Third Croatian Computer Vision Workshop

September 16, 2014, Zagreb, Croatia

CCVW 2014

University of Zagreb
Center of Excellence
for Computer Vision

PROCEEDINGS OF

Proceedings of the Croatian Computer Vision Workshop CCVW 2014, Year 2 ISSN 1849-1227
Publisher: University of Zagreb Faculty of Electrical Engineering and Computing, Unska 3, HR-10000 Zagreb, Croatia
http://www.fer.unizg.hr/crv/proceedings

CCVW 2014

Proceedings of the Croatian Computer Vision Workshop

Zagreb, Croatia, September 16, 2014

S. Lončarić, M. Subašić (Eds.)

Organizing Institution

Center of Excellence for Computer Vision,
Faculty of Electrical Engineering and Computing,
University of Zagreb, Croatia

Technical Co-Sponsors

IEEE Croatia Section
IEEE Croatia Section Computer Society Chapter
IEEE Croatia Section Computational Intelligence Chapter
IEEE Croatia Section Signal Processing Society Chapter

Donators

PhotoPay Ltd.
Visor Ltd.

Proceedings of the Croatian Computer Vision Workshop
CCVW 2014, Year 2

Editor-in-chief

Sven Lončarić (sven.loncaric@fer.hr)
University of Zagreb Faculty of Electrical Engineering and Computing
Unska 3, HR-10000, Croatia

Editor

Marko Subašić (marko.subasic@fer.hr)
University of Zagreb Faculty of Electrical Engineering and Computing
Unska 3, HR-10000, Croatia

Production, Publishing and Cover Design

Tomislav Petković (tomislav.petkovic.jr@fer.hr)
University of Zagreb Faculty of Electrical Engineering and Computing
Unska 3, HR-10000, Croatia

Publisher

University of Zagreb Faculty of Electrical Engineering and Computing
Unska 3, HR-10000 Zagreb, OIB: 57029260362

ISSN 1849-1227

Preface

On behalf of the Organizing Committee it is my pleasure to invite you to Zagreb for the 3rd Croatian Computer Vision Workshop. The objective of the Workshop is to bring together professionals from academia and industry in the area of computer vision theory and applications in order to foster research and encourage academia-industry collaboration in this dynamic field. The Workshop program includes oral and poster presentations of original peer reviewed research from Croatia and elsewhere. Furthermore, the program includes invited lectures by distinguished international researchers presenting state-of-the-art in computer vision research. Workshop sponsors will provide perspective on needs and activities of the industry. Finally, one session shall be devoted to short presentations of activities at Croatian research laboratories.

The Workshop is organized by the Center of Excellence for Computer Vision, which is located at the Faculty of Electrical Engineering and Computing (FER), University of Zagreb. The Center joins eight research laboratories at FER and research laboratories from six constituent units of the University of Zagreb: Faculty of Forestry, Faculty of Geodesy, Faculty of Graphic Arts, Faculty of Kinesiology, Faculty of Mechanical Engineering and Naval Architecture, and Faculty of Transport and Traffic Sciences.

Zagreb is a beautiful European city with many cultural and historical attractions, which I am sure all participants will enjoy. I look forward to meet you all in Zagreb for the 3rd Croatian Computer Vision Workshop.

September 2014

Sven Lončarić, General Chair

Acknowledgements

The 2014 3rd Croatian Computer Vision Workshop (CCVW) is the result of the committed efforts of many volunteers.

All included papers are results of dedicated research. Without such contribution and commitment this Workshop would not have been possible.

Program Committee members and reviewers have spent many hours reviewing submitted papers and providing extensive reviews which will be an invaluable help in future work of collaborating authors. Managing the electronic submissions of the papers, the preparation of the abstract booklet and of the online proceedings also required substantial effort and dedication that must be acknowledged. The Local Organizing Committee members did an excellent job to guarantee a successful outcome of the Workshop.

We are grateful to the Technical Co-Sponsors, who helped us in granting the high scientific quality of the presentations, and to the Donators that financially supported this Workshop.

Contents

Organizing Committee

General Chair

Sven Lončarić, University of Zagreb, Croatia

Technical Program Committe Chair

Marko Subašić, University of Zagreb, Croatia

Local Arrangements Chair

Tomislav Petković, University of Zagreb, Croatia

Publications Chair

Pavle Prentašić, University of Zagreb, Croatia

Technical Program Committee

Bart Bijnens, Spain
Hrvoje Bogunović, USA
Mirjana Bonković, Croatia
Karla Brkić, Croatia
Robert Cupec, Croatia
Albert Diosi, Slovakia
Hrvoje Dujmić, Croatia
Ivan Fratrić, Switzerland
Hrvoje Gold, Croatia
Mislav Grgić, Croatia
Sonja Grgić, Croatia
Andras Hajdu, Hungary
Edouard Ivanjko, Croatia
Bojan Jerbić, Croatia
Zoran Kalafatić, Croatia
Stanislav Kovačič, Slovenia
Zoltan Kato, Hungary
Josip Krapac, Croatia
Sven Lončarić, Croatia
Lidija Mandić, Croatia
Vladan Papić, Croatia
Renata Pernar, Croatia
Tomislav Petković, Croatia
Ivan Petrović, Croatia
Thomas Pock, Austria
Tomislav Pribanić, Croatia
Arnau Ramisa, Spain
Slobodan Ribarić, Croatia
Damir Seršić, Croatia
Darko Stipaničev, Croatia
Federico Sukno, Ireland
Siniša Šegvić, Croatia

Reviewers

Albert Diosi, Austraila
Arnau Ramisa, Spain
Edouard Ivanjko, Croatia
Federico Sukno, Argentina
Hrvoje Bogunović, USA
Karla Brkić, Croatia
Marko Subašić, Croatia
Robert Cupec, Croatia
Siniša Šegvić, Croatia
Stanislav Kovačič, Slovenia
Thomas Pock, Austria
Tomislav Petković, Croatia
Vladan Papić, Croatia
Zoltan Kato, Hungary
Zoran Kalafatić, Croatia

Experimental Evaluation of Vehicle Detection Based on Background Modelling in Daytime and Night-Time Video

Igor Lipovac, Tomislav Hrkać, Karla Brkić, Zoran Kalafatić and Siniša Šegvić
University of Zagreb
Faculty of Electrical Engineering and Computing
Email: igor.lipovac@infinum.hr, tomislav.hrkac@fer.hr, karla.brkic@fer.hr, zoran.kalafatic@fer.hr, sinisa.segvic@fer.hr

***Abstract*—Vision-based detection of vehicles at urban intersections is an interesting alternative to commonly applied hardware solutions such as inductive loops. The standard approach to that problem is based on a background model consisting of independent per-pixel Gaussian mixtures. However, there are several notable shortcomings of that approach, including large computational complexity, blending of stopped vehicles with background and sensitivity to changes in image acquisition parameters (gain, exposure). We address these problems by proposing the following three improvements: (i) dispersed and delayed background modelling, (ii) modelling patch gradient distributions instead of absolute values of individual pixels, and (iii) significant speed-up through use of integral images. We present a detailed performance comparison on a realistic dataset with handcrafted groundtruth information. The obtained results indicate that significant gains with respect to the standard approach can be obtained both in performance and computational speed. Experiments suggest that the proposed combined technique would enable robust real-time performance on a low-cost embedded computer.**

I. Introduction

In this paper we consider vehicle detection at urban intersections. Traditionally, this problem has been solved by inductive loop sensors capable of detecting the presence of a vehicle. However, applying inductive loops for vehicle detection is expensive, primarily due to the need to conduct construction works and to stop the traffic during the installation as well as for maintenance. Therefore, the traffic management companies search for alternative solutions which would enable easy sensor replacement and maintenance. Computer vision techniques are very suitable for this task.

The usual computer vision scenario involves a fixed-view camera above the road and suitable algorithms to detect moving objects of appropriate size. This leads us to the well known problem of background modelling for which numerous solutions have been proposed. For the application scenario which involves day and night video capture, it is necessary to have an adaptive background model. Another constraint that should be taken into account is that the target system should be suitable for mounting at road infrastructure elements. Therefore, it would be beneficial to develop algorithms that could run on embedded hardware, which would also significantly reduce the installation costs.

We first considered the baseline background modelling approach based on per-pixel Gaussian mixtures, and evaluated it in the typical urban intersection scenario. This preliminary evaluation identified several important problems.

1) during the red light phase, the vehicles stop for a relatively long period and due to the background adaptation tend to influence the background model;
2) the classical running average model with exponential learning curve [12] tends to overemphasize the influence of the waiting vehicles;
3) automatic camera adaptation causes significant changes of the image in some situations, leading to miss-detections;
4) the detection of the vehicles of the color similar to the color of the road is often unsuccessful.

In order to address these problems we evaluate several improvements to the baseline background modelling approach. Firstly, we delay the model used for object detection in order to reduce the influence of waiting cars to their own detection. The idea here is to use the background model built before the arrival of the stopped cars and thus avoid using an infected model. Secondly, we attempt to reduce the influence of waiting cars by introducing a more appropriate weighting of the incoming frames through a two-stage background model. Thirdly, we attempt to reduce the dependence on absolute pixel values by building gradient-based background models. In order to improve the resistance to noise and at the same time reduce computational complexity, we refrain from considering individual pixels and instead model the gradient distribution above an overlapping set of small rectangular image patches.

II. Related work

Computer vision-based approaches to the estimation of traffic flow parameters have been the subject of a lot of recent research. A common approach to separate foreground objects from the background scenery is based on background modelling. In such approaches, a statistical model that describes the background state of each pixel is constructed and subsequently compared to the current video frame. Pixels in which the difference is significant are considered to belong to foreground objects.

A number of methods for background model construction has been proposed. Especially popular have been time-adaptive Gaussian mixture models [14], [12], [15]. In these methods,

each pixel is represented with a set of weighted Gaussian distributions. Based on the assumption that background is visible most of the time in each pixel position, distributions are ordered according to their weights, and those more relevant are considered to model the background, while the remaining model the foreground. The per-pixel models are updated with each new observation, with older observations losing influence over time.

A comparison of several different background subtraction algorithms for detecting moving vehicles and pedestrians in urban traffic video sequences is given in [4]. The tested algorithms are classified as non-recursive (including simple frame differencing, median filtering, linear predictive filtering and non-parametric estimate of the pixel density function) or recursive (approximated median filter, Kalman filter and mixture of Gaussians). The evaluation is performed on four different video sequences with manually annotated moving objects in ten frames in each sequence as a ground truth. The algorithms are then evaluated by measuring precision and recall for different algorithm parameters. Mixture of Gaussians produces the best results, but median filtering offers a simple alternative with competitive performance. Frame differencing produces significantly worse results than all the other schemes.

Herrero and Bescós [9] provide another detailed overview and an evaluation of commonly used background subtraction techniques. The approaches covered by the overview are divided into simple (frame differencing, running average, median filtering), unimodal (Gaussian or chi-square modelling) and multimodal (mixtures of Gaussians, mean-shift algorithm, kernel density estimation and hidden Markov models). Evaluation is performed on video sequences from the dataset introduced in [13], obtained by combining separately recorded foreground and background videos, so the segmentation masks are known. The evaluation results suggest that chi-square modelling performs best in most scenarios. However, the authors note that mixtures of Gaussians and simple median filtering performed especially well in cases of highly dynamic backgrounds. Overall, the experimental findings in the evaluation supports the notion that relatively good results can be obtained with very simple techniques.

A more recent evaluation of bacground subtraction techniques with an emphasis on video surveillance is given by Brutzer et al. [3]. Nine background subtraction methods are compared at the pixel level. To alleviate the problem of ground truth collection, the authors rendered complex artificial scenes that address several challenges in background subtraction: gradual and sudden illumination changes, dynamic background, objects similar to background, shadows, initialization with foreground objects present and noise. The top-performing method is ViBe, a method proposed by Barnich and Van Droogenbroeck [1]. ViBe introduces several interesting innovations, e.g. storing a history of actual pixel values for a given pixel instead of building a statistical model, having a random update policy, doing background initialization from a single frame by assuming that neighboring pixels share a similar temporal distribution, etc.

Most background techniques assume a single rate of adaptation that determines how adaptive the model is to the change in pixel value. However, this can be inadequate in scenes such as traffic intersections, where objects move at a variety of speeds. A fast-adapting algorithm can miss detection of parts of homogeneous moving objects, since they quickly become part of the background. On the other hand, slow adapting algorithm leave long trails ("ghosts") behind initially stationary objects that suddenly start to move, such as cars waiting at the crossroad. Algorithms with slow adaptation rate are also more sensitive to sudden global illumination changes. To cope with this, Cheung and Kamath [5] propose a dual-stage algorithm that first builds a foreground mask using a slow-adapting Kalman filter, and then validates individual foreground pixels by a simple moving object model, built using foreground and background statistics as well as the frame difference.

Another approach was suggested by Harville [8]. He proposed a framework for guiding evolution of pixel-level mixture of Gaussians models by using feedback from higher-level modules, such as module for person detection and tracking, or module for detection of rapid changes in global illumination, camera gain or camera position. The feedback of each module can be classified either as positive, which serves to enhance correct foreground segmentation, or as negative, which aims to adjust the pixel-level background model in order to prevent the re-occurrence of detected foreground mistakes.

To improve the robustness of vehicle detection against illumination changes and small camera movements, as well as the ability to track vehicles in case of occlusions and crowded events, Batista et al. [2] propose a dual-stage approach consisting of pixel-level and block-level stages. The pixel-level stage uses a multi-layered and adaptive background modelling, based on three image models. Two of them are used to model the dynamics of the background allowing the system to cope with intensity variations, while the third is used in the cleaning/validation process, being a direct copy of the past image. The block-level stage performs a 8x8 block-region analysis to label the blocks belonging to different vehicles and track them over a stack of images.

As a part of the University of South California Clever Transportation Project, Kim et al. [10] propose a system for real-time traffic flow analysis. The system aims to replace traffic loop detectors with cameras utilizing computer vision techniques. A special coprocessor, the Viewmont video analytics coprocessor, has been provided by Intel, who is a partner on the project. The coprocessor is specifically tailored toward video processing, enabling significant speed-up when compared to a conventional CPU. At the time of writing there is no information about the coprocessor on Intel's webpage, and it does not seem to be commercially available. In order to use the system, one needs to define a region of interest where the traffic is most visible, and within it a series of virtual lines spanning across individual lanes. Background is subtracted using frame averaging, and moving objects are extracted. Morphological operations are applied to obtain crisp

boundaries of the moving objects and remove noise. Passing of the vehicles is detected by counting the relative proportion of pixels belonging to moving objects crossing a virtual line to the total number of pixels comprising the line. In the evaluation, the results obtained by the system are compared to the output from real loop detectors. The main two identified problems are dense traffic and vehicle shadows.

III. The standard approach and its shortcomings

All background modelling approaches assume that each particular image pixel in most video frames is projected from the background scenery. Thus, a fair estimate of the actual background should be obtainable by some kind of an average pixel value across many frames. By comparing the actual value with the estimated model a pixel can finally be classified into either the background or the foreground class.

Pixel averaging is usually achieved by fitting a Gaussian mixture model with few components (n=2 or n=3) to each individual pixel over a number of frames. Multiple components are useful since they can account for small camera motions due to vibrations. The recovered variances allow to perform the classification by taking into account the actual camera noise.

In order to avoid the need for storing a large number of video frames, the standard background modelling approach estimates the required per-pixel Gaussian mixtures by employing the exponential moving average. In this approach, the evolution of the single component model (μ, σ) at the pixel (x, y) of the current image I_k can be described with the following equations (note that the free parameter α regulates the adaptivity of the model).

$$\mu_k[x,y] = \alpha I_k[x,y] + (1-\alpha)\mu_{k-1}[x,y] \;, \tag{1}$$

$$\sigma_k^2[x,y] = \alpha(I_k[x,y] - \mu_k[x,y])^2 + (1-\alpha)\sigma_{k-1}^2[x,y] \;. \tag{2}$$

These equations are easily extended to the multi-component case by weighting the contribution of the current pixel with the distances from the component centers [15]. The model needs to be initialized on a suitable video sequence either by straight-forward Gaussian fitting (one component) or by the EM algorithm (multiple components).

Unfortunately, it is very difficult to find the parameter α of the standard approach (2) which achieves an optimal ballance between robustness to stopped objects and adaptivity to daily illumination changes. If α is too large, stopped cars start disturbing the model. If α is too small, the model will not be adaptive enough to follow illumination changes due to meteorological conditions. This could be improved by storing a history of values for each given pixel [1] and calculating the correct running average:

$$\mu_k[x,y] = \sum_{i=k-N}^{k-1} I_i[x,y] \;, \tag{3}$$

$$\sigma_k^2[x,y] = \sum_{i=k-N}^{k-1} (I_i[x,y] - \mu_k[x,y])^2 \;. \tag{4}$$

However, we refrain from that approach due to memory requirements which would be difficult to meet on a low-cost embedded computer. The standard approach is also very sensitive to global changes of the camera acquisition parameters, which occur whenever large white or black vehicle enter the field of view. Finally, the standard approach makes it difficult to process more than one video stream on a low-cost embedded computer. These shortcomings shall be addressed in the following subsections.

IV. The proposed improvements to the baseline approach

A. Two stage background model

In order to deal with long term illumination changes and stopped cars becoming part of the background model after a prolonged period of waiting for the traffic light change, we propose the following background modelling approach. Specific to this approach is that we build two background models and that is why we call it the two stage approach. The first model in our two stage approach is the baseline background model and it is updated with every frame of the video. The second model is refreshed every N frames with the representation from the first model that is $2N$ frames old. This way we disperse and delay the contribution of the images that were used for updating the first model and hopefully we create a model that is more robust and deals better with aforementioned problems. Also we keep our model adaptive to long term changes and we do not lose information because of the first stage model that is updated with every frame. Both single stage (baseline approach) and two stage model use exponential running average to update with the current frame. Each frame contribution in both single stage and two stage background model is discussed and presented.

The frame contribution in the standard model (2) features the contribution $C_k^{(1)}$ in the frame k:

$$C_k^{(1)} = \alpha \;, \tag{5}$$

$$C_{k-1}^{(1)} = \alpha(1-\alpha) \;, \tag{6}$$

$$C_{k-2}^{(1)} = \alpha(1-\alpha)^2 \;. \tag{7}$$

Let the index of the current frame again be given by k. Then the frame contribution of the two stage dispersed and delayed model in the frame i can be expressed in terms of the contribution of the standard one-stage model $C_i^{(1)}$ and the update parameter β:

$$C_i^{(2)} = \sum_{j=0}^{\lfloor \frac{k-i-1}{N} \rfloor} \beta(1-\beta)^j \cdot C_{i+j\cdot N}^{(1)} \tag{8}$$

The two contribution models are shown in Figure 1. In comparison with the standard model (left), in the two stage model the frame contribution is dispersed and the domination of the most recent frames in the final contribution is reduced.

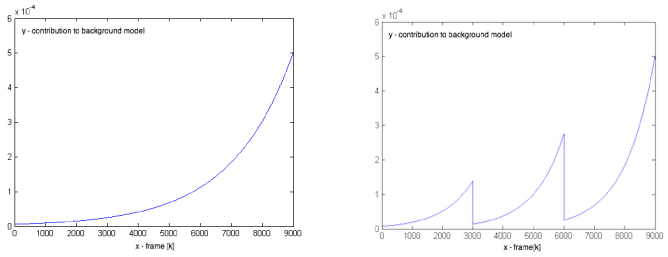


Fig. 1. Background model contribution depending on the frame number in the standard approach (left, $\alpha = 0.001$), and in the proposed two stage approach (right, $\alpha = 0.001$, $\beta = 0.5$, N=3000).

B. Addressing sudden changes of pixel brigthness

Sudden changes of pixel brigthness may occur either due to non-linear illumination variations such as clouds (dis-)occluding the sun or vehicle lights, or due to automatic adaptation of acquisition parameters (gain, shutter). The latter usually occur due to large vehicles with extremely light or extremely dark colours. Background models based on absolute pixel values do not deal very well in those situations because absolute pixel values can change significantly and cause false positive detections to occur. We have considered two different approaches to deal with this problem by designing background models with built-in invariance to absolute pixel brightness:

1) Modelling the gradient distributions: The simplest way to achieve invariance to absolute pixel brightness is to design a model working on image gradient. Image gradient is tricky to work with, since it is not informative in flat regions. Hence we aggregate the gradient information over image patches and represent its distribution in the form of a histogram of oriented gradients (HoG) [6]. Finally, we build the background model consisting of a number of such histograms, recovered over a collection of image patches.

In the implementation, we first divide the image area under the virtual inductive loops into cells grouped in overlapping blocks. Then we calculate the HoG descriptors for each cell and create a normalized feature vector from histogram values for each block. Since the blocks are overlapping, each cell contributes to more than one block. This way we get a model that is more invariant to sudden changes of local brightness, colour contrasts and shadow appearance.

2) Structure-texture decomposition: This approach is based on the image denoising algorithm [11], [7] which represents the input image f as a combination of structure u (representation at the coarse scale) and texture α (additive noise and fine details), as shown in Equation (9).

$$f = u + \eta \tag{9}$$

Performing this denoising process on images with a low amount of additive noise suppresses the texture component of an image and leaves the structure of the image intact. We get the texture component η by subtracting the structure component u from the original image f and use it to build a brightness invariant background model.

C. Speeding up frame processing

One of our goals was to achieve real-time detection system that could run on an embedded computer with a low performance processor. In order to accomplish this we introduce HoG calculation using integral image which is a data structure and algorithm for quickly and efficiently calculating the sum of values in a rectangular region of interest inside the image. We calculate gradients for each cell in a single pass over the image and once the integral image has been computed we can calculate the value of a specific rectangular cell in a constant time and that results in a significant speed-up over the straightforward MoG approach. Frame processing time is measured and presented in the following section, Table III.

V. Experiments

In order to evaluate the proposed detection models we have collected three realistic videos acquired at real urban intersections. Each detection model has a set of hyperparameters affecting detection performance. For example, background model learning rate is a hyperparameter of the detection model. Hyperparameters of the considered detection models are first trained by exhaustive grid-search on validation subsets. The detection models with best hyperparameters are finally evaluated on independent test subsets.

A. Datasets

We have performed experiments using three test videos taken by a camera placed on top of a traffic light post (cf. Fig. 2). The video footage from the first video is is taken during daylight hours and it is far better than the other two in terms of image quality and camera stability. The first video consists of 27500 frames showing one urban intersection. The background models are initially trained on the first 10000 frames. Of the remaining 17500 frames, 7500 were used for validation, 10000 for testing and every fifth frame was manually marked as a validation or a test sample. Therefore, the validation subset consists of a total of 1500 images and the test set contained 2000 images. The second and the third videos are acquired from a different location by a low cost camera producing images with a lot of noise. Additionally, there is significant background motion caused by the wind shaking the camera. The second video is taken during afternoon, and it consists of 27500 frames divided in 10000 frames for pre-training of the background model, 7500 for validation and 1000 for testing. The third video is taken during night-time and it consists of 37500 frames divided in 10000 frames for pre-training, 10000 for validation and 17500 for testing.

Fig. 2. Representative frames from the dataset 1 (left), dataset 2 (middle) and the dataset 3 (right).

The following rules were followed when annotating images in all three videos:

1) Frames in which the vehicle is completely covering the virtual loop area were marked as positives;
2) Frames in which the loop is completely empty were marked as negatives;
3) Frames where the loop area is partially covered were discarded.

Using this annotation policy, we have effectively reduced our first dataset to 1086 validation and 1561 test images, the second dataset was reduced to 1302 validation and 1781 test images and the third to 1365 validation and 3128 test images.

B. Validation

The next phase in experiments included validating parameters for each of the proposed background models. We decided to validate single stage MoG and HoG models, delayed MoG model and finally two stage dispersed and delayed MoG and HoG models. For each parameter affecting these background models we defined a definite set of possible values and performed validation tests using grid search optimisation approach to evaluate every possible detection model with a certain combination of parameter values. When validating standard models, the parameters taken into consideration were the learning rate, pixel difference threshold in MoG models for determining foreground pixels, maximum number of Gaussian mixtures in a MoG model and the difference threshold for a single cell needed to indicate the detection in HoG models. When validating dispersed and delayed models we introduce the learning rate of the short term standard model, the dispersed and delayed model learning rate and the number of frames used for building our short term model. We were interested in optimisation of the aforementioned parameters and wanted to determine at which level each of these parameters affects our results.

We evaluated detection model performance by measuring its precision and recall and calculating the average precision AP as the area under the precision-recall curve. If we denote the precision and recall in k-th data point by $P(k)$ and $r(k)$, then the AP can be determined as follows:

$$AP = \sum_{k}^{n} P(k)\Delta r(k) \tag{10}$$

Our reference parameter for drawing precision-recall curves was percentage of virtual loop covered by the vehicle for MoG models, and cell detection threshold for HoG models. For each considered approach we have found the best set of hyperparameters by exhaustive search on all three validation subsets. The identified best detection models are evaluated on independent test sets, as presented in the next section.

C. Testing

The six considered detection approaches are evaluated on the three independent test subsets. The obtained average precisions are summarized in Table I.

Note that the approach based on texture images (texture+MoG) has been evaluated only in the hardest conditions (night-time, test subset 3) due to huge computational requirements. The obtained results are superior to all other approaches (cf. Table II) despite the fact that the decomposition has been performed with the default parameter λ [7].

TABLE I
AVERAGE PRECISION ON THE THREE INDEPENDENT TEST SUBSETS.

Method	AP 1	AP 2	AP 3
MoG single stage (validated)	98,07%	75,81%	84,07%
MoG delayed (validated)	99,86%	80,29%	49,01%
MoG two stage (validated)	99,43%	82,89%	55,67%
HoG single stage (validated)	99,77%	78,88%	91,26%
HoG two stage (validated)	99,80%	86,59%	91,84%
texture + MoG (not validated)	-	-	97,07%

TABLE II
BEST DETECTION RESULTS IN NIGHT-TIME VIDEO (TEST SUBSET 3)

Method	TP	TN	FP	FN
HoG one stage (validated)	191	2890	22	24
HoG two stage (validated)	174	2896	16	41
texture + MOG (not validated)	208	2908	4	7

The comparison of precision-recall curves is shown in Figs. 3, 4 and 5. Desired operating conditions for each detection model are located close to the upper right corner in each graph. We note that the delayed and the two-stage MoG models perform very good on daylight datasets but fail during night-time. A closer examination showed that this is caused by the delaying which in some cases makes it very hard for the model to pick-up in time the illumination changes during twilight. The best performance is obtained by the HoG detection models who are therefore a clear winner of this evaluation, if we disregard texture decomposition due to computational requirements illustrated below. We note that the two-stage variants do consistently better, especially for the dataset 2. We believe that combining HoG and HoH (histogram of hue) features would actually add to current results, and they shall therefore be revisited in our future work.

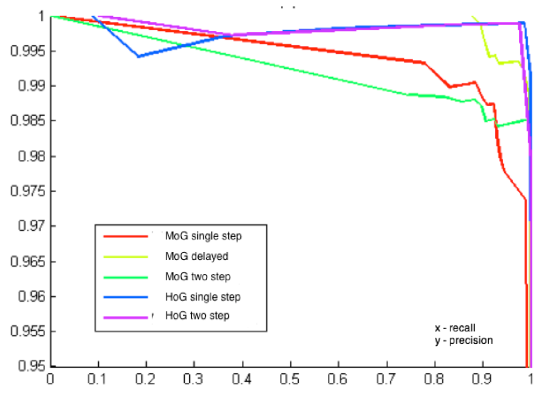


Fig. 3. Comparison of precision-recall curves on the test subset 1.

The average frame processing time (model update + detection) for the three main approaches is shown in Table III. The measurements were obtained using an Intel i5 1.7 GHz

processor. The reason behind fast processing of HoG models are optimizations based on integral images.

TABLE III
AVERAGE FRAME PROCESSING TIME

Method	per-pixel MoG	per-patch HoG	per-pixel texture+MoG
Time	3 ms	1-2 ms	300-500 ms

Texture decomposition shows great potential for building background models invariant to brightness changes, but it also adds a great deal of computational complexity, which makes it inappropriate for implementations on low cost embedded computers.

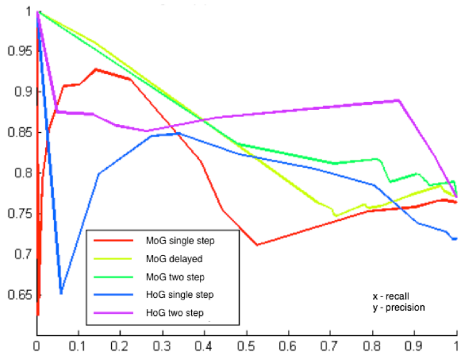


Fig. 4. Comparison of precision-recall curves on the test subset 2.

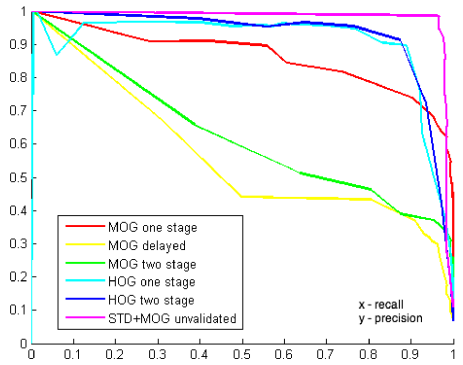


Fig. 5. Comparison of precision-recall curves on the test subset 3.

VI. Conclusion

We have addressed three practical issues which arose while applying background modelling for detecting vehicle presence in urban intersection video. Firstly, we have studied two alternative background models for decreasing sensitivity to disturbances due to illumination changes and automatic adjustments in camera hardware (mostly gain and shutter). The background model based on HOG cells has achieved nearly the same detection accuracy as texture decomposition while requiring much less computational resources. Secondly, we have shown that the HOG background model paired with the integral gradient images allows to have both the best of all worlds: the robustness to brightness-related disturbances and low computational complexity. Thirdly, we have proposed a two-stage modification to the widely used Gaussian update approach (exponential moving average) in order to better approximate the classic running average in the desired time interval. This modification consistently improved the results of HoG models, but failed for MoG models during twilight due to fast daylight illumination changes. Further experiments shall show whether this can be improved by reducing the delay of the background model. Future work shall be devoted to the extraction of additional features such as vehicle class, speed and inter-vehicle distance. Also we shall dedicate time to achieve combination of histogram approaches using both oriented gradients and hue values in order to try to improve presented results.

Acknowledgments

This work has been supported by the project VISTA - Computer Vision Innovations for Safe Traffic, IPA2007/HR/16IPO/001-040514, co-financed by the European Union from the European Regional Development Fund. This research has been supported by the research project Research Centre for Advanced Cooperative Systems (EU FP7 #285939). The three datasets have been kindly provided by Peek Promet d.o.o.

References

[1] O. Barnich and M. V. Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, 2011.

[2] J. Batista, P. Peixoto, C. Fern, and M. Ribeiro. A dual-stage robust vehicle detection and tracking for real-time traffic monitoring. In *Proceedings of the IEEE ITSC 2006*, pages 528–535, 2006.

[3] S. Brutzer, B. Hoferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *Proceedings of the 2011 IEEE CVPR*, CVPR '11, pages 1937–1944, Washington, DC, USA, 2011. IEEE Computer Society.

[4] S.-C. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. *Visual Communications and Image Processing 2004*, 5308(1):881–892, 2004.

[5] S.-C. S. Cheung and C. Kamath. Robust background subtraction with foreground validation for urban traffic video. *EURASIP J. Adv. Sig. Proc.*, 2005(14):2330–2340, 2005.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[7] J. Duran, B. Coll, and C. Sbert. Chambolle's Projection Algorithm for Total Variation Denoising. In *IPOL*, volume 3, pages 311–331, 2013.

[8] M. Harville. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. In *ECCV (3)*, pages 543–560, 2002.

[9] S. Herrero and J. Bescós. Background subtraction techniques: Systematic evaluation and comparative analysis. In *Proceedings of the 11th International Conference on Advanced Concepts for Intelligent Vision Systems*, ACIVS, pages 33–42, Berlin, Heidelberg, 2009. Springer-Verlag.

[10] S. H. Kim, J. Shi, A. Alfarrarjeh, D. Xu, Y. Tan, and C. Shahabi. Real-time traffic video analysis using intel viewmont coprocessor. In *DNIS*, pages 150–160, 2013.

[11] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, Nov. 1992.

[12] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, pages 2246–2252, 1999.

[13] F. Tiburzi, M. Escudero, J. Bescós, and J. M. M. Sanchez. A ground truth for motion-based video-object segmentation. In *ICIP*, pages 17–20. IEEE, 2008.

[14] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):780–785, July 1997.

[15] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *ICPR*, pages 28–31, 2004.

Multi-Label Classification of Traffic Scenes

Ivan Sikirić
Mireo d.d.
Buzinski prilaz 32, 10000 Zagreb
e-mail: ivan.sikiric@mireo.hr

Karla Brkić, Ivan Horvatin, Siniša Šegvić
University of Zagreb
Faculty of Electrical Engineering and Computing
e-mail: karla.brkic@fer.hr,
ivan.horvatin@fer.hr, sinisa.segvic@fer.hr

***Abstract*—This work deals with multi-label classification of traffic scene images. We introduce a novel labeling scheme for the traffic scene dataset FM2. Each image in the dataset is assigned up to five labels: settlement, road, tunnel, traffic and overpass. We propose representing the images with (i) bag-of-words and (ii) GIST descriptors. The bag-of-words model detects SIFT features in training images, clusters them to form visual words, and then represents each image as a histogram of visual words. On the other hand, the GIST descriptor represents an image by capturing perceptual features meaningful to a human observer, such as naturalness, openness, roughness, etc. We compare the two representations by measuring classification performance of Support Vector Machine and Random Forest classifiers. Labels are assigned by applying binary one-vs-all classifiers trained separately for each class. Categorization success is evaluated over multiple labels using a variety of parameters. We report good classification results for easier class labels (*road*, $F1 = 98\%$ and *tunnel*, $F1 = 94\%$), and discuss weaker results (*overpass*, $F1 < 50\%$) that call for use of more advanced methods.**

I. Introduction and related work

Traffic scene classification is an emerging topic with considerable importance in the field of intelligent transportation systems. With the increased availability of cameras in vehicles (either on mobile devices or as embedded hardware in luxurious car models), there are more and more possibilities for simplifying common intelligent transportation tasks. We are especially interested in improving fleet management systems. Fleet management systems are used to track the status of fleets of vehicles belonging to various kinds of companies (e.g. taxi, delivery, cargo transport etc.). They use GPS sensors to track the location of the vehicle, but have little information about the vehicle's environment. Some useful information about the vehicle's surroundings can be inferred by using a camera to record images from the driver's perspective, and then solving a classification problem to detect interesting types of traffic scenes and scenarios. For example, this approach can be used to identify traffic jams, or to differentiate open road environments from urban/rural roads or tunnels.

Image classification in general is a common topic in computer vision, extensively researched in great number of papers. Active research focuses mainly on recognizing images in a large number of diverse classes [1]. The performance of new image classification techniques is usually evaluated on one or more of many publicly available benchmark datasets (e.g. Pascal VOC, Caltech 101, LabelMe etc). This enables a simple and meaningful comparison of state-of-the-art methods applied on various domains.

A common approach to image classification is to first reduce the dimensionality of the image representation using an image descriptor, and then use a general-purpose classifier to perform the classification. Commonly used classifiers are Support Vector Machine (SVM) [2] and Random Forest [3]. Among the best performing general-purpose image descriptors are the bag-of-words model [4], [5], [6], and its derivatives: Locality-constrained Linear Coding (LLC) [7], Fisher vectors (FV) [8] and Spatial Fisher vectors (SFV) [9]. The basis of these methods is finding local image features (e.g. SIFT [10]) and expressing their distribution and relative spatial relations, thus producing a short code that represents the image. Another successful image descriptor is GIST [11], [12], which is not general-purpose, but is designed specifically for scene classification purpose. It captures a set of semantic properties of an image (e.g. naturalness or openness) by measuring responses from several orientation filter over a fixed grid.

