} differently. Define x_{left} as first value x left to x_0 for which the vertical projection $D^*_y(x)$ is below $max(h) * 0.05$. This ensures that pallets are placed on top of the block storage. Figure 5.7 shows a sample scenario for this type of operation. The image contains in the top: disparity histogram; limits of the unloading cuboid drawn on the input image; bottom: disparity image with principal disparity highlighted, the projections and the limits of the unloading cuboid.

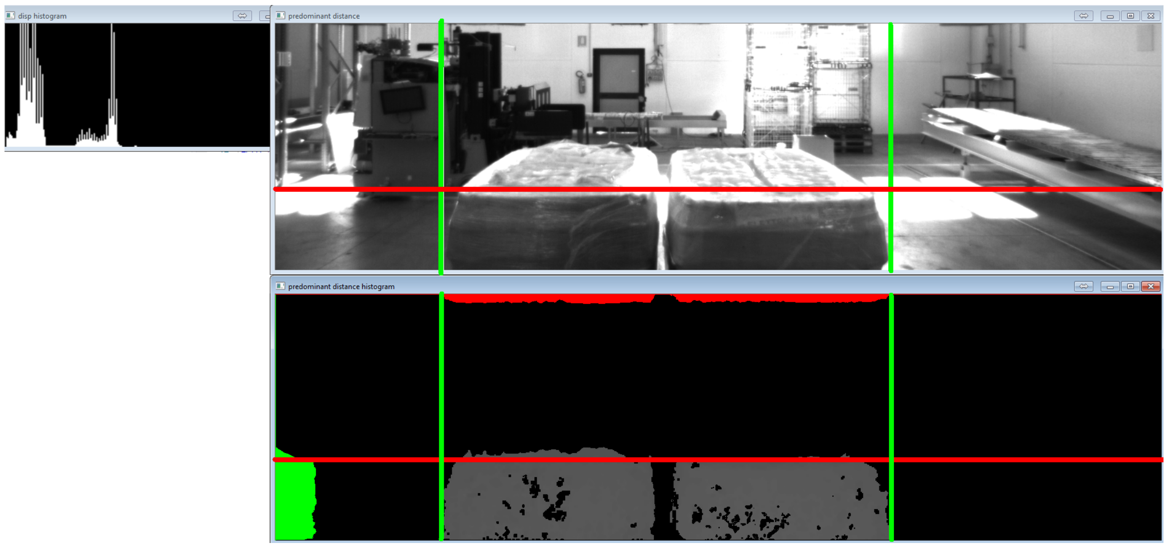


Figure 5.7: Unloading operation for block storage - visualization of processing steps

5.2.3 Experimental results

This section provides a quantitative evaluation of the proposed modules. These evaluations are performed on manually annotated datasets. We consider the problem of pallet detection to study the effectiveness of candidate generation schemes and the newly introduced feature types.

Evaluation is performed on images acquired from a real warehouse from Viano, Italy. All methods are trained on a training set called Viano2 and evaluated on a separate test set Viano2 or Viano3-5. All ground truth bounding box positions were manually determined by inspecting the images and marking pallets as rectangles.

Since all evaluation metrics depend on determining whether or not two rectangles overlap sufficiently, we state precisely what we consider as an overlap. Usually, for object detection intersection over union (PASCAL VOC criteria) is used to determine overlap. A relative overlap does not penalize absolute differences so severely for large objects. This is not desirable in applications with strict requirements. Another remark is our input images have a large resolution of 2048x2048 pixels, so 10-50 pixel differences are only small changes in position.

For this reason, we adopt an absolute overlap system instead of the traditional relative overlap. We define the absolute positioning error along the x axis E_x as the difference between the union and overlap of the intervals along the x axis of the two rectangles. E_y, The absolute positioning error along the y axis is defined analogously. We consider an overlap a **precise** match if $E_x \leq 15$ and $E_y \leq 15$; and a **normal** match if $E_x \leq 50$ and $E_y \leq 50$. Our overlap measures are more strict than the relative overlap of the pascal VOC measure because of the system requirements. Considering that typical pallet size is 500x100 pixels, a 50x50 pixel area is only 5 percent of this. A precise position amounts to an error of 7.5 pixels ≈ 1.5 cm using our hardware setup. Note, that the error measures the difference at both extremities, and thus it is roughly twice the error of the position of the center of mass. But by measuring overlap, we assure a better matching between the ground truth and the prediction rectangle.

We evaluate the detection accuracy of different approaches for pallet detection. All classifier models are trained on the Viano2 training set and evaluated on two different test sets: test set Viano2 which is somewhat similar in appearance but different to the training set having been acquired in the same recording session (this dataset contains the highest number of annotated pallets); and test set Viano3-5 which originates from several separate recording sessions. The second test set is more challenging and contains mostly difficult cases including over/under-exposed images; heavy glare; light artifacts. The composition of the sets is as follows: the training set contains 467 images and 891 labeled pallets (there can be zero or more than one pallet in each image); the test set Viano2 contains 7124 images and 9047 labeled pallets; test set Viano3-5 contains 224 and 356 labeled pallets. See Table 5.12 for a centralized view of the numbers presented here.

name	**nr. images**	**nr. pallets**	**nr. negative images**
Viano 2 training set	467	891	0
Viano 2 test set	7124	9047	1802
Viano 3-5 test set	224	356	44

Table 5.12: Composition of datasets acquired from Viano

Table 5.13 shows the detection accuracy on the two test sets using different configurations. Evaluation is performed on two test sets and enforcing two overlap criteria. The effect of adding new feature types is evaluated. We present test results using a boosted classifier with 100 and 1000 weak learners respectively. The number of negatives signifies the number of negative examples sampled from each training image during the learning phase. The training set can contain more than 1 million examples since positive samples are taken near the ground truth bounding boxes and negative samples are selected randomly from other zones. If we weigh the error on positive instances more by ω times, we can obtain a more precise localization.

Even though the shorter npd-linear feature has a good accuracy on the Viano2 test set, it performs worse on the harder test set. This is a possible case of overfitting since the training and test set from Viano2 are similar. Clear improvements can be seen with the new features and each additional feature improves the detection accuracy. Missed detections arise when the images are too dark, when the pallets are not fully visible or when false detections appear due to glare.

	Viano2		**Viano3-5**	
Features	**normal**	**precise**	**normal**	**precise**
100 weak learners + 100 negatives per image				
integral ftrs.	79.0 %	64.2 %	-	-
npd	80.6 %	65.1 %	80.9 %	40.1 %
npd-linear	84.6 %	77.1 %	74.2 %	30.3 %
npd+edge	90.3 %	83.4 %	81.7 %	45.5 %
npd+edge+lbp	97.1 %	90.2 %	87.7 %	46.0 %
npd+edge+lbp + $\omega = 10$	97.7 %	92.6 %	87.7 %	70.5 %
1000 weak learners + 1000 negatives per image				
integral ftrs.	92.0 %	75.4 %	77.0 %	38.0 %
npd+edge+lbp	100 %	94.9 %	93.5 %	65.7 %
npd+edge+lbp + $\omega = 2$	98.9 %	95.4 %	91.9 %	68.8 %

Table 5.13: Detection accuracy for multiple feature configurations

	Viano2		**Viano3-5**	
Configuration	**normal**	**precise**	**normal**	**precise**
100 weak learners + 100 negatives per image				
old features - depth 2	79.0 %	64.2 %	-	-
npd - depth 2	95.4 %	91.4 %	87.6 %	55.9 %
npd - depth 3	97.7 %	92.0 %	88.2 %	68.8 %
npd - depth 4	98.3 %	93.7 %	90.5 %	72.2 %
npd - depth 5	98.3 %	94.7 %	93.8 %	78.1 %
1000 weak learners + 1000 negatives per image				
old features - depth 2	92.0 %	75.4 %	77.0 %	37.9 %
npd - depth 2	100 %	94.7 %	93.8 %	57.3 %
npd - depth 5	98.9 %	94.9 %	97.5 %	64.9 %
2048 weak learners + 3 bootstrap rounds				
aggregate features - depth 2	99.4 %	46.3 %	85.4 %	25.8 %

Table 5.14: Detection accuracy for decision trees with different depths

In Varga et al. (2015) we compare the newly introduced pallet detection module with aggregate features to the previous approach from Varga and Nedevschi (2014). The detector that operates with **npd** features is also compared. The results with different detection module configurations are presented in Table 5.14. The previous method that uses boosted decision trees and integral features is shown as baseline for comparison. We also evaluate the influence of increasing the depth of the decision tree and our newly introduced features normalized pair differences (**npd**). According to the results, increasing the depth not only improves overall performance, it also has a positive effect on the detection accuracy for precise matches. This is essential for a more precise pallet localization. Increasing the depth only slightly increases training time and the execution speed for prediction. However, the number of parameters increases exponentially, the possible number of nodes is equal to $2^{d+1} - 1$, where d is the depth of the tree. This is is why we stop at depth 5. Increasing the depth of the decision trees also runs the risk of overfitting the training set. This is avoided by checking the error on an independent test set.

operation type	**nr. op.**	**nr. successful op.**	**percent**
loading - rack	103	103	100 %
loading - block	69	69	100 %
unloading - rack	82	82	100 %
unloading - block	69	67	97 %
total	323	321	99 %

Table 5.15: Field test results from Viano

Processing step	**Execution time [ms]**
Image rectification and undistortion	3
Candidate generation	10
Feature calculation	100
Classification	30
Non-maximum suppression and enhancements	0
Reconstruction and plane estimation	10
Total time	153

Table 5.16: Typical execution times for each processing step

5.2.4 Conclusions

The work presented in this part has been implemented and tested on two sites: a warehouse in Viano owned by Elettric80; a plant/factory in Bilbao owned by Coca-Cola Iberian Partners. Scientific results have been published during the system development in multiple publications: Varga and Nedevschi (2014) - describes the initial system, Varga et al. (2015) - shows modules used for unloading operations, Varga and Nedevschi (2016) - focuses on introducing the npd features for more accurate detection.

The tests show that the system is both accurate and relatively fast. System tests indicate that it is capable of providing precise enough information to the AGV for performing operations correctly. More testing and refinement is required before the system is fully ready.

5.3 Pedestrian detection methods

The current section presents different versions of pedestrian detection methods proposed by us. We start from the basic idea which is to use only a limited number of pedestrian scales and to avoid image resizing operation. Our system has been rewritten and optimized to attain over 20 frames per second execution time in the final version. It has also been extended with multiple channel types to achieve state-of-the-art results.

5.3.1 Pedestrian detection using reduced number of scales

Proposed solution

The main idea behind our approach for pedestrian detection is to combine successful candidate generation and to avoid unnecessary resize operations on images. The proposed RoI selector from section 4.2 can be applied to a pedestrian detection method to reduce the execution time. In general, approaches to pedestrian detection with sliding window require image resizing to detect pedestrians at different scales. We opt for a different route by performing all detection steps on the original image without resize operations. As demonstrated in the paper Varga and Nedevschi (2013a), this leads to a theoretical speed-up of up to a factor of 4.

For such an approach, the features must be calculable on variable sized detection windows in order to account for the different scales. Integral channel features are sums of rectangular regions of the image, proposed in Dollar et al. (2009a). If we normalize the features by their area, we obtain a scale independent representation of these features. Certain channels such as gradient magnitude are not scale invariant, i.e. regional sums of larger regions are not equal to regional sums of resized images (even if features are normalized by the size of the region).

We have chosen as features the same ones that have been established to perform the best for the task of pedestrian detection in Dollar et al. (2009b); Dollár et al. (2010). More precisely, the image channels on which we perform summations are: the three channels of the Luv image, gradient magnitude, 6 gradient orientation bins. This results in a total of 10 channels. Unfortunately, the descriptiveness of the HOG (Histogram of Oriented Gradients) features also means that they are not scale invariant, meaning that normalising by the area does not produce the same result as resizing the image and then performing histogram binning again. For this reason, we opt to define a few canonical scales and to train a separate classifier for each scale. Having separate classifiers for each scale eliminates the problem of scale invariance but introduces extra work for training, maintaining and predicting with multiple classifiers.

Our proposal is to work with only a limited number of scales. Standard approach is to split octaves into upto 12 parts and perform detection on each intermediate scale. Of course this leads to a loss in the tightness of the bounding boxes resulting from detection.

The idea behind limiting the number of scales is to decrease the computation time. What scales should we choose? Each scale is in direct correspondence with a bounding box height. In our initial publication from Varga and Nedevschi (2013a) we have selected target bounding box heights 64, 128, 256 and 512. Note, this corresponds to canonical scales of: 0.5, 1, 2 and 4 for a 128x64 detection window. These scales have been used in the literature in numerous works. However a sparse scale space with only a few target heights has not been explored before. A more principled method for choosing bounding box heights was developed in the followup paper Varga et al. (2014) - detailed in the next part. Classification is performed by a boosted classifier with two-level decision trees. In the earlier paper the Real AdaBoost implementation from OpenCV library is employed with 1000 weak learners.

Algorithm 15 formalizes the ideas presented above and describes the steps needed at detection time to obtain pedestrian bounding boxes. It is important to note, that feature calculation on the integral images is performed fewer times because of the reduced number of RoIs as opposed to calculating them for every region (step 5). This algorithm requires an already tuned region of interest selector and a trained classifier.

Algorithm 15 Pedestrian detection method with RoI selection

```
Input: Input image.
Output: Pedestrians as an array of rectangles and confidence values.
 1: Calculate channels for integral channel features.
 2: Apply RoI selection using Algorithm 10.
 3: Set detections = $\emptyset$
 4: for all RoIs do
 5:     Calculate the features from within the RoI
 6:     Classify the features using the appropriate classifier
 7:     if confidence $> \theta$ then
 8:         Add the RoI to the detections list along with the confidence value
 9:     end if
10: end for
11: Apply pairwise-max on detections
```

During the execution of the algorithm, the integral images for each image channel are calculated and stored (line 1). New we allow the RoI selection to filter out the majority of the candidates (line 2). The detection process continues only on the set of bounding boxes that are retained after the previous step (lines 4-10). As is usual, there are multiple detections clustered around the true object, this is resolved by performing a pairwise maximum selection based on classifier score (line 11).

Experimental results

We have evaluated our proposed detection method with RoI selection on the INRIA pedestrian benchmark. The training set contains 613 pictures with pedestrians, each picture can contain more than one pedestrian. The annotations are in the form of bounding boxes for each pedestrian. The negative set numbers 1218 images that do not contain pedestrians. It is one of the most widely used datasets for pedestrian detection evaluation. The initial negative samples for training the classifier are obtained by sampling each of the negative example images randomly for 10 bounding boxes of the required height. Also, a random resizing is applied before cropping the negative image to match the resizing operations from the positive examples. This is a common practice and it is referred to as bootstrapping.

First we discuss the results for tuning the parameters of the RoI selector. To measure the effectiveness of this part we establish two important criteria: speed-up and coverage. **Speed-up** is defined as the ratio between the total number of all possible candidates and the number of candidates remaining after RoI selection. This factor roughly reflects the expected speed gain since the execution time of the detection method depends linearly on the number of candidates. The **coverage** measure can also be called recall, it is the percentage of the positive examples that are retained. An ideal RoI selector should have 100 % coverage and a high speed-up factor to be effective. Also, the time consumed in during selection should be low enough to merit the introduction of this module. The results for different parameter settings is presented in Table 5.18.

This significance of the parameters are: *type* - filter applied for edge detection; σ - variance for Gaussian filtering; t_1 - threshold value for top detection; t_2 - threshold value for bottom detection. For Canny edge detection the parameter t_1 is the lower threshold and the higher is equal to $3t_1$.

By analyzing the results, we can draw the following conclusions. More smoothed images yield smaller coverage values because gradient magnitudes become smaller and this leads to rejection of more rectangles. This, however, simultaneously increases the gain in speed at the cost of false rejections. As expected, one must choose the appropriate kernel type and parameters by adjusting a trade-off between speed-up ratio and coverage. Also, the execution time of the edge detection method must be taken into consideration. For example, Canny edge detection takes a longer time.

The influence of the RoI selection module on the whole detection algorithm is illustrated by comparing the execution time of the whole system with and without this module. We can view typical execution times for small images, large images and the whole test set in Table 5.21. The general effect of introducing the module is a tenfold speed increase. Following standard procedure, we check the performance of each variant and plot the average miss rate versus false positives per image. A figure illustrating the table:roc curve for three variants is given in Figure 5.8. The figure shows no significant degradation in performance after introducing the RoI selection module.

Test unit	**Sobel**	**Canny**	**No RoI**
Test set	5 minutes	4 minutes	55 minutes
370x480 img	0.192 seconds	0.182 seconds	1.91 seconds
960x1280 img	1.032 seconds	2.895 seconds	33.02 seconds

Table 5.17: Comparison of execution times

Type	σ	t_1	t_2	d	**speed-up**	**coverage**
Sobel	0	100	2	32	46.52	0.98
	1	100	2	32	161.17	0.91
	2	100	2	32	430.21	0.83
Scharr	1	120	5	32	7.83	0.99
	2	120	5	32	10.05	0.98
Prewitt	0	100	2	32	148.41	0.94
	1	100	2	32	952.85	0.78
Canny	0	30	2	32	15.96	1.00
	1	30	2	32	23.90	1.00
	2	30	2	32	32.95	1.00
	2	30	5	32	144.21	0.99

Table 5.18: RoI parameter tests on the training set

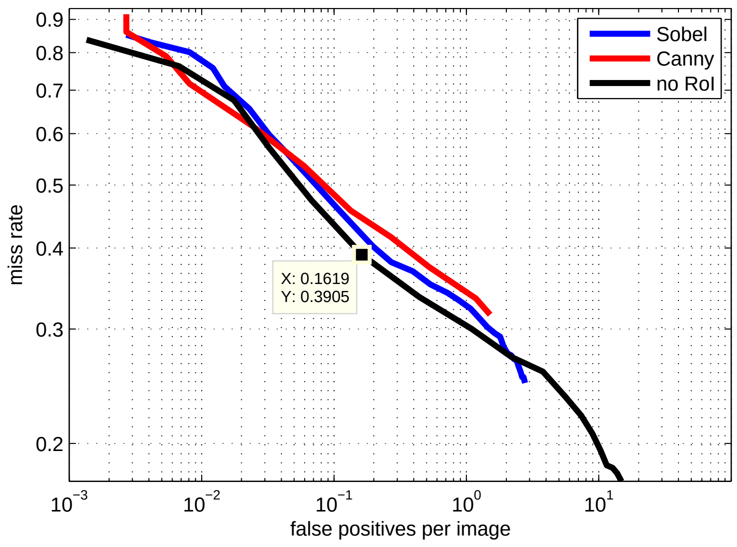


Figure 5.8: DET curve on the INRIA test set
A sample value is emphasized near the 10^{-1} false positive mark.

5.3.2 Pedestrian detection without image resize operations

Proposed solution

Our second version for the pedestrian detection system builds upon the previous one. Here, we only provide a short overview and state the differences. The main focus of this approach is to show how to achieve high-speed detection while maintaining detection performance as much as possible.

One of the key ideas is to detect pedestrians with only a reduced number of heights. Usually, detecting different heights entails resizing the image multiple times (constructing the image pyramid). This means that the number of scales is in one-to-one correspondence to the number of pedestrian heights. Our approach is opposed to the mainstream idea of using a dense scale space for image pyramid construction. Each image from the image pyramid is a rescaled version of the input image and has an associated detector of a given height. Typical detectors construct an image pyramid with up to 50-55 scales. The heights chosen here are based on statistical data acquired from the training dataset. We apply k-means clustering to obtain the 6 representative cluster centers for pedestrian bounding box heights. The number 6 was chosen because the detection rate was acceptable. A future analysis about how the detection performance is influenced by the number of heights will be performed. The difference compared to other sparse scale space methods (i.e. methods that limit the number pedestrian heights to consider) is that we do not have detection windows that are of the form $a2^n$, instead we select the representative centers based on the training data. Since detection performance degrades significantly at lower scales, we may omit smaller window heights for practical applications.

The proposed detection method uses a sliding window approach with the before mentioned 6 fixed window heights and a constant aspect ratio. Considering an exhaustive search at every position, scale and aspect ratio is not feasible nor necessary. In this work we opt for a candidate generation that accepts only candidate rectangles that have their center in the horizontal middle stripe. This selection can be motivated by studying the spatial distribution of detection window centers from the training dataset. This distribution has been observed on other datasets such as the Caltech Pedestrians Dollár et al. (2012).

Each sliding window height corresponds to a pedestrian at a specific scale and has an associated classifier. The classifiers are trained separately for each scale. Our aim is to totally eliminate image resizing and other operations on features. This is a key difference compared to other methods: Dollar et al. adjust the features based on scale in Dollár et al. (2010), while Benenson et al. Benenson et al. (2012) adjust the classifier. The detection window is moved to every valid position that is dictated by a region of interest selection method. Features are then calculated as sums of rectangular subregions for each candidate window and classification is performed. The sums of different image channels over a rectangular area can be calculated efficiently with integral images. The underlying image channels are Luv color-space, gradient magnitude and oriented gradient histogram with 6 bins. For feature extraction, we rely on a module provided by Dollar in Dollár (2006).

Key elements of this approach that help achieve high-speed and reliable detection are:

- No image resizing
- Smart and fast region of interest selection (candidate generation)
- Fast integral channel features calculation
- A cascade of boosted decision trees for classification
- Reduced number of pedestrian heights
- Custom implementation of all processing modules and code parallelization

The detection algorithm follows the well established canonical pipeline. It is composed of the following steps: preprocessing (resizing - not employed in this case, padding - when the image width is not divisible by 4, smoothing), region of interest selection (or candidate generation), feature channels extraction (Luv conversion, gradient computation, histogram bin aggregation), feature aggregation (rectangular region sums) only on the candidate regions, classification/prediction and non-maximum suppression.

An important aspect of the preprocessing step is how to treat images that do not have width divisible by 4. Since many operations run only if divisibility is ensured, we pad the images instead of cropping or resizing. This also helps to linearize the image in the memory. Note, that during the training phase many cropped small images are fed as input to the feature extractor. These result from clipping out only the pedestrian bounding boxes. This is why it is important to treat irregular sized images carefully. We perform no image resizing, although for larger input images this could be included to reduce the search space. Image smoothing is moved to the feature extraction phase, and gaussian bluring is replaced by a faster triangular filtering as in Dollar et al. (2009a).

Region of interest selection provides the candidate rectangles from which the features are extracted and classified. It is essential to restrict the number of these candidates to reduce the workload of the following modules from the pipeline. For this step, we have three main options. The first option is to use all possible bounding boxes with a given stride, fixed aspect ratio and height restricted to a set of values. The second option is to admit only the rectangles whose centers lie in the central horizontal stripe of the image. This is a heuristic that is easy to implement and it is deduced from the measurements from the pedestrian dataset (see section Location-based region of interest selection). The third option is to select candidate regions based on the edges in the image (as in Varga and Nedevschi (2013a)). Here we opt for the second approach because it is sufficient and assures high coverage. The selected pedestrian heights are: 60, 92, 136, 212, 340 pixels. The stride is set to 4 or 8 pixels, the fixed aspect ratio is 0.43 (width over height).

For feature extraction we rely on the Integral Channel Features module provided by Dollar in Dollár (2006). This was adapted from Matlab+mex to our C++ implementation that uses classes from the OpenCV 2.4.5 library. Feature extraction is fast because of clever usage of integral images, parallel computing of the channels and SSE instructions. The present module has the standard configuration of channels: luv, gradient magnitude and 6 gradient orientation bins.

Parameters have been set to: 5000 random features with an area of at least 25; shrinking factor of 4, triangular smoothing which is equivalent to a Gaussian blur with a sigma of 1. See Dollar et al. (2009a) for more details about the parameters.

Classification is performed with an ensemble of 5000 weak learners. Both the training and predicting have been reimplemented to optimize prediction speed. Discrete boosting is applied with two level decision trees. This option is motivated by Dollár et al. (2010), where the authors show that the boosting method does not have a large impact on the detection rate and that 2 level decision trees are the best for this task. The splitting criterion for the decision tree is the local training error, i.e. the best split is the one that minimizes the training error. Rejection thresholds for the cascade classifier are obtained via the direct backward pruning method applied on the training set. It was proposed in Viola et al. (2005b). In most cases it is preferred to obtain the thresholds on a validation set rather than the training set. When this validation is not performed, the rejection thresholds should be lowered in order to prevent the quick rejection of unseen positive examples. A simple recalculation of the thresholds can be performed in order to obtain rejection thresholds for any end threshold (see Viola et al. (2005b)).

At detection time all rectangles obtaining a classification score higher than a given threshold θ) are retained for non-maximum suppression. For every two overlapping rectangles, we retain only the one with the higher score. The overlap can be determined in multiple ways. Here, we use the formula: $o_{min} = \frac{R_1 \cap R_2}{min(R_1, R_2)}$, where R_1 and R_2 are the areas of the two rectangles, and the numerator contains the area of their intersection. This eliminates smaller bounding boxes from inside larger ones because in this case the overlap is high due to the min function from the numerator. For the same reason, it more aggressive than the usual alternative: the PASCAL VOC-type overlap measure $o = \frac{R_1 \cap R_2}{R_1 \cup R_2}$. This is why lower thresholds are suitable for this method. Another alternative that is useful for large number of detection windows is to perform a greedy elimination. The threshold for considering overlap is set to 0.4.

At the training phase, we repeat the same operations at each scale to generate classifier models. We first process the positive examples. Each bounding box of a person is cropped from the original image and resized with bilinear interpolation to the size of the current detection window (e.g. 60x26px). Adjustments are mare to center to the bounding box and to preserve the original aspect ratio (resizing factor is the same along the width and the height). To increase the diversity we also process the horizontally mirrored image. The 5000 features are calculated and saved for later with a positive label. These features are randomly generated rectangles from inside the detection window area. To obtain negative samples, we select 5000-7000 random crops from the list of images not containing any pedestrians. A uniform sampling is performed, i.e. if there are 5000 images, then one random rectangle is chosen from each, if there are 20000 images (a typical case), then one random rectangle is chosen from every 4th. For every random window, we calculate the features and save them for later with a negative label. Once all examples are processed the file containing the features can be fed as input to train the classifier.

Next, we perform bootstrapping. Call the initial model as model-x-0, where x stands for the height and 0 stands for no bootstrapping. By applying the classifier on negative images, we obtain at first many false positives. At each stage we retain 7000 of these false positives and append their feature vectors to the training file. After retraining, the classifier model-x-1 is obtained. This process is repeated 2-3 times until there are only a few false positives or no change is observed. Limiting the number of examples is essential to keep the training set balanced and it also helps reduce redundancy. We have observed that the classifier for the smallest scale produces false positives even after 4 rounds of bootstrapping. This is a clear signal indicating the failure of the classifier to learn a good model from the training data.

Next, we present some details about how the presented approach can be implemented on a mobile device such as a tabled or smartphone. Having the algorithm already implemented in C++, developing an Android application is possible since we can easily integrate the existing native code with Java code by making use of Java Native Interface (JNI) framework. The algorithm was ported on an Android mobile device having the following characteristics: NVIDIA Tegra 3 T30L chipset, Quad-core 1.2 GHz ARM Cortex-A9 CPU, NEON instruction set support. We have used JNI calls in order to capture the frames in a Java environment and send them for a faster processing in a native C++ environment. For a better performance we have used OpenCV 2.4.5 for Tegra 3 library accelerated for ARM NEON architectures and we took benefit from the multi-core processor by parallelizing the code with Qualcomm MARE Parallel Computing Library. We took advantage of the **pfor_each** functionality provided by MARE in order to substitute the **#pragma omp** OpenMP precompile directives used in our PC version. For handling the reading and writing of the training files we have used the AssetManager class provided by Java. This allows us to compress the files and perform one time writing in the internal memory of the portable device upon installation of the application and extract the data whenever we need during running the algorithm.

