Implementation and evaluation of a large-scale object detection system

A.Gepperth

November 17, 2010

Abstract

In this report, we give detailed information about implementation and evaluation efforts that were undertaken w.r.t. the system described in [1]. Particular emphasis is given to system-level learning of contextual dependencies, and in particular to the computation of visual features used by the employed learning algorithms. Another important issue treated here concerns the evaluation of the system: we give details about the evaluation procedures and the HRI RoadTraffic dataset which is the basis of the evaluation efforts. As a last point, we compare the HRI RoadTraffic dataset to other publicly available, annotated datasets for road traffic and present arguments why the HRI RoadTraffic dataset constitutes a suitable reference dataset for future benchmarking efforts.

1 System overview

The system described in [1] is shown in Fig. 1. It is a hybrid system composed from elements implementing neural as well as conventional image processing techniques. This section will review all relevant elements of the system, where emphasis is placed on the interfacing of elements rather than on implementation details which are described either in the appendices or in dedicated publications. Furthermore, we will give details about experimental procedures that were used to generate the results of [1]. Relevant parameter values of algorithms described in this report are given in 2. The presented system (see Figs. 2 for a detailed sketch) is of significant complexity and consists of approximately 8000 functional components. The system receives inputs obtained from a stereo camera, the vehicle-internal CAN bus and two laser range-finding sensors; it generates a list of entities that are judged to be relevant, i.e., cars and vehicles. Since the currently achieved execution speed is not sufficient for live operation, the system is not yet running in a vehicle. Rather, it receives its inputs by a timestep-based replay of recorded data, which is exactly equivalent to the way data would be received in our prototype vehicle.

Since the system is not operating on "live" data, it is possible to replay annotations (e.g., positions and identities of other traffic participants) as "virtual

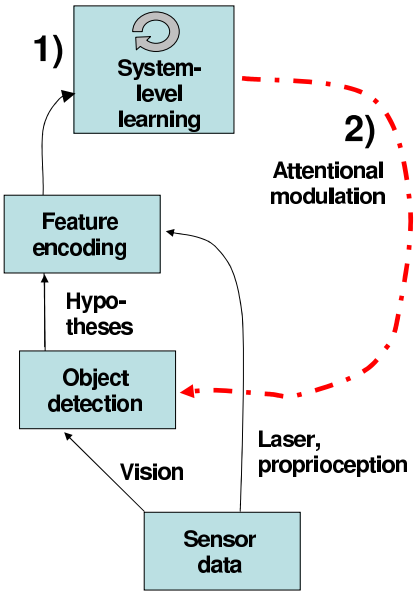

Figure 1: Illustration of the basic structure and the inherent novel points of the system presented in [1]. 1) Learning of system-level models for generating attentional modulation *during system operation* 2) Learning of multimodal models for attentional modulation. What kinds of models are learned effectively depends only on the processing results that are supplied to the system-level learning mechanism.

.

sensors", that is, as if they were obtained from measurements. This is useful for the training of system-level models (see [1]), although it is for convenience only because we will show that bootstrapping, i.e., model training without using annotated data, is possible and feasible (see [1]).

1.1 Interfacing of system components by population coding

Population coding is a biologically inspired way of encoding information. Basic properties of all population coding models[2, 3, 4] are, one the one hand, the representation of information on two-dimensional surfaces in analogy to cortical surfaces, and on the other hand, the concept of storing confidence distributions

