Biologically Inspired Online Learning of Visual Autonomous Driving

Kristoffer Öfjäll and Michael Felsberg

Biologically Inspired Online Learning of Visual Autonomous Driving

Kristoffer Öfjäll
kristoffer.ofjall@liu.se
Michael Felsberg
michael.felsberg@liu.se

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
Linköping, Sweden

Abstract

While autonomously driving systems accumulate more and more sensors as well as highly specialized visual features and engineered solutions, the human visual system provides evidence that visual input and simple low level image features are sufficient for successful driving. In this paper we propose extensions (non-linear update and coherence weighting) to one of the simplest biologically inspired learning schemes (Hebbian learning). We show that this is sufficient for online learning of visual autonomous driving, where the system learns to directly map low level image features to control signals. After the initial training period, the system seamlessly continues autonomously. This extended Hebbian algorithm, qHebb, has constant bounds on time and memory complexity for training and evaluation, independent of the number of training samples presented to the system. Further, the proposed algorithm compares favorably to state of the art engineered batch learning algorithms.

1 Introduction and Related Work

It has been observed that humans, although other senses may be of assistance, can successfully drive a car solely based on visual input. However, most approaches to autonomous vehicles employ a wide range of different sensors [8, 21, 22]. One exception was ALVINN, autonomous driving using an artificial neural network. However, great care had to be taken to maintain a balanced training set in terms of number of left turns, right turns and straight ahead examples [1]. In this work we aim to create a system that is able to learn to successfully navigate along a track using only visual input. No prior information regarding track features or layout is provided and training is online and based on demonstration: manually piloting the vehicle around the track.

One early biologically inspired learning algorithm is Hebbian learning [14]. The ideas of Hebbian learning originate from one particular idea of learning in biological systems presented by psychologist Donald Hebb. A simplified description of the general idea is that learning can come about by adjusting the strengths of synapses depending on the simultaneousness of activations of the neurons before and after the synapse in question [15]. Today technical systems employ more sophisticated learning algorithms such as Random Forest Regression [3] or Support Vector Regression [4], as those generally perform superior to the simpler algorithms such as Hebbian learning. However, we will show that Hebbian learning can achieve even superior results with some extensions. For improving the performance of the Hebbian approach, two key components have been identified: introducing a *non-linear* update step and weighting by *coherence* information. In the present work we propose such extensions and evaluate the performance of the algorithm for learning visual autonomous driving according to [5]. Since the performance in the driving task is not easily comparable to competing methods, we also evaluate the proposed learning algorithm on a visual pose estimation data-set from ECCV 2012.

For visual autonomous driving, the system must learn to associate the presence of certain input features with the appropriate action. What makes this a difficult task is that many features are not at all related to the appropriate action, while others are active in different cases where different actions are required. At the extreme, the set of active visual features in two different situations are exactly the same but the actions are different, such as just before turning left or right at an intersection.

Concerning learning the relation between input and output features, two properties are of interest, *coherence* and *evidence*. Any input feature often activated simultaneously with a certain output value will have high *evidence*. Traditional Hebbian learning is based on evidence. However, if a certain input feature is activated in combination with many different output values, the input-output relation is of low *coherence*. Any input feature activated in combination with one and only one output value will have a high *coherence*, independent of how often the specific input feature is activated (evidence). By also using the coherence information, more specific output predictions can be made.

In the ideal case, the full joint distribution of input features and output values is learned. However, in the general case with continuous inputs and outputs, the joint distribution can not be represented by a finite set of parameters and an approximate representation is needed. Limiting to second order approximations (Gaussian assumption or linear relations) is too restrictive in this case due to outliers and multi-modal distributions. Using mixtures of Gaussians such as Realtime Overlapping Gaussian Expert Regression (ROGER [12]) is computationally too demanding as it slows down with increasing number of training samples [7].

For any distribution with limited support, histograms provide an approximate representation where computational effort is independent of the shape of the represented distribution and where the number of bins directly controls the trade-off between computational demands and representation accuracy [27, 28]. However, the channel representation [10] (also known as population coding [25] in computational neuroscience) provides higher representation accuracy given the same number of channels/bins compared to histograms [6].

Our main contribution is *qHebb*, an extended Hebbian learning algorithm with a novel non-linear update rule (Sect. 3) and a novel coherence measure (Sect. 4), both based on channel representations. A brief introduction to the channel representation is provided in Sect. 2, which also introduces the notation used in this paper. The vision based autonomous system is presented in Sect. 5, and a comparison to state of the art methods is provided in Sect. 6, showing that qHebb outperforms state of the art methods.

