Fourth Croatian Computer Vision Workshop

September 22, 2015, Zagreb, Croatia

PROCEEDINGS OF CCVW 2015

University of Zagreb
Center of Excellence
for Computer Vision

Proceedings of the Croatian Computer Vision Workshop CCVW 2015, Year 3 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 2015

Proceedings of the Croatian Computer Vision Workshop

Zagreb, Croatia, September 22, 2015

S. Lončarić, J. Krapac (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

Donator

Visage Technologies AB

Proceedings of the Croatian Computer Vision Workshop
CCVW 2015, Year 3

Editor-in-chief

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

Editor

Josip Krapac (josip.krapac@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 4^{th} 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 4^{th} Croatian Computer Vision Workshop.

September 2015

Sven Lončarić, General Chair

Acknowledgements

The 2015 4th 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 diligently reviewed submitted papers and provided 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

Josip Krapac, University of Zagreb, Croatia

Local Arrangements Chair

Vedrana Baličević, University of Zagreb, Croatia

Publications Chair

Tomislav Petković, University of Zagreb, Croatia

Technical Program Committee

Vedrana Baličević, Croatia
Bart Bijnens, Spain
Hrvoje Bogunović, USA
Mirjana Bonković, Croatia
Karla Brkić, Croatia
Robert Cupec, Croatia
Albert Diosi, Slovakia
Hrvoje Gold, Croatia
Edouard Ivanjko, Croatia
Bojan Jerbić, Croatia
Zoran Kalafatić, Croatia
Stanislav Kovačič, Slovenia
Josip Krapac, Croatia
Sven Lončarić, Croatia
Thomas Mensink, The Netherlands
Vladan Papić, Croatia
Tomislav Petković, Croatia
Thomas Pock, Austria
Tomislav Pribanić, Croatia
Arnau Ramisa, Spain
Slobodan Ribarić, Croatia
Marko Subašić, Croatia
Damir Seršić, Croatia
Siniša Šegvić, Croatia

Reviewers

Hrvoje Bogunović, Austria
Robert Cupec, Croatia
Albert Diosi, Slovakia
Hrvoje Gold, Croatia
Edouard Ivanjko, Croatia
Zoran Kalafatić, Croatia
Stanislav Kovačić, Croatia
Tomislav Pribanić, Croatia
Arnau Ramisa, Spain

Measurement of Road Traffic Parameters based on Multi-Vehicle Tracking

Kristian Kovačić, Edouard Ivanjko and Niko Jelušić
Department of Intelligent Transportation Systems
Faculty of Transport and Traffic Sciences
University of Zagreb
Email: kristian.kovacic@fpz.hr, edouard.ivanjko@fpz.hr, niko.jelusic@fpz.hr

***Abstract*—Development of computing power and cheap video cameras enabled today's traffic management systems to include more cameras and computer vision based applications for monitoring and control of road transportation systems. Combined with image processing algorithms cameras are used as sensors to measure road traffic parameters like flow volume, origin-destination matrices, classify vehicles, etc. In this paper we propose a system for measurement of road traffic parameters (basic motion model parameters and macro-scopic traffic parameters). The system is based on Local Binary Pattern image features classification with a cascade of Gentle Adaboost classifiers to determine vehicle existence and its location in an image. Additionally, vehicle tracking and counting in a road traffic video is performed by using Extended Kalman Filter and virtual markers. The newly proposed system is compared with a system based on background subtraction. Comparison is performed by means of evaluating execution time and accuracy.**

I. Introduction

Recent decades can be characterized by a significant increase of the number of road vehicles accompanied by a build-up of road infrastructure. As a consequence, the number of traffic accidents has also increased. Statistics show that there is about 1.2 million fatalities and 50 million injuries related to road traffic accidents per year worldwide [1]. Statistics published by the Croatian Bureau of Statistics [2] reveal that in 2011 there were 418 killed and $18,065$ injured persons who were participating in traffic in the Republic of Croatia. In the attempt to reduce the amount of annual traffic accidents, a large number of different systems have been researched and developed. Such systems are part of the road infrastructure (horizontal and vertical signalization, variable message signs, etc.) or road vehicles (various advanced driver support systems). Vehicle manufacturers have implemented systems such as lane detection and lane departure, parking assistants, collision avoidance, adaptive cruise control, etc.

Today's road traffic measurement and management systems most often use computer vision algorithms and complementary metal oxide semiconductor (CMOS) video cameras. This is the consequence of the increase of computing power and decrease of CMOS video camera prices. Cameras are used as a part of the vehicle or road infrastructure. From the obtained video footage, high level road traffic information can be extracted. Such high level information includes incident detection, vehicle classification, origin-destination (OD) matrix estimation, etc. This information needs to be extracted in real-time with high accuracy. Mentioned information is crucial in advanced traffic management systems from the domain of intelligent transportation systems (ITS). Advanced traffic management systems have the goal to stabilize traffic flow and consequently enhance level of service for processes in a road traffic network from various aspects such as road safety, productivity loss, environment pollution, etc.

Basic parameters, which can define a basic motion model, of an individual vehicle are: time t needed to travel a specific distance s, current direction ϕ, velocity v, acceleration a and impulse of movement I. Additional parameters such as vehicle traffic flow volume $q\left[\frac{veh}{h}\right]$, traffic flow density $g\left[\frac{veh}{km}\right]$, traffic flow velocity $v\left[\frac{km}{h}\right]$, travel time in the traffic flow $t\,[h]$ and headway $S\,[m]$ can be used to define the state of a traffic network [3]. All mentioned parameters can be obtained using only video cameras as sensors.

Road traffic monitoring from the microscopic aspect is based on monitoring basic traffic parameters of each individual vehicle. These parameters are essential for traffic management on a smaller traffic network where mutual influence of vehicles needs to be analyzed. This is crucial to make the proper decision related to traffic management of a traffic network. In larger traffic networks macroscopic traffic parameters are also required for traffic management. With such parameters, physics and influence of each individual vehicle is not analyzed. Instead, traffic is represented by a statistical model where movement of vehicles is described using traffic flow. For a specific traffic flow, certain laws are applied by which it changes through time and space [3].

In this paper, we propose a system for road vehicle detection, tracking and counting. In section II, state-of-the-art (SOTA) approaches for measuring road traffic parameters are explained. Additionally, a similar system proposed in previous research is described. Section III describes the proposed system for vehicle detection. In this section, used image features and classifiers are explained in more detail. In section IV, experimental results are analyzed. We conclude and describe future work of this research in section V.

II. Measurement of road traffic parameters using video cameras

Measurement of traffic parameters (basic and complex) using video cameras is based on three main steps. First step

consists of detecting a vehicle in an image. Result of this step is the location and size of a vehicle in the image. Second stage consists of tracking individual vehicles through multiple consecutive images. In this step, vehicle trajectory can be obtained and analyzed for further computation of traffic parameters. Last step of traffic parameter measurement is vehicle counting by which parameters such as traffic flow volume can be computed.

A. Problems and basic approaches

As mentioned previously, there are three main steps in a system for traffic parameters measurement using video cameras. First step, which includes vehicle detection, can be performed using various computer vision methods. One of the commonly used methods is background subtraction. It is based on separating foreground and background segments of an image. This can be performed by comparing two or more consecutive images. When only two images are compared, simple subtraction between two pixels intensities at the same location in the images can be performed. By comparing the subtracted value and a specific threshold value, pixel on the specific location in an image can be classified as a foreground or background pixel [4]. Another approach of a background subtraction method is based on using a Gauss Mixture Model (GMM). GMM is based on computing a probability that certain pixel intensity value at a specific location in an image represents a foreground or background object. If a pixel in the image has probability to be part of the background larger than a specific threshold it is classified as background or otherwise as foreground [5]. Main deficiency of mentioned methods is their incapability to detect static vehicles in an image (e.g. parked vehicles or vehicles in a queue). Methods require a static camera from which the road traffic video is acquired (camera does not change its location and position under influence of wind, vehicle vibrations, etc.).

A similar method where pixels are compared in consecutive images is based on optical flow. Basic work principle of this method is to compute the optical flow of pixels or regions in images. This is achieved by computing an offsets of pixels or regions related to two or more consecutive images. Pixel displacement, direction and velocity are most commonly used as parameters of the optical flow.

System for vehicle detection based on machine learning can use a combination of artificial neural network (ANN), fuzzy logic and genetic algorithm based methods. Today's SOTA approach based on ANN is the deep learning method. Main benefits of deep learning are learning of a multi-step algorithm and representation learning. Another machine learning approach is based on classifiers and hard-coded image features. Both mentioned machine learning approaches require a learning dataset that consists of a large number of positive and negative samples. Positive samples represent images that contain an object and negative samples represent images that do not contain an object which wants to be detected in an image. In the approach based on classifiers and image features, image features are computed for every sample in the learning dataset. Many types of image features exist such as Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), HAAR, Local Binary Patterns (LBP), etc [6]. After mentioned features are computed for all samples in the learning dataset, learning process of the chosen classifier is performed. After the classifier is learned to perform classification of image features, object detection can be performed. Image features are computed from the new image and separated into two classes: (i) object class; (ii) and non-object class.

B. State of the art

To compute any of the previously mentioned road traffic parameters, system needs to perform image processing on videos obtained from road traffic video cameras. As mentioned in the previous section, one of tasks, which needs to be solved, is vehicle detection in an image. Cheng, Lin, Chen, et al. [7] proposed a system which uses a feature extraction and classification method for vehicle detection in an image. System is based on a Support Vector Machine (SVM) classifier, which can perform classification of features into one of two classes defined by Eq. 1. In Eq. 1, $f(x)$ is the result of classification performed by SVM and $K(sv, x)$ is the kernel function which uses a support vector sv and a feature vector x to perform weak classification multiplied by a coefficient α_n. The kernel function $K(sv, x)$ is a radial based function. The feature vector x uses Histogram of Oriented Gradient (HOG) image features. Hit rate represents ratio between the number of detected vehicles and ground truth. Hit rate is above 94% when a small number of false positive detection exists [7].

$$\begin{aligned} f(x) &= \alpha_1 K(sv_1, x) + \alpha_2 K(sv_2, x) + \\ &\quad \ldots + \alpha_m K(sv_m, x) \end{aligned} \tag{1}$$

In the system proposed in [8], vehicle detection and tracking is performed by two separated algorithms. Vehicle detection is performed by a background subtraction method and combination of morphological operations. Benefit of using these methods is algorithm simplicity and its fast execution. Object tracking is based on a Lucas-Kanade pyramid based optical flow algorithm. Overall accuracy of the system in experimental results is 77.9%, where object detection accuracy is 81.75% and object tracking accuracy is 95.3% [8].