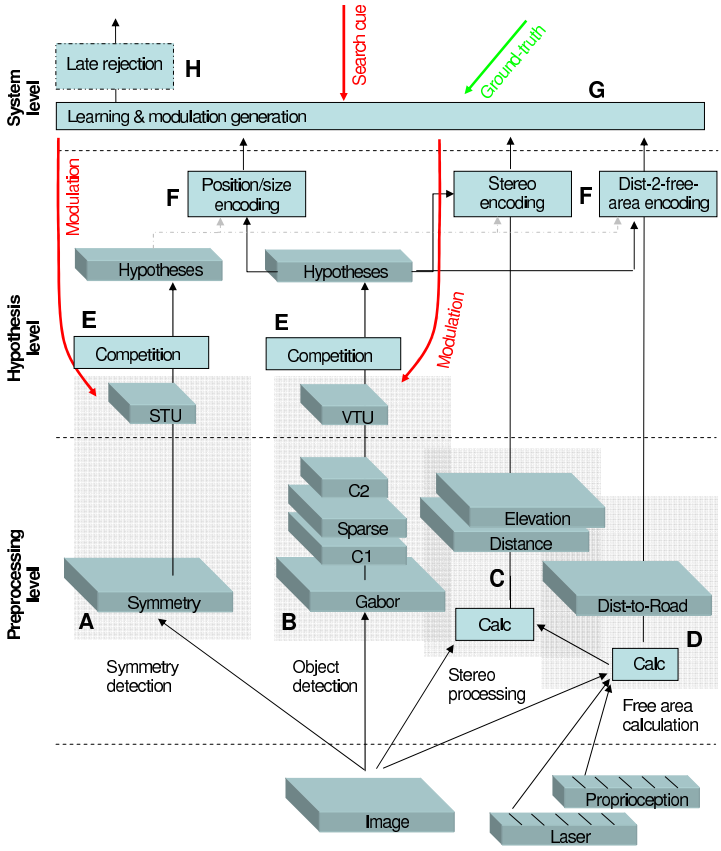

Figure 2: Global structure of the described hierarchical object detection system. Functional modules are object detection (**A,B**), stereo processing (**C**), free-area computation (**D**), competitive hypothesis selection (**E**) and population encoding (**F**). Attentional modulation is trained at the system-level **(G)**, linking hypothesis identity to elevation, distance, distance-to-free-area and 2D image position (**F**). System-level training happens in a supervised way using "true" object identities supplied by *ground-truth data*. Given an arbitrary desired object identity (the *search cue*), attentional modulation is applied to the hypothesis level of object detection, thus favoring the detection of objects of the desired identity. Data flows from symmetry detection (**A**) to other modules are identical to data flow from the appearance-based classifier (**B**) but are not shown for clarity. For comparison, we also implemented a "late rejection" module at system level (**H**) which uses a multilayer perceptron for directly (without influencing lower system levels) mapping population-coded quantities produced at the hypothesis level to an object identity decision.

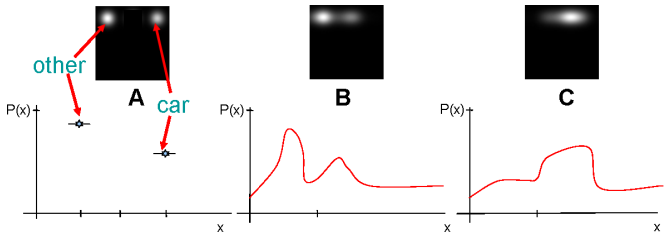

Figure 3: Transfer of different types of measurements to population codes. The particular type of measurement determines how it is translated into a population code. **A)** a discrete distribution from, e.g., an object classifier, is translated into a population code where only certain locations carry information. **B),C)** Quasi-continuous one-dimensional measurements (e.g., object elevation and distance) are encoded into population codes that are extended along one axis. Note that the uncertainty (multimodality) of measured distributions is transferred to the resulting population code. The precise way of encoding is determined on a case-by-case basis.

for all represented quantities.

Mathematically, a population code is therefore a collection of one or several two-dimensional lattices, where each lattice point ("neuron") stores a normalized confidence value corresponding to the belief that a certain property ("preferred stimulus") associated with this lattice point is present in the encoded information. These properties link population coding closely to the Bayesian approach to probability[5]. In particular, population coding represents encoded quantities as distributions over possible values, thus implicitly storing the associated uncertainty in accordance with the "Bayesian brain" hypothesis[6].

In order to be able to link system-level information by learning methods as described in [1], we convert such quantities into population codes. The system-level quantities we want to encode are confidence distributions which may be either one- or two-dimensional which we denote *source distributions*. The nature of source distributions may be discrete (i.e., having nonzero confidence values only at certain positions) or continuous as well as graded (with confidences assuming values in a range between 0.0 and 1.0) or binary. Examples of different kinds of source distributions and their population encoding are shown in Figs. 3 and 8. We employ the convolution coding technique [2] which essentially performs a convolution of the source distribution with a kernel of fixed size, in our case a Gaussian. In case a source distribution is one-dimensional, we embed it into a two-dimensional distribution along a specified axis before performing convolution coding.

The conversion of system-level quantities into population codes is not uniquely determined; we employ the freedom this gives us to make the encoding procedure as simple as possible. The quantities which are transferred to population

Figure 4: Selected example images from streams I-V of the HRI RoadTraffic dataset. All videos were taken in RGB color using a MatrixVision mvBlueFox camera at a resolution of 800x600. Used frame rates were 10Hz except for video II where a setting of 20Hz was used. Aperture was always set to 4.0 except for video IV where we used a value of 2.4. A self-implemented exposure control was used on both cameras, manipulating the gain and exposure settings of each camera.

codes are properties of object hypotheses generated by the system: elevation, distance, distance-to-free-area and retinal position/size as described in Secs. 5, 7 and 8.

