An Associative Perception-Action Structure Using a Localized Space Variant Information Representation

Gösta H Granlund

Computer Vision Laboratory, Department of Electrical Engineering,
Linköping University, SE-581 83 Linköping, Sweden
gosta@isy.liu.se

Abstract. Most of the processing in vision today uses spatially invariant operations. This gives efficient and compact computing structures, with the conventional convenient separation between data and operations. This also goes well with conventional Cartesian representation of data.

Currently, there is a trend towards context dependent processing in various forms. This implies that operations will no longer be spatially invariant, but vary over the image dependent upon the image content.

There are many ways in which such a contextual control can be implemented. Mechanisms can be added for the modification of operator behavior within the conventional computing structure. This has been done e.g. for the implementation of adaptive filtering.

In order to obtain sufficient flexibilility and power in the computing structure, it is necessary to go further than that. To achieve sufficiently good adaptivity, it is necessary to ensure that sufficiently complex control strategies can be represented. It is becoming increasingly apparent that this can not be achieved through prescription or program specification of rules. The reason being that these rules will be dauntingly complex and can not be be dealt with in sufficient detail.

At the same time that we require the implementation of a spatially variant processing, this implies the requirement for a spatially variant information representation. Otherwise a sufficiently effective and flexible contextual control can not be implemented.

This paper outlines a new structure for effective space variant processing. It utilises a new type of localized information representation, which can be viewed as outputs from band pass filters such as wavelets. A unique and important feature is that convex regions can be built up from a single layer of associating nodes. The specification of operations is made through learning or action controlled association.

1 Introduction

Most of the processing in vision today uses spatially invariant operations. This gives efficient and compact computing structures, with the conventional convenient separation between data and operations. This also goes well with conventional Cartesian representation of data.

Currently, there is a trend towards context dependent processing in various forms. This implies that operations will no longer be spatially invariant, but vary over the image dependent upon the image content.

There are many ways in which such a contextual control can be implemented. Mechanisms can be added for the modification of operator behavior within the conventional computing structure. This has been done e.g. for the implementation of adaptive filtering [5].

In order to obtain sufficient flexibilility and power in the computing structure, it is necessary to go further than that. To achieve sufficiently good adaptivity, it is necessary to ensure that sufficiently complex control strategies can be represented. It is becoming increasingly apparent that this can not be achieved through prescription or program specification of rules. The reason being that these rules will be dauntingly complex and can not be be dealt with in sufficient detail.

At the same time that we require the implementation of a spatially variant processing, this implies the requirement for a spatially variant information representation. Otherwise a sufficiently effective and flexible contextual control can not be implemented[2].

Most information representation in vision today is in the form of iconic arrays, representing the pattern of intensity and color or some function of this, such as edges, lines, convexity, etc. This is advantageous and easily manageable for stereotypical situations of images having the same resolution, size, and other typical properties. Increasingly, various demands upon flexibility and performance are appearing, which makes the use of array representation less attractive.

The increasing use of actively controlled and multiple sensors requires a more flexible processing and representation structure. The data which arrives from the sensor(s) is often in the form of image patches of different sizes, rather than frame data in a regular stream. These patches may cover different parts of the scene at various resolutions. Some such patches may in fact be image sequence volumes, at a suitable time sampling of a particular region of the scene, to allow estimation of the motion of objects [6]. The information from all such various types of patches has to be combined in some suitable form in a data structure.

The conventional iconic array form of image information is impractical as it has to be searched and processed every time some action is to be performed. It is desirable to have the information in some partly interpreted form to fulfill its purpose to rapidly evoke actions. Information in interpreted form, implies that it should be represented in terms of content or *semantic* information, rather than in terms of array values. Content and semantics implies *relations* between units of information or symbols. For that reason it is useful to represent the information as relations between objects or as *linked objects*. The discussion of methods for representation of objects as linked structures will be the subject of most of this paper, but we can already observe how some important properties of a desirable representation relate to shortcomings of conventional array representations:

- An array implies a given size frame, which can not easily be extended to incorporate a partially overlapping frame

– Features of interest may be very sparse over parts of an array, leaving a large number of unused positions in the array
– A description of additional detail can not easily be added to a particular part of an array

The following sections of this paper outline a new structure for effective space variant processing. It utilises a new type of localized information representation. The specification of operations is made through learning or action controlled association.

2 Channel Information Representation

A continuous representation of similarity requires that we have a *metric* or distance measure between items. For this purpose, information is in the associative structure expressed in terms of a *channel representation*[9, 4]. See Figure 1.

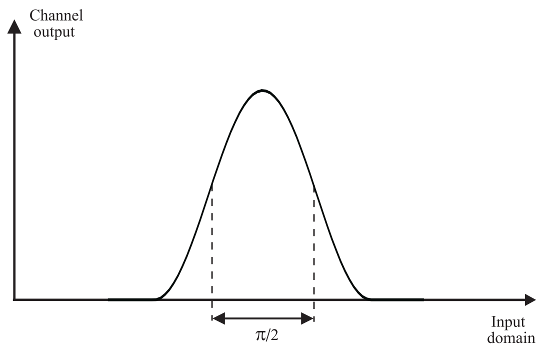

Fig. 1. Channel representation of some property as a function of match between filter and input pattern

Each channel represents a particular property measured at a particular position of the input space. We can view such a channel as the output from some band pass filter sensor for some property /citeg78a. An appropriate object evokes an output from the activated channel, corresponding to the match between the object presented and the properties of the filter, characterizing the passband of the channel. This resembles the function of biological neural feature channels. There are in biological vision several examples available for such properties; edge and line detectors, orientation detectors, etc [3, 8].

If we view the channel output as derived from a band pass filter, we can establish a measure of *distance* or *similarity* in terms of the parameters of this filter. See Figure 1. For a conventional, linear simple band pass filter, the phase distance between the flanks is a constant $\pi/2$. Different filters will have different band widths, but we can view this as a standard unit of similarity or distance, with respect to a particular channel filter.