The volume of work focused on classifying traffic scenes is considerably smaller than generic image classification research. A small number of works apply general-purpose methods on the problem [13]. Most works present methods that are crafted specifically for classification and understanding of traffic scenes. For instance, Tang and Breckon [14] identify three regions of interest in a traffic scene image: (i) a rectangular region near the center of the image, (ii) a tall rectangular region on the left side of the image and (iii) a wide rectangular region at the bottom of the image. Each of the three regions of interest is represented by a predefined set of local features, as specific features are expected to respond to specific structures which occur in a traffic scene image (e.g. road, or road edge). They introduce a new dataset with four classes: motorway, offroad, trunkroad and urban road. Mioulet et al. [15] build on the ideas of Tang and Breckon [14], retaining the three predefined regions of interest, but representing them with different types of local features and using dedicated hardware.

In our previous work [16], [13], we evaluated classification of traffic scenes in a single-label setup. The main focus was not on the selected labeling approach, but instead on minimizing the image representation size, and on discussing implementation issues specific to fleet management systems. In this paper we evaluate the multi-label classification performance of general purpose image classification methods on traffic scene images. We use the bag-of-words and GIST descriptors combined with SVM and Random Forest classifiers. The performance is evaluated on the FM2 dataset[1] [13] of traffic scene images. Publicly available labeling assigns a single label to each image, even in cases where an image clearly belongs to two or more classes. We introduce a novel labeling scheme for this dataset, in which each image is assigned up to five

[1] http://www.zemris.fer.hr/~ssegvic/datasets/unizg-fer-fm2.zip

labels: settlement, road, tunnel, traffic and overpass.

II. The FM2 dataset

The FM2 dataset contains 6237 images of traffic scenes captured on Croatian roads from the driver's perspective, mostly on highways. The resolution of the images is 640x480. Most of the images were taken on a clear and sunny day. No images were taken during nighttime.

The publicly available labeling of the FM2 dataset assigns a single label per image. In reality, many traffic scenes belong to more than one class (for example, classes *settlement* and *overpass* are not mutually exclusive). Using a single-label classifier in such cases results in an unnecessary loss of information. For that reason, a multi-label approach, in which a set of class labels can be assigned to a single image is a more appropriate solution. In our novel labeling scheme each image is assigned a set of class labels.

We selected five class labels: *settlement*, *road*, *tunnel*, *traffic* and *overpass*. The overview and brief description of the classes is given in Table I. Classes *settlement*, *open road* and *tunnel* describe the location of the vehicle, and their labels are mutually exclusive. Classes *overpass* and *traffic* were chosen because they are interesting for fleet management systems, as described in [16], [13]. The overpass class label usually coexists with the road label, but it can also occur in settlements. The traffic label can occur with any other label. It is also possible that it will be the only label assigned to an image (if a large truck directly in front of camera completely obstructs the view). Some examples of labeled images are shown in Figure 1.

III. Methods

In this paper, we compare two different image representations in a multi-label classification setting. The first considered representation is the bag-of-words model [17], and the second considered representation is the GIST descriptor [11], [12]. For each of these representations, we trained two different classifiers: Support Vector Machine (SVM) [2] and Random Forest [3].

A. Multi-label classification methods

Existing methods for multi-label classification fall into two main categories [18]: (i) problem transformation methods and (ii) algorithm adaptation methods. The problem transformation methods transform the original problem into one or more single-label classification or regression problems. The algorithm adaptation methods do not transform the problem, but rather they adapt the learning algorithms themselves to handle multi-label data. Since we want to evaluate (among other things) the performance of standard SVM algorithm on this problem, we focus on the problem transformation methods. Two most commonly used problem transformation methods [19] are *label power-set method* [20] and *binary relevance* [21].

Label power-set method works by assigning each distinct subset of labels that occurs in the data its own unique label, thus transforming multi-label problem into a single-label one. This method will capture any existing dependence between labels (e.g. in FM2 dataset label *overpass* must coexist with either *road* or *settlement* label, while it cannot coexist with *tunnel* label). One major problem with this approach is having a large number of classes: in case of K labels, the number of resulting classes can be up to 2^K. This usually leads to some classes being represented with very few examples. Since number of examples per class is already low in the FM2 dataset, we chose not to use this method. Instead, we used the binary relevance method.

Binary relevance method works by creating K datasets from the original dataset, where K is the number of classes, and training a separate classifier for each of them. Each of the K datasets contains the same samples as the original dataset, but the labels are different, as they indicate whether the given sample belongs to the class k. Once the transformed datasets are obtained, it is a simple matter to train a binary classifier on each of them. The output for each sample is the union of the outputs for all K classifiers. Even though this method is unable to learn the dependence between labels, it has other advantages. It is suited for applications where label relationships may change over datasets (e.g. it might be able to properly classify scenes with both labels *settlement* and *overpass*, even if no such examples were present in the original training dataset). Its main advantage, however, is its low computational complexity, which scales linearly with the number of classes.

B. The bag-of-words model

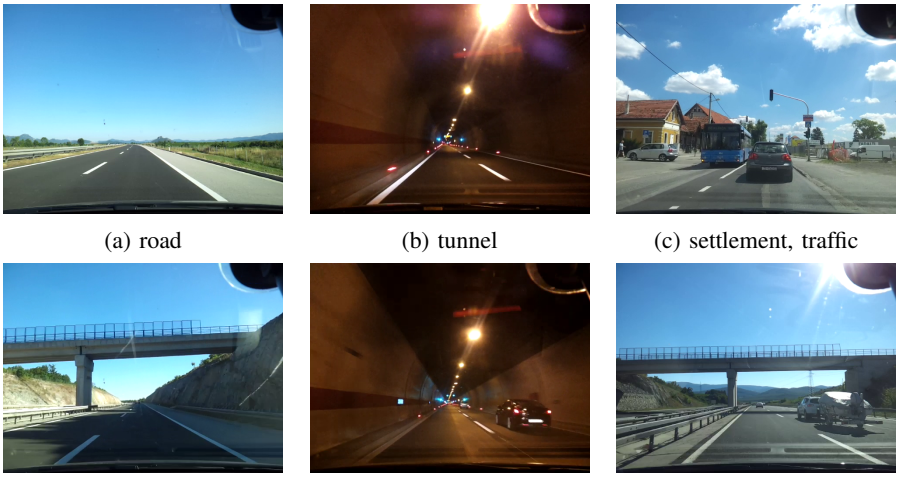
The bag-of-words image representation was adopted into computer vision from the field of text mining. In text mining, a bag-of-words model represents a textual document by a histogram of occurrences of words from a dictionary (thus disregarding their ordering in the document). Similarly, an image can be represented by a histogram of visual words. Local image features can be used as visual words, but the number of all possible local features is too large to represent a dictionary. For this reason, a dictionary of visual words is obtained by sampling local image features from each image in a dataset, and then clustering them into a set of more manageable size. Each cluster center represents a single visual word, and any local feature is considered to be the same visual word as its nearest cluster center. In this work we used SIFT (Scale Invariant Feature Transform) [10] algorithm to extract local features, and *k-means* clustering [22] to produce a dictionary of visual words.

Extraction of SIFT features was done using the implementation from the VLFeat library [23]. Local SIFT features are considered to be stable if they are invariant to changes in scale, orientation, illumination and noise. In the VLFeat library, the amount of features extracted from an image is regulated by two parameters of the extraction algorithm: *peak-thresh* and *edge-thresh*. The *peak-thresh* parameter represents the threshold on the contrast of the extracted features and is used to discard the low-contrast features. The *edge-thresh* parameter represents the threshold on the curvature of the extracted features, and is used to discard edge responses in favor of corner responses. The effect of varying these parameters can be seen on Figures 2, 3, 4 and 5.

A dictionary of visual words was obtained by organizing sampled local features into clusters. We used *k-means* cluster-

class label	class description	number of occurrences
settlement	vehicle is in a settlement	412
road	an open road scene	5239
tunnel	vehicle is in a tunnel, or directly in front of it	681
traffic	other vehicles are visible	2411
overpass	vehicle will soon be, or is already under an overpass	194

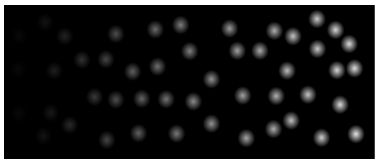
TABLE I: Selected class labels



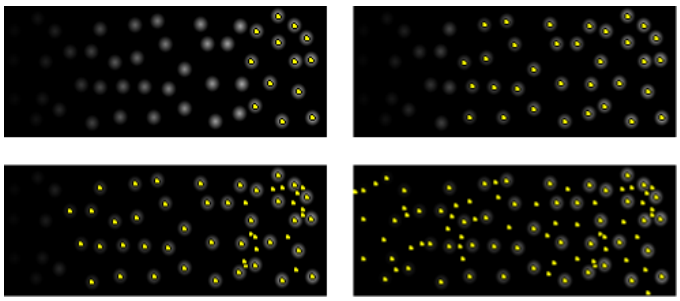
(a) road (b) tunnel (c) settlement, traffic

(d) road, overpass (e) tunnel, traffic (f) road, overpass, traffic

Fig. 1: Examples of labeled images

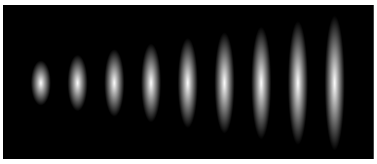


(a) Test image for setting the *peak-thresh* parameter

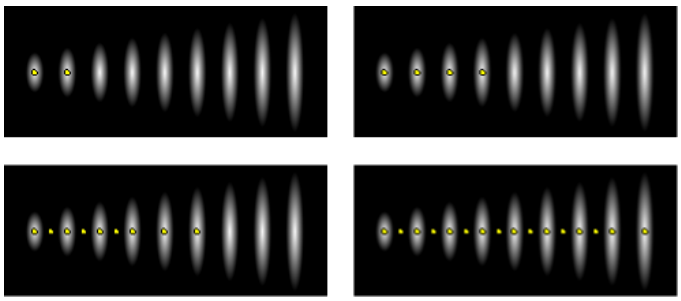


(b) Detected features for varying values of parameter *peak-thresh* (starting from top left: 20, 10, 5, 0)

Fig. 2: Effects of varying the *peak-thresh* parameter



(a) Test image for setting the *edge-thresh* parameter



(b) Detected features for varying values of parameter *edge-thresh* (starting from top left: 7, 10, 15, 25)

Fig. 3: Effects of varying the *edge-thresh* parameter

ing algorithm [22]. It is an iterative algorithm that minimizes the error term:

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in S_k} ||\mathbf{x}_i - \mu_k||^2 \quad (1)$$

where K is the desired number of clusters, μ_k is the centroid of cluster k, and S_k is the set of all feature vectors $\mathbf{x}_i$ in cluster k. The initialization of centroids is random, so this algorithm is run several times to increase the chance of finding the global optimum.

C. The GIST image descriptor

While the bag-of-words model can be applied to images of any kind, the GIST descriptor [11], [12] has been developed specifically for scene recognition. It is a low dimensional representation of the scene that captures perceptual features

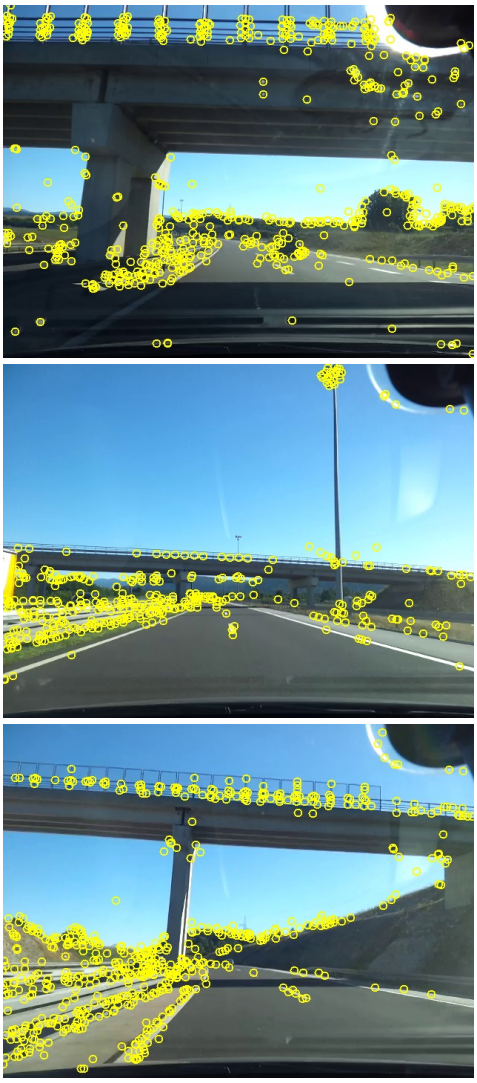


Fig. 4: SIFT features extracted on overpass images for extraction parameters of $edge{-}thresh = 10$ and $peak{-}thresh = 5$

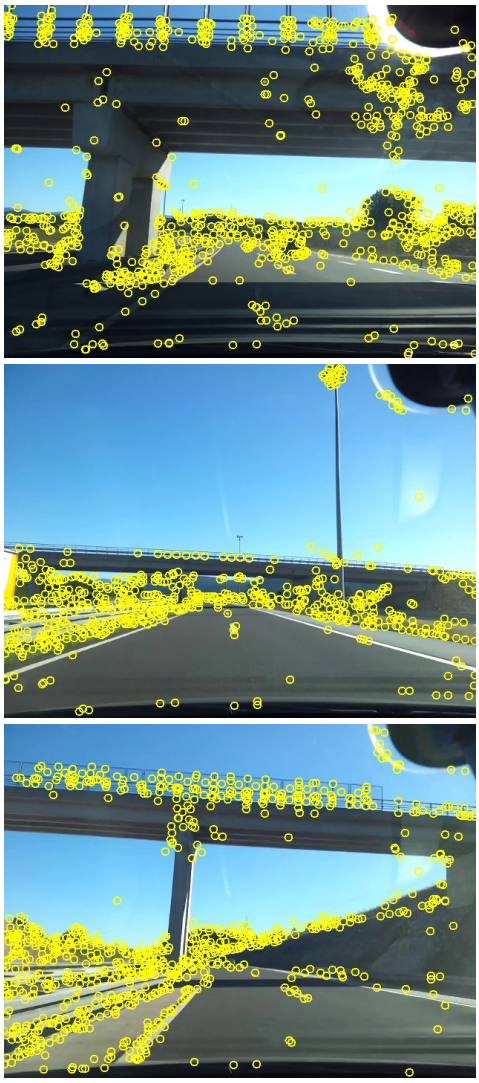


Fig. 5: SIFT features extracted on overpass images for extraction parameters of $edge{-}thresh = 10$ and $peak{-}thresh = 2$

of the scene that are meaningful to a human observer, such as naturalness, openness, roughness, etc. To calculate the GIST descriptor, one first subdivides the image into 16 regions (a 4×4 grid), and then concatenates the average energies of 32 orientation filter responses (8 orientations on 4 scales) for each cell. Therefore the length of the feature vector is $16 \cdot 32 = 512$. Since GIST is designed to ignore accidental presence of small objects in the scene, we expect it to perform better on class labels *road*, *settlement* and *tunnel* than on *traffic* and *overpass* (depending on how much the other vehicles / overpass are dominant in the scene).

D. Support Vector Machine

Support Vector Machine (SVM) [2] is a binary classifier which constructs a maximum-margin hyperplane that divides two sets of vectors. The construction of the hyperplane is done in the learning stage using labeled vectors. SVM is expected to generalize well because of maximizing the margin between sets. To allow for outliers in the learning dataset, we chose to use a variant of the algorithm called *soft-margin* SVM. It introduces an error term ξ_i that allows for misclassified instances, thus sacrificing linear separability in favor of stability:

$$\begin{aligned} &\arg\min_{\mathbf{w},\xi,b} \left(\frac{1}{2}||\mathbf{w}||^2 + C \sum \xi_i \right) \\ &y_i(\mathbf{x}_i \cdot \mathbf{w} - b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \end{aligned} \tag{2}$$

where y_i is the class label of point $\mathbf{x}_i$, and $\mathbf{w}$ and b are the parameters of the hyperplane. Parameter C can be used to choose how much error is to be allowed in the classification process. The lower it is, the more outliers will be tolerated. The higher it is, the closer we get to regular SVM algorithm. Figure 6 illustrates the effects of varying the parameter C. Big circles represent the vectors that are taken into consideration when maximizing the margin of the hyperplane.

E. Random Forest classifier

Random Forest classifier was developed by Breiman and Cutler [3]. The basic idea of the algorithm is to combine

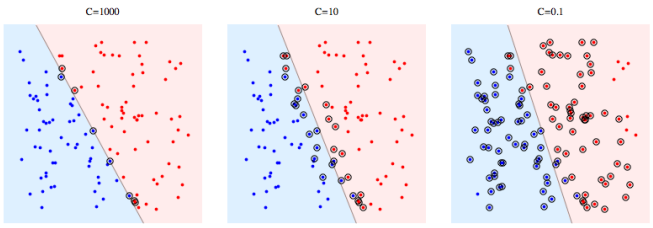


Fig. 6: Examples of margins for various values of C parameter in *soft-margin* SVM, illustrated on a toy problem in 2D space. Points belong to two classes, red and blue. Support vectors are circled. Figure courtesy of Yaroslav Bulatov.

many simple classifiers (decision trees) into a complex one. A decision tree is a classifier in which leaf nodes represent the outcome of the classification (class labels), and inner nodes (called decision nodes) direct the traversal of the tree by thresholding a specific subset of m attributes of the input feature vector. The m attributes evaluated at a given node are selected in a way that maximizes the information gain in the current subset of training data. Hundreds of samples are required to build a decision tree with good classification performance.

A Random Forest consists of many decision trees, where each tree is different because of randomized initialization. The final outcome of the classification is decided by voting of all decision trees. The error of a Random Forest classifier depends on the errors of individual decision trees, as well as on the level of correlation between trees (high correlation results in greater error). Parameter m directly affects the level of correlation and the error of individual trees. The higher the parameter m is, the greater the correlation, but the lower the error of trees becomes.

IV. Experiments and results

In this section we describe the performed multi-label classification experiments. Each image in the dataset was represented using both bag-of-words and GIST image descriptors. Since we used *binary relevance* multi-label classification method on the dataset with $K = 5$ classes, the labels of the dataset were separated into five distinct sets, one for each class. Subsequently, we trained five separate binary classifiers in a one-vs-all fashion. We used 70% of each set for training, while the rest was used for evaluation. The output for each sample is the union of the outputs for all K classifiers. The classifiers we evaluated were Support Vector Machine (SVM) and Random Forest. Two types of classifiers in combination with two types of image descriptors yield a total of four different classification setups.

For the GIST descriptor we used an implementation provided by its authors [11]. For the bag-of-words descriptor we used the solution developed in [24], which uses the VLFeat library [23] implementation of the SIFT algorithm. For the SVM and Random Forest classifiers we used the *scikit-learn* Python library [25]. The same library provides an implementation of *k-means* clustering algorithm, which was used to produce the dictionary of visual words in bag-of-words model. A simple grid search optimization was used to tune the parameters C and m of the classifiers, but the results were nearly identical for a wide range of parameter values.

The performance measure we chose to use if the F1 measure, which is the harmonic mean of precision and recall measures, and is calculated as:

$$F1 = \frac{2T_p}{2T_p + F_n + F_p} \tag{3}$$

where T_p, F_n and F_p are the number of true positives, false negatives and false positives, respectively.

The detailed per-class results are shown in Table II. All combinations of classifiers and descriptors have shown similar performance for every class. Very good performance was achieved on *road* and *tunnel* classes ($F1 \geq 0.94$). Moderate performance was achieved on *settlement* and *traffic* classes ($0.64 \leq F1 \leq 0.86$). Very poor performance was shown on the class *overpass* ($F1 \leq 0.51$). For successful classification of *overpass* and *traffic* images, in many cases it is necessary to consider some small detail of the scene (the overpass and vehicles are often in the distance, and rarely dominate the scene). Since GIST is designed for scene recognition, rather than being a general-purpose descriptor, it is not surprising that it often fails to capture such details. Similarly, our implementation of bag-of-words model is expected to have problems with the same type of images. Since the implementation we used extracts only stable SIFT features, it is likely that in many cases very few local features were extracted in the regions of important, but small details in the scene. It is important to note that the dataset contains several thousand images with the *traffic* label, but only a couple hundred with *overpass* label, which explains the difference in performance for those classes. The class *tunnel* is easy to classify because all the tunnel images are very similar to each other, and very different from images of other classes. On the other hand, the appearance of *settlement* images varies greatly, which makes their classification a more difficult task. To improve the classification performance of *settlement* class, we should include much more training examples. The best performance is achieved for *road* images, which are the most occurring image type in in the dataset, are not defined by small details in the scene, and are similar to each other in appearance.

V. Conclusion and future work

The proposed methods have shown remarkably good results for some class labels (*road* and *tunnel*), while performing rather poorly on some other class labels (*overpass*). Both classifiers (SVM and Random Forest), and both descriptors (bag-of-words) showed similar level of performance (both overall, and per-class). Therefore, we conclude that classes *settlement* and *traffic* are moderately hard to classify, and the class *overpass* is very hard to classify. This calls for use of more advanced methods, and expansion of the dataset to include much more instances of those classes (especially for the case of settlement, which is the class with the greatest variability in appearance). The GIST descriptor is designed for scene classification, and is expected to perform poorly in capturing small details in a scene (such as occasional vehicle and overpass in the distance). Performance of bag-of-words model is expected to be improved by using dense SIFT extractor, instead of the keypoint-driven one, because

SVM ($C = 1$) on bag-of-words

	precision	recall	F1
settlement	0.76	0.78	0.77
tunnel	0.94	0.94	0.94
road	0.98	0.97	0.98
traffic	0.62	0.70	0.66
overpass	0.47	0.39	0.42
average	0.86	0.88	0.87

Random Forest (500 trees) on bag-of-words

	precision	recall	F1
settlement	0.96	0.56	0.71
tunnel	0.99	0.89	0.94
road	0.97	1.00	0.98
traffic	0.79	0.53	0.64
overpass	1.00	0.06	0.11
average	0.92	0.83	0.86

SVM ($C = 1$) on GIST descriptor

	precision	recall	F1
settlement	0.59	0.97	0.73
tunnel	0.92	0.96	0.94
road	0.99	0.98	0.99
traffic	0.71	0.79	0.75
overpass	0.36	0.87	0.51
average	0.88	0.92	0.90

Random Forest (500 trees) on GIST descriptor

	precision	recall	F1
settlement	0.90	0.82	0.86
tunnel	0.99	0.91	0.95
road	0.99	0.99	0.99
traffic	0.89	0.75	0.81
overpass	1.00	0.30	0.46
average	0.96	0.90	0.92

TABLE II: results for linear SVM ($C = 1$) and Random Forest classifiers (500 trees) on the bag-of-words and GIST descriptors

in many occasions there were too few local features captured on some important part of the scene (such as a vehicle or an overpass in the distance). Other possible improvements include using RootSIFT method for comparing SIFT descriptors [26], adding spatial coding to bag-of-words (SPM or 1+4+3), and using RBF or other kernel in case of SVM. For our future work, we plan to expand the scope of multi-label classification experiments to the same extent as single-label experiments in our previous work [13]. This includes evaluating other types of image descriptors (Locality-constrained Linear Coding and Spatial Fisher vectors) as well as considering very small image representations. That will give us a strong basis for comparison of single-label vs multi-label classification performance from the user's standpoint.

ACKNOWLEDGMENTS

This research has been supported by the research project Research Centre for Advanced Cooperative Systems (EU FP7 #285939).

REFERENCES

[1] A. Pinz, "Object categorization," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, no. 4, 2005.

[2] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[3] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[4] F.-F. Li and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *CVPR*, (Washington, DC, USA), pp. 524–531, IEEE Computer Society, 2005.

[5] A. Bosch, A. Zisserman, and X. Muñoz, "Scene classification via pLSA," in *ECCV*, (Berlin, Heidelberg), pp. 517–530, Springer-Verlag, 2006.

[6] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, (Washington, DC, USA), pp. 2169–2178, IEEE Computer Society, 2006.

[7] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *CVPR*, pp. 3360–3367, 2010.

[8] F. Perronnin and C. R. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*, 2007.

[9] J. Krapac, J. J. Verbeek, and F. Jurie, "Modeling spatial layout with Fisher vectors for image categorization," in *ICCV*, 2011.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, 2004.

[11] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vision*, vol. 42, pp. 145–175, May 2001.

[12] A. Oliva and A. B. Torralba, "Scene-centered description from spatial envelope properties," in *BMCV*, (London, UK, UK), pp. 263–272, Springer-Verlag, 2002.

[13] I. Sikirić, K. Brkić, J. Krapac, and S. Šegvić, "Image representations on a budget: Traffic scene classification in a restricted bandwidth scenario," *IEEE Intelligent Vehicles Symposium*, 2014.

[14] I. Tang and T. Breckon, "Automatic road environment classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 476–484, June 2011.

[15] L. Mioulet, T. Breckon, A. Mouton, H. Liang, and T. Morie, "Gabor features for real-time road environment classification," in *ICIT*, pp. 1117–1121, IEEE, February 2013.

[16] I. Sikirić, K. Brkić, and S. Šegvić, "Classifying traffic scenes using the GIST image descriptor," in *CCVW 2013 Proceedings of the Croatian Computer Vision Workshop*, pp. 1–6, September 2013.

[17] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[18] G. Tsoumakas and I. Katakis, "Multi label classification: An overview," *International Journal of Data Warehouse and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[19] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning and Knowledge Discovery in Databases*, vol. 5782, pp. 254–269, 2009.

[20] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Proceedings of the 18th European Conference on Machine Learning*, ECML '07, (Berlin, Heidelberg), pp. 406–417, Springer-Verlag, 2007.

[21] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 22–30, Springer, 2004.

[22] J. MacQueen, "Some methods for classification and analysis of multivariate observations.." Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[23] A. Vedaldi and B. Fulkerson, "VLFeat - an open and portable library of computer vision algorithms," in *ACM International Conference on Multimedia*, 2010.

[24] I. Horvatin, "Multi-label classification of traffic scenes," Master's thesis, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2014.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Convolutional Neural Networks for Croatian Traffic Signs Recognition

Vedran Vukotić, Josip Krapac and Siniša Šegvić
University of Zagreb - Faculty of Electrical Engineering and Computing
Zagreb, HR-10000, Croatia
Email: vevukotic@gmail.com

***Abstract*—We present an approach to recognition of Croatian traffic signs based on convolutional neural networks (CNNs). A library for quick prototyping of CNNs, with an educational scope, is first developed[1]. An architecture similar to LeNet-5 is then created and tested on the MNIST dataset of handwritten digits where comparable results were obtained. We analyze the FER-MASTIF TS2010 dataset and propose a CNN architecture for traffic sign recognition. The presented experiments confirm the feasibility of CNNs for the defined task and suggest improvements to be made in order to improve recognition of Croatian traffic signs.**

I. Introduction

Traffic sign recognition is an example of the multiple classes recognition problem. Classical approaches to this problem in computer vision typically use the following well-known pipeline: (1) local feature extraction (*e.g.* SIFT), (2) feature coding and aggregation (*e.g.* BOW) and (3) learning a classifier to recognize the visual categories using the chosen representation (*e.g.* SVM). The downsides of these approaches include the suboptimality of the chosen features and the need for hand-designing them.

CNNs approach this problem by learning meaningful representations directly from the data, so the learned representations are optimal for the specific classification problem, thus eliminating the need for hand-designed image features. A CNN architecture called *LeNet-5* [1] was successfully trained for handwritten digits recognition and tested on the MNIST dataset [2] yielding state-of-art results at the time. An improved and larger CNN was later developed [3] and current state-of-the-art results on the GTSRB dataset [4] were obtained.

Following the results by [3], we were motivated to evaluate a similar architecture on the Croatian traffic signs dataset FER-MASTIF TS2010 [5]. To do so, we first developed a library that would allow us to test different architectures easily. After different subsets were tested for successful convergence, an architecture similar to *LeNet-5* was built and tested on the MNIST dataset, yielding satisfactory results. Following the successful reproduction of a handwritten digit classifier (an error rate between 1.7% and 0.8%, where LeNet-X architectures yield their results), we started testing architectures for a subset of classes of the FER-MASTIF TS2010 dataset.

In the first part of this article, CNNs are introduced and their specifics, compared to classical neural networks, are presented. Ways and tricks for training them are briefly explained. In the second part the datasets are described and the choice of a subset of classes for the FER-MASTIF TS2010 dataset is elaborated. In the last part of the paper, the experimental setup is explained and the results are discussed. Finally, common problems are shown and suggestions for future improvements are given.

[1] Available at https://github.com/v-v/CNN/

II. Architectural specifics of CNNs

Convolutional neural networks represent a specialization of generic neural networks, where the individual neurons form a mathematical approximation of the biological visual receptive field [6]. Visual receptive fields correspond to small regions of the input that are processed by the same unit. The receptive fields of the neighboring neurons overlap, allowing thus robustness of the learned representation to small translations of the input. Each receptive field learns to react to a specific feature (automatically learned as a kernel). By combining many layers, the network forms a classifier that is able to automatically learn relevant features and is less prone to translational variance in data. In this section, the specific layers (convolutional and pooling layers) of CNNs will be explained. A CNN is finally built by combining many convolutional and pooling layers, so the number of output in each successive layer grows, while size of images on the output is reducing. The output of the last CNN layer is a vector image representation. This image representation is then classified using a classical fully-connected MLP [3], or another classifier, *e.g.* an RBF network [1]).

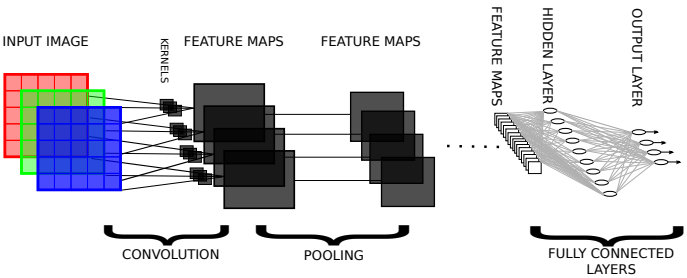


Fig. 1: Illustration of the typical architecture and the different layers used in CNNs. Many convolutional and pooling layers are stacked. The final layers consist of a fully connected network.

A. Feature maps

Fig. 2 shows a typical neuron (a) and a feature map (b). Neurons typically output a scalar, while feature maps represent

the two-dimensional output of an operation performed by a CNN unit. Typical operations performed in a CNN are convolutions and pooling operations. The former learns a feature (convolution kernel), while the latter only reduces the dimensionality by aggregation. These operations will be discussed in the following subsections.

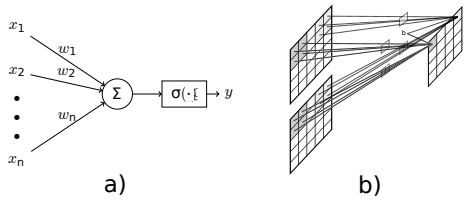


Fig. 2: CNN elements: a) a classical neuron and its connectivity b) a feature map in a convolutional operation (the two output pixels are computed with the same kernels)

B. Convolution

Convolutional layers compute feature maps by convolving the previous layer with a specific kernel. Let M^l be a feature map of layer l and M^{l-1} a feature map of the previous layer. The width and height of a feature map is indicated with M_w and M_h while the width and hight of kernels are indicated with K_w and K_h. Let S_w and S_h represent the horizontal and vertical steps of the kernel during a convolution operation. The sizes of the output feature maps (of the current layer) are then dependent on the sizes of feature maps from the previous layer, kernels and stepping factors. The output width and height are given by Eq (1) and Eq. (2), correspondingly.

$$M_w^l = \frac{M_w^{l-1} - K_w}{S_w} + 1 \tag{1}$$

$$M_h^l = \frac{M_h^{l-1} - K_h}{S_h} + 1 \tag{2}$$

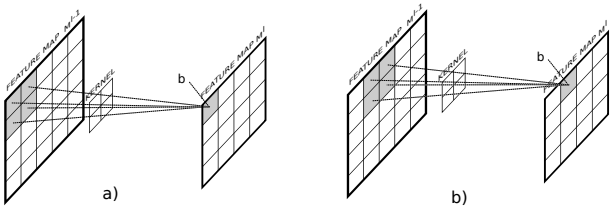


Fig. 3: Illustrated convolution a) first step b) next step, after moving the receptive field for S_w locations

After each convolution, a bias is added to each element and the result is passed through an activation function (see II-D).

Convolutional layers can either have full connectivity or sparse connectivity. In case of full connectivity, each feature map of the current layer is connected with every feature map from the previous layer. Each connection is represented by a kernel, so a convolutional layer that is fully connected will have $|M^l| \cdot |M^{l-1}|$ kernels. Fully connected convolutional layers are used by most authors, *e.g.* [7] and [3].

Sparse connectivity is a way of connecting feature maps in convolution operations where each feature map from the current layer is connected only to a specific subset of feature maps from the previous layer. The benefits of this approach are reduced computational complexity and improved generalization, as the network is forced to learn different features. When using fully connected convolutional layers there is a chance that the network will learn a less diverse set of features [1], [8].

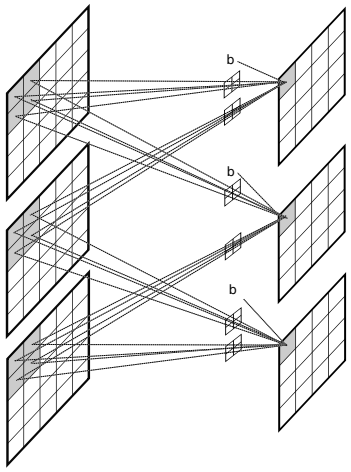


Fig. 4: An example of sparse connectivity of a convolutional layer. Each feature map is connected to only a subset of feature maps from the previous layer

C. Pooling

Pooling layers reduce dimensionality of feature maps from the previous layer by aggregating and representing the grouped features by one feature. An illustration of a generic pooling operation is shown in Fig. 5 b) where the features of the previous layer are grouped in 2×2 areas and are represented in the current map with one element. There are many different pooling operations but the most common ones are mean-pooling and max-pooling. The former represents the group with the average value of all the features withing the group, while the latter represents the group with the maximum element found within the group. Mean pooling was used in earlier works [1], but in recent works max pooling is mostly used, as it always outperforms mean pooling [9] and is additionally faster than mean pooling.

There are a few modern parametric pooling operations that can outperform max-pooling in terms of the quality of representation [10], [11]. However they are more computational expensive, require fine-tuning of additional hyper-parameters and were thus not used in this work.

