INFORMATION–DRIVEN ORGANISATION OF VISUAL RECEPTIVE FIELDS

CHRISTOPH SALGE

Department of Computer Science, University of Hertfordshire
Hatfield, AL10 9AB, United Kingdom
c.salge@herts.ac.uk

DANIEL POLANI

Adaptive Systems Research Group, Computer Science, University of Hertfordshire
Hatfield, AL10 9AB, United Kingdom
d.polani@herts.ac.uk

By using information theory to reduce the state space of sensor arrays, such as receptive fields, for AI decision making we offer an adaptive algorithm without classical biases of hand coded approaches. This paper presents a way to build an acyclic directed graph to organise the sensor inputs of a visual receptive field. The Information Distance Metric is used to repeatedly select two sensors, which contain the most information about each other. Those are then encoded to a single variable, of equal alphabet size, with a deterministic mapping function that aims to create maximal entropy while maintaining a low information distance to the original sensors. The resulting tree determines which sensors are fused to reduce the input data while maintaining a maximum of information. The structure adapts to different environments of input images by encoding groups of preferred line structures or creating a higher resolution for areas with simulated movement. These effects are created without prior assumptions about the sensor statistics or the spatial configuration of the receptive field, and are cheap to compute since only pairwise informational comparison of sensors is used.

Keywords: Information Theory; Adaptive Sensors

1. Introduction

Informed decision making for an agent embedded into an environment necessitates the collection of information. But more information does not entail easier decision making in general, since rich sensor arrays, such as a camera or an eye, require the difficult process of structuring, reducing and categorising a huge amount of input data. Most AI algorithms have problems with a high dimensional input state space. Looking for parallels in nature, it is also questionable whether biological systems even possess the ability to transport and process such amounts of data directly [2]. So, when modelling an artificial system, this problem is often avoided through structuring and reducing the data by hand. But at this point several assumptions have to be made about what environment the agent acts in, and what information might be relevant for the choices it has to make.

Information Theory offers a new insight into this problem because it measures information without the added bias of any semantic assumptions. The information bottleneck principle in [15] implies that by limiting the available information capacity a system might be forced to extract the relevant information. Also similar results in neuroscience [12] suggest that efficient encoding of neurons can be used to account for the statistical properties of the environment they are trained with. To make those approaches more plausible for biological systems there are trends to compute those mechanisms in a localised way [6]. The agglomerative bottleneck method [13] tries to apply this idea to the bottleneck mechanism. We aim to achieve similar results by constructing a "tree-like" network structure that iteratively fuses the two nodes that contain the least novel information about each other. This aims at a step-wise reduction of capacity while minimizing the loss in overall information. The resulting tree then encodes which sensors were fused at what point and how the resulting node was created. If a certain reduction of capacity is needed, the sensor nodes can be repeatedly fused until only a given number of nodes are left. There is a certain similarity between our algorithm and a method called "hierarchical data clustering"[4] were sensor inputs are repeatedly combined into larger and larger clusters, but this method does not normally produce a new sensor that represents the cluster, but compares whole clusters in its later iterations. The hierarchical clustering method also works orthogonal to our, by aiming to cluster all the sensor input of one image or sample with similar sensor inputs. Our method aims to "cluster" the single sensor together, and then fuses them into a new sensor.

To repeatedly fuse sensor nodes we first need to identify the nodes with the most information about each other. We employ a method used in spatial reconstruction of sensor arrays, a task initially addressed by [10]. The introduction of information distance [9] made it possible to recreate the original grid-like spatial structure of an optical array sensor without any positional data, and even relate those sensors to motion sensors and actuators in a robot. In this paper, we offer a way to expand this principle to not only calculate the neighbourhood relationships, but to find informational relations of whole sensor clusters. This normally involves the expensive calculation of multi-information but our approach aims to be relatively cheap, since only pairwise comparison of sensors is used. In fact, recent work [1] suggests that pairwise comparison might be sufficient if one is only interested in maximising the multi-information of the overall system.

The structure resulting from our reconstruction has interesting adaptive abilities that depend on the environment it was trained with. If we assume that the world that matters for an agent is the world it perceives with its sensors, the structure should depend only on the samples used to train the system, and should react to different properties of that data. We observed the detection of dominant line structures and different sensor resolutions depending on the informative content of different image regions.

This paper gives an overview of our approach. First, in the section "Method" our algorithm is explained in detail. The following section "Measurements" describes how we analysed the resulting structures. "Experiments" describes the setup of different sets of data as an input and the resulting structural properties. We then go on and discuss the implications of those results.