2.1 Sequentially ordered channels

There are several envelope functions with the general appearance of Figure 1, such as Gaussian and trigonometric functions. Functions which are continuous and have continuous derivatives within the resolution range are of interest. For the introductory discussion of channel representation, we assume the representation of a single scalar variable x, as an ordered one-dimensional sequence of band pass function envelopes x_k, which represent limited intervals, say $k - \frac{3}{2} \leq x \leq k + \frac{3}{2}$, of a scalar variable x. A class of functions which has some attractive properties for analysis is

$$x_k(x) = p_k(x) = \begin{cases} \cos^2(\frac{\pi}{3}(x-k)) & \text{if} \quad k - \frac{3}{2} \leq x \leq k + \frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The scalar variable x can be seen as cut up into a number of local but partially overlapping intervals, $k - \frac{3}{2} \leq x \leq k + \frac{3}{2}$, where the center of each interval corresponds to $x = k$. It should be observed that we use the notation of x without subscript for the scalar variable and x_k with subscript for the channel representation of scalar variable x. The channel output signals which belong to a particular set are bundled together, to form a vector which is represented in boldface:

$$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_k \ \dots \ x_K]^T \tag{2}$$

We assume for conceptual simplicity that the numbers k are consecutive integers, directly corresponding to the numbers of consecutive channels. This allows a more consistent treatment and a better understanding of mechanisms. We are obviously free to scale and translate the actual input variable in any desired way, as we map it onto the set of channels. An actual scalar variable ξ can be scaled and translated in the desired way

$$x = scale \cdot (\xi - translation) \tag{3}$$

to fit the interval spanned by the entire set of channels $\{x_k\}$. We will later see how other nonlinear scaling transformations can be made.

With each channel center representing consecutive integers, the distance between two adjacent channels in terms of the variable x is one unit. From Equation 1 it is apparent that the distance in terms of angle is $\frac{\pi}{3}$ or $60°$. We will in subsequent discussions refer to this as the typical channel distance of $\frac{\pi}{3}$ or $60°$.

In Figure 2 we have a one-dimensional set of 13 sequentially ordered channels. The position of each channel is indicated by the dashed lines. It is designed to provide a channel representation of scalars within a range $0 \leq x \leq 10$. To provide a continuous representation at the boundaries of this interval, the set of channels is padded with an extra channel at each end. Components from these channels are required to perform a reliable reconstruction back to a scalar value from the vector representation. In order to start adapting ourselves to the major purpose of processing of spatial data, we can view Figure 2 as a one-dimensional image

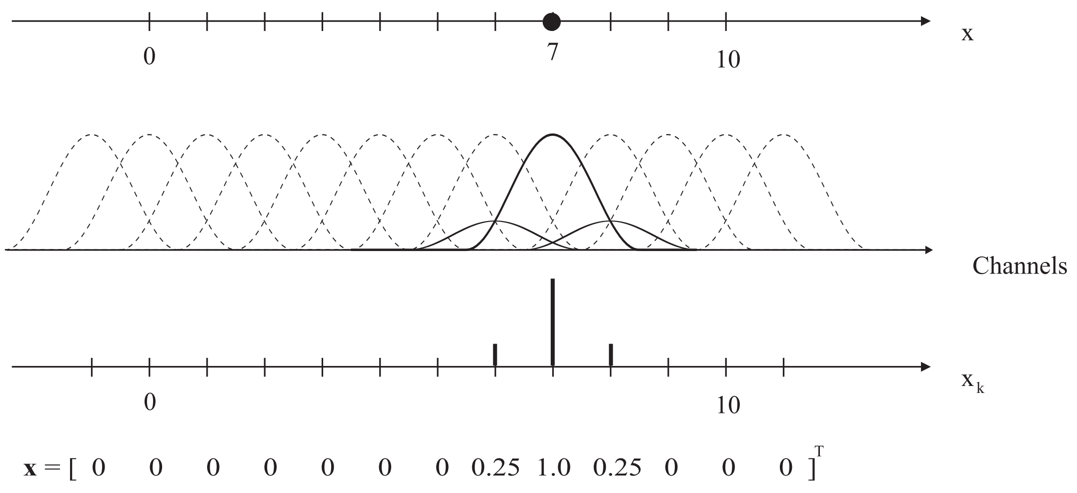

Fig. 2. Channel representation of a scalar $x = 7$

with a single simple object in the form of a dot. The channels are scaled to unit resolution between the filter centers, and the center values correspond to values

$$\mathbf{k} = [\ -1\ \ 0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ ]^T \tag{4}$$

If the set of channels is activated by a scalar $x = 7$, represented by a point at position $x = 7$, we will obtain a situation as indicated in Figure 2. We assume that the output of a channel is given by the position of the point $x = 7$ within its band pass function, according to Equation 1. The channels activated are indicated by the solid line curves. The scalar $x = 7$ will produce the vector $\mathbf{x}$ as indicated in Figure 2.

Below are a few additional examples which hopefully will shed some light on the representation, in particular at the boundaries. We still assume a set of 13 channels which are used to represent scalar values in the interval between 0 and 10.

$$\begin{array}{lcl}
x = 0.0 & \Rightarrow & \mathbf{x} = [\ 0.25\ \ 1.0\ \ 0.25\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]^T \\
x = 3.73 & \Rightarrow & \mathbf{x} = [\ 0\ \ 0\ \ 0\ \ 0\ \ 0.52\ \ 0.92\ \ 0.06\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ]^T \\
x = 9.0 & \Rightarrow & \mathbf{x} = [\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0.25\ \ 1.0\ \ 0.25\ \ 0\ ]^T \\
x = 10.0 & \Rightarrow & \mathbf{x} = [\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0.25\ \ 1.0\ \ 0.25\ ]^T
\end{array} \tag{5}$$