2 Channel Representations and Associative Learning

This section provides a brief introduction to channel representations and associative learning at an intuitive level, since these methods will be required for our proposed contributions in Sect. 3, 4. Readers unfamiliar with these methods are referred to more comprehensive descriptions in the literature [6, 7, 10] for details.

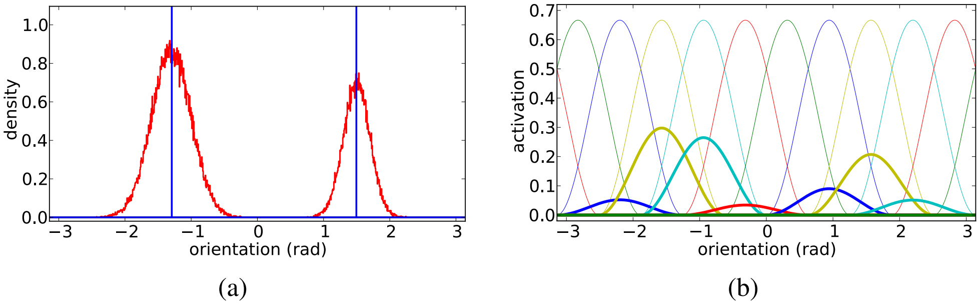

Figure 1: Illustration of a channel representation for orientation data. (a) the orientation data (density in red) stems from two modes with different variance and probability mass. The blue lines indicate the mode estimates as extracted from the channel representation. (b) the channel representation of the distribution of orientation data. The thin plots indicate the kernel functions (the channels) and the bold lines illustrate the corresponding channel coefficients as weighted kernel functions.

Channel representations have been proposed in 2000 [10]. The idea is to encode image features (e.g. intensity, orientation) and control signals (e.g. pose angles, steering angles) in a vector of soft quantization levels, the channels. An example is given in Fig. 1, where orientation values are encoded.

Readers familiar with population codes [25, 30], soft/averaged histograms [27], or Parzen estimators will find similarities. The major difference is that channel representations are very computationally efficient to encode (because of the regular spacing of the channels) and decode (by applying frame theory, see Sect. 2.2). This computational efficiency allows for computing channel representations at each image pixel or for small image neighborhoods, as used in channel smoothing [6], variants of bilateral filtering [20] and tracking using distribution fields [28].

2.1 Channel Encoding

The kernel function, $b(\cdot)$, is required to be non-negative, smooth and to have compact support. In this paper, $\cos^2$ kernels with bandwidth parameter h are used:

$$b(\xi) = \frac{2}{3}\cos^2(\pi\xi/h) \qquad \text{for } |\xi| < h/2 \quad \text{and 0 otherwise.} \tag{1}$$

The components of the *channel vector* $\mathbf{x} = (x_1, x_2 \ldots, x_K)^T$ are obtained by shifting the kernel function K times with increments $h/3$ (three simultaneously active channels). Encoding a value ξ gives the *channel coefficients* $x_k = b(\xi - \beta - kh/3)$. The number of channels, K, and the offset, β, are determined by the channel width, h, and the range of ξ, the variable to be encoded.

Multidimensional data can be encoded in two ways [11]: If the separate dimensions are independent, all dimensions are encoded separately and the resulting vectors are concatenated. If the dimensions are mutually dependent, the outer product of the channel vectors is built, similar to a joint histogram. The drawback of the joint channel representation is the exponential growth of number of coefficients with the number of dimensions. This is mitigated by sparse data structures, as the number of non-zero elements has a linear upper bound in the number of samples.

2.2 Robust Decoding

Probabilistically motivated methods require distribution representations and may work directly on channel representations and mappings or distances (divergences) thereof. If applied to estimation or control problems, the mode ξ of the represented distribution needs to be estimated robustly from a channel vector $\mathbf{x}$. Independently whether MAP, ML, or some other criterion is used, the position of a local maximum of the underlying distribution is required. Since $\cos^2$-channels establish a tight frame, the local maximum is obtained using three orthogonal vectors [9] $\mathbf{w}_1 \propto (\dots,0,2,-1,-1,0,\dots)^T, \mathbf{w}_2 \propto (\dots,0,0,1,-1,0,\dots)^T, \mathbf{w}_3 \propto (\dots,0,1,1,1,0,\dots)^T$ and

$$r_1 \exp(i2\pi\hat{\xi}/h) = (\mathbf{w}_1 + i\mathbf{w}_2)^T\mathbf{x} \qquad r_2 = \mathbf{w}_3^T\mathbf{x} \tag{2}$$

where $i = \sqrt{-1}$ denotes the imaginary unit, $\hat{\xi}$ is the estimate (modulo an integer shift determined by the position of the non-zero elements in $\mathbf{w}_k$, the *decoding window*), and r_1, r_2 are two confidence measures [9].