2. Method

This section will describe in detail the algorithm used to build the tree structure for the given sensor data. The algorithm is generally applicable for sensors that contain a certain amount of information interdependence, here we focus on visual compound sensors, like and eye or a camera, where each pixel of the image is associated with a single sensor. Each sensor is a random variable S that can assume different states. Since we deal with grey-level images, those states are integers between 0 and 255. To make it easier to identify informational dependencies we decided to use a regular binning to reduce those states to integers between 1 and 15. If we expose the visual array of sensors to a series of n sample images, each sensor assumes a certain sequence of states $S = s_1, ..., s_n$. Those random variables can be used to calculate the distribution of each sensor, and also the joint distribution of two sensors. To do so it is not necessary to know the spatial arrangement of the single sensors, this information is not used in our method at all. Also the order of those sequences is irrelevant, since every permutation of a sequence still results in the same distribution. Therefore the order in which the images are perceived by our visual array is also irrelevant for our method.

2.1. *Information Theory*

Dealing with random variables allows us to apply information theory as introduced by Shannon [11]. For a random variable X we calculate the entropy

$$H(X) = -\sum_x p(x) \cdot \log p(x) \tag{1}$$

that corresponds to the amount of uncertainty we encounter before we face the actual state of that variable. Information, in this context, can then be described as the reduction of uncertainty when the actual state is observed. To illustrate, a variable that has always the same state bears no uncertainty before observation, and hence yields no information after being observed. Its entropy is zero. The highest entropy means that the variables states are all of equal probability, and therefore offer the most information when actually observed.

The notion of conditional entropy

$$H(X|Y) = \sum_{y} p(Y = y) \cdot H(X|Y = y) \tag{2}$$

$$H(X|Y = y) = -\sum_{x} p(X = x|Y = y) \cdot \log(p(X = x|Y = y)) \tag{3}$$

puts two variables in relation. It describes how large the entropy of one variable is if the other one is already known. If it vanishes, it means that one variable can be perfectly predicted given the other. Given Bayes theorem

$$P(X|Y) = \frac{P(Y|X) \cdot P(X)}{P(Y)} \tag{4}$$

the conditional entropy can be calculated by subtracting a normal entropy of one variable from the joint entropy (6) of two variables

$$H(X|Y) = H(X,Y) - H(Y) \tag{5}$$

The joint entropy

$$H(X,Y) = -\sum_{x,y} p(x,y) \cdot \log(p(x,y)) \tag{6}$$

is then essentially a regular entropy calculated on the direct product of two random variables.

Note that, in an optimal case, for the conditional entropy to be calculated as in (5) the number of samples needs to be significantly larger than the square of the number of sensor states. Since the joint entropy is calculated on the direct product of two sensors, it uses the joint distribution of those two sensors. The random variable used to calculate this is a direct product of the two concerned variables, the number of states it can assume is therefore the product of the number of their states. If now the amount of samples is of the same order of magnitude as the states of the concerned random variables, the calculated entropy becomes meaningless, since most states are hit only once or not at all [7]. This is also a reason why our approach only uses pairwise comparison of two variables. If the informational dependencies of several variables are concerned, the calculation of joint entropies with several variables is necessary. Those “multi-information” entropies require an amount of samples that is significantly larger than the number of states to the power of the variables involved. This huge amount of samples can hardly be obtained, and would make calculation extremely time-consuming and therefore impractical for living reactive systems.

The measurement we like to employ for our pair wise comparison is an information distance $d(X, Y)$ introduced by Olsson et al. [9]. It is calculated as the sum of the two conditional entropies

$$d(X, Y) = H(X|Y) + H(Y|X). \tag{7}$$

It can be calculated using (5) as

$$d(X, Y) = 2 \cdot H(X, Y) - H(X) - H(Y). \tag{8}$$

This measurement was successfully used to reconstruct the spatial relation structure of unstructured sensors [9]. In [8] Olsson et al. also showed that the information distance outperformed other measurements such as the Kullback-Leibler divergence or the Jensen-Shannon divergence for the given task of spatial reconstruction of sensor fields. Compared to the prior mentioned divergence measures, the information distance also has the additional property of being a metric [3] if we consider random variables with one to one correspondence of states as equivalent. This will be important in subsection 2.3.

$$I(X, Y) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x) \cdot p(y)}. \tag{9}$$

$$d(X, Y) = H(X, Y) - (H(X|Y) + H(Y|X)) = H(X, Y) - I(X, Y) \tag{10}$$

Mutual Information as defined in (9) has a close relation to information distance. One can be calculated in regard to the other using Equation (10), but there are some significant differences between those two values. Given two random variables X and Y, the mutual information tells us how much information one variable contains about the other. But this measurement tells us nothing about the original joint entropy of the two variables $H(X, Y)$. If the joint entropy grows while the mutual information remains constant, the information distance, calculated as the difference between joint entropy and mutual information (10), will also increase. Therefore, the mutual information is not directly related to the information distance in general, unless the joint information of the two variables is fixed.