We can clearly see the necessity for padding with extra channels at the boundaries. Under the conditions stated earlier, we have the following values of x_k within an interval $k - \frac{3}{2} \leq x \leq k + \frac{3}{2}$:

$$
\begin{array}{lll}
x_k(k-\frac{3}{2}) & = \cos^2(-\frac{\pi}{2}) & = 0 \\
x_k(k-1) & = \cos^2(-\frac{\pi}{3}) & = 0.25 \\
x_k(k) & = \cos^2(0) & = 1 \\
x_k(k+1) & = \cos^2(\frac{\pi}{3}) & = 0.25 \\
x_k(k+\frac{3}{2}) & = \cos^2(\frac{\pi}{2}) & = 0
\end{array}
\tag{6}
$$

In relation to this, it can be shown that

$$\sum_k x_k(x) = 1.5 \qquad \text{if} \quad -\frac{1}{2} \le x \le K - \frac{1}{2} \tag{7}$$

where K is the last channel used for padding. This consequently gives a margin of 1/2 outside the second last channel. This means that the sum of all channel contributions over the entire channel set from the activation by a single scalar x is 1.5, as long as x is within the definition range of the entire set. Related properties are:

$$x_k(k-1) + x_k(k) + x_k(k+1) = x_{k-1}(k) + x_k(k) + x_{k+1}(k) = 1.5 \tag{8}$$

Most components of $\mathbf{x}$ are zero, with only two or three non-zero components representing the scalar value x as discussed earlier.

An array may be activated by more than one value or stimulus. In Figure 3 we have two scalars, at $x = 1$ and $x = 7$. It is apparent that as the difference between the two scalars decreases, there is going to be overlap and interference between the contributions. This indicates a need to worry about proper resolution, like for any sampling process. Still, the representation gives us the possibility to keep track of multiple events within a single variable, without their superimposing, something which a Cartesian representation does not allow.

2.2 Two-dimensional channels

Most of the information we want to deal with as input is two-dimensional, or possibly of even higher dimensionality. For that reason we will extend the definition to two dimensions, x and y:

$$
p_{kl}(x,y) = \begin{cases} \begin{cases} \cos^2(\frac{\pi}{3}\sqrt{(x-k)^2 + (y-l)^2}\,) \\ \text{if} \quad k - \frac{3}{2} \le x \le k + \frac{3}{2},\ l - \frac{3}{2} \le y \le l + \frac{3}{2} \end{cases} \\ 0 \qquad \text{otherwise} \end{cases}
\tag{9}
$$

The arrangement of sequential integer ordering with respect to k and l is similar to the one-dimensional case. The output from a channel is now dependent upon the distance, $d = \frac{\pi}{3}\sqrt{(x-k)^2 + (y-l)^2}$ from the center of a particular channel at position (k,l) in the array.

As we will see later, good functionality requires that there are several non-zero outputs generated from a sensor array. As we in this case are dealing with point objects, this requires that there is an overlap between the transfer functions

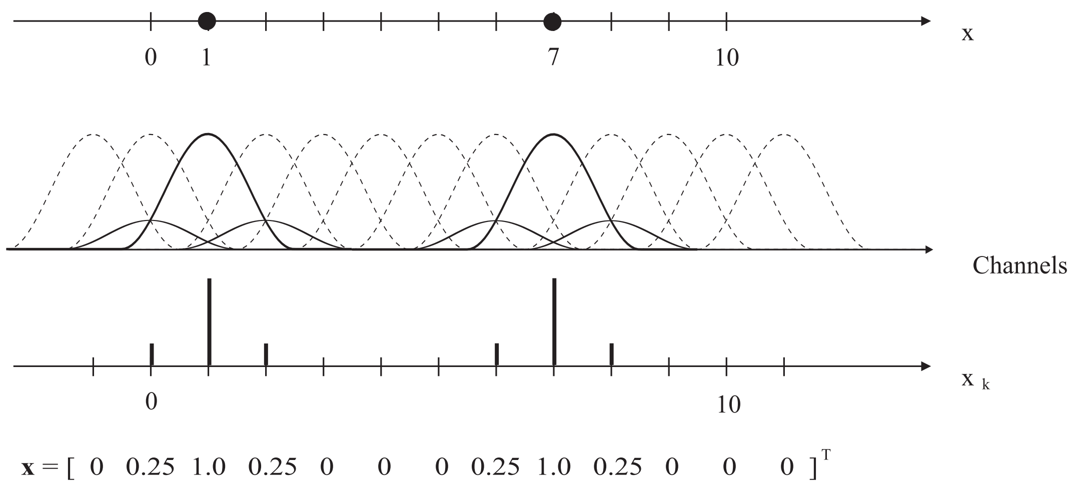

Fig. 3. Channel representation of two scalars at $x = 1$ and $x = 7$

of the different detectors. When the object is a line, or other spatially extended object, no overlap is required. Rather we will see that receptive fields of sensors normally only have to cover parts of the array.

So far, we have only dealt with the position dependent component of the channel band pass function. Generally, there is as well a component dependent upon some property of the sensor, such as dominant orientation, color, curvature, etc. Equation 9 will then take on the general form:

$$p_{klm}(x, y, \phi) = p_{kl}(x, y)p_m(\phi) = \begin{cases} \begin{cases} \cos^2(\frac{\pi}{3}\sqrt{(x-k)^2 + (y-l)^2}\,)\, p_m(\phi) \\ \text{if} \quad k - \frac{3}{2} \leq x \leq k + \frac{3}{2},\ l - \frac{3}{2} \leq y \leq l + \frac{3}{2} \end{cases} \\ 0 \quad \text{otherwise} \end{cases} \tag{10}$$

As this property is often modular and e.g. representing an angle, it has been given an argument ϕ. Because the use of modular channel sets is not restricted to this application, we will give it a somewhat more extensive treatment.