2 The HRI RoadTraffic dataset

For evaluation of object detection systems such as [1], we created the HRI RoadTraffic benchmark dataset. It consists of five distinct video streams recorded from a vehicle during a significant range of traffic, environment and weather conditions. All videos are around 15 minutes in length and were taken during test drives along a fixed route covering mainly inner-city areas, along with short times of highway driving. Relevant occuring objects are mainly vehicles of all types as well as traffic signs and static road elements (signal boards etc.). Please see Tab. 1 for details and Fig. 4 for a visual impression. For the quantitative evaluation of object detection performance, relevant objects are annotated in every tenth image of the recorded video streams, please refer to Sec. 9 for details. Additionally, processing results of the free area computation used in [1] (see Sec. 5) are included so that researchers can more easily reproduce our results. Quantities of general interest in image processing, such as the current speed and yaw rate, are also available. This dataset can be obtained by email request from the authors of [1]. Details about the content of the dataset are given in Sec. 12.

ID	weather	daytime	single images	annotated images
I	overcast,dry	afternoon	9843	957
II	low sun, dry	late afternoon	22600	949
III	heavy rain	afternoon	6725	643
IV	dry	midnight	6826	464
V	after heavy snow	afternoon	16551	867

Table 1: Details about the video streams in the HRI RoadTraffic dataset. Please note that streams II and V were recorded at a frame rate of 20Hz.

3 Multiscale processing by appearance-based classifier

Cars are detected at 16 different scales. These scales cover cars with a width between 35 and 442 pixels on a 800x600 input resolution. The smallest 6 scales are realized by learning differently sized SLP (single-layer perceptron) templates and using them on the original scene resolution. In this way, for distant cars the full pixel information could be exploited. For detecting nearer cars, the largest template is applied to an image resolution pyramid using a sample factor of 0.878. This is computationally more efficient than a further increase of the template size and also avoids over-fitting. Using this scale setup we can in principle detect cars in a distance range of 4 to 60 meters.

4 Training of appearance-based classifier

For the training of car templates at 6 scales (see previous section), we use annotations from the day streams I-III from the HRI RoadTraffic dataset. Each stream is divided into chunks of 30 seconds, where all chunks with odd index are used to obtain training data from annotations (see Sec. **??**) whereas the remaining chunks provide test data. For generating positive training examples, all fully visible (not occluded) cars are cropped from the training scenes. Training is performed separately for each of the 16 used scales using positive examples of at least the size required at a certain scale. Larger segments are down-scaled to the desired resolution, smaller segments are neglected. In this way, more positive examples can be used for the training of classifiers at smaller scales. Negative examples are generated by randomly selecting image regions not containing cars.

5 Free-area and distance-to-free-area

For computing the binary free area image $\tilde{F}^{\text{free-area}}(\vec{x}) \equiv \tilde{F}^0(\vec{x})$, we employ a hybrid approach based both on visual processing and laser-range finder devices. Detailed descriptions of the employed algorithms can be found in [7]. The feature map relevant to the operation of the described system is the *retinotopic*

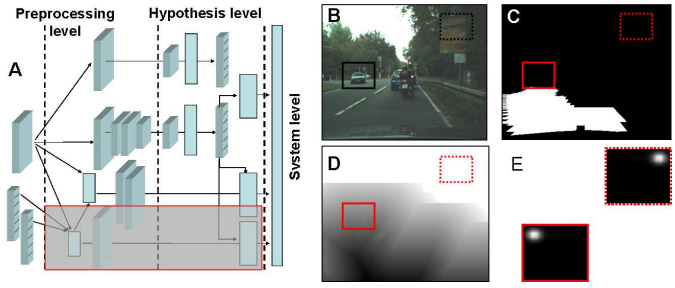

Figure 5: Performance example of free-area and distance-to-free-area computation for two object hypotheses. **A)** Embedding into processing system **B)** Video image **C)** computed free area $\tilde{F}^1(\vec{x})$ **D)** pixelwise distance-to-free area map $F^1(\vec{x})$. Each pixel value in the map is determined by that pixel's minimal distance to a computed free-area pixel. Due to computational reasons, an upper limit $d_{\max}$ is imposed. Note that distances are negative for pixels on the free area. **E)** Population codes obtained for two different object hypotheses. The encoded value is represented by a Gaussian of fixed variance whose center position varies along the x-coordinate depending on the distance-to-free-area of the center pixel of the object hypothesis. The y-coordinate of the center position is kept constant at $y = 10$.