2.2. *Building a Hierarchical Tree Structure*

For the next step we assume that each sensor S is a node in a directed graph. In the beginning no nodes are connected, and all the present nodes are members of a subgraph called "active nodes". As an initial step the information distances between all nodes are calculated using $d(.,.)$. The two nodes with the smallest distance in the subgraph of active nodes are selected. Since those nodes should have the highest informational dependency encoding them into one single node should result in the smallest loss of overall information. Those two nodes are removed from the subgraph of active nodes, and are both connected to a newly created node which is then inserted into the subgraph of active nodes. This new node will be called their parent node, the two original nodes are its child nodes. The random variable of the parent is then constructed using only the two random variables of its two child nodes. As a result the number of active nodes is reduced by one, and the number of nodes in general is increased by one. This step is then iterated until only one active node is left. The result is a directed acyclic graph, a tree structure, in which all leaves are nodes associated with real sensors, while all other nodes are random variables depending only on their child nodes. If the procedure is stopped n steps prior to the end the result are n trees that all encode a certain area of the input image, and reduce the input image to n sensor values.

2.3. *Construction of the Parent Node*

Since the tree building process is iterative, distances between the new parent node and all other existing active nodes have to be calculated. So we have to define a random variable Z that is the new parent node. An easy solution would be $Z = X \times Y$, the direct product of both child nodes variables X and Y. This solution is not satisfactory for two reasons: First, the calculation would encounter the common problems for calculating "multi-information" [1]. The number of states of Z would increase dramatically as the combination of nodes progresses. The calculation would become unfeasibly time-consuming, and the amount of samples would not be sufficient to get good statistical data. Second, by choice of design, we wanted to build a system that acts as an information bottleneck [15] and limits the available capacity to transport data, so as to force the system to determine which information should be preserved.

Thus the resulting random variable should not have more states than the original two that were joined. To achieve this, we created a mapping table that has as many rows and columns as the two original variables have states, in our case 15. Each cell of the mapping table now contains an integer between 1 and 15. This mapping table defines a function that assigns to each pair of input states the output state in the mapping table.

To optimise this discrete non-injective mapping function we formulate two objectives. We select the two nodes that are closest according to our information distance, and should therefore be able to encoded them with the least loss of information. We now need to define an objective that preserves as much information as possible. This should ensure that after each step the remaining nodes contain as much information about the original image as possible. The mutual information $I(Z,(X,Y))$ corresponds to the amount of information Z contains about (X,Y). But since $Z = f(X,Y)$, where f() is a deterministic function, all the information in Z is determined by (X,Y). It follows that $H(Z|X,Y) = 0$ and therefore $I(Z;(X,Y)) = H(Z)$, because all the information Z contains is about (X,Y). Therefore we can simply maximise the entropy $H(Z)$.

The other objective is motivated by the metric property [3] of our information distance $d(X,Y)$. We know that neighbouring nodes have a small distance and are therefore situated close to each other in the metric space defined by $d(X, Y)$. We want the resulting node Z to be situated near the two original nodes X and Y, so if successively new nodes are joined with Z, they will also be close to X and Y. Thereby our algorithm can find nodes that have a short information distance to a whole cluster of nodes by comparing only the distance to its common parent node. To achieve this closeness we aim to minimise the distances $d(Z, X)$ and $d(Z, Y)$. The total cost function

$$\Omega(Z, X, Y) = \lambda(d(Z, X) + d(Z, Y)) - (1 - \lambda) \cdot H(Z), \tag{11}$$

incorporates both terms and weights them with a weighting factor λ.

Since all of the 15×15 state pairs can be mapped to 15 states we deal with 15 to the power of 255 different functions. Searching all of them to find the optimal function is too complex, so we use a simple hillclimbing heuristic to find a good mapping function that performs well given our constraint function $\Omega(Z, X, Y)$. We start with a randomly initialised mapping table. A copy is made, and this copy is changed in one random location to a new random value. Both are compared according to their Ω value, and the better one is kept. This is repeated up to the point were 1000 consecutive changes did not improve the function.

3. Measurements

To analyse the properties of the tree structure we record and calculate the following properties:

3.1. *Partition Visualisation*

Each consecutive step of the algorithm reduces the active nodes by one. The remaining nodes are either original sensors itself, or the parent nodes of several sensors. To illustrate which regions of sensors are joined, and when those are joined we record the partition that is created by the remaining active nodes after each step. A grid is drawn that corresponds to the number and spatial relation of all sensors; each sensor is represented by a small square of colour. For each remaining active node all descendants of that node are coloured in the same way. So the partition image taken n steps prior to the termination of the algorithm has $n + 1$ different colours. All the nodes of the same colour are in that step encoded into a single active node. Since we used a sensor array of 15×15 sensors, all the partition images consist of 15×15 coloured squares.

3.2. *Tree Depth*

To identify areas where the adaptive sensor structure employs a higher resolution we also show another coloured image corresponding to the tree depth for each original sensor node. Since there is no guarantee that the tree structure our algorithms builds is balanced, some nodes might be further removed from the root of the tree.

Those will be coloured in a lighter grey-level, and we will regard those areas as having lower resolution. If we take the last n active nodes and use their states to gather information about a given image, a node that is further removed from those nodes has to encode its information together with several other nodes until it gets to the active node. Conversely, a node that is a direct child of an active node has to encode its information only with one other node. Being close to the root of the tree is therefore associated with having a higher resolution. Those nodes are coloured darker.