D. Activation functions

An activation function is a sigmoid-shaped function that maps an input to its output, constrained to a defined range. Just as in classical multilayer perceptrons, they are also used in convolutional neural networks where they are applied to each element of a feature map. To be able to use the error backpropagation algorithm for training of CNN the activation function should be derivable. The two most commonly used activation functions [8], [12] are the logistic function, defined

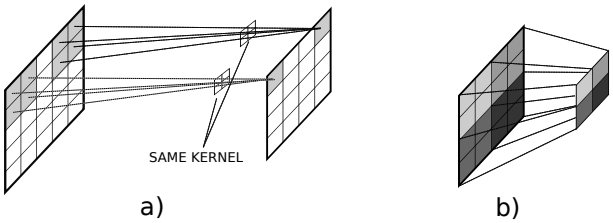


Fig. 5: a) Weight sharing in convolutional layers, the same kernel is used for all the elements within a feature map b) a generic pooling operation

in Eq. (3) and a scaled tanh sigmoidal function, defined in Eq. (4)

$$f(x) = \frac{2}{1 + e^{-\beta x}} - 1 \tag{3}$$

$$f(x) = 1.7159 \tanh\left(\frac{2}{3}x\right) \tag{4}$$

Weights and kernel elements are randomly initialized by using a uniform distribution [13]. However, the sampling interval depends on the choice of the activation function. For the logistic function, the interval is given in (5), while for the scaled tanh sigmoidal function the interval is given in (6). In those equations n_{in} indicates the number of neurons (or feature map elements) in the previous layer, while n_{out} indicates the number of neurons (or feature map elements) in the current layer. Improper initialization may lead to poor convergence of the network.

$$\left[\sqrt{\frac{6}{n_{in} + n_{out}}}, \sqrt{\frac{6}{n_{in} + n_{out}}}\right] \tag{5}$$

$$\left[-4\sqrt{\frac{6}{n_{in} + n_{out}}}, 4\sqrt{\frac{6}{n_{in} + n_{out}}}\right] \tag{6}$$

III. Training CNNs

In supervised training, convolutional neural networks are typically trained by the Backpropagation algorithm. The algorithm is the same as for multilayer perceptrons but is extended for convolutional and pooling operations.

A. Backpropagation for convolutional layers

For convolutional layers, the backpropagation algorithm is the same as for multilayer perceptrons (if every element of a feature map is treated as a neuron and every element of a kernel is treated as a weight) with the only exception that weights are shared inside a feature map. Fig. 5 a) illustrates weight sharing in a convolution between two feature maps. The weight sharing easily fits into backpropagation algorithm: because multiple elements of the feature map contribute to the error, the gradients from all these elements contribute to the same set of weights.

B. Backpropagation for pooling layers

Pooling layers require a way to reverse the pooling operation and backpropagate the errors from the current feature map to the previous one. For the case of max-pooling, the error propagates to the location where the maximal feature is located while the other locations receives an error of zero. For mean-pooling, the error is equally distributed within the locations that were grouped together in the forward pass and can be expressed as $\mathbf{E}' = \mathbf{E} \otimes \mathbf{1}$, where $\mathbf{E}'$ is the error of the previous layer, $\mathbf{E}$ the error of the current layer and $\otimes$ represents the Kronecker product.

IV. Improving learning

A. Backpropagation with momentum

The classical Backpropagation algorithm uses a global learning rate η to scale the weight updates. This modified version scales the learning rate dynamically depending on the partial derivative in the previous update. The method is defined in the Eq. (7), where α represents the amortization factor (typical values between 0.9 and 0.99).

$$\Delta w(t) = \alpha \Delta w(t-1) - \eta \frac{\partial E}{\partial w}(t) \tag{7}$$

B. Adding random transformations

Generalization can be improved by increasing the number of training samples. However that usually requires additional human effort for collection and labelling. Adding random transformations can increase the generalization capability of a classifier without additional effort. There are two main ways for deploying random transformations into a system: (1) by integrating them into the network, after the input layers [3] or (2) by generating additional samples and adding them to the training set [14]. In this work we opted for generating additional samples since the code is currently not optimized for speed, and adding an additional task to be performed for each iteration would further slow the learning process.

C. Dropout

A typical approach in machine learning when improving generalization consists of combining different architectures. However, that can be computationally quite expensive. The dropout method suggests randomly disabling some hidden units during training, thus generating a large set of combined virtual classifiers without the computational overhead [15]. For a simple multilayer perceptron with N neurons in one hidden layer, 2^N virtual architectures would be generated when applying dropout. Usually, half the units are disabled in each learning iteration and that is exactly what we used in this work.

V. Datasets

Two datasets were used in this work. The MNIST dataset of handwritten digits [2] and the FER-MASTIF TS2010 dataset of Croatian traffic signs [5] [16].

A. *MNIST*

The MNIST dataset consists of 10 classes (digits from 0 to 9) of 28×28 grayscale images. It is divided into a training set of 60000 samples and a testing set of 10000 samples. The dataset was unaltered except for preprocessing it to a mean of zero and unit variance.

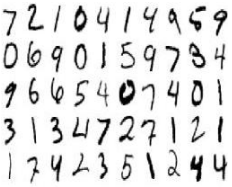


Fig. 6: Samples from the MNIST dataset

B. *FER-MASTIF TS2010*

This dataset consists of 3889 images of 87 different traffic signs and was collected with a vehicle mounted video camera as a part of the MASTIF (Mapping and Assessing the State of Traffic InFrastructure) project. In [17], images were selected by the frame number and split in two different sets, a training set and a test set. This method ensured that images of the same traffic sign do not occur in both sets. The sizes varies from 15 pixels to 124 pixels. We opted for classes containing at least 20 samples of sizes greater or equal to 44×44. Fig. 8 shows the nine classes that met that criteria.

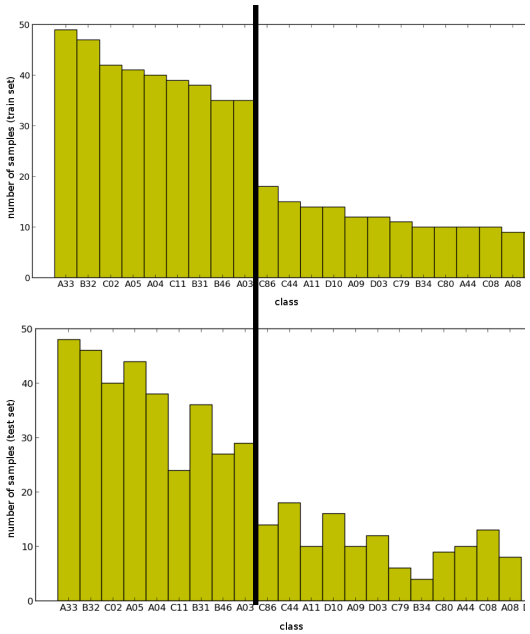


Fig. 7: Part of the histogram showing the number of samples bigger or equal to 44×44 for two sets. The black line separates the classes that we chose (more than 20 samples) for our experiments.

Each sample was first padded with 2 pixels on each side (thus expanding their size to 48×48) to allow the convolutional layers to relevant features close to the border. The selected subset of samples was then expanded by applying random transformations until 500 samples per class were obtained. The random transformations applied are: (1) rotation sampled from $\mathcal{N}(0, 5°)$ and (2) translation of each border (that serves the purpose of both scaling and translation), sampled from $\mathcal{N}(0, 1px)$. In the end, every sample was preprocessed so that it has a mean of zero and unit variance.

Fig. 8: The selected 9 classes of traffic signs.

VI. Experiments and Results

Different architectures were built and evaluated for each dataset. In the following section the architectural specifics of convolutional neural networks for each experiment are defined, their performance is illustrated and results are discussed.

A. *MNIST*

The following architecture was built:

- **input layer** - 1 feature map 32×32
- **first convolutional layer** - 6 feature maps 28×28
- **first pooling layer** - 6 feature maps 14×14
- **second convolutional layer** - 16 feature maps 10×10
- **second pooling layer** - 16 feature maps 5×5
- **third convolutional layer** - 100 feature maps 1×1
- **hidden layer** - 80 neurons
- **output layer** - 10 neurons

The network was trained by stochastic gradient descent with 10 epochs of 60000 iterations each. No dropout and no random transformations were used. Such network yielded a result of 98.67% precision on the MNIST test set. Table I shows the obtained confusion matrix. The three most common mistakes are as shown in Fig. 9. It can be noticed that samples that were mistaken share a certain similarity with the wrongly predicted class.

		Predicted class									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Actual class	**0**	973	0	1	0	0	0	3	1	2	0
	1	0	1127	4	1	0	0	1	0	2	0
	2	3	0	1020	0	0	0	0	4	5	0
	3	0	0	2	992	0	6	0	3	5	2
	4	1	0	1	0	963	0	4	0	2	11
	5	1	0	0	3	0	884	1	1	0	2
	6	10	2	0	0	1	2	943	0	0	0
	7	0	1	7	2	0	0	0	1014	1	3
	8	2	0	1	0	1	1	1	3	962	3
	9	3	2	0	3	1	3	1	4	3	989

TABLE I: Confusion matrix for the MNIST dataset

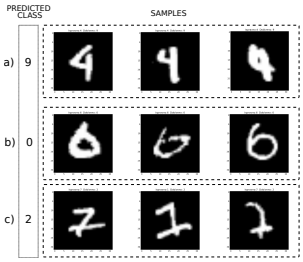


Fig. 9: The three most common errors on the MNIST dataset: a) number 4 classified as 9, b) number 6 classified as 0, c) number 7 classified as 2

B. FER-MASTIF TS2010

The following architecture was built:

- **input layer** - 3 feature maps 48×48
- **first convolutional layer** - 10 feature maps 42×42
- **first pooling layer** - 10 feature maps 21×21
- **second convolutional layer** - 15 feature maps 18×18
- **second pooling layer** - 15 feature maps 9×9
- **third convolutional layer** - 20 feature maps 6×6
- **third pooling layer** - 10 feature maps 3×3
- **fourth convolutional layer** - 40 feature maps 1×1
- **hidden layer** - 80 neurons
- **output layer** - 9 neurons

The network was trained with 20 epochs of 54000 iterations each by stochastic gradient descent. Random transformations were used (as defined in Section V) while dropout was not used. The trained network yielded a result of 98.22% on the test set. Table II shows the full confusion matrix on the test set and Fig. 10 shows the three most common errors made by the network.

		Predicted class								
		C02	A04	B32	A33	C11	B32	A05	B46	A03
Actual class	C02	100	0	0	0	0	0	0	0	0
	A04	0	99	0	0	0	0	1	0	0
	B32	0	0	97	3	0	0	0	0	0
	A33	0	0	0	100	0	0	0	0	0
	C11	0	0	0	0	100	0	0	0	0
	B31	0	0	0	0	0	100	0	0	0
	A05	0	0	0	0	0	0	98	0	2
	B46	0	0	0	0	0	0	0	100	0
	A03	0	3	0	0	0	0	7	0	90

TABLE II: Confusion matrix obtained on the FER-MASTIF TS2010 dataset

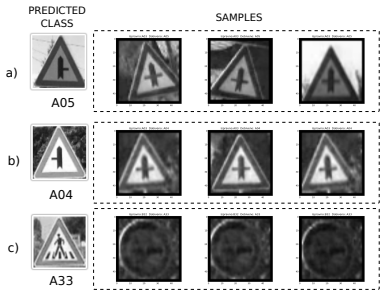


Fig. 10: The three most common errors on the FER-MASTIF TS2010 dataset a) A03 predicted as A05, b) A03 predicted as A04, c) B32 predicted as A33

C. FER-MASTIF TS2010 with smaller samples included

Fig. 11 shows the distribution of sample sizes (of the chosen 9 classes) in the FER-MASTIF TS2010 set. To determine the influence of sample size to the classification error, we scaled all samples to the same size (44×44 with padding leading to an input image of 48×48), matching the input of the network. In the previous experiment, samples smaller than 44×44 that were not used. However, in the following experiment they were included in the test set by upscaling to the required size.

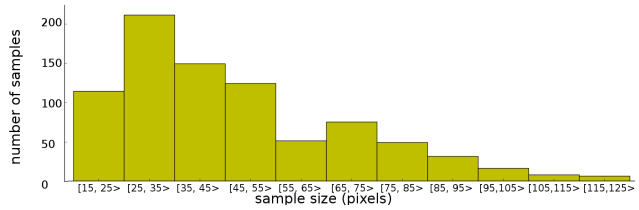


Fig. 11: Number of samples of different sizes (within the 9 chosen classes)

Fig. 12 shows the percentage of misclassified samples for different sizes. It can be seen that samples bigger than 45×45 produce significantly lower error rates than smaller samples, thus confirming our choice of the input dimensionality to the CNN.

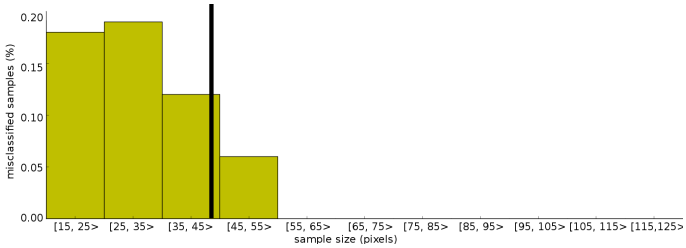


Fig. 12: Dependency of the error rate on the sample size. (The vertical line denotes the limit of 44px from where samples for the experiments in subsections A and B were taken.)

D. FER-MASTIF TS2010 with dropout

The same architecture was trained again with dropout in the final layers. We used the previously defined 9 classes and, again, only samples larger than 44×44. Using dropout improved the network generalization capability that yielded a result of 99.33% precision.

E. Comparison of learned kernels

Fig. 13 shows a few learned convolutional kernels from the first and second convolutional layers on the two different datasets. It is clearly visible that the network adapted to the dataset and learned different distinct features for each dataset.

VII. Common problems in training CNNs

CNNs are difficult to train because they have many hyper-parameters and learning procedure parameters, which need to be set correctly in order for learning to converge. In this section, we discuss a few issues that we had during the development of our library and the two networks.

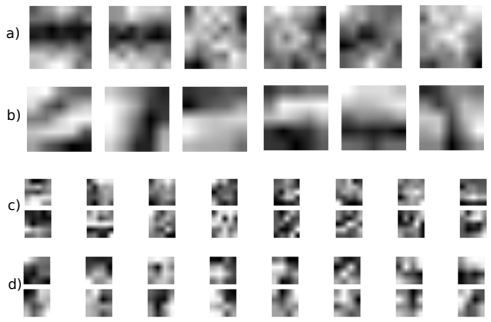


Fig. 13: Examples of learned kernels: a) first convolutional layer on MNIST b) first convolutional layer on FER-MASTIF TS2010 c) second convolutional layer on MNIST d) second convolutional layer on FER-MASTIF TS2010

Choosing architectural parameters like the number of layers and sizes of kernels and feature maps within each layer depend on the dataset. It makes sense to first choose the size of feature maps and kernels in the first layer according to the scale of the features in the dataset. Networks for dataset containing features of larger scale (like the FER-MASTIF TS2010 dataset) should use larger kernels and feature maps than networks that aim at classifying dataset with features of smaller scale (like the MNIST dataset). The same is valid for every other layer.

The **choice of an activation function** and number of neurons in each layer should match the **intervals for random initialization** for the weights and convolutional kernels. Also, it is suggested to preprocess the data to **zero mean and unit variance**.

CNNs are typically more difficult to train than MLPs, so when developing a new library it is recommended to **check the gradients** and **test for convergence** in all CNN layers. Like in classical NNs, whether a network will converge depends strongly on the **choice of the learning rate**. It is suggested to experiment with learning rates of different scales (*e.g.* 0.001, 0.01, 0.1, etc.) and to implement an adaptive algorithm that decreases the learning rate over time.

To improve the generalization capabilities of a CNN, it is suggested to use **dropout** or **sparse connectivity** between layers (dropout is a preferred method in modern state-of-the-art methods [3]) and including random transformations.

VIII. Conclusion and future work

We developed a library that enabled us to easily prototype and test different architectures of convolutional neural networks. After successfully testing the convergence of different elements of the library, we built a network similar to *LeNet-5* [1] and tested it on the MNIST dataset of handwritten digits where we obtained comparable results.

In the second part of this work, we analyzed and prepared the FER-MASTIF TS2010 dataset and built a convolutional neural network for recognizing a subset of 9 classes. Although limited to most numerous classes, the results were satisfactory.

Our library was developed having simplicity and clarity in mind, for educational purposes. It was not optimized for speed. Each training session lasted about a week. In case of building a bigger classifier for more classes of the FER-MASTIF TS2010 we suggest improving the speed of the convolutional operations and implementing mini-batch learning.

Regarding the FER-MASTIF TS2010 dataset, we suggest gathering more data. More data is suggested even for the 9 selected classes that had enough samples and yielded satisfactory results when used with random transformations, but especially for the remaining classes that didn't have enough samples to use (even when using random transformations to generate more samples).

References

[1] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, 1995.

[2] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits."

[3] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, 2012.

[4] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*.

[5] S. Šegvić, K. Brkić, Z. Kalafatić, and A. Pinz, "Exploiting temporal and spatial constraints in traffic sign detection from a moving vehicle," *Machine Vision and Applications*.

[6] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," *Neural Networks*, 2003.

[7] P. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis." in *ICDAR*, 2003.

[8] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[9] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *ICANN 2010*. Springer, 2010.

[10] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *ICPR*. IEEE, 2012, pp. 3288–3291.

[11] Y. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in vision algorithms," in *ICML*, 2010.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[13] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[14] I. Bonači, I. Kusalić, I. Kovaček, Z. Kalafatić, and S. Šegvić, "Addressing false alarms and localization inaccuracy in traffic sign detection and recognition," 2011.

[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[16] S. Šegvić, K. Brkić, Z. Kalafatić, V. Stanisavljević, M. Ševrović, D. Budimir, and I. Dadić, "A computer vision assisted geoinformation inventory for traffic infrastructure," *ITSC*, 2010.

[17] J. Krapac and S. Šegvić, "Weakly supervised object localization with large fisher vectors."

This research has been supported by the research project: Research Centre for Advanced Cooperative Systems (*EU FP7 #285939*).

Real Time Vehicle Trajectory Estimation on Multiple Lanes

Kristian Kovačić, Edouard Ivanjko and Hrvoje Gold
Department of Intelligent Transportation Systems
Faculty of Transport and Traffic Sciences
University of Zagreb
Email: kristian.kovacic@fpz.hr, edouard.ivanjko@fpz.hr, hrvoje.gold@fpz.hr

***Abstract*—Today's road traffic management systems using intelligent transportation systems solutions need real time measurements of various traffic parameters like flow, origin-destination matrices, vehicle type, etc. Cameras combined with image processing algorithms are being more and more used as the sensor capable to measure several traffic parameters. One such parameter, also important for accurate simulation of road traffic flow and evaluation of traffic safety, is the driving aggressiveness factor which can be estimated from the vehicles trajectory. In this paper an Extended Kalman Filter based approach to estimate vehicle trajectories on multiple lanes using only one static camera is described. To test the accuracy of the implemented approach a synthetic road traffic environment is developed. Real time capabilities of the approach are tested using real traffic video footage obtained from Croatian highways.**

I. Introduction

Today's traffic in urban areas is starting to cause heavy load to the existing road infrastructure. As road infrastructure in many cases cannot be modified (lack of build-up space), different approaches need to be taken in order to optimize traffic flow. Such approaches consist of applying intelligent transportation systems (ITS) which main goal is to apply a holistic approach for solving traffic problems using information and communication technologies. For optimal traffic control, ITS based systems need high quality traffic data in real time. Needed traffic data consists of various parameters such as traffic flow, distance between vehicles, velocity of vehicles, vehicle classification, etc. which all can be obtained from various sensors. Mostly used road sensors are inductive loops and nowadays video cameras also.

Video sensors or cameras combined with image processing algorithms are becoming an important approach to todays road traffic monitoring and control. From the obtained video footage high level traffic information can be extracted, i.e. incident detection, vehicle classification, origin-destination (OD) matrix estimation, etc. This information is crucial in advanced traffic management systems from the ITS domain. Commercial solutions for traffic monitoring by video cameras provide vehicle detection and tracking in scenes where there is a small amount of overlapping between the tracked vehicle and other objects (infrastructure or other vehicles). Additional drawback is that they need one camera per lane which is making such systems rather expensive. Proposed system described in this work has the main goal to achieve vehicle detection and tracking using only one camera for several lanes. Such a system can have a large number of ITS applications where it could be implemented for driver aggressiveness factor analysis, incident detection, traffic violation detection, etc. So, more high level traffic parameters measurements can be made enabling development of advanced autonomic or cooperative road traffic control approaches.

In today's society, where the vast majority of people drive on a daily basis in order to reach their destinations, aggressive driving has become a serious issue. Aggressive driving behaviors include speeding, driving too close to the car in front, not respecting traffic regulations, improper lane changing or weaving, etc. Obtaining such information in most cases is done by some kind of survey (eg. telephone survey) or by placing humans to monitor the traffic of a problematic area for a short amount of time [5]. Classic road sensors like inductive loops can not measure such driver behavior. Another approach of analyzing aggressiveness of driver behavior consists of using computer vision algorithms which process videos obtained from video cameras. By this approach, data can be obtained in real time with high accuracy [6]. Interesting data in this case is the vehicle trajectory on a road segment. By processing images from traffic video cameras, traffic violation can also be detected in the image as described in [7]. This system in combination with other ITS services can be useful for traffic law enforcement in cooperation with other agencies.

This paper is organized as follows: the second section describes the algorithm which performs vehicle detection and localization in the image. The third section describes the vehicle tracking algorithm which computes the vehicle trajectory. The fourth section describes optimizations made to the proposed system to ensure real time capabilities. The fifth section describes testing results of the proposed system. Paper ends with conclusion and future work description.

II. Vehicle detection

The first step in every vehicle detection algorithm beside importing of an image from a video stream is image preprocessing. After an image is imported it contains a certain percentage of noise. Noise complicates the vehicle detection process and significantly reduces the accuracy of the proposed system so it needs to be minimized. In [3], a Gaussian blur filter is used for noise reduction in a video footage. It reduces the number of details in the image including noise. In the proposed system a 5×5 matrix is used for the implemented Gaussian blur filter. Workflow of image preprocessing wherein renderings are distributed between the Central Processing Unit

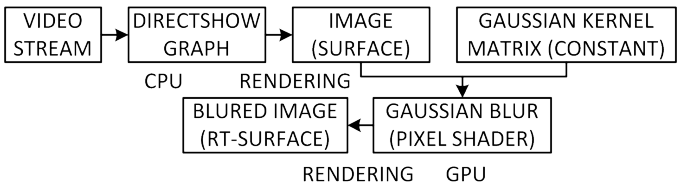


Fig. 1: Basic workflow of blur image preprocessing filter [1].

(CPU) and the Graphical Processing Unit (GPU) is given in Fig. 1.

After preprocessing of the imported image, various methods exist for vehicle detection. They can be divided into three types: optical flow methods; temporal difference methods; and background subtraction methods [9]. The system proposed in this work uses the background subtraction method. Workflow of the background subtraction method is shown in Fig. 2. Process consists of creating a background model of the scene and comparing computed background model with the latest preprocessed image imported from the video [2]. To create the background model the following equation is used:

$$BG_k = BG_{k-1} + \left\lfloor \frac{\sum_{i=1}^{n} sign(I_i - BG_{k-1})}{n} \right\rfloor, \qquad (1)$$

where BG_k represents the value of the specific pixel in the background model for the current frame and BG_{k-1} is the value of the specific pixel in the background model for the previous frame, I_i is the value of a certain pixel in i^{th} image, and n is the constant number of consecutive images stored in buffer ranging from the most recently acquired image k to the last image in the buffer $k - n + 1$. By comparing mentioned pixels in imported images, every pixel in the currently processed image can be classified. If the difference between the current image pixel value and the background model pixel value is larger than a specified threshold constant, the pixel is classified as a part of a foreground object. Otherwise it is considered as a part of the background model. The result of preprocessing and Fb/Bg image segmentation is given in Fig. 3.

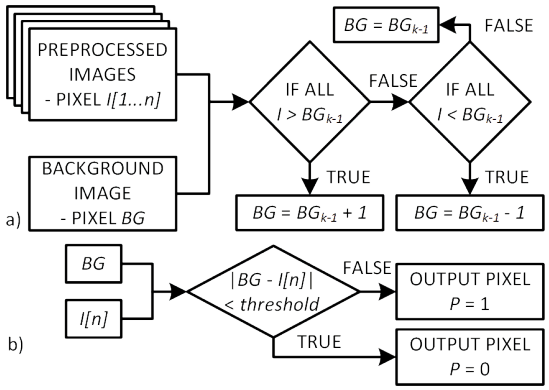


Fig. 2: Fg/Bg image segmentation workflow: a) background model creation, and b) background model and current image comparison [1].

Fig. 3: Original image (a) passed through preprocessing algorithm (b) and after Fg/Bg segmentation (c).

III. Trajectory estimation

When a moving vehicle is detected by the vehicle detection algorithm, its location in the image is obtained also. Detected vehicle location is given with (x, y) pixel coordinates and it contains information about the vehicle true location corrupted with noise. Noise disturbs vehicle tracking algorithm as measured vehicle location gets shifted for a certain value which is different for each image. This requires further processing of measured data.

The approach described in [4] uses data association and Kalman filtering for further processing of the object location. A data association algorithm is used to recognize the same object in series of consecutive images in order to track the respective vehicle or estimate its trajectory through time. As the measured object location contains noise, a Kalman filter is used to filter it. State model of the used Kalman filter is defined by object center (x, y) coordinates, area and velocity of an object. The system proposed in [8] uses genetic algorithms for data association and Kalman filter for trajectory estimation. Object detection is performed with background subtraction method based on mixture of Gaussian model [8].

The system proposed in this work processes object location using a modified data association algorithm mentioned in [4] and Extended Kalman Filter (EKF) for trajectory estimation. The first step in the data association algorithm is pixel clustering performed in the latest image obtained from the vehicle detection algorithm. Pixel clustering combines all adjacent pixels in the image into clusters. After all clusters are computed, they are compared with clusters from the previous image. If there is a positive match between two clusters in two consecutive images, both clusters are set to belong to the same object. If a cluster from the latest image has no match with any of the clusters in the previous image, it is considered to be a new object. If a cluster from the previous image has no match with any of the clusters in the latest image, it is considered that it has left the current scene. Matching criteria for cluster association in two consecutive images is given by the weights defined with the following equations:

$$w_{dist} = 1 - \frac{d - d_{min}}{d_{max} - d_{min}}, \qquad (2)$$

$$w_{area} = 1 - \frac{a - a_{min}}{a_{max} - a_{min}}, \qquad (3)$$

$$w_{cover} = \frac{a_{is}}{max(a_{obj}, a_{cl})}, \qquad (4)$$

$$w = \frac{w_{dist} + w_{area} + w_{cover}}{3}, \qquad (5)$$

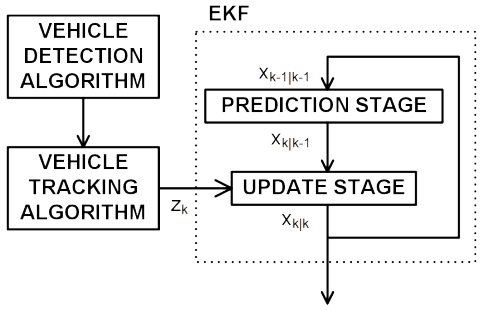


Fig. 4: Basic workflow of implemented EKF for vehicle trajectory estimation.

where d is distance between location of the specific cluster and estimated object location in pixels, d_{min} and d_{max} are minimum and maximum distance between all clusters and the processed object in pixels, a is difference between the cluster area (size) and the estimated object area, a_{min} and a_{max} are minimum and maximum difference between all clusters area and the estimated object area respectively, a_{is} is intersection area between cluster and object, a_{obj} is area of the object, and a_{cl} is the cluster area. All areas are expressed in pixels $[px]$.

To compute the distance between the location of a specific cluster and corresponding estimated object location their geometric centers are used. Cluster and object area are computed as their surrounding bounding box area. Matching gives a positive result only for cluster with the highest weight w and if $w_{cover} \geq \frac{2}{3}$.

EKF combines measured data with predicted state estimate. Result of this process can give more accurate trajectory then the one obtained by using measured data only. Basic workflow of the system is given in Fig. 4. The system first predicts state vector $x_{k|k-1}$ based on the state vector from the previous iteration performed by EKF in the update stage ($x_{k-1|k-1}$). Then the measurement obtained by the vehicle detection algorithm is combined with the latest state vector $x_{k|k-1}$ in the update stage. The obtained new state vector $x_{k|k}$ is used in the next iteration as input to the EKF ($x_{k-1|k-1}$). State vector can be defined with the following vector:

$$x = \begin{bmatrix} x_x \\ x_y \\ x_v \\ x_a \\ x_\phi \\ x_\omega \end{bmatrix}, \tag{6}$$

where x is state vector, x_x and x_y are vehicle x and y coordinates in the image in $[px]$, x_v is velocity in $[^{px}/_{s}]$, x_a is acceleration in $[^{px}/_{s^2}]$, x_ϕ is angle (direction) in $[rad]$ and x_ω is angular velocity of vehicle in 2D camera perspective in $[^{rad}/_{s}]$.

Measurement vector z can be defined with the following equation:

$$z = \begin{bmatrix} z_x \\ z_y \end{bmatrix}, \tag{7}$$

where z_x and z_y represent x and y coordinates of the vehicle in the image in $[px]$ obtained by vehicle detection algorithm.

Computation in the prediction stage is done by the following equations:

$$f(x) = \begin{bmatrix} x_x + x_v t \cos(x_\phi) + \frac{x_a[x_\omega t \sin(x_\omega t + x_\phi) + \cos(x_\omega t + x_\phi)]}{x_\omega^2} \\ x_y + x_v t \sin(x_\phi) - \frac{x_a[x_\omega t \cos(x_\omega t + x_\phi) - \sin(x_\omega t + x_\phi)]}{x_\omega^2} \\ x_v + x_a t \\ x_a \\ x_\phi + x_\omega t \\ x_\omega \end{bmatrix}, \tag{8}$$

$$x_{k|k-1} = f(x_{k-1|k-1}), \tag{9}$$

where $x_{k|k-1}$ is state vector and $x_{k-1|k-1}$ is state vector from previous iteration $k-1$ computed in update stage and t is interval (distance) between iteration k and $k-1$ expressed in the number of frames.

After the prediction stage is done, the predicted state vector is updated with the previous state $x_{k|k-1}$ and the latest measurements vector z_k. Computation of the new state vector $x_{k|k}$ is done using the following equations:

$$h(x) = \begin{bmatrix} x_x \\ x_y \end{bmatrix}, \tag{10}$$

$$y_k = z_k - h(x_{k|k-1}), \tag{11}$$

$$F_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{x_{k-1|k-1}}, \tag{12}$$

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{x_{k|k-1}}, \tag{13}$$

$$P_{k|k-1} = F_{k-1}\ P_{k-1|k-1}\ F_{k-1}^T + Q_{k-1}, \tag{14}$$

$$S_k = H_k\ P_{k|k-1}\ H_k^T + R_k, \tag{15}$$

$$W_k = P_{k|k-1}\ H_k^T\ S_k^{-1}, \tag{16}$$

$$x_{k|k} = x_{k|k-1} + W_k\ y_k, \tag{17}$$

$$P_{k|k} = (I - W_k\ H_k)\ P_{k|k-1}, \tag{18}$$

where y_k is innovation vector, $P_{k|k-1}$ is the covariance matrix of the predicted state estimate, F_{k-1} is the state transition matrix, $P_{k-1|k-1}$ is the covariance matrix of the predicted state estimate from the previous iteration, Q_{k-1} is the covariance

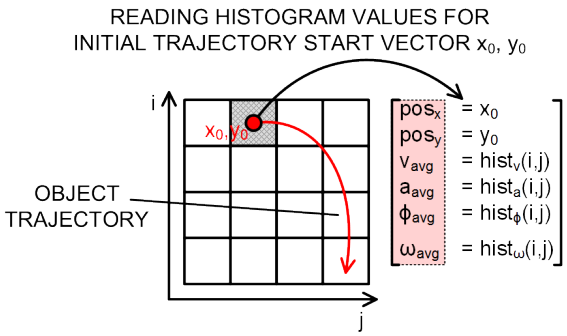


Fig. 5: Setting initial state values of EKF by using histograms.

Approach		Vehicle count		
		Total	Lane	
			Left	Right
Overlap check	Hits	126	65	61
	FP / FN	0/6	0/5	0/1
	Accuracy	95.6%	92.9%	98.4%
Trajectory check	Hits	129	68	61
	FP / FN	1/4	0/3	1/1
	Accuracy	96.2%	95.8%	96.8%
True vehicle count		132	70	62

Table 1: Counting results of the proposed system.

matrix of process noise, S_k is the innovation covariance matrix, H_k is the observation matrix, R_k is the covariance matrix of the observation noise, W_k is the Kalman gain matrix and I is identity matrix.

An important feature of the EKF is that before the first iteration can be computed, state vector $x_{k-1|k-1}$ and matrix $P_{k-1|k-1}$ need to be initialized. In the proposed system the initial values of vector $x_{k-1|k-1}$ are estimated by a histogram. Histogram is divided into $i \times j$ segments where each segment covers specific rectangular area of the image. Histogram is updated in every iteration of the EKF, where computed v, a, ϕ, ω components of a state vector $x_{k|k}$ from the EKF are added to the vector in the corresponding histogram segment. Histogram segments are determined by reading values of x and y components of a state vector $x_{k-1|k-1}$. Every component of a vector in the histogram segment represents the sum of all values in all previous iterations. If this sum is divided by the number of elements which were used in the sum, mean value can be obtained. In the first iteration of the EKF, values of x and y components of a state vector $x_{k-1|k-1}$ are set to values obtained directly from the vehicle detection algorithm and therefore they are not processed by the EKF. Other components of a state vector $x_{k-1|k-1}$ are set to mean values read from histogram as shown in the Fig. 5. After initialization, for every further EKF iteration, the state vector $x_{k-1|k-1}$ is computed only by the EKF (histogram values are ignored).

IV. Parallelization based speed-up

The first version of the implemented approach for vehicle detection has shown to be efficient from accuracy aspect according to the results given in Tab. 1. To ensure real time capabilities further development with aspect of using parallel computing abilities of today's CPU and GPU architecture has been done. Basic workflow of the proposed application which uses multi-threading (MT) and GPU support is shown in Fig. 6. Algorithms that process every pixel in the image can be time consuming for CPU even with use of Streaming SIMD Extensions (SSE) instructions support. Modern GPU architecture consists of many stream processors that can process data in parallel execution (SIMD instructions). This represents main reason for considering use of GPU in further development of current application. In the currently proposed system, image preprocessing and vehicle detection algorithm are entirely performed on GPU through pixel shaders. Pixel clustering is performed on CPU with MT support which improves performance of algorithm regarding execution time.

V. Results

The proposed system has been tested using real world road traffic video footage captured on a highway with two lanes near the city of Zagreb in Croatia. Camera was mounted above the highway and passing vehicles were recorded using a top view camera perspective as given in Fig. 7. Duration of the test video is $10\,[min]$. Obtained original video resolution is $1920 \times 1080\,[px]$ (RGB).

For vehicle detection results verification, two approaches for vehicle counting were tested. Both are based on markers (virtual vehicle detectors). Yellow and red rectangle markers are placed in the bottom part of the scene on each lane as shown in Fig. 7. Edges of markers are perpendicular to the image x and y axis. When a vehicle passes through a marker and a hit is detected, the counter for that marker is incremented. The first approach checks if an object is passing through a marker with its trajectory and the second approach performs check if an intersection between a marker and an object exists. Both approaches discard all objects whose trajectory direction is outside of a specific interval. In the performed test, all objects need to have their direction between $90 - 270\,[^\circ]$ in order not to be discarded. Objects also need to be on the scene for more than 30 frames. The value of the threshold constant used in Fg/Bg segmentation method is 10 and the number of consecutive images used when creating background model (n) is 105. Blue lines in Fig. 7 represent computed vehicle trajectory. Experimental results are given in Tab. 1. FP represents false positive and FN represents false negative hits. True vehicle count is acquired by manually counting all passed vehicles.

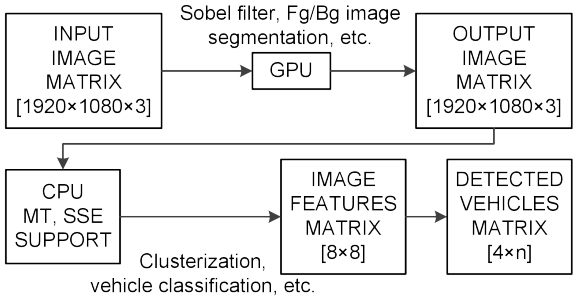


Fig. 6: Proposed workflow based on CPU/GPU computation distribution.