3 Modular Channel Sets

There are several situations where it is desirable to represent a modular or circular variable, such as angle, in a channel representation. There are two different cases of interest:

- Modular channel distance $\frac{\pi}{3}$
- Modular channel distance $\frac{\pi}{4}$

Of these, we will only deal with the first one:

3.1 Modular channel distance $\frac{\pi}{3}$

The structure easiest to deal with has a channel distance of $\frac{\pi}{3}$, similarly to the earlier treatment. In this case, the least complex structure contains three channels in a modular arrangement:

$$\phi_m(\phi) = p_m(\phi) = \begin{cases} \cos^2(\phi - m\frac{\pi}{3}) & \text{if} \quad \mid \phi - m\frac{\pi}{3} \mid \leq \frac{\pi}{2} \qquad m = 0, 1, 2; \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The scalar variable ϕ will be represented by a vector

$$\boldsymbol{\phi} = [\phi_0 \ \phi_1 \ \phi_2]^T = \{\phi_m\} \qquad m = 0, 1, 2 \tag{12}$$

As earlier, we use the notation of ϕ without subscript for the scalar variable and ϕ_m with subscript for the channel representation of scalar variable ϕ. The channels which belong to a particular set are bundled together, to form a vector which is represented in boldface, to the extent that this type font is available.

The modular arrangement implies that as the scalar argument increases from $\frac{2\pi}{3}$ it will not activate a fourth channel but map back into channel 1, which is equivalent to the dashed channel curve in Figure 4. This is the minimum number of channels which will provide a continuous representation of a modular variable. It is for example useful for the representation of orientation of lines and edges. It can be shown that this is the minimum number of filter components which give an unambiguous representation of orientation in two dimensions [5]. If we view the distance between adjacent channels to $\frac{\pi}{3}$ or 60° like in the earlier discussion, this implies that the total modulus for 3 channels is π or 180°. This is well suited for representation of "double angle" [5] features such as the orientation of a line. If it is desired to represent a variable with modulus 2π or 360°, the variable ϕ can be substituted by $\phi/2$ in Equation 11 above. Any different desired modulus can be scaled accordingly. There are several different ways to express the scaling. In this presentation we have tried to maintain the argument in terms of the $\cos^2()$ function as a reference.

Assuming a resolution of a 10 to 20 levels per channel, this will give a total resolution of 3° to 6° given modulus 180° and a total resolution of 6° to 12° given modulus 360°. This is sufficient for many applications. The modular arrangement is illustrated in Figure 4.

If a higher resolution is desired, more channels can be added in the modular set as required. There are several ways to express the scaling, such as in constant modulus or in constant channel argument. The way selected here is in terms of constant argument of the $\cos^2()$ function. This gives a variable modulus for the entire system, but makes it easy to keep track of the type of system. The generalized version becomes:

$$\phi_m(\phi) = p_m(\phi) = \begin{cases} \cos^2(\phi - m\frac{\pi}{3}) & \text{if} \quad \mid \phi - m\frac{\pi}{3} \mid \leq \frac{\pi}{2} \quad m = 0, 1, \ldots, M-1 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

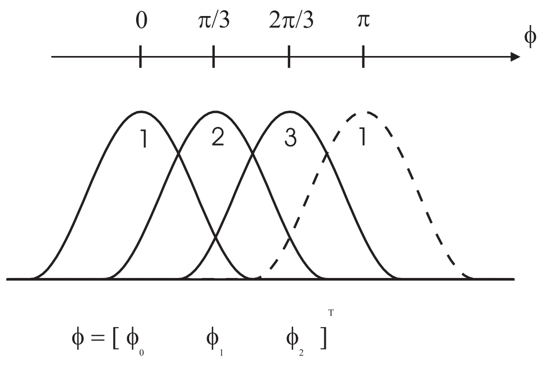

Fig. 4. Three component modular channel vector set

The scalar variable ϕ will be represented by a vector

$$\phi = [\phi_0 \; \phi_1 \; \ldots \phi_{M-1}]^T = \{\phi_m\} \quad m = 0, 1, \ldots, M-1 \quad \text{modulus} \quad M\frac{\pi}{3} \tag{14}$$

4 Variable Resolution Channel Representation

In the preceding discussion we have assumed a constant or linear mapping and resolution for the variable in question to the channel vector representation. There are however several occasions where a nonlinear mapping is desired.

4.1 Logarithmic channel representation

In many cases it will be useful to have a representation whose resolution and accuracy varies with respect to the value of the variable. As an example, we can take the estimated distance z to an object, where we typically may require a constant relative accuracy within the range.

We can obtain this using a logaritmic mapping to the channel representation.

$$z_k(z) = \begin{cases} \cos^2(\frac{\pi}{3}({}^b\log(z - z_0) - k)) & \text{if} \quad k - \frac{3}{2} \leq {}^b\log(z - z_0) \leq k + \frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

It is convenient to view the process of scaling as a mapping to the integer vector set. There are two cases of scaling which are particularly convenient to use:

– One octave per channel. This can for example be achieved by using a mapping $x = {}^2\log(z - z_0)$, where z_0 is a translation variable to obtain the proper scaling for z.
– One decade per two channels. This can for example be achieved by using a mapping $x = {}^{10}\log(z - z_0)/2$, where z_0 is a translation variable to obtain the proper scaling for z.

4.2 Arbitrary function mapping

A mapping with an arbitrary function $x = f(z)$ can be used, as long as it is strictly monotonous. It is possible to employ such a function to obtain a variable resolution in different parts of a scene, dependent upon the density of features or the required density of actions.

4.3 Foveal arrangement of sensor channels

A non-uniform arrangement of sensors with a large potential is the foveal structure. See Figure 5. A foveal window, has a high density of sensors with a small scale near the center of the window. More peripherally, the density decreases at the same time as scale or size of sensors increases. This is similar to the sensor arrangement in the human retina.

The low level orientation outputs from the sensor channels will be produced from the usual procedures. They should have a bandwidth, which corresponds to the size of the sensor channel field, as illustrated in Figure 5. This implies a representation of high spatial frequencies in the center and low frequencies in the periphery.