Parameter	**Description**	**Value**
N	number of weak learners	1000
M	number of features	5000
d	stride (grid step)	8
ρ	aspect ratio	0.43
h	pedestrian heights	{60, 92, 136, 212, 340}
T_o	overlap threshold for NMS	0.4
B	additional bootstrap samples at each stage	7000

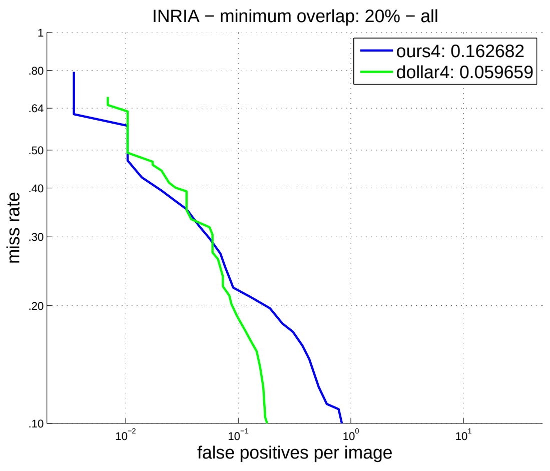
Table 5.19: Relevant parameters of the detection algorithm

Experimental results

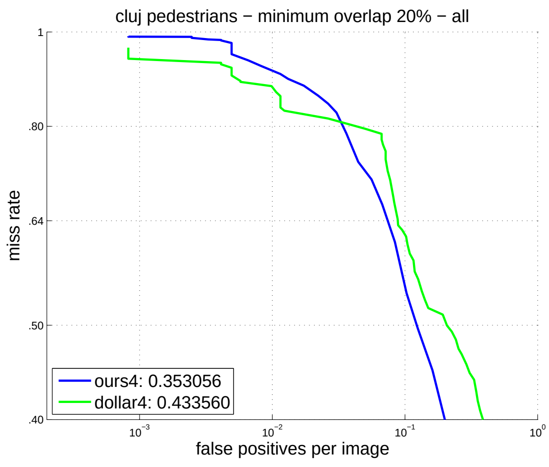
A labeled pedestrian dataset was gathered for the purpose of training and evaluation (see Figure 3). This set was captured using a smartphone placed on the windshield of a vehicle driving through the city. The purpose of this new dataset is to reproduce as closely as possible the real situation where the detection method would be applied. The dataset is available in both video (mp4) and image (png) format. Total video length is around 15 minutes. The video framerate is of 30fps. All images have 640x480px resolution (landscape). The total number of images is 27666. The set is broken into independent sequences based on separate recordings and contains mostly frames with pedestrians. We provide the pedestrian bounding boxes for each image in the dataset in a simple text file format. Note, that some pedestrians are unlabeled. This is the case only for very crowded scenes and for persons that are far away and thus appear to be very small. There can be up to 10 pedestrians in a single frame.

The pedestrian bounding boxes are divided into a nonoverlapping training set and test set. The initial training set contains 3205 bounding boxes with pedestrians and 219 negative examples (images with no persons) obtained by labeling every 10th frame. In order to enrich the dataset, we have interpolated the detections and have generated an interpolated training set consisting of 27318 pedestrian rectangles. Due to the reduced number of negative examples, it is recommended to augment the negative training set (images not containing any pedestrians) from another source (e.g. INRIA, Caltech or general images). For our training procedures we have augmented the examples from this dataset with a portion of the images (around 4500) from the Caltech Pedestrian Dataset that do not contain any pedestrians.

We provide some interesting and relevant statistics for the dataset. The distribution of the heights of the bounding boxes is given in Figure 1. Approximately 88% of the bounding boxes are smaller then 140px in height, while the predominant height is 90px. The width of the bounding boxes follows an exponential distribution shown in Figure 1. The most likely aspect ratio is 0.43, see Figure 2. An important statistic about the frequency of the centers of the bounding boxes reveals that only a negligible part (0.3%) of them are positioned outside the 200-300 horizontal stripe (see Figure 4.1). Also, due to the placement of the recording device, and because the pavement is on the right side, pedestrians occur more frequently on the right side.



(a) INRIA



(b) Cluj Pedestrians

Figure 5.9: DET cruves on pedestrian benchmarks

area	stride = 8	stride = 4	stride = 3
overlap = 0.40	0.361	0.353	0.367
overlap = 0.65	0.373	0.367	0.375
overlap = 0.70	0.384	0.372	0.379

Table 5.20: Area under the DET curve for different parameters on the Cluj Pedestrians dataset

To evaluate the execution time of each part of the system, we measure at least 1000 times the speed of each module. Table 5.21 summarizes the measurements that we have obtained. The bottleneck of the pipeline is the feature extraction module where heavy optimization has been performed, even so it is the slowest part. A further reduction of the number of the candidates could help drastically improve the speed. The number of candidates is around 2000-3000 depending on the scales. The estimated execution time (disregarding the visualization part) is 23.67 frames per second (around 42 milliseconds per frame). We compare the speed gain with our previous version from Varga and Nedevschi (2013a): 180 milliseconds on an image with a resolution of 370x480px would result in approximately 311ms running time on a 640x480px image, thus the speed gain is: $\frac{311}{42} = 7.4$. Time measurements on Android for a 640x480px resolution indicate a 1.21 FPS average frame rate (around 823 milliseconds per frame). This is the parallelized Android version which has an overall speed gain compared to the single threaded version of: $\frac{1781.89}{823.0}$ = 2.16 ($\frac{1491.29}{739.53}$ = 2.01 - feature extraction; $\frac{230.60}{78.73}$ = 2.92 - classification).

Step	PC [ms]	Android [ms]	Android + MARE [ms]
RoI selection	0.131	0.29	0.29
Feature Extraction	38.3	1491.29	739.53
Classification	3.38	230.60	78.73
NMS	0.096	0.13	0.13

Table 5.21: Execution time of different modules

The reduced execution time of the algorithm is due to the fact that almost all processing steps have been specifically rewritten for the detection task. Our implementation is in C++, compiled with Visual Studio 2010 compiler with OpenMP multithreading features enabled. Other settings include: fast code optimization enabled, fast floating point model, omit frame pointers. OpenCV 2.4.5 is the chosen library for image processing functions.

The workstation used to test our system has the following parameters: Intel Core i7 CPU, 3.5 GHz, 4 cores, 8 logical processors, 16 GB RAM. Most of the relevant operations are parallelized to use the processing power of the CPU efficiently.

5.3.3 Lazy Feature Extraction (LFE)

Proposed solution

We turn to the description of the third version of our pedestrian detection system. Our proposed approach relies on the well established combination of integral channel features and boosted classifiers. We focus on detection optimization and not on feature extraction. By delaying feature calculation at detection time to the last moment we hope the reduce the execution time while maintaining the same detection accuracy as the brute force approach. This optimization also allows us to consider a larger feature pool which should help the classifier to learn a better model.

More concretely, we use integral channel features and boosted two-level decision trees. Our method refines the work of Dollar et al. and uses the same feature channels. Thus, we consider ACF from Dollár et al. (2014) as a baseline. Even though we present our method using the features from ICF Dollar et al. (2009a), a precursor to ACF, any other features that can be separately calculated online can be employed.

Rectangular sums of image channels are fast to calculate but if the feature pool is very large (more than 10k), real-time detection is prohibited. We believe we can circumvent this drawback and still reap the benefits of using an enriched feature pool for classification. For convenience, in this work we will refer to the image channels from which the sums are calculated to as simply **channels** and to each rectangular sum as simply **features**. Each feature is specified by the channel on which it is calculated and the rectangular area of its support region.

Using cascaded boosted classifiers permits us to delay feature calculation until the very last moment: when it is needed for decision making. Specifically, suppose a boosted classifier model is available with M binary trees as weak learners. Each weak learner only needs a subset of the features to make a prediction. So we calculate only the features that are required inputs for the current weak learner. Typically, the detector is given a large number of candidates and eliminates most of them by cascading. If any partial hypothesis values (scores) are below the rejection threshold, the example is eliminated. This presents another opportunity to delay feature extraction since at each stage we only need the features for the candidates that have not been eliminated. This approach is new and it is in contrast to traditional methods which calculate all features at once for all candidates, which is clearly not necessary.

At detection time we perform the following operations. First, the image channels are computed at different scales. For this we employ available software provided by Dollar et al. Dollár (2006). Next, we use Algorithm 16 and the trained model to perform detection. Finally, pairwise non-maximum suppression is applied from the toolbox to retain only the bounding boxes with the highest confidence in case of overlaps.

Definitions of the terms appearing in Algorithm 16 : calculateImageChannels() is a function responsible for creating the necessary image channels; generateCandidates() provides the list of all candidates; calcFeatures() performs only the rectangle sums that are needed for the current stage and only for current candidates; $tocalc_i$ is a list containing the features that must be calculated for each stage - available from the training procedure; $tree_i$ is the model for the i-th decision tree that; rej_i is the rejection threshold for stage i, all candidates with scores less than

Algorithm 16 Lazy feature extraction and prediction

```
Input: color image, boosted classifier model with $M$ trees
Output: a list of rectangle and confidence value pairs
 1: $C$ = calculateImageChannels()
 2: $L$ = generateCandidates()
    $L = \{(r_i, s_i)|i = 1 : N\}, s_i = 0$
 3: $L_2$ = {}
 4: for i=1:$M$ do
 5:     $F$ = calcFeatures($C$,$L$,$tocalc_i$)
 6:     for j=1:size($L$) do
 7:         $s_{new} = s_j + predict(F, tree_i)$
 8:         if $s_{new} > rej_i$ then
 9:             $L_2 = \{L_2, (r_j, s_{new})\}$
10:         end if
11:     end for
12:     $L = L_2$
13:     $L_2 = \{\}$
14: end for
```

this number are rejected (cascading) and no further calculations are made for them; predict() uses the model of a decision tree to make a prediction based on some features F.

A more detailed explanation of the algorithm is given next. At detection time we first calculate the image channels at different scales. We start with a list of all possible candidates and their scores - initially set to 0. We remove negative instances from this list iteratively at each stage of the classification. We do not actually work with rectangles instead only with indexes to rectangles (pointers). Only a limited number of features are needed for a weak learner to make a prediction. Note, that this enables us to use only a small memory zone for holding the feature values. In the presented algorithm some features are recalculated. This can be circumvented easily by allocating a larger memory zone for feature storing but this also introduces another layer of lookup operations. After the features are ready, the classifier uses the current weak learner to update the score of each candidate. If the new score is above the rejection threshold for the stage, the candidate remains (it is added to a temporary second list L_2 which will be swapped with L - this is efficient because the lists L and L_2 can be implemented as simple preallocated arrays). Otherwise the candidate is eliminated and no further features are calculated for it in the later stages. At the end we are left with all the candidates that have a sufficiently large score.

Training procedure

Classifier training is based on well established procedures for boosted classifier training from the literature presented in Dollár et al. (2014); Benenson et al. (2012). We start with positive examples and 5000 random negative examples. Several bootstrapping rounds successively add a maximum of 5000 false positives and retrain the classifier, however the number of negative examples cannot increase above 10000. During training all the features are calculated and stored.

This means that training with a large number of features will be computationally expensive. Furthermore, it requires a significant amount of memory (see Implementation details). After each training stage, the features that are necessary for each decision tree are established (the optimal features to make the required splits). At detection time only these features need to be calculated and stored, which in turn reduces the space complexity for detection from $O(ND)$ to $O(N)$ - for 2-level decision trees it is exactly $3N$ floats - where N is the number of candidates, and D is the number of features. Memory requirements are an issue for training sessions with many features (over 100k) and a large number of training examples (over 15k).

For model training we use our custom implementation of Discrete AdaBoost with binary trees as weak learners. The key difference between our implementation and the classifier from the toolbox is that we do not perform feature quantization. This increases the memory requirement because we need to store all features and because we need to keep track of the sorted indexes. However, it increases detection accuracy, as shown by the experiments. It also entails a longer training time. Another difference is that we have different rejection thresholds at each stage of the boosting much like in Bourdev and Brandt (2005) and Viola et al. (2005b). Rejection thresholds are estimated on the training set by setting them to the minimal score of all the positive examples for each stage. There may be a need to adjust these thresholds for detection on another dataset to prevent premature elimination of positives. These values greatly influence the speed and the accuracy of the classifier so it is crucial that they are learned properly.

Time complexity

Next, we analyze the theoretical time complexity and the speed gain from the proposed optimization. Let $T(N)$ be the time required to perform detection starting from N candidates. The baseline time complexity for the brute force algorithm is of $O(ND)$, where D is the number of features and is typically larger than 5000. This signifies precalculating everything before prediction is performed. Using lazy feature extraction we can exploit the fact that candidates are gradually eliminated. The number of candidates at each prediction stage t is a decreasing function $f(t)$ - note, that f depends on N but for convenience we will hide this dependence. Because prediction with a single decision tree consists of a few memory look-ups and comparisons, we can assume it takes a constant time c. Let M be the number of weak learners which is the same as the number of stages. Using the optimizations the running time will have the following form:

$$T(N) = c\sum_{i=1}^{M} f(i) \leq c\int_{t=0}^{M} f(t) = c(F(M) - F(0)) \tag{5.15}$$

where $F(t)$ is a primitive function of $f(t)$, if N is considered a constant.

We analyze two forms for the function. If we assume a linear decrease in the number of candidates, i.e. $f(t) = N - at$, then the time complexity of the algorithm becomes:

$$T_{lin}(N) \leq c(NM - a/2 \cdot M^2) = O(NM) \tag{5.16}$$

Even if it is not apparent from the previous formula, this running time is always smaller than the original baseline because only a subset of the calculations is executed and normally $M < D$.

For our experiments typically $M = 2048$ and $D > 100k$.

More realistically, if we assume an exponentially decaying number of candidates then $f(t) = N \cdot exp(-at)$ and the running time complexity has the form:

$$T_{exp}(N) \leq c(-N/a \cdot exp(-aM) + N/a) = O(N) \tag{5.17}$$

This entails that the running time will be dominated only by the number of starting candidates and not by the dimension of the features D or the number of weak learners M. This result is justified in the experimental results section.

Implementation details

Our implementation is in Matlab making use of Dollar's toolbox and feature extraction that are available online - see Dollár (2006). The crucial parts are written in C++ and compiled with Visual Studio 2012 compiler with OpenMP multithreading features enabled (for training only). Other settings include: fast code optimization enabled, fast floating point model, omit frame pointers.

For memory efficiency we store all features as floats (4 bytes) and for instance indexes we use unsigned short (2 bytes). Optimization for training phase involves presorting the training instances according to each feature and saving the indexes. After this, all operations have a time complexity of at most $O(ND)$ (number of instances times the number of features). The most time consuming operation is finding the optimal threshold. This can be split into independent operations and parallelized. For detection we need to store the feature description (channel index and rectangular area) for each weak learner, i.e. $5 \cdot 3 \cdot M$ integers (4 bytes) for the M weak learners. We preallocate a static reusable memory zone for storing the calculated features and the integral image. This is done in Matlab as a field for the classifier model structure. Peak memory consumption is at 31GB when training with 145k features on Caltech.

Experimental results

We evaluated our detection algorithm on two relevant pedestrian detection benchmarks: INRIA - Dalal and Triggs (2005) and Caltech-USA - Dollar et al. (2009b). We abbreviate the method presented here as **LFE** standing for **lazy feature extraction**. Methods are scored based on the log-average miss rate using the *dbEval* script from the provided toolbox for the Caltech dataset.

Our approach compares directly to ICF Dollar et al. (2009a) since it uses the same features. But ACF Dollár et al. (2014) is an optimized version and the source code is provided at Dollár (2006), so we consider this as baseline. We will show that our modification outperforms the original method and can achieve above 30 fps depending on rejection threshold settings. The improvement comes from the possibility of extending the number of features and the number of weak learners without significantly increasing the execution time.

To extend the feature pool available from the ACF (this consists of 4x4 aggregated squares), we test several options. First, we augment with random rectangular sums in the same manner as in ICF. Second, we test all rectangles with a given maximal dimension. We always include the original square features. Our tests show that the latter approach enhances detection performance. This happens only when there are a large number of weak learners and the training set is sufficiently large. A large feature pool can enable the classifier to overfit the training data, that is why in this case the training set must be sufficiently large.

In the following we give specifics and define notations regarding the extended feature pool. Let a feature be described by the 5-tuple: (i, x_1, y_1, x_2, y_2), representing channel index, left, top, right and bottom of the rectangular support region. Let the notation m_2 indicate the set of all rectangles satisfying $x_2 - x_1 \leq 2$ and $y_2 - y_1 \leq 2$. m_{12}^2 refers to all rectangles satisfying $x_2 - x_1 \leq 12$ and $x_2 - x_1 = 0$ mod 2, similarly for y. If $x_1 = x_2$ and $y_1 = y_2$, then the feature represents a pixel from the channel. The number of features for some of the tested configurations are: $|m_0|$ = 5120 (baseline); $|m_2|$ = 78120; $|m_5|$ = 143370; $|m_{12}^2|$ = 127400. Note, that the size of these sets depends on the shrink factor, i.e. integer downsampling amount for channels.

Test1: Our training procedure provides better models for smaller number of weak learners than the baseline detector, however training takes longer (15 minutes for 2048 weak learners on INRIA with 4 rounds of bootstrapping). The principal difference is that our training module does not perform feature quantization and thus uses more memory than Dollar's. We compare the classifier generated by the toolbox with our classifier for different number of weak learners, the results are shown in Figure 5.10. The models are the outputs of the stages of the bootstrapping process. Our classifier outperforms the baseline, with larger improvements at smaller number of weak learners. LFE 2048 uses the same features as the baseline and achieves a score of 17.11 % at 28 fps, this shows that we maintain the accuracy and the speed of the baseline.

Test2: Figure 5.11 compares our detector with the state-of-the-art results on this dataset. We achieve a log-average miss rate of **15.71% at 26 fps**, a performance that improves 1-2% over the baseline. The optimal model uses a 128x64 detection window, 2048 weak learners, rectangular features of maximum dimension 5 (m_5) amounting to a total of 143370 features. At training we reduce the estimated rejection thresholds by 1 to avoid overfitting. At evaluation we change all rejection thresholds to a constant value of -4. By raising this value one can adjust the score/speed tradeoff (e.g. at a constant threshold = -2 the score is 16.95 at 34 fps).

Test3: The training set from Caltech is sufficiently large to avoid overfitting with a large feature pool. Our best configuration in terms of log-average miss rate obtains **35.30% at 7.13 fps**. Key to obtaining this result is a 64x32 detection window; reducing the shrink factor to 2 (which also increases the time needed to compute the channels to 41 ms / 24 fps hindering fast detection); more weak learners (4096) and more features (m_{12}^2: 127400 rectangles with maximal dimension of 12 and dimensions divisible by 2). This result shows that features can be learned automatically as opposed to the method from Zhang et al. (2014) (training for this model took 9.4 hours). Figure 5.13 compares LFE with state-of-the-art methods. Note that better performing methods employ additional information. InformedHaar from Zhang et al. (2014) has a running time of 1.6 seconds, ACF-Caltech+Nam et al. (2014) utilizes an extended training set, LDCF Nam et al. (2014) does not specify runtime but it requires at least 0.5 seconds to decorellate the features, Katamari Benenson et al. (2014) employs other features, motion information and person-to-person patterns. SpatialPooling Paisitkriangkrai et al. (2014b) has a running time of 2 seconds. Our goal was to improve on the original algorithm and the approach presented here can be used in conjunction with the previous techniques to enhance detection quality.

Test4: Increasing the feature pool size is beneficial on the Caltech dataset. Some relevant results are presented in Figure 5.12. Adding random features (rand 5k and 10k) did not show any promising results, sometimes performing worse than the baseline. More extensive testing was done using rectangular regions of different maximal dimensions (m_4, m_5, m_{12}^2). In general, better performance is obtained when the admissible rectangles are larger.

Test5: We have collected data about how many candidates remain after each stage of classification. We plot the average number of candidates on a log scale calculated from multiple 480 x 640 pixel sized images. Figure 5.14 shows the expected number of candidates decreases with different rejection threshold settings. The rejection threshold can be fixed to a negative constant or the rejection thresholds can be estimated during the training process producing a variable threshold for each stage. The log-plot shows that in all cases after a few initial stages the candidate pool shrinks rapidly so we can assume an exponential decay. This means that the running time is influenced only by the initial number of candidates.

Test6: To evaluate the execution time of the detection, we measure the time needed to produce detections on 100 images, each image having the standard size of 640x480 pixels. The speed and the accuracy of the detector depends highly on the way the rejection thresholds are set. Table 5.22 summarizes our measurements and compares it to the baseline providing also the average miss rate on the Caltech dataset. Columns are: method name, number of weak learners, rejection threshold, shrink factor, stride for candidates, number of features, miss rate and frames per second.

When the feature pool is increased, the results improve at almost no penalty for the execution time. Indeed, a better feature pool may actually lead to a faster rejection of more candidates. Execution time is virtually unaffected by the number of weak learners. The speed/error trade-off is controlled by adjusting the rejection thresholds. Setting the shrink factor to 2 (instead of 4 as in the baseline) decreases the frame rate because channel calculation takes longer in this case (41 ms / 24 fps). A shrink factor of 2 basically means working with features aggregated on 2x2 squares.

All calculations performed during detection are done on a single CPU core. Code parallelization inside the detection is not worth it since initializing the thread pool already costs more than most of the code. If we eliminate cascading (set the rejection thresholds to $-\infty$), and calculate all the features, the execution time increases significantly. Typical times needed for each part are in case of shrink size 4 features: channel calculation 19ms (provided by the toolbox), integral image calculation 2.7ms, setup and lookup table creation less than 1ms, interleaved lazy feature extraction and prediction 8.4ms.

The workstation used to test our system has the following parameters: Intel Core i7 CPU, 3.5 GHz, 4 cores, 8 logical processors, 32 GB RAM. Most of the relevant operations from the training phase are parallelized to use the processing power of the CPU efficiently. Speed measurements are provided in the Experimental Results section. For comparison, on our machine ACF runs at 32 fps.

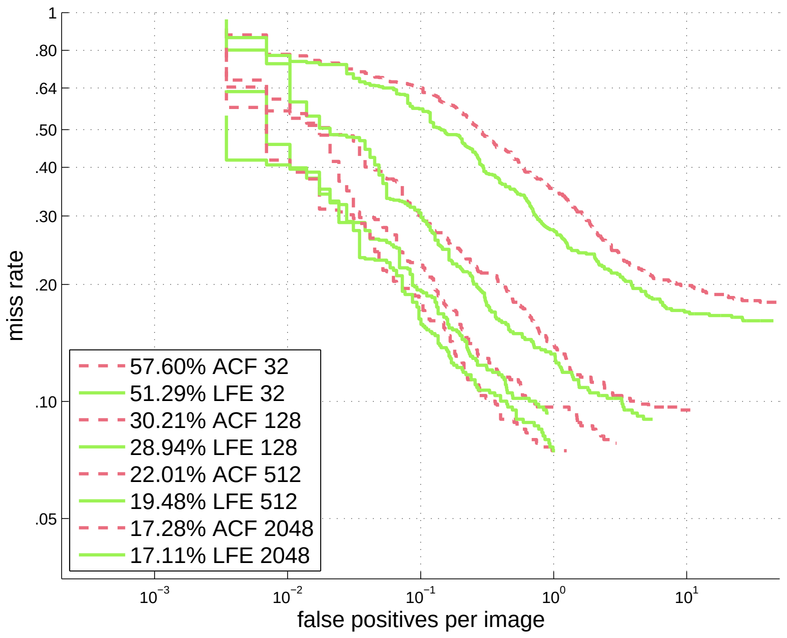


Figure 5.10: ROC curve and log average miss rate on INRIA Showing models with 32, 128, 512 and 2048 weak learners.

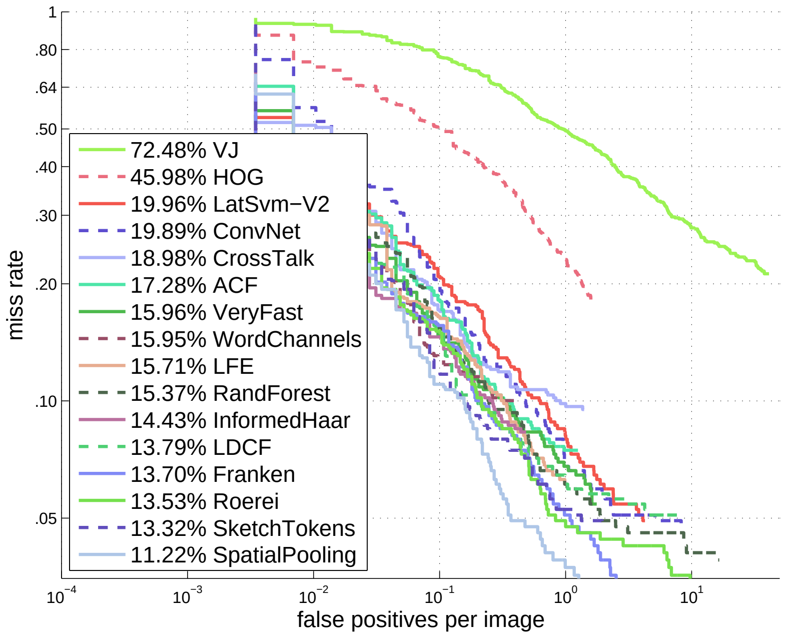


Figure 5.11: ROC curve and log average miss rate on INRIA Comparison with state-of-the-art methods.

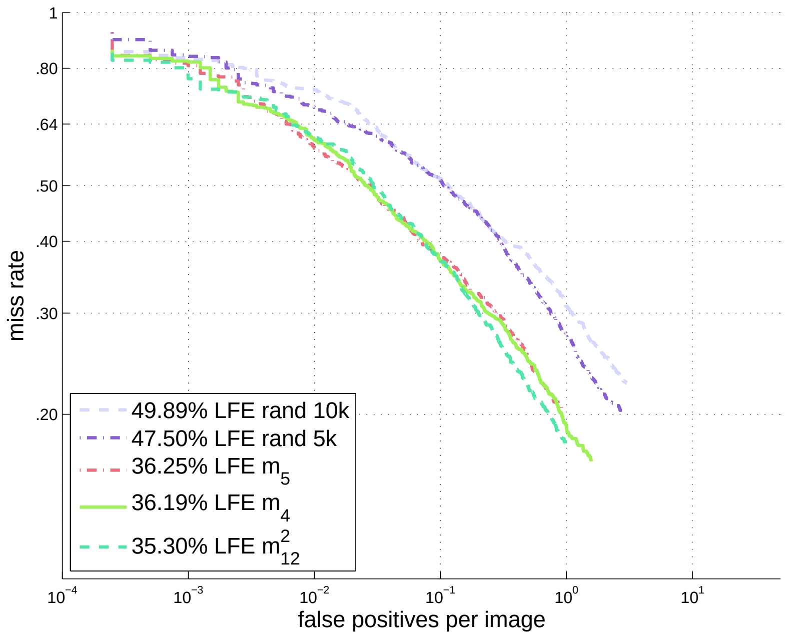


Figure 5.12: ROC curve and log average miss rate on Caltech-USA
Different feature pool configurations. Rand 5k stands for 5000 additional random rectangles.
Classifiers with added random features use 2048 weak learners and shrink size of 4.

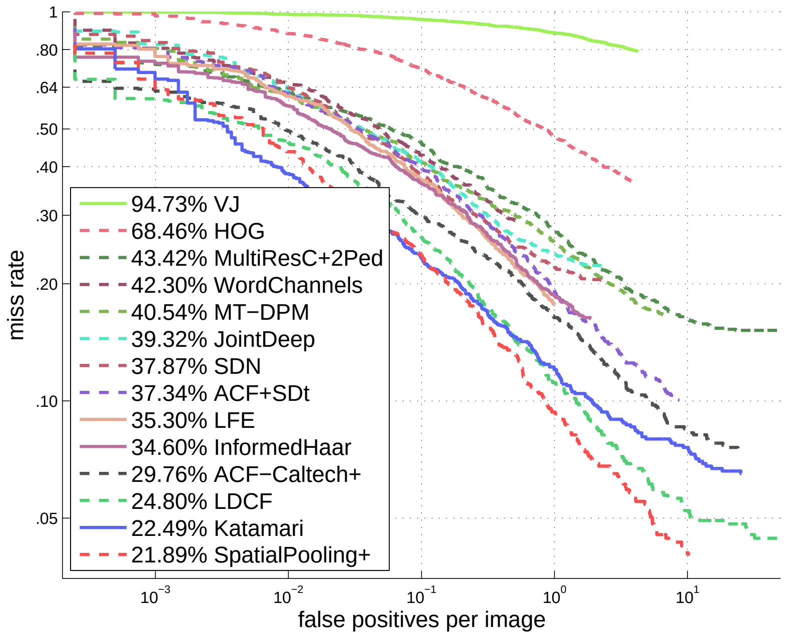


Figure 5.13: ROC curve and log average miss rate on Caltech-USA
Comparison with state-of-the-art methods.