Problem with both previously mentioned systems is that their accuracy highly depends on camera perspective and scenario situation. If the same system for vehicle detection is used on multiple camera mounting locations (above or beside the road), system parameters need to be adjusted for every location manually. In [9], the system proposed for vehicle detection and tracking tackles the mentioned multi-perspective problem. The system is based on Haar-like features and a cascade of Adaboost classifiers. Learning dataset consists of approximately million samples which are separated into smaller subsets depending on vehicle motion direction. Experimental results show that the hit rate is above 75% when the number of false positives is above 100 of total 700.

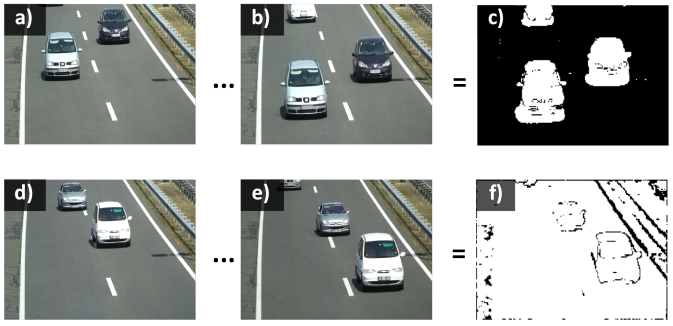


Figure 1. Original sets of images a-b and d-e processed by background subtraction method c and f.

Another problem with mentioned systems is their application in urban scenarios which represents a challenging environment. System proposed in [10] is based on a hybrid image template (HIT). The HIT consists of multiple image patches with various types of features including a sketch, texture, flatness and color. Image sketch patch is based on Gabor wavelet elements, the texture and flatness patch are based on Gabor filter [11] and color is extracted through histogram in HSV color space. A probability model of the HIT is defined for assisting vehicle detection. Vehicle detection is performed in three stages through the SUM-MAX procedure where in each stage a local image region with the maximal score is selected as a vehicle candidate. If this score is greater than the learned threshold, candidate is viewed as a detected vehicle object. The experimental results show that the HIT preferably extracts meaningful features including sketch, texture, flatness, and color for vehicle object, and improves the vehicle detection accuracy in complex urban traffic conditions. The proposed system adapts to slight vehicle deformation also.

C. Previous research

In previous research, authors proposed a system for vehicle detection based on the background subtraction method which is described in [18]. As a result of the background subtraction method, each pixel in the image is classified as 1 if it is a part of the foreground (moving vehicles) or 0 if it is a part of the background (static object). Main disadvantage of the background subtraction method is shown in Fig. 1. From the set of images between 1a-1b and 1d-1e, a background model is created and subtracted from the newest image, which result is shown in c and f. In the given examples, the result of background subtraction given in the image c has a small number of incorrectly classified pixels. In the image f most of pixels are incorrectly classified due to sudden light change in the whole image. This event most often occurs as a result of a large bright object (e.g. truck) passing the scene or when the camera changes its position in an event of strong wind.

III. Approach based on classifier learning

System proposed in this paper consists of three main components. First component performs vehicle detection, second component tracks detected vehicles in the image plane and third component performs vehicle counting in order to obtain traffic parameters.

A. Vehicle detection

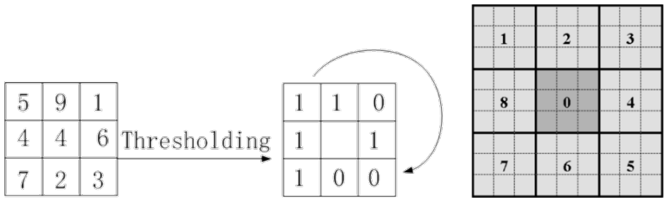
The proposed system in this paper is based on a method that extracts Multi-Scale LBP (MB-LBP) features from an image as described in [12] and performs classification with the Gentle Adaboost (GAB) algorithm explained in [13] and [14]. Methods are implemented in OpenCV library with CUDA support. Before vehicle detection can be performed in an image, a classifier needs to be learned to classify vehicle MB-LBP features in the image. The learning process is performed with a learning dataset consisting of two subsets of images:

- Images which contain vehicles (positive samples);
- Images which do not contain vehicles (negative samples).

From the individual subsets, MB-LBP features are extracted. The classifier is learned to separate specific features into two classes (with and without vehicles). Three versions of dataset are used to learn the classifier as described in continuation of this paper.

1) Image features: MB-LBP features represent modified version of LBP features that are originally described in [15]. In [15], LBP features are used for face detection, however they can be applied on different type of objects such as vehicles [14]. MB-LBP features are similar to LBP features and they are both described in continuation of this subsection. First step of LBP feature extraction is comparison of the specific pixel in an image with each of its 8 adjacent pixels. If an adjacent pixel intensity is greater than or equal to the middle pixel, cell with bit 1 is used in a binary string, or bit 0 otherwise as shown in Fig. 2a. Binary string consists of 8 bits where each bit is set to previously mentioned result of the adjacent pixel comparison. In Fig 2a, bits are stored clockwise into the binary string, where the binary string value would be 11010011. This procedure is repeated for every pixel in the region for which LBP feature extraction needs to be performed. MB-LBP feature extraction is based on multi-scale cells where each cell consists of more than one pixel as shown in Fig. 2b. By this method, MB-LBP features can describe a texture in the image in micro-scopic and macro-scopic level. After LBP or MB-LBP features are extracted for each pixel, a histogram is computed based on binary string values. In a LBP feature histogram, every combination of a uniform LBP feature is stored in a separate histogram bin with one additional bin space for all features that are not uniform. Uniform LBP features denote all binary strings which contain no more than two transitions between bit 0 and 1. In a MB-LBP feature histogram instead of using uniform features, first 63 features with the highest rank (occurrence) are stored in histogram bins with one additional histogram bin which summarizes all features which are below first 63 features with the highest rank [16].

2) Classifier: In the proposed system, cascade of classifiers is used for classification process. Decision stump is used as a weak classifier and Gentle Adaboost is used as a strong classifier. Decision stump as a weak classifier computes the



(a) LBP features (b) MB-LBP features

Figure 2. Principle of feature extraction [16].

hypothesis $h_t(x)$ with accuracy a slightly higher than random guessing. Multiple weak classifiers combined together in the strong classifier allow computation of the strong hypothesis with high accuracy. A weak classifier can be defined with Eq. 2, where $h_t(x)$ is a weak hypothesis, x is a feature vector and $threshold$ is a constant value used for comparison with a feature x_i in the feature vector x. The $threshold$ is determined in classifier learning process. Strong classifier is defined by Eq. 3, where $H(x)$ is a strong hypothesis, α_t is a weight factor computed in a learning step of classifier [13].

$$h_t(x) = \begin{cases} -1, & if\ x_i < threshold \\ 1, & if\ x_i \geq threshold \end{cases} \quad (2)$$

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \quad (3)$$

Cascade classifier consists of multiple stages, where in each stage is one of the strong classifiers. In the first stage, strong classifier computes $H(x)$ based on a small number of features in the feature vector x. In each following stage, a strong classifier uses an increased number of features in the feature vector x, obtaining more and more accurate hypothesis. This process reduces execution time of classification algorithm.

3) Classifier learning process: Before vehicle detection can be performed in the image, a classifier needs to be learned to classify vehicle LBP features in the image. The learning process is performed using a learning dataset with the properties described above.

- Images which contain vehicles (positive samples);
- Images which do not contain vehicles (negative samples).

From the individual subsets, LBP features are extracted. The classifier is learned to separate specific features into two classes (with and without vehicles). In this paper, three versions of datasets described in Tab. 1 are used to learn the classifier. The dataset created by the Faculty of Transport and Traffic Sciences is made from video footage of a road near the city of Zagreb obtained by a full HD camera with resolution $1920x1080$. Dataset contains images of a vehicle from front-side and back-side perspective obtained using a camera mounted above the road. Dataset made by TME [17] contains images of a vehicle only in back-side perspective obtained from a camera mounted in a vehicle.

Table 1: Number of samples in datasets.

Dataset created by	Number of positive samples	Number of negative samples	Total number of samples
Faculty of Transport and Traffic Sciences (FTTS)	4,298	1,930	6,228
Toyota Motor Europe (TME) Motorway dataset [17]	25,155	26,321	51,476
Combination of the both datasets	29,453	28,251	57,704

B. Vehicle tracking

Trajectory computation of a moving object in a set of consecutive images is performed by EKF. Each detected object in a scene is separately tracked by EKF. An EKF state model is based on a vector x defined in Eq. 4, where x_x and x_y represent location of the moving object on the x and y axis, x_v is the velocity and x_a is the acceleration of the moving object over an image plane, x_ϕ and x_ω are the object direction and change of direction in a time interval t. Prediction of a future model state is performed with Eq. 5, where $x_{k|k-1}$ is a predicted state model vector in the step k based on a state model vector $x_{k-1|k-1}$ from the previous step $k-1$ and t is the time interval between the steps $k-1$ and k.

$$x = \begin{bmatrix} x_x \\ x_y \\ x_v \\ x_a \\ x_\phi \\ x_\omega \end{bmatrix} \quad (4)$$

$$x_{k|k-1} = f(x_{k-1|k-1}, t) \quad (5)$$

After the prediction step has been finished, an update step is performed which is described in more details in [18]. For measurement step in EKF, data from the applied vehicle detection method (location, size) is used.

C. Vehicle counting

The last step in the proposed system is vehicle counting. Vehicle counting is based on virtual markers M_1 and M_2 shown in Fig. 3 as rectangles 0 and 1. Virtual markers are located in the bottom part of the image in order to detect that an object is leaving the scene. Detection method is based on checking overlaps between each moving object and virtual marker. When the background subtraction method is used, a vehicle counter will be increased by 1 if following requirements are fulfilled:

- The total number of images in which an object was present in the scene is greater than *TFC*;
- When the EKF is used for vehicle tracking, vehicle is overlapping with a virtual marker if the direction of vehicle movement is inside a specific interval $[\phi_{min}, \phi_{max}]$, where ϕ_{min} and ϕ_{max} are set by Eqs. 6-7.

