AN APPROACH TO ON-ROAD VEHICLE DETECTION, DESCRIPTION AND TRACKING

Nikolay Chumerin, Marc M. Van Hulle

K.U.Leuven, Laboratorium voor Neuro- en Psychofysiologie
Campus Gasthuisberg, Herestraat 49, B-3000 Leuven, BELGIUM
E-mail: {nikolay.chumerin, marc.vanhulle}@med.kuleuven.be

ABSTRACT

We present an approach to detecting *independently moving objects* (IMOs) in stereo video sequences acquired by on-board cameras on a moving vehicle. The proposed approach is based on the processing of two independent information streams: an *independent motion detection stream* and an *object recognition stream*. Fusion of these streams outputs allows our system to segment IMOs, track them, and even estimate some of their properties.

1. INTRODUCTION

Automatic intelligent navigation systems in cars and trucks have gained recently much attention. This can be explained by the constantly increasing motorization, which is in turn resulting in a corresponding growth in car accidents and road casualties.

The detection of *independently moving objects* (IMOs) in video sequences acquired during driving is a much more challenging task compared to moving object detection by a static observer. The problem is complicated by a number of factors such as ego-motion, camera vibrations, imperfect calibration of the on-board cameras, complex outdoor environments *etc*. A detailed review of this topic is out of scope of this work and we refer to [1, 2] for decent surveys.

In this study, we propose a novel approach that allows a detection of IMOs by processing and subsequent fusing two cooperative information streams (see Fig. 1): the *independent motion detection stream* and the *object recognition stream*. None of these streams alone can provide a satisfactory quality of the IMOs detection. Using only the motion stream leads to discontinuous and sparse IMOs representations. In this case, additional efforts are needed for delineation and classification of the IMOs. The recognition stream deals with static images (does not use temporal information) and therefore can not distinguish between independently moving and static (with respect to environment) objects. One should note that the idea of the two processing streams is widely accepted and in the visual neuroscience [3].

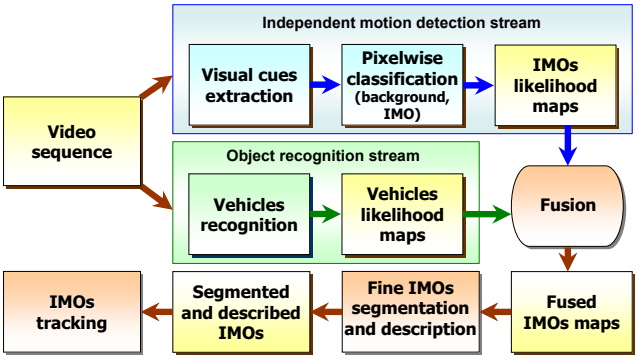

Fig. 1. Outline of the proposed model.

2. INDEPENDENT MOTION STREAM

The problem of *independent motion* detection can be defined as the problem of locating objects that move independently of the observer in his field of view. In our case, we build so-called *independent motion maps* where each pixel encodes the likelihood of belonging to an IMO. For each frame we build an independent motion map in two steps (see Fig. 1): early vision cues extraction and classification.

As vision cues we consider: *stereo disparity* (three components – for current, previous and next frame), *optical flow* (two components) and *normalized coordinates*[1] (two components). The optic flow and stereo disparity are computed using multiscale phase-based optic flow and stereo disparity algorithms [4, 5]. Unfortunately, there are no possibilities to

NC is supported by the Belgian Fund for Scientific Research – Flanders (G.0248.03).

MMVH is supported by the Excellence Financing program of the K.U.Leuven (EF 2005), the Belgian Fund for Scientific Research – Flanders (G.0248.03, G.0234.04), the Flemish Regional Ministry of Education (Belgium) (GOA 2000/11), the Belgian Science Policy (IUAP P5/04), and the European Commission (NEST-2003-012963, IST-2002-016276, IST-2004-027017)).

[1] By normalized coordinate system on a frame we mean the rectangular coordinate system with origin in the center of the frame, where upper-left corner is $(-1, -1)$ and lower-right is $(1, 1)$.

estimate reliably all these cues for every pixel in the entire frame. This means that the motion stream contains incomplete information, but this gap will be bridged after fusion with the recognition stream.