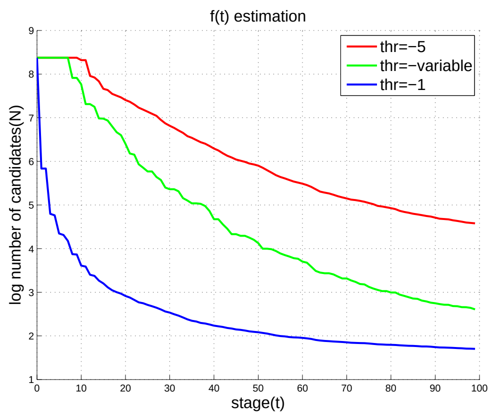


Figure 5.14: Log-plot of the average number of candidates after each cascade stage

n	M	**rej**	**shr**	**str**	**# f**	**mr**	**fps**
ACF	2048	-1	4	4	1280	44.22%	31.63
LFE	2048	-1	4	4	1280	45.40%	39.81
LFE	2048	-2	4	4	1280	44.69%	30.01
LFE	2048	$-\infty$	4	4	1280	44.78%	0.59
ACF	2048	-1	2	2	5120	39.59%	14.24
LFE	2048	-2	2	2	m_0	39.64%	7.42
LFE	2048	-2	2	2	m_2	40.96%	9.27
LFE	2048	-2	2	2	m_4	39.27%	10.12
LFE	2048	-2	2	2	m_{12}^2	36.75%	10.88
LFE	4096	-2	2	2	m_{12}^2	35.69%	10.88
LFE	4096	-3	2	2	m_{12}^2	35.30%	7.13

Table 5.22: Execution times and log-average miss rate on Caltech.

5.3.4 Multimodal Multiresolution Filtered Channels (MM-MRFC)

Proposed solution

In this part, we present a pedestrian detection method proposed in Costea et al. (2017) relying on the standard sliding window detection paradigm. The detector employs the classical features introduces in Dollar et al. (2009a). To enhance detection performance new feature channels are proposed: 2D and 3D context channels, geometric and symmetrical channels. These channels exploit multisensorial perception information from color, motion and depth. The visual information can be extracted from cameras. Depth can be obtained from laser point clouds when such a sensor is available or from stereo reconstruction. Motion information can be obtained from analyzing adjacent frames from the video stream.

Our personal contributions for the paper are: the study of scale invariance and feature value correction; 3D point cloud alignment to road plane. In the following we summarize the contributions of the paper and describe personal contributions in more detail.

Multiresolution channels were introduced in Daniel Costea and Nedevschi (2016). The work proposed to successively filter different channel types using a 3x3 box filter and applying a horizontal and vertical different filter on each result. The purpose of such an operation is to capture edges at different scales. Since the filters involved are simple, this can be applied at minimal computational costs.

2D context channels capture information about the 2D location of objects. This channel enables the classifier to learn constraints on the vertical and horizontal positions of objects. The three 2D context channels are: vertical, horizontal and symmetric-horizontal. Vertical channels at each position are simply equal to the current image row divided by the image height, horizontal channels are equal to the current image column divided by the image width, symmetric channels are equal to the distance from the middle column divided by the half of the image width.

Symmetrical channels are designed to have high response for vertically symmetric regions defined on specific ranges. An additional channel is generated which is the sum of symmetry channels for all range values. The channel values are calculated for each position (x, y) and range r using the horizontal derivative D_x:

$$S_r(x,y) = \sum_{i=r/2}^{r} \left(\frac{D_x(x-i,y) - D_x(x+i,y)}{D_x(x-i,y) + D_x(x+i,y)} \right)^2 \tag{5.18}$$

3D context channels encode 3D information obtained from the point cloud. A superpixel segmentation of the image is used. All points are projected onto the image. We can assign X,Y and Z values to each superpixel according to the points that fall inside. This information allows the classifier to distinguish between near and distant objects.

To obtain a more relevant representation we align the point cloud so that the y axis encodes height above the ground plane. For this purpose, we estimate the ground plane using RANSAC plane fitting on the point cloud. A rotation matrix is found which aligns the y axis to the plane normal $n = \begin{pmatrix} n_x & n_y & n_z \end{pmatrix}^t$. The steps to obtain the matrix are given below. The final rotation matrix R is composed of three rotations, each designed to eliminate a component from the normal vector. $rot(angle, axis)$ return a rotation matrix with angle around the given axis.

$$\begin{aligned}
\alpha &= -atan2(n_x, n_z) \\
R_a &= rot(\alpha, y) \\
u &= R_a \cdot n \\
\beta &= -atan2(u_z, u_y) \\
R_b &= rot(\beta, x) \\
v &= R_b \cdot u \\
w &= R_b \cdot R_a \cdot \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^t \\
\gamma &= -asin(||w \times \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^t ||) \\
R_g &= rot(\gamma, y) \\
R &= R_g \cdot R_b \cdot R_a
\end{aligned} \tag{5.19}$$

After alignment the y axis can be used to generate a binary channel by thresholding on a certain value to distinguish between objects on and above the ground plane.

Geometric context channels leverage superpixel segmentation information. From the previous step we have assigned positions to each superpixl. Superpixels are grouped together if they are less than 0.5 meters apart. After the grouping we form channels encoding the dimensions (heigh, width and area) of each group.

Scale correction

Ideally, when computing classification features for pedestrians, the feature values should not depend on the size of the pedestrian. The scale invariance of the classification features is lost due to the use of a single image feature scale and single classifier model for all pedestrian scales. Enabling scale invariance should further increase the robustness of classification.

The features in this work are sampled from a grid based on the bounding box size. In order to work with a single classifier, the features should have the same values when extracted from the same object but at different scales. Note, that since we do not perform resize operations the approximations for scale changes from Dollár et al. (2010) do not apply in our case.

We define the ratio function (or correction factor) for feature type f as: the feature value extracted at at scale s of a point divided by the feature value at the original scale of the same point. It is important to note, that due to rescaling the position of the point changes in the image so:

$$r_f(s) = f(s, x, y)/f(1, x/s, y/s) \tag{5.20}$$

Our goal is to extract features at all scales using only the image at its original scale. For this we need a model for the function $r_f(s)$ for different feature types. This would enable us to write:

$$f(s,x,y) = r_f(s) \cdot f(1, x/s, y/s) \tag{5.21}$$

We will determine the form of the ratio function for different feature types. In the first part we ignore discretization errors, resizing artifacts and consider the image as a continuous signal to determine the form of the ratio function theoretically. In the second part, we collect data from the Caltech dataset and perform a linear fit to find the form of the ratio function empirically.

Theoretical estimation

In the following we estimate the theoretical ratio between the features from s times larger/smaller bounding boxes and the original boxes. Ideally, color features should not change since the new position for the features coincide with the positions obtained from the grid which is adapted to the new scale:

$$I_s(x,y) = I(x/s, y/s) \tag{5.22}$$

where I_s denotes the the image at scale s (s-times smaller/larger) and I signifies the original scale. This shows that $r_I(s) = 1$ for color features. The result is subject to discretization artifacts. Applying the derivative shows that $r_{pd}(s) = s^{-1}$ since:

$$\frac{\partial}{\partial x} I_s(x,y) = \frac{1}{s} \frac{\partial}{\partial x} I(x/s, y/s) \tag{5.23}$$

For the gradient magnitude we also have $r_M(s) = s^{-1}$ since:

$$M_s(x,y) = \frac{1}{s} M(x/s, y/s) \tag{5.24}$$

The factor is also transmitted to gradient orientation channels since these are proportional to the magnitude.

Empirical estimation

In order to estimate the correction factor empirically we extract features from the Caltech dataset at multiple scales. For each feature type we find its maximum value. We retain only features that are above $f_{max}/5$ (20% of the maximum feature value). The form of the ratio function similar to the one suggested in Dollár et al. (2010):

$$f(s) = ae^{-\lambda s} f(0) = exp(log(a) - \lambda s) f(0) \tag{5.25}$$

where $f(s)$ is the feature after a downsampling of 2^s and $f(0)$ is the feature at the original scale. According to the previous model, a linear fit for $log(f(s)/f(0))$ determines a and λ. Note, that we approximate the ratio function for pointwise features, whereas in Dollár et al. (2010) the regional sum is considered. To obtain graphs compatible with their work we would need to plot $\frac{f(s)}{2^{2s} f(0)}$ as a function of s because the regional sum introduces a $(2^s)^2$ term (not done here).

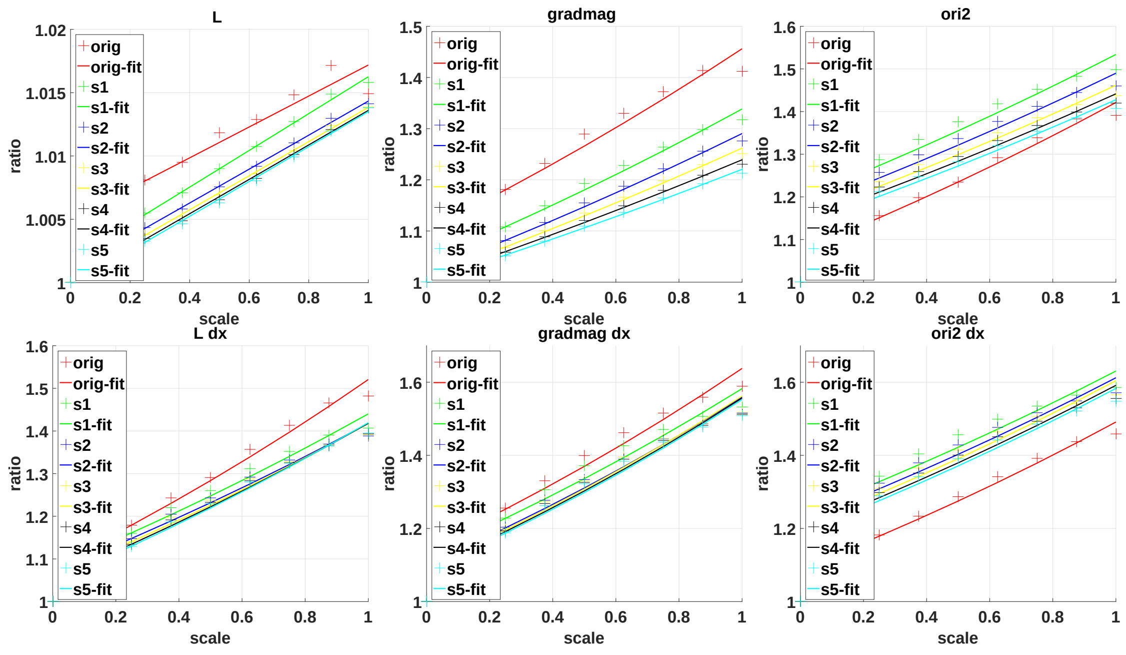


Figure 5.15: Gathered data points for the mean ratio and fitted polynomials
The figures show on the x-axis $-log_2(s)$ and on the y-axis the ratio function $r(-log_2(s))$ for different feature types: L-channel, L-channel with x derivative filter applied, gradient magnitude, gradient magnitude with x derivative filter applied, second gradient orientation bin channel, the same with dx applied. The different lines from the graph plot the behavior of the original channel and the 5 smoothed channels, i.e. *sx* signifies x number of smoothing filters.

Figure 5.15 shows the data points that indicate the mean ratio and the linear fit for 6 representative feature types. We plot the ratio function in terms of $-log2(s)$ for values $s \in [0, 1]$. This was used for linear fit and $r(s)$ is obtainable via a variable change.

The graphs demonstrate that: for color channels, the features retain their values after resize operations, just as the theoretical model predicted: $r_I(s) = 1$; partial derivative operations do not conform to the theoretical model: $r_{pd}(s) = s^{-0.585}$ (at a shrinking of 2 it is around 1.5 and not 2); smoothing operations decrease the exponent further; the first smoothing operation for orientation features behaves differently (see the change between *orig* to *s1* and *s1* to *s2*).

Experimental results

For evaluation we use standard protocols for each dataset and we measure the performance of our method on three benchmarks: Caltech Pedestrians, KITTI Object Detection, and Tsinghua-Daimler Cyclists. In the case of Caltech benchmark, we compute the log-average miss rate for the precision in the range of $[10^{-2}, 10^{0}]$ false positives per image (FPPI) for the reasonable setup. In the case of KITTI, we calculate the average precision (AP) for the recall range of $[0, 1]$ for easy, moderate and hard setups. For the Tsinghua-Daimler dataset we find the average precision for *easy ignore* and *easy discard* configurations.

Caltech - Pedestrian

We provide the results on the Caltech dataset both using the standard training set and with the extended training set. Figure 5.16 -left shows the results along with the state of the art. The figure from the right emphasizes what each method uses as input. Compared to other approaches the miss rate is in the top 5 and the detector is capable of running at 30 FPS.

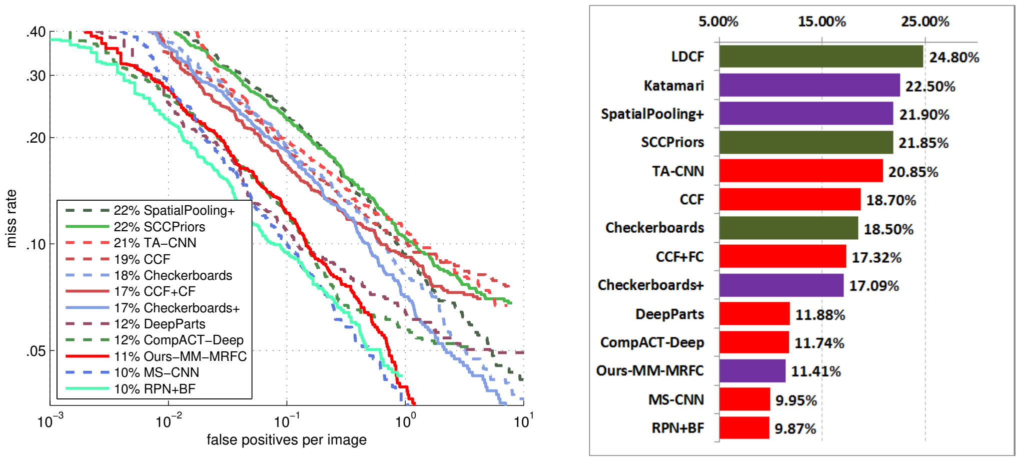


Figure 5.16: Caltech comparison with state of the art
On the right, colors indicate: green - standard training set; purple - multimodal information; red - deep learning

Channel Type		**Caltech MR** - reasonable -
Color	MRFC no s.c.	26.53
	MRFC	23.17
	+ 2D spatial	20.80
	+ 2D symmetry	18.26
Motion	+ SDt	17.29
	+ MM-MRFC	16.11

Table 5.23: Results on Caltech in % after introducing each feature channel
Additional channels are added to the ones above them. MRFC no s.c. does not use scale correction, while the standard variant of MRFC does.

KITTI - Object

On the KITTI benchmark we perform evaluation with the same validation/training split that was used in Chen et al. (2015). In Table 5.24 we show the incremental improvements of the proposed solution on the validation set. Each proposal increases the AP values demonstrating the usefulness of each new feature channel.

The results for the test set compared to other approaches can be found in Table 5.25. We present the results for pedestrians and cars. Because the number of training samples for the bicyclist class is small, we evaluate this object class on another dataset. It can be seen that a competitive performance is achieved for both object classes at significantly lower computational costs. Pedestrian detection runs at 25 FPS and the car detection at 20 FPS.

Context Type		**KITTI AP**		
		Easy	Moderate	Hard
Color	MRFC no s.c.	62.84	59.98	51.10
	MRFC	67.14	61.45	52.76
	+ 2D spatial	69.58	63.83	54.83
	+ 2D symmetry	70.28	64.75	55.66
3D stereo	+ spatial	77.88	70.30	60.63
	+ geometric	77.97	70.61	61.47
	+ MRFC	82.53	74.82	65.95
3D LIDAR	+ spatial	77.88	70.93	61.91
	+ geometric	79.92	72.48	63.13
	+ MRFC	84.26	76.34	67.18
Motion	+ MRFC	85.25	77.72	68.28
BB fitting		86.96	78.87	69.33

Table 5.24: Score results on KITTI (in %) after introducing each feature channel Additional channels are added on top of the ones above them. 3D channels can be computed either from stereo disparity or from 3D point cloud. MRFC stands for multiresolution filtered channels.

Tsinghua-Daimler - Cyclist

The presented pedestrian detector can easily be trained to detect cyclists given a suitable training set. The Tsinghua-Daimler benchmark Li et al. (2016) is an ideal benchmark for evaluating the detection of bicyclist, considering that it contains 22161 annotated cyclist instances in over 30k images. These were recorded in the urban traffic of Beijing. The dataset also provides 3D stereo information for each image frame.

The reader can view the ROC curve of other approaches in Li et al. (2016). Multiple approaches were evaluated in the aforementioned paper such as: traditional boosting based sliding window (ACF, LDCF); deep learning approaches with different object proposals (Selective Search, Edge Boxes, Stereo Proposal) and architectures (VGG, ZF); deformable part models (DPM). We train three detector for narrow, intermediate and wide bicyclists similarly to the other sliding window approaches. Table 5.26 shows the standings on this dataset in terms of AP

Method	Time		Cars			Pedestrians		
			Easy	Moderate	Hard	Easy	Moderate	Hard
FusionDPM	30s	CPU	-	-	-	59.51	46.67	42.05
ACF	1s	CPU	-	-	-	60.11	47.29	42.90
VOTE-3Deep	1.5s	CPU	76.79	68.24	63.23	68.39	55.37	52.59
MV-RGBD-RF	4s	GPU	-	-	-	73.30	56.59	49.63
FilteredICF	2s	CPU	-	-	-	67.65	56.75	51.12
DeepParts	1s	GPU	-	-	-	70.49	58.67	52.78
CompACT-Deep	1s	GPU	-	-	-	70.69	58.74	52.71
Regionlets	1s	CPU	84.75	76.45	59.70	73.14	61.15	55.21
RPN+BF	0.6s	GPU	-	-	-	75.45	61.29	56.08
Faster-RCNN	2s	GPU	86.71	81.84	71.12	78.86	65.90	61.18
Mono 3D	4.2s	GPU	92.33	88.66	78.96	80.35	66.68	63.44
3DOP	3s	GPU	93.04	88.64	79.10	81.78	67.47	64.70
SDP+RPN	0.4s	GPU	90.14	88.85	78.38	80.09	70.16	64.82
MS-CNN	0.4s	GPU	90.03	89.02	76.11	83.92	73.70	68.31
MM-MRFC	0.05s	GPU	90.63	88.45	78.32	82.18	70.02	64.74

Table 5.25: Comparison with state of the art on the KITTI test set - Average Precision in %

for easy ignore and easy discard cases. Only the easy setup is considered since the training annotations contain examples with height greater than 60 pixels. Our proposed approach achieves the best performance on this dataset.

Method	Easy ignore	Easy discard
SS-FRCN-VGG	0.767	0.638
EB-FRCN-VGG	0.838	0.726
SP-FRCN-VGG	0.872	0.786
DPM	0.894	0.816
LDCF	0.898	0.762
ACF	0.898	0.778
DPM-bboxpred	0.905	0.823
MM-MRFC	0.931	0.826

Table 5.26: Average precision values on the Tsinghua-Daimler

Computational costs

A significant benefit of the proposed solution is its low computational cost. Due to the simplicity of the multiresolution filtering scheme and the proposed context channels feature computation for an individual image takes less than 20 ms on a GPU for a 0.5 MP image when considering all modalities. Classification of sliding windows takes around 20-30 ms. There are 5 classifiers for the car detector but the classification time for a single one is faster than in the case of a pedestrian classifier. Classification time does not increase significantly with the number of detectors. Full frame pedestrian detection is achievable at over 20 fps, currently being the fastest top-performing solution.

5.3.5 Conclusions

The presented pedestrian detection methods were published in Varga and Nedevschi (2013a), Varga et al. (2014) and Costea et al. (2017). The initial works have received attention from the research community because they provide simple and fast algorithms that are capable of running even on mobile devices and they offer acceptable detection performance. The later works focus on improved detection accuracy while maintaining a low execution time.

In the first part we have shown our initial proposal consisting of combining region of interest selection and detection. In the second part we have described a solution which is capable of running in real-time at over 25 frames per second.

In the third part we have presented an optimization approach to pedestrian detection. This approach focuses on alternating between feature extraction and classification. During this we can eliminate unnecessary calculations because we can reject multiple candidate windows. We have shown that our presented technique is capable of extending an available state-of-the-art method (ACF). It is applicable to a wide range of methods that employ features calculated locally by summing or aggregating. Increasing the number of weak learners or the number of features does not increase the execution time of the presented detection method. This allows us to consider a larger feature pool for training. The tests show that a larger feature pool is more advantageous on a dataset with more images, probably because this avoids overfitting. In some situations our boosted classifier performs better compared to Dollar's who use feature binning and thus loose some information during training.

In the fourth part we have described a detection method that relies on multimodal information. Input from different sensors, such as laser scanners and stereo-cameras, can greatly improve detection performance. We have shown that detection at high speed is possible even with multiple channel types used for feature calculations. The approach also uses a scale correction mechanism to avoid recomputing features at different scales. The results show that, at the time of publication, the method achieved state-of-the-art results, competing with top-performing deep learning approaches.

In the future we intend to use additional contextual information for detection as suggested by the latest publications. Speeding up the detection by considering object proposals is a possible way to further improve detection speed. We also intend to test this technique for general object detection.

Chapter 6

Conclusions

The aim of the current thesis was to study and develop fast and robust object detection methods. Relevant contributions were made to several processing steps such as: feature extraction, candidate generation, classifier implementation and system architecture.

In chapter 2 we have provided an overview of different feature descriptors for object detection starting from low level color descriptors, through texture and shape features and onto more complicated features such as Histogram of Oriented Gradients. We have provided the necessary theoretical foundation for developing classifiers. Next, we have studied candidate generation methods from the literature, which is a crucial step for systems that have to perform fast detection. The *related works* chapter also contains the enumeration and classification of multiple detection approaches for specific object types.

Several important conclusions can be drawn from the *related works* chapter:

- Relevant features lie at the core of object detection methods. Good features are half of the solution for the entire detection task. In most situations, general features are not well-suited for a particular object type and specific ones need to designed for the task at hand.
- Most features are not invariant to illumination changes. Robust applications require features that are resistant to changes in the illumination of the scene in order to ensure that the system functions in all operating conditions equally well.
- Ensemble classifiers trained with AdaBoost are one of the best-performing classifiers for many detection tasks. They provide lightning fast prediction capability with no expense to classification accuracy. Also, a high dimensional feature vector can be used with no penalty to execution time if the cascading technique is employed. Transforming the classifier into a soft cascade is possible by rejecting examples as quickly as possible, which further increases the classification speed.
- Candidate generation is essential to practical applications. Sliding window techniques have come a long way to provide fast detection even on large images. Still, checking all possible positions and scales is not feasible. Candidate generation must not miss good

candidates and should provide as few proposals as possible. At the same time, it must be fast enough to not hinder the speed of the whole detection pipeline.

Existing solutions for pallet detection and load handling were studied in section 2.6. Regarding these systems and methods, we can draw several conclusions. Even though there are many solutions for automatic load handling, the current ones have important drawbacks. In the following, we enumerate these and point out how our proposals push beyond the current methods.

- Most detection systems rely on sensors that provide information only along a 2D scanline. This prevents such systems to quickly construct a 3D view of the scene for high level reasoning about the objects present in the scene. Our vision-based approach enables such a 3D reconstruction and reasoning due to the usage of stereo cameras.

- Many approaches require installation of landmarks (fiducials) on pallets and on the infrastructure which increases installation time and cost unnecessarily. Our system does not need such an operation and is functional without the mounting of special equipment.

- Evaluating the pallet detection module on an extensive dataset is required to demonstrate the validity of the pallet detection approach. Our system was tested on two extensive datasets acquired from a real warehouse numbering over 8000 images and 9000 pallets. The images contain various scenarios from block storage and rack storage. They were acquired in different lighting conditions and from various viewpoints. Other systems from the technical literature were evaluated on only 100-300 images, and some of them only in laboratory conditions.

- Vision-based approaches often make use of unreliable information to detect pallets such as visible edges and pallet color. Our detector relies on the combination of multiple features that include edges but also other intensity-based features and texture features. A robust classifier coupled with candidate generation module enables detection even when the pallet is partially occluded and when the edges are not clearly visible.

- Most approaches based on laser scanline information need several seconds to perform pallet detection. Execution time of our system to treat a request is around 1 second with room for optimization if this is required.

- Perhaps most importantly, our system incorporates many modules that tackle different tasks. We provide both pallet detection and position estimation for loading operations and also treat unloading operations. Our system incorporates these modules in a successful manner and reuses many components to achieve different tasks.

Section 2.7 presented existing approaches for pedestrian detection. Several methods have good detection performance and efficient execution time. However, for applications on smartphones and other mobile devices with limited processing capabilities, simpler methods are required. We have made changes to existing methods to obtain such approaches in section 3.6. The suggested improvements lie in the following:

- avoid image resizing operations;
- use only a limited number of scales for detection;
- employ features based on integral images which can be quickly calculated.

After the study performed on existing approaches, we have proposed improvements. The theoretical improvements presented in Chapters 3-5 lie in defining new features types for detection, optimizing feature extraction and classifier training, and in proposing new algorithms and methods for specific detection tasks. We highlight the **theoretical contributions** of our thesis:

- Proposal of new features, called Normalized Pair Differences, on intensity images - These features take all intensity pairs from a region and compute their normalized pairwise differences to obtain a descriptor that is invariant to illumination changes. The large dimensionality of the feature vector ensures a high detection accuracy.
- Design of a fast algorithm for training decision trees with AdaBoost - The algorithm execution time is linear in the size of the training set. It is based on presorting all training instances based on each feature and maintaining the sorted structure during decision tree splits at training time.
- Design and implementation of several candidate generation methods both for pedestrian detection and for pallet detection - These methods rely on gradient magnitude to obtain candidate boxes for objects. Integral image computation enables a fast execution time.
- Design and implementation of an automatic image annotation method that relies on compactness and label transfer - Compactness can be used to estimate the similarity between a query image and images from the training dataset that are annotated. Several label transfer methods have been proposed to transfer labels from similar images onto the query image.
- Design of a fast pedestrian detection method by considering only a reduced number of scales - The scales are chosen based on statistics from the training set. The reduced number of scales impacts the detection accuracy only slightly as shown in the experimental results.
- Optimizing detection via Lazy Feature Extraction which defers feature calculation until it is necessary. - By exploiting the fact that several candidates can be eliminated during the classifier cascade, we have avoided calculating the features for these candidates.
- Performing scale correction for filtered channels - We have estimated the correction factor required for several types of features for the purpose of avoiding expensive image resize operations. We have leveraged multimodal information for detection by proposing feature channels computed from depth, motion and context information.

- Proposal of a complex pallet detection method relying on candidate generation and multiple features types - The method combines multiple sources of information, such as: image intensity features, image gradient, stereo information. The system is capable of treating unloading operation requests to detect the target position of the pallet that needs to be unloaded.

All of the presented algorithms were implemented and tested in real systems. Implementation raised other considerations such as efficient execution, low memory consumption, robustness, multithreading and communication with other modules. In order to evaluate the detection methods, the construction of evaluation benchmarks were necessary. We enumerate the **applicative contributions** of our thesis:

- Implementation of the classification module - The module is used as a building block in almost all of the implemented object detection systems. It is efficient and capable of producing state-of-the-art detection accuracy. The classification module was developed as a C++ project. It provides functions for classifier training and prediction along with evaluation methods. It can be accessed at [1].