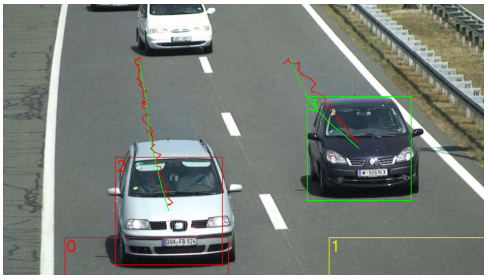


Figure 3. Testing of vehicle detection, tracking and counting algorithms.

$$\phi_{min} = \left(\frac{1}{8} + 1\right)\pi[rad] \quad (6)$$

$$\phi_{max} = \left(\frac{7}{8} + 1\right)\pi[rad] \quad (7)$$

When the feature classification method is used, a vehicle counter will be increased by 1 if following requirements are fulfilled:

- The total number of images in which an object was present in the scene is greater than TFC;
- The total travelled distance of an object in the image plane is larger than $\frac{1}{5}max(w,h)$, where w is the image width $[px]$ and h is the image height $[px]$.

IV. Experimental results

In this section experimental results are given which show comparison between previously mentioned methods for vehicle detection and tracking from aspects of accuracy and execution time. Used video footage is recorded by a camera mounted above the bypass of the city of Zagreb at the location Ivanja Reka. The camera records the vehicle front side and therefore combination of FTTS and TME dataset is used. Results of testing the background subtraction method are given in Tab. 2, where image resolution of 240×135 $[px]$ is used in vehicle detection, *FP* is the number of false positive detections, *FN* is the number of false negative detections and *GT* is the groundtruth data, *TFC* is the minimum number of images in which an object needs to be detected to be classified as vehicle. Testing of the background subtraction method is performed with and without EKF as noted in Tab. 2 on the whole video footage. Accuracy *A* is defined by Eq. 8. Average execution time of background subtraction method in all testings has the same value of 16 *[ms]*. Experimental results of the feature classification method are given in Tab. 3, where *S* is the number of stages in a cascade classifier, *MHR* (Minimum Hit Rate) is the minimum ratio between the number of detected objects and the total number of objects which have to pass into a next learning stage of a cascade classifier and *MCC* (Minimum Cluster Count) is the minimum number of individual clusters detected by the feature classifier which an object (vehicle) needs to contain. Testing of the proposed method is performed with and without EKF same as in the case of the background subtraction method. Execution time in *[ms]* is given in Fig. 4 where the fastest measurement regarding the execution time was approximately 53 *[ms]*. In tested methods, a false negative detections occur if a virtual marker has not been activated due to a vehicle passing through the scene. This occurs if a vehicle is not detected on every image and the total number of images, on which vehicle was present, is less than *TFC* parameter. A false positive detection occurs if a vehicle is wrongly classified into two or more vehicles in the scene.

Table 2: Experimental results for the background subtraction method.

Image resolution	Method parameters		Counted vehicles	
	th	TFC	FP / FN / GT	A [%]
Without EKF				
240×135	10	15	4 / 2 / 133	95
240×135	10	30	2 / 6 / 133	94
240×135	8	15	5 / 6 / 133	92
240×135	8	20	5 / 8 / 133	90
With EKF				
240×135	10	10	4 / 4 / 133	94
240×135	10	30	3 / 6 / 133	93
240×135	8	10	4 / 6 / 133	92
240×135	8	30	4 / 8 / 133	91

$$\left(1 - \frac{FP + FN}{GT}\right) \times 100\% \quad (8)$$

The system based on the background subtraction method was tested on all combinations of the following method parameters:

- Image resolution used in the detection process $= \{240 \times 135, 480 \times 270, 960 \times 540\}$;
- Threshold value $th = \left\{ \begin{array}{l} 5, 8, 10, 15, 20, 25, \\ 30, 35, 40, 45, 50 \end{array} \right\}$;
- $TFC = \{10, 15, 20, 30\}$;
- $EKF = \{used, notused\}$.

Testing values for parameters of the system based on the feature classification method where:

- Image resolution used in the detection process $= \{240 \times 135, 480 \times 270, 960 \times 540\}$;
- $TFC = \{3, 8, 15\}$;
- $MHR = \{0.99, 0.9925, 0.995, 0.9975, 0.999875\}$;
- The number of stages in a cascade $N = \{5, 10, 20\}$;
- $EKF = \{used, notused\}$.

V. Conclusion & Future work

In this paper, experimental results of two systems for vehicle detection and tracking in road traffic video footage are compared. Testings are performed with various system parameter values where maximum achieved vehicle counting accuracy is 95%. From obtained experimental results, it can be concluded that with the feature classification method similar accuracy can be obtained as with the background subtraction method. System based on the background subtraction method has the advantage that it can detect any moving object in the video footage. As a consequence, determination of system parameters is a simple process. Main disadvantage of this

Table 3: Experimental results for the feature classification method.

Image resolution	Method parameters				Counted vehicles	
	S	MHR	MCC	TFC	FP / FN / GT	A [%]
Without EKF						
240 × 135	20	0.995	2	8	0 / 6 / 133	95
240 × 135	20	0.999875	5	3	0 / 7 / 133	95
240 × 135	20	0.999875	5	15	0 / 7 / 133	95
240 × 135	20	0.995	2	3	3 / 5 / 133	94
480 × 270	20	0.995	5	3	3 / 6 / 133	93
With EKF						
240 × 135	20	0.995	2	3	0 / 9 / 133	93
240 × 135	20	0.995	2	8	0 / 9 / 133	93
240 × 135	20	0.995	2	15	0 / 9 / 133	93
240 × 135	20	0.999875	5	3	0 / 9 / 133	93
240 × 135	20	0.999875	5	8	0 / 9 / 133	93

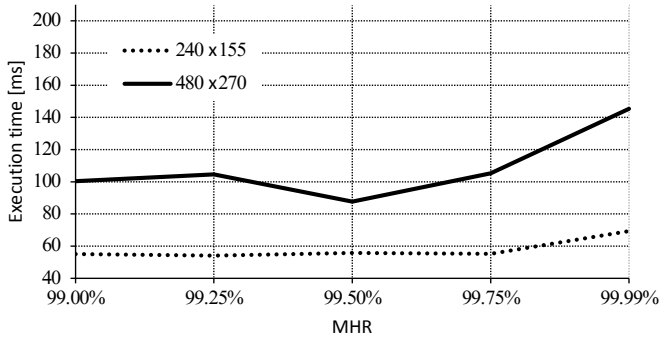


Figure 4. Execution time of the proposed system based on the feature classification method.

method is its sensitivity to camera vibrations and sudden lighting changes in an image which can cause increase of false positive and false negative detections.

When a feature classification method is used, the system is robust to camera vibrations and sudden lighting changes in the image, and its accuracy keeps the same value. Disadvantage of this method is the requirement for a classifier learning process and a large dataset in order to perform accurate object detection. This requirement consumes a lot of time spent on preparation of a dataset and the classifier learning process. Mentioned method increases the number of system parameters, where determination of initial values of parameters is a more complex process which requires a lot of testings and system analysis. Another disadvantage of the newly proposed feature classification based method is its execution time which is substantially longer than in the case of the background subtraction method.

Future work of this research would consists of expanding the current dataset with side images of vehicles, images of heavy vehicles (trucks) and motorcycles. This would allow detection of vehicles in various poses. Optimization of currently developed method from an aspect of execution time would need to be examined in order to provide real-time vehicle detection. Additional analysis of the system accuracy when other image features are used such as SURF, HOG or SIFT will be made also.

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". The authors also thank Mario Muštra for his helpful suggestions and comments.

REFERENCES

[1] C. Hughes, R. O'Malley, D. O'Cualain, M. Glavin and E. Jones: "New Trends and Developments in Automotive System Engineering: Chapter Trends Towards Automotive Electronic Vision Systems for Mitigation of Accidents in Safety Critical Situations" - InTech, 2011., pp. 493-512

[2] E. Omerzo, S. Kos, A. Veledar, K. Šakić Pokrivač, L. Pilat, M. Pavišić, A. Belošević, and Š. V. Njegovan: "Deaths in Traffic Accidents" - Croatian Bureau of Statistics, 2012.

[3] G. Kos and I. Dadić: "Theory and organization of traffic flows" - Faculty of Transport and Traffic Sciences, University of Zagreb, 2007.

[4] M. Piccardi: "Background subtraction techniques: a review" - Proceedings of IEEE Conference SMC, Vol. 4, The Hague, October, 2004., pp. 3099-3104

[5] D. Arita, T. Tanaka, A. Shimada and R. I. Taniguchi: "A fast algorithm for adaptive background model construction using parzen density estimation" - Proceedings of IEEE Conference AVSS, London, September, 2007., pp. 528–533

[6] X. Jiang: "Feature extraction for image recognition and computer vision" - Proceedings of IEEE Conference ICCSIT, Beijing, August, 2009., pp. 1-15

[7] K.-M. Cheng, C.-Y. Lin, Y.-C. Chen, T.-F. Su, S.-H. Lai and J.-K. Lee: "Design of Vehicle Detection Methods With OpenCL Programming on Multi-Core Systems" - Proceedings of IEEE Conference ESTIMedia, Montreal, October, 2013., pp. 88-95

[8] K. Kiratiratanapruk and S. Siddhichai: "Vehicle Detection and Tracking for Traffic Monitoring System" - Proceedings of IEEE Conferenece TENCON, Hong Kong, November, 2006., pp. 1-4

[9] R. Feris, S. Pankanti and B. Siddiquie: "Learning Detectors from Large Datasets for Object Retrieval in Video Surveillance" - Proceedings of IEEE Conference ICME, Melbourne, July, 2012., pp. 284-289

[10] Y. Li, B. Li, B. Tian, F. Zhu, G. Xiong and K. Wang: "Vehicle Detection based on the Deformable Hybrid Image Template" - Proceedings of IEEE Conference ICVES, Dongguan, July, 2013., pp. 114-118

[11] V. S. N. Prasad and J. Domke: "Gabor Filter Visualization" - Technical report, University of Maryland, 2005.

[12] M. Kurt, B. Kir and O. Urhan: "Local binary pattern based fast digital image stabilization" - IEEE Signal Processing Letters, IEEE Journals & Magazines, Vol. 22, No. 3, 2015., pp. 341-345

[13] G. Lemaître and M. Radojević: "Directed reading: Boosting algorithms" - Heriot-Watt University, Universitat de Girona, Universite de Bourgogne, December, 2009.