2.3 Associative Learning

The general problem considered in *associative* learning using channel representations [17] is to find the mapping from some N-dimensional input space of vectors with components $\xi_j^{(n)}$, $n \in [1,N]$, to some 1D output space (we only consider 1D output in this paper) with corresponding values η_j, where the input and output vectors are encoded in channel vectors $\mathbf{x}_j$ and $\mathbf{y}_j$, respectively, as described in Sect. 2.1. The mapping is taken to be linear in the channel domain such that a *linkage matrix* $\mathbf{C}$ is found, mapping the channel representations of inputs $\mathbf{x}_j$ to the channel representations of the corresponding outputs $\mathbf{y}_j$,

$$\mathbf{y}_j = \mathbf{C}\mathbf{x}_j \tag{3}$$

from which the mode estimates $\hat{\eta}_j$ are extracted (Sect. 2.2). Previous approaches can be summarized as minimizing $E(\mathbf{C}) = \sum_{j=1}^{J} \psi(\mathbf{y}_j, \mathbf{C}\mathbf{x}_j)$ given channel encoded training data pairs $(\mathbf{x}_1,\mathbf{y}_1),\dots,(\mathbf{x}_J,\mathbf{y}_J)$. The function $\psi(\cdot)$ is a suitable distance function, e.g. an L_q metric [19] or an α-divergence [7], and the minimization might be conditioned to be non-negative [19]. Note that although the mapping is linear in the channel domain, it can still represent both non-linear and multi-modal mappings in the domains of ξ and η [7].

Although not explicitly expressed in the literature, a *Hebbian associative* learning approach would be $\mathbf{C} = \sum_{j=1}^{J} \mathbf{D}_j$ where $\mathbf{D}_j = \mathbf{y}_j\mathbf{x}_j^{\mathrm{T}}$, outer products of channel encoded training data pairs $(\mathbf{x}_j, \mathbf{y}_j)$, such that active channels in $\mathbf{x}$ would activate channels in $\mathbf{y}$ in proportion to how often and much each pair of channels have been simultaneously activated in the training data. This learning scheme can trivially be converted to an online learning method, $\mathbf{C}_j = \mathbf{C}_{j-1} + \mathbf{D}_j$. As with conventional Hebbian learning, there are issues such as elements in $\mathbf{C}$ growing unbounded and that biased training data would generate a biased mapping [14].

3 The Proposed qHebb Update Equation

In this section we consider a learning system where input and output are one dimensional entities and discuss its properties. The generalization to multi-dimensional inputs (and outputs) is straightforward using concatenated channel vectors or outer products of channel vectors.

Each element (using indexing notation $[\cdot]_{kl}$) in the linkage matrix $\mathbf{C}$ represents a possible link from certain input values to certain output values. If a certain element $[\mathbf{C}]_{kl}$ is large, inputs activating the lth input channel will activate the kth channel of the output representation. Increasing the value of this element $[\mathbf{C}]_{kl}$ strengthens the connection (learning) while decreasing its value weakens the connection (forgetting).

Introducing a forgetting factor $0 < (1-\gamma) < 1$, resulting in the new update rule $\mathbf{C}_j = (1-\gamma)\mathbf{C}_{j-1} + \gamma\mathbf{D}_j$ mitigates the unlimited element growth, however learning and forgetting are coupled. Increasing the learning rate γ puts more weight on the new training sample (the outer product of the output and input representations). This accelerates the learning when the corresponding element is activated by the current training data but also leads to faster forgetting when the current element is not activated.