- Creating and manually labeling the Cluj pedestrians dataset - The dataset is the result of recording urban traffic scenes from Cluj-Napoca with a smartphone placed on the windshield of a car. The video is divided into non-overlapping training set and test set. The training set contains 3205 bounding boxes with pedestrians and 219 negative examples (images with no persons) obtained by labeling every 10th frame. In order to extend the dataset, we have interpolated the bounding boxes, and we have generated an interpolated training set consisting of 27318 pedestrian rectangles.

- Creating and manually labeling the Viano 2-5 datasets for pallet detection - The dataset originated from a warehouse from Viano, Italy owned by Elettric80, which was used for logistics operations. The dataset spans multiple recording sessions and contains over 9000 images with pallets and unloading operations. The dataset is split into training and testing with each pallet labeled manually by providing the bounding box for it.

- Implementation and testing of the proposed pedestrian detection systems - The different proposals were tested on multiple datasets: INRIA pedestrians, Cluj pedestrians, Caltech-USA, KITTI Object Detection. A modified version was installed on a tablet and tested in real traffic conditions.

- Implementation and testing of the proposed pallet detection system - It was integrated at two factory locations: Viano, Italy and Bilbao, Spain. The detection system was running onboard an Automated Guided Vehicle in normal working conditions.

[1]Classification library repository

We have proposed several improvements to pedestrian detection methods. The progress in this domain was published in several conference papers Varga and Nedevschi (2013a), Varga et al. (2014) and Costea et al. (2017).

The proposed pallet detection system for an autonomous load handling system was implemented and tested. The different iterations of the system are described in multiple papers: Varga and Nedevschi (2014), Varga et al. (2015) and Varga and Nedevschi (2016).

The main objective of the thesis was achieved: We have developed several detection systems relying on candidate generation. The systems achieve good detection accuracy and work reasonably fast. The scientific impact of the work can be evaluated by the number of published papers in international conferences and journals. The list of published papers is provided after the bibliography section.

Bibliography

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Amit, Y. (2002). *2D Object Detection and Recognition Models, Algorithms, and Networks*. MIT Press.

Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A., and Ferguson, D. (2015). Real-time pedestrian detection with deep network cascades. In *Proceedings of BMVC 2015*.

Arbeláez, P. A., Pont-Tuset, J., Barron, J. T., Marqués, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *CVPR*, pages 328–335. IEEE.

Baglivo, L., Biasi, N., Biral, F., Bellomo, N., Bertolazzi, E., Lio, M. D., and Cecco, M. D. (2011). Autonomous pallet localization and picking for industrial forklifts: a robust range and look method. *Measurement Science and Technology*, 22(8):085502.

Bay, H., Tuytelaars, T., and Gool, L. J. V. (2006). SURF: Speeded up robust features. In *ECCV*, pages I: 404–417.

Belongie, S., Malik, J., and Puzicha, J. (2000). Shape context: A new descriptor for shape matching and object recognition.

Benenson, R., Mathias, M., Timofte, R., and Gool, L. J. V. (2012). Pedestrian detection at 100 frames per second. In *CVPR*, pages 2903–2910. IEEE.

Benenson, R., Mathias, M., Tuytelaars, T., and Gool, L. J. V. (2013). Seeking the strongest rigid detector. In *CVPR*, pages 3666–3673. IEEE.

Benenson, R., Omran, M., Hosang, J., and Schiele, B. (2014). Ten years of pedestrian detection, what have we learned? In *ECCV-CVRSUAD*. IEEE.

Birchfield, S. and Tomasi, C. (1998). A pixel dissimilarity measure that is insensitive to image sampling. *IEEE PAMI*, 20(4):401–406.

Bolles, R. C. and Fischler, M. A. (1980). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Image Understanding Workshop*, pages 71–88.

Bostelman, R., Hong, T., and Chang, T. (2006). Visualization of pallets. In *SPIE Optics East*.

Bourdev, L. and Brandt, J. (2005). Robust object detection via soft cascade. In *CVPR*, pages II: 236–243.

Breiman, L. et al. (1984). *Classification and Regression Trees*. Wadsworth.

Broggi, A., Cerri, P., and Ghidoni, S. (2005). A correlation-based approach to recognition and localization of the preceding vehicle in highway environments. In *CIAP*, pages 1166–1173.

Byun, S. and Kim, M. (2008). Real-time positioning and orienting of pallets based on monocular vision. In *ICTAI (2)*, pages 505–508. IEEE Computer Society.

Cai, Z., Saberian, M. J., and Vasconcelos, N. (2015). Learning complexity-aware cascades for deep pedestrian detection. *CoRR*, abs/1507.05348.

Carneiro, G., Chan, A. B., Moreno, P. J., and Vasconcelos, N. (2007). Supervised learning of semantic classes for image annotation and retrieval. *IEEE PAMI*, 29(3):394–410.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S., and Urtasun, R. (2015). 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432.

Cheng, M.-M., Zhang, Z., Lin, W.-Y., and Torr, P. H. S. (2014). BING: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, pages 3286–3293. IEEE.

Chua, T.-S., Tang, J., Hong, R., Li, H., Luo, Z., and Zheng, Y. (2009). NUS-WIDE: a real-world web image database from national university of singapore. In *Proceedings of the 8th ACM International Conference on Image and Video Retrieval, CIVR 2009, Santorini Island, Greece, July 8-10, 2009*. ACM.

Chunsheng Fang, A. R. (2009). Probsim-annotation: A novel image annotation algorithm using a probability-based similarity measure. In *20th Midwest Artificial Intelligence and Cognitive Science Conference, Fort Wayne, Indiana*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273.

Costea, A. D. and Nedevschi, S. (2014). Word channel based multiscale pedestrian detection without image resizing and using only one classifier. In *CVPR*, pages 2393–2400. IEEE.

Costea, A. D., Varga, R., and Nedevschi, S. (2017). Fast boosting based detection using scale invariant multimodal multiresolution filtered features. In *CVPR*, pages 2393–2400. IEEE.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

Cucchiara, R., Piccardi, M., and Prati, A. (2000). Focus based feature extraction for pallets recognition. In *BMVC*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893.

Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *ECCV*, pages II: 428–441.

Daniel Costea, A. and Nedevschi, S. (2016). Semantic channels for fast pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2360–2368.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

Dollár, P. (2006). Piotr's Image and Video Matlab Toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *PAMI*.

Dollár, P., Appel, R., and Kienzle, W. (2012). Crosstalk cascades for frame-rate pedestrian detection. In *ECCV*.

Dollár, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *BMVC*, pages 1–11. British Machine Vision Association.

Dollar, P., Tu, Z. W., Perona, P., and Belongie, S. (2009a). Integral channel features. In *BMVC*.

Dollar, P., Wojek, C., Schiele, B., and Perona, P. (2009b). Pedestrian detection: A benchmark. In *CVPR*, pages 304–311.

Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE PAMI*, 34(4):743–761.

Duda, R. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *CACM*, 15:11–15.

Dunlop, H. (2010). Scene classification of images and video via semantic segmentation. In *CVPR Workshop on Perceptual Organization in Computer Vision*.

Endres, I. and Hoiem, D. (2010). Category independent object proposals. In *ECCV*, volume 6315, pages 575–588. Springer.

Endres, I. and Hoiem, D. (2014). Category-independent object proposals with diverse ranking. *IEEE PAMI*, 36(2):222–234.

Enzweiler, M. and Gavrila, D. M. (2009). Monocular pedestrian detection: Survey and experiments. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(12):2179–2195.

Ess, A., Leibe, B., Schindler, K., and Gool, L. J. V. (2008). A mobile vision system for robust multi-person tracking. In *CVPR*, pages 1–8.

Everingham, M., Gool, L. J. V., Williams, C. K. I., Winn, J. M., and Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Felzenszwalb, P. F. (2001). Object recognition with pictorial structures. In *MIT AI-TR*.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D. A., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE PAMI*, 32(9):1627–1645.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54.

Feng, S., Manmatha, R., and Lavrenko, V. (2004). Multiple bernoulli relevance models for image and video annotation. In *CVPR (2)*, pages 1002–1009.

Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *CVPR*, pages II: 264–271.

Gerónimo, D., López, A. M., Sappa, A. D., and Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *IEEE PAMI*, 32(7):1239–1258.

Grubinger, M., Clough, P., Muller, H., and Deselaers, T. (2006). The iapr benchmark: A new evaluation resource for visual information systems. In *International Conference on Language Resources and Evaluation*, Genoa, Italy.

Guillaumin, M., Mensink, T., Verbeek, J. J., and Schmid, C. (2009). Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, pages 309–316. IEEE.

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*, pages II: 807–814.

Hosang, J. H., Benenson, R., and Schiele, B. (2014). How good are detection proposals, really? *CoRR*, abs/1406.6962.

Hough, P. V. C. (1962). A method and means for recognizing complex patterns. U.S. Patent No. 3,069,654.

Iba, W. and Langley, P. (1992). Induction of one-level decision trees. In *Proceedings of the ninth international conference on machine learning*, pages 233–240.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE PAMI*, 20(11):1254–1259.

Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Joblove, G. H. and Greenberg, D. (1978). Color spaces for computer graphics. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12(3):20–25.

Kim, J. C., Lee, K. M., Choi, B. T., and Lee, S. U. (2005). A dense stereo matching using two-pass dynamic programming with generalized ground control points. In *CVPR*, pages II: 1075–1082.

Kim, W., Helmick, D., and Kelly, A. (2001). Model based object pose refinement for terrestrial and space autonomy. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space, Montreal, Quebec, Canada*.

Kolmogorov, V. and Zabih, R. (2001). Computing visual correspondence with occlusions using graph cuts. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 508–515. IEEE.

Kuhl, F. P. and Giardina, C. R. (1982). Elliptic fourier features of a closed contour. *Computer graphics and image processing*, 18(3):236–258.

Labayrade, R., Aubert, D., and Tarel, J.-P. (2002). Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 646–651. IEEE.

Lampert, C. H. (2010). An efficient divide-and-conquer cascade for nonlinear object detection. In *CVPR*, pages 1022–1029. IEEE Computer Society.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages II: 2169–2178.

Li, F. F. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages II: 524–531.

Li, X., Flohr, F., Yang, Y., Xiong, H., Braun, M., Pan, S., Li, K., and Gavrila, D. M. (2016). A new benchmark for vision-based cyclist detection. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 1028–1033. IEEE.

Lim, J. J., Zitnick, C. L., and Dollár, P. (2013). Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, pages 3158–3165. IEEE.

Linde, O. and Lindeberg, T. (2004). Object recognition using composed receptive field histograms of higher dimensionality. In *ICPR*, pages II: 1–6.

Ling, H. B. and Jacobs, D. W. (2007). Shape classification using the inner-distance. *IEEE PAMI*, 29(2):286–299.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157.

Luo, P., Tian, Y., Wang, X., and Tang, X. (2014). Switchable deep network for pedestrian detection. In *CVPR*, pages 899–906. IEEE.

Makadia, A., Pavlovic, V., and Kumar, S. (2008). A new baseline for image annotation. In *ECCV*, pages III: 316–329.

Marín, J., Vázquez, D., López, A. M., Amores, J., and Leibe, B. (2013). Random forests of local experts for pedestrian detection. In *ICCV*, pages 2592–2599. IEEE.

Mathias, M., Benenson, R., Timofte, R., and Gool, L. J. V. (2013). Handling occlusions with franken-classifiers. In *ICCV*, pages 1505–1512. IEEE.

McCulloch, W. P. W. S. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Nam, W., Dollár, P., and Han, J. H. (2014). Local decorrelation for improved detection. *CoRR*, abs/1406.1134.

Nedevschi, S., Bota, S., and Tomiuc, C. (2009). Stereo-based pedestrian detection for collision-avoidance applications. *IEEE Trans. Intelligent Transportation Systems*, 10(3):380–391.

Nygårds, J., Högström, T., and Wernersson, Å. (2000). Docking to pallets with feedback from a sheet-of-light range camera. In *IROS*, pages 1853–1859. IEEE.

Ouyang, W. and Wang, X. (2013). Single-pedestrian detection aided by multi-pedestrian detection. In *CVPR*, pages 3198–3205. IEEE.

Pages, J., Armangue, X., Salvi, J., Freixenet, J., and Marti, J. (2011). Computer vision system for autonomous forklift vehicles in industrial environments. *The 9th. Mediterranean Conference on Control and Automation*.

Paisitkriangkrai, S., Shen, C., and Hengel, A. v. d. (2014a). Pedestrian detection with spatially pooled features and structured ensemble learning. *arXiv preprint arXiv:1409.5209*.

Paisitkriangkrai, S., Shen, C., and van den Hengel, A. (2014b). Strengthening the effectiveness of pedestrian detection with spatially pooled features. In *Computer Vision–ECCV 2014*, pages 546–561. Springer.

PAN-Robots (2013). Pan-robots plug and navigate robots for smart factories. http://www.pan-robots.eu/.

Park, D., Zitnick, C. L., Ramanan, D., and Dollár, P. (2013). Exploring weak stabilization for motion feature extraction. In *CVPR*, pages 2882–2889. IEEE.

Penatti, O. A. B., Valle, E., and da Silva Torres, R. (2012). Comparative study of global color and texture descriptors for web image retrieval. *J. Visual Communication and Image Representation*, 23(2):359–380.

Platt, J. et al. (1999). Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methodssupport vector learning*, 3.

Porikli, F. M. (2005). Integral histogram: A fast way to extract histograms in cartesian spaces. In *CVPR*, pages I: 829–836.

Pradalier, C., Tews, A., and Roberts, J. M. (2008). Vision-based operations of a large industrial vehicle: Autonomous hot metal carrier. *J. Field Robotics*, 25(4-5):243–267.

Quinlan, J. R. (1992). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworths.

Rowley, H. A., Baluja, S., and Kanade, T. (1998). Neural network-based face detection. *IEEE Trans. Pattern Anal. Mach. Intell*, 20(1):23–38.

Russell, C. B., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vision*, 77(1-3):157–173.

Russell, S. and Norvig, P. (1995). *Artificial intelligence: a modern approach*. Pearson.

Schapire, R. (1990). The strength of weak learnability. *MACHLEARN: Machine Learning*, 5.

Scharstein, D. and Szeliski, R. S. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42.

Seelinger, M. J. and Yoder, J.-D. (2006). Automatic visual guidance of a forklift engaging a pallet. *Robotics and Autonomous Systems*, 54(12):1026–1038.

Serre, T. and Poggio, T. (2010). A neuromorphic approach to computer vision. *Commun. ACM*, 53(10):54–61.

Shu, X. and Wu, X.-J. (2011). A novel contour descriptor for 2d shape matching and its application to image retrieval. *Image and vision Computing*, 29(4):286–294.

Sivic, J. and Zisserman, A. (2009). Efficient visual search of videos cast as text retrieval. *IEEE PAMI*, 31(4):591–606.

Torralba, A., Fergus, R., and Freeman, W. T. (2007). Tiny images. In *Massachusetts Institute of Technology, AI Lab*.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

Ursani, A. A., Kpalma, K., and Ronsin, J. (2007). Texture features based on fourier transform and gabor filters: an empirical comparison. In *International Conference on Machine Vision*, pages 67–72.

van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *IEEE PAMI*, 32(9):1582–1596.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory*. Springer, Berlin, 2nd edition.

Varga, R., Costea, A., and Nedevschi, S. (2015). Improved autonomous load handling with stereo cameras. In *Intelligent Computer Communication and Processing (ICCP), 2015 IEEE International Conference on*, pages 251–256. IEEE.

Varga, R., Costea, A., Szakats, I., and Nedevschi, S. (2012). Efficient real-time contour matching. In *Intelligent Computer Communication and Processing*, pages 193–200. IEEE.

Varga, R. and Nedevschi, S. (2013a). Gradient-based region of interest selection for faster pedestrian detection. In *Intelligent Computer Communication and Processing*, pages 147–154. IEEE.

Varga, R. and Nedevschi, S. (2013b). Label transfer by measuring compactness. *IEEE Transactions on Image Processing*, 22(12):4711–4723.

Varga, R. and Nedevschi, S. (2014). Vision-based automatic load handling for automated guided vehicles. In *Intelligent Computer Communication and Processing*, pages 239–245. IEEE.

Varga, R. and Nedevschi, S. (2016). Robust pallet detection for automated logistics operations. In *VISAPP*. IEEE.

Varga, R., Vesa, A. V., Jeong, P., and Nedevschi, S. (2014). Real-time pedestrian detection in urban scenarios. In *Intelligent Computer Communication and Processing*, pages 113–120. IEEE.

Viola, P. and Jones, M. (2001a). Rapid object detection using a boosted cascade of simple features. *Proc. CVPR*, 1:511–518.

Viola, P., Jones, M. J., and Snow, D. (2005a). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161.

Viola, P. A. and Jones, M. J. (2001b). Robust real-time face detection. In *ICCV*, page 747.

Viola, P. A., Platt, J. C., and Zhang, C. (2005b). Multiple instance boosting for object detection. In *NIPS*.

Walk, S., Majer, N., Schindler, K., and Schiele, B. (2010). New features and insights for pedestrian detection. In *CVPR*, pages 1030–1037. IEEE.

Walter, M. R., Karaman, S., Frazzoli, E., and Teller, S. J. (2010). Closed-loop pallet manipulation in unstructured environments. In *IROS*, pages 5119–5126. IEEE.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University.

Wojek, C. and Schiele, B. (2008). A performance evaluation of single and multi-feature people detection. In *DAGM*.

Wojek, C., Walk, S., and Schiele, B. (2009). Multi-cue onboard pedestrian detection. In *CVPR*, pages 794–801.

Xiang, Y., Zhou, X. D., Chua, T. S., and Ngo, C. W. (2009). A revisit of generative model for automatic image annotation using markov random fields. In *CVPR*, pages 1153–1160.

Yan, J., Zhang, X., Lei, Z., Liao, S., and Li, S. Z. (2013). Robust multi-resolution pedestrian detection in traffic scenes. In *CVPR*, pages 3033–3040. IEEE.

Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *ECCV*, pages B:151–158.

Zhang, S., Bauckhage, C., and Cremers, A. B. (2014). Informed haar-like features improve pedestrian detection. In *CVPR*, pages 947–954. IEEE.

Zhang, S., Benenson, R., and Schiele, B. (2015). Filtered channel features for pedestrian detection. *CoRR*, abs/1501.05759.

Zheng, D., Zhao, Y., and Wang, J. (2004). Features extraction using a gabor filter family. In *Proceedings of the sixth Lasted International conference, Signal and Image processing, Hawaii*.

Zhou, F., Feng, J.-F., and Shi, Q.-y. (2001). Texture feature based on local fourier transform. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 2, pages 610–613. IEEE.

Zhu, C. and Wang, R. (2012). Local multiple patterns based multiresolution gray-scale and rotation invariant texture classification. *Inf. Sci*, 187:93–108.

Zhu, Q. A., Yeh, M. C., Cheng, K. T., and Avidan, S. (2006). Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, pages II: 1491–1498.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *ECCV (5)*, volume 8693 of *Lecture Notes in Computer Science*, pages 391–405. Springer.

Appendix A

Published Papers

A.1 In ISI rated international journals

1. **Robert Varga**, Sergiu Nedevschi, ”Label Transfer Using a New Similarity Measure: Compactness”, Transactions on Image Processing, 2013, no. 12, vol. 22, pp. 4711-4723 (impact factor: 3.625)

2. Lorenzo Sabattini, Mika Aikio, Patric Beinschob, Markus Boehning, Elena Cardarelli, Valerio Digani, Annette Krengel, Massimiliano Magnani, Szilard Mandici, Fabio Oleari, Christoph Reinke, Davide Ronzoni, Christian Stimming, **Robert Varga**, Andrei Vatavu, Sergi Castells Lopez, Cesare Fantuzzi, Aki Myr, Sergiu Nedevschi, Cristian Secchi, Kay Fuerstenberg, ”Advanced AGV systems for industrial logistics: the PAN-Robots project”, IEEE Robotics and Automation Magazine - accepted for publication

A.2 In ISI indexed conference proceedings

3. Arthur Daniel Costea, **Robert Varga**, Sergiu Nedevschi, ”Fast Boosting based Detection using Scale Invariant Multimodal Multiresolution Filtered Features”, Computer Vision and Pattern Recognition, Honolulu, Hawaii, July 21-26, 2017

4. **Robert Varga**, Sergiu Nedevschi, ”Improved Autonomous Load Handling with Stereo Cameras”, Intelligent Computer Communication and Processing 2015, Cluj-Napoca, Romania, pp. 251-255

5. **Robert Varga**, Andreea Valeria Vesa, Pangyu Jeong, S. Nedevschi, ”Real-time Pedestrian Detection in Urban Scenarios”, Intelligent Computer Communication and Processing 2014, Cluj-Napoca, Romania, pp. 113 - 120

6. **Robert Varga**, Sergiu Nedevschi, ”Vision-based Automatic Load Handling for Automated Guided Vehicles”, Intelligent Computer Communication and Processing, 2014, Cluj-Napoca, Romania, pp. 239 - 245

A.3 In IEEE Xplore conference proceedings

7. **Robert Varga**, Sergiu Nedevschi, ”Robust pallet detection for automated logistics operations”, International Conference on Computer Vision Theory and Applications, 2016, Rome, Italy, pp. 470-477

8. **Robert Varga**, Sergiu Nedevschi, ”Gradient-based Region of Interest Selection for Faster Pedestrian Detection”, Intelligent Computer Communication and Processing 2013, Cluj-Napoca, Romania, pp. 147 - 154

9. **Robert Varga**, Arthur Costea, Istvan Szakats, Sergiu Nedevschi, ”Efficient Real-time Contour Matching”, Intelligent Computer Communication and Processing 2012, Cluj-Napoca, Romania, pp. 193 - 200

10. Arthur Costea, **Robert Varga**, Tiberiu Marita, Sergiu Nedevschi, ”Refining Object Recognition Using Scene Specific Object Appearance Frequencies”, Intelligent Computer Communication and Processing 2011, Cluj-Napoca, Romania, pp. 179 - 185

In other proceedings of international conferences

11. Szilard Mandici, **Robert Varga**, Andrei Vatavu, Sergiu Nedevschi, ”Vision-based Perception Systems for Automated Guided Vehicles”, The Fifth International Workshop On Cyber Physical Systems, 2016, Bucuresti

12. **Robert Varga**, Mihai Hulea, ”Comparative Analysis of Urban Traffic Control Algorithms”, International Conference on Automation, Quality and Testing, 2010, Cluj-Napoca, Romania, Student Session

A.4 Independent citations

1. Batista, Natlia C., and Guilherme AS Pereira. ”A Probabilistic Approach for Fusing People Detectors.” Journal of Control, Automation and Electrical Systems 26.6, 2015, pp. 616-629.

2. Tchapmi, Lyne P. ”Pedestrian Detection on Mobile Devices.”, Mobile Vision course final project, Stanford, 2015

3. Krug, Robert, et al. ”The Next Step in Robot Commissioning: Autonomous Picking and Palletizing.”, IEEE Robotics and Automation Letters 1.1, 2016, pp. 546-553.

4. Ebsil Sherly, G. T., and Mrs S. Vanitha Sivagami. ”Automatic Annotation of Images using Label Transfer.”, International Journal of Advanced Research in Computer Engineering and Technology (IJARCET), Volume 3, Issue 6, June 2014, pp. 2224-2228

5. Yao, Wei, Otmar Loffeld, and Mihai Datcu. ”Application and Evaluation of a Hierarchical Patch Clustering Method for Remote Sensing Images.”, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Vol. 9, Issue: 6, June 2016, pp. 2279 - 2289

6. Huang, Cheng-Ming, Yi-Ru Chen, and Li-Chen Fu. ”Visual tracking of human head and arms using adaptive multiple importance sampling on a single camera in cluttered environments.” IEEE Sensors Journal 14.7, 2014, pp. 2267-2275.

Appendix B

Listings of three papers

Label Transfer by Measuring Compactness

Robert Varga, and Sergiu Nedevschi, *Member, IEEE,*

Abstract—This paper presents a new automatic image annotation algorithm. First, we introduce a new similarity measure between images: compactness. This uses low level visual descriptors for determining the similarity between two images. Compactness indicates how close test image features lie to training image feature cluster centers. The measure provides the core for a k-nearest neighbor type image annotation method. Afterwards, a formalism for defining different transfer techniques is devised and several label transfer techniques are provided. The method as whole is evaluated on four image annotation benchmarks. The results on these sets validate the accuracy of the approach, which outperforms many state-of-the-art annotation methods. The method presented here requires a simple training process, efficiently combines different feature types and performs better than complex learning algorithms, even in this incipient form. The main contributions of this work are: the usage of compactness as a similarity measure which enables efficient low level feature comparison and an annotation algorithm based on label transfer.

Index Terms—Information search and retrieval, scene analysis, object recognition, automatic image annotation

1 Introduction

Automatic annotation of images in its simplest form aims at labelling images with keywords from a dictionary. This procedure is necessary mainly to enable content based search and a better organization of images. If this can be realised automatically, human users are freed from the painstaking work of parsing thousands of images. This also ensures that the obtained labels are not influenced by the specific taste and inclination of the human annotators.

Today more and more images are stored in image archives and databases both locally on computers and on the Internet. As multimedia data stored grows in size it is becoming more difficult to maintain and organize them. Saving resources such as images in annotated form allows for efficient access and organization. This is the primary reason why automatic image annotation is deemed useful. Labeled images can be retrieved based on their text labels which reflect their content as opposed to providing visual content such as an example image for retrieval.

One of the main problems in this area of research is the difficulty of inferring high level textual labels from low level visual features. This is referred to in the literature as *semantic gap* to emphasize that a bridge must be built between the two. Even though low level visual features such as color, edges and texture can be extracted easily from images it is most difficult to organize and summarize them.

Another important issue in the domain concerns *weak labeling*. Weak labeling means that even though learning algorithms have access to annotated images two factors may prevent learning from these examples. The first one is regarding situations where even though a label is not present, the object or concept represented by it is contained in the image. Such images are considered as negative examples by learning algorithms which is undesirable. The second effect stems from the fact that labels are not linked to special regions of the image instead they are given for the image as whole. Thus it is impossible to determine exactly which bounded area it refers to.

The goal of this paper is to introduce a new approach to image annotation, but not a fully optimized method. In this regard its goal is similar to [1] i.e. to create a new approach on which more complex methods can be built. Currently, global feature vectors are employed for image comparison. This means that a lot of fine detail is lost when transforming the informative but noisy low level features to a global one which represent the image as a whole. Comparing low level features from two images directly is the other extreme where we encounter other problems: noise and high computation time. The proposed approach lies between these two extremes and enjoys the advantages from low level feature descriptiveness while maintaining a low run-time.

The contribution of this work lies mainly in the definition of compactness and using it as a similarity measure between images. Even though compactness is present in many places in the literature in the form of within-cluster sum of squares, in the form proposed here it is much more general. Firstly, because it is defined on two arbitrary sets: the data points and some other set considered to be centers. Secondly, the definition contains a general distance function. Thirdly, the norm-based definition has interesting theoretical properties. Fourthly, and perhaps most importantly, applying this measure in the current context is a totally new idea. We also provide a formalism for defining label transfer

• *R. Varga and S. Nedevschi are with the Computer Science Department, Technical University of Cluj-Napoca, Romania, E-mail: robert.varga@cs.utcluj.ro, sergiu.nedevschi@utcluj.ro*

techniques based on weight function. This formalism permits the mathematical description of several label transfer techniques.

2 Related Work

Research in the domains of object recognition and scene recognition has produced numerous methods for automatic annotation of images/videos. The purpose of this section is not to present the state-of-the-art but rather to put our proposed approach in context. Here, we will categorise these methods into two high-level classes. Even though algorithms from one of the class can differ radically, the underlying approach is the same.

2.1 Keyword-based annotation methods

The first category of methods create models for every label (or keyword, or concept) from the dictionary. We refer to a model as representation of a label. At annotation time the relevant labels are determined using these models and the extracted low level visual features from the test image. In most of the cases the inclusion of a label in the annotation is based on a binary decision. One of the disadvantages of these methods is that annotation with a keyword is based on a single decision and not based on multiple possibilities.