[14] H. Liang, G. Teodoro, H. Ling, E. Blasch, G. Chen and L. Bai: "Multiple kernel learning for vehicle detection in wide area motion imagery" - Proceedings of IEEE Conference FUSION, Singapore, July, 2012., pp. 1629–1636

[15] T. Ojala, M. Pietikäinen and D. Harwood: "A comparative study of texture measures with classification based on feature distributions" - Pattern Recognition, Vol. 29, No. 1, January, 1996, pp. 51-59

[16] S. Liao, X. Zhu, Z. Lei, L. Zhang and Stan. Z. Li: "Learning Multi-scale Block Local Binary Patterns for Face Recognition" - Proceedings of IEEE Conference ICB, August, 2007., pp. 828-837

[17] C. Caraffi, T. Vojir, J. Trefny, J. Sochman and J. Matas: "A system for real-time detection and tracking of vehicles from a single car-mounted camera" - Proceedings of IEEE Conference on Intelligent Transportation Systems, Anchorage, September, 2012., pp. 975-982

[18] K. Kovačić, E. Ivanjko and H. Gold: "Real Time Vehicle Trajectory Estimation on Multiple Lanes" - Proceedings of 3rd CCVW2014, Zagreb, September, 2014., pp. 21-26

Towards Reversible De-Identification in Video Sequences Using 3D Avatars and Steganography

Martin Blažević*, Karla Brkić*, Tomislav Hrkać*
*University of Zagreb, Faculty of Electrical Engineering and Computing, Croatia
Email: martin.blazevic@fer.hr, karla.brkic@fer.hr, tomislav.hrkac@fer.hr

***Abstract*—We propose a de-identification pipeline that protects the privacy of humans in video sequences by replacing them with rendered 3D human models, hence concealing their identity while retaining the naturalness of the scene. The original images of humans are steganographically encoded in the carrier image, i.e. the image containing the original scene and the rendered 3D human models. We qualitatively explore the feasibility of our approach, utilizing the Kinect sensor and its libraries to detect and localize human joints. A 3D avatar is rendered into the scene using the obtained joint positions, and the original human image is steganographically encoded in the new scene. Our qualitative evaluation shows reasonably good results that merit further exploration.**

I. Introduction

Protecting the privacy of law-abiding individuals in surveillance footage is becoming increasingly important, given the widespread use of video surveillance cameras in public places such as streets, transit areas, shopping centers, banks, etc. Unrestricted access to such footage without privacy protection could enable identification and tracking of individuals in real time. In practice, privacy protection of persons in image and video data can be achieved using de-identification techniques. De-identification is a process of reversibly removing or obfuscating identifying personal biometric features, such as skin, hair and eye color, body posture, clothing, birthmarks, tattoos etc.

We envision a de-identification pipeline that enables reversible concealment of persons' identities in video sequences. Our pipeline aims to de-identify all hard biometric features, while taking into account most soft biometric features (e.g. tattoos, birth marks, clothing). The steps in our pipeline are as follows: (i) detecting humans in video sequences, (ii) segmentation of the detections, (iii) human pose estimation (joint modeling), (iv) rendering naturally-looking 3D models of humans on top of human detections, and (v) steganographically encoding the de-identified information in the video, as illustrated in Figs. 1 and 2. Building this pipeline is our long term goal, and in this work we focus on qualitatively exploring the last two stages of the pipeline, i.e. (iv) rendering naturally-looking 3D models of humans on top of human detections and (v) steganographically encoding the de-identified information in the video. Previous stages of the pipeline have been explored in our work [1], [2].

II. Related work

Each stage of the proposed detection pipeline is a broad topic of research in computer vision, so we limit ourselves to briefly reviewing methods that could be applicable in individual stages. Note that in a practical de-identification system one should expect the boundaries between the first three stages to become blurred: detecting, segmenting and estimating the pose of humans can often become an iterative process where the stages collaborate to find the most likely hypothesis.

Modern methods for pedestrian detection include the HOG detector [3] (histograms of oriented gradients), deformable part models [4], integral channel features [5], [6], convolutional neural networks [7], [8], [9], the Viola-Jones detector [10] and its problem-specific extensions [11], etc. Using background subtraction [12], [13] or recently introduced object proposals [14] for person detection might also be considered if the problem scenario is sufficiently constrained. Jointly employing several methods could be a promising strategy, as recent comparative studies [5], [15], [6] have shown that pedestrian detection in non-ideal conditions (moving camera, small-scale pedestrians, occlusions) is a hard problem that benefits from combining multiple features.

Having a detection bounding box, fine segmentation-based localization of pedestrians can be achieved by e.g. one of the variants of the GrabCut segmentation algorithm [16]. Furthermore, detection, tracking and segmentation can be done in a bootstrapping loop, as e.g. in the work of Hernandez-Vela et al. [17]. They propose automating GrabCut by combining HOG-based human full-body detection and face and skin color detection with tracking and segmentation models. In similar spirit, Poullot and Satoh [18] propose an algorithm called VabCut, useful for video foreground object segmentation when the camera is not stationary, where object locations are inferred using motion.

Human pose estimation can be combined with detection in a part-based model framework, such as in the work of Andriluka et al. [19] who propose a general part-based approach based on pictorial structures. Similarly, Yang and Ramanan [20] use a flexible mixture model that encodes both co-occurences and spatial relations between human body parts. Toshev and Szegedy [21] propose estimating human pose using deep learning, formulating it as a regression problem towards body joints. In general, human pose estimation is difficult, as each joint has

(a) the input image (b) pedestrian detection (c) segmentation

(d) pose estimation (e) model rendering (f) steganographic encoding

Fig. 1: The visualisation of our de-identification pipeline.

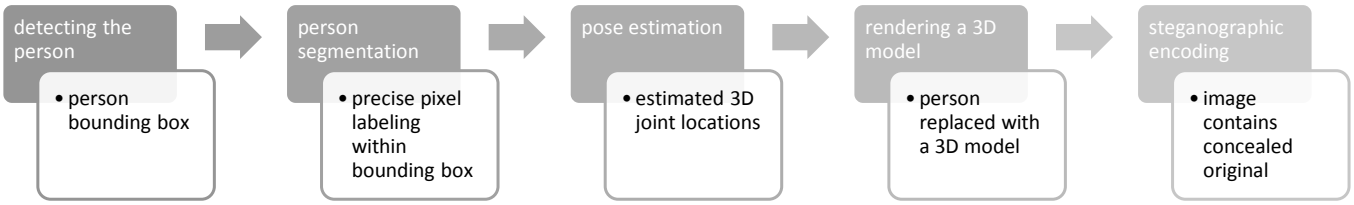


Fig. 2: The steps in our envisioned de-identification pipeline.

many degrees of freedom, and the appearance of body parts can vary considerably due to clothing, body shape, viewing angle etc. [20]. The problem can be somewhat simplified by introducing another layer of data: the depth map of the scene, and using a dedicated sensor such as Microsoft Kinect. Shotton et al. [22] introduce the Kinect's pose estimation algorithm based on randomized decision forests, body part recognition and depth image features. The algorithm achieves acceptable mAP rates and works very fast on dedicated hardware.

With localized joints and precise positions of human body parts, it is possible to replace portions of or the entire rendered human with an artificial realistically-looking model. The majority of related work in this area focuses on face replacement. For instance, Blanz et al. [23] propose a face exchange algorithm that enables fitting morphable face models to an image, effectively replacing the face in the image with the rendered model, taking into account pose and illumination parameters. Dale et al. [24] propose a system for face replacement in videos, enabling replacing the face and its motions due to speech and facial expressions with another face. Samaržija and Ribarić [25] propose using active appearance models for face detection and de-identification.

Finally, the information concealed by rendering the 3D model can be stored hidden in the new image using steganography, a technique used for secretive communication by hiding information in digital media. Steganography is also referred to as the art and science of communicating which hides the existence of the communication [26]. Steganographic encoding of images aims to exploit the image format in order to store the secret information so that the changes to the image are imperceptible. Various algorithms exist [27], and in this work we focus on LSB insertion [28] that encodes the secret information by modifying the least significant bits of the carrier image.

III. The envisioned de-identification pipeline

Our long-term goal is building a de-identification pipeline that enables privacy protection of persons in video sequences by rendering a naturally-looking 3D human model in place of each detected person. Simultaneously, the image of the per-

son is steganographically encoded in the replacement image. Hence, the naturalness of the scene is preserved, while the private information is stored in a secure manner.

In this work, we focus on exploring the last two stages of the pipeline, namely replacing the person in the video with a 3D model and encoding the replaced image in the new image using steganography. In order to be able to render a 3D model at the correct place, we must first detect and localize the person and estimate her pose, i.e. find the 3D positions of joints. As mentioned previously, this is a hard problem [20], [15], therefore, in this work we investigate the general feasibility of our approach by choosing a close to best case scenario. We ask ourselves how well the 3D model rendering and steganographic encoding would perform if the person is reliably detected and her joint positions are localized. To ensure this scenario, we use dedicated hardware, namely the Microsoft Kinect sensor. Using Kinect SDK, we are able to obtain a reasonably reliable pose estimation of a human standing close to the sensor. We now describe the process of pose estimation, 3D model rendering and steganographic encoding in more detail.

A. *Recovering the 3D skeleton from Kinect data*

Microsoft Kinect is a multifunctional physical device which enables the detection of human voice, movement and face gestures. It combines an RGB camera, 3D depth sensor cameras, a microphone array and a software pipeline for processing color, depth, and skeleton data [29]. The RGB camera enables capturing a color image. Depth sensor cameras consist of an infrared (IR) emitter and an IR depth sensor, which enable capturing a depth image. The distances between the observed object and the sensor itself represent the depth information. This type of information is used for detecting and precise tracking of human body or skeleton.

The Kinect for Windows Software Development Kit (SDK) is a set of drivers, Application Programming Interfaces (APIs), device interfaces, and tools for developing Kinect-enabled applications [30]. The Kinect sensor and its software libraries use the Natural User Interface (NUI) libraries to interact with custom applications. NUI represents the core of the Kinect for Windows API. It supports fundamental image and device management features. It also makes sensor data, such as color and depth image data, audio data, as well as skeleton data, which Kinect sensor generates in real time by processing the depth data, accessible from inside user developed applications. There are a few configuration prerequisites for applications that use or display data from the skeleton stream, such as the one provided in this paper. The used streams should be identified, initialized and opened. Skeletal tracking must also be enabled. Buffers for holding sensor data should be allocated and the new data retrieved for each of used streams every new frame.

Kinect is able to detect and recognize up to six users standing in front of the sensor, or rather in the field of view (FOV) of the camera, and accurately track, extract and provide detailed information on up to two of them. For a user to be tracked, no specific pose, gesture or calibration action needs to be performed.