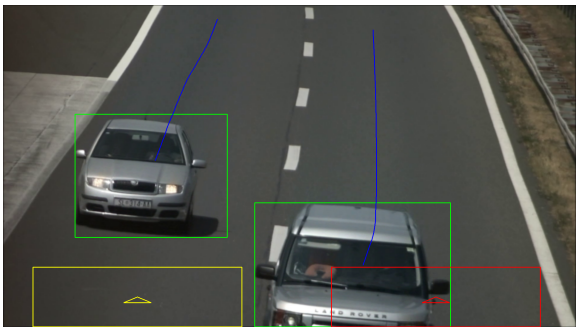


Fig. 7: Vehicle tracking and counting on two lanes.

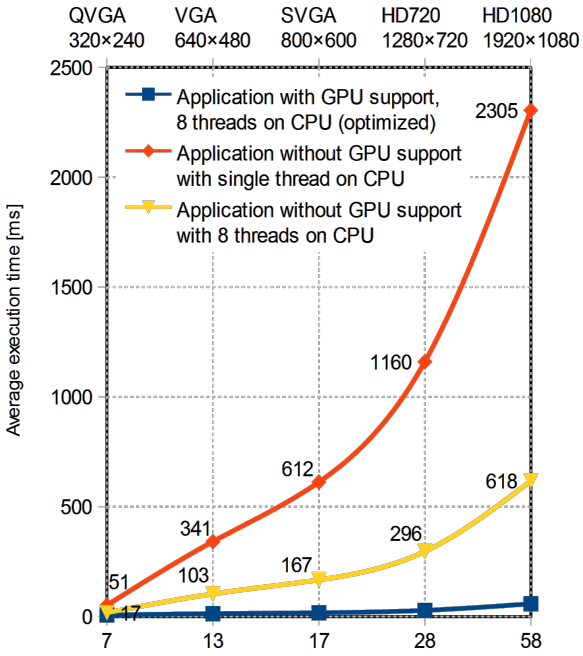


Fig. 8: Execution time of the proposed system.

In Fig. 8 execution time is given for various resolutions tested on a Windows 7 (64bit) computer with CPU Intel Core i7 - 2,4 GHz, GPU NVIDIA Quadro K1000M and 8 GB RAM. In the experimental testing, both approaches (overlap and trajectory check) for vehicle counting had the same execution time. From the acquired results it can be concluded that real time vehicle detection can be performed on SVGA $800 \times 600\,[px]$ resolution and lower using a standard PC computer. On SVGA resolution, $17\,[ms]$ is required to process a single frame. This enables maximum frame rate of $58\,[fps]$. At QVGA $320 \times 240\,[px]$ resolution, $142\,[fps]$ can be achieved with $7\,[ms]$ required to process a single frame. It can also be concluded that the approach with trajectory check gives better results regarding accuracy than the approach with overlap check.

For testing of the implemented EKF based vehicle trajectory estimation, a synthetic road traffic video was made in Autodesk 3ds Max. Video of synthetic environment simulates passing of one vehicle on a road with two lanes. As the true position of a vehicle in the synthetic environment is known, the implemented EKF can be tested for its trajectory estimation

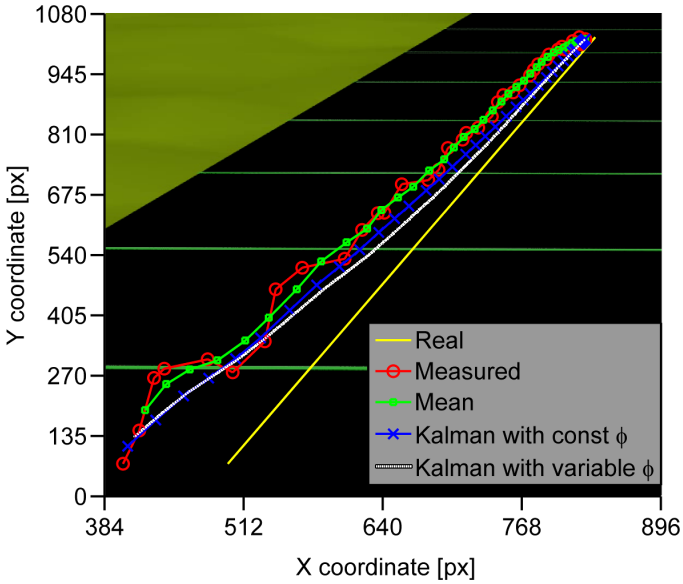


Fig. 9: Comparison of real, measured, mean and EKF vehicle trajectory.

accuracy. In Fig. 9 different trajectories obtained by various methods are compared. The real trajectory represents movement of vehicle geometric center defined during development of the synthetic video. The measured trajectory is computed by taking data (vehicle trajectory) from the vehicle detection algorithm and adding noise to it in order to simulate values which would be obtained by processing real world road traffic video. Noise is defined by standard uniform distribution in the interval $[-2.5, 2.5]$ and it is added to each vector of the vehicle trajectory. The mean trajectory is computed by taking the last 3 x and y coordinates of the measured trajectory and computing the mean value of them. So measurement noise can be reduced without significantly affecting vehicle location estimation accuracy. EKF trajectories are obtained by using two different state models. The first model is already described in the previous section. The second model is based on the first model with the angular velocity component removed.

In Fig. 10, x-axis represents number of frame for which error is computed and y-axis represents amount of error in

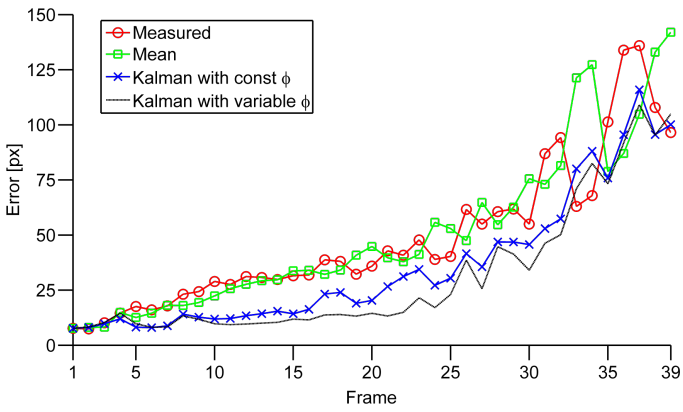


Fig. 10: Comparison of error in measured, mean and EKF vehicle trajectories.

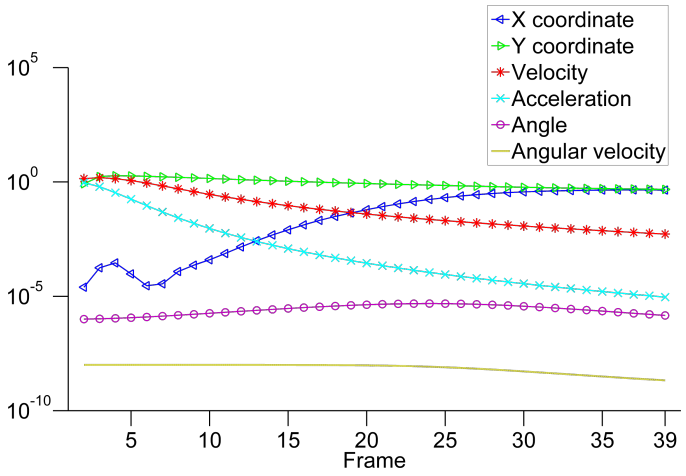


Fig. 11: Change of covariance matrix $P_{k|k}$ components.

$[px]$. Vehicle position error can be computed for any frame using the following equation:

$$err(k) = \sqrt{\left(x_r^{(k)} - x_f^{(k)}\right)^2 + \left(y_r^{(k)} - y_f^{(k)}\right)^2}, \qquad (19)$$

where k is the frame number, $x_r^{(k)}$ and $y_r^{(k)}$ are real measured values in $[px]$ of x and y coordinates for specific trajectory vector, $x_f^{(k)}$ and $x_f^{(k)}$ are filtered values in $[px]$ (using mean value method or EKF) of x and y coordinates for specific trajectory vector. Mean error value for the measured trajectory is $48.4\,[px]$, for trajectory obtained by mean method is $50.2\,[px]$, for trajectory obtained by EKF with constant vehicle direction angle ϕ is $35.9\,[px]$ and EKF with variable vehicle direction angle ϕ is $31.2\,[px]$.

Covariance matrix $P_{k|k}$ changes through 39 frames as shown in Fig. 11, where blue and green lines are x and y coordinates, red line is velocity component, cyan is acceleration, purple is angle and yellow is angular velocity component of the uncertainty matrix $P_{k|k}$. From the Fig. 11 it can be concluded that estimate covariances for y coordinate, angle and angular velocity have no rapid change in their values over whole vehicle trajectory. Opposite to that estimate covariances for velocity and acceleration decrease rapidly over time. Estimated covariance for x coordinate increases rapidly till it approximately reaches the value of estimate covariance for y coordinate. Vehicle velocities and acceleration can be estimated more accurately than the (x, y) vehicle coordinates. This can be explained that tracked vehicle moves in the video and enlarges during tracking. So, location measurements are more accurate when the vehicle is detected (vehicle enters the scene) than when it leaves the scene.

In Fig. 7 estimated trajectories of several vehicles on multiple lanes can be seen. The presented processed traffic scene proofs that the implemented system can successfully simultaneously detect and track vehicles on multiple lanes in real time.

VI. Conclusion

In this paper a system for vehicle detection and tracking on multiple lanes based on computer vision is proposed. The developed system uses only one camera to detect and track vehicles on multiple lanes. It solves drawbacks of the currently available commercial systems that use one camera per road lane. The first vehicle detection results are promising with an accuracy of over 95%.

Additionally, vehicle trajectory estimation has been added to the existing system. Because the trajectory of a vehicle contains a large ratio of noise, trajectory is filtered by EKF. For testing of the implemented EKF filter, synthetic environment was developed in which groundtruth vehicle trajectory is known. From the testing results in which the groundtruth data is compared with the computed data, it can be concluded that EKF can improve trajectory estimation accuracy. As the proposed system is computationally expensive it was optimized by implementing ability to execute specific image processing algorithms (preprocessing, Fg/Bg image segmentation) on GPU. The algorithm for pixel clustering which was too complex to execute on GPU was optimized by implementing CPU MT support.

Future work consists of developing a tracking system which would be able to perform license plate recognition and vehicle type classification of detected vehicles. So additional data can be obtained from which further analysis of the road traffic video footage could be made.

Acknowledgment

This work has been supported by the IPA2007/HR/16IPO/001-040514 project “VISTA - Computer Vision Innovations for Safe Traffic” which is co-financed by the European Union from the European Regional and Development Fund and by the EU COST action TU1102 - “Towards Autonomic Road Transport Support Systems”. Authors wish to thank Dominik Kovačić for developing the synthetic road traffic environment.

References

[1] K. Kovačić, E. Ivanjko, H. Gold, Real time vehicle detection and tracking on multiple lanes, WSCG, 2014, Czech Republic

[2] K. Kovačić, E. Ivanjko, S. Varela, Real time vehicle country of origin classification based on computer vision, ISEP, 2014, Slovenia

[3] V. Braut, M. Čuljak, V. Vukotić, S. Šegvić, M. Ševrović, H. Gold, Estimating OD matrices at intersections in airborne video - a pilot study, MIPRO, 2012, Croatia

[4] J. M. Jeong, T. S. Yoon, J. B. Park, The specific object tracking algorithm based on Kalman filter in an environment where similar objects are existing, ICCAS, 2013, Korea

[5] E. Wells-Parker, J. Ceminsky, V. Hallberg, R. W. Snow, G. Dunaway, S. Guiling, M. Williams, B. Anderson, An exploratory study of the relationship between road rage and crash experience in a representative sample of US drivers, Accident analysis and prevention, vol. 34, Social Science Research Center, Mississippi State University, 2002, USA

[6] H. B. Kang, Various approaches for driver and driving behavior monitoring: a review, ICCV Workshops, 2013

[7] Y. C. Chung, J. M. Wang, S. W. Chen, A vision based traffic light detection system at intersections, Journal of Taiwan Normal University: Mathematics, Science & Technology, 2002, China

[8] H. Wang, M. W. Ren, J. Y. Yang, Object tracking based on genetic algorithm and Kalman filter, CIS, 2008, vol. 1, pp. 80-85, China

[9] Y. Xia, S. Ning, H. Shen, Moving targets detection algorithm based on background subtraction and frames subtraction, ICIMA, 2010, vol. 1, pp. 122-125, China

Warning and Prohibitory Traffic Sign Detection based on Edge Orientation Gradients

Darko Jurić and Sven Lončarić
University of Zagreb, Faculty of Electrical Engineering and Computing,
Department of Electronic Systems and Information Processing
Unska 3, 10000 Zagreb, Croatia
Email: {darko.juric, sven.loncaric}@fer.hr

***Abstract*—A method for traffic sign shape detection is presented, which does not require extensive training and is based on fast template matching suitable for real-time applications. The proposed method is able to detect prohibitory and warning signs. Post processing phase enables detection verification thus drastically increasing detection accuracy. All method parameters, if required, can be tweaked without the need for time-consuming retraining procedures. The method shows robustness to various conditions that often impose problems for color based systems. The proposed method can be used standalone or with already developed methods as a verification layer.**

***Keywords*—computer vision, traffic sign detection, template matching**

I. Introduction

Traffic sign detection and recognition is an important problem in advanced driver assistance systems (ADAS). ADAS utilize sophisticated algorithms, which not only address traffic sign detection and recognition but other applications including lane detection, driver distraction detection, and parking assistance. Poor visibility and/or driver fatigue can be one of the main reasons that can lead to traffic accidents. While traffic sign detection systems are commercially available, they still present an open research topic. In recent years, speed limit detection systems have been included in top-of-the-line vehicle models from various manufactures, but these systems are still limited to a subset of all traffic signs. Therefore, there is a need for traffic sign detection and recognition methodology, which is capable of detecting a wider selection of traffic signs. Nowadays, embedded devices not only contain basic FPGA units, which can be designed to support vectorized array addition, but CPU support as well thus enabling much broader range of usage. The important feature is ability to tweak detection parameters without additional training. Many methods that are considered state-of-the-art do not have such possibility like [1] and [2]. The paper is organized as follows. A short overview of related work is presented in Sec. II. Sec. III describes the proposed method. Sec. IV gives used parameters and results, which are obtained on videos that contain challenging lightening conditions and extensive clutter. Conclusion is given in Sec. V.

II. Related work

A. Color-based techniques

Some parts of the method presented in [3] are based on simple thresholding to distinguish a particular color (red) from others. However, various factors such as color change due to illumination factor (e.g. time of a day) or non-uniform illumination that produces shadows need more sophisticated methods like [4] in which a light source color is estimated. In [5] features are extracted after image segmentation process which is done by using Gabor filters, K-means clustering and smoothing filters. In [6] CIECAM97 color model is used where only hue and chroma channels are used. Various daylight viewing cases are considered. A general drawback of color based techniques is that they require special measures to achieve luminance invariance.

B. Shape-based techniques

The most common technique for shape-based approaches is Hough transform. A method that uses a modified Hough transform to detect circular, triangle and square shaped signs is presented in [7]. Distance transform (DT) based techniques are another popular approach for traffic sign detection. In [8], a modified DT technique is used. First, edge orientations are extracted from the image, then a DT image is built for every edge orientation. A score match is obtained in a way that a template is rotated and scaled and matched against every DT image. The maximum score is taken as the score match. Hough transform is more constrained regarding shape diversities – it can approximate only simple shapes and traditional template searching can be time consuming.

C. Machine learning based techniques

The most common used technique is based on work of Viola and Jones presented in [9]. Although it was originally developed for face-detection it can be successfully used for detection of other objects, including traffic signs. In [2], a traffic sign detection and recognition method is presented, which uses the mentioned technique. Although Viola-Jones approach is widely used it suffers from the following problems. A large amount of training samples is needed to train the detector. For some problems this imposes a significant obstacle. By introducing more object classes several detectors must be used in order to classify them. This can impose performance penalty.

III. Proposed method

In this paper, we present a novel method for traffic sign detection, which is based on fast edge orientation template matching [10]. The proposed method is robust to clutter and is computationally efficient and suitable for real-time

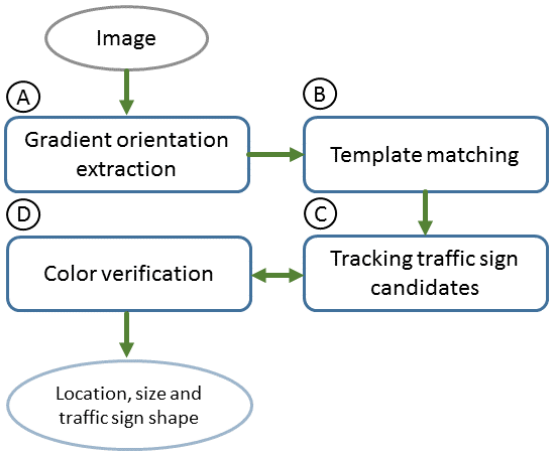


Fig. 1: A method composed of template matching as the core part and post processing stages which include shape verification as well as continuous tracking and detection consistency verification.

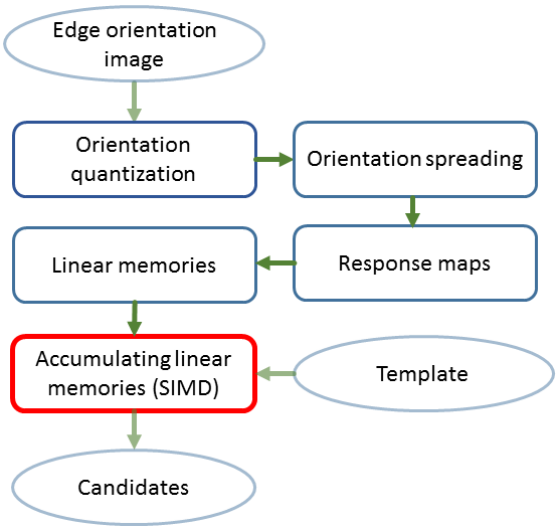


Fig. 2: Template matching pipeline. The red color highlights the modified part.

applications. The outline of the proposed method is shown in Fig. 1. The template matching algorithm [10] is modified in this paper in order to improve its accuracy by using a different gradient extraction method as described in Sec. III-A and by extending the maximum number of features per template, described in Sec. III-B1. However, the modified template matching procedure is still not sufficient for successful traffics sign detection due to higher false positive rate which is a result of an extensive clutter (e.g. forest along the road).

The post processing phase consists of two steps. The first step involves continuous traffic sign detection and tracking, described in Sec. III-C which reduces false positives considerably. The second phase eliminates structures that have similar shapes but do not have similar border color - validates traffic sign border-color (redness) described in Sec. III-D.

A. *Gradient orientation extraction*

The crucial preprocessing step is edge extraction, where an edge orientation image is an input image for the template matching method [10]. One of the main reasons that image orientations are chosen is their robustness to extensive light changes and image noise. Gradient magnitude performs poorly in scenes with background clutter due to many false positives. In [10], a multi-channel Sobel operator has been used where a pixel orientation is taken from the channel that has the largest magnitude. In addition, orientations are filtered in a way that only dominant orientations in a 3x3 neighborhood are retained. Instead of the multichannel Sobel operator and orientation filtering a standard grayscale Canny implementation is used in the proposed method. By using this operator better results are obtained in terms of a reduced number of false positives in highly cluttered images (e.g. forest as background).

B. *Template matching*

After the gradient orientation extraction presented in Sec. III-A is performed, the modified template matching algorithm is used in order to detect circular and triangular shapes - traffic sign candidates. The outline of the template matching procedure is shown in Fig. 2. Only relevant parts needed to understand the modifications are going to be briefly discussed. For full explanation of the procedure see [10].

The template matching procedure uses quantized edge orientations as features where each quantized orientation is encoded in 8-bit vector as a single bit flag. The template is matched against every location within the image using the operation equivalent to sliding a template over the input image. In contrast to the standard template matching procedure the input image is preprocessed so that the matching procedure can be executed very fast by adding long arrays - linear memories [10]. Each linear memory cell contains similarity in range $[0..n]$, where n is maximum user-specified similarity between two quantized orientations These values are limited by the number of bits in 8-bit value. The linear memory addition can be done very efficiently by using SIMD processor architecture. The addition process is done as follows.

Each template feature, see Fig. 5, which consists of a location and its corresponding quantized orientation is used to select linear memory (8-bit vector). The selected memories are added to the 8-bit result vector initialized by zeroes. The result vector contains the raw template matching scores in range [0..255] for each location. The values are rescaled to the range [0..100]. The result vector is thresholded in order to extract only strong matches. For details about the linear memory content and creation see [10].

1) Extending number of features: The 8-bit result vector dictates the maximum number of features per template because each linear cell contain value which maximum is n and and the number of linear memories being added corresponds to the number of template features. See Fig. 3 for details. Therefore the maximum number of features per template, in the original paper, is limited to $\lfloor 255/n \rfloor$. This number of features has not shown to be enough to robustly detect triangular and circular shapes in strong clutter such as forest. To overcome this limitation the linear maps are first added

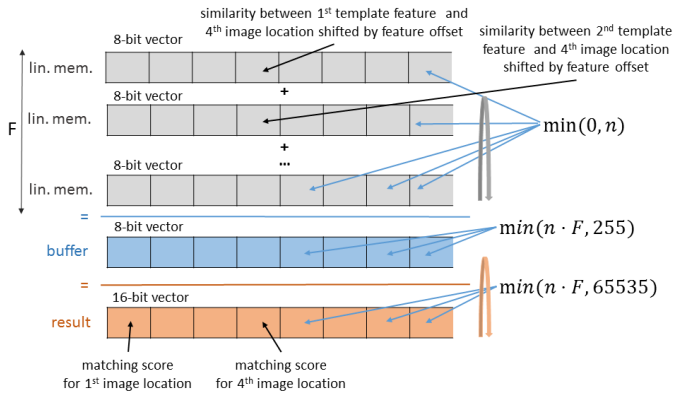


Fig. 3: The proposed procedure for linear memory addition - template matching. Linear memories are added to the buffer which is periodically added to the final result vector. n represents the maximum orientation similarity, and F is the number of features of a template.

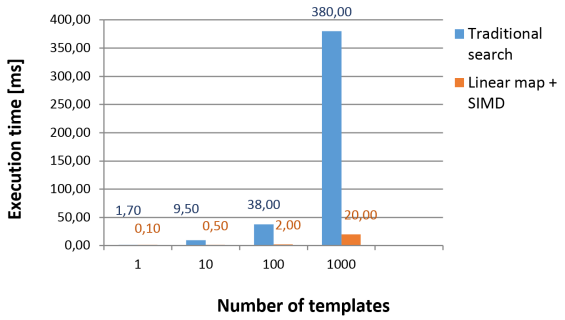


Fig. 4: Traditional and speeded-up template matching performance comparison. The number of features per template is 100. All tests are conducted on Intel Core i5-2500K CPU using single core.

to a temporary buffer - 8 bit vector, and then before the buffer overflows, the buffer is added to the final result array (16-bit), which contains template matching scores. See Fig. 3 for details. This procedure is repeated for each template feature. The buffer is cleared after its values are accumulated to result array. The result is the increased maximum number of features $\lfloor 65535/n \rfloor$. Fig. 4 shows the significant speed-up over traditional template matching approach.

2) Building traffic sign templates: Feature extraction can be performed directly on color images, but better results are obtained if a binarized image is used, which is first smoothed. The number of extracted features is fixed (it does not depend on an object size). At far distances Canny edge detector usually cannot detect outer and inner circle or triangle, which are the integral parts of the warning and prohibitory signs. Therefore, to detect those traffic sign candidates simple circular and triangle shapes are used - see Fig. 5. A scale and rotation transforms are applied to those shape images in order to build templates of various sizes.

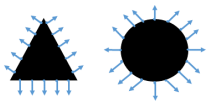


Fig. 5: The samples of warning and prohibitory sign template. Features (quantized gradient orientations) are superposed. In practice 100 features per template is used.

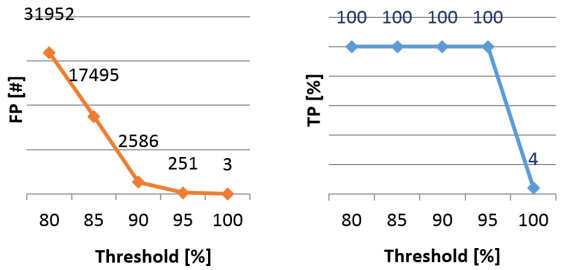


Fig. 6: The performance of the template matching procedure. The left graph shows the number of false positives in relation to the threshold - the minimum percentage of matched template features. The right graph represents the true positive percentage in relation to the threshold.

C. Object tracking

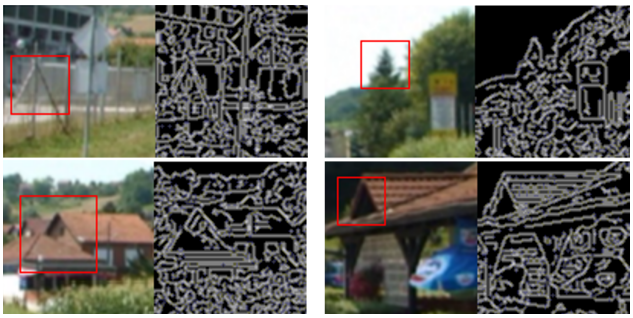
Object tracking is achieved using Kanade-Lucas tracker [11]. Some features due to extensive noise become invalid in a sense they re no longer matched correctly. This problem manifests in inaccurate tracked point speed and orientation. Those features are filtered out by using their magnitude and orientation as parameters as follows: mean and variance are calculated. All features which magnitude and orientation variance are within $\pm\sigma$ are taken. Another difficulty is that objects that are supposed to be tracked are moving with the same speed as well as background so the additional filtering is hard to apply. The minimum number of detections per trajectory is set in advance. This phase greatly decreases the number of false positives. The second stage provides additional filtering using the border-redness feature Sec. III-D. Note that using temporal constraint a traffic sign may not be detected at a long range because those types of verifications are carried over several video frames.

D. Color verification

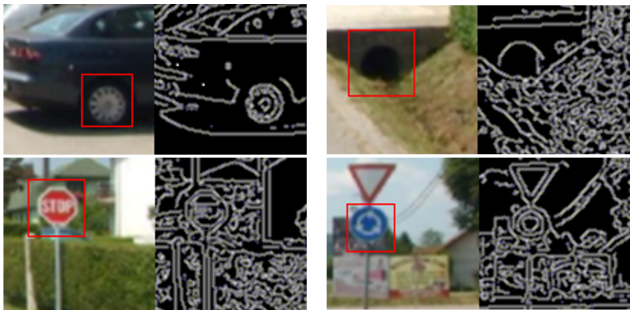
As shown in Fig. 7 false positives are still present after enforcing temporal constraint. The proposed verification step takes advantage of the template representation and traffic sign border redness - see Fig. 8 for details. The redness measure is computed as:

$$I_r = I_R - I_G \tag{1}$$

where I_r is redness image, and I_R and I_G are red and green image channels. Extracting just red image channel is not enough because many green objects have high red component, therefore the Eq. 1 is used to calculate image redness. After the initial match the best matched template is taken and normals are constructed at the feature locations - see Fig. 8. The length



(a)



(b)

Fig. 7: Samples of false positive detections that are remained after temporal consistency enforcement. a) Warning sign false positives b) Prohibitory sign false positives.

of the normal vector is proportionally resized according to the template scale as shown in Fig. 9. As the matched templates sometimes do not fit to the object's edge, the worst case is taken, where the matched templates are partly matched with both sign borders as shown in Fig. 10.

Therefore a length of normals positioned in each template feature position are increased in both directions. Pixels values are taken along normals where the bilinear interpolation is used. The obtained pixel values are smoothed in a way that pixel values are clustered by kNN [12] algorithm in two classes and a median operation is applied to the clustered array in order to obtain final class indices. The pre-processed array can have two minima corresponding to the inner and outer sign border or just one minima corresponding to the outer sign border due to low picture quality and small size where the redness spreads into traffic sign interior. There should be at least one transition from maxima to minima, jump, in order to classify the normal as valid. This procedure is taken for every normal as shown in Fig. 9. At the end valid lines are counted and compared against threshold.

IV. Experimental results

Validation of the proposed method has been conducted using the following parameter values. The number of features per template is set to 100, the neighbourhood size is 5 pixels and the maximum similarity between angles is set to 4. The number of quantized orientations (1st and 2nd quadrant) is 8. There are just two template prototypes: filled circle and filled triangle that represent the prohibitory and the warning

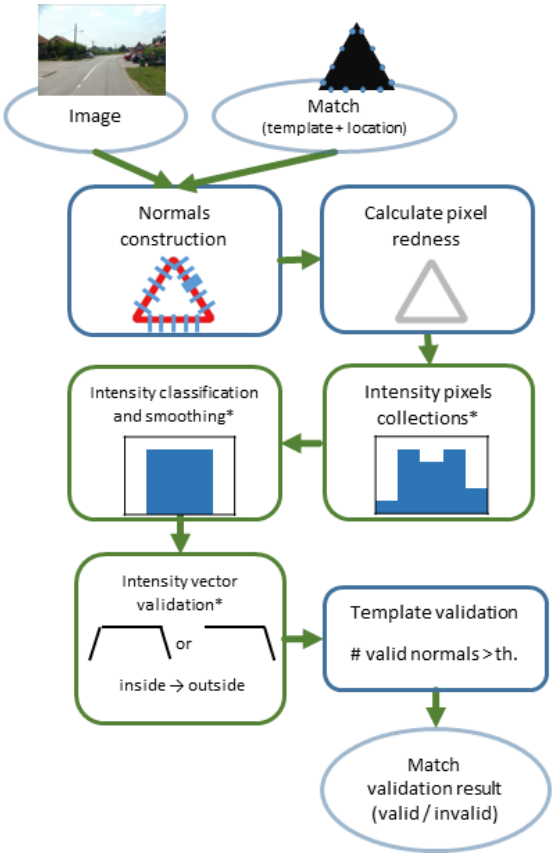


Fig. 8: Flow diagram of the color verification method. Operations executed on each normal are marked with $*$.

(a) (b)

Fig. 9: Color (redness) verification of a warning sign and a circular shape. Each image group shows original image, detected shape with superposed normals and image redness. Valid normals are marked with blue, red otherwise.

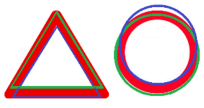


Fig. 10: A best template in a match area partly matches outside and inside circle in prohibitory sign or triangle is partly-matched in warning signs.

	Video 0	Video 1	Video 2	Video 3	Video 4	Video 5
Area (urban / rural)	rural	rural	urban / rural	urban / rural	rural	urban / rural
Duration (hh:mm:ss)	00:01:42	00:12:40	0:11:06	00:14:34	00:07:22	00:02:13
Prohibitory #	1	14	6	9	1	22
Warning #	3	12	17	17	5	2

TABLE I: Video sequences used for traffic sign detection and the number of signs in each sequence. $Video\ 0$ is used for parameter settings.

signs. The original size of a template is 120x120. They are scaled until their size does not drop below 20x20 and rotated ± 5 degrees. There are 216 binarized templates (108 circles + 108 triangles), which are smoothed with Gaussian of σ value 3. The minimum number of matched features is set to 95% meaning that in this case 95 features must be correctly matched to mark the location as a candidate. The maximum number of concurrent verifications is 4. Each tracker can use up to 15 KL features. If there are less than 2 features an object is considered lost. There should be at least one detection in 3 frames and at least one positive verification. The minimum number of valid normal vectors in shape verification is 60 (60 % as 100 features per template is used).

A single video has been used for parameter selection during the training process. Table I shows information about the video sequences used in the experimental validation. The proposed method is evaluated on five very challenging videos with the length of more than 45 minutes in total. By using only template matching algorithm with a fixed threshold of 95 a very large number of false positive detections is obtained as shown in Fig. 7 and Fig. 6. The most important parameter is the template matching threshold, which is set to relatively high due to extensive background clutter in non-urban scenes. The performance of a detection without and with the post processing stage is shown in Table II.

V. Conclusion

This paper presents a new method for detection of prohibitory and warning signs. The experimental validation has shown that the proposed traffic sign detection procedure can detect traffic signs in various conditions. By enforcing temporal consistency the number of false positives is significantly reduced, but without reducing the true positive rate. The color verification method in post-processing stage exploits pre-built templates and image redness as described thus reducing false positive rate even further. Although detection of only two traffic sign types is addressed the proposed method could be extended to detect other traffic signs by incorporating other templates into the existing set. In addition the presented method can be combined with machine-learning techniques to increase detection speed (e.g. fast template matching + Viola-Jones) or to efficiently reduce false positives (e.g. Viola-Jones + fast template matching).

Acknowledgment

This research has been partially supported by the European Union from the European Regional Development Fund by the project IPA2007/HR/16IPO/001-040514 VISTA - Computer Vision Innovations for Safe Traffic.

References

[1] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. López-Ferreras, "Road-sign detection and recognition based on support vector machines," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 2, pp. 264–278, 2007.

[2] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler, "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," in *Intelligent Vehicles Symposium, 2005. Proceedings.* IEEE, 2005, pp. 255–260.

[3] S. Varan, S. Singh, R. Sanjeev Kunte, S. Sudhaker, and B. Philip, "A road traffic signal recognition system based on template matching employing tree classifier," in *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, vol. 3. IEEE, 2007, pp. 360–365.

[4] M. Bénallal and J. Meunier, "Real-time color segmentation of road signs," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, vol. 3. IEEE, 2003, pp. 1823–1826.

[5] J. F. Khan, S. M. Bhuiyan, and R. R. Adhami, "Image segmentation and shape analysis for road-sign detection," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 83–96, 2011.

[6] X. W. Gao, L. Podladchikova, D. Shaposhnikov, K. Hong, and N. Shevtsova, "Recognition of traffic signs based on their colour and shape features extracted using human vision models," *Journal of Visual Communication and Image Representation*, vol. 17, no. 4, pp. 675–685, 2006.

[7] M. Á. García-Garrido, M. Á. Sotelo, and E. Martín-Gorostiza, "Fast road sign detection using hough transform for assisted driving of road vehicles," in *Computer Aided Systems Theory–EUROCAST 2005*. Springer, 2005, pp. 543–548.

[8] D. Gavrila, "Traffic sign recognition revisited," in *Mustererkennung 1999*. Springer, 1999, pp. 86–93.

[9] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[10] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 5, pp. 876–888, 2012.

[11] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 2, p. 3, 2001.

[12] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

Sign type / Video	1	2	3	4	5	1	2	3	4	5
Prohibitory #	**14**	**6**	**9**	**1**	**22**	**14**	**6**	**9**	**1**	**22**
Detected	14	6	9	1	22	14	6	9	1	22
False discovery	42	20	12	5	0	1	2	2	0	0
Missed	0	0	0	0	0	0	0	1	0	1
Warning #	**12**	**17**	**17**	**5**	**2**	**12**	**17**	**17**	**5**	**2**
Detected	12	15	17	4	2	12	17	17	5	2
False discovery	40	45	30	51	0	1	2	1	2	0
Missed	0	2	0	1	0	1	2	0	0	0
			a)					b)		

TABLE II: Number of true positives, false positives and false negatives per each sequence using a) template matching algorithm with temporal consistency enforcement, b) template matching algorithm + temporal consistency enforcement + color verification. The number of false discoveries is obtained by counting how many false objects are detected in each video, thus it does not match the sum of true positives and false negatives.