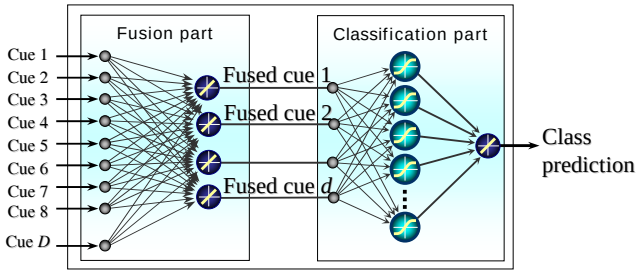

Fig. 2. MLP used as classifier in motion stream.

We consider each pixel as a multidimensional vector with visual cues as components. We classify all the pixels (which have every component properly defined) in two classes: IMO or background. We have tried a number of setups for classification, but the optimal performance was obtained with a multilayered perceptron (MLP) with three layers: a linear (4–8 neurons), a nonlinear layer (8–16 neurons), and one linear neuron as output.

After training, the MLP can be used for building an IMO likelihood map I for the entire frame:

$$I(x, y) = p\left(IMO|(x, y)\right), \qquad (1)$$

where x, y are pixel coordinates.

Fig. 3. (Left) Frame number 342 of motorway3 sequence. (Right) Matrix I, output of the motion stream for the same frame. Value $I(x, y)$ is defined as probability of pixel (x, y) being part of an IMO.

3. RECOGNITION STREAM

For the recognition of vehicles and other potentially dangerous objects (such as bicycles and motorcycles but also pedestrians), we have used a state of the art recognition paradigm – the convolutional network LeNet, proposed by LeCun and colleagues [6]. Modifications of LeNet were successfully exploited for generic object recognition [7] and

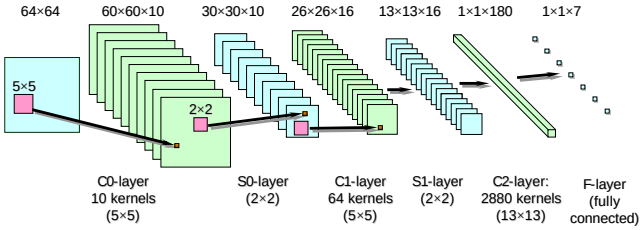

Fig. 4. LeNet – a feed-forward convolutional neural network, used in recognition stream.

even for autonomous robot's obstacle avoidance system [8]. We have used the CSCSCF configuration of LeNet (see Fig. 4) comprising six layers: three convolutional layers (C0, C1, C2), two subsampling layers (S0, S1) and one fully connected layer (F). As an input LeNet receives a 64×64 grayscale image. Layer C0 convolves the input with ten 5×5 kernels, adds (ten) corresponding biases, and passes the result to a squashing function[2] to obtain ten 60×60 feature maps.

In layer S0, each 60×60 map is subsampled to a 30×30 map, in such a way that each element of S0 is obtained from a 2×2 region of C1 by summing these four elements, by multiplying with a coefficient, adding a bias, and by squashing the end-result. For different S0's elements, the corresponding C1's 2×2 regions do not overlap. The S0 layer has ten coefficient-bias couples (one couple for each feature map). Computations in C1 are the same as in C0 with the only difference in the connectivity: each C1 feature map is not obtained by a single convolution, but as a sum of convolutions with a set of previous (S0) maps (see Table 1). Layer S1 subsamples the feature maps of C1 in the same manner as S0 subsamples the feature maps of C0. The final convolutional layer C2 has kernels sized 13×13 and 180 feature maps which are fully connected to all S1's 16 feature maps. It means that the number of C2 kernels is $16 \times 180 = 2880$, and the corresponding connectivity matrix should have all cells shaded. The output layer consists of seven neurons, which are fully connected to C2's outputs. It means that each neuron in F (corresponding to a particular class *background*, *cars*, *motorbikes*, *trucks*, *buses*, *bicycles* and *pedestrians*) just squashes the biased weighted sum of all C2's outputs.