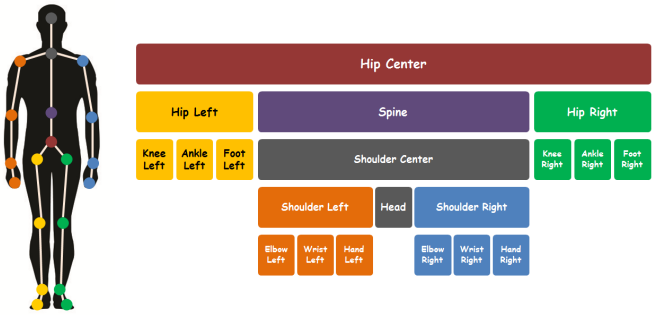


Fig. 3: The hierarchy of the joints in Kinect skeletal tracking.

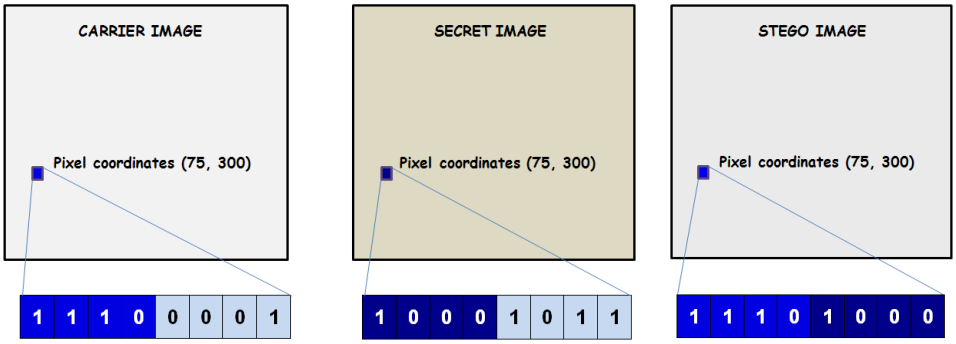
Hierarchy of joints defined by the skeletal tracking system is shown in Fig. 3. Unlike joints, bones are not explicitly defined in the API. Each bone is uniquely specified by the parent and the child joints that connect the bone. Bones rotation and movement follows the movement and orientation of its child joint. Joint *HipCenter* is the root joint and parent to all other, children joints. Other joints can have multiple parents. For example, joint *ShoulderCenter* has *HipCenter* as one of its parents and *Spine* joint as other.

B. *Rendering the 3D model*

In this section we explain how the movement of the person in front of the camera can be transferred to the movement of the virtual 3D model in the virtual scene by using Kinect. The model itself is created through a process of forming the shape of an object. It can be made in software which enables modeling, simulation, animation and rendering, such as Maya, Blender, MakeHuman and others. A 3D model is essentially a mathematical representation of a 3D object. It is a set of points in 3D space interconnected by lines, curves, surfaces, etc.

The virtual model used in this paper represents a virtual character or avatar, comprised of a skeleton and a mesh. The skeleton has one bone for each movable part of the model and it is used to animate the avatar [31]. The skeleton does not have a graphical representation, rather each bone is linked to a defined set of vertices. A collection of interconnected vertices defines the character's visible surface or skin of the model – mesh, which appears in the rendered frame. The process of building the character's armature or skeleton and setting up its segments to ease the animation process is called *rigging*. The process of attaching the defined bones to a mesh is called *skinning*. Rigging and skinning are prerequisites for model to be used in *skeletal animation* – a technique of animating the defined model used in this paper.

Nowadays, skeletal animation is widely used because it facilitates the often highly complex process of animating human characters by providing a simplified user interface. The implementation used in this paper is based on Kinect SDK, and it uses an extended version of Content Pipeline called *Skinned Model Pipeline* or *Skinned Model Processor*. This



(a) Carrier Image - Blue Component (b) Secret Image - Blue Component (c) Stego Image - Blue Component

Fig. 4: The process of creating a stego image.

pipeline enables 3D models, textures, animations, effects, fonts and similar to be imported to the project and processed. Once loaded, the 3D model can be rendered and animated.

Our solution enables the model in the scene to mimic the movements and gestures of people that are in the FOV of the Kinect camera. By using the Video data API (Data sources and Streams) the application can locate and retrieve information for twenty joints of the tracked user and compute real-time tracking information. The avatar is animated using the skeleton stream. The processing algorithm handles all of tracked skeletons and maps the movement of tracking to a movements of 3D model over time. The Bone Orientation API is used for calculating bone orientations. Also, it handles the animation by updating the skeleton movements and location with calculated information about transformations, such as translation, rotation and scaling of specific bone, from the relative bone transforms of Kinect skeleton.

Before the 3D model is rendered, it must be positioned in the virtual scene. The relationship between the avatar and the scene is defined and the position and size of the model in the scene is determined. Using the Color Stream, the Kinect video sequence is captured and set as background in the scene. Color video frame is stored and rendered as texture. XNA Skinning processor from Microsoft XNA Framework 4.0 [31] is used for rendering the 3D model skinned mesh on the screen, together with texture, lighting and optional shader effects.

C. Concealing the de-identified data in videos

To preserve and conceal the identity of the person in front of the Kinect device, we use the LSB insertion [28] steganographic algorithm. The picture of the virtual character rendered over the color image is used as vessel data or carrier image [32]. The part of the original image containing the person is embedded into the carrier image.

The LSB insertion algorithm [28] implements the idea of hiding one image inside another by using the least significant bits of one image to hold information about the most significant bits of another, secret image. The intuition of the approach is that with a change of the least significant bits, color value changes for each of the three components would be minor,

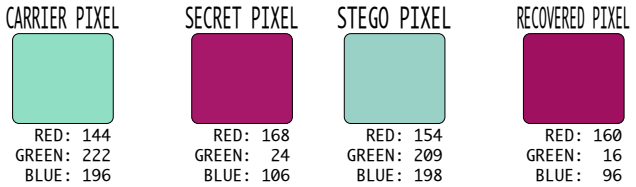


Fig. 5: Concealing the secret image in the carrier image using LSB insertion.

and the change in resulting color would not be conspicuous. If, for example, 4 bits are used for hiding the information, and one of pixel's components, such as blue, has a binary value of 11100001 (decimal 225), first 4 bits will be kept unchanged. Other 4 least significant bits will be replaced with 4 most significant bits from the secret image which will be hidden. If the value of the blue component of corresponding pixel in secret image is 10001101 (decimal 139), the combined picture, so called *stego-media*, will have the value of blue component of specific pixel set as 10111000 (decimal 232). Fig. 4 shows the process of replacing 4 bits of blue component of carrier image, with corresponding bits of secret image.

If we examine the numeric values of the resulting color combination of the R, G and B components we can notice that they have not changed much. Fig. 5 shows how pixels of different color can be combined without major changes in carrier pixel. The hidden image can be restored from stego-media using the reversed process. However, the values will be slightly different from the original values. In the process of extracting the hidden image, the last 4 least significant bits of stego image will be used as most significant bits of recovered image. The remaining least significant bits will be filled with zeros. The original image can also be recovered from the stego image. It is done by removing the exact number of least significant bits and filling the gaps with zeros. To process the whole image, it is necessary to loop over the combined image pixels and process all of the 3 components – red, green and blue.

Fig. 6: Top: two sample images captured by the Kinect device. Bottom: rendered avatars and pixelated silhouettes.

IV. Experiments

In this section, we qualitatively evaluate the output of the two stages of our system: (i) joint detection and 3D model rendering, and (ii) steganographic encoding.

A. Model rendering

Two example images produced by our system are shown in Fig. 6. As can be seen from Fig. 6, the rendered virtual character does not always completely cover the person in the video. This is due to erroneous output of the Kinect skeletal tracking system, as some joints can be incorrectly localized. Consequently, the positioning of the 3D model in the virtual scene can then be wrong.

In order to improve the results, we introduce the pixelization of the human silhouette, i.e. we pixelize the sections of the color image where the person is visible. By additionally blurring the image, the person is made unrecognizable. The color image itself is retrieved from the color stream which stays intact. For tracking the position of the skeleton another stream is used, and because of that pixelization or any other kind of image editing does not affect the movement of 3D model, so information about the movement and gestures of the person in camera's FOV is preserved.

The exact location of the person in the video frame is determined by using the Coordinate mapping API. Skeleton points are mapped to depth points, and assuming that resolution of depth and color images is identical, the screen coordinates of an object, or in our case the person, can be extracted. In the color image, pixel values are changed in the area that surrounds the person. The image processed this way is then used as background in the virtual scene, so when rendered, the 3D model covers this exact image, as can be seen in Fig. 6.

B. Steganographic encoding

As described previously, for steganographic encoding we use the LSB insertion algorithm [28]. The original image has a resolution of 640×480 pixels, with 8 bits per component. For output, we use the uncompressed Bitmap file format, which is suitable for easily storing and extracting the additional data.

Fig. 7 illustrates the quality of the stego image and the recovered image when using a variable number of bits for storing information. By increasing the number of bits for storing the secret image data, the quality of the secret image will increase. By using 5 or more bits of the original image to store the secret image, the stego image quality drops, while at the same time the quality of the recovered image increases. The quality of these two images is leveled when using 4 bits for hiding data. Stego image is then similar to the carrier image and the degradation of quality is not visible. It is also possible to recover a high-quality hidden image.

V. Conclusion

The solution presented in this paper offers detecting and tracking the movement of a person and hiding their identity. The motion of the person is captured using Kinect. The device sends the information about the skeleton position and movement of the bones which is then mapped to the movement of a virtual 3D humanoid model placed inside a virtual scene. In the scene itself, each frame of the color stream from Kinect is saved as texture and displayed in the background. The 3D model is always positioned closer to the camera, following the position of user, so when it is rendered, it covers the person's identity, but the movement and gestures are preserved.

The aim of this paper was to develop an application that will hide the identity of person by using a digital avatar and accurate tracking of person's movement. The goal was achieved, but some limitations still exist. The 3D model cannot perform a broad range of body movements. Actions and motions like jumping, rotating in position and crouching are not supported. There are some additional limits which pertain to the Kinect device. Two cameras, RGB and IR, are not physically located in the same spot, so a pixel in the image from one camera does not depict the same thing as a pixel in the image from the other camera, even though their screen coordinates are identical. This can cause errors in tracking of movements, which will then be mapped to a motions of a 3D model. Also, sudden moves will not be tracked accurately. Another restriction is the one related to the distance between objects and sensor. If an object or a person is too far away from the sensor, the camera will not recognize it. The same applies to an object being too close.