distance-to-free-area representation $F^0(\vec{x})$. It is computed as follows:

$$F^0(\vec{x}) = \begin{cases} min_{\vec{x}',\ \tilde{F}^0(\vec{x}')\neq 0} d(\vec{x},\vec{x}') & \tilde{F}^0(\vec{x}) = 0 \\ -min_{\vec{x}',\ \tilde{F}^0(\vec{x}')=0} d(\vec{x},\vec{x}') & \tilde{F}^0(\vec{x}) \neq 0 \end{cases}$$

For a hypothesis with center pixel $\vec{x}_c$, the population-coded representation of distance-to-free-area, $f^0(\vec{p})$ is computed at the hypothesis level of the system as

$$\vec{p}_c = \left[\frac{min(F^0(\vec{x}_c), d_{\max})}{d_{\max}} n, 10\right]^T$$
$$f^0(\vec{p}) = \exp \frac{(\vec{p}-\vec{p}_c)^2}{2\sigma^2} \quad (1)$$

Please view Fig. 5 for examples of free-area computation and the transfer of the corresponding distance-to-free-area measurement to population codes.

6 Symmetry-based object detection

Confidence maps are generated by finding regions in the image I that are symmetrical along the vertical axis. In real-world images, only parts of the image

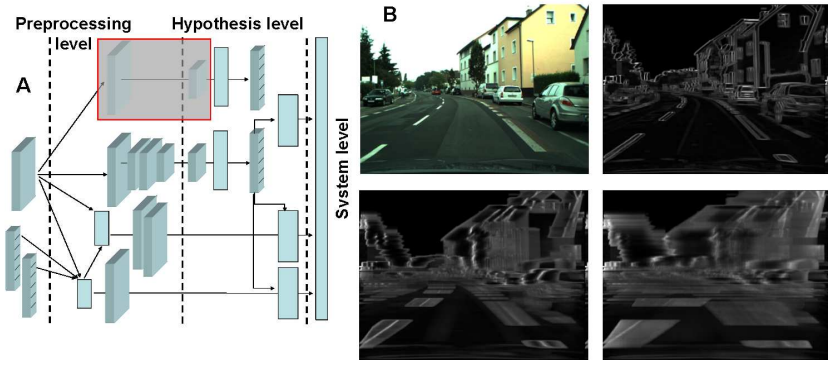

Figure 6: Performance example of symmetry-based object detection. **A)** Embedding into processing system. **B)** Input image and generated multi-scale confidence map. A total of $K = 8$ scales is used for symmetry, corresponding to filters of pixel height $h = 3$ and half-width $\frac{w}{2} = 5, 9, 13, 18, 25, 35, 49, 69$. Shown are confidence maps corresponding to filter widths 5,25 and 69.

are symmetrical, so we define a local measure $L_{i,l}(\vec{x})$ for a window around a pixel at $\vec{x}$ at spatial scale determined by i, l. By considering a fixed y-window defined by $l = 1$, $L_{i,l}(\vec{x})$ effectively becomes $L_i(\vec{x}), i = 0, \ldots, K-1$ where K denotes the number of spatial scales:

$$L_i^2(\vec{x}) = \sum_{k\in[-i,\ldots,i]} \sum_{j\in[-1,\ldots,1]} \parallel I(\vec{x} - k\vec{e}_x - j\vec{e}_y) - I(\vec{x} + k\vec{e}_x - j\vec{e}_y) \parallel^2 \quad . \tag{2}$$

Small values of this measure denote symmetrical regions, large values denote asymmetrical regions. However, when looking closer at Eqn. 2 it becomes clear that large homogeneous parts of the image (e.g. the sky) always result in small values for L_i in all windows. As homogeneous regions usually do not contain objects, we want to inhibit those regions using the variance σ_i^2 in the local patches as a structure measure

$$\sigma_i^2(\vec{x}) = \frac{\sum_{k\in[-i,\ldots,i]} \sum_{j\in[-1,\ldots,1]} \left[I(\vec{x} - k\vec{e}_x - j\vec{e}_y) - \mu_k(\vec{x})\right]^2}{(2i+1)(2l+1)} \tag{3}$$

with

$$\mu_i(\vec{x}) = \frac{\sum_{k\in[-i,\ldots,i]} \sum_{j\in[-1,\ldots,1]} I(\vec{x} - k\vec{e}_x - j\vec{e}_y)}{(2i+1)(2l+1)} \quad . \tag{4}$$

Given a window size i defining the spatial scale of symmetry calculation, a confidence map $S_i(\vec{x})$ is calculated from Eqn. 2 and Eqn. 3 as follows:

$$S_i(\vec{x}) = e^{-\frac{1}{2\sigma^2} L_i^2(\vec{x})} \cdot \sigma_i^2(\vec{x}) \; . \tag{5}$$

Figure 6 shows an example of the multiscale confidence map for a given input image.