There are various types of models employed in the literature for representing a label: Gaussian Mixture Models[2] characterize a label as a multimodal Gaussian distribution defined on the feature space, Dirichlet Distribution[3] is a latent variable model that models the joint and conditional distribution of of the labels given the image, SVM classifier models[4] are supervised learning models that learn a separating plane between the data points of different classes in the feature space, Bayesian Hierarchical Models[5] are used to infer labels by employing a patch based representation of the input image using a Bayesian inference, Multiresolution Hidden Markov Models[6], [7] learn both spatial and multiresolution relationships between features, Markov Random Fields[8] increase annotation performance by learning spatial relationships between pixels.

Next we describe some methods from this category in more detail. In Supervised Multiclass Learning[2] each label is modelled as a Gaussian Mixture Model. The model corresponding to a specific semantic label is created by applying a hierarchical version of Expectation Maximization(EM) on the visual descriptors from each training image which share the keyword. By substituting descriptor values from a test image into each probability density function - the previously obtained GMMs - one can determine the conditional probability of each concept given the visual descriptors. Annotations are formed by taking the first 5 concepts with highest log-probability. Although this method constructs models in a very efficient manner, the training process is long mainly because of EM.

The works from [4], [14] rely on classifying global image features in the form bag-of-words features for scene recognition. These features constitute a global histogram for an image or a region. Each extracted local feature vector is associated to the closest element from a codebook of local features and the frequency of each center from the codebook constitutes the histogram (i.e. the histogram is the discrete distribution of the local features). This histogram construction method corresponds to average pooling, other alternatives are available such as max-pooling[11], geometric l_p-norm pooling[12], Geometric Consistency Pooling in Superpixels[13]. A classifier (e.g. SVM) is trained using these histograms and the available labels. However, to produce more than one output one must use multiple binary classifiers (one for each keyword to form a multiclass SVM). This makes it necessary to form a training set containing negative examples, images which are not labelled with the respective keyword. The assembling of such a set implies extra work and one can always provide new negative examples, which clearly is a drawback of binary classifiers.

2.2 Retrieval-based annotation methods

The second category of methods are based on the idea that similar images have the same labels. The key point in these methods is to define similarity measure between two images. At annotation time one can retrieve similar images from the database using the similarity measure and use the labels from these images to form the annotation. To emphasize the difference between this category and the former one, we mention that here models are practically constructed for each training image. These methods make use of *label transfer* since labels are passed on from similar images using different strategies to form annotations. Because the proposed approach falls into this category we will present similar approaches from the literature.

A recent publication[1] presents a baseline method for k-nearest neighbor image annotation. The images are represented by different types of global features: color histograms from various color spaces, Gabor and Haar wavelets for texture descriptors. The similarity between images will be the inverse of the distance between the global descriptors of the two images. Different feature types contribute equally in the calculation of the final distance value. Label transfer is then obtained in a greedy manner, giving importance to the first match.

The authors in [9] rely on a large image database of 80 million images from the Web to perform a k-nearest neighbor label transfer. The raw pixel values of 32x32 form the global feature vectors and an adaptive distance function is used as a similarity measure. The large number of learning examples compensate for the usage of only low level features. Another large dataset focusing on scene classification is presented in [10]. The publication evaluates state-of-the-art methods for large-scale scene recognition and makes use of multiple features.

One can take the matching technique one step further by allowing more general metrics such as the Mahalanobis distance. The parameters of the distance metric are obtained using Metric Learning techniques. The results using such method are described in [15], [16], which is one of the currently best performing methods on several benchmarks.

Other approaches for annotating images are considered in [17], where each object is described using attributes. The work [18] focuses on learning object attributes together with object categories. In [19] the authors represent the image as a high-level vector of object detector responses denoted as Object Bank.

3 Compactness based matching

In the following we define the notations used throughout the paper. Let $\mathcal{I} = \{I_1, I_2, ..., I_{Ni}\}$ denote the images constituting the training database, and $\mathcal{L} = \{l_1, l_2, ..., l_M\}$ the vocabulary containing the semantic labels (or keywords, words, concepts, tags). Ground-truth information is represented by associating to every image from $\mathcal{I}$ a set of labels from $\mathcal{L}$: $G = \{(I, L) | I \in \mathcal{I}, L \subset \mathcal{L}\}$. We denote with X the descriptors extracted from a test image, which is a set of descriptor vectors x_i, each having the dimension D. The descriptors from training image I_n are called $X^{(n)} = \{x_i^{(n)} | i = \overline{1, T_n}\}$, where T_n is the number of descriptors extracted from image n. The set of centers extracted from training image n is the set $C^{(n)} = \{c_i^{(n)} | i = \overline{1, K}\}$, the set of all centers is notated with $\mathbf{C} = \cup C^{(n)}$. Note that ultimately these are sets of points in a high dimensional space.

Current annotation methods that are based on image retrieval extract a global feature vector from each image and compare these vectors using a distance function. But is it possible to compare local features? Even though the global representation enables a fast comparison details of the particular image is lost. This is the main reason why it would be better if comparison of local features could be obtained efficiently. It is obvious that comparing each local feature from one image to each from another would be practically infeasible. This is why we propose to represent each training image with a set of relevant descriptor centers. These are obtained using the k-means clustering algorithm[20] applied on all the descriptor vectors extracted from that particular image. The k-means algorithm can be substituted with another method with better performance, but we use it for simplicity. We need to define then a distance - a matching score - between some new data points represented by test image features and a set of centers. This is where compactness comes in.

3.1 Compactness definition

Compactness is a measure that indicates how close data points are to a set of centers. The compactness between a set of points X and the centers C is given by:

$$\mathbf{c}(X, C) = \frac{1}{|X|} \sum_{i=1}^{|X|} \min_j d(x_i, c_j) \tag{1}$$

where $j \in \overline{1, K}$, $|X|$ denotes the cardinality of the set X and $d(x, y)$ is a metric defined on the D dimensional space. The previous definition states that compactness is the sum of the distances of each point from X to the closest point from C. Here X refers to any points in general, and in particular it can be the same as the set of features extracted from image n, $X^{(n)}$ in which case $|X| = T_n$. In our experiments we have found that the L^1 distance performs best in this context compared to the L^2 or the Chi-Square metric.

A less restrictive definition uses L^p norms instead of the distance function. This is useful in practice because it avoids extracting roots and has interesting properties.

$$\mathbf{c}(X, C) = \frac{1}{|X|} \sum_{i=1}^{|X|} \min_j ||x_i - c_j||_p^p \tag{2}$$

Note, when applying k-means on a set of points X the objective function to minimize is exactly the compactness of the centers and the point set X. So the following are equivalent to the L^2 norm compactness applied to the same points from which the clusters centers were obtained: within-cluster sum of squares; the minimum sum of squares; distortion function; potential function (the literature uses a multitude of terms referring to this value).

The compactness is always positive since it is a sum of distances or norms. Also, if we suppose that the points X are characterized by centers C then:

$$\mathbf{c}(X, C) < \mathbf{c}(X, C'), \forall C' \neq C \tag{3}$$

The last inequality(3) holds if k-means truly finds the set of centers that minimize the compactness. This can be ensured by running the algorithm multiple times with different initial center guesses and using different optimized initialization techniques in order to avoid local minimas[21]. More details about the k-means algorithm employed here along with specific parameters are described in section 6.1.

To use compactness as a similarity measure between two images one must first find their representation in some feature space denoted by X and Y respectively. Afterwards, one of the images - consider in this case the second image - is characterized by the cluster centers of the features. At this step we obtain the set C from Y. The similarity is then calculated as the compactness of the features from the first image to the cluster centers of the second image, more precisely $\mathbf{c}(X, C)$. If the asymmetry of this measure is an issue the compactness can be evaluated with the roles of the images swapped, however in practice the training images will be compactly represented by cluster centers and it is much more practical to reuse these.

3.2 Interpretation and motivation

We now investigate what this measure represents. Suppose we are given a set of points X and we want to find their compactness relative to some set of centers C. Consider the following partitioning of X around each $c_k \in C$ (Voronoi partitioning):

$$X_k = \{x \in X | k = argmin_j\{||x - c_j||\}\}, k = \overline{1, |C|} \quad (4)$$

This states that the sets X_k contain all the points that have center c_k as the closest center to them. Clearly the sets X_k are mutually disjoint sets and $\cup X_k = X$. Then the following identity is true for compactness that uses the L^p norm:

$$\begin{aligned} \mathbf{c}(X, C) &= \frac{1}{|X|}\sum_{k=1}^{|C|} |X_k| \mathbf{c}(X_k, c_k) \\ &= \frac{1}{|X|}\sum_{k=1}^{|C|}\sum_{x \in X_k} ||x - c_k||_p^p \quad (5) \end{aligned}$$

$$= \frac{1}{|X|}\sum_{k=1}^{|C|}(\sum_{x \in X_k} ||x - \overline{x_k}||_p^p + |X_k| \cdot ||\overline{x_k} - c_k||_p^p)$$

Where $\overline{x_k}$ are the centers of mass for the points X_k. The decomposition is true because the partitions contain only the closest elements to c_k. Tha last step follows from a well known lemma involing L^p norms:

$$\sum_{x \in X} ||x - c||_p^p = \sum_{x \in X} ||x - x_c||_p^p + |X| \cdot ||c - x_c||_p^p \quad (6)$$

We show this in the L^2 case in 2D, it can be easily extended for any p and any dimensions - we use $p = 1$ and $p = 2$ in this work. Let x_c denote the center of mass of the points x, so $x_c = \frac{1}{|X|}\sum_{x \in X} x$. Consider the translated coordinate system Ouv that has as the origin this center of mass. Also for convenience, rotate the axis such that the point c has the representation $(c_u, 0)$. In this coordinate system we can write for any point from X, $x_i = (u_i, v_i)$:

$$||x_i - c||^2 = (c_u + u_i)^2 + v_i^2 \quad (7)$$

Summing over all the points:

$$\begin{aligned} \sum_i ||x_i - c||^2 &= \sum_i {c_u}^2 + \sum_i (u_i^2 + v_i^2) + 2c_u \sum_i u_i \\ &= \sum_i ||x_i||^2 + |X|{c_u}^2 \quad (8) \end{aligned}$$

The last step uses the fact that the center of mass is now the origin. If the elements of X_k are considered to be i.i.d. random variables then $\overline{x_k} = E[X_k]$ and $\mathbf{c}(X_i, x_{ck}) = Var[X_k]$ if the compactness uses the Euclidian norm. The identity (5) shows some important characteristics of the compactness measure. It is composed out of two terms: the first term is the variance of the partitions and the second is the distance between the original centers and the centers of the partitions X_k. This means that the compactness will be minimal if the variance is low and the centers are close. Consequently, compactness indicates how tightly the points are situated around the centers. It may happen that $\mathbf{c}(Y, C) < \mathbf{c}(X, C)$ for some test data points Y and training data points X. This is the case when the test points Y have the same centers but a lower variance than X.

3.3 Comparison to two other similarity measures

We now compare compactness to two similarity measures and show its advantages over them. The two alternatives considered are: distances defined on bag-of-words type histogram descriptors, and the probability of data points fitting a Gaussian Mixture Model. Other similarity measures rely on distances defined on raw image pixel values or on histograms. Mutual Information[22], for example, is defined as the difference between the individual entropies and the joint entropy of the two images. Compactness assumes that one of the images is represented succinctly by a set of centers which reduces computation time.

If the bag of words approach is used then every image will be characterized by a histogram which reflects the distribution of the closest prototypes associated to each feature. The prototypes are k-means cluster centers and together they form the dictionary. The disadvantage in this case is that some relevant centers for the current image may not be present in the global dictionary. This prohibits the correct comparing of the images since important centers will be mapped to other centers from the dictionary. Even if all the relevant centers of the image are inside the dictionary, if two images have the same histograms we cannot determine how close or far they are even though there may be significant differences between them.

By studying Figure (1) we can analyze two cases where the histogram representation fails. The 2D training points which are partitioned in two are marked with black circles and were drawn from the same two normal distributions on both figures. The test points corresponding to two partitions are blue squares. In both cases the histogram for the test points will coincide with that of the training points. In the left graph (case a) both the test and the training points have the same centers, but the the test points have a larger variance. In the right graph (case b) the distributions have the same variance but the centers of the partitions are different. In all cases histograms will indicate that the test points are from the same distribution as the training points even though there are significant differences. This drawback is eliminated by the compactness which takes into account both the spread factor - variance - and the displacement between the centers.

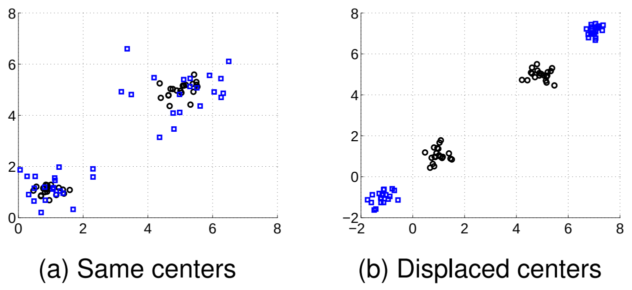


Fig. 1: Two cases were the histogram representation fails

Associating every point to the closest center lies at the heart of the compactness. Consider now that the image is characterized by a Gaussian Mixture Model as in [2]. In this case the similarity between it and some feature set X is the probability that the data fits the model:

$$P(X|\pi_k, m_k, \Sigma_k) = \prod_{x\in X} \sum_{k=1}^{K} \pi_k G(x, m_k, \Sigma_k) \tag{9}$$

This compares every x with every center m_k and thus always penalizes good matches, since if a point is close to some m_k it will be far from all the other $K-1$ centers. Regardless of how well the data fits the distribution this penalty to the likelihood is always applied. Another issue with GMM obtained through EM is that on many occasions fixing the number of centers to a constant gives poor distributions since two or more centers are described by only one Gaussian with a covariance matrix of large values. A model with large covariance values will tag as similar a wide range of points which is clearly undesirable. This effect becomes visible if an image is retrieved as similar for many image queries.

4 Label Transfer

Label transfer is achieved by constructing a histogram h. Each of its bins corresponds to a concept from $\mathcal{L}$, so $h \in \mathbb{R}^M$. We take into consideration the labels of the best N matches. After histogram construction, the labels with the highest corresponding bin value will be chosen to form the final annotation. Depending on how the histogram bins are incremented, different transfer schemes can be obtained.

The result of the matching procedure can be modelled as a function μ, which returns an ordered list of indexes, based on the compactness between the test image descriptors and each of the training image centers: $\mu(I) =< i_1, i_2, ..., i_N >$, where $\mathbf{c}(X, C^{(i_1)})$ is the minimal compactness, the one with i_2 is the second smallest and so on.

We define a weight function $\omega : \overline{1,N} \to \mathbb{R}$. Every label from the training image I_{i_n} increments the corresponding bin in the histogram by $\omega(n)$:

$$h = \sum_{n=1}^{N} \sum_{l\in L_n} \omega(n)\delta_l \tag{10}$$

L_n represents the labels from n-th match, more precisely from the training image I_{i_n} from the list $\mu(I)$. These labels are available from ground-truth information G. $\delta_l \in \mathbb{R}^M$ is vector a containing zeros on all positions except at the position uniquely associated to label l where it is one. By changing the expression of the weight function ω different types of transfer techniques can be achieved. In the following we present some particular cases.

4.1 Equal contribution transfer

In this case every match from the list $\mu(I)$ contributes evenly to the histogram. We have:

$$\omega_0(n) = 1, \forall n = \overline{1,N} \tag{11}$$

This is the most elementary type of transfer, it can be viewed as a majority voting scheme. It has the advantage that it eliminates those labels that only appear in a few matches. However, if the matching technique is good, we want to give more importance to the best matches.

4.2 Transfer based on rank

Rank based transfer entails weighing the best matches more and decreasing the weight exponentially based on the rank. In this case the weight function has the following form:

$$\omega_a(n) = 2^{a(1-\frac{k-1}{N-1})}, \forall n = \overline{1,N} \tag{12}$$

The parameter a can be tuned to obtain the best results. Of course the base of the exponent can be any number $b > 1$ or equivalently we can choose a to be $a' log_2 b$. One can see that $\omega_a(1) = 2^a$ and $\omega_a(N) = 1$. Note that weighing the matches equally (case w_0) is a special case of this function where $a = 0$. This is why, at parameter testing these two functions fall into the same category.

Note that this gives importance to matches according to their position regardless of their distance to the test instance. It may well be that all matches are very close, in this case the rank is not relevant.

4.3 JEC type transfer

The technique presented in [1] favours the best match and the rest of the labels are transferred based on their appearance frequencies in the training set. This case corresponds to the following form:

$$\omega_J(1) = 10, \omega_J(n) = 1, \forall n = \overline{2,N} \tag{13}$$

The histogram values will be updated on the last step in order to take into account the label frequencies. It is a greedy technique and thus depends on a good first match.

4.4 Transfer based on distance

To take into account the compactness values $\mathbf{c}_n$ of each of the matches, the following weight function is defined:

$$\omega_d(n) = 2^{b(1-\mathbf{c}_n/\mathbf{c}_1)}, \forall n = \overline{1, N} \quad (14)$$

This is particularly useful where compactness values are relevant, however the first match will always receive the same weight, i.e. because this weight function is relative to $\mathbf{c}_1$ it does not treat the case where even the best match is far away from the test instance.

4.5 Multiple features

Histogram construction using weight functions can be easily extended to the case where we intend to use multiple features. We begin by constructing the histogram normally for the first descriptor type. Then we save the histogram instead of resetting the bins to zero and repeat the process for the matches obtained from the other descriptor types. In this way every descriptor contributes to the final histogram which will provide the annotations.

In this paragragh we will refer to an instance of the algorithm that uses a specific kind of local feature simply as "a method" in order to simplify explanation. In this case the definition for the transfer histogram becomes:

$$h = \sum_m \eta_m \sum_n \sum_{l \in L_{m,n}} \omega(n)\delta_l \quad (15)$$

where the index m refers to the method number, η_m is the weight of the method m and $L_{m,n}$ is the set of labels from the n-th match using method m. In our experiments we have set $\eta_m = 1, \forall m$, i.e. we weigh each feature type equally. Note that the order of applying different methods is irrelevant.

4.6 Considering appearance frequency

After the histogram has been constructed using one of the weight functions described before, it can be updated by the frequency of each label from the training set. This is the number of times it appeared in the training set. For each non-zero position of the histogram (corresponding to labels that appeared at least once in the matches) we add a value proportional to the frequency (f_l) of the corresponding label:

In this case equation (15) is extended as:

$$h = \sum_m \eta_m \sum_n \sum_{l \in L_{m,n}} \omega(n)\delta_l + \varphi \sum_{l \in \cup L_{m,n}} f_l \delta_l \quad (16)$$

The parameter φ is set in such a way so that $\varphi \max f_l < \min \omega(n)$, i.e. frequency values are secondary to weights. This may seem to reduce the influence of this factor but the goal is to use this information only in uncertain cases, for example when we have two concepts with the same histogram value. JEC type transfer requires the histogram to be constructed using the previously defined equation.

Algorithm 1 Training

```
Ensure: centers from all training images.
1: for all training images $I_n$ do
2:     Extract local features $X^{(n)}$
3:     Apply k-means using $K$ centers to obtain $C^{(n)}$
4:     Save centers
5: end for
```

Algorithm 2 Testing

```
Require: Training image centers.
Ensure: Annotations for every test image.
1: for all test images $I$ do
2:     Extract local features $X$
3:     Sample $X$ to get $B$
4:     for all training image $I_n$ do
5:         find $\mathbf{c}(B, C^{(n)})$
6:     end for
7:     Obtain the first $N$ best matches using $\mu(I)$
8:     Transfer ground-truth labels from matches to obtain annotation using (10)
9: end for
```

5 Algorithm Description

In this section we provide the high level steps required for the training process and for effective image annotation. Also the effect of different parameters on the execution time is discussed. The training involves the steps described by Algorithm 1.

We have fixed the number of clusters for the k-means algorithm to $K = 20$ for all our experiments based on some preliminary tests. If multiple features will be used for annotation it is necessary to run the training for each feature type. Note that in this way we form the building blocks for more complex methods that use different feature combinations and the training is done only once for each feature type.

In order to annotate an image the following operations from Algorithm 2 are to be executed. If multiple features are used then these operations are performed for each feature type and the equation (15) or (16) is used once at the end to form the transfer histogram.

6 Experimental Results

6.1 Implementation Details

All tests for the Corel5k dataset were run on a machine that has an Intel 2.66 GHz processor with two cores and 2GB RAM. For the larger datasets we performed the tests on the computing grid of the Technical University of Cluj-Napoca[23]. The number of parallel processes was set to 200 or 400. The application was implemented using C/C++. Libraries included were: OpenCV - vision library, vlfeat - for SIFT extraction. K-means implementation provided by OpenCV[24] was used running each time for a maximum of 200 iterations, with tolerance of 10^{-7}, five trials and kmeans++ center initialization by Arthur and Vassilvitskii[21]. The vlfeat library is utilized for dense SIFT extraction[25]. Multithreaded implementation was developed for matching and SIFT extraction in parallel from multiple image channels. The tests on the Corel5k involving high dimensional descriptors (SIFT and different combinations) were run on two threads in parallel.

6.2 Local descriptors

This section contains details about the local descriptors used for testing. A summary of the local descriptors is given in Table 1. Feature extraction strategy employed is dense sampling on a grid with displacements of 2 pixels. Each of the following paragraphs describes a different feature type.

The first feature type we present is a simple color descriptor that is obtained by first resizing the image to maximum a dimension of 64 pixels, while retaining the aspect ratio. This operation performs an averaging and additionally it reduces execution time. Afterwards, the feature vector at each pixel will contain the triplets from RGB, Lab and HSV color spaces. Feature dimension is 9. Despite its simplicity, this descriptor can produce surprisingly good results in this context. The importance and the efficiency of color descriptors is demonstrated by the fact that almost all annotation methods make use of this information. It is natural then to include a local descriptor based on color in our experiments.

A recently published texture descriptor called WLD[26] is also tested. Here we use a local histogram variant of the descriptor. We extract the excitation and gradient orientation values at every pixel and construct histograms of these on 8x8 blocks. These histograms will be the local features. We use the single resolution variant with 8 angle bins, 6 excitation bins and the parameter S (the number of excitation subbins) is set to 1. Final feature dimension is 48. This texture descriptor was evaluated on the Brodatz texture benchmark and has obtained superior results compared to SIFT, Gabor and several other texture descriptors (see [26] for results).

Histogram of Oriented Gradients(HOG) descriptors are extracted both from a grayscale image and from all three channels of the RGB image. The number of angle bins is set to 12. Feature dimension is 12, respectively 36 for color type. This descriptor was successfully applied for pedestrian detection [27] and other objects as well[28].

Discrete Cosine Transform coefficients have been utilized with great success in SML[2]. The coefficients are obtained on a 8x8 region using matrix multiplication and dimension reduction can be obtained easily be considering only the upper left corner of the resultant matrix[29]. We use the descriptors from the 3 channels of the Lab color space.

Scale Invariant Feature Transform(SIFT[30]) features are extracted on a dense grid from the image transformed into the Opponent Color Space. This sampling strategy has been proven to be the most effective in [31]. This descriptor has the largest total dimension from the ones used here. SIFT has properties such as scale and rotation invariance which are very useful for object recognition.

TABLE 1: Local descriptors employed

Descriptor	Type	Dimension
RGB	color	3
Lab	color	3
HSV	color	3
RGB+Lab+HSV	color	9
WLD	texture	48
HOG	texture	12
color HOG	texture	36
DCT	texture	63/192
dense SIFT	texture	128
dense SIFT-OCS	texture	384
Law	texture	10
Gabor	texture	12

Law texture descriptors[32] are obtained by filtering the image with the 16 Law convolution kernels. The result from the outer product of 4 one dimensional kernels and does not include the W kernel. No energy is calculated, but instead these raw values are used. The resulting descriptor is of size 10.

Another texture descriptor is obtained by filtering the image with different Gabor filters. The filters have 4 different orientations and 3 different scales, which is enough to represent 97% of the image energy [33]. The results of the filters form a the feature vector at each pixel. The applications of Gabor filters include mainly texture descriptors [34], [35].

Visual descriptor extraction using dense sampling can yield many feature vectors. If the image has height **h** and width **w** and a sampling strategy with displacement **d** is used then we have: $|X| = \left\lfloor \mathbf{h} \cdot \mathbf{w}/\mathbf{d}^2 \right\rfloor$, where $\lfloor x \rfloor$ indicates the floor function. Using every vector from the set for compactness calculation would be practically inefficient (high execution time). This is why a uniform sampling is applied on the set X and the compactness is calculated on the reduced set B. This is not the same as increasing the grid spacing since it may be that $|X| \neq k^2|B|$, for some natural number k. So let $B = \{b_i = x_{i\Delta} | i\Delta < |X|\}$, i.e. B contains every Δ-th sample from X. In this case $|B| = \lfloor |X|/\Delta \rfloor$ and we refer to the cardinality of the set as the bag size.

6.3 Time complexity analysis

The execution time for annotation is dominated by the matching process. The time complexity of matching using compactness is given by:

$$\mathcal{O}(T \cdot B \cdot K \cdot D)$$

where T is the number of training images; B is the 'bag' size - the number of features selected for compactness calculation; K is the number of clusters for k-means; D is the dimension of the feature vector. This can be optimized by halting distance calculation if the current distance exceeds the minimum distance obtained up to that point. The execution time grows linearly with the feature dimension, this is why it is recommended to apply dimension reduction techniques such as Principal Component Analysis (PCA) on large feature vectors such as SIFT. Matching can be parallelized easily at the highest level (at T) by dividing the training set into groups for each thread. We compare this to a Bag-of-words approach where the histogram construction and the matching costs:

$$\mathcal{O}(F \cdot D \cdot C + T \cdot C)$$

where F is the number of features extracted from an image, C is the number of keywords from the codebook. This shows that the first approach requires more time and is linearly proportional to the number of training instances. However, methods of later type usually require a much larger feature vector (large D). Even though the time complexity is high, relatively low execution time can still be achieved. This is demonstrated by providing the execution times of different cases in the experimental results section.

6.4 Evaluation protocol

The protocol for evaluation follows that which was already outlined in previous works (such as [2]) in order to enable comparison between methods. The different databases are split into two disjoint sets: training set - used for extracting k-means centers; test set - for the evaluation of the method. No information about the ground-truth labels of the test set are used when generating the automatic annotations. The automatically generated annotations are afterwards compared with the human given ones to obtain metric values.

We label each image with exactly five labels. For each keyword from the dictionary that appeared at least once in the test ground-truth we calculate the precision and recall values. For each label we define the following numbers:

- l_h - the number of times l appears in the test ground-truth;
- l_a - the number of times l was provided in an annotation by the automatic annotation method;
- l_c - the number of correct annotations with the label l.

In this setting the precision is $\mathbf{p} = l_c/l_a$ and the recall is $\mathbf{r} = l_c/l_h$. In order to obtain a global score we find the average precision and recall. These are obtained by averaging the precision and recall values of all the keywords which appear at least once in the test ground-truth. Another metric used is the number of non-zero-recalls. This is calculated as: $\mathbf{nzr} = \sum_{l_{ic}>0} 1$.

Additionally we introduce an indicator rarely used for evaluating annotation methods. The $F1$ score is the harmonic mean of the average precision and the average recall. It enables us to look at a single value for finding the best parameters and makes it easier to compare different annotation methods. By taking the harmonic mean, the score is closer to the lesser value, so a high $F1$ score can only be achieved with both a high precision and high recall.

6.5 Evaluation on Corel5K

We performed extensive testing on this database using different underlying features for the matching method. The structure of this database has been described in the previous works (e.g.[2]). We mention only that the training set has 4500 images and the test set consists of 500 images, the size of the dictionary is 374. The metric values for matching using only one type of feature, as well using multiple features are shown in Table 3 . The numbers next to the feature type indicate the dimension of the descriptor vector.