Fig. 11: Warning and prohibition sign detection. Each pair represents the original image with found detections and the edge magnitude image. The obtained results from template matching procedure are filtered by traffic sign candidate tracking and color border verification as described in III-C and in III-D respectively. The execution time is shown in the upper-left corner of the each original image. All tests are conducted on Intel Core i5-2500K CPU.

Object Tracking Implementation for a Robot-Assisted Autism Diagnostic Imitation Task

Kruno Hrvatinić, Luka Malovan, Frano Petric, Damjan Miklić and Zdenko Kovačić
University of Zagreb, Faculty of Electrical Engineering and Computing, LARICS Laboratory, Unska 3, 10000 Zagreb, Croatia
Email: larics@fer.hr

***Abstract*—Autism spectrum disorders (ASD) is a term used to describe a range of neurodevelopmental disorders affecting about 1% of the population, with increasing prevalence. Due to the absence of any physiological markers, diagnostics is based purely on behavioral tests. The diagnostic procedure can in some cases take years to complete, and the outcome depends greatly on the expertise and experience of the clinician. The predictable and consistent behavior and rapidly increasing sensing capabilities of robotic devices have the potential to contribute to a faster and more objective diagnostic procedure. However, significant scientific and technological breakthroughs are needed, particularly in the field of robotic perception, before robots can become useful tools for diagnosing autism. In this paper, we present computer vision algorithms for performing gesture imitation. This is a standardized diagnostic task, usually performed by clinicians, that was implemented on a small-scale humanoid robot. We describe the algorithms used to perform object recognition, grasping, object tracking and gesture evaluation in a clinical setting. We present an analysis of the algorithms in terms of reliability and performance and describe the first clinical trials.**

I. Introduction

Autism spectrum disorder (ASD) is a developmental disorder characterised by impairment in social interaction, verbal and nonverbal communication and by repetitive behaviours and interests. It has become a commonly diagnosed neurodevelopmental disorder, with increasing prevalence rates, affecting about one in every 100 children [8], and there are no medical markers of autism that could be used in a diagnostic process. Therefore, the diagnosis of ASD is based solely on behavioural observations made by experienced clinicians. However, specific behaviors that are included in diagnostic frameworks and the point at which individual differences in behavior constitute clinically relevant abnormalities are largely arbitrary decisions [4]. *There exists a need for quantitative, objective measurements of social functioning for diagnosis, for evaluating intervention methods, and for tracking the progress of individuals over time* [6]. Modern robotic devices have the potential to fill this need. They can perform actions consistently and repeatably, and evaluate the child's reactions in a quantitative and unbiased way. The tendency of autistic children to interact with technical devices more than with humans around them is another argument in favor of employing robotics for improving the autism diagnostics procedures.

As a starting point for our work, we have chosen the ADOS test [3], which represents the state of the art in autism diagnostics. As described in a preliminary report [5], we have chosen four tasks from the ADOS test and adapted them to the capabilities of the Nao [1] humanoid robot. All of the tasks require the robot to perform a specific action and monitor and evaluate the reaction of the child. In this paper, we present the computer vision algorithms that we have implemented in order to perform the *Functional and symbolic imitation task.* We describe the algorithms and present preliminary results of evaluating their effectiveness in a clinical setting.

The paper is organized as follows. In Section II, we describe the imitation task in detail, and mention the specific circumstances relevant for image processing algorithm design. The three major components of our processing pipeline, namely image segmentation, object tracking and gesture recognition, are described in Sections III, IV, V respectively. Algorithm evaluation results are described in Section VI, and concluding remarks are given in Section VII.

II. Imitation task description

The purpose of the imitation task is to assess the ability of a child to perceive and reproduce a demonstrated gesture. From the point of view of a robotic examiner, the task has three phases:

1) Demonstration,
2) Encouragement for imitation,
3) Observation.

To start the task, the robot needs to make sure the child has its head turned towards it. This can be accomplished by using the robot's built-in face detection software. In the demonstration phase, the robot picks up an object, for example a cup, and demonstrates a gesture, such as drinking, accompanied by appropriate sounds. It then places the object in front of the child and encourages it, by speaking and pointing, to perform the same gesture. Finally, it tracks the object in order to estimate and evaluate the child's reaction. The task can be performed using an object with an obvious everyday function, e.g., using a cup to demonstrate drinking, which is called *functional imitation*. It can also be performed using for example a wooden cylinder to demonstrate the flight of an aeroplane, in which case it is called *symbolic imitation*.

The hardware platform used in our research is the Nao H25 v4.0 humanoid robot, equipped with a 960p30fps RGB camera. In terms of image processing algorithms necessary for successfully conducting the imitation task, the robot needs to perform image segmentation, in order to find the manipulated object. After demonstrating the gesture and releasing the object, the robot needs to keep tracking the object in order to evaluate the gesture reproduced by the child. Since we are considering autism diagnostic procedures performed in clinical settings, we have some control of the environment layout, and the number and type of objects that are visible to the robot.

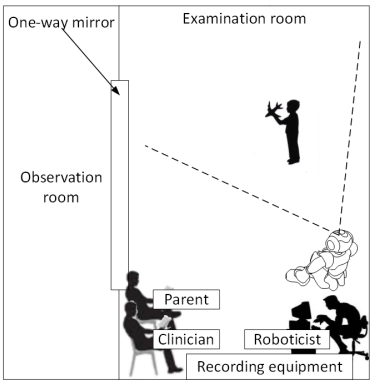


Fig. 1: Layout of the examination room.

Taking into account the layout of the examination room that was used for our research, shown in Figure 1, we have made the following simplifying assumptions:

1) the object can be segmented by color, i.e., the color of the manipulated object is significantly different from the background),
2) the only person in the area of the room visible to the robot is the child, i.e., if the object moves, it is because either the robot or the child are moving it.

On the other hand, the following requirements must be satisfied by the image processing algorithms:

1) the background is dynamic because of robot motion,
2) multiple objects need to be tracked simultaneously, in order to register both the motion of the imitation object and the motion of the child,
3) objects need to be tracked during temporary occlusions,
4) the algorithms need to provide soft real-time performance, running on the robot's onboard computer.

III. Image segmentation

Image segmentation is the first step of the image processing pipeline. It separates the objects of interest, which are in this case the imitation object and the child's hands and face, from the rest of the image. We have extended the work described in [2], based on histogram back projection, to allow simultaneous segmentation of several objects. We make use of the assumption that the tracked object is of a specific, uniform color, which is not over represented in the environment.

A. Histogram creation

Based on [7], [9], the YUV colorspace has been chosen for segmentation. For each class of tracked object, an U-V histogram must be provided (Y data is discarded), constructed from previously gathered training data. If we denote by $N_o(k), k \in [1, K] \subset \mathbb{N}$ the number of pixels belonging to an object, that are grouped in histogram bin k (out of a total of K bins)[1], then the probability distribution of a given color interval can be approximated by

$$P(k|o) \approx \bar{N}_o(k) = \frac{N_o(k)}{\sum_{i=1}^{K} N_o(i)}. \quad (1)$$

[1]The implemented algorithm uses $64 \times 64 = 4960$ bins. The number of bins per dimension is four times smaller than the data resolution to provide some robustness to slight color variations

Similarly, the a priori probability of a color interval occurring can be approximated by

$$P(k) \approx \bar{N}_{bg}(k) = \frac{N_{bg}(k)}{\sum_{i=1}^{K} N_{bg}(i)}, \quad (2)$$

where $N_{bg}(k)$ is the total number of pixels in the image that are sorted in interval k. Finally, the a priori probability of a pixel belonging to the object can be approximated by

$$P(o) \approx \frac{\sum_{i=1}^{K} N_o(i)}{\sum_{i=1}^{K} N_{bg}(i)}. \quad (3)$$

Combining (1), (2) and (3) by the Bayes rule

$$P(o|k) = \frac{P(k|o)P(o)}{P(k)}, \quad (4)$$

we obtain (after canceling corresponding terms)

$$P(o|k) \approx \frac{N_o(k)}{N_{bg}(k)}, \quad (5)$$

which says that the probability of a pixel of some known color being part of an object can be approximated by dividing the object histogram with the image histogram. That probability will always be less then or equal to 1 because $N_o(k) \leq N_{bg}(k)$ and it will be equal to 1 for object colors only. Several pictures are used, obtained in different lighting conditions and against different backgrounds, to create a histogram for each object with background. For each image, a second histogram is created by including only manually selected object pixels, and then apply equation (5) on this pair of histograms. This process allows for a segmentation procedure that is more robust to varying lighting conditions and the presence of similar colors in the background.

B. Image binarization

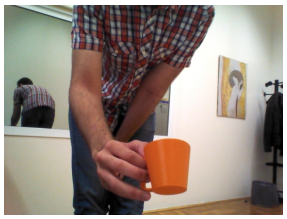
In order to identify objects in the input images obtained by the robot's camera, the image is first converted to YUV space, and then filtered with a small window size median filter to reduce noise. For every object we wish to track, *histogram back projection* is performed on the input image. This yields one single channel image of probability per object. Separating the object from the background is done by *hysteresis thresholding*. In the first step of this procedure the image is binarized with a fixed higher threshold. This yields a sub segmented image: parts that surely belong to the object are separated, but some parts of the object are missing. The area of the object is then expanded by assigning to it adjacent pixels whose value is greater than the lower threshold. This procedure is capable of accurately segmenting an object while ignoring areas of similar color that belong to the background.

Examples of the hysteresis thresholding procedure are shown in Figure 2. Darker green areas in Figure 2b represent areas segmented with the higher threshold, lighter green areas were selected by the lower threshold. The bottom figures demonstrate the ability of the algorithm to reject background artifacts similar in color to the tracked object. The areas colored red in Figure 2d, which satisfy only the lower threshold, are rejected as outliers.

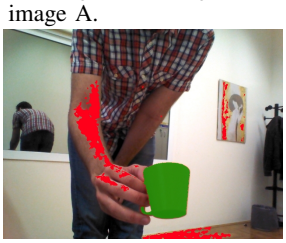
As noted earlier, each object of interest is segmented individually, yielding one binary image per object. All the

(a) Original image A.

(b) Segmented regions of image A.



(c) Original image B.



(d) Segmented regions of image B.

Fig. 2: Two examples of the hysteresis threshold procedure.

images are then merged into one scene representation using the binary OR operator. This composite binary image consists of spatially 8-connected groups of pixels which represent one or several objects. Individual objects are then extracted and tracked using combined metrics of distance and color, as described in the following section.

IV. Object tracking

Object segmentation and tracking are complementary procedures in our implementation. Information about object position in the previous time step provides a prediction for its position in the current image. Segmentation provides the update step for the current object position.

A. Object model

For successful tracking, the object needs to be represented by a simple model that captures all relevant information. Because knowledge of the exact shape of the object is not necessary for the tracking task, we use an ellipse model described by $m_i = (c_{xi}, c_{yi}, a_i, b_i, \theta_i)$, where (c_{xi}, c_{yi}) is the ellipse centroid, a_i and b_i are semi-axis lengths and θ_i is the rotation angle relative to the horizontal axis.

The ellipse model is constructed from the group of 8-connected pixels representing the object by calculating the covariance and centroid of their distribution. Let the object be represented by M pixels $\mathbf{p_j} = [x_j, y_j]^T, j \in [1, M] \subset \mathbb{N}$. The pixel coordinate distribution is expressed as:

$$E[\mathbf{p}] = \frac{\sum_{j=1}^{M} \mathbf{p_j}}{M} = [\mu_x, \mu_y]^T, \tag{6}$$

$$\mathbf{\Sigma} = \begin{bmatrix} m_{xx} & m_{xy} \\ m_{xy} & m_{yy} \end{bmatrix}, \tag{7}$$

Fig. 3: Examples of ellipse models.

where

$$\mu_x = \frac{\sum_{j=1}^{M} x_j}{M}, \mu_y = \frac{\sum_{j=1}^{M} y_j}{M}, \tag{8}$$

$$m_{xx} = \frac{\sum_{j=1}^{M} (x_j - \mu_x)^2}{M}, m_{yy} = \frac{\sum_{j=1}^{M} (y_j - \mu_y)^2}{M}, \tag{9}$$

$$m_{xy} = \frac{\sum_{j=1}^{M} (x_j - \mu_x)(y_j - \mu_y)}{M}. \tag{10}$$

Assuming a Gaussian distribution of points with expectation $E[\mathbf{p}]$ and covariance matrix Σ, the semi-axes lengths of an ellipse containing the points within a 90% confidence interval can be obtained as:

$$a = 2\sqrt{s\lambda_1}, b = 2\sqrt{s\lambda_2} \tag{11}$$

where λ_1, λ_2 are the covariance matrix eigenvalues, and s is the value of the χ^2 distribution for two degrees of freedom with $\alpha = 1 - 0.9 = 0.1$. For the selected confidence interval, $s = 4.605$, eigenvalues are calculated by the equation:

$$K = \sqrt{m_{xx}^2 + m_{yy}^2} - 4(m_{xx}m_{yy} - m_{xy}^2), \tag{12}$$

$$\lambda_1 = \frac{m_{xx} + m_{yy} + K}{2}, \lambda_2 = \frac{m_{xx} + m_{yy} - K}{2}, \tag{13}$$

and the tilt angle of the ellipse θ is calculated as:

$$\theta = \arctan\left(\frac{m_{xx} - \lambda_1}{-m_{xy}}\right). \tag{14}$$

Examples of segmented objects and their corresponding ellipse models are shown in figure 3.

B. Assigning pixels to objects

Assigning segmented pixels to each object model combines object model information available from the previous time step (prediction), with the groups of 8-connected pixels obtained by segmentation (correction). We make the following assumptions in the assignment procedure:

1) Each pixel of each group belongs to at least one object
2) A pixel can belong to several objects
3) Only one group of pixels can be assigned to each object

These assumptions take into account cases in which objects are in contact or are covering each other and are met in most real situations. It should be noted that the aforementioned assumptions allow for a situation where no pixels are assigned to an object. This is interpreted as the object having left the scene.

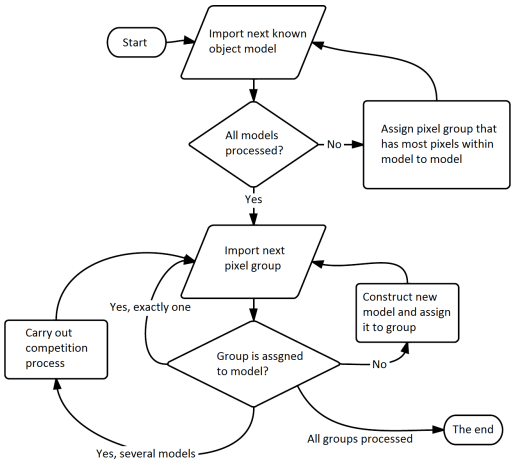


Fig. 4: Flow chart of the procedure for establishing object-pixel correspondence.

In order to perform the assignment, we use the metric

$$d(m_i, \mathbf{p_j}) = \left|\left| \begin{bmatrix} \frac{1}{a_i} & 0 \\ 0 & \frac{1}{b_i} \end{bmatrix} \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \left(\mathbf{p_j} - \begin{bmatrix} c_x \\ c_y \end{bmatrix} \right) \right|\right| \tag{15}$$

which represents the distance of pixel j from object i as the distance of the pixel from the origin of the coordinate system obtained by mapping the ellipse to a unit circle. This metric has the useful property that $d(m_i, \mathbf{p_j}) < 1$ for all pixels inside the ellipse. The assignment procedure is outlined in Figure 4. In the first step, groups of pixels are assigned to objects by calculating the distance metric for each object-pixel pair and, for each object, storing the number of group pixels for which the distance metric is $d(m_i, \mathbf{p_j}) < 1$. This results in a matrix containing the number of pixels which belong to a specific group and also fall within a specific object's ellipse model. After all pixels have been checked, the group which has the largest number of pixels inside an object's ellipse model is assigned to that object. This way at most one group is assigned to each model. If a pixel group was not assigned to any object during this step, it is considered to be a new object.

The second step solves cases in which one group is assigned to several objects, meaning that (partial) occlusion has occurred. To resolve this, those objects must divide the group's pixels among themselves. All the pixels in such contested groups are assigned to one of the contesting objects according to the minimum value of the metric

$$d_2 = d(m_i, \mathbf{p_j})(1 + P_c(\mathbf{p_j})), \tag{16}$$

where $P_c(\mathbf{p_j})$ represents the probability that pixel $\mathbf{p_j}$ belongs to the histogram with which object i is segmented, calculated by taking the histogram value for that pixel's color.

After assigning all segmented pixels to objects (either existing ones or new ones), we perform the prediction step. Assuming that an object in motion will continue its motion in similar direction and with similar speed, the ellipse model center $m_j[k]$ is translated by

$$\overrightarrow{v_2} = 0.5([c_x[k], c_y[k]] - [c_x[k-1], c_y[k-1]])^T \tag{17}$$

The factor 0.5 is introduced because in actual situations such as putting a cup back on the table or clapping hands, sudden deceleration happens more often than sudden acceleration.

V. Gesture recognition

The implemented gesture recognition algorithm offers a simple way of locating the parts of an object's trajectory which match a certain shape. A gesture model is defined as a list of motion directions and compared to the trajectory. This algorithm can determine if a gesture exists in objects trajectory, but can also determine how many times, when and for how long the gesture has been performed.

A. Selection of storage space for trajectories

Assuming the object model is known in every time step, it is possible to construct a trajectory of the object in image space by storing the centroid in every time step. Such a representation is appropriate for use with a fixed camera but fails to accurately describe the object's movement when the camera is in motion, as is the case with a camera mounted to the actuated head of a robot.

Instead, using a properly calibrated pinhole camera model, a line can be drawn through the camera's focus and the recorded point in the image plane. If the pitch and yaw angles α_y, α_z of this line with regards to a coordinate system fixed to the robot's body are used as trajectory points, the trajectory becomes independent of the robot's head rotation. This is the approach used for trajectory storage in the implemented algorithm, as it produces a larger virtual field of view and allows the robot to be placed much closer to the child.

B. Gesture model

In order for a gesture to be detected, it must be described by a model. The model used was selected based on the following requirements:

1) Able to represent gestures in (α_y, α_z) space
2) Invariant to scaling by width
3) Invariant to scaling by height
4) Invariant to translation
5) Not invariant to rotation

The selected model is based on the trajectory angle β, the angle of the object's velocity vector computed as a difference between consecutive trajectory points. Let the beginning of the gesture be defined as an interval χ_1 of angle β i.e. the object beginning to move in a certain direction. Building off this definition, one can define all other segments of the trajectory also with respect to the trajectory angle, which results in the gesture being represented as a set of time-sequential angle intervals. A trajectory will satisfy such a model if its direction smoothly crosses from one interval to another.

To enable modularity and simplify defining new gestures, all gestures are to be described through the use of standard angle intervals which are defined as follows:

$$S(i, \delta) = \left[\frac{2(i-1)\pi}{8} - \delta, \frac{2(i+1)\pi}{8} + \delta \right], \{i \in \mathbb{N} | 0 \leq i \leq 7\} \tag{18}$$

Equation (18) defines eight intervals of width $\frac{\pi}{4} + 2\delta$ with centers in $\frac{i\pi}{4}$. These intervals can be interpreted as the directions $\{right, up-right, up, up-left, left, down-left, down, down-right\}$ with an overlap between adjacent directions of δ.

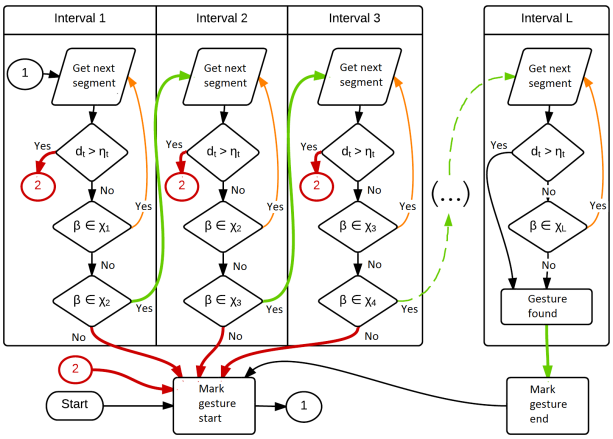


Fig. 5: Flow chart of procedure for comparing model and trajectory

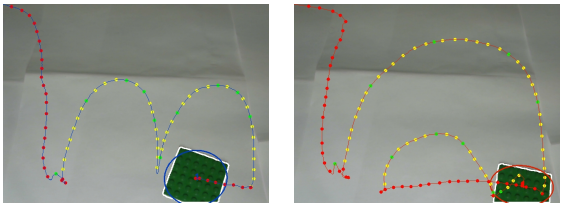


Fig. 6: Comparison procedure

This model meets all the specified requirements and allows for a comparison algorithm with $O(N)$ complexity. It is human-readable and can be made robust to measurement errors and the inadvertent inaccuracy of a person performing the gesture by adjusting the parameter δ of the standard intervals.

C. Comparing models and trajectories

Figure 5 shows the algorithm used for comparing a model with a trajectory. The objective is to find every interval of point indices $[k_{start}, k_{stop}]$ among trajectory points $\Gamma[k] = [t[k], \alpha_y[k], \alpha_z[k]]^T$ in which all the trajectory points represent a gesture defined by the model $\chi = [\chi_1, \chi_2 \cdots \chi_L]$ consisting of L time-sequential angle intervals. The sensitivity of the algorithm can be adjusted by using the distance and time limit parameters η_d, η_t whose default values are $\eta_d = 0.05rad, \eta_t = 1500ms$.

Since the procedure relies on dividing the trajectory into segments, point distances in angle and time need to be defined:

$$d_\theta(\Gamma[s_m], \Gamma[s_n]) = \sqrt{(\alpha_y[s_m] - \alpha_y[s_n])^2 + (\alpha_z[s_m] - \alpha_z[s_n])^2} \quad (19)$$

$$d_t(\Gamma[s_m], \Gamma[s_n]) = t[s_m] - t[s_n] \quad (20)$$

A trajectory segment is defined as an interval of trajectory indices $[s_n, s_m], n > m$ for which $d_\theta > \eta_d$ and/or $d_t > \eta_t$ holds. The next segment of the trajectory can be retrieved by increasing n from $n = m + 1$ until one of these conditions is met. If the stopping condition is distance travelled, the object is moving and the trajectory angle β for the current trajectory segment can be calculated:

$$\beta = \arctan\left(\frac{\alpha_z[s_m] - \alpha_z[s_n]}{\alpha_y[s_m] - \alpha_y[s_n]}\right) \quad (21)$$

The angle β is compared to the angle interval that corresponds to the current state. If β is within the current state's angle interval, the gesture is still being performed and the next segment is retrieved. If β is outside the current state's angle interval, the trajectory has either transitioned into the next state's interval or the part of the trajectory from this part onward does not represent the gesture. If the next state exists and β belongs to its angle interval, the algorithm sets that state to be the current state, otherwise the current state is set to the beginning state and the gesture start point is reset.

If a segment that does not match the current state's angle interval is found, it is possible the gesture has just been successfully completed. This is detected by using a simple rule: if the algorithm is in the last state in the model, the gesture is valid and finished.

Figure 6 shows the algorithm used for comparing the model with the trajectory. Points marked with circles represent the ends of segments and the color of the circle corresponds to the action taken by the algorithm. Green circles represent transitions to the next state or the ending points of the gesture, yellow circles represent segments whose angles β correspond to the current state's angle interval and red circles represent the ending points of segments that do not represent the gesture. The colours on figure 5 also roughly correspond to these transitions.

VI. Experimental evaluation

Gestures that the robot recognizes during the imitation task and their representations are summarized in table I.

TABLE I: Gesture representation

Gesture	Representation	Alternative
Frog	up-left, left, down-left	up-right, right, down-right
Drink	up, down	N/A
Airplane	left, right	N/A

A "frog jumping" gesture is defined as a series of arches (in both directions), with motion having a noticeable horizontal displacement. A "drinking" gesture can be seen as a narrow frog jump, with little or no horizontal displacement, resulting in the gesture being modelled as upward motion followed by downward motion. Due to ambiguity of human interpretation, airplane gestures are simply modelled to be characterized by alternating left and right motion, which may not be the most accurate model but is chosen because there are no parts of the model that are similar to the previous two gestures.

A. Benchmarking

Benchmarking of the segmentation was performed through 1000 images with a framerate of 15 FPS with head tracking turned on and with head tracking turned off. The object histogram used was constructed offline using 40 images of the object in 640×480 resolution. The images were captured using the robot's internal camera under various lighting conditions and backgrounds and the pixels belonging to the object were marked by a human operator using image editing software. The background image was then combined with pixel data to produce a 64×64 bin two-dimensional U-V histogram, as described in Section III. The object was tracked successfully

TABLE II: Benchmarking results

Gesture	Tracking	Tests	As frog	As drink	Success rate
Frog	Head OFF	10	7/10	2/10	7/10
Frog	Head ON	10	5/10	3/10	5/10
Drink	Head OFF	10	0/10	10/10	10/10
Drink	Head ON	10	0/10	10/10	10/10

TABLE III: Comparison of **H**uman and **R**obot observations

Session	Drink		Frog		Airplane	
	H	R	H	R	H	R
Child #1	✓	✓	×	×	✓	×
Child #2	✓	✓	✓	✓	-	
Child #3	Not performed (child not interested)					
Child #4	Not performed (robot malfunction)					

in every image without interruptions, confirming the efficiency of the segmentation algorithm.

Gesture recognition tests were then performed for the frog and drinking gesture, both shown to the robot 10 times.

As table II shows, the drinking gesture was recognized much more consistently than the frog gesture, also, the drinking gesture was not mistaken for the frog gesture while the opposite occurred several times. This is caused by the similarities between these two gestures and variability in human performance. Due to that variability, smaller horizontal displacement during the frog-like gesture can occur, which is in some cases interpreted as upward motion instead of up-right or up-left, resulting in incorrect interpretation of the gestures.

However, in actual use cases the robot is expecting a certain gesture meaning misclassifications are less of an issue. Additionally the frog gesture is expected to be performed by the child multiple times as opposed to the drink gesture which the child performs only once. The algorithm is therefore accurate enough for the purpose it was developed for.

B. Clinical evaluation of imitation task

During the first clinical tests, sessions with four children (three with ASD, one typically developing) were performed, the results of which are presented in table III. Observations made by the robot are compared to those obtained by the expert clinicians, which were additionally validated through analysis of video recordings.

As can be seen in table III, only two sessions were successfully performed, with one failing due to the child not being interested in interaction with the robot, while the other failed due to robot malfunction. In the first session with the child, the robot correctly interpreted the drinking behaviour, while failing to detect the airplane motion. Robot successfully detected that the child did not perform a frog-like gesture. During the session with the second child, the robot correctly detected both the drinking and frog jumping gestures. Since the robot failed to correctly demonstrate the airplane gesture, the results for that part of the task were not obtained.

VII. Conclusions

In this paper, we have described the implementation of an object tracking algorithm for gesture evaluation in the context of robot-assisted autism diagnostics. The purpose of the algorithm is to detect and accurately classify a child's ability to reproduce a simple gesture, such as drinking or the imitation of a jumping frog. The implementation exploits specific conditions of the clinical setting in which the diagnostic procedure is performed. Because the environment is semi-controlled, and the properties of the manipulated objects can be more or less freely chosen, object detection is primarily based on color. Significant features of the described tracking procedure are the ability to track multiple objects simultaneously as well as robustness to background noise and partial object occlusion. It is capable of extracting individual objects and keeping them separated under strong interactions, even when the objects belong to the same class, i.e., are of the same color. The gesture recognition part is based on comparing the observed trajectory to a parametric trajectory description that is compact and invariant to scaling, translation and rotation. Initial deployment in a clinical setting confirmed the usability of the approach.

As the presented work is part of a larger effort aimed at developing a robot capable of acting as an assistant in autism diagnostic procedures, future work will be focused on performing clinical evaluations on a larger population of children. Some drawbacks of the described tracking procedure and classification procedure, such as sensitivity to lighting conditions and handling of full occlusions will be improved.

Acknowledgment

This work has been supported in part by ACROSS - Centre of Research Excellence for Advanced Cooperative Systems (European FP-7 Capacities "Research Potential" program, grant no. 285939, FP7-REGPOT-2011-1) and by the Croatian Science Foundation under the ADORE project (HRZZ-93743-2014).

References

[1] Aldebaran Robotics. *Nao Software Documentation*, v1.14.5 edition, 2013. https://community.aldebaran-robotics.com/doc.

[2] Antonis A Argyros and Manolis IA Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *Computer Vision-ECCV 2004*, pages 368–379. Springer, 2004.

[3] C. Lord, M. Rutter, P.C. Dilavore, and Risi. S. *Autism Diagnostic Observation Schedule*. Western Psychological Services, 2002.

[4] C.F. Norbury and A. Sparks. Difference or disorder? Cultural issues in understanding neurodevelopmental disorders. *Developmental Psychology*, 49(1):45–58, 2013.

[5] F. Petric, M. Cepanec, J. Stošić, S. Šimleša, K. Hrvatinić, A. Babić, L. Malovan, D. Miklić, and Z. Kovačić. Four tasks of a robot-assisted autism spectrum disorder diagnostic protocol: First clinical tests. In *Global Humanitarian Technology Conference (GHTC)*. IEEE, 2014. (To appear).

[6] B. Scassellati, Henny A., and M. Mataric. Robots for use in autism research. *Annual Review of Biomedical Engineering*, 14(1):275–294, 2012.

[7] J-C Terrillon, Mahdad N Shirazi, Hideo Fukamachi, and Shigeru Akamatsu. Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 54–61. IEEE, 2000.

[8] K. Williams, S. MacDermott, G. Ridley, E.J. Glasson, and J.A. Wray. The prevalence of autism in Australia. Can it be established for existing data? *Journal of Paediatrics and Child Health*, 44(9):504–510, 2008.

[9] Ming-Hsuan Yang, David Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):34–58, 2002.

Fast Localization of Facial Landmark Points

Nenad Markuš†, Miroslav Frljak†, Igor S. Pandžić†, Jörgen Ahlberg‡, and Robert Forchheimer‡

† University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia
‡ Linköping University, Department of Electrical Engineering, SE-581 83 Linköping, Sweden

***Abstract*—Localization of salient facial landmark points, such as eye corners or the tip of the nose, is still considered a challenging computer vision problem despite recent efforts. This is especially evident in unconstrained environments, i.e., in the presence of background clutter and large head pose variations. Most methods that achieve state-of-the-art accuracy are slow, and, thus, have limited applications. We describe a method that can accurately estimate the positions of relevant facial landmarks in real-time even on hardware with limited processing power, such as mobile devices. This is achieved with a sequence of estimators based on ensembles of regression trees. The trees use simple pixel intensity comparisons in their internal nodes and this makes them able to process image regions very fast. We test the developed system on several publicly available datasets and analyse its processing speed on various devices. Experimental results show that our method has practical value.**

I. Introduction

The human face plays an essential role in everyday interaction, communication and other routine activities. Thus, automatic face analysis systems based on computer vision techniques open a wide range of applications. Some include biometrics, driver assistance, smart human-machine interfaces, virtual and augmented reality systems, etc. This serves as a strong motivation for developing fast and accurate automatic face analysis systems.

In this paper we describe a novel method for estimating the positions of salient facial landmark points from an image region containing a face. This is achieved by extending our previous work in eye pupil localization and tracking [1]. The developed prototype achieves competitive accuracy and runs in real-time on hardware with limited processing power, such as mobile devices. Additionally, one of the main advantages of our approach is its simplicity and elegance. For example, we completely avoid image preprocessing or the computation of special structures for fast feature extraction, such as integral images and HOG pyramids: the method works on raw pixel intensities.

A. Relevant recent work

Significant progress has been achieved recently in the area of facial landmark localization. The methods considered to be state-of-the-art are described in [2], [3], [4]. The approach described by Belhumeur et al. [2] outperformed previously reported work by a large margin. It combines the output of SIFT-based face part detectors with a non-parametric global shape model for the part locations. The main drawback with this approach is its low processing speed. Cao et al. [3] described a regression-based method for face alignment. Their idea is to learn a function that directly maps the whole facial shape from the image as a single high-dimensional vector. The inherent shape constraint is naturally encoded in the output. This makes it possible to avoid parametric shape models commonly used by previous methods. As this is a tough regression problem, they fit the shape in a coarse-to-fine manner using a sequence of fern[1] ensembles with shape-indexed pixel intensity comparison features. The developed system is both fast and accurate. The system developed by Sun et al. [4] is based on a deep convolutional neural network trained to estimate the positions of five facial landmarks. Additionally, simpler networks are used to further refine the results. The authors report state-of-the-art accuracy results. Recently, deep neural networks started to outperform other methods on many machine learning benchmarks (for example, see [6]). Thus, this is not at all surprising. However, neural networks are usually slow at runtime as they require a lot of floating point computations to produce their output, which is particularly problematic on mobile devices. Chevallier et al. [7] described a method similar to the one we present in this paper. We address this later in the text.

II. Method

The basic idea is to use a multi-scale sequence of regression tree-based estimators to infer the position of each facial landmark point within a given face region. We assume that this region is known in advance. This does not pose a problem since very efficient and accurate face detectors exist (including our own work, currently submitted for review [8]). In contrast to most prior work, we treat each landmark point separately, disregarding the correlation between their positions. Of course, a shape constraint could be enforced in the post-processing step and there are many methods to achieve this. We have decided to exclude this step in order to focus on landmark localization itself. We explain the details of the method in the rest of this section and compare it with previous approaches.

A. Regression trees based on pixel intensity comparisons

To address the problem of image based regression, we use an optimized binary decision tree with pixel intensity comparisons as binary tests in its internal nodes. Variations of this approach have already been used by other researchers to solve certain computer vision problems (for example, see [9], [10], [5]). We define a pixel intensity comparison binary

[1] A simplified decision tree, see [5].

test on image I as

$$\text{bintest}(I; \mathbf{l}_1, \mathbf{l}_2) = \begin{cases} 0, & I(\mathbf{l}_1) \leq I(\mathbf{l}_2) \\ 1, & \text{otherwise,} \end{cases}$$

where $I(\mathbf{l}_i)$ is the pixel intensity at location $\mathbf{l}_i$. Locations $\mathbf{l}_1$ and $\mathbf{l}_2$ are in normalized coordinates, i.e., both are from the set $[-1, +1] \times [-1, +1]$. This means that the binary tests can easily be resized if needed. Each terminal node of the tree contains a vector that models the output. In our case, this vector is two-dimensional since we are interested in estimating the landmark position within a given image region.

The construction of the tree is supervised. The training set consists of images annotated with values in $\mathbb{R}^2$. In our case, these values represent the location of the landmark point in normalized coordinates. The parameters of each binary test in internal nodes of the tree are optimized in a way to maximize clustering quality obtained when the incoming training data is split by the test. This is performed by minimizing

$$Q = \sum_{\mathbf{x} \in C_0} \|\mathbf{x} - \bar{\mathbf{x}}_0\|_2^2 + \sum_{\mathbf{x} \in C_1} \|\mathbf{x} - \bar{\mathbf{x}}_1\|_2^2, \qquad (1)$$

where C_0 and C_1 are clusters that contain landmark point coordinates $\mathbf{x} \in \mathbb{R}^2$ of all face regions for which the outputs of binary test were 0 and 1, respectively. The vector $\bar{\mathbf{x}}_0$ is the mean of C_0 and $\bar{\mathbf{x}}_1$ is the mean of C_1. As the set of all pixel intensity comparisons is prohibitively large, we generate only a small subset[2] during optimization of each internal node by repeated sampling of two locations from a uniform distribution on a square $[-1, +1] \times [-1, +1]$. The test that achieves the smallest error according to equation 1 is selected. The training data is recursively clustered in this fashion until some termination condition is met. In our setup, we limit the depth of our trees to reduce training time, runtime processing speed and memory requirements. The output value associated with each terminal node is obtained as the weighted average of ground truths that arrived there during training.