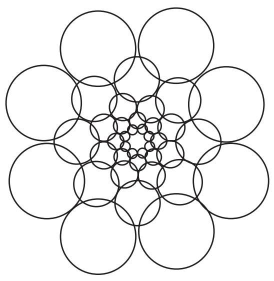

Fig. 5. Foveal arrangement of channels in sensor window

As the computing structure can easily deal with a non-uniform arangement of sensors, there is a great deal which speaks in favor of a foveal arrangement of sensors. It provides a high resolution at the center of the visual field. While the lower resolution towards the periphery does not provide detailed information, it is sufficient to relate the high resolution description to its background, as well as to guide the attentive search mechanism to regions of interest.

5 Arrangement of Channels

The abstract function of this associative structure is to produce a mapping between a set of arbitrarily arranged channels for feature variables, and a set of sequentially ordered channels for response variables. This constitutes a process of recognition.

We assume two distinctly different categories of channel representations:

1. Sequentially ordered channels for response variables
2. Arbitrarily arranged channels for sensor and feature variables

What we have been dealing with so far, can be said to imply the first category of response variables. This reflects the fundamental property that response states are defined along, at least locally, one-dimensional spaces. We assume an availability of consecutive, sufficiently overlapping channels which cover these spaces.

In general, there is no requirement for a regular arrangement of channels, be it on the input side or on the output side. The requirement of an orderly arrangement comes as we need to interface the structure to the environment, e.g. to determine its performance. We typically want to map the reponse output channel variables back into scalar variables in order to compare them with the reference. The mapping back into scalars is greatly facilitated by a regular arrangement.

5.1 Arbitrarily arranged sensor channels

For sensor channels, we assume an arrangement which is typically two-dimensional, or in general multi-dimensional. While the response space is assumed to be one-dimensional as described above, the sensor or feature space is assumed to be populated with arbitrarily arranged detectors, where we have no guarantee for overlap or completeness. See Figure 6. As we will see, there is no problem for the associative structure to use an arbitrarily arranged array of input channels, as long as it is stable over time, because an important part of the learning process is to establish the identity of input sensor or feature channels.

The preferential orientation sensitivity of a sensor is indicated as a line segment, and the extent of the spatial sensitivity function is indicated by the size of the circle. As indicated in this figure, detectors for orientation may typically have no overlap, but rather be at some distance from each other. The reason is that an expected object, such as a line, has an extent, which makes it likely that it will still activate a number of sensor channels

Like any other analysis procedure, this one will not be able to analyze an entire image of say $512 \cdot 512$ elements in one single bite. It is necessary to limit the size of a processing window onto the image. We assume that a sensor map window contains $40 \cdot 40 = 1.6 \cdot 10^3$ orientation detectors, distributed as a two-dimensional array. Each orientation detector contains a combination of an edge and a line detector to produce a quadrature bandpass output. Detectors are assumed to be

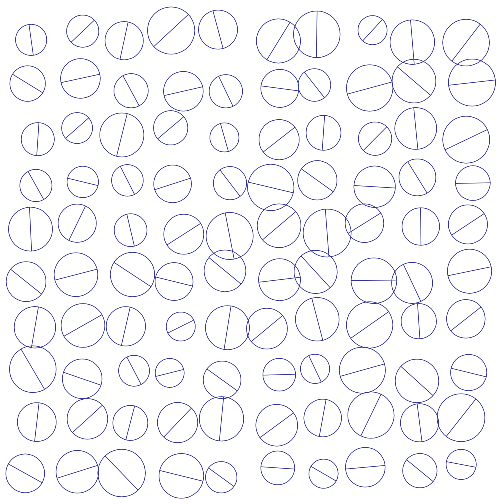

Fig. 6. Example of random arrangement of orientation detectors over space.

distributed such that we only have one detector for some preferred orientation within some neighborhood. This will give a lower effective resolution with respect to orientation over the array, corresponding to around $20 \cdot 20 = 400$ orientation detectors with a full orientation range. Detectors will have to be distributed in an arrangement such that we do not have the situation that there are only detectors of a particular orientation along a certain line, something which may happen with certain simple, regular arrangements.

Given no overlap between sensors, the reader may suspect that there will be situations, where an applied line will not give an output from any sensor. This is true, e.g. when a line is horizontal or vertical in a regular array. It is however no problem to deal with such situations, but we will leave out this case from the present discussion.

The channel representation has an elegant way to represent the non-existence of information, which is something totally different from the value 0. This is very different from the representation in a common Cartesian array, where all positions are assumed to have values, which are as well reliable. The channel representation does not require such a continuity, neither spatially, nor in terms of magnitude. This allows for the creation of a more redundant representation.

6 Feature Vector Set for Associative Machinery

A sensor channel will be activated depending upon how the type of stimulus matches, and how its position matches. The application of a line upon an array as indicated in Figure 6, will evoke responses from a number of sensors along the lenghth of the line.

All sensor channels which we earlier may have considered as different vector sets, with different indices, will now be combined into a *single* vector. We can obviously concatenate rows or columns after each other for an array such as in Figure 6; we can freely concatenate vectors from different sensor modalities one after the other. We will in the ensuing treatment for simplicity assume that *all* sensor channels to be considered, are bundled together into a *single sensor channel vector set*:

$$\mathbf{x} = [x_1 \; x_2 \; \ldots \; x_K]^T \qquad k = 1, \ldots, K \tag{16}$$

We can see each sensor channel as an essentially independent wire, carrying a signal from a band pass filter, describing some property at some position of the image. It is assumed that we have a set of such sensor channels within some size *frame of interpretation*, which is a subset or window onto the image to be be interpreted, which is substantially smaller than the entire image. The *frame of interpretation* may in practise contain somewhere between 10^2 to 10^4 sensor channels, which is equivalent to the dimensionality K of $\mathbf{x}$, dependent upon the problem and available computational resources. This vector is very sparse however, because most sensors do not experience a matching stimulus, which gives the vector a density, typically from 10^{-1} to 10^{-3}.

A particular length of line at a particular position and orientation, will produce a stimulation pattern reflected in the vector $\mathbf{x}$ which is unique. As we will see next, this vector can be brought into a form such that it can be associated with the state vectors (length, orientation, position) related to it.

The set of features used for association, derives from the above mentioned sensor channels, as illustrated in Figure 7.