3.3. *Identification Probability*

When we reduce the sensor input to increasingly fewer nodes, an ability we want to retain is to differentiate between different sensor inputs. We want to be able (as much as possible) to identify the original sensor images $i \in I$ even if the data is already reduced, where I is a set of all the sample images. To measure this, we define $g(i,t)$ as the function that considers the tree structure when t active nodes are left, and produces the output those nodes would create, given the input sample image i. By counting how many input images in I produce the same output of $g(i,t)$, we can use Bayes Theorem (4) to calculate the conditional probability of every image as

$$p(i|g(i,t)) = \frac{p(g(i,t)|i) \cdot p(i)}{p(g(i,t))} = \frac{p(i)}{p(g(i,t))} \tag{12}$$

The result of this function is the probability that we can identify a specific input image i if we could only look at $g(i,t)$. If we sum that up for every image i and divide it by the number of input images we get

$$\Psi(t) = \frac{\sum_I p(i|g(i,t))}{|I|}. \tag{13}$$

$\Psi(t)$ has a value between one and zero, and describes how large our probability is to identify the original sample image given only the information of the t last remaining nodes. The value of $\Psi(t)$ is a measurement of how ambiguous the output of the remaining tree nodes is, the lower the value the higher the ambiguity. The value of $\Psi(t)$ decreases monotonically as t becomes smaller. In some cases it already starts below one for the full number of sensor nodes, because some of the images in the sample space are identical.

It should be noted at this point that our aim is not to produce the best possible compression algorithm, in which case the value of $\Psi(t)$ would be of paramount importance. But in our case we just want to observe $\Psi(t)$ to determine the information preserving capabilities of the algorithm. Also, the eventual collapsing of different input states into the same states is a necessary step of categorisation.

4. Experiments

This section will describe several experiments which mainly differ in their input data. The general parameters stay the same, unless otherwise noted. The sensor inputs or colours are binned into 15 equally sized bins, and λ is set to 0.5. The differences in the structures built by the algorithm are created by the environments perceived by the sensors.

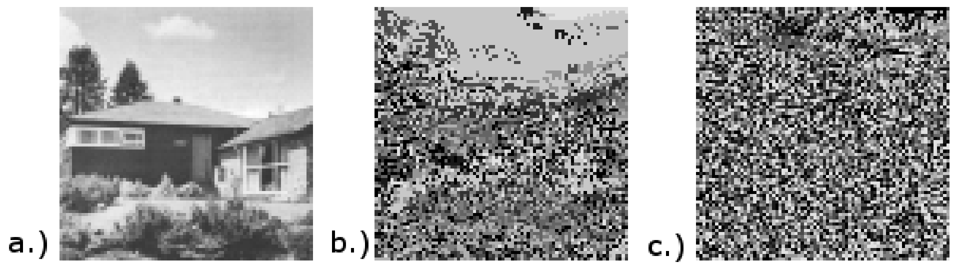


Fig. 1. a.) A 100×100 part of the image used to create the samples b.) and c.) are a visualisation of 100×100 top level node values corresponding too the position of the input mask in image a.), b.) uses $\lambda = 1.0$ c.) $\lambda = 0.0$

4.1. *Normal Images*

For the first set of input images we create a set of 15×15 sensor nodes, so we need images of the size of 15×15 pixels as samples. We copy 10 000 samples into our stack by moving a 15×15 mask over the image shown in Fig. 1. The image of the house was selected because it contains a range of different grey values and different structures. Other images with comparable properties, like the famous Lena image, produced similar resulting structures. We tested the algorithm over a range of about 10 different images and our observation was that the resulting structures are qualitatively the same.

First we observed the identification probability $\Psi(t)$ as seen in Fig. 2, in regard to the remaining number of nodes. As expected, the value of $\Psi(t)$ shrinks when the number of nodes is reduced. In the first phase the reduction is quite small and the structure is still able to discriminate between most of the original sample images. A phase transition appears when the number of nodes reaches an amount where even a perfect encoding would not be able to encode all the samples differently. If every node has 15 states, and we deal with 10 000 samples this limit lies between the fourth and the third node, but the prediction probability already starts to break down a few nodes earlier, around the seventh or eighth. But up to that level, it can tell most images apart. So a look at the structure when it is most reduced, but still able to discriminate can be expected to be informative.