For conventional Hebbian learning acting directly on data vectors, a non-linear combination of values would offset the learned mapping. However for associative Hebbian learning (acting on channel encoded data) a monotonic combination of values can be used without disturbing distribution maxima. We suggest a power combination parameter $q \geq 1$, and the *qHebb learning equation*

$$\mathbf{C}_j = \left((1-\gamma)\mathbf{C}_{j-1}^q + \gamma\mathbf{D}_j^q\right)^{\frac{1}{q}} \tag{4}$$

where matrix exponentiation is to be taken element-wise. All elements of $\mathbf{C}_j$ are bounded from above as (element-wise) $\mathbf{C}_j \leq \max(\mathbf{C}_{j-1}, \mathbf{D}_j)$ follows from (4) and the elements of $\mathbf{D}_j$ are bounded by the squared maximum channel activation, $\max_\xi b(\xi)^2$. All elements of $\mathbf{C}_j$ are non-negative (bounded from below) via non-negative channel basis functions. Note that linear update is a special case ($q = 1$).

Increasing q shifts the weight towards the larger of each two corresponding elements in $\mathbf{C}$ and $\mathbf{D}$. If $[\mathbf{D}]_{kl} > [\mathbf{C}]_{kl}$, i.e. the current training sample is dominating, increased q leads to faster learning. On the other hand, if $[\mathbf{D}]_{kl} < [\mathbf{C}]_{kl}$, increased q leads to slower forgetting. Using both q and γ, learning rate and forgetting rate can be set independently. Increased q also decreases the effect of biased training sets on the learned mapping. Letting $q \to \infty$, (4) becomes $\mathbf{C}_j = \max\left(\mathbf{C}_{j-1}, \mathbf{D}_j\right)$, i.e. learning is immediate and the system never forgets.

4 Coherence and Evidence of Channel Representations

For combinations of multiple channel encoded measurements of an entity, two properties characterizing the combined channel vector are of interest. Here we refer to them as *evidence* and *coherence*.

Evidence is what is referred to as r_2 in Sect. 2.2, the L_1 norm of the decoding window. When combining channel vectors by addition, r_2 is proportional to the number of samples accumulated within the current decoding window. Consider an input feature that is always activated, evidence related to this feature will always be maximal, however activation of this feature will not provide any information regarding the value of the output. Based on these observations, we propose a different measure, *coherence*.

We will define *coherence* as a measure of the consistency of samples resulting in a mode. See Fig. 1, where the right mode has higher coherence than the left mode. We propose a weighted use of the learned linkage matrix, where each input element is weighted by the coherence of its contribution to the output.

4.1 Definition of Coherence

Several norms and properties of channel encoded entities have been proposed and evaluated in the literature [9, 18], however, coherence has not previously been suggested, although it has been suggested for the structure tensor [2]. For notational and conceptual clarity and without loss of generality, basis functions are assumed to be centered at integer positions in this section ($h = 3$).

As shown in [9] and indicated in Sect. 2.2, decoding of $\cos^2$ channel vectors (determining estimates of ξ) are carried out according to

$$\begin{pmatrix} r_1 \cos(\frac{2\pi}{3}(\xi - l)) \\ r_1 \sin(\frac{2\pi}{3}(\xi - l)) \\ r_2 \end{pmatrix} = \mathbf{W}\mathbf{x}_l = \begin{pmatrix} 2 & -1 & -1 \\ 0 & \sqrt{3} & -\sqrt{3} \\ 1 & 1 & 1 \end{pmatrix} \mathbf{x}_l \tag{5}$$

where l selects the decoding window and $\mathbf{x}_l$ is the corresponding three elements from the channel vector $\mathbf{x}$ to be decoded. It follows that for $r_1 = 0$ (all elements in $\mathbf{x}_l$ equal), decoding is ambiguous, and for larger values of r_1, the estimate of ξ is less dependent on small perturbations of the channel coefficients, however the absolute value of r_1 varies with the scaling of the channel coefficients.

A function that generalizes [2] and that we propose as coherence measure is

$$\mathrm{coh}(\mathbf{x}_l) = \frac{r_1^2}{r_2^2} = \frac{1}{\mathbf{1}^{\mathrm{T}}\mathbf{x}_l\mathbf{x}_l^{\mathrm{T}}\mathbf{1}} \mathbf{x}_l^{\mathrm{T}} \begin{pmatrix} 4 & -2 & -2 \\ -2 & 4 & -2 \\ -2 & -2 & 4 \end{pmatrix} \mathbf{x}_l \tag{6}$$