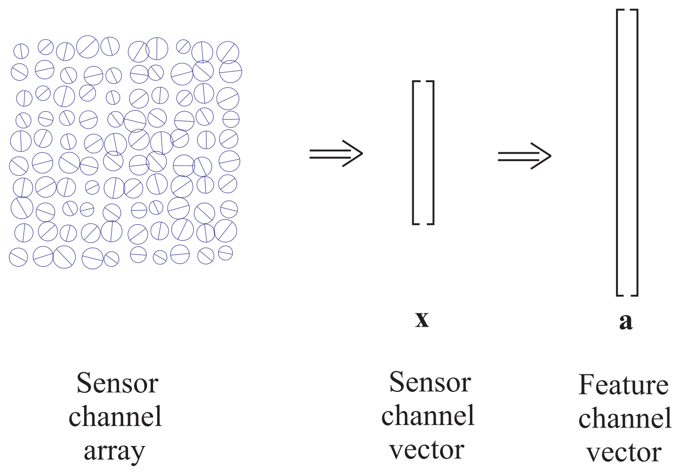

Fig. 7. Illustration of steps in going from sensor array to feature vector

We will use the notation:

– Sensor channel vector set: $\mathbf{x} = [x_1\ x_2\ \ldots\ x_K]^T = \{x_k\} \qquad k = 1, \ldots, K$
– Feature channel vector set: $\mathbf{a} = [a_1\ a_2\ \ldots\ a_H]^T = \{a_h\} \qquad h = 1, \ldots, H$

The sensor channel vector $\mathbf{x}$ is an arbitrary but fixed one-dimensional arrangement of the outputs from the two-dimensional sensor channel array. The sensor channel vector $\mathbf{x}$ forms the basis for the *feature channel vector*, $\mathbf{a}$, which is to be associated with the response state. The feature vector can contain three different functions of the sensor vector:

1. **Linear components** This is the sensor channel vector itself, or components thereof. This component will later be denoted simply as $\mathbf{x}$.
2. **Autocovariant components** These are product components of type $(x_1x_1,\ x_2x_2,\ \ldots\ , x_kx_k)$, which are the diagonal elements of the covariance matrix. The corresponding vector containing these components will be denoted as $\mathbf{xx}^T_{auto}$.
3. **Covariant components** These are product components of type $(x_1x_2,\ x_1x_3,\ \ldots\ , x_{k-1}x_k)$, which are the off-diagonal elements of the covariance matrix. The corresponding vector containing these components will be denoted as $\mathbf{xx}^T_{cov}$.

The feature vector used for association will be:

$$\mathbf{a} = \begin{bmatrix} \mathbf{x} \\ \mathbf{xx}^T_{auto} \\ \mathbf{xx}^T_{cov} \end{bmatrix} \tag{17}$$

of which in general only the last covariant components will be present. Experiments indicate that the covariant feature components are the most descriptive as they describe coincidences between events, but the existence of the others should be kept in mind for various special purposes such as improved redundancy, low feature density, etc.

From this stage on, we will not worry about the sensor channel vector set $\mathbf{x}$, and only use feature channel vector set, $\mathbf{a}$. For that reason you will see some of the indices recycled for new tasks, which will hopefully not lead to any confusion.

Before we go into the rest of the associative structure, and how this feature vector is used, we will recognize the fact that we can recover the conventional scalar meaning of data expressed as a channel vector.

7 Reconstruction of Scalar Value From Channel Vectors

It should first be made clear that this computing structure is intended for consistent use of information represented as channel signals as explained earlier. Input to a computing unit will have the channel representation, as will normally the output. The output from one unit or computing stage will be used as input to another one, etc.

As a system of this type has to interface to the external world, input or output, requirements become different. For biological systems there are sensors available which do give a representation in this form, as well as that output actuators in the form of muscle fibers can directly use the channel signal representation.

For technical systems, it will be necessary to provide interfaces which convert between the conventional high resolution cartesian signal representation and the channel representation. We have in the introduction discussed how this is accomplished for input signals. We will now look at how this can be done for output signals as well. Output signals which will be used to drive a motor or similar device, or for visualization of system states.

The output from a single channel u_k of a response vector $\mathbf{u}$, will not provide an unambiguous representation of the corresponding scalar signal u, as there will be an ambiguity in terms of the position of u with respect to the center of the activated channel u_k. This ambiguity can be resolved in the combination with adjacent channel responses within the response vector $\mathbf{u} = \{u_k\}$. By using a sufficiently dense representation in terms of channels, we can employ the knowledge of a particular similarity or distance between different channel contributions.

It can be shown that if the distance between adjacent channels is 60° or less, we can easily obtain an approximative reconstruction of the value of u as a linear phase. Reconstruction of the scalar value u_e which corresponds to a particular response vector $\mathbf{u}$, formally denoted as *sconv*:

$$u_e = sconv(\mathbf{u}) = sconv(\{u_k\}) \qquad k = 1, \ldots, K \tag{18}$$

can, given the earlier discussion, be implemented in several ways. We will however leave out the details of the computation in this context.

8 System Structure for Training

The general aspects of training are obviously related to the current large field of Neural Networks [7]. Training of a system implies that it is exposed to a succession of pairs of samples of a feature vector $\mathbf{a}$ and a corresponding response vector $\mathbf{u}$.

There are several ways in which this training can be done, but typically one can identify a training structure as indicated in Figure 8.

The *Pseudorandom Training Sequencer* supplies transformation parameters of a training pattern to the system. The most characteristic property of this is that the output variables are guaranteed to vary continuously. The training variables have to cover the space over which the system is expected to operate. The Pseudorandom Training Sequencer is expected to produce its output in conventional digital formats.