It is well known that a single tree will most likely overfit the training data. On the other hand, an ensemble of trees can achieve impressive results. A popular way of combining multiple trees is the gradient boosting procedure [11]. The basic idea is to grow trees sequentially. Each new one added to the ensemble is learned to reduce the remaining training error further. Its output is shrunk by a scalar factor called shrinkage, a real number in the set $(0, 1]$, that plays a role similar to the learning rate in neural networks training. We set this value to 0.5 in our experiments.

B. Estimating the position of a landmark point

We have observed that accuracy and robustness of the method critically depend on the scale of the rectangle within which we perform the estimation. If the rectangle is too small, we risk that it will not contain the facial landmark at all due to the uncertainty introduced by face tracker/detector used to localize the rectangle. If the rectangle is big, the detection is more robust but accuracy suffers. To minimize these effects, we learn multiple tree ensembles, each for estimation at different scale. The method proceeds in a recursive manner, starting with an ensemble learned for largest scale. The obtained intermediate result is used to position the rectangle for the next ensemble in the chain. The process continues until the last one is reached. Its output is accepted as the final result. This was inspired by the work done by Ong et al. [12].

The output of regression trees is noisy and can be unreliable in some frames, especially if the video stream is supplied from a low quality camera. This can be attributed to variance of the regressor as well as to the simplicity of binary test at internal nodes of the trees: Pixel footprint size changes significantly with variations in scale of the eyes and we can expect problems with aliasing and random noise. These problems can be reduced during runtime with random perturbations [13]. The idea is to sample multiple rectangular regions at different positions and scales around the face and estimate the landmark point position in each of them. We obtain the result as the median over the estimations for each spatial dimension.

We would like to note that Chevallier et al. [7] described a similar method for face alignment. The main difference is that they use Haar-like features instead of pixel intensity comparisons to form binary tests in internal nodes of the trees. Also, they do not perform random perturbations in runtime. This is presumably not needed with Haar-like features as they are based on region averaging which is equivalent to low pass filtering and this makes them more robust to aliasing and noise.

III. Experimental analysis

We are interested in evaluating the usefulness of the method in relevant applications. Thus, we provide experimental analysis of its implementation in the C programming language. We compare its accuracy with the reported state-of-the-art and modern commercial software. Also, we analyse its processing speed and memory requirements.

A. Learning the estimation structures

We use the LFW dataset [14] and the one provided by Visage Technologies (http://www.visagetechnologies.com/). Both consist of face images with annotated coordinates of facial landmarks. These include the locations of eyebrows, nose, upper and lower lip, mouth and eye corners. Overall, the total number of annotated faces is around $15\ 000$. We intentionally introduce position and scale perturbations in the training data in order to make our system more robust. We extract a number of samples from each image by randomly perturbing the bounding box of the face. Furthermore, as faces are symmetric, we double the size of the training data by mirroring the images and modifying the landmark point coordinates in an appropriate manner. This process results in a training set that consists of approximately $10\ 000\ 000$ samples.

Each landmark point position estimation structure is learned independently in our framework. We have empirically found that 6 stages with 20 trees of depth equal to 9 give good results in practice. The ensemble of the first stage

[2] 128 in our implementation.

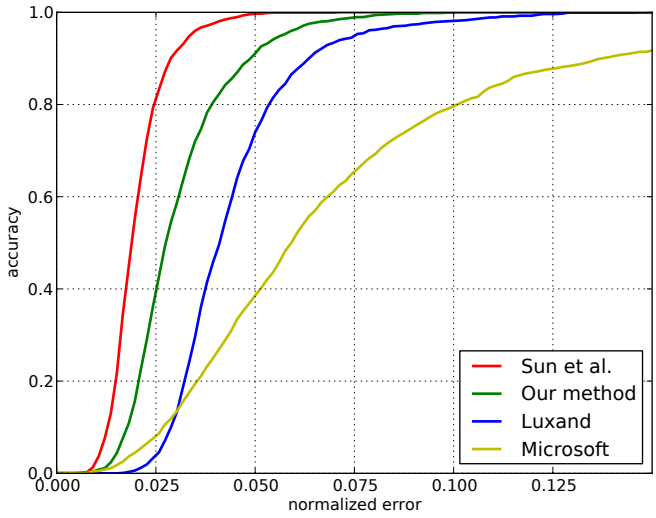


Fig. 1. Accuracy curves on the BioID dataset.

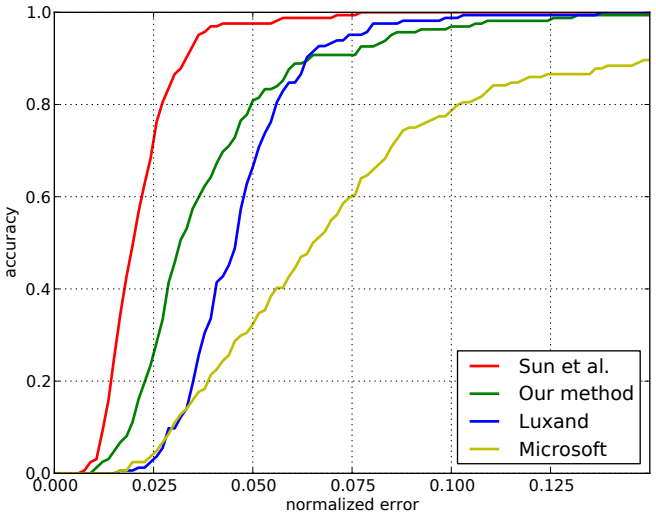


Fig. 2. Accuracy curves on the LFPW dataset.

is learned to estimate the position of a particular landmark point from the bounding box of the face. Each next stage is learned on a training set generated by shrinking the bounding box by 0.7 and repositioning its center at the position output by the ensemble of the previous stage. This process proceeds until the last stage is reached. The learning of the whole estimation structure for a single landmark point takes around one day on a modern desktop computer with 4 cores and 16 GB of memory.

B. Accuracy analysis on still images

We use the BioID [15] and LFPW [2] face datasets to evaluate the accuracy in still images. The normalized error [15] is adopted as the accuracy measure for the estimated landmark point locations:

$$e = \frac{1}{N} \sum_{n=1}^{N} \frac{D_n}{D}, \tag{2}$$

where N is the number of facial landmarks, D is the distance between the eyes and D_n is the Euclidean distance between the estimated landmark position and the ground truth. The accuracy is defined as the fraction of the estimates having an error smaller than a given number. Roughly, an error of 0.25 corresponds to the distance between the eye center and the eye corners, 0.1 corresponds to the diameter of the iris, and 0.05 corresponds to the diameter of the pupil.

First, we use the BioID and LFPW datasets to compare our system to the state-of-the-art method based on convolutional neural networks [4] and two modern commercial systems, one provided by Microsoft [16] and the other by Luxand [17]. We follow the protocol from [4] to obtain normalized errors for five facial landmarks (eye centers, mouth corners and tip of the nose). The results are reported in figures 1 and 2. We can see that our method outperforms both commercial systems in accurate landmark point localization ($e \approx 0.05$) but the neural network-based system is clearly the best. We could not include the method by Cao et al. [3] in this comparison since no code/binaries are available to reproduce the results. We also excluded the results published by Chevallier et al. [7] since they used the evaluation methodology where they partitioned BioID in two parts: one for training and the other for accuracy analysis. It has been argued that this evaluation methodology is flawed since the learning procedure overfits some particular features present only in the used dataset and thus yields performance that is not representative in the general case [18].

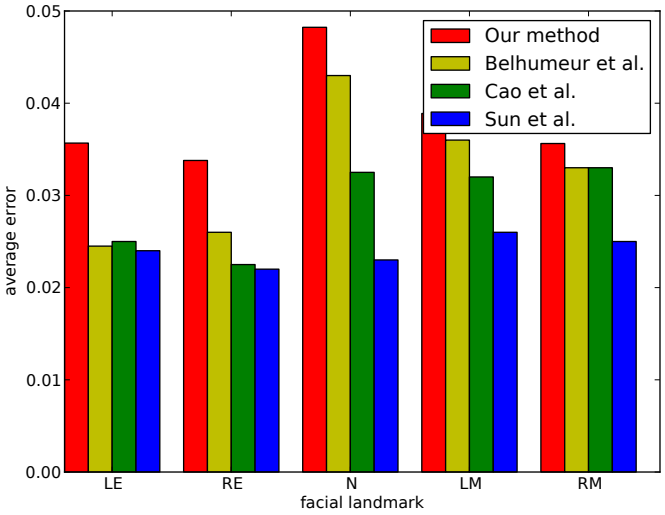


Fig. 3. Average accuracy comparison on the LFPW test set (249 images).

In order to compare our method also with the methods excluded from the first experiment, we performed a second comparison. This one is based on average errors reported on the LFPW dataset (accuracy curves could not be obtained due to the lack of data in [2] and [3]). The average error for 5 facial landmarks can be seen in Figure 3. As the average error is sensitive to outliers and LFPW faces vary greatly in pose and occlusions, our method performs worse than other approaches that use some form of shape constraint.

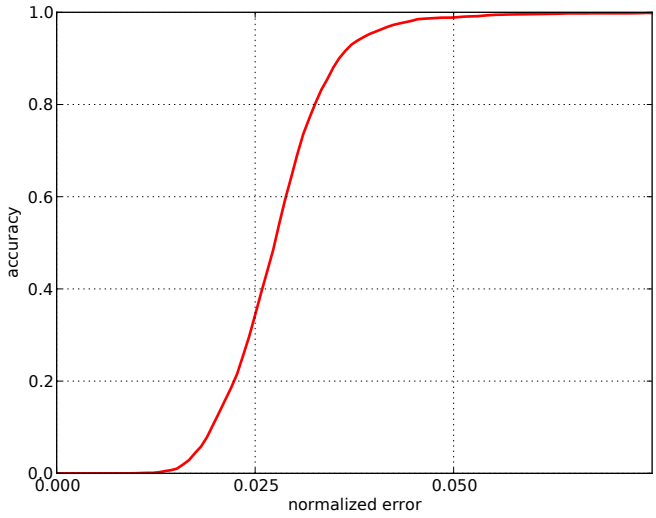


Fig. 4. Accuracy curve for The Talking Face Video.

Device	CPU	Time [ms]
PC1	3.4GHz Core i7-2600	1.3
PC2	2.53GHz Core 2 Duo P8700	2.1
iPhone 5	1.3GHz Apple A6	5.0
iPad 2	1GHz ARM Cortex-A9	8.8
iPhone 4S	800MHz ARM Cortex-A9	10.9

TABLE I. AVERAGE TIMES REQUIRED TO ALIGN 5 FACIAL LANDMARKS.

Nevertheless, we can see that, on average, the landmark positions estimated by our system are within the pupil diameter from the ground truth.

C. Tracking facial features

We use the Talking Face Video [19] to evaluate our system quantitatively in real-time applications. The video contains 5000 frames taken from a video of a person engaged in conversation. A number of facial landmarks were annotated semi-automatically for each frame with an active appearance model trained specifically for the person in the video. These annotations include the locations of eye centers, mouth corners and the tip of the nose. The normalized error averaged over the video sequence obtained by our system was equal to 0.028. Accuracy curve can be seen in Figure 4. These results show that most of the time our system estimated the positions of facial landmarks with high accuracy.

D. Processing speed and memory requirements

Processing speeds obtained by our system on various devices can be seen in Table I. Our system uses a single CPU core although the computations can easily be parallelized. Both Cao et al. [3] and Sun et al. [4] vaguely[3] report processing speeds on modern CPUs: their systems localize 29 and 5 facial landmarks, respectively, in 5 and 120 [ms], respectively.

[3]For example, we are not sure if they used multi-core processing in runtime (both papers mention it at some point).

Each landmark position estimator consisting of 120 trees, each of depth 9, requires around 700 kB of memory. In our opinion, these relatively large memory requirements are one of the drawbacks with our approach as they are inconvenient for some applications, such as face tracking in web browsers or on mobile devices. The problem can be addressed by quantizing the outputs in the leafs of each tree. In the current implementation, we represent each output with two 32-bit floating point values.

E. Qualitative results

Some qualitative results obtained by our system can be seen in Figure 5 and in the video available at http://youtu.be/xpBXpI39s9c. Furthermore, we prepared a demo application for readers who wish to test the method themselves. It is available at http://public.tel.fer.hr/lploc/.

IV. CONCLUSION

Numerical results show that our system is less accurate than the reported state-of-the-art but more accurate than two modern commercial products while being considerably faster, in some cases even by a factor of 50. Its landmark point position estimations are, on average, in the pupil diameter ($e \approx 0.05$) from human-annotated ground truth values. Processing speed analysis shows that the system can run in real-time on hardware with limited processing power, such as modern mobile devices. This enables fast and reasonably accurate facial feature tracking on these devices. We believe that the method described in this paper achieves acceptable accuracy and processing speed for a lot of practical applications.

ACKNOWLEDGEMENTS

This research is partially supported by Visage Technologies AB, Linköping, Sweden, by the Ministry of Science, Education and Sports of the Republic of Croatia, grant number 036-0362027-2028 "Embodied Conversational Agents for Services in Networked and Mobile Environments" and by the European Union through ACROSS project, 285939 FP7-REGPOT-2011-1.

REFERENCES

[1] N. Markuš, M. Frljak, I. S. Pandžić, J. Ahlberg, and R. Forchheimer, "Eye pupil localization with an ensemble of randomized trees," *Pattern Recognition*, 2014. 1

[2] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," in *CVPR*, 2011. 1, 3

[3] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," in *CVPR*, 2012. 1, 3, 4

[4] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *CVPR*, 2013. 1, 3, 4

[5] M. Ozuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code," in *CVPR*, 2007. 1

[6] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, 2006. 1

[7] L. Chevallier, J.-R. Vigouroux, A. Goguey, and A. Ozerov, "Facial landmarks localization estimation by cascaded boosted regression," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2013. 1, 2, 3

Fig. 5. Some results obtained by our system on real-world images. Notice that we use more facial landmark point detectors than in the experimental section.

[8] N. Markuš, M. Frljak, I. S. Pandžić, J. Ahlberg, and R. Forchheimer, "A method for object detection based on pixel intensity comparisons," http://arxiv.org/abs/1305.4537v1, 2013. 1

[9] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, 2011. 1

[10] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *TPAMI*, 2012. 1

[11] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, 2001. 2

[12] E.-J. Ong, Y. Lan, B. Theobald, and R. Bowden, "Robust facial feature tracking using selected multi-resolution linear predictors," *TPAMI*, 2011. 2

[13] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," in *CVPR*, 2010. 2

[14] M. Dantone, J. Gall, G. Fanelli, and L. V. Gool, "Real-time facial feature detection using conditional regression forests," in *CVPR*, 2012. 2

[15] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz, "Robust face detection using the hausdorff distance." Springer, 2001, pp. 90–95. 3

[16] "Microsoft research face sdk beta," 2013. [Online]. Available: http://research.microsoft.com/en-us/projects/facesdk/ 3

[17] "Luxand facesdk," 2013. [Online]. Available: http://www.luxand.com/facesdk/ 3

[18] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR*, 2011. 3

[19] "The talking face video," 2013. [Online]. Available: http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html 4

Recognizing 3D Objects from a Limited Number of Views using Temporal Ensembles of Shape Functions

Karla Brkić*, Siniša Šegvić*, Zoran Kalafatić*, Aitor Aldomà,‡, Markus Vincze‡
*University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia
‡Automation and Control Institute, Vienna University of Technology, Austria
Email: karla.brkic@fer.hr

***Abstract*—We consider the problem of 3D object recognition, assuming an application scenario involving a mobile robot equipped with an RGB-D camera. In order to simulate this scenario, we use a database of 3D objects and render partial point clouds representing depth views of an object. Using the rendered point clouds, we represent each object with an object descriptor called temporal ensemble of shape functions (TESF). We investigate leave-one-out 1-NN classification performance on the considered dataset depending on the number of views used to build TESF descriptors, as well as the possibility of matching the descriptors built using varying numbers of views. We establish the baseline by classifying individual view ESF descriptors. Our experiments suggest that classifying TESF descriptors outperforms individual ESF classification, and that TESF descriptors offer reasonable descriptivity even when very few views are used. The performance remains very good even if the query TESF and the nearest TESF are built using a differing number of views.**

I. Introduction

3D object recognition is one of the essential tasks in practical robotics. In order to interact with the world, robots must be capable of understanding which objects they encounter in it. Although the objects in reality are three-dimensional, the dimensionality of the data that a robot perceives depends on the sensors used to acquire it. In this paper, we assume a scenario in which a robot is equipped with an RGB-D camera (e.g. Kinect or an equivalent sensor). The robot moves around the object of interest, acquiring a number of depth views of the object, as illustrated in Figure 1. We are interested exclusively in the depth channel, so the RGB image is discarded. We have previously proposed [1] a method for integrating multiple depth views of a 3D object into a single descriptor, called *temporal ensemble of shape functions descriptor* (TESF). In order to build a TESF descriptor, depth information for each view is represented as a point cloud. These point clouds are then represented using individual *ensemble of shape functions descriptors* (ESF) [2], and individual ESFs are combined to form TESF descriptors. A TESF descriptor can be built using an arbitrary number of object views. The resulting descriptor is always of the same length. Therefore, TESF descriptors are particularly suitable for object classification in practical robotics, as a robot can acquire varying numbers of views for different objects.

In this paper, we build on earlier work with temporal ensembles of shape functions. Previous experiments, detailed in [1], focused on building TESF descriptors using a *fixed number* of object views. In addition, these views were designed to capture the object from all possible sides. TESF descriptors proved to be very discriminative, performing equally well or slightly better than state of the art solutions [3]. Still, some open questions important in the context of practical robotics remained:

1) how descriptive are TESF descriptors when built on very few object views,
2) can a TESF descriptor classifier built using t_1 views be applied on TESF descriptors built using t_2 views, $t_1 \neq t_2$?

The goal of this work is to provide detailed experiments to answer these open questions. We do so by adhering to experimental setup from [1], where actual depth data is simulated using rendered 3D models.

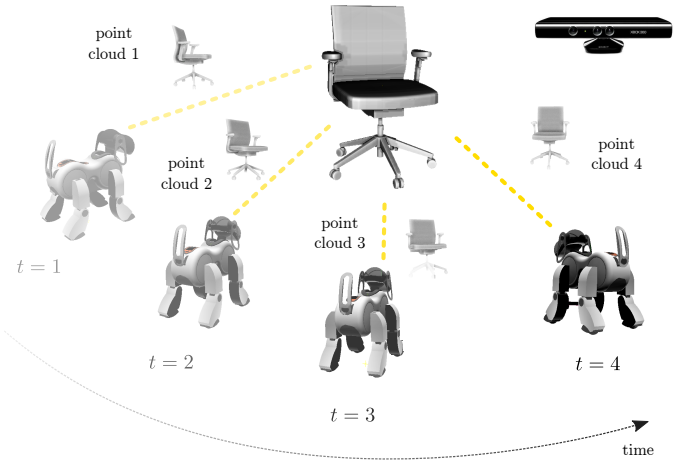


Fig. 1: Our application scenario. The robot moves around the object of interest (in this case, chair) and acquires several views (in this case, four) using a depth camera. These views are represented as point clouds.

II. Related work

The problem of 3D object recognition appears in computer (i.e. robot) vision and in computer graphics. From the computer graphics perspective, the focus is on 3D object retrieval,

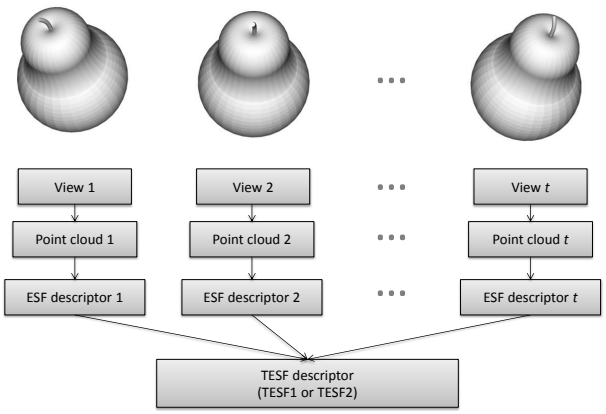


Fig. 2: From object views to TESF descriptors.

i.e. finding 3D objects similar to a query object. From the computer vision perspective, the focus is on individual real world 3D object classification.

Combining these two perspectives is becoming increasingly common, and it is difficult to draw a hard line between them. For example, Wohlkinger et al. [4] propose a 3D object recognition framework for depth sensors that is trained exclusively on CAD models downloaded from the Internet. Tenorth et al. [5] propose decomposing CAD models of daily used objects to learn about their functional parts, which could be useful in robotic applications. Ohbuchi and Furuya [6] propose a method for retrieving 3D models based on a single-view query using bags of visual features, which is an idea from computer vision. Chen et al. propose measuring similarity of contours extracted from rendered images of models, a purely computer graphics approach that could easily be generalized to computer vision. Daras and Axenopoulos [7] introduce a 3D object retrieval framework that supports multimodal queries (sketches, images or 3D models). Further details on 3D object retrieval can be found in a number of surveys, e.g. [8] or [9].

Temporal ensembles of shape functions (TESF descriptors) [1], used in this paper, are based on ensembles of shape functions [2] and spatio-temporal appearance descriptors [10]. They combine a single-view object recognition method with an idea of temporal integration from the domain of video analysis. They can be readily applied in computer graphics. When working with real images, however, RGB data itself is not sufficient, as depth channel is necessary to generate ensembles of shape functions. Different views of an object are represented as 3D point clouds, encoded as ensembles of shape functions and efficiently combined into a single descriptor. This procedure is illustrated in Figure 2.

III. Temporal Ensembles of Shape Functions

To build a TESF descriptor of an object, we assume that a number of partial point clouds representing the object are available. In our envisioned application, these partial point clouds could be obtained e.g. by a Kinect-like sensor, so that each partial point cloud represents one depth view of the object as seen by the sensor. Each of these partial views is first represented using ensembles of shape functions (ESF), introduced by Wohlkinger and Vincze [2].

A. The ESF descriptor

The basic idea of the ensemble of shape functions descriptor is to represent a partial point cloud by distributions of values of characteristic shape functions. Each point cloud represents a (partial) surface of the object. The descriptor is an extension of the idea of shape functions, as introduced by Osada et al. [11].

In order to represent the partial point cloud by a characteristic shape function, we randomly sample pairs of points on the partial point cloud surface and measure their distance. We then categorize the connecting line segment as lying mostly on the surface, lying mostly off the surface, or being a combination of both cases. For each of the cases, we maintain a 64-bin histogram of recorded distances. For instance, if we measure point distance of 12.27, and the line segment connecting the two selected points is lying mostly off the partial surface, then the value 12.27 will be entered into the histogram for lines lying off surface.

The described procedure is repeated for randomly sampled point triplets. However, instead of measuring point distances, we now measure the area of the spanned triangle, as well as a predefined angle in the triangle. The triangle is again characterized as lying mostly on the surface, lying mostly off the surface, or a combination of both, and the measured area and angle are entered into the appropriate histogram.

By randomly sampling point pairs and triplets and measuring the obtained distances, areas and angles, we obtain a total of 9 histograms. Additionally, we measure the ratios of point triplet line distances, and enter them into another histogram. This results in a total of 10 64-bin histograms that are representative of individual partial point clouds. In order to obtain the ESF descriptor of a partial point cloud, we concatenate the described ten histograms into a single 640-dimensional descriptor which is then normalized. Note that the ESF computation algorithm ensures invariance to translation and rotation.

In practice, in order to perform the described calculations we approximate the partial point surface with a voxel grid. Decisions whether a line or a triangle is on the surface, off the surface or both are made on the basis of that grid approximation. Further implementation details can be found in [2].

B. Integrating individual ESF descriptors

ESF descriptors are built on a per-view basis, meaning that for t partial views of an object we will obtain t ESF descriptors. In a real robotics scenario, we would like to classify the object that the robot is seeing using all available descriptors. However, most standard classifiers expect a single fixed-length feature vector as an input, so we need a way to integrate t ESF descriptors into a single descriptor.

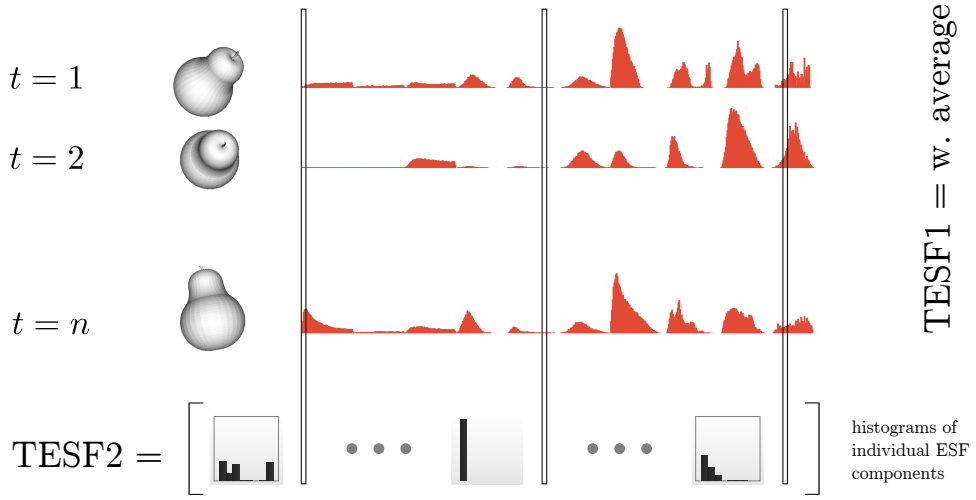


Fig. 3: Calculating TESF1 and TESF2 descriptors. Given a number of views and their corresponding ESF descriptors (drawn in orange), TESF1 is obtained by weighted averaging of all the available ESFs. TESF2 is obtained by histogramming each ESF component (represented here by framing three example components with a rectangle and building a histogram out of the collected data).

In [1] we proposed two variants of integrating ESF descriptors into a single descriptor, namely

1) Temporal Ensembles of Shape Functions of the First Order (TESF1), and
2) Temporal Ensembles of Shape Functions of the Second Order (TESF2).

Let us assume that the robot is moving around the object and looking at it, so that in each point in time θ we obtain a view v_θ and its corresponding ESF descriptor $\mathrm{ESF}(v_\theta)$. We now proceed to describe the algorithms for calculating TESF1 and TESF2 descriptors.

C. The TESF1 descriptor

Assume that we wish to build the TESF1 descriptor at the point in time t. We have already obtained t views of the object and their corresponding ESF descriptors $\mathrm{ESF}(v_\theta)$, $1 \leq \theta \leq t$. In order to obtain the TESF1 descriptor from these views, we simply do a weighted averaging of the t ESF descriptors:

$$\mathrm{TESF1}(t) = \sum_{\theta=1}^{t} \alpha_\theta \mathrm{ESF}(v_\theta). \quad (1)$$

Factors α_θ are arbitrary, and should be set depending on a particular application. For instance, it might be the case that all the views are equally relevant, so it makes sense to set $\alpha_\theta = 1\ \forall\theta$. On the other hand, it might be the case that we know something important about the way the views were acquired, for instance that the robot is moving away from the object. In that particular case it makes sense to decrease the importance of latter views, e.g. by setting $\alpha_\theta = \frac{c}{\theta}$, where c is an arbitrary constant. Different strategies for choosing α_θ can be derived by analogous reasoning.

TESF1 is a simple and effective descriptor [1]. However, when averaging individual ESF descriptors, a lot of information is lost. Two very different sets of ESF descriptors can produce the same TESF1 descriptor. To address this problem, we have proposed the TESF2 descriptor.

D. The TESF2 descriptor

The TESF2 descriptor is designed to be more expressive than TESF1, by explicitly modeling the distribution of ESF descriptor components over time.

Let us define a component vector, $\mathbf{c}_i(t)$, as a vector of values of the i-th component $\mathrm{ESF}(v_\theta)[i]$ of the ESF descriptor $\mathrm{ESF}(v_\theta)$ up to and including time t, $1 \leq \theta \leq t$:

$$\mathbf{c}_i(t) = (\mathrm{ESF}(v_1)[i],\ \mathrm{ESF}(v_2)[i],\ \ldots,\ \mathrm{ESF}(v_t)[i])^T. \quad (2)$$

For a given point in time t, we will have a total of 640 component vectors $\mathbf{c}_i(t)$, where $1 \leq i \leq 640$. There will be one component vector for each of the 640 components of the modeled ESF descriptors. In time t, the length of each of these component vectors will be equal to t.

To obtain the TESF2 descriptor in time t, each of the 640 component vectors is treated as a set of measurements and modeled with a k-bin histogram. The TESF2 descriptor is a concatenation of the bin frequencies of all 640 histograms, which can be written as

$$\mathrm{TESF2}(t) = [\mathcal{H}_k(\mathbf{c}_1(t)), \mathcal{H}_k(\mathbf{c}_2(t)), \ldots, \mathcal{H}_k(\mathbf{c}_{640}(t))]^T. \quad (3)$$

The function $\mathcal{H}_k(\mathbf{c})$ builds a k-bin histogram of values contained in the vector $\mathbf{c}_i$ and returns a vector of histogram bin frequencies. Given that the individual components of the grid vector correspond to bins of individual ESF histograms, TESF2 descriptors can be thought of as building histograms of the second order, i.e. histograms of histograms.

Figure 3 illustrates the differences in calculating TESF1 and TESF2 descriptors.

E. Solving the TESF2 binning problem

In practical implementations of the histogramming function $\mathcal{H}_k(\mathbf{c})$, there is a problem in defining bin bounds of the built histograms. The maximum theoretical value that can be encountered in the components of an ESF descriptor is 1, as ESF descriptors are normalized. In practice, ESF descriptor components are typically of the order of magnitude of 10^{-3}, given that the normalization is performed over 640 vector components. Therefore, it makes sense to employ non-linear binning, in order to increase the binning resolution around typically encountered values. In our experiments, we employ the non-linear binning scheme from [1]. Further details on the binning are omitted from this work, and the interested reader is referred to [1]

IV. Experiments with a limited number of views

In order to test the performance of TESF descriptors built using varying numbers of views, we adhere to the experimental setup from [1]. To simulate the performance of a Kinect-like sensor, we use a database of 3D object models and partially render each model from 20 different angles. The 20 views are evenly distributed on a sphere around the object, and each rendered view is assigned an index from 1 to 20. We employ a 1-nearest neighbor classifier [12] using Euclidean distance and leave-one-out cross-validation to obtain performance estimates. We use the publicly available implementation of ESF available in the Point Cloud Library (PCL) [13]. The number of point-pair and point-triplet samples is set to 40000, and a $64 \times 64 \times 64$ voxel grid is used.

A. The 3D-Net database

In our experiments, we use a subset of the 3D-Net database [2]. The considered subset consists of 1267 objects grouped into 55 classes organized in a hierarchy according to the WordNet database [14]. The object classes range from common household objects to vehicles and animals. A word cloud visualization of all the used object categories is shown in Figure 4, and a few example objects in Figure 5.

B. The baseline: individual ESF descriptor matching

The simplest way to classify an object seen from multiple views is to ignore the fact that the views are related and to treat each view as an individual observation. In the case of 1-NN classification, object class is then determined by comparing a view with a database of all stored views of all the training objects. Assume that we are using the described subset of 1267 objects from 3D-Net as our training set and 20 views per object. This effectively means comparing the ESF descriptor of a view of a new object with $1267 \times 20 = 25430$ stored ESF descriptors. This procedure will be 20 times slower than analogous matching of TESF1 descriptors and $\frac{20}{k}$ times slower than analogous matching of TESF2 descriptors, where k denotes the number of bins used for building TESF2 descriptors.

Fig. 4: A word cloud visualization of the considered classes from the 3D-Net database. Word size is proportional to the number of instances of the given class.

Fig. 5: Examples of objects from the 3D-Net database.

In order to speed up processing, one could consider matching only corresponding views. For example, if we assume our setup where 20 views are spaced at predefined positions around the object, view at position 14 of a query object could be compared only to views at position 14 of all other objects. The major problem with this kind of setup is that objects in the 3D-Net database are not necessarily equally aligned in the 3D coordinate system, and their orientation is not known. Taking the same view of two objects of the same class in the 3D-Net database might yield quite different surfaces, as illustrated in Figure 6. Therefore, we instead opt for comparing the ESF

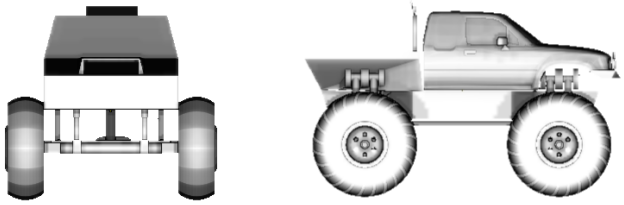


Fig. 6: The same view (view 3) of two different trucks from the 3D-Net database. The trucks are oriented differently in the 3D space, so viewing them from the same position yields two different surfaces.

descriptor of a query view with all other views of all other objects.

To investigate classification performance of individual ESF descriptor matching, we set up the following experiment. For each of the 1267 objects in the training set, we consider all 20 views of the object, and try to match each view's ESF descriptor with the remaining $(1267-1)\times 20$ ESF descriptors of all views of all other objects.

This kind of matching yields a 1-NN leave-one-out classification accuracy of 65.99%. In [1], we have shown that by using TESF1 descriptors on the same problem one obtains a classification accuracy of 77.03%, and by using TESF2 descriptors a classification accuracy of 82.64%. Hence, using TESF descriptors we obtain not only faster, but also more accurate classification.

C. Varying the number of views for query TESFs

Although TESF descriptors perform very well when sufficient number of views is sampled around the object of interest, there are two essential questions that need to be addressed when applying TESF descriptors in a practical robotics scenario: (i) what is the influence of the number of used views on classification accuracy, and (ii) is this approach robust to a changing number of views?

As mentioned before, we are assuming that a robot is moving around an object of interest and acquiring a number of views of the object. These views are then represented by their ESF descriptors. The robot need not be able to see the object from all angles, and the number of acquired views can vary depending on the situation. Using these views, the robot builds a TESF descriptor and tries to determine which class the object belongs to by finding the nearest neighbor to the built TESF descriptor in its *object descriptor database*. The object descriptor database contains precomputed TESF descriptors of training objects. The stored TESF descriptors need not be built on the same views as the query descriptor. Both view angles and the total number of views might differ.

To investigate the robustness of TESF classification to these changes in views, we vary the number of views used in building TESF descriptors of query objects and measure classification performance. At the start of each experiment, we randomly select n view indices from the rendered 20 views of objects in the 3D-Net (e.g. when selecting three views, views 3, 6 and 17 might be chosen). When evaluating leave-one-out classification accuracy, we select individual objects from the training set and build their TESF descriptors using only the chosen n view indices. 1-NN classification of these descriptors is then performed by searching for the nearest TESF descriptor in the rest of the set. However, for the remainder of the set (which simulates the object descriptor database of the robot) we use the full 20-view range TESF descriptors. In other words, when finding the nearest neighbor the left out object is represented by a TESF descriptor using n views, and this descriptor is then compared to TESF descriptors of other objects built using 20 views to find the closest match. We test both TESF1 and TESF2 descriptors.

Randomly selecting view numbers can be biased. It is possible to select very close views, leading to small information gain over a single view scenario. On the other hand, it is also possible to select views that are very far apart, which might not be the case in real robotic scenarios, leading to overly optimistic classification rates. To alleviate this problem, we repeat the random selection of views 10 times for each considered number of views n, and we report the average obtained classification accuracy and the standard deviation. Results are summarized in Table I.