LeNet scans the input image (left frame) in two scales, 320×256 and 640×512, with a 64×64 sliding window and in 8 and 16 steps, respectively. For each position of the sliding window, we add the output of the class to the corresponding (window) range in a 320×256 matrix. In such a way we obtain seven matrices $R_0, \ldots, R_6$ which, after normalization, are regarded as likelihood maps for the considered classes.

[2] $f(x) = A \tanh(Sx)$, $A = 1.7159$ and $S = 2/3$ according to [6].

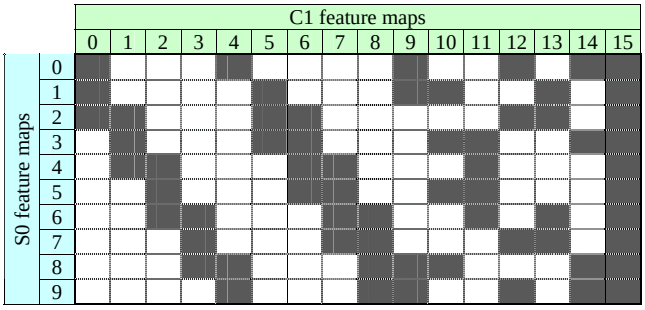

Table 1. S0-C1 connectivity matrix. Shaded cell which belongs to the i-th column and j-th row indicates that the j-th feature map of S0 participates in the computation of the i-th feature map of C1. For example, to compute the fourth feature map of C1, one has to find a sum of convolutions of S0 feature maps 0, 8 and 9 with correspondent kernels. The number of kernels in C1 (the number of shaded cells in the table) is 64.

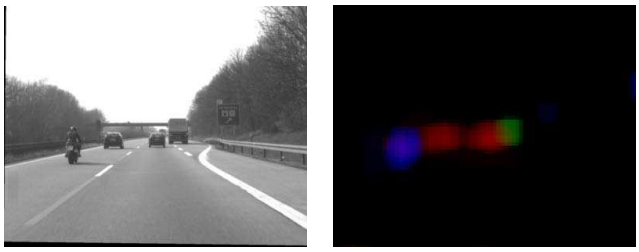

Fig. 5. (Left) Frame number 342 of motorway3 sequence. (Right) Output of the recognition stream for the same frame. Here, we used different colors to present different classes: black for background, red for cars, blue for motorcycles and green for trucks.

Note that, for further processing, the most important map is R_0, which corresponds to the background class and the so-called *non-background* map is obtained as $(1 - R_0)$. The rest of the maps $R_1, \ldots, R_6$ are responsible only for IMO classification.

4. TRAINING

For training both streams, we used two rectified stereo video sequences, each consisting of 450 frames. We have labeled IMOs in all left frames of the sequences. These labels were used for training motion stream classifier as well as for preparing the training dataset for LeNet.

We have used small batches with the increasing size version of the BFGS Quasi-Newton algorithm for the independent motion classifier training. Samples for each batch were randomly taken from all the frames of all the scenes. Training was stopped after reaching 0.04 (MSE) performance.

To train LeNet, we have prepared a dataset of 64×64 grayscale images (approximately 67500 backgrounds, 24500 cars, 2500 motorbikes, 6200 trucks, 1900 bicycles, 78 buses, and 3500 pedestrians). We have doubled the dataset by including horizontally flipped versions of all the samples. Images were taken mainly from publicly available object recognition databases (LabelMe[3], VOC[4]). A stochastic version of the Levenberg-Marquardt algorithm with diagonal approximation of the Hessian [6] was used for LeNet training. Training was stopped after reaching a misclassification rate less than 1.5%. To increase the robustness of the classification, we have run the training procedure several times, every time by adding a small (2%) amount of uniform noise and by randomly changing the intensity (97–103%) of the each training sample.

5. FUSION OF THE STREAMS

Fusion of the streams for a particular frame is achieved in three steps.