The training data is input to the *External World Simulator* or interface. There it is generating the particular transformations or modes of variation for patterns, that the system is supposed to learn. This can be the generation of movements of the system itself, which will modify the precepts available

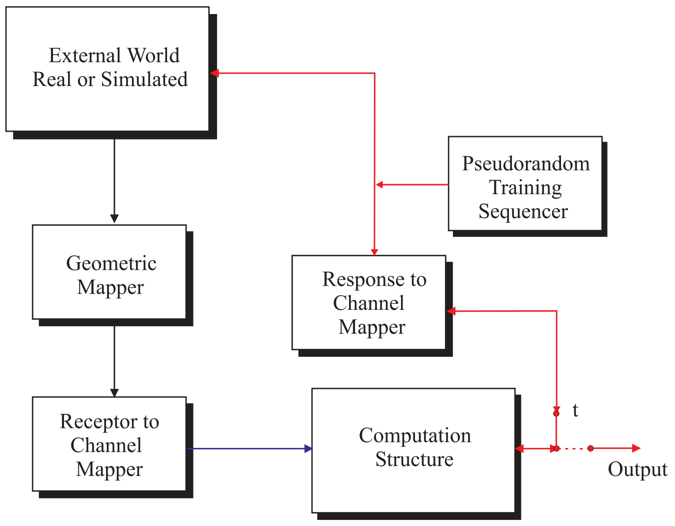

Fig. 8. System structure as set up for training, in interaction with the environment

The *Geometric Mapper* will in the general case produce a two-dimensional projection from a three-dimensional world, such as to implement a camera.

The *Receptor to Channel Mapper* will in the general case convert an image projected onto it, into a parallel set of channels, each channel describing some property according to the discussion earlier.

The training data, representing transformations to the input pattern, is also supplied to the *Response to Channel Mapper*, where it is converted from conventional Cartesian format to the channel representation of the response state, as discussed earlier. This information is supplied directly to the output side of the associative computation structure.

8.1 Basic Training Procedure

The basic training procedure is to run the Pseudorandom Training Sequencer to have it vary its output. This will have an effect onto the external world model in that something changes. The response variables out from the Pseudorandom Training Sequencer will also be fed to the output of the Computing Structure.

We will in this discussion assume *batch mode training*, which implies that pairs of corresponding feature vectors $\mathbf{a}$ and response vectors $\mathbf{u}$ are obtained for each one of N samples, which form matrices $\mathbf{A}$ and $\mathbf{U}$:

$$\begin{cases} \mathbf{U} = [\mathbf{u}_1 \ \mathbf{u}_2 \ \ldots \ \mathbf{u}_n \ \ldots\ldots \ \mathbf{u}_N] \\ \mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_2 \ \ldots \ \mathbf{a}_n \ \ldots \ \mathbf{a}_N] \end{cases} \tag{19}$$

These matrices are related by the linkage matrix $\mathbf{C}$:

$$\mathbf{U} = \mathbf{C}\mathbf{A} \tag{20}$$

From this matrix equation, the coupling or linkage matrix **C** can be solved, superficially expressed as

$$\mathbf{C} = \mathbf{U}/\mathbf{A} \tag{21}$$

The feature matrix **A** may contain tens of thousands of features, represented by tens of thousands of samples. This implies that the method of solution has to be chosen carefully, in order not to spend the remaining part of the millenium solving the equation.

There are now very fast numerical methods available for an efficient solution of such systems of linear equations. These methods utilize the sparsity of the **A** and **U** matrices; i.e. the fact that most of the elements in the matrices are zero. Although this is an important issue in the use of the channel representation, it is a particular and well defined problem, which we will not deal with in this presentation. One of the methods available is documented in a Ph.D. Thesis by Mikael Adlers: Topics in Sparse Least Squares Problems [1].

8.2 Association as an approximation using continuous channel functions

What happens in the computing structure during training, is that the response channel vector **u** will associate with the feature channel signal vector **a**. The association implies that the output response is approximated by a linear combination of the feature channel signals. This is illustrated for an actual case in Figure 9.

We can now compute the approximating function for a particular response node k over the sample points n:

$$u_{kn} = \sum_h c_{kh} a_{hn} \tag{22}$$

The association during the training implies finding the coefficients c_{kh} which implement this approximation. We can see that a particular response channel function is defined over some interval of samples, n, from the training set. We can vary u continuously, and as different channels $..., u_{k-1}, u_k, u_{k+1}, ...$ are activated, their approximation in terms of similarly activated features $..., a_{i-1}, a_i, a_{i+1}$ can be computed. The resulting optimization coefficients $..., c_{kh}, ...$ will constitute *quantitative links* in the linkage matrix **C** between the input feature side and the output response side.

Taken over all response nodes, k, this is written in matrix terms as:

$$\mathbf{u}_n = \mathbf{C}\mathbf{a}_n \tag{23}$$

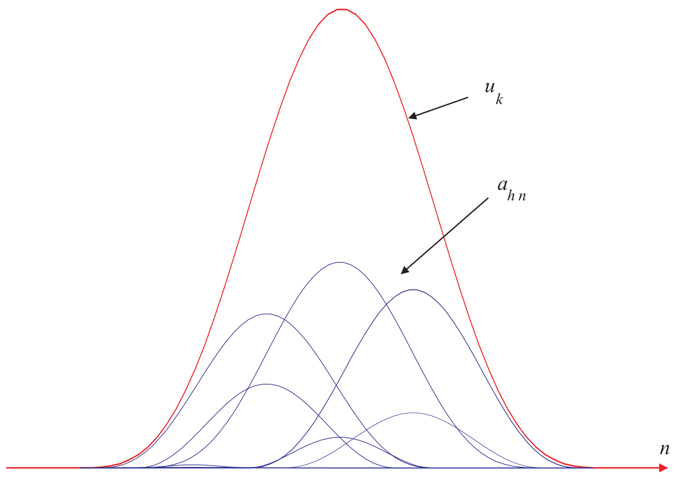

Fig. 9. Illustration of procedure to approximate a response channel function, u_k, with a set of feature channel functions, a_h, over an interval of sample points n

For the entire training set of vectors $\mathbf{u}_n$ and $\mathbf{a}_n$ this is written in matrix form as before:

$$\mathbf{U} = \mathbf{CA} \tag{24}$$

Having somehow completed a training procedure for the entire range of values of u, we can change the switch to output, from the training position `t`, in Figure 8. After this we can present an unknown pattern with feature vector $\mathbf{a}$, within the definition range as input to the system, after which the system will use the linkage matrix derived, $\mathbf{C}$, to compute the actual value of $\mathbf{u}$.

$$\mathbf{u} = \mathbf{Ca} \tag{25}$$

When the training is completed, the computation of the preceding expression for an unknown vector $\mathbf{a}$ is extremely fast, due to the sparsity of the vectors and matrices involved.