To clearly show the advantage of using compactness over histogram distances we provide test results on the Corel5k using the same features and transfer method w_J as in [1]. In addition we provide the best results obtained using one of the proposed transfer methods - w_a refers to the rank based exponential weighing with the subscript parameter a having the optimal value. Table (2) shows that in all cases compactness ensures a higher average precision and the same or higher recall. We have used L^1 metric for comparison and not Kullback-Leibler divergence for the Lab colorspace as in [1]. The proposed weighing further improves score values boosting both precision and recall.

To find the best parameters we have performed a grid search varying several parameters in the ranges given below. Test time can be saved because matches are obtained once for each bag size and afterwards different transfer techniques can be applied. The results with the highest $F1$ score are presented in Table (3). As mentioned before, we determined that L^1 distance behaves best in this context for compactness calculation. Parameter ranges used for testing are:

- bag size - $|B| \in \{50k | k \in \overline{1,10}\}$;
- neighbourhood size - $N \in \{5, 10, 15\}$;
- weight function type - w_a, w_J or w_d;
- weight function w_a parameter - $a \in \overline{0,5}$;
- weight function w_d parameter - $b = 300$;
- considering frequency or not - $\varphi \in \{0, 10^{-3}, 2 \cdot 10^{-3}\}$;
- number centers per image - $K = 20$.

The Table 3 contains metric values using different features on the Corel5k benchmark. Entries are ordered based on $F1$ measure that guided us in deciding which method is better. The last column shows the average execution time in seconds for a single image annotation using a single threaded execution on the machine described in section 7.1. ('-' signifies no data available). Execution time is given for the best parameter combination and it depends on bag size. Simple color descriptors

TABLE 2: Comparison using the same feature type

Feature	Precision	Recall	NZR	F1
JEC+RGB	20	23	110	21.39
JEC+Lab	20	25	118	22.22
JEC+HSV	18	21	110	19.38
Comp+RGB+w_J	21.98	24.38	121	23.12
Comp+Lab+w_J	21.29	24.80	123	22.91
Comp+HSV+w_J	19.33	26.94	128	22.51
Comp+RGB+w_5	21.58	26.85	123	23.93
Comp+Lab+w_5	22.34	25.62	123	23.87
Comp+HSV+w_3	21.95	26.68	124	24.09

behaved surprisingly well compared to different texture descriptors. Also the low dimensionality of this feature permits a very low execution time. Color variants of the texture descriptors perform better than gray-scale ones. The lower part of the table contains combinations of features. This confirms that the annotation method successfully combines multiple features and produces better results than using individual features.

We now compare our results with previous state of the art results in Table 4. Each percentage is taken from the indicated reference. Optimal configuration found by our tests is: using color descriptors along with DCT63 and SIFT, with the parameters set to: $K = 20, B = 200, \varphi = 2 \cdot 10^{-3}, N = 5, a = 3$ (last line from Table (3)). We note that Compactness based methods produce similar results to SML when using the same features (see DCT63 and DCT192 in Table 3) but using SIFT proves to be better. By efficiently utilizing multiple features our simple approach outperforms many methods from the literature based on the $F1$ score including MBRM, SML, JEC, ProbSim. MRFA does not provide exactly 5 labels at annotation which helps to achieve higher scores. The better results of TagProp can be explained by the fact that it employs Metric Learning which could also be used in our context to improve results.

6.6 Effect of parameters

Some general remarks can be made about the influence of different parameters on the metric values. We analyse results from Corel5k in detail. It is possible that the behaviour on other databases is different. The effect of each parameter is analysed by fixing the other ones to their optimal values. Multiple cases are considered where necessary.

Increasing the bag size B has moderate effect on score values. This can be studied using Figure (3). Score values increase and oscillate, and in some cases reach a maximum for fairly low values of B. Because bag size linearly influences the execution time lower bag size values such as 100 or 200 can be utilized. This is practical because it achieves faster execution while maintaining near-optimal performance. The oscillating behaviour is due to the errors introduced from sampling.

TABLE 3: Compactness based annotation results using different feature types on Corel5k

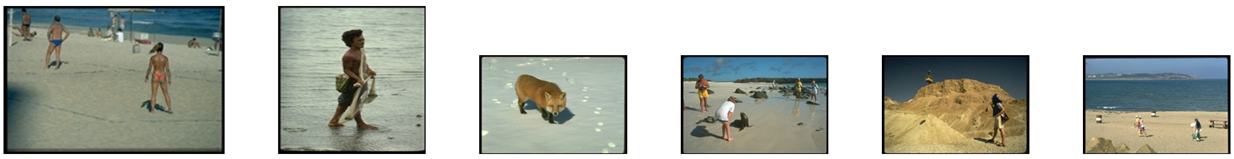
Feature	Precision	Recall	NZR	F1	exec
Gabor(12)	7.46	8.67	76	8.02	-
HOG(9)	11.22	11.57	85	11.40	-
Law(9)	13.82	17.56	105	15.47	-
color HOG(36)	14.36	17.57	109	15.80	-
WLD(48)	16.99	18.42	108	17.67	1.83
SIFT(128)	17.00	24.94	122	20.21	-
CSIFT(256)	19.49	24.51	120	21.72	-
color(9)	22.71	27.06	128	24.69	1.39
DCT(63)	22.32	28.27	129	24.95	0.48
DCT(192)	22.82	29.02	129	25.55	5.28
SIFT-OCS(384)	23.75	31.23	140	26.98	22.58
WLD + color(57)	26.45	27.88	120	27.15	4.4
SIFT + WLD + color(441)	30.19	31.99	131	31.06	18.23
SIFT + DCT63 + color(456)	30.15	32.17	133	31.13	20.62

TABLE 4: Comparison with state-of-the-art Corel5k

Method	Precision	Recall	NZR	F1
MBRM[36]	24	25	137	24.48
SML[2]	23	29	137	25.6
JEC[1]	27	32	139	29.2
ProbSim[37]	25.4	36.5	106	29.7
Compactness	30.15	32.17	133	31.13
MRFA-grid[36]	31	36	172	33.31
TagProp[15]	32.7	42.3	160	36.8

We now study the influence of neighbourhood size N. Better results were obtained using smaller N values. This may be due to the relatively small size of the database, so most of the images have few good matches among the training instances. We have found that $N = 5$ produces best results for individual features and on some occasions $N = 10$ for multiple feature case. Further fine-tuning could involve experiments considering $N \in \{1, 2, 3, 4\}$. This also demonstrates that the matching technique is efficient because the first few matches provide good labels to transfer.

Figure (4) contains metric values using different transfer techniques. We have associated $a = -2$ to JEC-type transfer and $a = -1$ to distance-based transfer. We have found that in almost all cases weighing based on rank performs best. In some cases we obtain better results with ω_d or ω_J, but the general recommendation is ω_a. The scores with $a = 0$ are almost always lower than the optimal scores obtained using $a = 3$ or $a = 4$. Overall tendency here suggests to accord significantly more importance to the best match. To provide a more encompassing overview Table (6) shows score values



(a) Test image (b) Match 1 (c) Match 2 (d) Match 3 (e) Match 4 (f) Match 5

Fig. 2: Sample matches and annotation from Corel5k - Predicted labels for test image a): **water, beach, tree, people, sand**. Showing only best five matches based on SIFT features. Note that incorrect labels from match 2 (confusion between sand and snow) get filtered out because of the transfer technique.

TABLE 5: Sample annotations using color+DCT63+SIFT from Corel5k

Prediction	people swimmers pool water athlete	stone pillar temple people sculpture	people outside museum dance tree	cars tracks formula wall straightaway	sky mountain tree snow sky
Ground-truth	people pool swimmers water	pillar temple sculpture stone	tree people dance outside	cars formula tracks wall	clouds mountain sky snow

using only RGB features, every row corresponds to a constant bag size and every column contains a different transfer technique.

If we consider frequency information it can increase overall performance ($F1$ score). However, this almost always entails an increase in precision and a decrease in recall and NZR values. The reason for this is that favouring the more frequent terms reduces the chance to annotate with rare labels. The influence of the frequency was tested using 3 different values: zero influence, minimal influence setting $\varphi = max f_l$ as suggested, and twice the previous value. The second case give higher $F1$ score in general.

Computation time varies in accordance with the time complexity formulas derived in section 5. It is linear with respect to feature dimension and also with respect to bag size. These two parameters can control the execution time, modifying the bag size has only minor negative effects on annotation performance. Even though the IAPR-TC12 and ESP-game datasets are much larger annotation time still remains fairly low due to the optimizations mentioned (halting calculation when distance exceeds the current N-th maximum).

6.7 Evaluation on IAPR-TC12

This image collection consists of 20,000 still natural images taken from locations around the world and comprising an assorted cross-section of still natural images[38]. The same images are used from the IAPR-TC12 database as those in [1] in order to compare results in a correct manner. This database is larger, the training set numbers 17825 images and the test set contains 1980 images with 291 labels. The image annotations and test/training split is obtained from the files located at

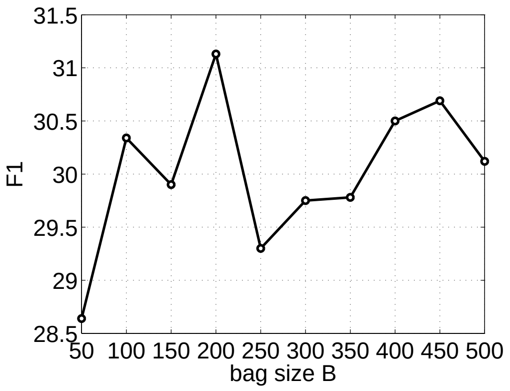
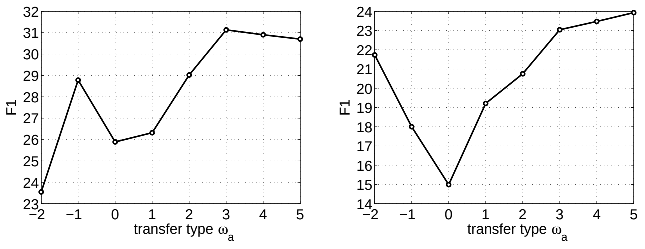


Fig. 3: The influence of bag size - color+DCT63+SIFT



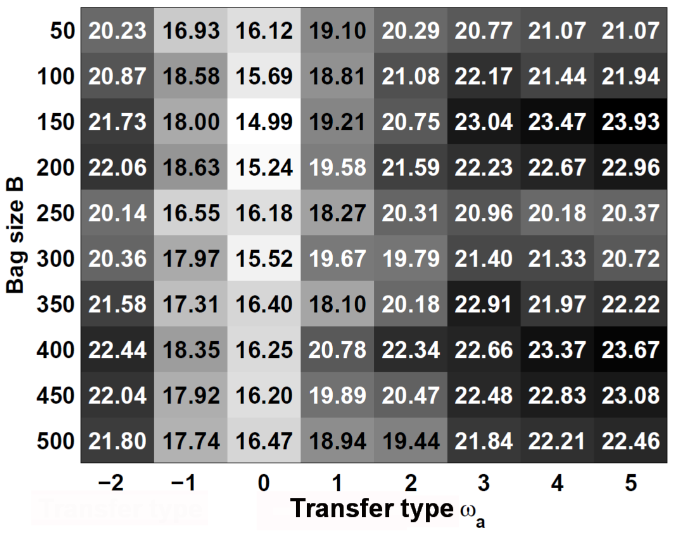
(a) multiple features (b) RGB only

Fig. 4: The influence of transfer type

the web-page [1].

The metric values are calculated using all the labels from the ground-truth. This is the right way to obtain the number of correct labels however recall values will be lower. This is so because we only provide 5 labels,

1. Makadia annotation files



Bag size B \ Transfer type ω_a	−2	−1	0	1	2	3	4	5
50	20.23	16.93	16.12	19.10	20.29	20.77	21.07	21.07
100	20.87	18.58	15.69	18.81	21.08	22.17	21.44	21.94
150	21.73	18.00	14.99	19.21	20.75	23.04	23.47	23.93
200	22.06	18.63	15.24	19.58	21.59	22.23	22.67	22.96
250	20.14	16.55	16.18	18.27	20.31	20.96	20.18	20.37
300	20.36	17.97	15.52	19.67	19.79	21.40	21.33	20.72
350	21.58	17.31	16.40	18.10	20.18	22.91	21.97	22.22
400	22.44	18.35	16.25	20.78	22.34	22.66	23.37	23.67
450	22.04	17.92	16.20	19.89	20.47	22.48	22.83	23.08
500	21.80	17.74	16.47	18.94	19.44	21.84	22.21	22.46

TABLE 6: The influence of parameters on F1 score

TABLE 7: Compactness based annotation results using different feature types on IAPR-TC12

Feature	Precision	Recall	NZR	F1	exec
color(9)	23.89	23.63	216	23.76	2.0
DCT(63)	25.24	24.64	225	24.94	6.3
SIFT(384)	31.82	32.45	245	32.13	17.0
SIFT+DCT+color	42.9	22.6	228	29.6	44.0

TABLE 8: Comparison with state-of-the-art IAPR-TC12

Method	Precision	Recall	NZR	F1
MBRM[39]	24	23	223	23.48
JEC[1]	28	29	250	28.49
Compactness	42.9	22.6	228	29.6
TagProp[15]	46.0	35.2	266	39.88

and in cases where in the ground-truth there are more than 5, we inevitably end up marking some labels as not recalled.

We provide some sample annotations for this dataset in Table 9. Notice that a lot of images have much more labels than 5. The results for this database (Table 7) again indicate that fairly good results can be obtained using simple color descriptors. However, SIFT features outperform other features mostly by reaching an $F1$ score of 32.13. It can be seen that the combination of different features is more successful on this database. Average precision value increases with 11%. Note that combining features results in lower recall and higher precision values. This is natural since more features provide more ”opinions” about the correct label and the consensus tends to reflect the truth.

The comparison made in Table 8 shows that Compactness obtains much better precision than MBRM and JEC (by 15%). Recall and NZR values are lower, but we mention here that using JEC-type transfer similar values were obtained as in [1].

TABLE 10: Compactness based annotation results using different feature types on ESP-game

Feature	Precision	Recall	NZR	F1	exec
Law-color(30)	16.36	16.07	217	16.22	2.32
WLD(48)	19.65	17.33	228	18.42	3.61
color(9)	19.73	19.28	230	19.50	1.56
DCT(63)	21.49	20.50	236	20.99	7.92
SIFT(384)	22.75	20.42	230	21.52	35.13
WLD+color	31.07	19.78	227	24.17	11.63
SIFT+DCT+color	34.67	21.29	233	26.38	39.45

TABLE 11: Comparison with state-of-the-art ESP-game

Method	Precision	Recall	NZR	F1
JEC[1]	22	25	224	23.4
Compactness	34.67	21.29	233	26.38
TagProp[15]	39.2	27.4	239	32.25

6.8 Evaluation on ESP game

This dataset is the result of an experiment involving collaborative human annotation. The subset of pictures used is the same as in[1]. More exactly: 19659 training images, 2185 test images, annotated with 269 different labels. An advantage of this set is that it is a result of an agreement between multiple annotators, so annotations are not biased by individual preference.

Table 10 contains results using a limited set of features and their combination. Five sample annotations are provided in Table 12. In this case WLD texture descriptor and color descriptors collaborate well. This may be so because in this set texture can discriminate instances better than in previous datasets. To enable comparison with the existing methods we summarize other results in Table 11.

6.9 Evaluation on NUS-WIDE

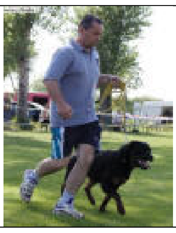
NUS-WIDE[40] is a large image dataset consisting of 269,648 images and associated tags from Flickr. This was created by a research team from the National University of Singapore, who also provide tags for 81 concepts. It is suitable for testing label transfer annotation algorithms. We have obtained this dataset by downloading the images using the provided URLs, however 36515 images are either missing or are blank, which can be detrimental for annotation precision.

We have carried out experiments using the proposed color descriptor and we have compared the obtained results with the NUS-wide Lab histogram based k-NN annotation baseline [40]. The only difference between the two methods is the distance calculation. In the first case we have used compactness and in the second case the L1 distance between global Lab histograms as in [40]. We could not directly use the feature vectors provided with the dataset because they are global feature vectors and compactness operates on local features, but the underlying feature type is the same. For every test image we

TABLE 9: Sample annotations using color+DCT63+SIFT from IAPR-TC12

Prediction	view river jungle middle cloud	pool people woman tree man	building front ornament trouser jacket	bike country helmet side short	sky mountain cloud desert bush
Ground-truth	cloud hill jungle middle palm range river view	chair man people pool woman	building column front jacket ornament person trouser	bike cap country cycling cyclist hand helmet jersey racing road short side	cloud desert mountain shrub sky

TABLE 12: Sample annotations using SIFT from ESP-game

					
Prediction	people sky crowd tree blue	man black dog grass tree	coin gold round circle money	sky blue people tower building	old man shirt glasses book
Ground-truth	crowd man people pole sky tree	black dog grass green guy man run shoes white	circle coin gold old round square	blue building people sky tower	book glasses green hand man old shirt

generate 5 labels. If the ground truth information specifies n labels we evaluate the performance on the first m labels, where m = min(5; n). We present the annotation performance in Figure 5. It is given in terms of precision for each concept and in terms of mean average precision (MAP). Concepts with more training examples - such as clouds, person, sky - have a significantly higher precision value for both methods. The k-NN based method has more concepts with non-zero precision and performs better for some concepts with more training examples. However, for most concepts compactness provides a higher precision. The MAP obtained with compactness is of 6.21 in comparison with 4.8 corresponding to the k-NN based classification algorithm.

7 Conclusions

In this paper we have presented a new technique for matching images. This can be employed in a nearest neighbour image annotation method. Several transfer techniques have been proposed and analysed.

In the experimental section we have provided metric values on four benchmarks to validate the presented method. This demonstrates that compactness outperforms the histogram distance. Furthermore the proposed transfer technique improves score values. The annotation method using multiple features does better than most the state-of-the-art algorithms. We stress that our goal was to show that compactness can be considered a useful alternative to image matching and not to provide a complete algorithm. This would entail careful feature selection, balancing the weights of each feature, changing the distance functions.

We enumerate the advantages of the presented approach:

- conceptually simple;
- simple and fast training process;
- flexible - can easily work with different underlying low level image descriptors;
- can efficiently combine different feature types (e.g. color and texture);
- does not need segmented images;
- does not need negative examples for training;
- robust - even with untuned parameters provides good results;
- competes with and outperforms complex learning algorithms.

As a drawback, we mention that the matching phase is more time consuming than some currently used methods (such as distance between global histograms from the bag-of-words approach) and is linearly dependent on the database size, like any k-NN matching annotation algorithm. However, we have shown by indicating execution times that even so, annotation time is well within acceptable ranges. This is why this approach can be utilized to provide good quality annotations in 5-10 seconds on the machine described in Section 7.1.

Contributions that result from the presented research:

- original idea to use compactness as a ”distance” measure between images, that enables us to effectively compare local descriptors;
- providing a formalism for defining label transfer

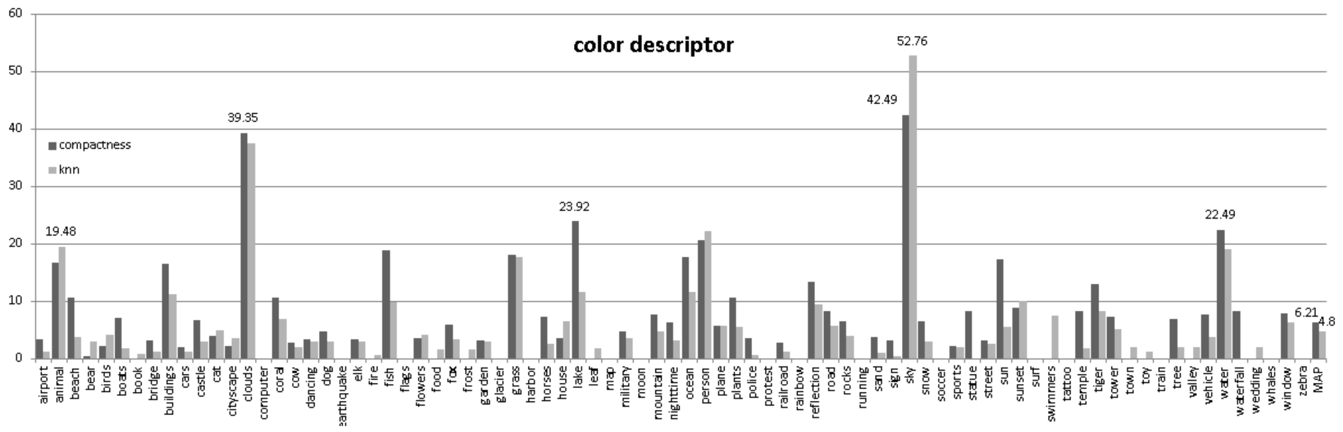


Fig. 5: Precision values for each concept and MAP on the NUS-WIDE dataset

techniques;

- devising and testing of elementary transfer types;
- validation and result analysis on 4 different datasets that proves the efficiency of the method.

Future work will involve experimenting with different feature types and their various combinations in order to obtain optimal results. Different implementation ideas for matching execution time reduction are under consideration. New weight function types for transfer are also under research. Another variant of the algorithm would be to use GMMs to represent images instead of k-means centers. It would also be desirable to find distance functions that weigh important features more.

Acknowledgements

This work was supported by the Romanian Ministry of Education and Research, through the project ”SmartCoDrive”, code PN II PCCA 2011 3.2-0742, from 03.07.2012 (2012-2015). The authors would like to thank Kobus Barnard for providing the Corel5K data set. We thank A. Makadia for making public the annotations and training/testing splits employed in his work for the IAPR-TC and ESP datasets.

References

[1] A. Makadia, V. Pavlovic, and S. Kumar, “A new baseline for image annotation,” in *ECCV*, 2008, pp. III: 316–329.

[2] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos, “Supervised learning of semantic classes for image annotation and retrieval,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 394–410, Mar. 2007.

[3] Blei, D. M., Jordan, and M. I., “Modeling annotated data,” in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. Multimedia information retrieval, 2003, pp. 127–134.

[4] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *CVPR*, 2006, pp. II: 2169–2178.

[5] F. F. Li and P. Perona, “A bayesian hierarchical model for learning natural scene categories,” in *CVPR*, 2005, pp. II: 524–531.

[6] J. Li and J. Z. Wang, “Real-time computerized annotation of pictures,” *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 985–1002, Jun. 2008.

[7] J. Li and J. Z. Wang, “Automatic linguistic indexing of pictures by a statistical modeling approach,” *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 25, no. 9, pp. 1075–1088, 2003.

[8] A. Llorente, R. Manmatha, and S. M. Rüger, “Image retrieval using markov random fields and global image features,” in *CIVR*, S. Li, X. Gao, and N. Sebe, Eds. ACM, 2010, pp. 243–250.

[9] A. Torralba, R. Fergus, and W. T. Freeman, “80 million tiny images: A large data set for nonparametric object and scene recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[10] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “SUN database: Large-scale scene recognition from abbey to zoo,” in *CVPR*. IEEE, 2010, pp. 3485–3492.

[11] T. Serre, L. Wolf, and T. Poggio, “Object recognition with features inspired by visual cortex,” in *CVPR*, 2005, pp. II: 994–1000.

[12] J. Feng, B. Ni, Q. Tian, and S. Yan, “Geometric ℓp-norm feature pooling for image classification,” in *CVPR*. IEEE, 2011, pp. 2697–2704.

[13] L. Cao, R. Ji, Y. Gao, Y. Yang, and Q. Tian, “Weakly supervised sparse coding with geometric consistency pooling,” in *CVPR*. IEEE, 2012, pp. 3578–3585.

[14] H. Dunlop, “Scene classification of images and video via semantic segmentation,” in *CVPR Workshop on Perceptual Organization in Computer Vision*, 2010.

[15] M. Guillaumin, T. Mensink, J. J. Verbeek, and C. Schmid, “Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation,” in *ICCV*. IEEE, 2009, pp. 309–316.

[16] J. Verbeek, M. Guillaumin, T. Mensink, and C. Schmid, “Image annotation with tagprop on the MIRFLICKR set,” in *11th ACM International Conference on Multimedia Information Retrieval*, 2010.

[17] A. Farhadi, I. Endres, D. Hoiem, and D. A. Forsyth, “Describing objects by their attributes,” in *CVPR*, 2009, pp. 1778–1785.

[18] S. J. Hwang, F. Sha, and K. Grauman, “Sharing features between objects and their attributes,” in *CVPR*. IEEE, 2011, pp. 1761–1768.

[19] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li, “Object bank: A high-level image representation for scene classification semantic feature sparsification,” in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*. Curran Associates, Inc, 2010, pp. 1378–1386.

[20] S. P. Lloyd, “Least squares quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, pp. 128–137, 1982.

[21] Arthur and Vassilvitskii, “k-means++: The advantages of careful seeding,” in *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 2007.

[22] P. A. Viola and W. M. Wells, “Alignment by maximization of mutual information,” *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, Sep. 1997.

[23] E. Cebuc, A. Suciu, K. Marton, S. Dolha, and L. Muresan, “Implementation of cryptographic algorithms on a grid infrastructure,” in *aqtr, vol. 2, pp.1-6, 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 2010.

Robust pallet detection for automated logistics operations

Robert Varga and Sergiu Nedevschi
Technical University of Cluj-Napoca, Romania
{*robert.varga, sergiu.nedevschi*}*@cs.utcluj.ro*

Keywords: object detection; pallet detection; stereo reconstruction

Abstract: A pallet detection system is presented which is designed for automated forklifts for logistics operations. The system performs stereo reconstruction and pallets are detected using a sliding window approach. In this paper we propose a candidate generation method and we introduce feature descriptors for grayscale images that are tailored to the current task. The features are designed to be invariant to certain types of illumination changes and are called normalized pair differences because of the formula involved in their calculation. Experimental results validate our approach on extensive real world data.

1 INTRODUCTION

Automated Guided Vehicles perform (AGVs) logistics operations without human intervention. This requires the existence of a sensor capable of estimating the position of the pallet that needs to be loaded by the machine. This work focuses on developing a machine vision-based detection system for pallets.

Pallets are wooden supports designed to hold goods and are easily graspable by the forklift because of its pockets. Pallets are standardized and for our purposes they are handled from only one side. We desire a flexible detection module that can identify the relative position of the pallet from any image under various lighting conditions.

Stereo cameras offer a good solution for 3D sensing applications. The cost of such systems is lower compared to laser scanners. Also camera systems offer a full 3D view as opposed to 2D scan lines and the possibility of high level reasoning on data. The main drawback of such systems is the difficulty of working in poor and rapidly changing illumination conditions.

We have studied previous vision-based attempts at this problem and found that they are lacking because of the following reasons: they rely on features that do not possess good invariance properties; detection performance is poor in general and especially in dark regions; most systems are not thoroughly evaluated.

For the above mentioned reasons we propose improvements which constitute the main contributions of this work:

- Original candidate generation method that enables fast detection by quickly rejecting certain regions;
- The proposal of new grayscale features invariant to certain types of illumination changes.

The paper is organized as follows: Section 2 presents existing approaches and important contributions from the image processing literature: edge detection; feature vector extraction; classification. In Section 3 we describe our proposed system and give details about each processing step. Section 4 shows experimental results that validate our system. Section 5 concludes the paper.

2 RELATED WORK

2.1 Sensor types

The specific topic of load handling is not a well-researched area. Approaches for autonomous load handling use different types of sensors to obtain an understanding about the environment. In (Weichert et al., 2013) the authors discuss the advantages of several sensors for this task. We will group these approaches into two main categories based on the sensors used: range sensors and vision-based sensors (monocular or stereo cameras). In the following we describe relevant approaches from each category.

Some available systems rely on laser scanner data. In most cases the sensor provides data along a 2D scanline. Using laser has the advantage over cameras that it is able to operate in complete darkness and it is not affected by lighting conditions.

In (Walter et al., 2010) a detection system is presented for the autonomous manipulation of a robotic lift truck. The authors use closest edge detection applied on the sensor point cloud. SICK industries manufacture laser scanners for multiple purposes. A work from (Bostelman et al., 2006) presents a pallet detection method using such sensors. A solution is provided for unloading operations inside trucks. The walls of the trucks are detected by applying a Hough transform (Hough, 1962), (Duda and Hart, 1972). The paper (Katsoulas and Kosmopoulos, 2001) uses laser sensors to detect the positions of boxes of standard dimensions. Kinect sensors can be employed for distance estimation as in (Oh et al., 2013). However, they are not suitable for an industrial environment and they offer a small field of view.