1. Intersection of the independent motion map I with the mask M of the most probable locations of the IMOs in the frame (see Fig. 6):

$$F_1(x, y) = I(x, y)M(x, y). \quad (2)$$

2. Intersection of the previous result F_1 with the non-background map $(1 - R_0)$:

$$F_2(x, y) = F_1(x, y)(1 - R_0(x, y)). \quad (3)$$

3. Intersection of the previous result F_2 with the likelihood maps $R_1, \ldots, R_6$ of each class, which results in six maps $L_1, \ldots, L_6$ (one for each class, except the background):

$$L_k(x, y) = F_2(x, y)R_k(x, y), \quad k = 1, \ldots, 6. \quad (4)$$

The first step is necessary for rejecting regions of the frame where the appearance of the IMOs is implausible. After the second step we obtain crucial information about regions which have been labeled as non-backgrounds (vehicles or pedestrians) and which, at the same time, contain independent moving objects. This information is represented as saliency map F_2 which we will use further for IMOs detection/description and in the tracking procedure. The third step provides us the information needed in the classification stage.

[3] http://labelme.csail.mit.edu/

[4] http://www.pascal-network.org/challenges/VOC/

Fig. 6. Matrix M, mask of possible IMO appearance in a frame.

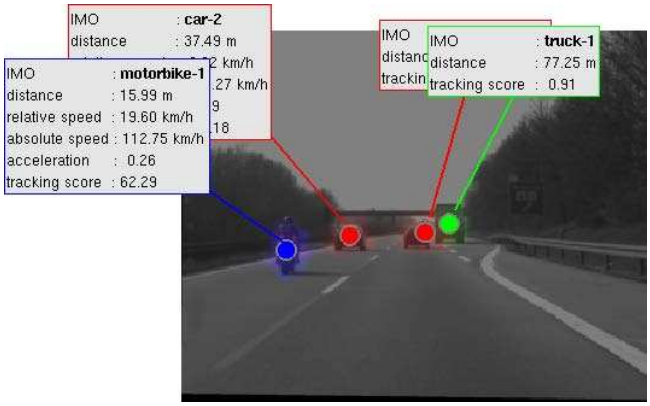

Fig. 7. Result of tracking and description of IMOs.

6. IMOS DETECTION AND TRACKING

For an IMO detection we have used a simple technique based on detection of the local maximas of the maps defined in (3). We have performed a spatio-temporal filtering (i.e. for i-th frame we apply smoothing of a three-dimensional array – a concatenation of the $(i-2)$-th, $(i-1)$-th, i-th, $(i+1)$-th and $(i+2)$-th two-dimensional maps along the third time-dimension). Then we search for local maximas in the entire (i-th) filtered frame and consider them as the IMO centers $\mathbf{x}_k$ for this frame.

For a tracking of each IMO, we have introduced a parameter called *tracking score*. For a particular IMO, we increase this parameter when in the next frame, only in a small neighbourhood of the IMO center there is a good candidate for the considered IMO in the next frame, namely the IMO with the same class label, and approximately with the same properties (size, distance and relative speed in depth). Otherwise, the tracking score is decreased. An IMO survives while the tracking score is above a fixed threshold. The tracking score works as a momentum and allows the system to keep tracking an IMO even when there are no sufficient data in the next few frames.

7. CLASSIFICATION AND DESCRIPTION OF THE IMOS

As soon as we are able to detect IMOs, it becomes possible to classify them and retrieve their properties (size, absolute speed in depth, relative speed in depth, time to contact, absolute acceleration *etc*).

The *class* c_k of the k-th IMO we define as:

$$c_k = \arg \max_{1 \le c \le 6} \{L_c(\mathbf{x}_c)\}, \qquad (5)$$

where $\mathbf{x}_k = (i_k, j_k)$ is the center of the k-th IMO (in image domain D) and L_c are the maps, defined in (4).