with $\mathbf{1} = (1\ 1\ 1)^{\mathrm{T}}$ and where the last equality follows from (5). It can easily be verified that $\mathrm{coh}(\mathbf{x}_l) = 0$ when decoding is ambiguous and $\mathrm{coh}(\mathbf{x}_l) = 1$ for a single encoded value or for a combination of encodings of the same value. Further, $\mathrm{coh}(\alpha\mathbf{x}_l)$ is independent of scale ($\alpha > 0$) and, $\mathrm{coh}(\mathbf{x}_l)$ decreases monotonically with a wider distribution of the encoded values within the decoding window. These results build upon properties of the $\cos^2$ kernel, namely that for any value ξ within the representable range of a channel representation, the L_1 and L_2 norms of the corresponding channel vector are constant [9] (and specifically independent of the position of ξ with respect to channel centers). These properties do not hold for Gaussian or B-spline kernels. Coherence as defined is a property related to a specific decoding window, and we further define the coherence of a full channel vector as the coherence of the strongest mode. The strongest mode is defined as the decoding window with largest evidence.

4.2 Application to Associative Mappings

For a link matrix $\mathbf{C}$, the coherence of each column is indicative of how precise statements regarding the output is made by the activation of the corresponding input channel. This can be used to reduce the influence of input channels producing vague (wide) output predictions. We thus define the coherence of a link matrix, $\mathrm{coh}(\mathbf{C})$, to be the vector of coherences of each column of $\mathbf{C}$. Further we propose a weighted application of the link matrix, cf. (3), with a scalar parameter $\kappa \geq 0$, allowing for adjustment of the coherence weight influence

$$\mathbf{y} = \mathbf{C}\,\mathrm{diag}\left(\mathrm{coh}(\mathbf{C}) + \kappa\right)\mathbf{x} \qquad , \tag{7}$$

where $\mathrm{diag}(\mathbf{v})$ is the diagonal matrix with elements $\mathbf{v}$. The algorithm is summarized in Fig. 2.

Learning: (given training data pair (ξ,η))	Prediction: (given input data ξ)
$\mathbf{x} \leftarrow \mathrm{encode}(\xi)$	$\mathbf{x} \leftarrow \mathrm{encode}(\xi)$
$\mathbf{y} \leftarrow \mathrm{encode}(\eta)$	$\mathbf{y} \leftarrow \mathbf{C}\,\mathrm{diag}\,(\mathrm{coh}(\mathbf{C}) + \kappa)\,\mathbf{x}$
$\mathbf{C} \leftarrow \left((1-\gamma)\mathbf{C}^q + \gamma(\mathbf{y}\mathbf{x}^{\mathrm{T}})^q\right)^{\frac{1}{q}}$	$\hat{\eta} \leftarrow \mathrm{decode}(\mathbf{y})$

Figure 2: Summary of computational steps for qHebb learning and weighted prediction.

5 Online Learning of Autonomous Driving

The proposed extensions are essential for the successful online learning of autonomous driving based solely on visual perception. The input layer of the system consists of a single gray-scale camera and a generic, holistic representation of the whole visual field, using visual Gist [24]. No information regarding what kind of track or what kind of visual features that define the track is provided in advance. The system is supposed to learn features of the track together with correct driving behavior from the visual Gist features and the manual control signals provided during the initial training phase [5]. Visual information is encoded using 4 scales, 8 orientations and 8 by 8 spatial channels resulting in an 2048-dimensional Gist feature vector. Each feature is represented using 7 channel coefficients and the channel vectors are concatenated such that input is represented by a vector with 14336 elements. The output (steering signal) is represented using 15 channels and qHebb is used to learn a linkage matrix from visual input features to the steering representation.

During training, the teacher controls the car with a standard remote control transmitter. Training examples (visual features and the teacher's corresponding steering signals) are processed online on-board the mobile robot, which is not possible with the approach in [5]. When the teacher releases the control, the learned linkage matrix is used to predict steering signals and the robot immediately continues autonomously.

The proposed coherence weighting is essential as many visual features are always activated or show activation patterns independent of the appropriate steering signal. After training, columns in the linkage matrix corresponding to those features are constant and hence activates all output (steering) channels when such an input feature is activated. Only a few features show activation patterns dependent on the appropriate steering signal. Without coherence weighting, the activation from steering dependent features drowns in the noise of activations from features independent of steering. The non-linear update (4) is required for mitigating the effects of unbalanced training data *e.g.* from differences in the numbers of left and right turns during initial training.