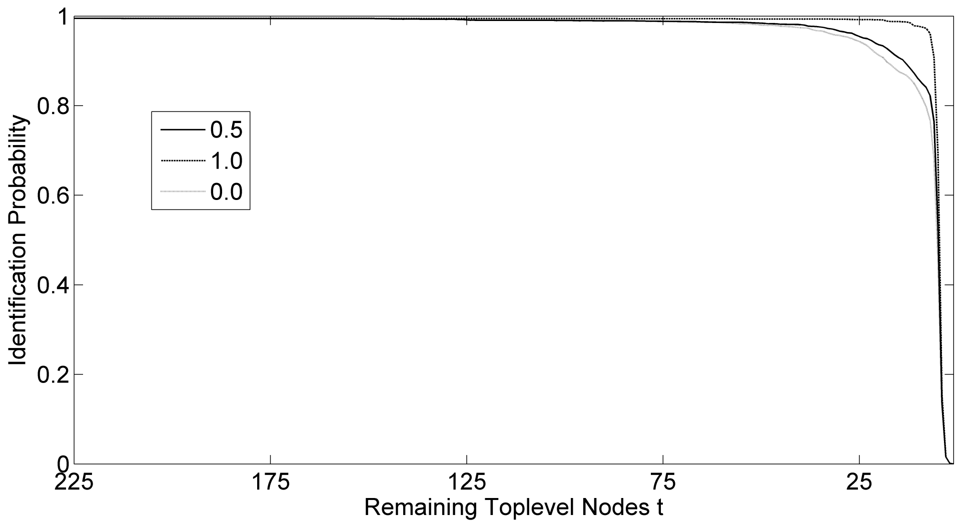


Fig. 2. Graphs depicting the identification probability $\Psi(t)$ depending on the decreasing number of active nodes t. Then three graphs use different values of 1.0, 0.5 and 0.0 for λ

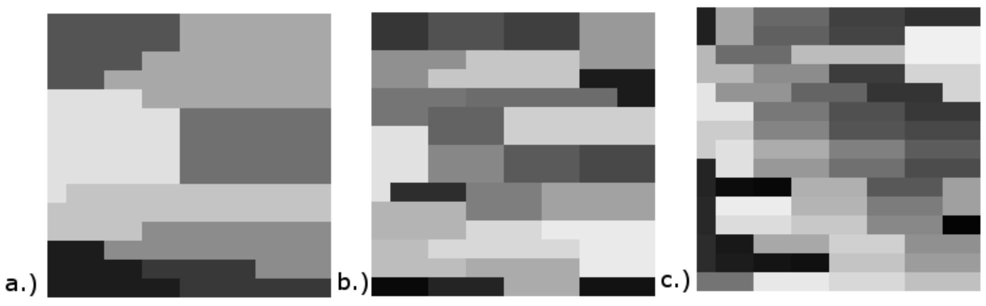


Fig. 3. Partition visualisation of the original 15×15 input sensor nodes created with the input data shown in Fig. 1, a.) contains 8 remaining toplevel nodes b.) contains 26 nodes c.) contains 60 nodes.

If we look at the partition visualisation for eight nodes in Fig. 3, we see an even partition that could have been created by a human tasked with reducing the amount of sensor data to eight single values. Roughly equally sized blobs of adjacent sensors are joined into one. A look at all the other partition images shows, that first neighbours are joined, and then those are iteratively fused into bigger and bigger sensor clusters. The property of those clusters to be of equal size and evenly spread over the whole sensor grid is pertained at every reduction step.

4.2. *Different λ-values*

Fig. 2 also shows the identification probability depending on different values of λ. Those influenced the way how the parent node was constructed. If we are only concerned about the entropy of the parent node and set λ to 0.0, we get a better value for $\Psi(t)$ but the parent nodes are not close to their child nodes anymore in regard to the information distance. To illustrate this one has to imagine that the information metric $d(.,.)$ defines a metric space[3]. Groups of sensors that are mutually close in information distance form clusters in that space. A subset of the points in that space now contains all the random variables that have the maximum entropy, which is the same as saying, all their states have an equal probability in their distribution. If we now try to encode two sensors from one of those original clusters there is no guarantee that a point of that very subset is close by in regard to the distance $d(.,.)$. But,if the resulting parent node of two sensors is only constructed with the entropy constraint, it would be placed somewhere in that subset. Therefore, it could be arbitrarily far away in terms of information distance from the cluster of random variables its child nodes are in and would not naturally fuse with nodes from its original cluster. Our statistics support this claim by showing that after most nodes are fused once the distance between newly fused nodes increased dramatically, meaning that the nodes fused now are far away in regard to their information distance. This effect does not occur if we set λ to a higher value. The graph in Fig. 4 shows how the different value of λ influences the distance of joint nodes. The bump in the middle of the graph for $\lambda = 0$ is the mentioned increase in distance after all the nodes have been joined once.

If we set λ to 1.0, we are not concerned with the entropy any more, and our $\Psi(t)$ value decreases faster as t drops. Even so, it is still slowly decreasing before it enters the phase transition. On the other hand the distance between sensor clusters is much more stable. Another interesting property that becomes visible is the value of the top level node. If we use the tree structure to inspect every given 15 time 15 patch of the original image in Fig. 1 and then record the value of the top level node in those coordinates, we obtain the image shown in Fig. 1.b. Note, that the grey levels are arbitrarily chosen. This image offers a primitive segmentation, and certain regions of the original image produce the same resulting top level node. In comparison the top-level node-image created for $\lambda = 0.0$ looks like random noise.