For the scene to be more natural, we used a humanoid avatar, although we are planning to replace it in the future improvements with an even more realistic one. In future research, we will also consider using multiple Kinect devices for capturing the joints data.

Acknowledgments

This work has been supported by the Croatian Science Foundation, within the project "De-identification Methods for Soft and Non-Biometric Identifiers" (DeMSI, UIP-11-2013-1544). This support is gratefully acknowledged.

(a) 1 hidden bit (b) 2 hidden bits (c) 4 hidden bits (d) 7 hidden bits

Fig. 7: Pairs of stego images (top) and recovered images (bottom) when using variable number of bits for storing the secret information.

REFERENCES

[1] K. Brkić, T. Hrkać, and Z. Kalafatić, "Detecting humans in videos by combining heterogeneous detectors," in *Proc. MIPRO*, May 2015.

[2] T. Hrkać and K. Brkić, "Iterative automated foreground segmentation in video sequences using graph cuts," in *Proc. GCPR (to be published)*, Oct 2015.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR*, pp. 886–893, 2005.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 1627–1645, Sept. 2010.

[5] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proc. BMVC*, BMVA Press, 2009. doi:10.5244/C.23.91.

[6] R. Benenson, M. Omran, J. Hosang, , and B. Schiele, "Ten years of pedestrian detection, what have we learned?," in *ECCV, CVRSUAD workshop*, 2014.

[7] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning.," in *Proc. CVPR*, pp. 3626–3633, 2013.

[8] W. Ouyang and X. Wang, "A discriminative deep model for pedestrian detection with occlusion handling," in *Proc. CVPR*, pp. 3258–3265, June 2012.

[9] W. Ouyang, X. Zeng, and X. Wang, "Modeling mutual visibility relationship in pedestrian detection," in *Proc. CVPR*, pp. 3222–3229, June 2013.

[10] P. Viola and M. Jones, "Robust real-time object detection," in *IJCV*, 2001.

[11] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *Int. J. Comput. Vision*, vol. 63, pp. 153–161, July 2005.

[12] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *ICPR (2)*, pp. 28–31, 2004.

[13] S. Brutzer, B. Hoferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *Proc. CVPR*, (Washington, DC, USA), pp. 1937–1944, IEEE Computer Society, 2011.

[14] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *Proc. CVPR*, 2014.

[15] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, pp. 743–761, Apr. 2012.

[16] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut" - interactive foreground extraction using iterated graph cuts," in *Proc. SIGGRAPH*, 2004.

[17] A. Hernandez-Vela, M. Reyes, V. Ponce, and S. Escalera, "Grabcut-based human segmentation in video sequences," *Sensors*, vol. 12, pp. 15376–15393, Nov. 2012.

[18] S. Poullot and S. Satoh, "Vabcut: A video extension of grabcut for unsupervised video foreground object segmentation," in *Proc. VISAPP*, 2014.

[19] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited: People detection and articulated pose estimation," in *Proc. CVPR*, 2009.

[20] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *Proc. CVPR*, (Washington, DC, USA), pp. 1385–1392, IEEE Computer Society, 2011.

[21] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proc. CVPR*, pp. 1653–1660, June 2014.

[22] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, pp. 116–124, Jan. 2013.

[23] V. Blanz, K. Scherbaum, T. Vetter, and H.-P. Seidel, "Exchanging Faces in Images," in *Proceedings of EG 2004*, vol. 23, pp. 669–676, Sept. 2004.

[24] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister, "Video face replacement," in *Proc. SIGGRAPH Asia*, (New York, NY, USA), pp. 130:1–130:10, ACM, 2011.

[25] B. Samaržija and S. Ribarić, "An approach to the de-identification of faces in different poses," in *Proc. MIPRO*, pp. 1246–1251, May 2014.

[26] H. S. (ed.), *Recent Advances in Steganography*. InTech, 2012.

[27] T. Morkel, J. H. P. Eloff, and M. S. Olivier, "An overview of image steganography.," in *ISSA* (J. H. P. Eloff, L. Labuschagne, M. M. Eloff, and H. S. Venter, eds.), pp. 1–11, ISSA, Pretoria, South Africa, 2005.

[28] N. F. Johnson and S. Jajodia, "Exploring steganography: Seeing the unseen," *Computer*, vol. 31, pp. 26–34, Feb. 1998.

[29] Z. Zhang, "Microsoft kinect sensor and its effect," *MultiMedia, IEEE*, vol. 19, pp. 4–10, Feb 2012.

[30] K. Lai, J. Konrad, and P. Ishwar, "A gesture-driven computer interface using kinect," in *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on*, pp. 185–188, April 2012.

[31] A. S. Lobao, B. P. Evangelista, and R. Grootjans, *Beginning XNA 3.0 Game Programming: From Novice to Professional*. Berkely, CA, USA: Apress, 1st ed., 2009.

[32] E. Kawaguchi and R. O. Eason, "Principles and applications of bpcs steganography," 1999.

An Extension to Hough Transform Based on Gradient Orientation

Tomislav Petković and Sven Lončarić
University of Zagreb
Faculty of Electrical and Computer Engineering
Unska 3, HR-10000 Zagreb, Croatia
Email: {tomislav.petkovic.jr, sven.loncaric}@fer.hr

***Abstract*—The Hough transform is one of the most common methods for line detection. In this paper we propose a novel extension of the regular Hough transform. The proposed extension combines the extension of the accumulator space and the local gradient orientation resulting in clutter reduction and yielding more prominent peaks, thus enabling better line identification. We demonstrate benefits in applications such as visual quality inspection and rectangle detection.**

***Index Terms*—Hough transform, gradient orientation**

I. Introduction

The Hough Transform (HT) is a commonly used method for line detection that is successfully applied in a large range of vision problems, starting from specific applications in industrial and robotic vision and extending to a general unconstrained problem of line detection in natural images. The success of the HT is primarily based on recasting a complex global line detection problem into a simple task of finding local peaks (concentrations of votes) in some parameter space.

The HT was proposed by Paul V. C. Hough [1] in 1962 and was introduced to computer vision community by Duda and Hart [2]. A comprehensive review of the HT was given by Illingworth and Kittler [3] in 1989. Later research introduced a randomized Hough transform (RHT) and its variants [4], [5], [6], [7] that eliminate the need for the quantized parameter space. Other improvements include better and more robust peak detection [8], [9] and extraction of line length [10], [11], [8]. Most of the above mentioned HT variants use only coordinates of extracted edge points and disregard other data that is often extracted from the input image during the edge or ridge detection.

In this paper we propose to improve upon a HT extension first suggested by O'Gorman and Clowes [12] that uses the gradient orientation to place a limit on the range of line orientation θ in the (θ, ρ) parameter space; if the detected gradient orientation is θ_0 then votes are only accumulated for the predetermined range $\Delta\theta$ around the θ_0, the interval $\langle\theta_0 - \Delta\theta, \theta_0 + \Delta\theta\rangle$. We extend this approach by combining the gradient orientation with the extension of accumulator space that makes straight line parametrization non-unique, but, combined with the range limit on orientation θ, offers advantages of further clutter reduction and yields more prominent peaks.

The paper is organized as follows: In Section II a brief review of HT is given. In Section III a proposed accumulator array extension is introduced. In Section IV some results are presented and discussed. We conclude in Section V.

II. The Hough Transform

Hesse normal form of a straight line is

$$\vec{r} \cdot \hat{n} - \rho = 0, \tag{1}$$

where $\vec{r} = \vec{\imath}x + \vec{\jmath}y$ is the location vector of the point (x, y), $\hat{n} = \vec{\imath}\cos\theta + \vec{\jmath}\sin\theta$ is the unit normal vector of the straight line and $\rho \geq 0$ is the distance to the origin. For the HT Eq. (1) is usually rewritten as

$$\rho = x\cos(\theta) + y\sin(\theta), \tag{2}$$

which defines a sinusoid in (θ, ρ) parameter space that corresponds to a point (x, y) in the input image. For the HT a sinusoid defined by Eq. (2) is drawn in the parameter space for every edge point (x, y). Straight lines present in the input image are given by (θ, ρ) coordinates of the local peaks in the parameter space.

The parameters θ and ρ are usually limited to either $[-\frac{\pi}{2}, \frac{\pi}{2}] \times \langle -\infty, +\infty\rangle$ or $[-\pi, \pi] \times [0, +\infty\rangle$ intervals, i.e. to ranges that produce unique mapping. For real world images $\pm\infty$ limit of parameter ρ is replaced by $\rho_{\max}$ defined by the finite size of the image.

Points (x, y) are selected by an edge detector, most often by thresholding a gradient of the input image. Therefore, in addition to point coordinates, the direction and magnitude of the gradient are also known. Let g_x and g_y be components of the gradient in x and y directions. The gradient direction vector is perpendicular to the local edge so, as shown in [12], the orientation θ can be estimated as

$$\theta \approx \operatorname{atan}\frac{g_y}{g_x}, \tag{3}$$

and all sinusoids in (θ, ρ) space may be drawn only for a small interval of angles centered around the estimate of Eq. (3), thus reducing the clutter.

III. Proposed Accumulator Extension

We propose to extend the accumulator so $\theta \in [-\pi, \pi]$ and $\rho \in [-\rho_{\max}, \rho_{\max}]$, where $\rho_{\max}$ is determined by the size of the input image. This extension makes the straight line parametrization in (θ, ρ) space non-unique: every straight line

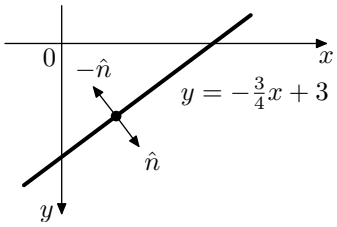


Fig. 1: A straight line $y = -\frac{3}{4}x + 3$ with two unit normals in opposite directions.

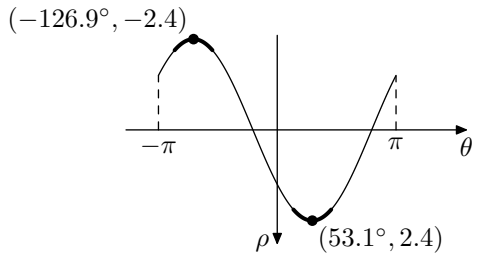


Fig. 2: HT of a point $(1.44, 1.92)$ on a line $y = -\frac{3}{4}x + 3$.

in the image is represented by two different points in the proposed accumulator range. However, those two representations have exactly opposite normal directions. For example, consider the line $y = -\frac{3}{4}x + 3$ shown in Fig. 1; two unit normals are $\hat{n}$, pointing away from the origin, and $-\hat{n}$, pointing toward the origin. The normal $\hat{n}$ corresponds to the equation $\frac{12}{5} = \frac{3}{5}x + \frac{4}{5}y$, while the opposite normal $-\hat{n}$ corresponds to the equation $-\frac{12}{5} = -\frac{3}{5}x - \frac{4}{5}y$, which defines the same straight line. The HT of the point $(1.44, 1.92)$ closest to the origin marked with dot in Fig. 1 is shown in Fig. 2; two dots in Fig. 2 correspond to two possible directions of the straight line normal.