Learning of online visual autonomous driving using Locally Weighted Projection Regression [29] has also been evaluated [23]. The experiment was successful, however the dimensionality of the input feature space had to be reduced using a pre-calculated PCA. Also, time and memory requirements increase with more training data as more local models are generated. Fig. 3 provides a photo of the system and the different approaches are summarized in table 1. See the supplementary video.

6 Pose Estimation

Due to the stochastic nature of online driving, a second experiment is performed for quantitatively comparing the proposed algorithm to state of the art methods. This experiment is a

Method	Successful Driving	Training Data Proc. Speed
Convolutional Networks [26]	No, (offline predictions)	Days (batch)
Random Forest [5]	Yes, static track	Hours (batch)
LWPR [23]	With input projection	Initially video rate, slows down
Associative Hebbian	No	Video rate (online)
Proposed Weighted qHebb	Yes, dynamic track	Video rate (online)

Table 1: Summary of approaches for visual autonomous driving.

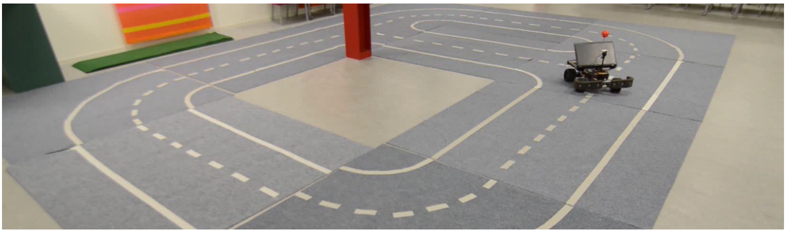

Figure 3: A single frame from the supplementary video[2]. A robot learning purely vision based autonomous driving on a track. No information regarding track layout, track features or driving behavior is provided beforehand. All is learned from demonstration: manually driving the robot. Despite the complexity of the visual input, the proposed approach is able to associate configurations of visual features to appropriate control signals in real time, such that autonomous operation follows directly after the initial training phase, without even stopping the robot.

reproduction of the ECCV 2012 experiment by [16]. A mapping from face images to head yaw angle for a subset of the Multi-PIE data-set [13] is learned. This is similar to the autonomous driving experiment in that a mapping from an image to an angle is to be learned, and in both cases, manual tuning of image feature extractors to the task is not allowed. Due to space constraints, we must refer to [16] for a comprehensive description of the experiment.

Images are cropped and information from a subset of the images are randomly removed (square blocks of size 0.1 times image width at 5 random locations). For different scenarios, either none, 20%, 40% or 80% of the images are corrupted. The local orientation in the images is calculated and the orientation images are down-sampled to 25 by 30 pixels. The set of images is divided into two equal sets for training and evaluation with 1565 each.

For qHebb associative learning (with infinite q), the orientation images are encoded using $15 \times 10 \times 10$ (1500 elements) channels in a Channel Coded Feature Map (CCFM) configuration [19] where 15×10 channels are used for spatial encoding and 10 channels are used for orientation encoding. The local orientation vector magnitude is used for weighting the orientation channels. The output space, head yaw, uses 16 channels. Coherence weighting (7), with $\kappa = 2$, is used for predicting head yaw on the evaluation set.

For Robust Regression (RR [16]) and Random Forest Regression (RFR [3]), the $25 \times 30 \times 2$ (1500 elements) orientation images are vectorized. Training on each full training set (with different ratios of corrupted versus non-corrupted images) took 12 hours for RR while qHebb took 15 seconds, both are MATLAB implementations. The results are presented in

[2] Also available at *http://users.isy.liu.se/cvl/ofjall/bmvc2014.mp4*

Evaluation after	391 training samples				1565 training samples			
Corrupted Images	0%	20%	40%	80%	0%	20%	40%	80%
RR (batch)	15.08	15.99	15.67	17.33	12.09	13.12	13.81	15.84
qHebb	12.24	12.22	12.92	13.46	10.84	10.98	11.46	11.64
qHebb, weighted	**9.61**	**9.75**	**9.45**	**9.71**	10.18	10.18	9.82	10.04
RFR (batch)	10.10	10.75	11.62	12.50	**7.26**	**7.91**	**8.40**	**9.20**
SVR (batch)	DNF	DNF	DNF	DNF	DNF	DNF	DNF	DNF
ROGER	DNF	DNF	DNF	DNF	DNF	DNF	DNF	DNF

Table 2: Mean angular error (degrees) over the full evaluation set. Best results in boldface.