4.3. *Dominant Line Images*

In previous experiments is was apparent that the sensor are likely to be grouped horizontally. This was not an artefact of the algorithm; if the image was turned by 90 degrees, the grouping would be vertical. This would indicate that the direction of the most frequent line structures is horizontal in our images, and therefore a higher degree of informational dependence existed between horizontally adjacent pixels.

To investigate this further, we used samples containing only diagonal stripes. This dominant line structure reflects in the tree structure. The partition images

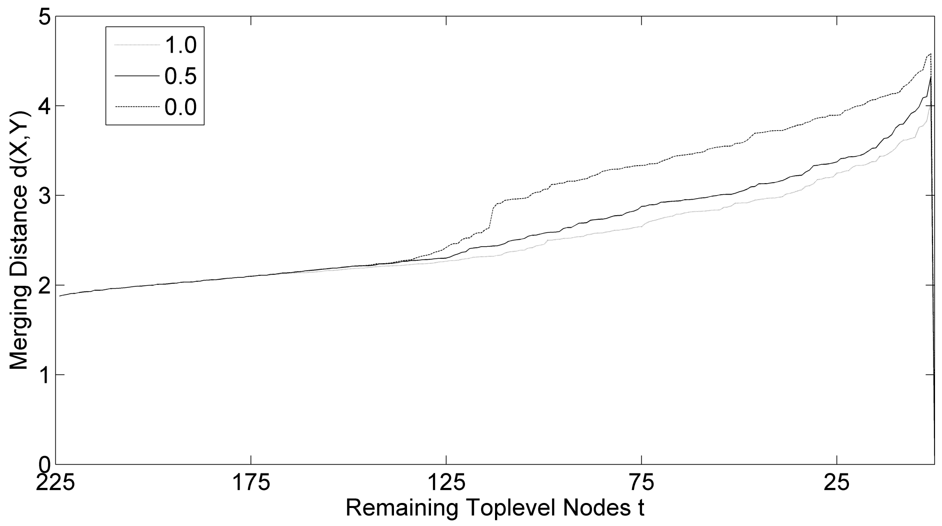


Fig. 4. Graphs depicting distance d(.,.) of the currently joined nodes depending on the decreasing number of active nodes t. The graphs use a.) $\lambda = 0.0$ b.) $\lambda = 0.5$ c.) $\lambda = 1.0$

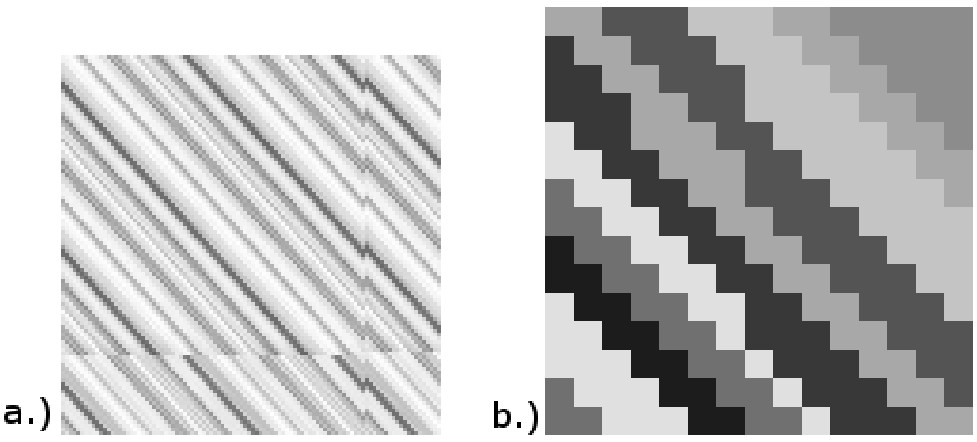


Fig. 5. a.) An 114×114 image with dominant diagonal line structure to create samples. The line artifact is intentional, a result of the repetition of the image to create more samples b.) resulting partition with eight remaining toplevel nodes

show that at first the neighbouring nodes are joined, most already with diagonal neighbours. The partition in Fig. 5 depicts the larger sensor clusters, which are all of the form of elongated diagonal stripes. Apart from that, the structure is comparable to the other structures we encountered in earlier experiments. This appears to be a reasonable adaptation, since most of the pixels on a diagonal line assume the same or similar value, thus joining them inflicts no great information loss.

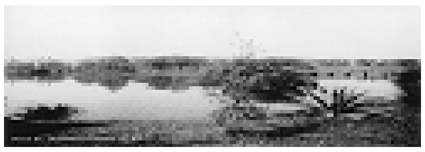
This result could also be interpreted as a rudimentary line or feature detection algorithm. If a certain feature, such as a line, appear regularly in an input image the group of sensors that would simultaneously detect that feature would develop a higher informational dependence. This is demonstrated in this experiment for diagonal lines. In this case all the sensors on a diagonal line were much more likely to perceive the same input, and would therefore be grouped together in the resulting tree. There would still be no explicit node in the tree that would indicate the occurrence of a line with a corresponding value change, but rather the algorithm would implicitly use the existing line to assume certain features of the environment, and could then in turn use the remaining capacity to encode some other less predictable aspects of its sensor inputs.