Our first observation is that TESF1 descriptors consistently perform worse than TESF2 descriptors, as is to be expected given that TESF2 descriptors are more expressive. However, TESF1 descriptors built using 8 views and above offer an improvement over individual ESF descriptor matching. For TESF2, we see that even with using very few views (3), we are still likely to see an increase in performance over individual ESF 1-NN classification (although the standard deviation of performance is quite large). The classification performance seems to steadily increase as we add more views, and the standard deviation of the performance drops. As our views are equally positioned around the object, selecting more and more views means obtaining more information about the object. At 8 views the object is already reasonably well represented, and at 15 views the standard deviation is very small, which means that regardless of which exact 15 views we choose, the majority of the surface of the object will be seen and captured in the resulting TESF descriptor.

D. Building the object descriptor database from fewer views

In previous experiments, we considered matching TESF descriptors of query objects built using randomly selected 3, 5, 8 and 15 views with an object descriptor database built using 20 object views. Now, we investigate how classification performance would change if our object descriptor database instead contained descriptors built using the same number of views as the query object. Our goal is to see whether good classification accuracy can be obtained even if the objects stored in the database were seen from a limited number of views, which is s realistic expectation in robotic applications. To that end, we repeat the experiment described in the previous

Number of views n	TESF1 accuracy [%]	TESF2 accuracy [%]
1 (ESF)	65.99	
3	57.64 ($\sigma = 4.27$)	70.19 ($\sigma = 4.10$)
5	62.00 ($\sigma = 5.75$)	75.94 ($\sigma = 3.09$)
8	71.87 ($\sigma = 1.93$)	80.64 ($\sigma = 1.14$)
15	76.55 ($\sigma = 0.81$)	82.09 ($\sigma = 0.46$)
20 [1]	77.03	82.64

TABLE I: Influence of the number of views used to build TESF descriptors on classification accuracy. For each number of views (3, 5, 8, 15), we randomly select view indices 10 times and run leave-one-out 1-NN cross-validation. Nearest neighbors are found from an object database of 20-view TESF2 descriptors. We report mean classification accuracy and standard deviation over the 10 iterations.

Number of views n	Accuracy over 10 runs [%]
3	54.97 ($\sigma = 4.01$)
5	66.85 ($\sigma = 7.34$)
8	81.31 ($\sigma = 0.61$)
15	82.09 ($\sigma = 0.46$)

TABLE II: Using an object descriptor database containing descriptors built from fewer than 20 views. For each number of views (3, 5, 8, 15), we randomly select view indices 10 times and run leave-one-out 1-NN cross-validation. Nearest neighbors are found in a database built using the same number of views as the query objects, but differing in view indices. We report mean classification accuracy and standard deviation over the 10 iterations.

section, but this time with an object descriptor database built using the same number of views as the query: 3, 5, 8 and 15. Given that TESF2 consistently performed better than TESF1 in previous experiments, this experiment is done with TESF2 descriptors only.

In this experiment, we again measure the leave-one-out classification accuracy over 10 iterations, randomly selecting the views used for building the query objects in each iteration. The object descriptor database is built only once for each considered number of views, and view indices for building the object descriptor database are selected randomly (e.g. for 5 views they are 17, 12, 14, 19, and 2).

Results are summarized in Table II. We see that for 3 and 5 views the results are worse than when 20-view descriptors are used in the object descriptor database, while for 8 and 15 views the results are quite similar. Adequate accuracy is obtained if the object descriptor database is built using a sufficient number of views that need not necessarily include all view angles. This finding means that TESFs could be useful in scenarios where learning, i.e. building the object descriptor database, is conducted *on the robot*. Assume a robot equipped with some kind of grasper, for instance a humanoid robot with arms. If the robot is holding an object in its hand, it cannot see it from all sides without moving it to the other hand and manipulating it. However, by simply rotating its hand to obtain more views of the object, it could learn the object well enough to be able to classify similar objects later.

V. Conclusion and outlook

We have shown that TESF descriptors retain a lot of descriptivity when built using only a few views, and offer performance increases over simple matching of ESF descriptors. Our method can effectively combine any number of views into a single descriptor, and the more surface of the object is covered with the views, the better TESF performs. TESF descriptors can be compared to one another in a 1-NN classification setting even if they were built using varying number of views. This makes them especially interesting in robotic applications, where the robot sees an object of interest from a number of views t_1, but stores representations of similar objects built using t_2 views, where in general $t_1 \neq t_2$. Further analysis should be performed regarding the usability of TESF descriptors in the worst case scenario, where the robot sees an object from a limited viewpoint, i.e. from a series of very close views.

Acknowledgments

This research has been supported by the research project Research Centre for Advanced Cooperative Systems (EU FP7 #285939).

References

[1] K. Brkić, A. Aldomá, M. Vincze, S. Šegvić, and Z. Kalafatić, "Temporal Ensemble of Shape Functions," in *Eurographics Workshop on 3D Object Retrieval*, (Strasbourg, France), pp. 53–60, Eurographics Association, 2014.

[2] W. Wohlkinger and M. Vincze, "Ensemble of Shape Functions for 3D Object Classification," in *IEEE International Conference on Robotics and Biomimetics (IEEE-ROBIO)*, 2011.

[3] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3D shape descriptors," in *Symposium on Geometry Processing*, 2003.

[4] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3DNet: Large-scale object class recognition from CAD models," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 5384–5391, IEEE, 2012.

[5] M. Tenorth, S. Profanter, F. Balint-Benczedi, and M. Beetz, "Decomposing CAD models of objects of daily use and reasoning about their functional parts," in *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[6] R. Ohbuchi and T. Furuya, "Scale-weighted dense bag of visual features for 3D model retrieval from a partial view 3D model," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 63–70, Sept 2009.

[7] P. Daras and A. Axenopoulos, "A 3D shape retrieval framework supporting multimodal queries," *Int. J. Comput. Vision*, vol. 89, pp. 229–247, Sept. 2010.

[8] J. W. Tangelder and R. C. Veltkamp, "A survey of content based 3D shape retrieval methods," *Multimedia Tools Appl.*, vol. 39, pp. 441–471, Sept. 2008.

[9] Q. Liu, "A survey of recent view-based 3D model retrieval methods," *CoRR*, vol. abs/1208.3670, 2012.

[10] K. Brkić, A. Pinz, S. Šegvić, and Z. Kalafatić, "Histogram-based description of local space-time appearance," in *Proceedings of the 17th Scandinavian Conference on Image Analysis*, SCIA'11, pp. 206–217, 2011.

[11] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Trans. Graph.*, vol. 21, pp. 807–832, Oct. 2002.

[12] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, pp. 21–27, January 1967.

[13] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[14] C. Fellbaum, ed., *WordNet: an electronic lexical database*. MIT Press, 1998.

Experimental Evaluation of Multiplicative Kernel SVM Classifiers for Multi-Class Detection

Valentina Zadrija
Mireo d.d.
Zagreb, Croatia
Email: valentina.zadrija@mireo.hr

Siniša Šegvić
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia
Email: sinisa.segvic@fer.hr

***Abstract*—We consider the multi-class object detection approach based on a non-parametric multiplicative kernel, which provides both separation against backgrounds and feature sharing among foreground classes. The training is carried out through the SVM framework. According to the obtained support vectors, a set of linear detectors is constructed by plugging the foreground training samples into the multiplicative kernel. However, evaluating the complete set would be inefficient at runtime, which means that the number of detectors has to be reduced somehow. We propose to reduce that number in a novel way, by an appropriate detector selection procedure. The proposed detection approach has been evaluated on the Belgian traffic sign dataset. The experiments show that detector selection succeeds to reduce the number of detectors to the half of the number of object classes. We compare the obtained performance to the results of other detection approaches and discuss the properties of our approach.**

I. Introduction

A long-standing goal of computer vision has been to design a system capable of detecting various classes of objects in cluttered scenes. Traditionally, this task has been solved by building a dedicated detector for each class. This approach requires a significant number of examples per each class, which may not be available. In order to overcome the problem, additional partitioning into subclasses can be performed. However, the problem is that domain-based partitioning may not be optimal for the task. In case of multi-view object detection, it can also be time consuming and error prone. Therefore, it would be desirable to omit the manual partitioning stage and embed the process into the classifier itself.

Another interesting idea is to train a single classification function for all classes jointly. This approach may exploit feature sharing among classes in order to improve classification against backgrounds. The feature sharing offers great potential for: (i) improving the detection rate for classes with a low number of examples, and (ii) reducing the runtime computational complexity.

In this paper, we train a joint classification function with the multiplicative kernel as presented in [1]. However, in contrast to [1], where the authors aim to solve the detection and recognition problems at the same time, we focus on the task of detection. Once the object locations are known, object class can be determined for those locations only, thus alleviating the runtime complexity.

Multi-class detection and feature sharing is achieved by means of a non-parametric multiplicative kernel. The approach avoids the partitioning into subclasses by using the foreground training samples as class membership labels. More details are given in section III. After the training, we construct a set of linear detectors as described in section III-A. According to [1], each detector corresponds to a single foreground training sample, which makes a detector set extremely large and inefficient for the detection task. The contributions of our work are as follows: (i) we propose an efficient detector selection in order to identify a representative set of detectors out of the large initial pool as described in section III-B, (ii) we show the properties of the multiplicative kernel method and compare the results with other methods on a Belgian traffic sign dataset (BTSD) [2] as described in section IV.

II. Related Work

In recent years, a lot of work has been proposed in the area of multi-class object detection. However, the work presented in [3] achieved a significant breakthrough in the area. The authors consider multiple overlapping subsets of classes. The experiments have shown that a jointly trained classification function requires a significantly smaller number of features in order to achieve the same performance as independent detectors. More specifically, the number of features grows logarithmically with respect to the number of subclasses.

The approach presented in [4] employs a tree-based classifier structure called Cluster Boosted Tree (CBT) in order to solve the multiview detection problem. The tree structure is constructed automatically according to the weak learners selected by the boosting algorithm. The node splits are achieved by means of unsupervised clustering. Therefore, in contrast to [3], this approach does not require manual partitioning into classes, but it implies the hierarchical feature sharing.

The authors in [5] consider a classifier structure comprised out of several boosted classifiers. This approach also avoids manual partitioning into classes, but the classifiers do not share weak learners. Initially, the training data is partitioned randomly into subsets. At each round of training, the sample is (re)assigned to the subset corresponding to the classifier that has the highest probability of classifying that sample. The resulting classifiers are further transformed into decision trees, which reduce the average number of weak learner evaluations during classification.

The concept of feature sharing is also explored through shape-based hierarchical compositional models [6], [7]. Different object categories can share parts or an appearance. The

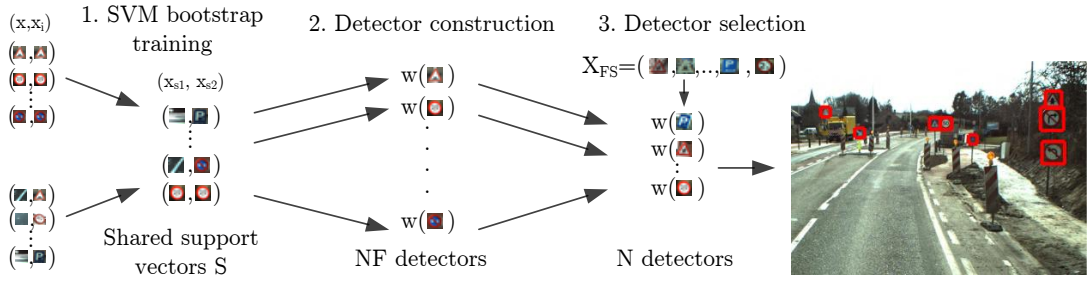


Fig. 1. An Overview of the multiplicative kernel detection pipeline. Note that we use HOG vectors instead of image patches.

parts on lower hierarchy levels are combined into larger parts on higher levels. In general, parts on lower levels are shared amongst various object categories, while those in higher levels are more specific in category.

In the more recent work [8], the authors employ Hough forest for multi-class detection. In contrast to the above methods, this approach uses the implicit shape model for detection rather than sliding window. The key feature of this approach is the separation between feature appearance and location. The appearance-based features are shared across classes providing a generalization against backgrounds. However, in order to become discriminative for individual classes, location needs to be fixed. Due to the feature sharing, the number of necessary votes grows sub-linearly with respect to the number of classes.

III. Methodology

We employ a non-parametric detection approach proposed in [1] as shown in steps one and two of Fig. 1. In the step three, we introduce a novel detector selection algorithm. The approach avoids the traditional partitioning of foreground object classes into subclasses. We treat the multi-class detection as a binary problem considering all foreground classes as a single class. Therefore, this method is also applicable in cases where the labels for foreground classes are not available. We obtain the required functionality by organizing the training samples into pairs $(x,\ x_i)$ as shown in Fig. 1. The first element corresponds to the HOG vector of either foreground or background image patch. The second element x_i corresponds to the HOG vector of the foreground image patch, $i \in \{1..NF\}$, where *NF* denotes the number of foreground training samples. The purpose of x_i is to denote the membership to foreground classes. The idea behind this concept is that the descriptors belonging to the same foreground class or subclass share certain amount of similarity. However, rather than defining the partitioning by ourselves, we let the classification function to do that job. In this way, the foreground training pairs are actually duplicated foreground HOG descriptors. However, each background sample x can be associated with any foreground sample in order to form a valid negative pair. A total number of negative pairs is therefore huge and corresponds to $NF \cdot NB$, where *NB* denotes the number of background samples x.

The classification function $C(x, x_i)$ is therefore trained jointly for all classes and provides a following decision:

$$C(x, x_i) \begin{cases} > 0, \text{x is a sign from the same class as } x_i \\ \leq 0, \text{otherwise} \end{cases} \quad (1)$$

In order to provide the above functionality, the function $C(x, x_i)$ is defined as follows:

$$C(x, x_i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot k_i(x_{s2}, x_i) \cdot k_x(x_{s1}, x) \quad (2)$$

The pair $x_s = (x_{s1}, x_{s2})$ denotes a support vector, where x_{s1} corresponds to the HOG descriptor of either foreground or background sample, while x_{s2} denotes the assigned foreground training sample. Further, α_s denotes the Lagrange multiplier assigned to the support vector. The term $k_i(x_{s2}, x_i)$ denotes the *between-foreground* kernel. The purpose of this kernel is two-fold. Firstly, it is used to measure the similarity between foreground classes, thus enabling the feature sharing. Secondly, this kernel is also responsible for the foreground partitioning, i.e. it produces higher values for the similar pairs of foreground training samples. The $k_x(x_{s1}, x)$ term denotes the *foreground-background* kernel used for separation against backgrounds.

The classification function training can be achieved through the SVM framework. Due to the memory constraints, we cannot include all $NF \cdot NB$ negative pairs in the SVM training at once. Therefore, similar to [1] and [9], we also perform bootstrap training in order to identify the hard negatives. Initially, *NB* negative pairs are chosen randomly and SVM optimisation is performed. The obtained model is evaluated for all negative pairs and false positives are added to the negative sample set. The process converges when there are no more false positives to add.

A. Detector Construction

The individual detectors $w(x, x_i)$ are constructed from the support vectors *S* by plugging the specific foreground sample values x_i into (2). With the known value of a parameter x_i, the value of *between-foreground* kernel can be precomputed. We have chosen the non-linear Radial Basis Function (*RBF*) kernel for k_i. On the other hand, the *foreground-background* kernel k_x is evaluated at detection time, so it should be efficiently computed. Therefore, we have chosen the linear kernel $k_x = x^T \cdot x$ for that purpose. According to the chosen kernels, we obtain the following detector $w(x, x_i) = w(i) \cdot x$, where $w(i)$ denotes the vector of linear classifier weights:

$$w(i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot RBF(x_{s2}, x_i) \cdot x_{s1}^T \quad (3)$$

The feature sharing is achieved through support vectors *S*. The obtained number of the detectors corresponds to the number of

Given
1: Foreground samples $X_{FS} = \{x_j\}$, $j \in \{1 \dots NS\}$
2: Detectors $W = \{w(i)\}$, $i \in \{1 \dots NF\}$
Initialize
3: Evaluate W on X_{FS}: to each example x_j, assign a set of detectors $E(x_j)$ which detect that example
TRAVERSE(X_{FS}, W):
4: Find the example x_m with the maximum nr. of detectors
5: **repeat**
6: Find the detector $w(r) \in E(x_m)$ with the minimum nr. of detections
7: **while**
8: **if** removing $w(r)$ doesn't result with false negatives
9: Remove $w(r)$ from W, break
10: **else**
11: Find the next detector $w(r) \in E(x_m)$ with with minimum nr. of detections
12: **if** $w(r)$ is not found
13: Proceed to the next example x_m with maximum nr. of detectors
14: **else**
15: Find the example x_m with the maximum nr. of detectors
16: **until** all examples are traversed
17: **return** selected set of detectors W

Fig. 2. Detector selection algorithm.

foreground training samples *NF*. Evaluating all the detectors at runtime is related to the *k*-nearest neighbours (*k*-NN) method [10], with a parameter $k = 1$, i.e. the object is simply assigned to the class of the single nearest neighbour selected among all detectors. This approach would result in extremely slow detection. Therefore, the number of detectors needs to be reduced somehow. The authors in [1] propose clustering. On the other hand, we argue that the selection algorithm presented in the next section is a better approach.

B. Detector Selection

We apply detector selection according to the pseudo-code outlined in Fig. 2. The goal is to remove detectors which do not contribute to the overall detection score. The resulting set of detectors obtains the same recall as the original set of detectors on the selection samples. The set of samples X_{FS} is comprised out of foreground HOG vectors as shown in step three of Fig. 1. Note that selection and training datasets are disjoint. We traverse the selection examples according to the number of detectors which detect particular example. In each round, we consider the example with the maximum number of detectors (*line 4*). The algorithm is greedy, i.e. among the detectors that detect that example, we select the detector with the lowest number of detections (*line 6*) as a candidate for removal. The condition in *line 8* ensures that the recall remains unchanged by removing the detector. There is a possibility, that removing every detector from the $E(x_m)$ would result in a false negative (*line 12*). In that case we proceed to the next example with the lowest number of detectors (*line 13*). Due to the fact that the algorithm is greedy, it may get stuck in a local optimum. However, as the experiments in section IV show, it achieves a very high performance rate.

Fig. 3. Visualisation of the support vectors for the multiplicative kernel [12].

IV. CASE STUDY

As a case study, we apply the described methodology for traffic sign detection. Traffic signs have been chosen because they are characterized by a very large variation with respect to the sign shape and the type of the ideogram. We use a subset of the BTSD [2] containing 19 different classes shown in Fig. 4, Fig. 5 and Fig. 7. The selected classes capture a representative subset of traffic signs: triangular with a red rim, circular with a red rim and white or blue interior as well as the rectangular blue signs.

A. Implementation Details

There are 1024 sign images used for training and 393 sign images used for detector selection. There are also 11200 background training patches extracted from 16000 background images. The HOG vectors [11], computed from training images, are cropped to 24 $\times$ 24 pixels. We use 10-fold stratified cross-validation to determine the model parameters.

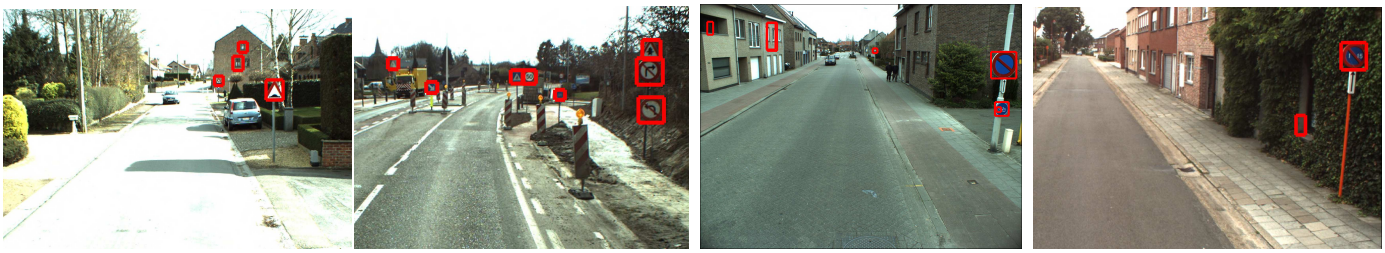
The test dataset contains 1027 images in 1628 $\times$ 1235 resolution. There are approximately 3 images per a single physical traffic sign. The training and test datasets are disjoint. We use the sliding window approach and scan the image using the 1.05 scale step. We also consider 5 aspect ratios, i.e. from ratio 1 to 3 with the step of 0.4 (height/width). We set the scale space search range from 24 $\times$ 24 to 330 $\times$ 435 pixels.

B. Results

All experiments are performed on a 3.4 GHz Intel Core i7-3770 CPU. We perform the bounding box evaluation according to the scheme laid out in PASCAL object detection challenges [13]. In order to evaluate performance, we use the area under the precision-recall curve (*AuPR*) and area under the receiver operating characteristics curve (*AuROC*). As indicated in [14], the *AuPR* metric gives a more informative picture of a method's performance for highly skewed datasets. The sliding window approach employed in this work generates a large number of background candidate windows and only a small number of windows containing a foreground object. Therefore, the *AuPR* measure would seem to be the most relevant choice for the metric. However, as the results in the section IV-B1 show, an algorithm that optimizes the *AuPR* is not guaranteed to optimize the *AuROC* measure. More specifically, the method with the maximum *AuPR* may not be the optimal choice because of the very poor recall (detection rate). In addition, we also report the detection rate and false positive rate per image (*FP/img*). The rest of the section is organized as follows. We discuss the multiplicative kernel results in section IV-B1. Further, we compare these results with other approaches in section IV-B2.

(a) Distribution of selected detectors for *Mult. Selected* (b) Distribution of detectors for *Mult. Clustered* (c) Distribution of false negatives for *Mult. Selected* (d) Distribution of false negatives for *Mult. Clustered*

Fig. 4. A comparison of detector selection (*Mult. Selected*) and clustering (*Mult. Clustered*) for multiplicative kernel.



(a) Speed bump (b) Right of the way at the next crossroads (c) No parking allowed (d) No parking from the 1st to the 15th of the month

Fig. 5. Example of detections for traffic signs without an assigned detector for *Mult. Selected* method. All instances of these signs are detected in test images.

1) Multiplicative Kernel Results: The SVM training described in section III yields a shared set of support vectors containing 653 vectors out of which 23% corresponds to the positive training pairs. Fig. 3 shows the examples of support vectors obtained from the original image patches according to the procedure described in [12]. The support vectors correspond to the image patches containing the traffic signs captured under difficult conditions, i.e. rotated signs and hard perspective views. According to the obtained support vector set, 1024 linear detectors are constructed. The number of detectors is then reduced using the following methods:

a) Multiplicative Selected: This method employs the detector selection algorithm as described in section III-B.

b) Multiplicative Clustered: This method employs the clustering as proposed in [1]. As a clustering method, *k*-medoids with Euclidean distance is used. The purpose is to determine if the detector selection is really necessary or the problem can be solved with clustering.

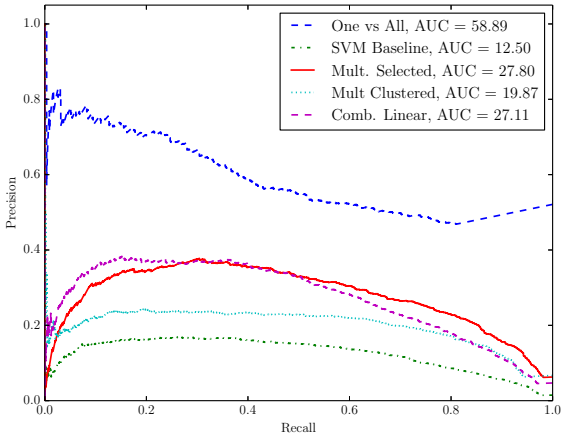
The results shown in Fig. 4 and Fig. 5 point out three important consequences which indicate that the *Multiplicative Selected* outperforms the *Multiplicative Clustered* approach.

Firstly, the selection approach converges with the number of detectors (10) which is almost 50% percent smaller than the number of classes (19). This suggests sub-linear detection complexity. The target number of cluster centres was then configured with the same value in order to provide a fair comparison. The distribution in Fig. 4a points out that the *Parking reserved for disabled people* is the most difficult sign for detection using the *Multiplicative Selected* approach. Hence, there are three detectors assigned to detect that sign. In general, rectangular signs are assigned five detectors, triangular three and circular only two signs. This would suggest that circular signs benefit the most from feature sharing. This is quite different from the detectors obtained by clustering, where the circular signs exhibit the maximum number of detectors (5) as shown in Fig. 4b.

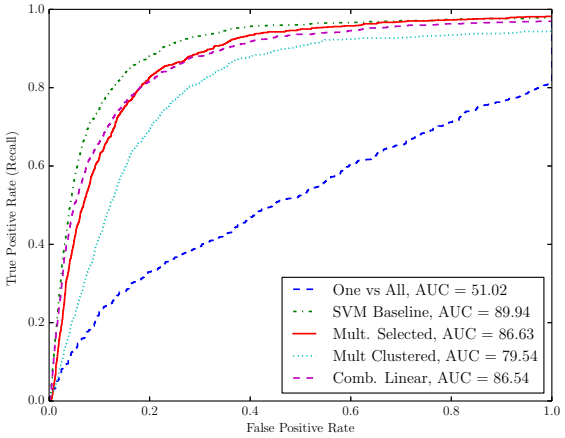
Secondly, the selection approach yields a better set of detectors than the clustering. The false negative distribution shown in Fig. 4c and Fig. 4d supports that fact. The number of false negatives obtained by the *Multiplicative Clustered* method (62) is 3.1 times larger than the one obtained using the *Multiplicative Selected* (20). In addition, despite the fact that the *Speed limit* is assigned the maximum number of cluster centres in *Multiplicative Clustered* method, this particular sign exhibits the maximum number of false negatives (33). On the other hand, selection approach yields only one detector for the *Speed limit* class. However, almost all *Speed limit* signs are detected as shown on Fig. 4c. This suggests that the detectors obtained by clustering are not optimal for a specific class and that the selection is a better option. Further analysis of the false negatives for the *Multiplicative Selected* method shows interesting results. One would expect that most of false negatives belong to the classes which do not have a corresponding detector. However, this is not entirely true, i.e. 9 out of 20 false negatives belong to such classes. The signs with the assigned detector, *Parking allowed for all vehicles*, *Speed limit* and *Intersection with priority to the right* have the maximum number of false negatives (3). However, a detailed analysis shows that these false negatives are a consequence of difficult conditions, rather than the fact that they are "hard to detect". Typical examples include perspective views, worn-out, occluded or difficult annotations as shown on Fig. 7.

Thirdly, there are 4 classes without an assigned detector that do not have any false negatives shown in Fig. 5. Note that Fig. 5b and Fig. 5c also contain detections of other signs. This shows that the feature sharing improves performance with respect to the dedicated detector approach.

2) Comparison with Other Approaches: We compare the performance of the multiplicative kernel with the following methods. All methods are implemented in C++ using the SVM-Light library [15] for the SVM training.



(a) PR curve



(b) ROC curve

Fig. 6. Comparison with other approaches.

a) One vs All: This configuration includes 19 class-specific linear SVM detectors. Traffic signs are not used as negatives.

b) SVM Baseline: This is a baseline jointly-trained approach. The classification function employs the linear SVM which treats all traffic sign classes as a single foreground class:

$$C(x) = \sum_{x_s \in S} \alpha_s \cdot x_s^T \cdot x \tag{4}$$

The parameter x_s denotes a support vector, which is a HOG vector of either foreground or background image patch.

c) Combined Linear: This approach is similar to the multiplicative kernel because it also uses the notion of foreground training samples as class memberships. The classification function employs a single linear kernel with a concatenated vector $[x, x_i]$ as a parameter, where x denotes a feature vector which we want to classify and x_i a foreground training sample.

$$C(x, x_i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot [x_{s1}, x_{s2}]^T [x, x_i] \tag{5}$$

The parameter $x_s = (x_{s1}, x_{s2})$ denotes a support vector. This method also produces a set of detectors $w(x, x_i)$, each corresponding to the specific foreground example x_i. Similar as in section III-A, individual detectors are constructed by plugging the x_i value into (5). We obtain the following expression for the detector $w(x, x_i) = w \cdot x + b(i)$, where:

$$w = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot x_{s1}^T \tag{6}$$

$$b(i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot x_{s2}^T \cdot x_i \tag{7}$$

In contrast to (3), where the weight vector depends of the foreground sample x_i, the equation (6) shows that this value is the constant for all detectors. As a consequence, the detector with the maximum value $b(i)$ exhibits the same recall value as the entire set of detectors. Therefore, in order to make the detection process practical, we use only that detector at runtime. The purpose of this method is to assess if the multiplicative kernel is really necessary to solve the problem.

TABLE I. PERFORMANCE OF DIFFERENT DETECTION APPROACHES

Method	N	AuPR	AuROC	Det. rate	FP/img
One vs All	19	58.8	51	81.2	0.9
SVM Baseline	1	12.5	89.9	97.8	69.8
Mult. Selected	10	27.8	86.6	98.2	15.6
Mult. Clustered	10	19.8	79.5	94.4	14.5
Comb. Linear	**1**	27	**86.5**	96.9	21.2

Fig. 7. Examples of false negatives for the *Multiplicative Selected* method.

Table I and Fig. 6 show the comparison results. For each method, we report the number of employed SVM linear detectors *N*. Lower number of detectors enables faster detection. The results point out four important facts.

Firstly, the individually trained *One vs All* method produces the best results in the PR space with respect to the other jointly trained methods. However, this is not a relevant measure because this method exhibits the lowest detection rate (81.2%) and the lowest *AuROC* value (51). Fig. 6b shows that this method performs worse than chance for higher FPR values. This is a result of the fact that certain true positives have lower detection score than false positives. This proves that feature sharing increases the performance in the ROC space.

Secondly, among the jointly trained methods, the *SVM Baseline* yields the best result in ROC space at the cost of the worst performance in the PR space (12.5). This method exhibits almost 70 FP per image. Therefore, it does not produce a good separation between foregrounds and backgrounds and it is not applicable for the detection task. Further, all jointly trained methods exhibit a low *AuPR* value. One possible way to improve the performance of these methods would be

to apply the cascading approach. For example, training the *Multiplicative Selected* on hard negatives obtained from the first stage *SVM Baseline* classifier would decrease the number of FP and increase precision.

Thirdly, the selection algorithm employed in the *Multiplicative Selected* outperforms clustering in the *Multiplicative Clustered* method [1] in both ROC and PR space for 7% and 8%, respectively.

Fourthly, the multiplicative kernel employed in the *Multiplicative Selected* produces marginally better results with respect to the linear kernel of the *Combined Linear* in both PR and ROC space. In addition, the *Combined Linear* method is 10 times faster than the *Multiplicative Selected*. However, the recall of the single detector of the *Combined Linear* method is equivalent to the recall of the complete detector set produced by the equations (6) and (7). On the other hand, the recall of the *Multiplicative Selected* could be improved by increasing the number of detectors produced by the detector selection procedure. The target number of detectors N could be used as a stop condition in pseudo-code given in Fig. 2. To conclude, the *Combined Linear* yields the acceptable performance in ROC space at the low detection complexity. This result is somewhat different than the original hypothesis that multiplicative kernel is necessary in order to provide both feature sharing between foregrounds as well as the separation against backgrounds.

V. Conclusion

We explore the properties of the multiplicative kernel [1] for multi-class detection. In order to evaluate performance, we use the Belgian traffic sign dataset [2].

There are several benefits of the multiplicative kernel approach. Firstly, the experiments show that feature sharing increases the *AuROC* up to 35.6% with respect to the individual detector approach. This is because the individual detectors require a substantial number of samples per class to achieve good performance. Secondly, this approach avoids the partitioning into subclasses by using the foreground training samples as class membership labels and the multiplicative kernel. As a consequence, the number of obtained detectors equals to the number of foreground samples. Thirdly, we use a selection procedure in order to determine a representative set of detectors from the large initial detector pool. The experiments show that our approach converges with the number of detectors which is half the size of the number of classes. This suggests sub-linear detection complexity. In addition, detector selection yields better results than the original clustering approach [1].

We also discuss the following issues. Firstly, whether the multiplicative kernel is necessary in order to provide feature sharing and separation against backgrounds. In contrast to our initial hypothesis, the multiplicative kernel obtains marginally better results with respect to the linear kernel based upon the same concept of foreground samples as class memberships. This suggests that this task could also be efficiently solved with the linear kernel approach which is 10 times faster. Secondly, we are also interested in the method precision. The results in the PR space show that the multiplicative kernel yields maximal *AuPR* of 27.8%, which is half the precision obtained by the individual detector approach. Therefore, for future work, we propose the cascading approach in order to increase the precision.

Acknowledgment

This work has been supported by research projects Vista (EuropeAid/131920/M/ACT/HR) and Research Centre for Advanced Cooperative Systems (EU FP7 #285939). The authors wish to thank Thomas Mensink for useful suggestions regarding the choice of *Combined Linear* approach for comparison.

References

[1] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff, "Learning a family of detectors via multiplicative kernels," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 3, pp. 514–530, 2011.

[2] M. Mathias, R. Timofte, R. Benenson, and L. J. V. Gool, "Traffic sign recognition - how far are we from the solution?" in *IJCNN*. IEEE, 2013, pp. 1–8.

[3] A. Torralba, K. Murphy, and W. Freeman, "Sharing visual features for multiclass and multiview object detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 5, pp. 854–869, 2007.

[4] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real adaboost," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, 2004, pp. 79–84.

[5] T.-K. Kim and R. Cipolla, *Multiple classifier boosting and tree-structured classifiers*. Berlin: Springer, 2013, pp. 163–196.

[6] S. Fidler and A. Leonardis, "Towards scalable representations of object categories: Learning a hierarchy of parts," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–8.

[7] L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille, "Part and appearance sharing: Recursive compositional models for multi-view," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 1919–1926.

[8] N. Razavi, J. Gall, and L. Van Gool, "Scalable multi-class object detection," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1505–1512. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2011.5995441

[9] V. Zadrija and S. Segvic, "Multiclass road sign detection using multiplicative kernel," *CoRR*, vol. abs/1310.0311, 2013.

[10] D. Bremner, E. Demaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Toussaint, "Output-sensitive algorithms for computing nearest-neighbour decision boundaries," in *Algorithms and Data Structures*, ser. Lecture Notes in Computer Science, F. Dehne, J.-R. Sack, and M. Smid, Eds. Springer Berlin Heidelberg, 2003, vol. 2748, pp. 451–461. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45078-8_39

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 886–893 vol. 1.

[12] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba, "Hoggles: Visualizing object detection features," in *ICCV*. IEEE, 2013, pp. 1–8.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[14] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 233–240. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143874

[15] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 169–184. [Online]. Available: http://dl.acm.org/citation.cfm?id=299094.299104

Point Cloud Segmentation to Approximately Convex Surfaces for Fruit Recognition

Robert Cupec, Damir Filko, Ivan Vidović, Emmanuel Karlo Nyarko, Željko Hocenski
Faculty of Electrical Engineering
J. J. Strossmayer University of Osijek
Osijek, Croatia
robert.cupec@etfos.hr

***Abstract*—A fruit recognition approach based on segmenting the point cloud acquired by a 3D camera into approximately convex surfaces is considered. A segmentation approach which transforms a depth image into a triangular mesh and then segments this mesh into approximately convex segments is applied to depth images of fruits on trees. An analysis of the results obtained by this approach is performed with the intention to determine how successful the studied method is in detecting fruit as separate objects in a point cloud. The reported analysis gives a valuable insight into the potential applicability of the tested methodology in the preprocessing stage of a fruit recognition system as well as its drawbacks.**

Keywords—fruit recognition;3D camera; convex sets; segmentation

I. INTRODUCTION

The very purpose of robotics and automation is to free the humans from heavy, tedious and repetitive work. Although many production tasks which had been done by humans in the past are today performed by machines, there are many operations in industrial and agricultural production process which still require human involvement due to the lack of suitable technical solutions. Tasks like textile handling, fruit picking and many assembling tasks, although rather simple for a human, represent a real challenge for a machine. Solving such tasks requires highly advanced environment perception capable of interpreting unstructured scenes and objects of varying shapes.