Fig. 4 (mean prediction error over evaluation set as a function of number of training data used from the training set) and in table 2. For comparison, since RR and RFR only run in batch mode, the training and evaluation are repeated for subsets of the training data (using 12, 135, 391, 782, 1173 and 1565 training samples respectively). Support Vector Regression(SVR [4]) and ROGER [12] could not be used. SVR generated constant predictions independent of training set size and parameter changes. ROGER did not produce any useful results within a computation time of three days, marked as did not finish (DNF) in table 2.

For ROGER, the implementation of the original author Daniel Grollman is used. The RR implementation is the one by the original authors Dong Huang et al. For SVR, the Matlab implementation by Francesco Parrella is used. Finally, for RFR, the mex interface by Abhishek Jaiantilal to the C code by Andy Liaw et al. is used.

The results for RR using the full training set are similar to those reported by [16]. After 135 training samples, qHebb associative learning on 80% corrupted images outperform RR on corruption free images. From table 2 it is clear that qHebb without coherence weighting is less affected by image corruption than RR. Using coherence weighting, qHebb results are not degraded by image corruption at all. Performance even seem to increase slightly with higher ratios of corrupted images. We suspect the corrupted images actually may reduce the over-fitting visible as a slight performance decrease for large numbers of training data. At 391 training samples, qHebb with coherence weighting outperform all competing methods while qHebb without coherence weighting is outperformed by RFR. For more than 1000 training samples, RFR produces results slightly superior to qHebb. However, RFR is trained in batch mode and requires storing all samples which is not feasible in an online learning scenario as time and memory complexity increases with the number of training samples.

7 Conclusion

We have presented extensions to the Hebbian learning framework which enables online learning of autonomous driving with a direct mapping from low level image features to control signals, which is robust against unbalanced training data. To the best of our knowledge, this is the first demonstration of such a system with finite bounds on memory and time complexity. For the visual pose estimation experiment, the channel based Hebbian learning system is the only online learning algorithm able to manage the high-dimensional input. With the proposed extensions, the performance is increased to match the best batch (offline) learning algorithms. Concerning computational time, the batch methods required several hours for processing the 1565 training samples while the proposed method processed more

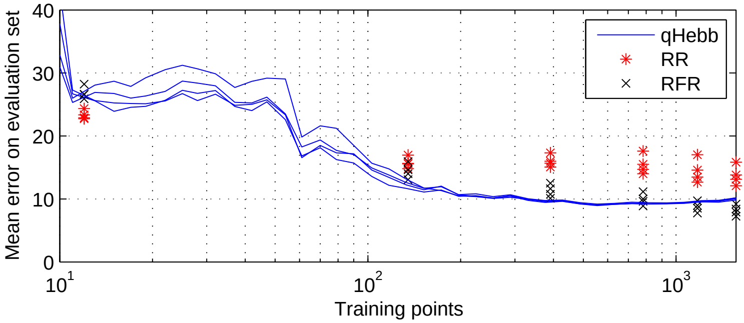

Figure 4: Robust Regression [16], Random Forest Regression [3] and associative learning using qHebb (with infinite q and coherence weighting). 0%, 20%, 40% and 80% corrupted images respectively.

than 100 training images per second (Matlab implementation). In the autonomous driving system, processing time was shorter than required for video rate processing both during training and autonomous operation.

The ability of the channel representation to represent multi-modal distributions is a fundamental advantage in many applications. For autonomous driving, e.g. at T-junctions, channel-based systems will choose either turning right or turning left, but never the mean (going straight). Prior information, either from an operator or from a higher level system, may be used to select modes. At the same time, such a prior is not strong enough to cause the car to turn left or right at a straight section of a road, see the supplementary video.

8 Acknowledgements

This work has been supported by SSF through a grant for the project CUAS, by VR through a grant for the project ETT, through the Strategic Area for ICT research CADICS, and ELLIIT.

References

[1] Parag Batavia, Dean Pomerleau, and Chuck Thorpe. Applying advanced learning algorithms to ALVINN. Technical Report CMU-RI-TR-96-31, Robotics Institute, Pittsburgh, PA, October 1996.

[2] J. Bigün and G. H. Granlund. Optimal orientation detection of linear symmetry. In *Proceedings of the IEEE First International Conference on Computer Vision*, pages 433–438, London, Great Britain, June 1987.