4.4. *Frog View*

In our third experiment we wanted to synthesise data from a certain viewpoint. Imagine a frog sitting close to a pond, he is fixed in one position and can only move his head up, down and to the side. Its visual field is always a 15×15 pixel size piece of the image shown in Fig. 6. To create a sufficient amount of sample data the frog looks randomly at a 15×15 part of the image. Note that the image height is 42 pixels, roughly the double of the visual field. So it is possible for the frog to see an image only depicting sky or only depicting ground but most of them contain contributions from both.

The resulting tree structure is an unbalanced tree which is visualised with the tree depth image in Fig. 6. Also, in Fig. 7 the upper half of the sensors are already joined into a single cluster, while the lower half still has several clusters left. This adaptation to the images appears reasonable, since the lower half of the images offer more interesting features to see, while the upper half contains mostly sky. So, to know what part of the image one was looking at, having a better resolution in the lower half is desirable.

In our last experiment we introduced a moving object, a “fly”, into the visual field of our frog. It was symbolised by a random black dot that appeared in the top four rows of sensors. To model the fact that the “fly” moves in front of the background it was just copied over the piece of background perceived without the “fly”. Since the order of the sensor input is irrelevant for the results of our algorithm (a permutation of the inputs would yield the same result) this is not approach to motion detection. By making our input time-invariant the order of the time slices becomes irrelevant. The information motion would creates is only available as if it was projected into one single time slice. This loss of knowledge about the order of time can only reduce our information about the actual movement, but a certain amount of information should remain. And this is the information our algorithm a.)



b.)

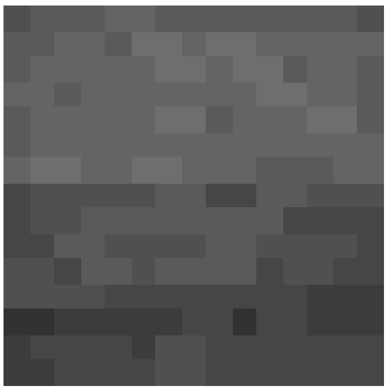
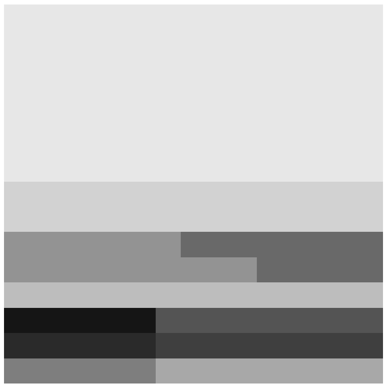


Fig. 6. a.) the picture that was used to create a 15 × 15 pixel view, the image itself is 42 × 142 pixels big. b.) the tree depth shows that the lower part of the image has a better resolution, its nodes are closer to the root and therefore darker coloured.

reacts to.

a.)



b.)

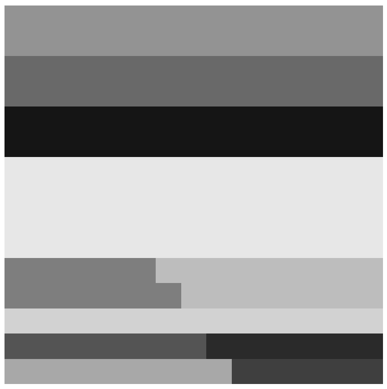


Fig. 7. A direct comparison between two partition images with 11 nodes. a.) created with a static view, the top is already merged into one single node. b.) with an added moving object in the first 4 rows, those rows are now of higher resolution because they are still split into two sensor clusters.

This modification changed the distribution of resolution again, and now the upper part of the image got a better resolution, as seen in the compared partition images in Fig. 7. The tree depth analysis also showed a slight shift of resolution towards the upper four rows. Note so, that the amount of sensor nodes depicted in Fig. 7. might not be to useful to actually track the fly, but is good in illustrating the difference in resolution to a human observer. To track the fly the frog would want to use a less reduced sensor network, in which this imbalance in resolution would also be present but less obvious to the human eye. On that level a lot of sensors would be grouped together with four or five other sensors, while the sensors with high resolution would still be grouped in groups of two or three.

This adaptation also would appear useful to a living organism, since spotting a moving object in front of a static background is useful. So if those objects are small, but have a tendency to appear in the same region, a better resolution in that area might be desired.

5. Discussion

The analysis of our network-like tree structure indicates that, if nothing else, the algorithm offers a way of encoding a broad range of sensor information while still being able to discriminate it. This effect seems to be persistent over a range of different input images. Furthermore, if we define the environment as the images perceived or used for the algorithm a certain amount of adaptation to this environment takes place. If images with unusual properties are offered to the algorithm, the resulting data structure then encodes “meta-information” about its world into its structure.