7 Distance and elevation computation

We employ a parametric plane estimation technique[8] to determine the position of the ground plane in car-centered coordinates, and then in a subsequent step the distance of all valid stereo pixels to that plane. This results in the *retinotopic elevation representation* $F^{\text{elevation}}(\vec{x}) \equiv F^1(\vec{x})$.

Given such a retinotopic elevation representation and an object hypothesis with center $\vec{x}_c$ and width and height w, h, we can, at the hypothesis level, compute a population-coded system-level quantity in several steps. First of all, a sub-region of the hypothesis is defined by shrinking the original hypothesis to 70% of its width and height. Then, a normalized histogram $h(e)$ containing Z bins is computed, spanning elevation values e from $e_{\min} = -3m$ to $e_{\max} = 20m$. Finally, the histogram is transferred to a population-coded representation by creating a Gaussian with peak value $h(e)$ at a unique position for each histogram

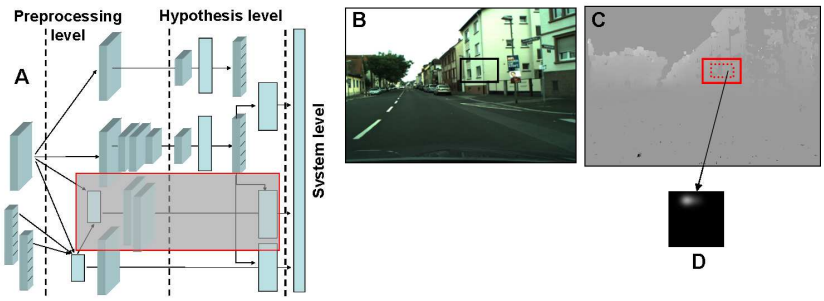

Figure 7: Examples of stereo processing for elevation calculation. **A)** embedding into processing system **B)** video image with object hypothesis **C)** dense elevation map $F^1(\vec{x})$. **D)** population code $f^1(\vec{p})$ resulting from this measurement. Depending on the distribution of elevation values, such population codes will be more or less strongly multimodal, thus reflecting the uncertainty of the associated measurement.

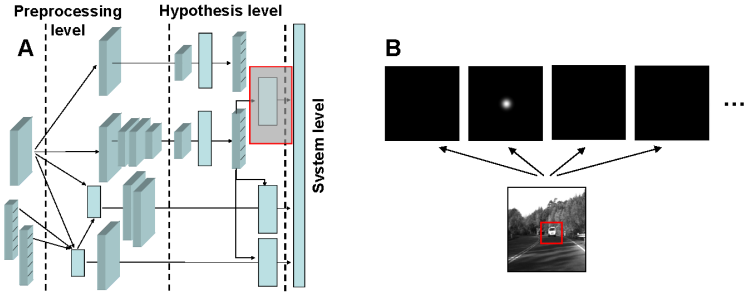

Figure 8: Population encoding of hypothesis position and size. **A)** Embedding into processing system. **B)** Performance example of size-dependent position encoding into a pyramid of population codes. Given a hypothesis as shown in the video image, a Gaussian of fixed variance is created at the pyramid level corresponding to the hypothesis' size at the center position of the hypothesis. Larger hypotheses would be represented by population codes at different pyramid levels.

bin e. The whole transformation sequence is given by

$$\begin{aligned}
\text{ROI} &= \{\vec{x} | \vec{x} \in [\vec{x}_c \pm \frac{0.7w}{2}\vec{e}_x \pm \frac{0.7h}{2}\vec{e}_y]\} \\
\Delta &= \frac{e_{\max} - e_{\min}}{Z} \\
\tilde{h}(e) &= \sum_{\vec{x} \in \text{ROI}} \chi_e(F^2(\vec{x}) \quad \text{where } \chi_e(x \in [e, e+\Delta]) = 1,\ \chi_e(x \notin [e, e+\Delta]) = 0 \\
h(e) &= \frac{h(e)}{\sum_e \tilde{h}(e)},\ e = i\Delta, i \in [0, Z-1] \\
f^1(\vec{p}) &= \sum_e \exp - \frac{(\vec{x} - (\frac{e-e_{\min}}{e_{\max}} n, 10)^T)^2}{2\sigma^2}
\end{aligned} \tag{6}$$

Please see Fig. 7 for an example of the transfer of elevation measurements to population codes. An identical procedure is used to compute the population-coded distance representation $f^2(\vec{p})$ using the retinotopic distance map $F^2(\vec{x})$ and an object hypothesis with center $\vec{x}_c$ and width and height w, h.