For the k-th IMO's *size*, σ_k, estimation, we search for a $\sigma > 0$, where the first minimum of the function (6) takes place.

$$\Delta_k(\sigma) = \int_D \left| L_{c_k}(\mathbf{x}_k) e^{-||\mathbf{x}_k - \mathbf{x}||^2/\sigma^2} - L_{c_k}(\mathbf{x}) \right| d\mathbf{x}. \qquad (6)$$

The IMO's *distance* estimation is a crucial point in the retrieval process. Using an averaged (in a small neighborhood of the IMO's center) disparity and known cameras calibration parameters, we have computed the distance to the IMO. To compensate for instabilities in the distance estimations, we have used an robust linear regression based on the previous five estimates.

Most of the present-day motor vehicles are being equipped with an increasing number of electronic devices, including control units, sensors, actuators *etc*. All these devices communicate with each other by vehicle data bus. During recording sessions we have stored some crucial data from the vehicle data (namely CAN) bus synchronized with video. This has provided us with precise estimations of ego-motion speed.

The *relative speed in depth*, we estimated as the derivative (with respect to time) of the distance using robust linear regression based on the last five estimations of the distance. To estimate the *time to contact*, we have divided the averaged distance by the averaged relative speed in depth. Using the precise value of the ego-motion speed from the CAN-bus data, and simply by adding it to the relative speed in depth we have also obtained the *absolute speed in depth* of the considered IMO.

The derivative of the absolute speed in depth can be considered as an estimation of the *acceleration* (it is true only in the case when the ego-heading is collinear to the heading of the entire IMO). An example of IMO tracking and the retrieved properties is shown in Fig. 7.

8. CONCLUSIONS

The two streams approach we presented is successful in IMOs detection and classification, and allows for an easy tracking and properties retrieval (Fig. 7). By mixing IMO maps and class likelihood maps we increase the reliability of the detected IMOs and automatically clean up the false positives. This is a crucial issue when video streams obtained from moving cameras are used.

The main drawback of the presented approach is the unsatisfactory computation time. The existing model is still far from real-time systems, but we see a number of ways to increase its speed. In order to speed up the most computationally expensive part (recognition) we propose:

- to build the object likelihood maps not for the entire frame, but only for salient regions of the independent motion map (this has obvious biological support: for biological visual systems, attention is drawn to moving objects);
- to replace LeNet's first processing layer (C0) with a bank of Gabor-like (fixed/non-trainable) filters which are used for visual cue extraction in the independent motion stream;
- to optimize sliding window scanning procedure in order to eliminate redundant computations of the overlapping regions.

Both processing streams of the model have feed-forward architectures, which can be easily implemented in hardware such as Field-Programmable Gate Arrays (FPGAs). Moreover, as far as the streams are independent, they can be implemented as separate FPGAs, working in parallel.

As it was mentioned in Section 7, via CAN-bus we can access to on-board sensor data. Some of the sensors can provide us with precise and reliable information. For example, Adaptive Cruise Control (ACC) system uses a LIDAR (Light Detection and Ranging) sensor to track up to ten objects and estimate their relative position, speeds and accelerations. Unfortunately ACC system alone can not distinguish the IMOs from the static objects. We see the incorporation of the ACC track data in the our model as one more way to improve the model.

9. REFERENCES

[1] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: a review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 5, pp. 694–711, 2006.

[2] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and Vision Computing*, vol. 21, no. 4, pp. 359–381, 2003.

[3] L.G. Ungerleider and T. Pasternak, "Ventral and dorsal cortical processing streams," *The Visual Neurosciences*, vol. 1, no. 34, pp. 541–562, 2004.

[4] K. Pauwels and M.M. Van Hulle, "Optic flow from unstable sequences containing unconstrained scenes through local velocity constancy maximization," Edinburgh, 2006, vol. 1, pp. 397–406.

[5] S.P. Sabatini, G. Gastaldi, F. Solari, J. Diaz, E. Ros, K. Pauwels, M.M. Van Hulle, N. Pugeault, and N. Krueger, "Compact and accurate early vision processing in the harmonic space," Barcelona, 2007.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," 1998, vol. 86, pp. 2278–2324.

[7] Y. LeCun, F.J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," 2004, vol. 2.

[8] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," *Advances in neural information processing systems*, vol. 18, 2006.