If a certain sensor region offers low or no information it might be reasonable to get rid of it, or at least reduce its sensors capacity. The first frog experiment mimics this approach by fusing numerous sensors that only ever depict empty sky. In this case the informational dependence between the sensor was extremely high, and a whole area of sensors could be predicted with only few values. It seems only efficient to reduce the actual sensor input to only a few states. Several biological parallels exist to that idea, like the reduced sensitivity of the skin in regions other than the hands, or even animals that get rid of certain sensors that they do not need anymore. The second frog experiment also shows that motion creates a lower informational dependence in the sensor input, which in turn results in better sensor resolution.

The other observed adaptation, not in resolution, but in the actual spatial configuration of joined sensors concerned a world where a certain line structure was dominant. This intended exposure to a very specific set of data created a highly specialised structure that would have problems coping with a more general environment. But this weakness again has some analogue in nature. Cats that are raised in a world where only horizontal lines are present become unable to perceive other lines [14]. If we perceive this as a information theoretical trade-off between accuracy and efficiency [5] the adaptation performed here becomes reasonable. The fact that it might prove fatal if the conditions in the world change is unfortunate, but concurs with phenomena in actual animals.

The system presented here still has several limitation. First, its insensitivity to the actual order of input states makes it unable to perceive the flow of time. It also partitions the image in a very strict sense, so it is not able to produce a result, where one sensor influences two of the resulting high level nodes.

The question also arises how practical a system such as this would be in a biological system. First it should be noted that the tree structured network always contains the same number of nodes. Just the connections differ, and even the number of connections is always the same. So an implementation in hardware would be possible, since no new nodes would actually be created. The other advantage of this system is that only pairwise comparison is used, thus no global control mechanism has to be used to guide the process.

In conclusion we presented a fast and adaptive process that can pre-structure rich sensor data into a data space of desired size, while tightly controlling the amount of information loss.

6. Future Prospects

Since the algorithm only deals with random variables, it can also be applied to a wide range of sensor data rather than only receptive fields. It is possible to compare sensors with different amount of states, and to incorporate the states of actuators as well. Extensions to non robotic fields of sensor analysis are, such as sensor networks, can be envisioned.

To further extend this model, new aspects could be incorporated into the algorithm. In the context of robotics or reactive systems the ability to update the tree structure according to realtime data would be useful. Also, an explicit incorporation of time, e.g. in form of adjacent time slices, into the model could open new possibilities. An analysis of neighbouring time slices could be a first step in this direction.

References

[1] Ay, N. and Knauf, A., Maximizing multi-information, *Kybernetika* **42** (2007) 517–538.

[2] Bialek, W., Physical limits to sensation and perception, *Ann. Rev. Biophys. Biophys. Chem.* **16** (1987) 455–478.

[3] Crutchfield, J. P., Information and its metric, in *Nonlinear Structures in Physical Systems Pattern Formation, Chaos and Waves* (Springer Verlag, 1990), pp. 119–130.

[4] Kraskov, A., Stogbauer, H., Andrzejak, R. G., and Grassberger, P., Hierarchical clustering based on mutual information (2003), `http://www.citebase.org/abstract?id=oai:arXiv.org:q-bio/0311039`.

[5] Lars Olsson, C. L. N. and Polani, D., Information trade-offs and the evolution of sensory layouts, in *Proc. Artificial Life X* (2004).

[6] Linsker, R., A local learning rule that enables information maximization for arbitrary input distributions, *Neural Comput.* **9** (1997) 1661–1665.

[7] Nemenman, I., Shafee, F., and Bialek, W., Entropy and inference, revisited, *Advances in neural information processing systems* **14** (2002).

[8] Olsson, L., Nehaniv, C., and Polani, D., Measuring informational distances between sensors and sensor integration, in *Proc. Artificial Life X* (2006).

[9] Olsson, L., Nehaniv, C. L., and Polani, D., Sensory channel grouping and structure from uninterpreted sensor data, in *in 2004 NASA/DoD Conference on Evolvable Hardware* (IEEE Computer Society Press, 2004), pp. 24–26.

[10] Pierce, D. and Kuipers, B., Map learning with uninterpreted sensors and effectors, *Artificial Intelligence* **92** (1997) 169–229.

[11] Shannon, C. E., A mathematical theory of communication, *Bell System Technical Journal* **27** (1948) 379–423.

[12] Simoncelli, E. P. and Olshausen, B., Natural image statistics and neural representation, *Annual Review of Neuroscience* **24** (2001) 1193–1216.

[13] Slonim, N., Friedman, N., and Tishby, N., Multivariate information bottleneck (Morgan Kaufmann Publishers, 2001), pp. 152–161.

[14] Stryker, M. P., Sherk, H., Leventhal, A. G., and Hirsch, H. V. B., Physiological consequences for the cats visual cortex of effectively restricting early visual experience with oriented contours, *Journal of Neurophysiology* **41** (1987).

[15] Tishby, N., Pereira, F. C., and Bialek, W., The information bottleneck method, in *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing* (1999), pp. 368–377.