8 Position- and size-related analysis

At hypothesis level (see Fig. 2), we encode the pixel coordinates of the hypothesis center, $\vec{x}_c$ and its pixel size s in a multiscale population code $f_i^3(\vec{p}),\ i = 0, \ldots, K$ where K is equal to the number of pyramid levels produced by the appearance-based classifier (see Sec. 3). Hypothesis size s is calculated as a function of

Figure 9: Example of single-image performance evaluation. A) Hypothesis matching an annotation (true positive case) B) hypothesis not matching any annotation in the current image (false positive case) C) annotation matched by one or more hypotheses D) annotation not matched by any hypothesis (false negative case) E) annotation that is not considered due to size constraints (voluntary annotation), see text. Such annotations do not constitute a false negative case when matched by a hypothesis, but neither a true positive case otherwise. Note that non-cars are not annotated because this would require knowledge about the match measure and the kinds of hypotheses produced by our system, which cannot/should not be known at annotation time.

hypothesis width w and height h and quantized to K bins. Subsequently, the population-coded representation of hypothesis position, $\mathrm{pos}(\vec{x})$ is calculated as depicted in Fig. 3 and inserted into the resulting multiscale population code at index s. In detail, this reads:

$$\begin{aligned} s &= \mathrm{quantize}_K \sqrt{w * h} \\ \mathrm{pos}(\vec{x}) &= \exp - \frac{(\vec{x} - \vec{x}_c)^2}{2\sigma^2} \\ f_i^3(\vec{x}) &= \begin{cases} \mathrm{pos}(\vec{x}) & i = s \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{7}$$

A visualization of this process is given in Fig. 8.

9 Evaluation measures for object detection

Annotations for evaluation For study presented in [1], only annotations containing vehicles are considered. As visualized in Fig. 10, for an image $i \in [0, I-1]$ in a video sequence, a single annotation a_k, $k = 0, \ldots, A_i - 1$ is comprised of a rectangular area a_j^{ROI} described as a set of points, an identity $a_k^{\mathrm{id}} \in \{0, 1\}$ and an occlusion value $a_k^{\mathrm{occ}} \in [0, 1]$. The latter figure indicates the

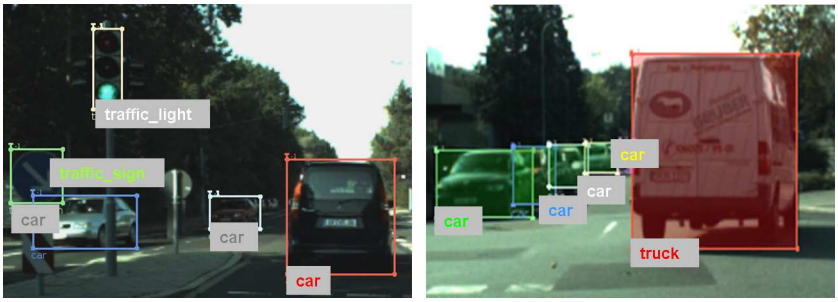

Figure 10: Examples of recorded streams and annotated information. Annotated objects are mainly vehicles and static traffic-relevant scene elements such as traffic signs, signal boards and the free/drivable road area. Each annotation consists of a polygonal area (always rectangular in the case of vehicles), an identity and an occlusion value (not shown). In order to reduce the annotation effort, only every tenth image in a video sequence was annotated. Care was taken to guarantee the annotation of *all* vehicles where annotations were performed. As can be seen from the images, we use what we term *semantic annotations*, which means that is has been tried to mark the whole area containing an object even if it is partially occluded.

percentage of the object that can not be seen. Both the rectangular area and the occlusion value can be used to filter annotations as described in [1].

Evaluation measures The evaluations in [1, 9] are always based on single-image measures averaged over the whole of a video sequence. A fundamental issue is the concept of *match*: for image i in a video sequence, we say that a hypothesis $h_j,\ j = 0,\ldots,H_i - 1 <= H - 1$, defined by a center pixel a_k^{c} and a width/height $a_k^{\mathrm{w/h}}$, *matches* an annotation $a_k,\ k = 0,\ldots,A_i - 1$ if the center pixel is contained in the area defined by the annotation. We consider a hypothesis to be *false positive* if it does not match any annotation in the current image. Conversely, we consider the absence of a hypothesis a *false negative* if there is an annotation which is not matched by any hypothesis. For each image $i \in [0, I-1]$, we compute the number of *false positive* hypotheses and *false negative* annotations (see Fig. 9), denoting them by fpi_i, fni_i. Furthermore denoting the total number of annotations in the video sequence by N_{p}, we obtain two quality measures denoted *false positives per image* (fppi) and *recall*.