In this paper, robotized fruit picking problem is considered. In order to solve this task, a robot must be capable of recognizing fruit on trees, usually occluded by leaves and branches and under different lighting conditions. Unlike office or household objects which are commonly used as case studies in object recognition research [28] – [30], the shape, size and color of fruit varies among samples of the same sort, which makes their recognition more difficult. A robust and efficient fruit recognition method should exploit both shape and color properties of fruit.

The fruit recognition approach studied in this paper relies on the fact that the shape of many fruit sorts can be approximated by convex shapes. The proposed approach consists of segmenting the point cloud acquired by a 3D camera into approximately convex surfaces and applying an appropriate classifier based on color and shape in order to distinguish between the fruit and other objects such as leaves and branches. The focus of this paper is on the segmentation stage, while the selection of an appropriate classification method is not covered. The results of the point cloud segmentation method proposed in [1] are analyzed in order to assess its applicability in a fruit recognition scheme. In an ideal case, the applied method should provide such point cloud segmentation where each fruit is represented by a single approximately convex surface. In reality, however, due to a limited sensor accuracy and measurement noise, the points representing a fruit can be merged with the points of adjacent leaves or branches into an approximately convex surface which exceeds the boundaries of the considered fruit. Furthermore, due to occlusion and gaps in the point cloud caused by light absorption effects and specular reflection, the set of points representing a single fruit can be split into two or more mutually disconnected subsets resulting in oversegmentation. In order to evaluate the applicability of the investigated segmentation approach, an experimental analysis is performed whose goal is to determine the percentage of successful segmentation results, i.e. the cases where a fruit is represented by a single segment, as well as the percentage of false segmentation results, where a single fruit is represented by two or more segments or included in a segment which extends to adjacent objects.

The paper is structured as follows. The overview of the research in the field of fruit recognition is given in Section II. The segmentation method which is in focus of this paper is described in Section III. In Section IV, an experimental analysis of the applicability of the considered segmentation method for usage in a fruit recognition system is presented. The discussion of the obtained results and some possible directions for future research are given in Section V.

II. RELATED RESEARCH

Fruit detection, recognition or localization on trees has been subject of many scientific papers. Fruit recognition systems are used for different applications, like automatic harvesting, yield estimation, phenotyping, breeding, etc. There are many different approaches to solve the mentioned problems and some of them are listed below.

One of the first papers in this field is [2] and it proposes a method for fruit size and location detection. The method is based on image color information and thus retraining is necessary every time when fruit color changes significantly. The detection accuracy under favorable conditions can reach up to 75 %. Another color based approach for automatic recognition of Fuji apple based on red color difference and finding the maximum between-class variance in histogram is given in [3]. Unlike [2], the algorithm is not sensitive to lightening conditions, but because it relies on red color difference it is limited to Fuji apples or similar color fruit recognition. The obtained successful recogniton was over 88%, but average error rate reached up to 18%. Detection of citrus fruits using fusion of color map and luminance map is proposed in [4]. The recognition accuracy under different lightening conditions was up to 86,81% with a false detection rate of 2,25%. Forward feature selection algorithm (FFSA) for color feature selection is used in [5]. Different classifiers are applied on selected features to seperate blueberry fruit from the background and to classify the detected fruits according to matureness. A mature citrus recognition method based on YCbCr color model is proposed in [6]. Using an improved fuzzy C-means algorithm, morphological operations and optimized Circular Hough transform (CHT), fruits center and radius are detected. Algorithm for citrus fruit and obstacles recognition based on multi-class support vector machine applied on color features is proposed in [7]. The recognition rate was 92,4% and branches with diameter larger than 5 pixels were also detected.

Instead of color features, authors in [8] used fruit shape prosperities for wine grape berries recognition in controled environment. After the image preprocessing, CHT is applied for berries detection. Wine grape berries sizes estimation is discussed in [9]. First berries are detected using CHT, then histogram of oriented gradients (HoG) and gist features are extracted followed by application of conditional random field for classifying circles as berry and non-berry. Experiments showed on average 1 mm difference of estimated size compared to manual measurement. The authors also mentioned possible usage of depth image to precisely determine berries size. Immature citrus recognition and counting based on shape analysis and texture classification is proposed in [10]. Shape analysis based on CHT is used to detect circles while texture classification is used for false positives removal. Algorithms used for texture analysis are: support vector machine (SVM), Canny edge detection combined with a graph-based connected component algorithm, Hough line detection and scale invariant feature points (SIFT) algorithm. The method is tested on images captured in natural outdoor conditions and recognition and counting rate was 80,4%.

In [11] a method for recognizing and counting peppers in cluttered greenhouse is presented. The proposed method is a two-step approach based on detecting peppers from multiple views and applying a statistical method for combining the detection results. Experiments are performed on 28 000 images and 74,2% accuracy of detection is achieved. A new method for grapevine berries size and weight determination and it's extension to detection of other fruits is proposed in [12]. The first step of the proposed method is peduncle detection and it is based on a novel signature of the contour. The reported experiments show that the grapevine berry weight and size can be correctly estimated with correlation coeficient $R^2 > 0,96$ and $R^2 > 0,97$ respectively. For other fruits, estimation of the size is also accurate with $R^2 > 0,93$. A vision system for automatic apple harvesting is proposed in [13]. The method consists of four steps: preprocessing by vector median filter, image segmentation by seeded region growing method using color features, color and shape feature extraction and fruit detection using SVM. In the reported experiments 89% of apples were successfully detected. The idea of applying convexity analysis in fruit detection is used in [14], where intensity model is created from 2D image and a convexity test is applied to that model in order to detect apples on trees. The proposed method detected 94% of visible apples, while 14% of the identified objects were false positives.

Apple fruit counting and diameter estimation using thermal image is proposed in [15]. The captured image is transformed using normal difference index (NDI) and after that morphological operations, feature extraction, particle selection and classifier evaluation are applied. The correlation coefficient with manual measuring was 0,88 and 0,70 for apple counting and diameter estimation respectively. A drawback of the method is recognizing the fruits growing deep in the tree-crown and the authors sugest usage of some shape detection algorithm for the mentioned problem. Fusion of vision and thermal images for oranges detection is proposed in [16]. After image registration, Laplacian pyramid transform (LPT) and fuzzy logic are used for image fusion. It is shown that fuzzy logic performs better and that image fusion improves fruit detection when visible image was over-exposed or the fruit was warmer.

In addition to the approaches mentioned so far which are based on 2D images, many methods which use stereo vision and 3D sensors are also proposed. In [17] stereo vision is used for apple and pear detection and automatic harvesting. Detection is based on a color intensity threshold and Linear Color Models (LCM). The method is tested only in controlled laboratory conditions. To overcome problems with uneven illumination, partly occluded surfaces and similar background features, the authors in [18] used a combination of the object's color, texture and 3D shape properties. Recognition of clustered tomatoes using binocular stereovision is proposed in [19]. The method is based on depth map filtering, Otsu thresholding and edge curvature analysis. The recognition accuracy of the clustered tomatoes was 87,9% when the leaf or branch occlusion rate was less than 25%. Localization of oranges for automatic harvesting is proposed in [20]. Harvesting is based on matching of oranges detected in stereo images acquired from two robot arms and on applying double traveling salesman problem (DTSP) for arm path computation.

A laser based computer vision system for automatic orange harvesting is proposed in [21]. Both range and reflectance information is used to generate four characteristic primitives: contour pixels, crown pixels, convex regions and reflectivity-based regions. The generated primitives are used for spherical object recognition and localization. The main drawback of the proposed method is the image acquisition time which is 25 seconds for a 160 x 160 image. Another laser based approach is

proposed in [22]. The applied sensor is equipped with red and infrared laser diodes and both beams scan an object simultaneously. Cherries are detected using the difference in the spectral reflection characteristics between the laser beams.

Good surveys on computer vision methods and algorithms used in automation of fruit production process can be found in [23] – [26].

Analogously to the research presented in [14], the approach proposed in this paper relies on the convexity property of the fruit. However, instead of analyzing intensity profiles between two edge points in order to detect convex surfaces in 2D image, we use a 3D camera and detect convex surfaces in the obtained point clouds. We believe that by application of a 3D camera a more robust performance can be achieved, because the detection of convex surfaces in point clouds acquired by such cameras is less dependent on illumination conditions.

In comparison to the work presented in [21], where laser sensors are applied to detect geometric structures in the scene, we investigate the application of a low-cost 3D camera which acquires point clouds with a frame rate of 30 fps. Thereby, a much faster detection of convex structures is achieved. On the other hand, the sensor used in the experiments presented in this paper provides less accurate measurements then the equipment applied in [21] and we want to determine how useful the measurements obtained by such sensor are in the context of fruit recognition.

III. Segmentation of 3D Point Clouds into Approximately Convex Surfaces

A point cloud obtained by a 3D camera such as Kinect or PrimeSense sensor represents a set of 3D points lying on the scene surfaces visible from a particular viewpoint. Each of these points is the closest point to the camera in a particular direction, i.e. there are no solid non-transparent objects between a point in a considered point cloud and the camera. Hence, each point in the point cloud projects onto the different point on the camera image plane. Such a point cloud can therefore be regarded as a *depth image*, i.e. an image where each pixel is assigned its distance from the camera. An example of an RGB image and the corresponding depth image is shown in Fig. 1. (a) and (b) respectively. The depth image is presented in grayscale, where the pixel intensity is proportional to the depth except in the case of black color which represents the pixels which are not assigned a depth value.

The efficient point cloud segmentation into approximately convex surfaces proposed in [1] requires a preprocessing step in which a *triangular mesh* is built from the point cloud. A triangular mesh is a representation of a point cloud by a set of triangles such that for each point in the point cloud there is a triangle whose distance to this point is less than a predefined threshold value. By representing a point cloud with a triangular mesh a reduction of data is achieved since a triangle defined by three points usually represents a set of many points of the point cloud. This data reduction results in computation time saving. It is especially efficient for scenes dominated by low curvature surfaces, e.g. indoor scenes with large planar surfaces. To generate a triangular mesh from a point cloud we use the algorithm based on recursive triangulation refinement resulting

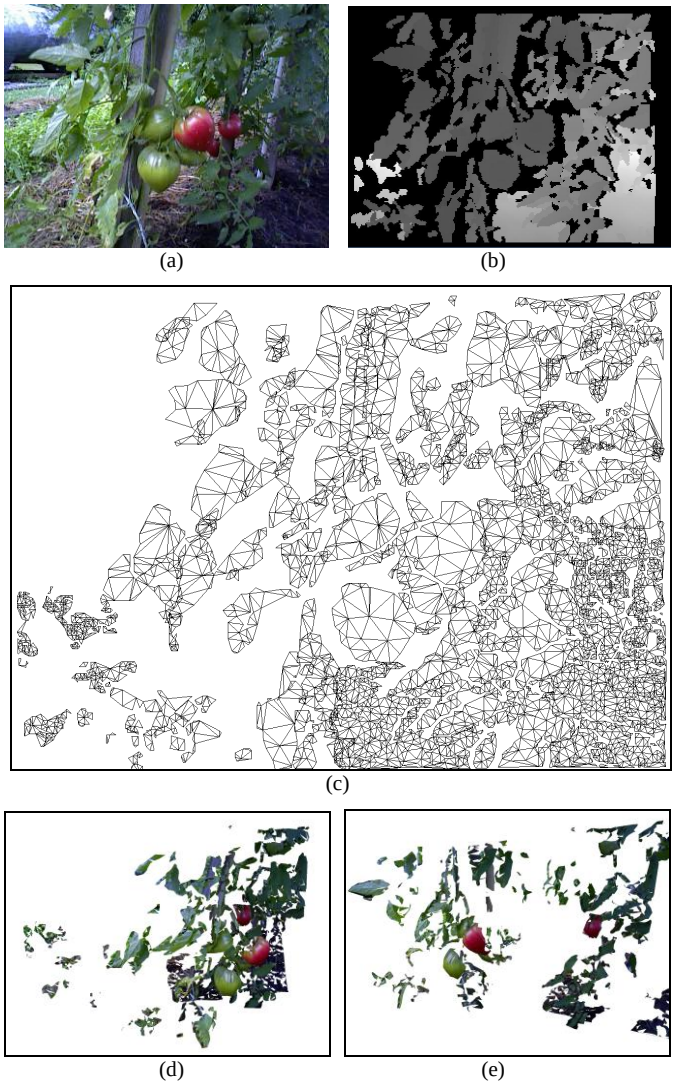


Fig. 1. (a) Sample RGB image; (b) corresponding depth image; (c) triangular mesh; (d), (e) triangular mesh with the RGB image projected onto it presented in two different views.

in a Delaunay triangulation proposed in [27]. After the triangulation is completed, each triangle for which the angle between its normal and the optical ray corresponding to one of its vertices is greater than 73° is rejected from the mesh because such triangles mostly represent depth discontinuities rather than an object surface. Furthermore, all triangles with less than 50% supporting image points not assigned depth are also rejected from the mesh, because they mostly represent artefacts bridging gaps in the depth map rather than true surfaces present in the scene. The triangular mesh created from the depth image shown in Fig. 1. (b) is presented in Fig. 1. (c). The obtained 3D model of the scene is visualized in Fig. 1. (d) and (e) by projecting the RGB image shown in Fig. 1. (b) onto the triangular mesh shown in Fig. 1. (c).

The triangular mesh is then segmented into approximately convex surfaces. A method which builds a hierarchical convex approximation of 3D shapes given a tetrahedral mesh is proposed in [31]. Since a software implementation of that method is not publically available, it is hard to assess its

computational efficiency. Nevertheless, from the data provided in [31], we estimated that the method proposed in [1] is faster, hence we used this method.

The procedure for segmentation of a triangular mesh applied in [1] is described in the following. First, the largest triangle is selected. This triangle represents the initial single-triangle set which is expanded by an iterative region growing procedure. In each iteration, a triangle adjacent to one of the triangles in the expanding set is added to this set, while preserving the property of the set that the distance of every triangle in the set from the convex hull of the set is below a predefined threshold. The region growing procedure stops, when no additional triangle can be appended to the considered set without losing the aforementioned approximate convexity property. After one segment is completed, the algorithm continues applying the same procedure to the remaining triangles which are not assigned to any segment. The described segmentation approach is presented as Algorithm 1 given in the following.

Algorithm 1: Point Cloud Segmentation into Approximately Convex Surfaces

Input: Depth image, ε
Output: Set of segments Σ

1 Create triangular mesh A from the input depth image using the recursive triangulation refinement method proposed in [27].
2 $A' \leftarrow A$.
3 **Repeat**
4 $F \leftarrow$ triangle from A' with the greatest area.
5 Remove F from A'.
6 $M \leftarrow \{F\}$.
7 $\Gamma \leftarrow$ set of all edges of F.
8 **Repeat**
9 $E \leftarrow$ edge from Γ for which the angle between the normals of the triangles sharing that edge is the smallest.
10 Remove E from Γ.
11 $F' \leftarrow$ triangle from A' which is not in M and one of its edges is E.
12 **If** F' is adjacent triangle to two or three triangles from M **then**
13 Add F' to M.
14 **else**
15 $M' \leftarrow M \cup \{F'\}$.
16 Compute convex hull $H_{M'}$ of M'.
17 **If** the distance between M' and $H_{M'}$ is not greater than ε **then** $M \leftarrow M'$.
18 **end if**
19 **If** F' is added to M **then**
20 Remove F' from A'.
21 Add all edges of F' which are on the boundary of M to Γ.
22 **end if**
23 **until** Γ is empty
24 Add M to Σ.
25 **until** A' is empty
26 **Return** Σ.

A detailed explanation of how to compute the convex hull in line 16 and how the distance between a segment and its convex hull in line 17 is defined is given in [1]. The output of the presented algorithm is the set Σ of segments representing approximately convex surfaces. How close a surface must be to its convex hull in order to be considered approximately convex is specified by a user defined parameter ε. An example of segmentation to approximately convex surfaces is shown in Fig. 2, where segments obtained by applying the described method to the triangular mesh shown in Fig. 1. (c) are displayed.

IV. Experiment

In this section an analysis of the results obtained by applying the segmentation algorithm proposed in [1] to RGB-D images of fruit on trees is presented. Several sequences of RGB-D images were taken in an orchard by a hand held PrimeSense Carmine 1.09 short range sensor operating in 100μm-mode. The image resolution was 320 × 240. From these sequences four sets of images were selected, each set containing images of one of the following fruit sorts: plum, nectarine, pear and tomato. One of the criteria for image selection was that the images had been taken while the camera was steady or moving very slowly in order to avoid motion blur or misalignment of depth and RGB image. The other criterion was that the fruit in the image is not deep in a shadow so that it can be unambiguously distinguished by visual inspection. Although the evaluated method uses only depth images, RGB images are used for visual inspection of the results by a human evaluator and therefore, clear visibility of the fruit in the image is very important in order for the evaluator to be able to correctly categorize the results obtained by the evaluated approach.

The investigated segmentation approach described in Section III was applied to the four sets of acquired RGB-D

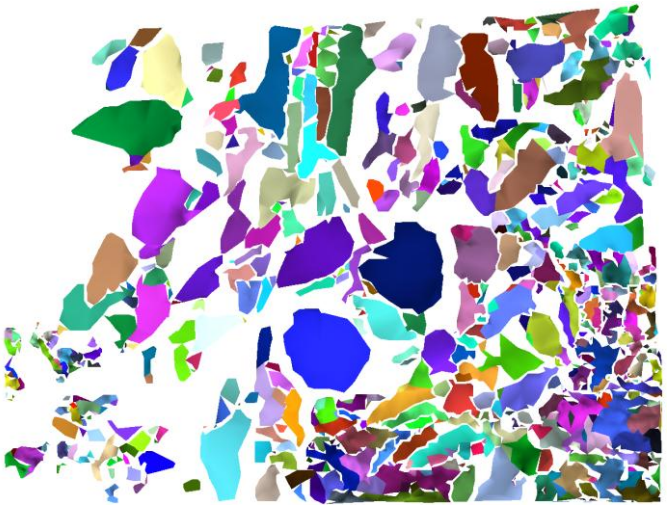


Fig. 2. Segmentation of the triangular mesh shown in Fig. 1. (c) into approximately convex surfaces. Each segment is displayed in a different color.

images. Since the precision of the applied sensor degrades with the distance, the tested algorithm was configured to consider only the points of the acquired point clouds within the distance of 0.6 m from the camera. Hence, in the following, the term point cloud refers to this close subset of an original point cloud. The threshold for generating triangular meshes from the depth images was set to 2 mm, which means that the maximum distance between any point in the point cloud and the closest triangle of the mesh in the direction of the camera optical axis is 2 mm at most. The segmentation threshold ε explained in Section III was set to 5 mm.

The obtained segmentation results were visually inspected by human evaluators which categorized all segments representing fruits into three categories:

C1: the fruit is represented by a single segment where the boundary of this segment is very close to the fruit boundary;

C2: the fruit is represented by two or more segments, where the convex hull of these segments is very close to the fruit boundary;

C3: the fruit is represented by a segment whose boundary extends over a significant portions of adjacent objects.

The results of this analysis are presented in TABLE I.

Examples of three considered segmentation categories of possible segmentation result are shown in Fig. 3. For illustrative purposes, one segment is manually selected between all segments obtained by the evaluated approach in each example. In the top row a correct segmentation is shown. It can be seen in the 3D display (right) that although two pears are positioned very close to one another, the tested segmentation algorithm separated them into two segment according to their approximately convex shape. The middle row of Fig. 3. is an example of oversegmentation, where the visible surface of the tomato is represented by two segments because of a concavity appearing on the top of the fruit. An example of false segmentation, where a fruit is merged together with adjacent leafs into one segment, is shown in the bottom row of Fig. 3. In this example, the plum is positioned among leafs in such a way that its surface aligns with their surfaces forming an approximately convex surface.

The average execution time on a standard PC with an Intel Core i5-3230M CPU at 2.6GHz measured over a sequence of 1433 images was 0.114 s.

From the presented analysis it can be concluded that there are a high percentage of samples where fruits are merged with adjacent leaves or branches (C3). Such merging occurs e.g. in the case where a leaf covers a fruit tightly so that the union of the point sets representing the fruit and the leaf is an approximately convex surface. This indicates that additional cues such as color and shape constraints must be used in order to achieve reliable segmentation. Furthermore, a significant percentage of fruit samples are oversegmented (C2). This can be caused by small concavities in the fruit shape or sensor noise. These effects could be reduced by increasing the segmentation threshold ε. This would, however, increase the percentage of undersegmentation (C3). The applied segmentation method is very sensitive to the choice of the parameter ε and, therefore, its value should be adjusted by an optimization procedure, which is not considered in this paper.

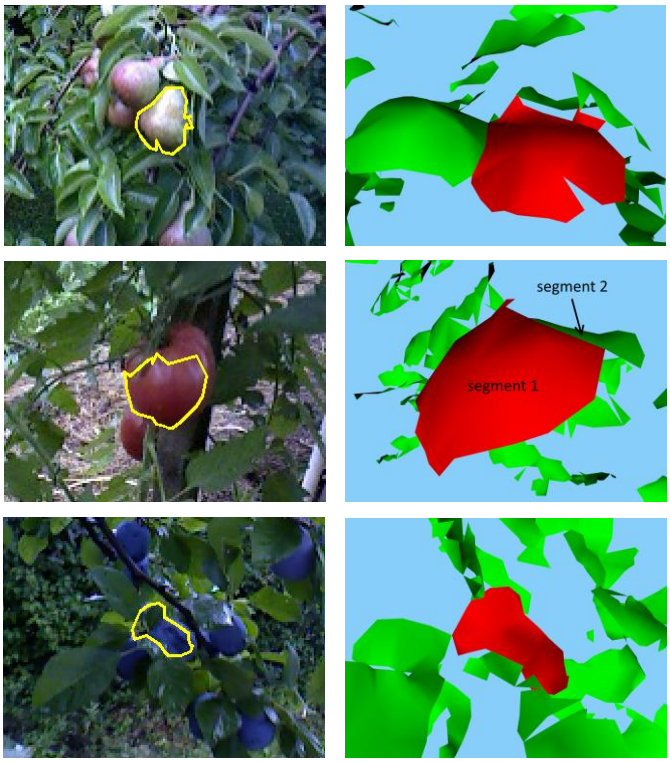


Fig. 3. Sample results of segmentation to approximately convex surfaces. Segments representing fruits are outlined by yellow contours in RGB images (left) and displayed by red color (right). Examples of a correct segmentation (top), oversegmentation (middle) and merging of fruit with adjacent objects (bottom) are presented. Notice that the tomato in the middle row is represented by two segments denoted by *segment 1* and *segment 2*.

V. Conclusion

The purpose of the research reported in this paper was to determine whether the segmentation of point clouds acquired by a 3D camera into approximately convex surfaces has a potential to be used as a preprocessing step of a fruit recognition system as well as to identify its drawbacks. The obtained experimental results demonstrate that a relatively high percentage of fruit in a close range of 0.6 m from the camera can be detected by the studied method as separate segments. However, oversegmentation as well as merging with adjacent leaves or branches also appears very frequently. In order to cope with the oversegmentation problem a higher-level

TABLE I. SEGMENTATION RESULTS

			Result Category		
Fruit sort	**#images**	**#fruits**	***C1* (%)**	***C2* (%)**	***C3* (%)**
plum	99	653	76.6	4.0	19.5
nectarine	50	147	66.7	11.6	21.8
pear	61	407	71.7	7.9	20.4
tomato	38	96	59.4	9.4	31.3

algorithm which would group adjacent segments according to their color and shape features can be applied. Furthermore, the undersegmentation could be reduced by introducing a color cue in the segmentation framework. Parameterized shape models of particular fruit sorts can be used for grouping of adjacent segments as well as for splitting the segments resulting from false merging of fruits with adjacent objects.

The investigation reported in this paper can be considered as a preliminary research in the development of a fruit recognition system which uses an RGB-D camera as the sensor and performs segmentation of the acquired point cloud into primitives before applying a suitable classifier. A more accurate evaluation of this approach can be made by testing it in combination with state-of-the-art classifiers.

REFERENCES

[1] R. Cupec, E. K. Nyarko, D. Filko, "Fast 2.5D Mesh Segmentation to Approximately Convex Surfaces, " Proc. of the 5th European Conference on Mobile Robots (ECMR), pp. 127-132, Örebro, Sweden, 2011

[2] R. C. Harrell, D. C. Slaughter and P. D. Adsit, "A Fruit-Tracking System for Robotic Harvesting, " Machine Vision and Applications, vol. 2, issue 2, pp 69-80, 1989.

[3] T. Kataok, Y. Ot and T. Hirom, "A Segmentation Algorithm for the Automatic Recognition of Fuji Apples at Harvest, " Biosystems engineering, vol. 83, issue 4, December 2002.

[4] J. Lu, N. Sang, Y. Hua and H. Fu, "Detecting citrus fruits with highlight on tree based on fusion of multi-map, " Optik - International Journal for Light and Electron Optics, vol. 125, issue 8, pp. 1903-1907, April 2014.

[5] H. Li, W. S. Lee and K. Wang, "Identifying blueberry fruit of different growth stages using natural outdoor color images, " Computers and Electronics in Agriculture vol. 125, pp. 91-101, August 2014.

[6] H. Peng, X. Zou, J. Xiong, Y. Chen, A. Guo and K. Chen, "Recognition of Mature Citrus in Natural Scene under the Occlusion Condition, " Journal of Information & Computational Science, vol. 11, issue 6, pp. 1947-1958, April 2014.

[7] L. Qiang, C. Jianrong, L. Bin, D. Lie and Z. Yajing, "Identification of fruit and branch in natural scenes for citrus harvesting robot using machine vision and support vector machine, " International Journal of Agricultural and Biological Engineering, vol. 7, no. 2, pp. 115-121, 2014.

[8] E. A. Murillo-Bracamontes, M. E. Martinez-Rosas, M. M. Miranda-Velasco, H. L. Martinez-Reyes, J. R. Martinez-Sandoval and H. Cervantes-de-Avila, "Implementation of Hough transform for fruit image segmentation, " Procedia Engineering, vol. 35, pp. 230-239, 2012.

[9] R. Roscher, K. Herzog, A. Kunkel, A. Kicherer, R. Töpfer and W. Förstner, "Automated image analysis framework for high-throughput determination of grapevine berry sizes using conditional random fields, " Computers and Electronics in Agriculture, vol. 100, pp. 148-158, January 2014.

[10] S. Sengupta and W. S. Lee, "Identification and determination of the number of immature green citrus fruit in a canopy under different ambient light conditions, " Biosystems Engineering, vol. 117, pp. 51-61, January 2014.

[11] Y. Song, C. A. Glasbey, G. W. Horgan, G. Polder, J. A. Dieleman, G. and W. A. M. van der Heijden, "Automatic fruit recognition and counting from multiple images, " Biosystems Engineering, vol. 118, pp. 203-210, February 2014.

[12] S. Cubero, M. P. Diago, J. Blasco, J. Tardáguila, B. Millán and N. Aleixosd "A new method for pedicel-peduncle detection and size assessment of grapevine berries and other fruits by image analysis, " Biosystems Engineering, vol. 117, pp. 62-72, January 2014.

[13] W. Jia, D. Zhao, F. Cheng, B. Xua, Y. Zhanga and J. Wang, "Automatic recognition vision system guided for apple harvesting robot, " Computers & Electrical Engineering, vol. 38, issue 5, pp. 1186-1195, September 2012.

[14] E. Kelman and R. Linker, "Vision-based localisation of mature apples in tree images using convexity, " Biosystems Engineering, vol. 118, pp. 174-185, February 2014.

[15] D. Stajnko, M. Lakota and M. Hočevar, "Estimation of number and diameter of apple fruits in an orchard during the growing season by thermal imaging, " Computers and Electronics in Agriculture, vol. 42, issue 1, pp. 31-42, January 2004.

[16] D. M. Bulanon, T. F. Burks and V. Alchanatis, "Image fusion of visible and thermal images for fruit detection, " Biosystems Engineering, vol. 113, pp. 12-22, May 2009.

[17] D. Font, T. Pallejà, M. Tresanchez, D. Runcan, J. Moreno, D. Martínez, M. Teixidó and J. Palacín, "A Proposal for Automatic Fruit Harvesting by Combining a Low Cost Stereovision Camera and a Robotic Arm, " Sensors, vol. 14, issue 7, pp. 11557-11579, 2014.

[18] J. Rakun, D. Stajnko and D. Zazula, "Detecting fruits in natural scenes by using spatial-frequency based texture analysis and multiview geometry, " Computers and Electronics in Agriculture, vol. 76, issue 1, pp. 80-88, March 2011.

[19] R. Xiang, H. Jiang and Y. Ying, "Recognition of clustered tomatoes based on binocular stereo vision, " Computers and Electronics in Agriculture, vol. 106, pp. 75-90, August 2014.

[20] A. Plebe and G. Grasso, "Localization of spherical fruits for robotic harvesting, " Machine Vision and Applications, vol. 13, issue 2, pp. 70-79, November 2001.

[21] A. R. Jiménez, R. Ceres and J. L. Pons, "A vision system based on a laser range-finder applied to robotic fruit harvesting, " Machine Vision and Applications, vol. 11, issue 6, pp. 321-329, May 2000.

[22] K. Tanigaki, T. Fujiura, A. Akase and J. Imagawa, "Cherry-harvesting robot, " Computers and Electronics in Agriculture, vol. 63, issue 1, pp. 65-72, August 2008.

[23] A. R. Jiménez, R. Ceres and J. L. Pons, "A Survey of Computer Vision Methods for Locating Fruit on Trees, " Transaction of the ASAE, vol. 43 issue 6, pp. 1911-1920, 2000.

[24] A. R. Jiméneza, A. K. Jain, R. Ceres and J. L. Pons, "Automatic fruit recognition a survey and new results using Range - Attenuation images, " Pattern Recognition, vol. 32, issue 10, pp. 1719-1736, October 1999.

[25] K. Kapach, E. Barnea, R. Mairon, Y. Edan and O. Ben-Shahar, "Computer vision for fruit harvesting robots – state of the art and challenges ahead, " International Journal of Computational Vision and Robotics vol. 3, issue 1/2, pp. 4-34, April 2012.

[26] P. Li, S. Lee and H. Hsu, "Review on fruit harvesting me thod for potential use of automatic fruit harvesting systems, " Procedia Engineering vol. 23, pp. 351-366, 2011.

[27] F. Schmitt and X. Chen, "Fast segmentation of range images into planar regions, " In IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 710-711, 1991.

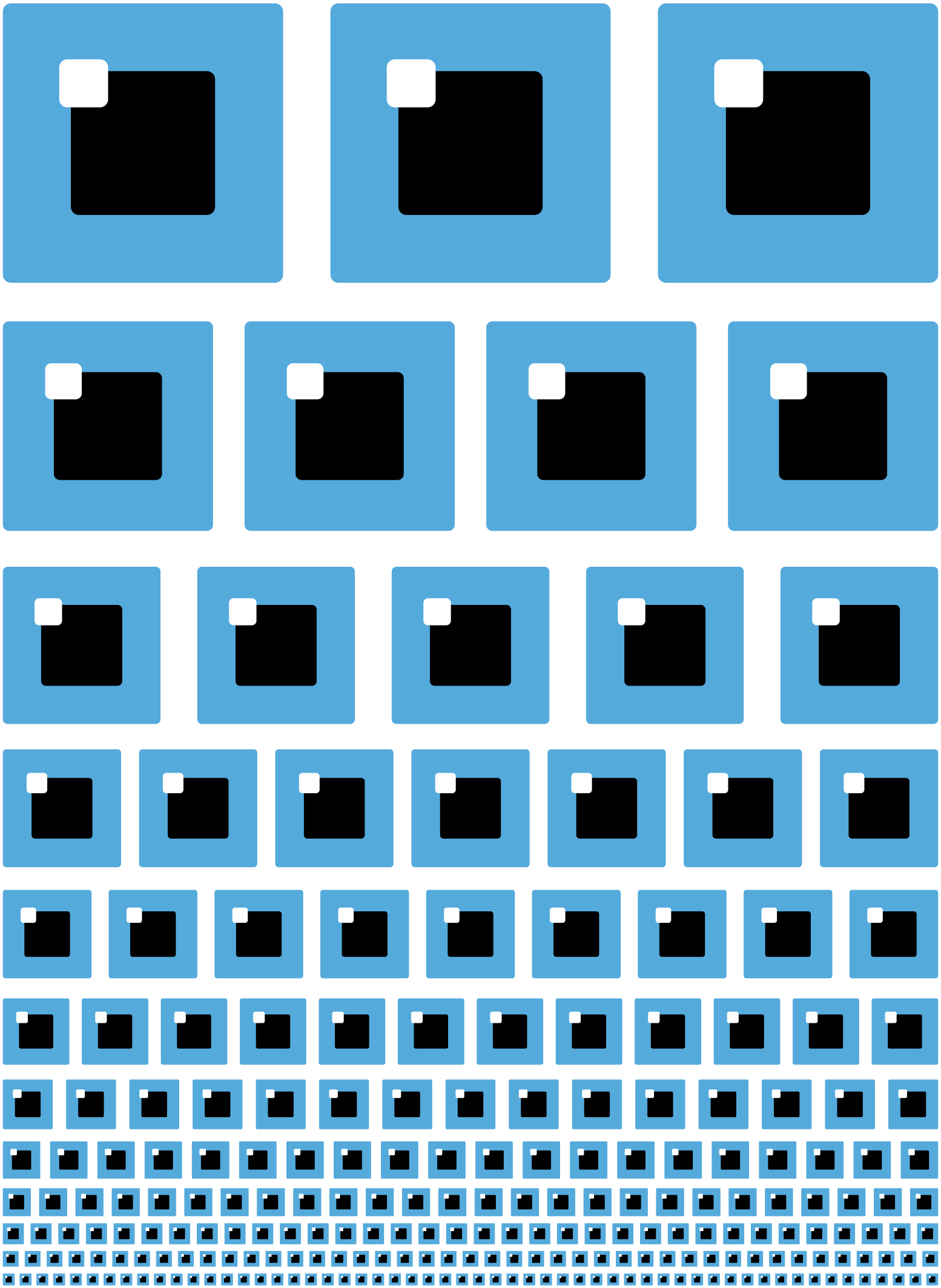
[28] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypothesis verification method for 3d object recognition," Proc. of the European Conference on Computer Vision (ECCV), pp. 511-524, Florince, Italy, 2012.

[29] J. Tang, S. Miller, A. Singh, and P. Abbeel, "A textured object recognition pipeline for color and depth image data," Proc. of the International Conference on Robotics and Automation (ICRA), pp. 3467-3474, Saint Paul, US, 2012.

[30] C. Papazov and D. Burschka, "An efficient ransac for 3D object recognition in noisy and occluded scenes," Proc. of the 10th Asian Conference on Computer Vision (ACCV), pp. 135-148, Queenstown, New Zealand, 2010.

[31] M. Attene, M. Mortara, M. Spagnuolo and B. Falcidieno, "Hierarchical Convex Approximation of 3D Shapes for Fast Region Selection", Computer Graphics Forum, vol. 27, issue 5, pp. 1323–1332, 2008.

Author Index



Proceedings of the Croatian Computer Vision Workshop CCVW 2014, Year 2 ISSN 1849-1227
Publisher: University of Zagreb Faculty of Electrical Engineering and Computing, Unska 3, HR-10000 Zagreb, Croatia
http://www.fer.unizg.hr/crv/proceedings