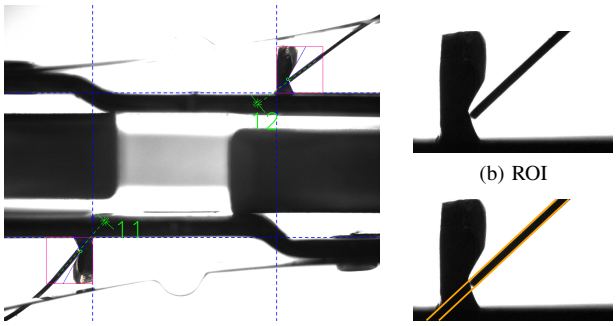
However, the parametrization in proposed extended accumulator becomes *unique* if the notion of line direction is introduced; let the line direction vector $\hat{d}$ be defined so $\hat{n}$ and $\hat{d}$ form a right-hand coordinate system. Therefore, instead of straight lines we are detecting oriented straight lines.

The proposed extension depends on the gradient orientation. Let g_x and g_y be components of the gradient at (x, y). The orientation θ for point at (x, y) can be estimated as

$$\theta \approx \operatorname{atan2}(g_y, g_x), \tag{4}$$

so the atan function of Eq. (3) yielding angles in the $[-\frac{\pi}{2}, \frac{\pi}{2}]$ interval is replaced by atan2 that yields angles in the $[-\pi, \pi]$ interval. The sinusoid in (θ, ρ) space is drawn only for a small interval of angles centered around the orientation estimate of Eq. (4), however, due to additional separation of points with opposing gradients we expect further clutter reduction in the HT domain.

Examining again the Fig. 2 demonstrates the difference: the plain HT would increment the accumulator array for all points on a sinusoid over $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ range, the HT using the gradient direction to limit the angle range would increment the accumulator on the segment around $(53.1°, 2.4)$ point, and the proposed method would increment the accumulator on either segment around $(53.1°, 2.4)$ or around $(-126, 9°, -2.4)$ point, depending on the gradient orientation.



(a) Annotated QC image (b) ROI (c) Found lines

Fig. 3: QC example: Positioning of a flat spring in regard to a bearing structure must be examined. (a) is annotated QC image showing measures of interest. (b) shows a ROI. (c) shows found boundary lines delineating a flat spring.

IV. Results and Discussion

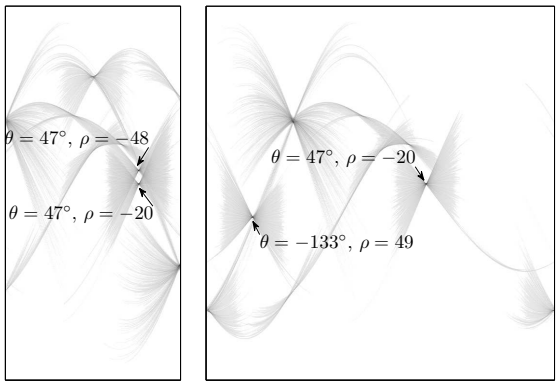
In this Section we present several examples demonstrating the advantages of the proposed extension to the HT. For all input images the coordinate axes are as shown in Fig. 1 with the origin in the center. For all accumulator arrays the coordinate axes are as shown in Fig. 2 with the origin in the center. Values of accumulator arrays are mapped through a square root function, linearly scaled to available dynamic range and inverted; pure black corresponds to the highest accumulator value, white is zero, and mid-levels are gray. Such mapping compresses the dynamic range, reduces the intensities of peaks, and makes butterfly shapes around peaks clearly visible. Line distance to origin ρ is measured in pixels and orientation θ in degrees. Range $\Delta\theta$ around θ_0 was set to $22.5°$.

A. Visual Quality Control

The HT is often applied in industrial vision tasks. We give two examples in visual quality control (QC) where a thin structure must be delineated.

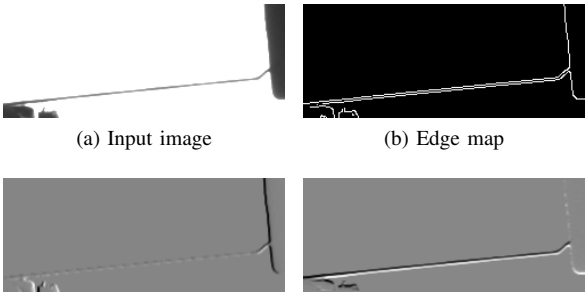
The first example is QC of a thin flat spring whose position against the bearing structure must be inspected. An example is shown in Fig. 3: the input image is annotated showing structures of interest. The accumulator arrays for the regular HT and for the proposed HT are shown in Fig. 4: note the increased separation of two peaks that correspond to the upper and lower straight lines that delineate the thin flat spring. This separation effect is caused by the proposed accumulator extension and the use of the extracted gradient orientation; it will reduce clutter in the accumulator when spatially close features in the input image have different gradient orientations.

The second example demonstrates a more difficult visual QC task where position and inclination angles of a thin flat spring must be inspected. The flat spring is about one pixel thin, which is the main difficulty. The input image and steps of



(a) Regular (b) Proposed

Fig. 4: Accumulator arrays for input image 3b. (a) is for regular HT where $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$. (b) is for proposed HT where $-\pi < \theta < \pi$; note the improved separation of two marked lines.



(a) Input image (b) Edge map

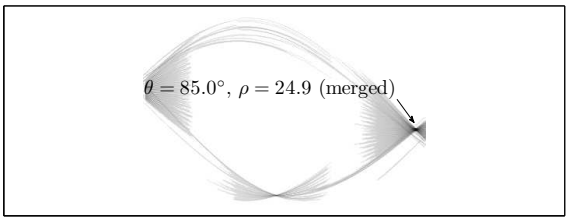
(c) g_x (d) g_y

Fig. 5: QC example: Examination of flat spring positioning. (a) is input image. (b) is edge map normally used HT. (c) and (d) are partial derivatives used to produce an edge map; two edges delineating the flat spring have clearly different g_y, but have the same properties in the edge map.

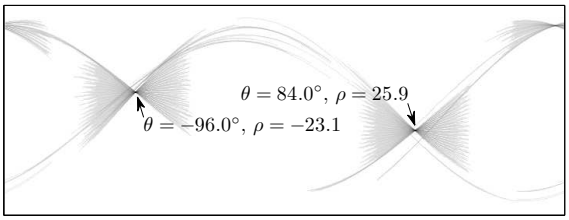
edge extraction are shown in Fig. 5: note the extreme thinness of the structure.

For regular HT an edge map shown in Fig. 5b is used. The problem with using only the edge map is that two edges corresponding to the upper and lower delineating straight line are close both in the image space and in the parameters space. Indeed, in the accumulator array shown in Fig. 6a two peaks are merged together and the separation of two delineating lines is not possible. Furthermore, the position of the peak corresponds neither to the upper nor to the lower delineating line, but is instead determined by the overlap of the butterfly shapes.

The proposed extension to the HT uses a gradient orientation in addition to the edge map. The gradient orientation is extracted from the gradient components g_x and g_y that are shown in Fig. 5, (c) and (d). Note that two edges of interest clearly have different properties when g_x and g_y are considered. Those differences are lost in the edge map where



(a) Accumulator array for regular HT where $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$



(b) Accumulator array for proposed HT where $-\pi < \theta < \pi$

Fig. 6: HT accumulator arrays for image 5a. Note merging of two peaks in (a) and a clear separation in (b).

Fig. 7: Lines delineating the flat spring extracted using the proposed HT.

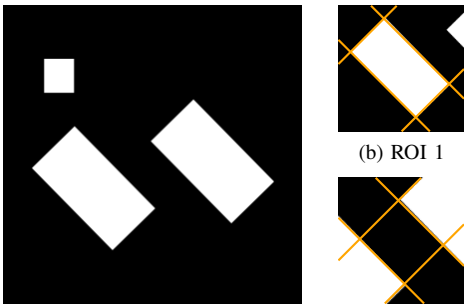
we observe only a combination $\sqrt{g_x^2 + g_y^2}$ (or $|g_x| + |g_y|$ for time-critical applications). Using the gradient orientation as proposed enables clear separation of two peaks as shown in Fig. 6 and extraction of both delineating lines is possible using the proposed accumulator array. The extracted lines using the proposed scheme are shown in Fig. 7.

For the two QC examples the gradient was computed using the Sobel operator, the edge map was computed using the Canny method of OpenCV [13] with 210 as the upper threshold, and the resolution of the accumulator was set to $\Delta\rho = 1$ and $\Delta\theta = 0.5°$.

B. Rectangle Detection

Another interesting example is rectangle detection in the HT domain proposed by Jung and Schramm [14]. Their method applies the regular HT to a sliding window and examines constellations of four peaks in the accumulator array. When the window center is at rectangle center a four-peak two-pair constellation $\big\{\{(\theta_1, \rho_1), (\theta_2, \rho_2)\}, \{(\theta_3, \rho_3), (\theta_4, \rho_4)\}\big\}$ describing a rectangle satisfies the following properties:

1) peaks in a pair have the same orientation, $\theta_1 = \theta_2 = \alpha$ and $\theta_3 = \theta_4 = \beta$,
2) orientations α and β are separated by $90°$ degrees,



(a) Input (b) ROI 1 (c) ROI 2

Fig. 8: A synthetic example for rectangle detection. Two ROIs are selected: ROI 1 (b) is centered around lower left rectangle and is an example of a proper response, and ROI 2 (c) is centered in the middle between two large rectangles and is an example of a false response.

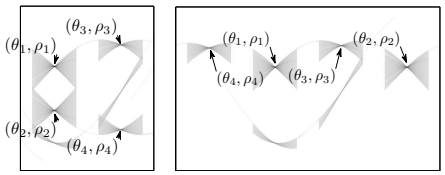
3) two peaks of a pair have canceling distance to origin, $\rho_1 + \rho_2 = 0$ and $\rho_3 + \rho_4 = 0$,
4) two peaks of a pair have the same height (equal to lengths of rectangle's sides a and b), and
5) the vertical distances within each pair are equal to lengths of rectangle's sides, $|\rho_1 - \rho_2| = a$ and $|\rho_3 - \rho_4| = b$.