$$\text{match}(h_j, a_k) = \begin{cases} 1 & h_j^{\text{c}} \in a_k^{\text{ROI}} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{fp}(h_j) = 1.0 - \mathcal{H}(\sum_k \text{match}(h_j, a_k))$$

$$\text{fn}(a_k) = 1.0 - \mathcal{H}(\sum_j \text{match}(h_j, a_k))$$

where $\mathcal{H}$ denotes the Heaviside function which is defined by $\mathcal{H}(x \leq 0) \equiv 0,\ \mathcal{H}(x > 0) \equiv 1$. Finally, we have

$$\text{fpi}_i = \sum_j \text{fp}(h_j), \quad \text{fni}_i = \sum_k \text{fn}(a_k)$$

$$\text{fppi} = \frac{1}{I} \sum_i \text{fpi}_i, \quad \text{recall} = 1 - \frac{1}{\sum_i A_i} \sum_i \text{fni}_i.$$

The reason we compute the measure fppi, the averaged number of false positives per image, instead of the more conventional false positive rate is which is defined as the total number of false positive detections divided by the total number of non-car objects, is simple: in contrast to the total number of true positives (e.g., annotated cars), the total number of non-car objects in a video stream is not a reliable or fixed quantity; it depends on many factors such as the used match measure, the used object detection mechanism a.s.o. In the interest of presenting an objective evaluation, we therefore use the fppi measure which does not require the total number of non-car objects for its definition (see also [10] for a justification of the fppi measure).

For a fixed parameterization of the system, the performance is given by a point in a recall/fppi-diagram. By plotting these two quantities against each other for variations of the detection thresholds θ_{class} or θ_{symm}, we obtain plots similar in interpretation to receiver-operator-characteristics (ROCs). Such ROC-like plots will be used for visualizing object detection performance in [1].

For performance evaluation in [1, 9], we only consider annotations whose associated occlusion value is less than 80%.

10 Implementation details

The constituent parts of the presented system are implemented mainly in C; some non-time critical parts are implemented in Python. We use the component-based RTBOS middleware [11] to interface the various system parts in an uncomplicated, visual manner, while making use of the parallelization abilities of present-day computer hardware. The system is running on a single high-end workstation with multiple cores running Linux at a frame rate of 0.3 fps.

Table 2: Global parameters used for experiments

Parameter	explanation	value
H	max. Nr of Hypotheses	40 or 10 (Sec. 9)
N,M	image/feature map size	400,300 [1]
n,m	population code size	64,64 [1]
σ	peak width for pop. encoding	4 (Sec. 1.1)
θ_{class}	classifier selection threshold	0.0 for eval.[1]
θ_{symm}	symmetry selection threshold	0.0 for eval.[1]
θ_{MLP}	symmetry selection threshold	0.0 for eval. [1]
ϵ	system-level learning rate	0.00002 or 0.0001 [1]
e_{min}	min. elevation value	-3.0m (Sec. 7
e_{max}	max. elevation value	20m (Sec. 7)
d_{min}	max. dist-2-free-area value	70 (see Sec. 5)
Z	hist. bins for pop.encoding	100 (Sec. 1.1)
K	nr of pyramid scales	16 (classifier, Sec. 3) or 8 (symmetry, Sec. 6)
n_{blocking}	blocking interval	30s [1]

11 Comparison of the HRI RoadTraffic dataset to other benchmark datasets

There exists, by now, a number of annotated vehicle datasets which are often used for benchmarking the performance of object detection systems in a comparable way. For traffic related areas of interest, the most notable datasets are the CBCL StreetScenes Database (see, e.g., [12]) and the UIUC Image Database for Car Detection[13]. Another popular benchmark are the datasets of the yearly PASCAL object detection challenges, which also contain traffic objects but are not restricted to them.

In contrast to the HRI RoadTraffic dataset described here, these datasets consist of monocular still images instead of continuous stereo video. Apart from the usefulness of stereo information, we believe that the possibilities of processing continuous video streams are manifold, since, object detection could be supported by, e.g., tracking algorithms. In addition, the number of annotated images is significantly larger in the HRI RoadTrafic dataset; furthermore, our annotations include information about object occlusion which not contained in the other described datasets. Finally, the HRI road traffic dataset contains (for each image) additional information, such as the results of the free-area computation described in Sec. 5, as well as speed/yaw rate information.