9 Properties of the Linkage Matrix C

We have related the set of response states $\mathbf{U}$ and the corresponding percept vectors $\mathbf{A}$ with the matrix equation

$$\mathbf{U} = \mathbf{CA} \tag{26}$$

This matrix equation does not generally have a unique solution, but it can be underdetermined or overdetermined.

For the system to perform as desired, we require a solution with some particular properties:

1. The coefficients of matrix **C** shall be non-negative, as this gives a more sparse matrix and a more robust system. A traditional unrestricted least squares solution tends to give a full matrix with negative and positive coefficients, which do their best to push and pull the basis functions to minimize the error for the particular training set. A particular output may in this case be given by the difference between two large coefficients operating upon a small feature function, which leads to a high noise sensitivity.
2. The coefficients of matrix **C** shall be limited in magnitude, as this as well gives a more robust system. One of the ways to achieve this is to set elements of **A** below a certain threshold value to zero. This is related to the lower treshold part of the S curve, often assumed for the transfer function of real and artificial neurons.
3. Matrices **A** and **U** are sparse, and the entire system can be dealt with using fast and efficient procedures for solution of sparse systems of equations for values between two limits.
4. Coefficients of matrix **C** which are below a certain threshold shall be eliminated altogether, as this gives a matrix with lower density, which allows a faster processing using sparse matrix procedures. If desired, a re-optimization can be performed using this restricted set of coefficients.

After the linkage matrix **C** has been computed, we can obtain the response state **u** as a function of a particular feature vector **a** as

$$\mathbf{u} = \mathbf{C}\mathbf{a} \tag{27}$$

9.1 Multiple Response State Variables

So far we have only discussed the situation for a single response or state variable u. We will normally have a number of state variables $u, v, w, t, \ldots$. As we change the value of the additional variable v, however, the set of features which is involved for a particular value of u will vary, and we can suspect that different models would be required. This is true in general, but if feature vectors exhibit a sufficiently high degree of locality, the simple model structure proposed will still work. In such a case, the solution for three response variables in matrix terms can be expressed as

$$\begin{cases} \mathbf{U} = \mathbf{C}^u \mathbf{A} \\ \mathbf{V} = \mathbf{C}^v \mathbf{A}\mathbf{W} = \mathbf{C}^w \mathbf{A} \end{cases} \tag{28}$$

The reason why this works is again the extreme degree of locality of feature components. This means that as v varies, new feature components move into the mapping and old components move out transparently for the single model available. Due to the beauty of the channel representation, this means that channels which are not active will not disturb the matching process for those who are active.

10 Applications of Associative Structure

As the purpose of this paper is to give a description of the principles of the associative channel structure, we will in this context only give a few comments on results from applications.

The structure has with great success been used to estimate various properties in an image, ranging from description of line segments to structures containing corners or in general curvature. There are various approaches which can be used.

The structure has also been used for a view-centered object description procedure, which is able to recognize the object car from several different angles and also give an estimate of the view angle.

As in any other descriptive system, the mapping of certain properties are invariant, while others are not. In the associative procedure, the system will detect such properties by itself, or it can be guided in the choice of such properties.

11 Concluding Remarks

Learning in any robot system or biological system does not take place in parallel over a field of features and responses. Learning takes place along one-dimensional trajectories in a response state space. The reason for this is that a system, like a human or a robot, can only be at "one place at a time". As it moves from one place to another, which really implies from one state to another, it can only do so continuously due to its mass and limited power resources. Consequently, the system will move along a one-dimensional, continuous trajectory in a multidimensional space. This continuity is one of the few hard facts about its world, that the system has to its disposal to bring order into its perception of it, and it has to make the best possible use of it.

Acknowledgements

The author wants to acknowledge the financial support of the Swedish National Board of Technical Development, as well as of WITAS: The Wallenberg Laboratory for Information Technology and Autonomous Systems. Credits go to several people in the staff of the Computer Vision Laboratory of Linköping University, for discussions around ideas. Special thanks go to Per-Erik Forssén for his development of a MATLAB channel computation toolbox, something which has been invaluable in the development and tests of implementations.

References

1. M. Adlers. *Topics in Sparse Least Squares Problems.* PhD thesis, Linköping University, Linköping, Sweden, Dept. of Mathematics, 2000. Dissertation No. 634.
2. D. H. Ballard. Animate vision. Technical Report 329, Computer Science Department, University of Rochester, Feb. 1990.

3. M. F. Bear, B. W. Connors, and M. A. Paradiso. *Neuroscience. Exploring the Brain.* Williams & Wilkins, Baltimore, USA, 1996. ISBN 0–683–00488–3.
4. G. H. Granlund. The complexity of vision. *Signal Processing*, 74(1):101–126, April 1999. Invited paper.
5. G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision.* Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.
6. Gösta Granlund. Does Vision Inevitably Have to be Active? In *Proceedings of SCIA99, Scandinavian Conference on Image Analysis*, Kangerlussuaq, Greenland, June 7–11 1999. Also as Technical Report LiTH-ISY-R-2247.
7. S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company, 1994.
8. I. P. Howard and B. J. Rogers. *Binocular Vision and Stereopsis.* Number 29 in Oxford Psychology Series. Oxford University Press, New York, 1995. ISBN 0–19–508476–4.
9. K. Nordberg, G. Granlund, and H. Knutsson. Representation and Learning of Invariance. In *ICIP*, Austin, Texas, November 1994. IEEE.