A hybrid approach from (Baglivo et al., 2011) combines two types sensors: a laser scanner and a camera. A fusion is performed at object level between the detection from the color image and the points from the laser. Edge template matching with distance transform is applied on the color image. Both sensors must agree on the detection, ensuring robustness. The system requires the calibration of the two sensors. The authors have evaluated their system on 300 examples with results indicating a good localization precision. They have found difficulties due to lighting conditions in 5 cases.

Vision-based approaches employ multiple cues: in (Kim et al., 2001) line-based model matching is used; (Pages et al., 2011) performs colour-based segmentation; (Seelinger and Yoder, 2006) uses easily identifiable features (landmarks, fiducials); (Cucchiara et al., 2000) employ corner features, region growing and decision tree; in (Byun and Kim, 2008) least squares model fitting is applied. Most authors perform evaluation on a small dataset or in laboratory conditions. The work (Seelinger and Yoder, 2006) presents results on 100 operations with a success rate of 98%. Also, their approach requires the installation of landmarks on each pallet.

A paper from (Varga and Nedevschi, 2014) presents a detection approach relying on integral channel features. The authors evaluate their system on an impressive dataset containing 8000 test images. Other approaches include: (Nygårds et al., 2000), (Prasse et al., 2011), (Pradalier et al., 2008).

2.2 Detection approaches

Sliding window object detection is one of the most commonly used approaches employed in the technical literature. Typical examples of particular detectors include face detectors (Viola and Jones, 2001), (Yang et al., 2002), pedestrian detectors (Dollár et al., 2012), (Benenson et al., 2014), (Dollár et al., 2014). The success of this general approach can be attributed to the fact that it uses a powerful classifier to discern between background and target object. Since the classifier is a cascade it eliminates zones without objects quickly.

Features for detection should capture structure, texture and color if possible. Some of the more important features that are relevant for this work are: any edge feature defined on the image gradient (Mikolajczyk et al., 2003); Histogram of Oriented Gradients (Dalal and Triggs, 2005) - developed originally for pedestrian detection; Haar features (Viola and Jones, 2001); integral channel features (Dollar et al., 2009); CENSUS features (Zabih and Woodfill, 1994); Local Binary Patterns and their histograms (Ojala et al., 1994), (Ojala et al., 1996).

Fast and accurate detection is possible with boosted classifiers (Schapire, 1990) and soft cascades (Bourdev and Brandt, 2005). This was first proposed by Viola & Jones for face detection in (Viola et al., 2005) but since has been adopted to pedestrian detection (Dollár et al., 2010). Many top performing methods on benchmarks utilize such classifiers for their speed.

3 Proposed approach

Our proposed solution relies on exploiting two main sources of visual information: intensity images and stereo cameras. The intensity image provides information about 2D localization of the pallets. The stereo cameras are used to obtain the 3D position and orientation of the pallet relative to the cameras. We have found 3D-based detection less reliable because of poor reconstruction quality at pallet pockets.

Although our pallet detector is an application of the standard sliding window technique our system has to generate bounding boxes that are tight and precise. The requirements regarding exact localization are strict. Pallets need to be localized with a precision of 1 cm. This explains why experimenting and developing specific features are required. Also, the detection method should be highly accurate.

In the following we first present the processing steps required for detection. Stereo reconstruction is described at a glance. Next, we provide details about the candidate generation module. Afterwards, we introduce descriptive features proposed specifically for pallet detection. We have proposed several validation steps at the post processing stage for more robustness and enhanced localization.

3.1 Stereo reconstruction

Reconstruction is performed with semi-global matching and CENSUS local descriptors. Our system makes use of the rSGM implementation (Spangenberg et al., 2014). CENSUS/LBP descriptors have been found to be a reliable local descriptor for many practical applications including those from the automotive industry. Semi-global matching (Hirschmuller, 2005) offers the advantage of having smooth disparity maps and it is fast enough for our purposes. The rSGM implementation is fast and runs on CPU. It includes optimizations with SSE instructions and it is a top performing method on stereo benchmarks.

3.2 Edge and line detection

For improving edge detection quality we rely on extracting normalized gradient values. This has been proposed and employed in calculating HOG (Dalal and Triggs, 2005) features and also in modern pedestrian detection algorithms (Dollár et al., 2010). Normalized gradient values are obtained by box-filtering the gradient magnitude and dividing the original gradient magnitude and other channels by the filtered values. This ensures successful edge detection even in dark regions.

In the following we provide the exact steps for calculating the normalized gradient maps. The gradient components along the x and y axes are obtained in a standard way by convolution with Sobel filters:

$$G_x = I * S_x \tag{1}$$

$$G_y = I * S_y \tag{2}$$

The gradient magnitude is defined as the L_1 norm of the two components:

$$M = |G_x| + |G_y| \tag{3}$$

The box filtered magnitude will act as a normalization factor:

$$\hat{M} = M * B \tag{4}$$

where B is a square box-filter of dimension w x w. Typical values for w are odd numbers from the interval $[5, 25]$. It is important to note that this filtering can be performed in $O(1)$ time per pixel for any filter size w. Filtering with a Gaussian would increase the computation with no significant benefit. The normalized magnitude and the normalized gradient components are obtained by dividing the original values with the box filtered gradient magnitude (pixel by pixel):

$$\overline{M} = M/(\hat{M} + \varepsilon) \tag{5}$$

$$\overline{G_x} = \lambda \cdot G_x/(\hat{M} + \varepsilon) \tag{6}$$

$$\overline{G_y} = \lambda \cdot G_y/(\hat{M} + \varepsilon) \tag{7}$$

All division and summation operations in the previous definitions are carried out element by element. The small constant $\varepsilon = 5e-3$ avoids division by zero. The multiplier λ is required for converting the normalized values into the $[0, 255]$ interval.

Intuitively this operation produces strong responses where the relative change in intensity is large compared to the average intensity change in the neighboring region. This improves edge detection in poorly illuminated regions.

3.3 Candidate generation

Considering all possible positions for sliding window detection results in a large number of possible candidates (see experimental results sections for typical numbers). It is not feasible to classify each possible candidate to see whether or not it is a pallet. This is why it is important to have a good candidate generation module. The main characteristics should be:

- high coverage - the module should not miss any real pallet positions (i.e. low number false negatives, high recall);
- fast to evaluate - can be executed instantly in comparison to following modules;
- high rejection rate - it should accept only a limited number of candidates to speed up, help and validate further processing steps.

Currently we are working with two main approaches for candidate generation. These improve the baseline approach which is just to take every possible rectangle at valid positions and scales. Edge-based candidate generation relies on edge detection while the other alternative uses stereo information. We provide details in the following.

3.3.1 Edge-based candidate generation

Since the frontal view of pallets is a rectangle the candidate generator should produce a list of candidate rectangles. For this we first employ the normalized gradient in the y direction as in eq. 7 to detect important horizontal lines called horizontal guide lines. A histogram that accumulates gradient values along each line is used to find local maxima. In other words we perform a projection along the horizontal direction. Since the structure of the image usually contains strong horizontal lines this step is robust and we can rely on the extracted guidelines later on.

Vertical lines are detected only between guideline pairs that respect the dimension constraints. These

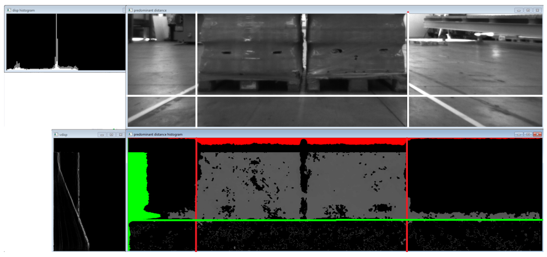


Figure 1: Stereo-based candidate generation; top-left - disparity histogram; top-right - original image with reduced region of interest marked; bottom-left - v-disparity map; bottom-right - disparity map with only the neighborhood of principal disparity highlighted, also the projections along the two axis are visualized and the new boundaries detected

lines are detected where the sum of gradient along x direction exceeds a certain percentage (10 %). The resulting candidate rectangles arise from combining vertical edges that fit the dimension constraints regarding width, height and aspect ratio.

3.3.2 Disparity-based candidate generation

We can limit the region of interest for processing by considering only the objects with fronto-parallel surfaces. The reason for this is that the axis of the stereo system is roughly perpendicular to the target pallets. Such objects appear as a line in the v-disparity and u-disparity map. Also, they lie on the disparity plane with high appearance frequency. We define the principal disparity as the disparity value that corresponds to the highest local maximum from the disparity histogram. The highest local maximum is considered because this corresponds to the obstacle in front of the camera. We call principal disparity plane the plane obtained by selecting only points that are close to the principal disparity. This is equivalent to highlighting only the objects that are closest from the visual scene.

Once the principal disparity value is determined the region of interest can be limited to the zone where such disparity values are frequent. We do this by starting from the extremities (left, right and bottom) and shrink the boundary of the original region of interest until the frequency of the preponderant disparity exceeds a limit (see Figure 1). Principal disparity also gives us information about the approximate and expected dimensions of the pallets in the image plane. This also reduces the number of possible candidates. We apply normal edge-based candidate generation on the reduced region of interest and apply the new constraints found regarding the size of the pallet.

3.4 Feature extraction

The principal characteristic features of pallets are their structure. It is therefore important to have features that capture the structure of the pallet. Previous work used integral features defined on manual rectangular subregions, edge features, Hough transform and corner features. We have experimented with other features for two reasons: to capture the structure of the pallet in a concise way and to ensure a representation that is more invariant to illumination changes.

3.4.1 Proposed grayscale features - *normalized pair differences*

Our goal was to introduce a grayscale feature that is sufficiently descriptive and also invariant to illumination changes. A simple way to model illumination change is multiplication by a constant value. Technically, this represents a gain change, but it is a good approximation. The features should be unaffected by this kind of operation. Weber's law states that "just-noticeable difference between two stimuli is proportional to the magnitude of the stimuli" (Ross and Murray, 1996). Features therefore should be defined as ratios to capture relative change. This idea was employed before in other descriptors such as WLD (Chen et al., 2010), however here we propose a different form.

We use this principal to calculate our features. An option would be to normalize features by dividing with the mean of the surrounding region. However, we do not want the surrounding region to affect the descriptor of the pallet. Instead we want and invariant representation that will be the same for the same pallet. This observation leads to the necessity of defining features using only the intensity values inside the bounding box.

First, the bounding box is resized to a fixed size (5 x 20). This reduces the pallet to a smaller number of intensity values and also amounts to a low pass filtering. It is necessary to remove the regions corresponding to pallet pockets. These regions are not part of the object and bear no relevance to the detection task. Second, we take each possible pair of intensity values. The sample intensity values are denoted f_i and are obtained from the previous downsampling operation. See Figure 2 for illustration of the defined concepts.

We denote these features as normalized pair differences (**npd**). Feature values are calculated by considering all pairs, taking the difference and dividing by the first value from each pair. A sigmoid-type function is applied afterwards. Intensity features where the mask is 0 are not used:

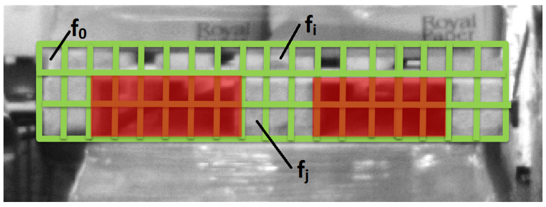


Figure 2: Feature grid of 3 x 16 overlaid on a pallet. After resizing each square cell will become one single intensity value. The cells from the red region are not used (mask = 0)

$$D_k = tan^{-1}\left(\frac{f_i - f_j}{f_i + \varepsilon}\right) \tag{8}$$

The role of the inverse tangent function is to limit the range of the features, i.e. it is used as a sigmoid-type function. All possible pairs taken from valid positions form a signature that describes the pallet. Adding a small number $\varepsilon = 1e-2$ to the denominator avoids checking for zero division and simplifies the code for the algorithm. It is easy to see that if all intensity values are uniformly multiplied with a value α, signifying a change in illumination, the value of the descriptor does not change.

This signature will be compared by the classifier at detection time. The signature should remain roughly the same even after illumination changes. We use a rectangle grid of dimension 5 x 20. The dimension of this type of feature vector is 1350 (some pairs are missing from the $\binom{100}{2} = 4950$ because we exclude the zones from the pockets).

3.4.2 Edge features

We also define edge features on rectangular areas near the pallet boundary in order to help in precise localization. The edge features are calculated on the normalized gradient channel. The descriptors are defined in equation 9 as normalized sums of the normalized gradient values calculated on rectangular areas depicted in Figure 3. The upper edges of the pockets are not used since they can be covered by plastic hanging from the palletized goods. The dimension of this type of feature vector is 9.

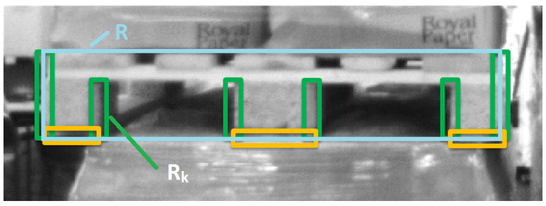


Figure 3: Support regions for calculating normalized gradient sums

$$E_k = \frac{1}{area(R_k)} \sum_{(x,y)\in R_k} \overline{M}(x,y) \tag{9}$$

3.4.3 LBP histogram

For texture features we use a histogram of local binary patterns. This has shown to be a reliable texture descriptor and it is also employed in many stereo matching systems. LBP descriptors also possess good illumination invariant properties since only the relative order of intensity values are important. The dimension of this type of feature vector is 256.

$$H_k = \frac{1}{area(R)} \sum_{(x,y)\in R} [mask(x,y) = 1][lbp(x,y) = k] \tag{10}$$

The last definition uses the standard Iverson bracket notation: $[expr]$ is 1 if the logical expression $expr$ is true and 0 otherwise. The histogram is simply the count of each type of LBP feature that is in a valid position. The counts are normalized appropriately with the area of the bounding box R(see Figure 3). The area refers to only the zone from the rectangle that is not invalidated by the mask.

3.5 Classification and detection

Boosted decision trees offer both high classification accuracy and fast prediction time. Since prediction is made by comparing individual features against threshold the time taken does not depend on the dimension of the feature vector. Since we know beforehand the number of desired pallets we can keep only the pallets with the highest confidence values.

The classifier is trained using the positive examples available from the manual annotations. Negative samples are generated automatically from each training image from regions that surely do not contain any pallets. Retraining the classifier with hard negatives has proven not to be helpful.

3.6 Refinement and validation

We have found it best to enable the detector to return matches that are not precisely localized and then refine their position and scale. Bounding boxes that have good aspect ratio will have their scores improved by a multiplicative factor of 2. In cases where 2 pallets are required to be detected we boost the scores of each candidate pair that lies on the same y position and have approximately the same size.

The standard non-maximum suppression that is applied to every overlapping bounding box pair is slightly modified. In case of an overlap only the candidate with the higher score is retained. Two bounding boxes are considered to overlap if the overlap along the x axis is larger than 10 % and if the overlap along the y axis is larger than 0. A small overlap between detected bounding boxes along the x axis is possible when the pallets are far away and close to each other.

Since we have knowledge about the number of pallets that are required to be detected we can return only the most confident detections. Final pallet position is reconstructed from the plane fitted on the rectangular bounding box that is detected. This also provides us the orientation of the object.

4 Experimental results

All processing steps have been implemented in C++. The project uses OpenCV library for low-level image processing functions such as the bilateral filter, box filter, image reading/writing.

4.1 Feature properties

We run tests to evaluate the invariance properties of the features we use. A sequence containing 317 measurements is recorded of a static pallet with varying exposure time. The change in exposure time modifies the appearance of the pallet from barely visible to saturated white. Descriptors are extracted from the same region. We evaluate the mean and the maximum of the standard deviations of each component. Also, the Euclidean distance is calculated between each descriptor pair and the mean and the maximum is found. We divide by the feature dimension for a fair comparison. All feature values are normalized to be in the range [-1, 1]. Table 1 shows the results, entries are ordered from top to bottom from least invariant to most invariant (we show only values for differences). The **npd** features have similar properties as the **lbp** histogram but they are more descriptive and structure information is maintained. These features change less under the tested conditions compared to the intensity and edge features.

Feature	dim.	mean diff.	max diff.
intensity	53	3.78e-02	1.09e-01
edge	53	2.44e-02	4.23e-02
npd	1327	2.74e-03	6.11e-03
lbp	256	2.93e-04	8.43e-04

Table 1: Measuring exposure invariance properties of different descriptor types

4.2 Pallet detection accuracy

For evaluation purposes we use the same dataset and the same criteria as the work from (Varga and Nedevschi, 2014). The dataset was acquired from a real warehouse and was manually labeled. The detector is trained on a subset of the whole dataset. This part does not overlap with the test set on which we perform all evaluation. Two test sets are available: test set 1 which is somewhat similar to the training set having been acquired in the same recording session, also this contains the most annotated pallets; and test set 2 originating from a separate recording session. The second test set is more challenging and contains mostly difficult cases. The composition of the sets is as follows: training set contains 467 images and 891 labeled pallets (there can be zero or more than one pallet in each image); test set 1 contains 7122 images and 9047 labeled pallets; test set 2 contains 224 images and 356 labeled pallets. The final model installed in the system on the AGV was trained on all the available data.

The values of some of the parameters are given in the following. Region of interest dimensions: 400 x 1440; Bilateral filter sigma in the coordinate space $\sigma_x = 5$; Gradient box filter dimension $w = 15$; Gradient multiplier $\lambda = 40$; Horizontal edge detection non-maximum suppression neighborhood size $h = 3$; Vertical edge detection non-maximum suppression neighborhood size $v = 3$.

Since all scores depend on determining whether or not two rectangles overlap sufficiently we state precisely what we consider as an overlap. Usually for object detection intersection over union (PASCAL VOC criteria) is used to determine overlap. Here, we define the absolute positioning error along the x axis E_x as the difference between the union and overlap of the intervals along the x axis of the two rectangles. E_y, The absolute positioning error along the y axis is defined analogously. We consider an overlap a **precise** match if $E_x \leq 15$ and $E_y \leq 15$; and a **normal** match if $E_x \leq 50$ and $E_y \leq 50$. Our overlap measures are more strict than the relative overlap of the pascal VOC measure because of the system requirements. E_x is approximately equals twice the positioning error in pixels. A precise position amounts to an error of 7.5 pixels $\approx$ 1.5 cm using our hardware setup.

Candidate generation algorithms are evaluated by checking if every bounding box defined in the ground truth is provided by the module. The percentage of recalled bounding boxes is defined as the coverage. A box is recalled if it overlaps sufficiently with the ground-truth box. We have considered an absolute overlap when the absolute positioning error is less than 15 px along both axis. The results with different methods on the training dataset is presented in Table 2. Even though we do not achieve full coverage, rectangles near the ground truth are obtained. Using post processing and corrections the localization precision of the detection can be improved.

Method	**Coverage**	**Avg. nr. candidates**
All(5,5)	100 %	1370k
Grid(7,7)	99.40 %	508k
Edge(5,3)	99.52 %	374k
Normalized gradient (3,3,15)	98.81 %	35k

Table 2: Comparison of different candidate generation schemes. The approach from the last row offers an acceptable coverage while drastically reducing the number of candidates generated per image. The numbers in the parentheses indicate the step size in horizontal and vertical direction and the filter size (where applicable).

We now turn to evaluating pallet detection accuracy. Table 3 shows the detection accuracy on the two test sets using different configurations. The effect of adding new feature types is evaluated. We present test results using a boosted classifier with 100 and 1000 weak learners respectively. The number of negatives signifies per image is set in accordance with the power of the classifier. The training set can contain more than 1 million examples. If we weigh the error on positive instances more by ω times we can obtain a more precise localization. The npd-linear feature performs worse on the harder test set 2. Clear improvements can be seen with the new features and each additional feature improves the detection accuracy. Missed detections arise when the images are too dark, when the pallets are not fully visible or when false detections appear due to glare from the plastic covering the palletized goods.

The typical running times for the processing modules are: rectification and disparity map generation 60 ms; candidate generation 20 ms; feature extraction 800 ms; classification 300 ms. All these operations are performed on the region of interest of size 400 x 1440 = 0.576 Mpixels. Training the classifier with approximately 1 million examples and the feature vector of dimension 1591 takes a couple of hours.

5 CONCLUSIONS

The purpose of this work was to present a pallet detection method. We have improved on existing results by designing and implementing a better candidate generation module and providing better features. Detection accuracy was evaluated on a large test set and compared to an existing approach. Our system performed much better in every category.

	test set 1		**test set 2**	
Features	**normal**	**precise**	**normal**	**precise**
100 weak learners + 100 negatives/image				
integral ftrs.	79.0	64.2	-	-
npd	80.6	65.1	80.9	40.1
npd+edge+lbp	97.1	90.2	87.7	46.0
npd+edge+lbp + $\omega = 10$	97.7	92.6	87.7	70.5
1000 weak learners + 1000 negatives/image				
integral ftrs.	92.0	75.4	77.0	38.0
npd+edge+lbp	100	94.9	93.5	65.7
npd+edge+lbp + $\omega = 2$	98.9	95.4	91.9	68.8

Table 3: Detection accuracy in percentages for multiple model configurations; evaluation on both test sets; normal localization and precise localization is considered. For comparison we include the integral features from (Varga and Nedevschi, 2014) (code is provided by the authors).

We have learned that normalized gradient values enable a more robust edge detection and permit us to generate a small set of candidates. More descriptive features result in higher detection accuracy.

Future work will involve optimizing the execution time of the feature extraction module because it currently dominates the pipeline. Increasing the localization precision with post-processing steps is also of interest.

ACKNOWLEDGEMENTS

PAN-Robots is funded by the European Commission under the 7th Framework Programme Grant Agreement no. 314193. The partners of the consortium thank the European Commission for supporting the work of this project. We would like to thank our project and Elettric80 for the opportunity to perform tests at the warehouse in Viano, Italy.

REFERENCES

Baglivo, L., Biasi, N., Biral, F., Bellomo, N., Bertolazzi, E., Lio, M. D., and Cecco, M. D. (2011). Autonomous pallet localization and picking for industrial forklifts: a robust range and look method. *Measurement Science and Technology*, 22(8):085502.

Benenson, R., Omran, M., Hosang, J., and Schiele, B.

(2014). Ten years of pedestrian detection, what have we learned? In *ECCV-CVRSUAD*. IEEE.

Bostelman, R., Hong, T., and Chang, T. (2006). Visualization of pallets. In *SPIE Optics East*.

Bourdev, L. and Brandt, J. (2005). Robust object detection via soft cascade. In *CVPR*, pages II: 236–243.

Byun, S. and Kim, M. (2008). Real-time positioning and orienting of pallets based on monocular vision. In *ICTAI (2)*, pages 505–508. IEEE Computer Society.

Chen, J., Shan, S., He, C., Zhao, G., Pietikäinen, M., Chen, X., and Gao, W. (2010). Wld: A robust local image descriptor. *IEEE Trans. Pattern Anal. Mach. Intell*, 32(9):1705–1720.

Cucchiara, R., Piccardi, M., and Prati, A. (2000). Focus based feature extraction for pallets recognition. In *BMVC*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893.

Dollár, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *PAMI*.

Dollár, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *BMVC*, pages 1–11. British Machine Vision Association.

Dollar, P., Tu, Z. W., Perona, P., and Belongie, S. (2009). Integral channel features. In *BMVC*.

Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Trans. Pattern Anal. Mach. Intell*, 34(4):743–761.

Duda, R. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *CACM*, 15:11–15.

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*, pages II: 807–814.

Hough, P. V. C. (1962). A method and means for recognizing complex patterns. U.S. Patent No. 3,069,654.

Katsoulas, D. and Kosmopoulos, D. I. (2001). An efficient depalletizing system based on 2d range imagery. In *IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA.*, volume 1, pages 305–312. IEEE.

Kim, W., Helmick, D., and Kelly, A. (2001). Model based object pose refinement for terrestrial and space autonomy. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space, Montreal, Quebec, Canada*.

Mikolajczyk, K., Zisserman, A., and Schmid, C. (2003). Shape recognition with edge-based features. In *BMVC*.

Nygårds, J., Högström, T., and Wernersson, Å. (2000). Docking to pallets with feedback from a sheet-of-light range camera. In *IROS*, pages 1853–1859. IEEE.

Oh, J.-Y., Choi, H.-S., Jung, S.-H., Kim, H.-S., and Shin, H.-Y. (2013). An experimental study of pallet recognition system using kinect camera. In *Advanced Science and Technology Letters Vol.42 (Mobile and Wireless 2013)*, pages 167–170.

Ojala, T., Pietikainen, M., and Harwood, D. (1994). Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *ICPR*, pages A:582–585.

Ojala, T., Pietikainen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59.

Pages, J., Armangue, X., Salvi, J., Freixenet, J., and Marti, J. (2011). Computer vision system for autonomous forklift vehicles in industrial environments. *The 9th. Mediterranean Conference on Control and Automation*.

Pradalier, C., Tews, A., and Roberts, J. M. (2008). Vision-based operations of a large industrial vehicle: Autonomous hot metal carrier. *J. Field Robotics*, 25(4-5):243–267.

Prasse, C., Skibinski, S., Weichert, F., Stenzel, J., Müller, H., and Hompel, M. T. (2011). Concept of automated load detection for de-palletizing using depth images and RFID data. *International Conference on Control System, Computing and Engineering (ICCSCE)*, pages 249–254.

Ross, H. and Murray, D. J. (1996). *E.H.Weber on the tactile senses 2nd ed.* Hove: Erlbaum (UK) Taylor and Francis.

Schapire, R. (1990). The strength of weak learnability. *MACHLEARN: Machine Learning*, 5.

Seelinger, M. J. and Yoder, J.-D. (2006). Automatic visual guidance of a forklift engaging a pallet. *Robotics and Autonomous Systems*, 54(12):1026–1038.

Spangenberg, R., Langner, T., Adfeldt, S., and Rojas, R. (2014). Large scale semi-global matching on the CPU. In *Intelligent Vehicles Symposium*, pages 195–201. IEEE.

Varga, R. and Nedevschi, S. (2014). Vision-based automatic load handling for automated guided vehicles. In *Intelligent Computer Communication and Processing*, pages 239–245. IEEE.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proc. CVPR*, 1:511–518.

Viola, P. A., Platt, J. C., and Zhang, C. (2005). Multiple instance boosting for object detection. In *NIPS*.

Walter, M. R., Karaman, S., Frazzoli, E., and Teller, S. J. (2010). Closed-loop pallet manipulation in unstructured environments. In *IROS*, pages 5119–5126. IEEE.

Weichert, F., Skibinski, S., Stenzel, J., Prasse, C., Kamagaew, A., Rudak, B., and ten Hompel, M. (2013). Automated detection of euro pallet loads by interpreting PMD camera depth images. *Logistics Research*, 6(2-3):99–118.

Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Trans. Pattern Anal. Mach. Intell*, 24(1):34–58.

Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *ECCV*, pages B:151–158.

Real-time Pedestrian Detection in Urban Scenarios

VARGA Robert, VESA Andreea Valeria, JEONG Pangyu, NEDEVSCHI Sergiu
Technical University of Cluj Napoca
{robert.varga, pangyu.jeong, sergiu.nedevschi}@cs.utcluj.ro, andreeavaleriavesa@gmail.com
Telephone: (800) 555–1212

***Abstract*—A real-time pedestrian detection system is presented that runs at 24 fps on standard VGA resolution input images (640x480px) using only CPU processing. The detection algorithm uses a variable sized sliding window and intelligent simplifications such as a sparse scale space and fast candidate selection to obtain the desired speed. Details are provided about the initial version of the system ported on a mobile device. We also present a new labeled pedestrian dataset that was captured from a moving car that is suitable for training and testing pedestrian detection methods in urban scenarios.**

***Keywords*—Pedestrian detection; object recognition; region of interest selection; mobile devices**

I. Introduction

As we move closer and closer to a fully autonomous car with collision detection, lane departure warning, adaptive cruise control, autonomous emergency braking, pedestrian detection seems to be the next big wave in offering advanced modern car safety features. Early detection could help reduce the number of accidents by heightening the awareness of the driver in time. In recent years there has been a dramatic increase in the number of people using a smartphone or a tablet as part of their everyday life, confirming the demand for having a fast and accurate detection system integrated with mobile devices. With such an increase and a technology that is constantly evolving, bringing faster and faster processors on the market, the challenge for building a pedestrian detection system on a mobile device proves to be a very good subject open for research. Even though well-established methods running on CPU and specialized hardware exist there is still a long way from fast and accurate detection on portable devices.