Going beyond the area of vehicle detection, there exists an important dataset containing pedestrian annotations which is comparable to HRI RoadTraffic: the Daimler pedestrian detection benchmark dataset[14]. It contains a very large number of cropped pedestrian images for classifier training, and a fully annotated sequence of $\tilde{2}7$ minutes of driving for evaluation purposes. The image resolution is 640x480 recorded with a monocular grayscale camera. Additional data such as free-area and speed/yaw rate information are not included; however

the total number of annotated objects is larger than in the HRI RoadTraffic dataset.

12 Technical description of the HRI RoadTraffic dataset

The HRI RoadTraffic dataset comes in 5 directories, one for each stream. In each directory, there are the following entires:

leftImages This subdirectory contains numbered images from the left camera. The images are actually grayscale but in Bayer format. They can be demosaiced with any standard image processing toolbox to obtain RGB color images.

rightImages This subdirectory contains numbered images from the right camera. They are identical in nature to the left camera images described in the previous paragraph.

annotations This subdirectory contains XML files that can be read with the Matlab toolbox[1] provided by the LabelMe project[15]. This toolbox also allows to access the occlusion value defined for each annotation. XML files can be linked to camera images via their numbers.

freeArea This subdirectory contains numbered binary PNG images (either 0 or 255) indicating the free area computed from the corresponding left camera image. Each binary image can be linked to images via its number.

proprioception This subdirectory contains a single text file, each line of which contains a timestep, current yaw rate (in degrees/s), current steering wheel angle (in degrees) and current speed in km/h. The correspondence with camera images has to be established via the timestep information.

parameters Thus subdirectory contains a single text file with the camera calibration parameters used for all recordings. They are stored human-readable format using the common conventions for camera parameters.

timesteps???.txt A text file containing one row per image. The first column gives the image index, the second index gives the timestep of the image having that index.

Files in the subdirectories *leftImages, rightImages, annotations* and *freeArea* have a unique consecutive number which is used to link images to each other through different subdirectories. Indices can be mapped to physical timesteps by evaluation if the timesteps text file in each directory. Proprioceptive data

[1] labelme.csail.mit.edu/LabelMeToolbox/index.html

were recorded independently at different frequency. They can be matched to image data with the knowledge that 1 second corresponds to 32000 timesteps.

References

[1] A Gepperth, S Hasler, S Rebhan, and J Fritsch. Biased competition in visual processing hierarchies: a learning approach using multiple cues. *Cognitive Computation*, 2010.

[2] A Pouget, P Dayan, and RS Zemel. Inference and computation with population codes. *Annu Rev Neurosci*, 26:381–410, 2003.

[3] W. Erlhagen, A. Bastian, D. Jancke, A. Riehle, and G. Schner. The distribution of neuronal population activation (dpa) as a tool to study interaction and integration in cortical representations. *J Neurosci Methods*, 94(1):53–66, Dec 1999.

[4] RS Zemel, P Dayan, and A Pouget. Probabilistic interpretation of population codes. *Neural Comput*, 10(2):403–430, Feb 1998.

[5] CM Bishop. *Pattern recognition and machine learning*. Springer-Verlag, New York, 2006.

[6] DC Knill and A Pouget. The bayesian brain: the role of uncertainty in neural coding and computation. *Trends Neurosci*, 27(12), 2004.

[7] T Michalke, R Kastner, J Fritsch, and C Goerick. A generic temporal integration approach for enhancing feature-based road-detection systems. In *Intelligent Transportation Systems Conference*, Peking, 2008. IEEE Press.

[8] Martin Heracles, Bram Bolder, and Christian Goerick. Fast detection of arbitrary planar surfaces from unreliable 3D data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[9] Jens Schmuedderich, Nils Einecke, Stephan Hasler, Alexander Gepperth, Bram Bolder, Robert Kastner, Mathias Franzius, Sven Rebhan, Benjamin Dittes, Heiko Wersing, Julian Eggert, Jannik Fritsch, and Christian Goerick. System approach for multi-purpose representations of traffic scene elements. In *13th International IEEE Annual Conference on Intelligent Transportation Systems*, 2010.

[10] P Dollar, C Wojek, B Schiele, and P Perona. Pedestrian detection: A benchmark. In *CVPR*, June 2009.

[11] Antonello Ceravola, Marcus Stein, and Christian Goerick. Researching and developing a real-time infrastructure for intelligent systems. *Robotics and Autonomous Systems*, 2007.

[12] L. Wolf and S.M. Bileschi. A critical view of context. *IJCV*, 69(2):251–261, August 2006.

[13] Dan Roth Shivani Agarwal, Aatif Awan. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11), 2004.

[14] M Enzweiler and DM Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2008.

[15] BC Russell, A Torralba, KP Murphy, and WT Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3), 2008.