[3] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324.

[4] Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1997.

[5] Liam Ellis, Nicolas Pugeault, Kristoffer Öfjäll, Johan Hedborg, Richard Bowden, and Michael Felsberg. Autonomous navigation and sign detector learning. In *Robot Vision (WORV), 2013 IEEE Workshop on*, pages 144–151. IEEE, 2013.

[6] M. Felsberg, P.-E. Forssén, and H. Scharr. Channel smoothing: Efficient robust smoothing of low-level signal features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):209–222, 2006.

[7] M. Felsberg, F. Larsson, J. Wiklund, N. Wadströmer, and J. Ahlberg. Online learning of correspondences between images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[8] John Folkesson and Henrik Christensen. Outdoor exploration and slam using a compressed filter. In *ICRA*, pages 419–426, 2003.

[9] P.-E. Forssén. *Low and Medium Level Vision using Channel Representations*. PhD thesis, Linköping University, Sweden, 2004.

[10] G. H. Granlund. An Associative Perception-Action Structure Using a Localized Space Variant Information Representation. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Germany, September 2000.

[11] G. H. Granlund and A. Moe. Unrestricted recognition of 3-d objects for robotics using multi-level triplet invariants. *Artificial Intelligence Magazine*, 25(2):51–67, 2004.

[12] Daniel H Grollman. *Teaching Old Dogs New Tricks: Incremental Multimap Regression for Interactive Robot Learning from Demonstration*. PhD thesis, Brown University, 2010.

[13] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image and Vision Computing*, 28(5):807 – 813, 2010. ISSN 0262-8856. doi: 10.1016/j.imavis.2009.08.002. Best of Automatic Face and Gesture Recognition 2008.

[14] Simon S Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall, Upper Saddle River, N.J., 1999. ISBN 0-13-273350-1.

[15] Donald Hebb. *The Organization of Behavior*. Wiley, New York, 1949.

[16] Dong Huang, Ricardo Silveira Cabral, and Fernando De la Torre. Robust regression. In *European Conference on Computer Vision (ECCV)*, 2012.

[17] Björn Johansson. *Low Level Operations and Learning in Computer Vision*. PhD thesis, Linköping University, Computer Vision, The Institute of Technology, 2004.

[18] Björn Johansson, Tommy Elfving, Vladimir Kozlov, Yair Censor, Per-Erik Forssén, and Gösta Granlund. The application of an oblique-projected landweber method to a model of supervised learning. *Mathematical and Computer Modelling*, 43:892–909, April 2006.

[19] Erik Jonsson. *Channel-Coded Feature Maps for Computer Vision and Machine Learning*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, February 2008. Dissertation No. 1160, ISBN 978-91-7393-988-1.

[20] Michael Kass and Justin Solomon. Smoothed local histogram filters. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 100:1–100:10, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0210-4.

[21] T. Krajnik, P. Cristoforis, J. Faigl, H. Szuczova, M. Nitsche, M. Mejail, and L. Preucil. Image features for long-term autonomy. In *ICRA workshop on Long-Term Autonomy*, May 2013.

[22] J. Leonard, J. P. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, volume 56, chapter A Perception-Driven Autonomous Urban Vehicle. Springer Verlag, 2010.

[23] K. Öfjäll and M. Felsberg. Online learning of vision-based robot control during autonomous operation. In Yu Sun, Aman Behal, and Chi-Kit Ronald Chung, editors, *New Development in Robot Vision*. Springer, Berlin, 2014. ISBN 978-3-662-43858-9.

[24] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

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

[26] Maria Schmiterlöw. Autonomous path following using convolutional networks. Master's thesis, Linköping University, Computer Vision, The Institute of Technology, 2012.

[27] David W. Scott. Averaged shifted histograms: Effective nonparametric density estimators in several dimensions. *Annals of Statistics*, 13(3):1024–1040, 1985.

[28] Laura Sevilla-Lara and Erik Learned-Miller. Distribution fields for tracking. In *IEEE Computer Vision and Pattern Recognition*, 2012.

[29] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural Comput.*, 17:2602–2634, December 2005. ISSN 0899-7667. doi: 10.1162/089976605774320557. URL http://dl.acm.org/citation.cfm?id=1119418.1119423.

[30] R. S. Zemel, P. Dayan, and A. Pouget. Probabilistic interpretation of population codes. *Neural Computation*, 10(2):403–430, 1998.