II. Related work

This section provides references to important contributions for pedestrian detection. For a comprehensive review on the subject the reader is advised to consult reviews and surveys such as [1], [2], [3].

It is essential for pedestrian detectors to rely on features that help discriminate objects from the background. Histogram of Oriented Gradients[4] is a well established feature for pedestrian detection. Other relevant features are: dense SIFT[5], [6], Histogram of Flow[7], Color self similarity[8], Co-occurrence Histogram of Oriented Gradients[9], Shapelet [10], Local Binary Patterns [11]. Optimizing calculation with integral images has enabled fast calculation of HOG and similar features [12], [13], [14]. Contour based features can also provide useful information [15], [16]. Another crucial ingredient is a fast classifier. In this field boosted ensemble classifiers [17], [18], [19], [20], [21], [22] dominate along with fast SVMs such as linear SVM or Histogram Intersection Kernel SVM [23].

Most top performing modern detection methods rely on integral features and perform detection using sliding windows [1], [20], [24], [25]. Lightning fast methods do exist but they rely on either specialized hardware (GPU[26], [27], [22] or FPGA [28]) or on stereo information (depth) [21], [29]. Another alternative would be a part-based detector [30].

Most of the implementations using ARM architecture are targeted at the embedded market. A couple of such embedded systems for automotive applications that posses an integrated pedestrian detection capability are: Mobileye Pedestrian Collision Warning (PCW) system, Toyota Pedestrian-Avoidance Steer Assist or Volvos Pedestrian Detection With Full Auto Brake system. However, there is no extensive work so far regarding a standalone Android application capable of performing real time pedestrian detection. Our aim is to offer a starting point in building such a solution.

III. Proposed approach

The detection algorithm for the current system follows our previous work from [31]. Here, we only provide a short overview and state the differences. The main focus of this paper is to show how to achieve high-speed detection while maintaining detection performance as much as possible.

One of the key ideas is to detect pedestrians with only a reduced number of heights. Usually detecting different heights entails resizing the image multiple times (constructing the image pyramid). This means that the number of scales is in one-to-one correspondence to the number of pedestrian heights. Our approach is opposed to the mainstream idea of using a dense scale space for image pyramid construction. Each image from the image pyramid is a rescaled version of the input image and has and associated detector of a given height. Typical detectors construct an image pyramid with up to 50-55 scales. The heights chosen here are based on statistical data acquired from the training dataset. We apply k-means clustering to obtain the 6 representative cluster centers for pedestrian bounding box heights. The number 6 was chosen because the detection rate was acceptable. A future analysis about how the detection performance is influenced by the number of heights will be performed. The difference compared to other sparse scale space methods (i.e. methods that limit the number pedestrian heights to consider) is that we do not have detection windows that are of the form $a2^n$, instead we select the representative centers based on the training data. Since detection performance degrades significantly at lower scales we may omit smaller window heights for practical applications.

The proposed detection method uses a sliding window approach with the before mentioned 6 fixed window heights and a constant aspect ratio. Considering an exhaustive search at every position, scale and aspect ratio is not feasible and also not necessary. In this work we opt for a candidate generation that accepts only candidate rectangles that have their center in the horizontal middle stripe. This selection can be motivated by studying the spatial distribution of detection window centers from the training dataset. This distribution has been observed on other datasets such as the Caltech Pedestrians [1].

Each sliding window height corresponds to a pedestrian at a specific scale and has an associated classifier. The classifiers are trained separately for each scale. Our aim is to totally eliminate image resizing and other operations on features. This is a key difference compared to other methods: Dollar et al. adjust the features based on scale in [20], while Benenson et al. [21] adjust the classifier. The detection window is moved to every valid position that is dictated by a region of interest selection method. Features are then calculated as sums of rectangular subregions for each candidate window and classification is performed. The sums of different image channels over a rectangular area can be calculated efficiently with integral images. The underlying image channels are Luv color-space, gradient magnitude and oriented gradient histogram with 6 bins. For feature extraction we rely on a module provided by Dollar [32].

Key elements of this approach that help achieve high-speed and reliable detection are:

- No image resizing
- Smart and fast region of interest selection (candidate generation)
- Fast integral channel features calculation
- A cascade of boosted decision trees for classification
- Reduced number of pedestrian heights
- Custom implementation of all processing modules and code parallelization

IV. Implementation details

The reduced execution time of the algorithm is due to the fact that almost all processing steps have been specifically rewritten for the detection task. Our implementation is in C++, compiled with Visual Studio 2010 compiler with OpenMP multithreading features enabled. Other settings include: fast code optimization enabled, fast floating point model, omit frame pointers. OpenCV 2.4.5 is the chosen library for image processing functions.

The workstation used to test our system has the following parameters: Intel Core i7 CPU, 3.5 GHz, 4 cores, 8 logical processors, 16 GB RAM. Most of the relevant operations are parallelized to use the processing power of the CPU efficiently. Speed measurements are provided in the Experimental Results section VI.

A. Detection algorithm

The detection algorithm follows the well established pipeline format having the following steps: preprocessing (resizing - not employed in this case, padding - when the image width is not divisible by 4, smoothing), region of interest selection (or candidate generation), feature channels extraction (Luv conversion, gradient computation, histogram bin aggregation), feature aggregation (rectangular region sums) only on the candidate regions, classification/prediction and non-maximum suppression.

An important aspect of the preprocessing step is how to treat images that do not have width divisible by 4. Since many operations run only if divisibility is ensured we pad the images instead of cropping or resizing. This also helps to linearize the image in the memory. Note, that during the training phase many cropped small images are fed as input to the feature extractor. These result from clipping out only the pedestrian bounding box. This is why it is important to treat irregular sized images carefully. We perform no image resizing, although for larger input images this could be included to reduce the search space. Image smoothing is moved to the feature extraction phase and gaussian bluring is replaced by a faster triangular filtering as in [13].

Region of interest selection provides the candidate rectangles from which the features are extracted and classified. It is essential to restrict the number of these candidates to reduce the workload of the following modules from the pipeline. For this step we have three main options. The first option is to use all possible bounding boxes with a given stride, fixed aspect ratio and height restricted to a set of values. The second option is to admit only the rectangles whose centers lie in the central horizontal stripe of the image. This is a heuristic that is easy to implement and it is deduced from the measurements from the pedestrian dataset (see section V). The third option is to select candidate regions based on the edges in the image (as in [31]). Here we opt for the second approach because it is sufficient and assures high coverage. The selected pedestrian heights are: 60, 92, 136, 212, 340 pixels. The stride is set to 4 or 8 pixels, the fixed aspect ratio is 0.43 (width over height).

For feature extraction we rely on the Integral Channel Features module provided by Dollar [32]. This was adapted from Matlab+mex to our C++ implementation that uses classes from the OpenCV 2.4.5 library. Feature extraction is fast because of clever usage of integral images, parallel computing of the channels and SSE instructions. The present module has the standard configuration of channels: luv, gradient magnitude and 6 gradient orientation bins. Parameters have been set to: 5000 random features with an area of at least 25; shrinking factor of 4, triangular smoothing which is equivalent to a Gaussian blur with a sigma of 1. See [13] for more details about the parameters.

Classification is performed with an ensemble of 5000 weak learners. Both the training and predicting have been reimplemented to optimize prediction speed. Discrete boosting is applied with two level decision trees. This option is motivated by [20], where the authors show that the boosting method does not have a large impact on the detection rate and that 2 level decision trees are the best for this task. The splitting criterion for the decision tree is the local training error, i.e. the best split is the one that minimizes the training error. Rejection thresholds for the cascade classifier are obtained via the direct backward pruning method [33] on the training set. In most cases it is preferred to obtain the thresholds on a validation set rather than the training set. When this validation is not performed the rejection thresholds should be lowered in order to prevent the quick rejection of unseen positive examples. A simple recalculation of the thresholds can be performed in order to obtain rejection thresholds for any end threshold (see [33]).

At detection time all rectangles obtaining a classification score higher than a given threshold θ) are retained for non-maximum suppression. For every two overlapping rectangles we retain only the one with the higher score. The overlap can be determined in multiple ways. Here, we use the formula: $o_{min} = \frac{R_1 \cap R_2}{min(R_1, R_2)}$, where R_1 and R_2 are the areas of the two rectangles, and the numerator contains the area of their intersection. This eliminates smaller bounding boxes from inside larger ones because in this case the overlap is high due to the min function from the numerator. For the same reason it more aggressive than the usual alternative: the PASCAL VOC-type overlap measure $o = \frac{R_1 \cap R_2}{R_1 \cup R_2}$. This is why lower thresholds are suitable for this method. Another alternative useful for large number of detection windows is to perform a greedy elimination. The threshold for considering overlap is set to 0.4.

B. Training procedure

At the training phase we repeat the same operations at each scale to generate classifier models. We first process the positive examples. Each bounding box of a person is cropped from the original image and resized with bilinear interpolation to the size of the current detection window (e.g. 60x26px). Adjustments are mare to center to the bounding box and to preserve the original aspect ratio (resizing factor is the same along the width and the height). To increase the diversity we also process the horizontally mirrored image. The 5000 features are calculated and saved for later with a positive label. These features are randomly generated rectangles from inside the detection window area. To obtain negative samples we select 5000-7000 random crops from the list of images not containing any pedestrians. A uniform sampling is performed, i.e. if there are 5000 images, then one random rectangle is chosen from each, if there are 20000 images (a typical case), then one random rectangle is chosen from every 4th. For every random window we calculate the features and save them for later with a negative label. Once all examples are processed the file containing the features can be fed as input to train the classifier.

Next, we perform bootstrapping. Call the initial model as model-x-0, where x stands for the height and 0 stands for no bootstrapping. By applying the classifier on negative images we obtain at first many false positives. At each stage we retain 7000 of these false positives and append their feature vectors to the training file. After retraining the classifier model-x-1 is obtained. This process is repeated 2-3 times until there are only a few false positives or no change is observed. Limiting the number of examples is essential to keep the training set balanced and also helps reduce redundancy. We have observed that the classifier for the smallest scale produces false positives even after 4 rounds of bootstrapping. This is a clear signal indicating the failure of the classifier to learn a good model from the training data.

TABLE I: Relevant parameters of the detection algorithm

Parameter	Description	Value
N	number of weak learners	1000
M	number of features	5000
d	stride (grid step)	8
ρ	aspect ratio	0.43
h	pedestrian heights	{60, 92, 136, 212, 340}
T_o	overlap threshold for NMS	0.4
B	additional bootstrap samples at each stage	7000

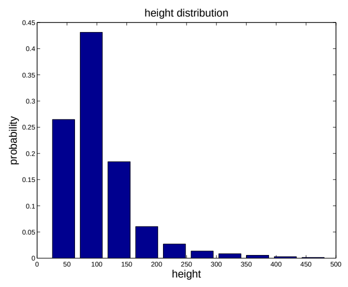
C. Implementation for Android mobile devices

In this section we present some details about how the presented approach can be implemented on a mobile device such as a tabled or smartphone. Having the algorithm already implemented in C++, developing an Android application seemed a very good solution for us since we could easily integrate the existing native code with Java code by making use of Java Native Interface (JNI) framework. The algorithm was ported on an Android mobile device having the following characteristics: NVIDIA Tegra 3 T30L chipset, Quad-core 1.2 GHz ARM Cortex-A9 CPU, NEON instruction set support. We have used JNI calls in order to capture the frames in a Java environment and send them for a faster processing in a native C++ environment. For a better performance we have used OpenCV 2.4.5 for Tegra 3 library accelerated for ARM NEON architectures and we took benefit from the multi-core processor by parallelizing the code with Qualcomm MARE Parallel Computing Library. We took advantage of the **pfor_each** functionality provided by MARE in order to substitute the **#pragma omp** parallel functionality provided by OpenMP used in our PC version. For handling the reading and writing of the training files we have used the AssetManager class provided by Java. This allows us to compress the files and perform one time writing in the internal memory of the portable device upon installation of the application and extract the data whenever we need during running the algorithm.

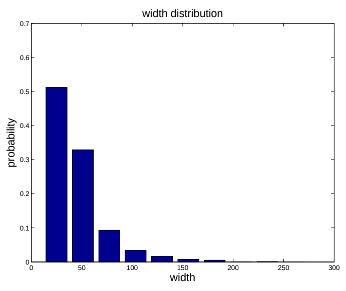
V. Cluj pedestrians dataset

A labeled pedestrian dataset was gathered for the purpose of training and evaluation (see Figure 3). This set was captured using a smartphone placed on the windshield of a vehicle driving through the city. The purpose of this new dataset is to reproduce as closely as possible the real situation where the detection method would be applied. The dataset is available in both video (mp4) and image (png) format. Total video length is around 15 minutes. The video framerate is of 30fps. All images have 640x480px resolution (landscape). The total number of images is 27666. The set is broken into independent sequences based on separate recordings and contains mostly frames with pedestrians. We provide the pedestrian bounding boxes for each image in the dataset in a simple text file format. Note, that some pedestrians are unlabeled. This is the case only for very crowded scenes and for persons that are far away and thus appear to be very small. There can be up to 10 pedestrians in a single frame.

The pedestrian bounding boxes are divided into a non-overlapping training set and test set. The initial training set

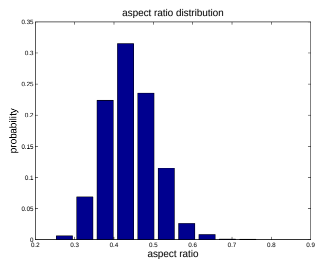


(a) height distribution

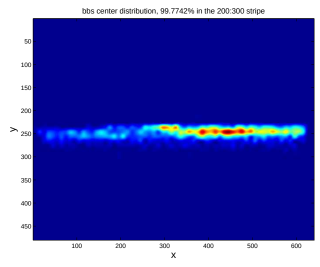


(b) width distribution

Fig. 1: Bounding box dimension statistics for the Cluj-pedestrians dataset



(a) aspectratios



(b) width distribution

Fig. 2: Bounding box aspect ratio and center frequency for Cluj Pedestrians

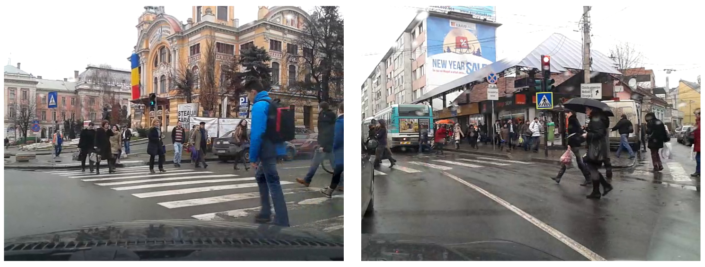


Fig. 3: Sample images from the Cluj Pedestrians dataset

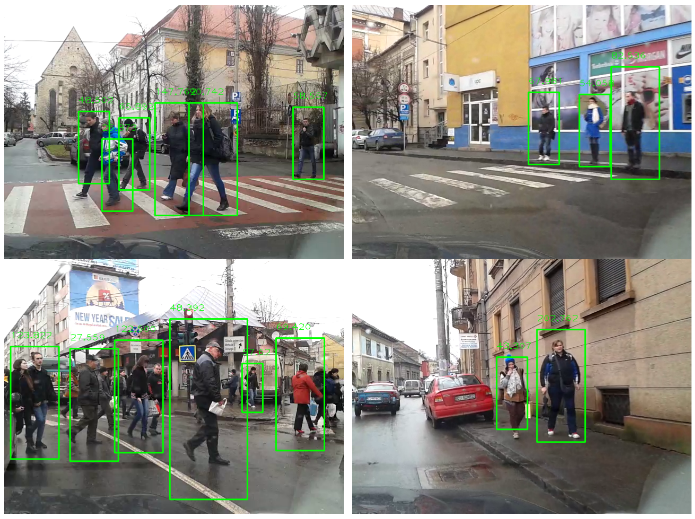


Fig. 4: Sample detections from Cluj Pedestrians dataset

contains 3205 bounding boxes with pedestrians and 219 negative examples (images with no persons) obtained by labeling every 10th frame. In order to enrich the dataset we have interpolated the detections and have generated an interpolated training set consisting of 27318 pedestrian rectangles. Due to the reduced number of negative examples it is recommended to augment the negative training set (images not containing any pedestrians) from another source (e.g. INRIA, Caltech or general images). For our training procedures we have augmented the examples from this dataset with a portion of the images (around 4500) from the Caltech Pedestrian Dataset [34] that do not contain any pedestrians.

We provide some interesting and relevant statistics for the dataset. The distribution of the heights of the bounding boxes is given in Figure 1. Approximately 88% of the bounding boxes are smaller then 140px in height, while the predominant height is 90px. The width of the bounding boxes follows an exponential distribution shown in Figure 1. The most likely aspect ratio is 0.43, see Figure 2. An important statistic about the frequency of the centers of the bounding boxes reveals that only a negligible part (0.3%) of them are positioned outside the 200-300 horizontal stripe (see Figure 2). Also, due to the placement of the recording device and because the pavement is on the right side, pedestrians occur more frequently on the right side.

VI. Experimental results

To evaluate our detection method we perform tests on two datasets: the well established INRIA benchmark and the newly acquired Cluj Pedestrians dataset. We directly compare our method to FPDW [1] [32] because it is based on it and also the source code and the detection module is available. The evaluation entails generating predictions with a low accepting threshold and generating the DET curve (detection error trade-off) by accepting detections at different thresholds. A predicted bounding box is correct if it overlaps with a ground truth box by at least 20%. The reason for the lower overlap threshold is to take into consideration the possible scale differences. The overlap is the usual Pascal VOC fraction: $\frac{R_1 \cap R_2}{R_1 \cup R_2}$, where the numerator is the intersection and the denominator is the union of the two rectangles in question. A point on the DET curve represents a fixed threshold for classification and corresponds to a miss rate and a false positives per image. This operation point can be chosen depending on the application and its requirements. In order to directly compare methods the area under the DET curve up to the one false positives per image line is calculated. A lower area means better detection performance.

First, we report on the results on the INRIA dataset. For this dataset we select the following 6 bounding box heights: {124, 204, 288, 388, 516, 664}, which were estimated as cluster

[1] http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html

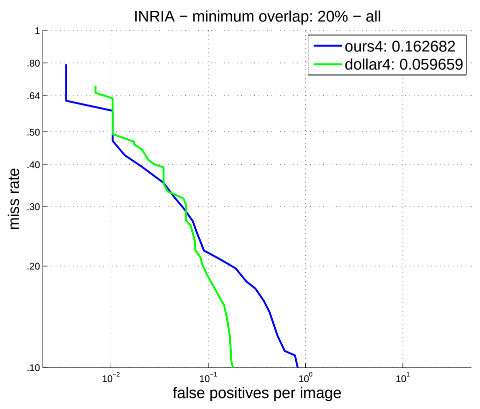


Fig. 5: DET curve on the INRIA Pedestrian dataset

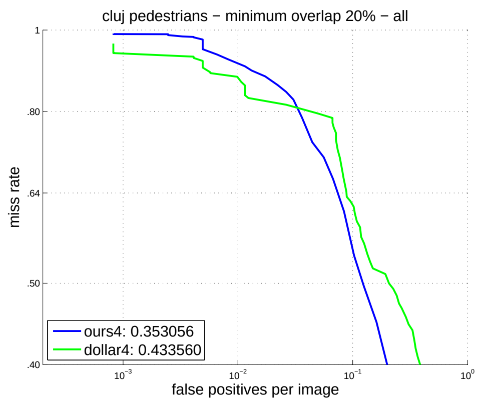


Fig. 6: DET curve on the Cluj Pedestrians dataset

centers from the training data. The DET curve is presented in Figure 5, where we compare our method with FPDW. Even though the behavior for our algorithm is similar up the 0.1 fppi line the overall performance of the FPDW is better - 0.059 compared to 0.16 area under the DET curve. This can be motivated by the fact that we use only a limited number of scales to perform detections and other simplifications.

We also evaluate both methods on the Cluj Pedestrians dataset (see Figure 4 for sample detections). For this dataset we select the following 5 bounding box heights: $\{60, 92, 136, 212, 340\}$. Note that the pedestrian heights for the INRIA dataset have larger values compared to those for the Cluj Pedestrians dataset. This dataset is much harder because it contains crowded scenes and small pedestrians. The bounding boxes for FPDW were generated using the code provided by the authors. The same evaluation code was used as in the case of the INRIA dataset. The DET curve is presented in Figure 6. In this case our method performs better at almost every point of the DET curve and in overall - 0.35 in comparison with 0.43 area under the DET curve. This can be explained by a more similar training set to the test set for our method. We have performed a grid search for optimal parameters. The parameters that can be changed after the classifier models are generated are: the stride (spatial grid step size); the overlap threshold for non-maximum suppression; the number of pedestrian heights to consider. We provide some representative results in Table II. Minimal areas under the DET curve are obtained at stride 4 and at overlap threshold of 0.4 (the criterion of overlap is o_{min}). The number of bootstrap rounds is set to 4, but for larger scales the process converges at rounds 2-3 and the classifier does not generate any false positives on the training set.

To evaluate the execution time of each part of the system we measure at least 1000 times the speed of each module. Table III summarizes the measurements that we have obtained. The bottleneck of the pipeline is the feature extraction module where heavy optimization has been performed, even so it is the slowest part. A further reduction of the number of the candidates could help drastically improve the speed. The number of candidates is around 2000-3000 depending on the scales. The estimated execution time (disregarding the visualization part) is 23.67 frames per second (around 42 milliseconds per frame). We compare the speed gain with our previous version

TABLE II: Area under the DET curve for different parameters on the Cluj Pedestrians dataset

area	**stride = 8**	**stride = 4**	**stride = 3**
overlap = 0.40	0.361	0.353	0.367
overlap = 0.65	0.373	0.367	0.375
overlap = 0.70	0.384	0.372	0.379

TABLE III: Execution time of different modules

Step	**PC [ms]**	**Android [ms]**	**Android + MARE [ms]**
RoI selection	0.131	0.29	0.29
Feature Extraction	38.3	1491.29	739.53
Classification	3.38	230.60	78.73
NMS	0.096	0.13	0.13

from [31]: 180 milliseconds on an image with a resolution of 370x480px would result in approximately 311ms running time on a 640x480px image, thus the speed gain is: $\frac{311}{42} = 7.4$. Time measurements on Android for a 640x480px resolution indicate a 1.21 FPS average frame rate (around 823 milliseconds per frame). This is the parallelized Android version which has an overall speed gain compared to the single threaded version of: $\frac{1781.89}{823.0} = 2.16$ ($\frac{1491.29}{739.53} = 2.01$ - feature extraction; $\frac{230.60}{78.73} = 2.92$ - classification).

VII. Conclusion

In this paper we have presented a real-time PC implementation of a pedestrian detection algorithm able to run on Android mobile devices. The key idea is to use the sliding window detection approach on a restricted zone of the image. We employ only 4-6 scales for detection which reduces the execution time drastically. Even with this sparse scale space the detections are acceptable. The scales are chosen based on statistical information gathered from the training set. For each scale we train a separate classifier. By evaluating our method on two datasets we have shown that the method is competitive.

Future work will focus on generating candidate regions based on methods that rely on segmentation (such as Objectness and Selective Search). Although there are many existing approaches with high recall none of them can provide object locations within milliseconds which is a necessity for both fast

PC and Android applications. We will also study the effect of a sparse scale space on the detection accuracy and how many scales are really necessary for detection. Further optimizations will be performed on the Android version especially in the feature extraction part.

Acknowledgment

This research was funded by the CoMoSeF project, code PN II PCCA PN-II-PT-PCCA-2011-3.2-0742 from 20.09.2012 (2012-2016).

References

[1] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 34, no. 4, pp. 743–761, 2012.

[2] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, Dec. 2009.

[3] D. Gerónimo, A. M. López, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 7, pp. 1239–1258, 2010.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, pp. I: 886–893.

[5] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, 1999, pp. 1150–1157.

[6] B. Fulkerson, A. Vedaldi, and S. Soatto, "Localizing objects with smart dictionaries," in *ECCV*, 2008, pp. I: 179–192.

[7] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *ECCV*, 2006, pp. II: 428–441.

[8] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," in *CVPR*. IEEE, 2010, pp. 1030–1037.

[9] T. Watanabe, S. Ito, and K. Yokoi, "Co-occurrence histograms of oriented gradients for pedestrian detection," in *PSIVT*, 2009, pp. 37–47.

[10] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in *CVPR*, 2007, pp. 1–8.

[11] Y. Cao, S. Pranata, and H. Nishimura, "Local binary pattern features for pedestrian detection at night/dark environment," in *ICIP*, B. Macq and P. Schelkens, Eds. IEEE, 2011, pp. 2053–2056.

[12] Q. A. Zhu, M. C. Yeh, K. T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *CVPR*, 2006, pp. II: 1491–1498.

[13] P. Dollar, Z. W. Tu, P. Perona, and S. Belongie, "Integral channel features," in *BMVC*, 2009.

[14] F. M. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," in *CVPR*, 2005, pp. I: 829–836.

[15] D. M. Gavrila and S. Munder, "Multi-cue pedestrian detection and tracking from a moving vehicle," *International Journal of Computer Vision*, vol. 73, no. 1, pp. 41–59, Jun. 2007.

[16] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *CVPR*. IEEE, 2013, pp. 3158–3165.

[17] J. Friedman, T. Hastie, and R. Tibshirani, "Special invited paper. additive logistic regression: A statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–374, 2000.

[18] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *CVPR*, 2005, pp. II: 236–243.

[19] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, Jul. 2005.

[20] P. Dollár, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, F. Labrosse, R. Zwiggelaar, Y. Liu, and B. Tiddeman, Eds. British Machine Vision Association, 2010, pp. 1–11.

[21] R. Benenson, M. Mathias, R. Timofte, and L. J. V. Gool, "Pedestrian detection at 100 frames per second," in *CVPR*. IEEE, 2012, pp. 2903–2910.

[22] R. Benenson, M. Mathias, T. Tuytelaars, and L. J. V. Gool, "Seeking the strongest rigid detector," in *CVPR*. IEEE, 2013, pp. 3666–3673.

[23] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *CVPR*, 2008, pp. 1–8.

[24] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *ECCV*, 2012.

[25] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *PAMI*, 2014.

[26] T. Machida and T. Naito, "GPU & CPU cooperative accelerated pedestrian and vehicle detection," in *ICCV Workshops*. IEEE, 2011, pp. 506–513.

[27] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele, "Sliding-windows for rapid object class localization: A parallel technique," in *Pattern Recognition (DAGM)*, May 2008.

[28] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, "FPGA-based real-time pedestrian detection on high-resolution images," in *CVPR Workshops*. IEEE, 2013, pp. 629–635.

[29] A. Ess, K. Schindler, B. Leibe, and L. van Gool, "Robust multi-person tracking from moving platforms," in *Logic and Probability for Scene Interpretation*, ser. Dagstuhl Seminar Proceedings, A. G. Cohn, D. C. Hogg, R. Möller, and B. Neumann, Eds., no. 08091. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.

[30] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 9, pp. 1627–1645, 2010.

[31] R. Varga and S. Nedevschi, "Gradient-based region of interest selection for faster pedestrian detection," in *Intelligent Computer Communication and Processing*. IEEE, 2013, pp. 1–2.

[32] P. Dollár, "Piotr's Image and Video Matlab Toolbox (PMT)," http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.

[33] P. A. Viola, J. C. Platt, and C. Zhang, "Multiple instance boosting for object detection," in *NIPS*, 2005.

[34] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *CVPR*, 2009, pp. 304–311.