In practice equality requirements are relaxed to absolute differences being less than chosen thresholds. The proposed HT allows expansion of the rectangle detection scheme by introducing the orientations of any of the four sides. The constellation properties change to:

1) peaks in a pair have the opposite orientations, $\theta_1 = \theta_2 + 180° = \alpha$ and $\theta_3 = \theta_4 + 180° = \beta \pmod{180°}$,
2) orientations α and β are separated by $90°$ degrees,
3) signs of all four ρ's are the same,
4) two peaks of a pair have equal distance to origin, $\rho_1 = \rho_2$ and $\rho_3 = \rho_4$,
5) two peaks of a pair have the same height (equal to lengths of rectangle's sides a and b), and
6) absolute sum of distances ρ within a pair is equal to lengths of rectangle's sides, $|\rho_1 + \rho_2| = a$ and $|\rho_3 + \rho_4| = b$.

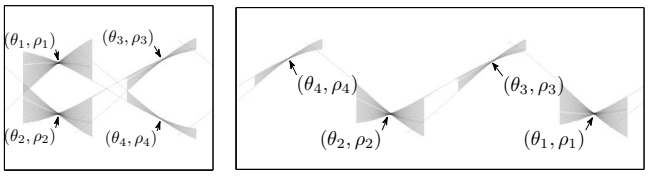
A synthetic example image with two ROIs is shown in Fig. 8. A typical rectangle constellation for the regular HT is shown in Fig. 9a. For the proposed HT the constellation of Fig. 9a is transformed to a constellation shown in Fig. 9b: four peaks around the origin are unwrapped into a line, the distance between four peaks along the θ axis is $90°$, and all ρ's have the same sign (gradients of all edges have directions either toward or away from the origin of the sliding window).

The original method [14] detects a false rectangle in ROI 2 (Fig. 8c) as the constellation shown in Fig. 10a matches. The proposed extension successfully eliminates such false rectangles as the constellation shown in Fig. 10b does not have all ρ's with the same sign (gradients of two edges are toward and of other two are away from the origin of the sliding window).



(a) Regular (b) Proposed

Fig. 9: Constellations describing a true rectangle corresponding to the sliding window shown in Fig. 8b.



(a) Regular (b) Proposed

Fig. 10: Constellations describing a false rectangle corresponding to the sliding window shown in Fig. 8c. Note that the false rectangle will not be detected if proposed HT is used as the constellation (b) does not match.

For two examples the gradient was computed using the Sobel operator, the edge map was computed using the Canny method of OpenCV [13] with 50 as the upper threshold, and the resolution of the accumulator was set to $\Delta\rho = 0.5$ and $\Delta\theta = 0.5°$.

C. Discussion and Implementation

The proposed extension to the HT is simple and effective. It can be easily included into advanced Hough schemes such as RHT.

The characteristic butterfly shape around peaks in the parameter space remains the same so accurate and robust peak detection scheme [8] and line segment estimator [10] are directly applicable and may be used without further modifications.

Note that this approach is different than computing the regular Hough transform twice, once for bright-to-dark and once for dark-to-bright edges as it avoids the problem of the preferred direction, i.e. reversing the preferred direction transforms any bright-to-dark edge to a dark-to-bright edge thus making all edges orthogonal to the preferred direction difficult to detect.

We have implemented the proposed improved HT in C/C++. The implementation is based on OpenCV [13] and is freely available under BSD license at http://www.fer.unizg.hr/ipg/resources/HT.

V. Conclusion

We proposed a simple and effective extension to the regular HT. The proposed extension combines the extension of the

accumulator array and the local gradient orientation resulting in clutter reduction and yielding more prominent peaks, thus enabling better line identification. We have demonstrated benefits in applications such as visual quality inspection and rectangle detection.

ACKNOWLEDGMENT

This research has been supported in part by the Research Centre for Advanced Cooperative Systems (EU FP7 #285939). The authors would like to thank Elektro-kontakt d.d. Zagreb, specifically mr. Ivan Tabaković, for provided images.

REFERENCES

[1] P. V. C. Hough, "Method and means for recognizing complex patterns," U.S. Patent 3,069,654, Dec. 1962, United States Atomic Energy Commission.

[2] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[3] J. Illingworth and J. Kittler, "A survey of the Hough transform," *Computer vision, graphics, and image processing*, vol. 44, no. 1, pp. 87–116, 1988.

[4] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized Hough transform (RHT)," *Pattern recognition letters*, vol. 11, no. 5, pp. 331–338, 1990.

[5] L. Xu and E. Oja, "Randomized Hough transform (RHT): basic mechanisms, algorithms, and computational complexities," *CVGIP Image Understanding*, vol. 57, pp. 131–131, 1993.

[6] J. Matas, C. Galambos, and J. Kittler, "Progressive probabilistic Hough transform." British Machine Vision Conference, 1998.

[7] L. Xu and E. Oja, "Randomized Hough transform," in *Encyclopedia of Artificial Intelligence*, J. R. R. Dopico, J. Dorado, and A. Pazos, Eds. IGI Global, 2008, pp. 1354–1361.

[8] Y. Furukawa and Y. Shinagawa, "Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space," *Computer Vision and Image Understanding*, vol. 92, no. 1, pp. 1–25, 2003.

[9] S. Guo, T. Pridmore, Y. Kong, and X. Zhang, "An improved Hough transform voting scheme utilizing surround suppression," *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1241–1252, 2009.

[10] M. Akhtar and M. Atiquzzaman, "Determination of line length using Hough transform," *Electronics Letters*, vol. 28, no. 1, pp. 94–96, 1992.

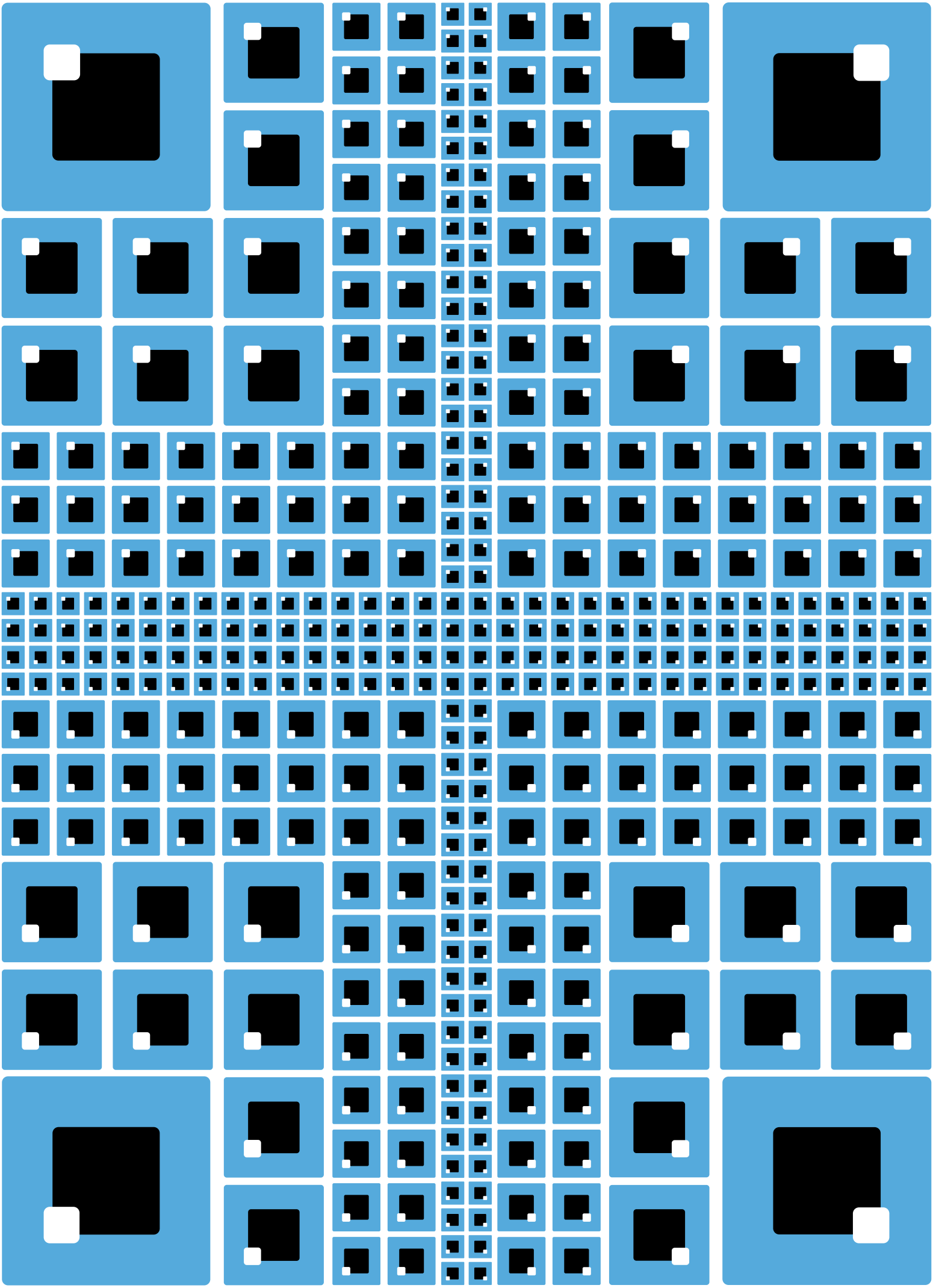
[11] M. Atiquzzaman and M. Akhtar, "A robust Hough transform technique for complete line segment description," *Real-Time Imaging*, vol. 1, no. 6, pp. 419–426, 1995.

[12] F. O'Gorman and M. Clowes, "Finding picture edges through collinearity of feature points," *IEEE Transactions on Computers*, vol. 25, no. 4, pp. 449–456, 1976.

[13] "OpenCV [Open Source Computer Vision] library," http://opencv.org/, Jan. 2014.

[14] C. R. Jung and R. Schramm, "Rectangle detection based on a windowed Hough transform," in *Computer Graphics and Image Processing, 2004. Proceedings. 17th Brazilian Symposium on*. IEEE, 2004, pp. 113–120.

Author Index



Proceedings of the Croatian Computer Vision Workshop CCVW 2015, Year 3 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