Perception as a Dynamic Activation of Relational Matrices

The Associative Model, Applied Situated Cognition and Action-based Perception

Steve Olivecrona[1] and Dirk Derom[2]

[1] Massey University, New Zealand

[2] School of Psychology, Victoria University of Wellington, New Zealand

Abstract

The storage of data within the live sciences is a challenging and highly complex task, which has been steadily gaining attention in recent literature. Various solutions have been proposed ranging from ontology design to metadata recommendations over standardization of database design. Though valid in their effort to provide the life sciences with well-structured data warehouses, their focal point is to categorize and structure the data in a rigid, often category-tree way. Due to their fairly static characteristics, their storage becomes less dynamic than real live systems tend to be, inducing a gap between the system and their representation in these warehouses. Here we present an experimental model to be applied to the storage and retrieval of information based on an associative information system's sensory and motor state change data, aiming to represent the dynamics of a dynamic perceptual system. The model and database implementation use a universal information storage structure holding both data and metadata within the same structure. This model is characterized by the emphasis on associative information about the represented system derived from raw data, which are in their turn produced by the associative system's interactions with the environment. Instead of defining objects using descriptive relations, this model stores relations between occurents where the represented system is not replicated in its various components, but defined by its relations when they occur. This model therefore represents *the dynamics and interaction of systems such as human perception, rather than* imposing *artificial boundaries and qualities. In essence, the model is an alternative to perceptual knowledge accumulation, which, as we show, can be applied to a database design.*

Introduction

Recent developments within neuroscience have shed light on how perception could be encoded in the organism. Using various recording methods, neuroscience has been able to locate key brain areas processing stimuli received by the medium, be it human, robot or non-human organisms. Mental representational structures, as bearers of information, are widely accepted as being engaged in procedural computation enabling continuously growing, changing and transforming procedures and representations leading to the expression of appropriate behaviour of the agent involved. This so called 'Representational Theory of Mind' (RTM), whether in its strong or weak form, is the dominant theory in the neurosciences used to model functional brain maps and explain human behaviour in terms of the activity of brain area networks. Though questions remain unanswered regarding the difference between effective and functional connectivity [1-3], the working hypothesis defines information and the processing of

information as a representational-based (often modular) processing of specific inputs thus producing meaningful behaviour. Human agency is here described in terms of an internal brain process responding to external situations. Strong mechanistic claims have been proposed explaining behavioural mechanisms in terms of its components, relata and operations [4-6] defining brain processes and behaviour in terms of quantifiable internal operators relying large body of evidence within the neurosciences. Context in this approach is considered to be a measurable yet complex set of elements within the environment.

Interestingly, recent experiments [7, 8] show a more tight connection between the environment and the agent, where interaction accounts for more than an appropriate response of a particular system in a specific context. These data-driven contextual experiments could hint to the pivotal claim of e.g. situated cognition or extended mind, namely the claim that behaviour is the expression of fundamentally intertwined agents and their environment. In these views, the distinction between agent and environment is rejected and replaced by a mutual dependency with little or no need for internal representations of either objects or contexts as the driving element of human agency. More specifically, situated cognition refutes the idea of conceptual knowledge, and claims that knowing happens 'on the fly' being directly and constantly generated relatively to context, culture and language. Accumulating knowledge is then improving the incorporation of context, culture and language as opposed to accumulating representational concepts. The refutation of category-based behaviour as proposed by Gibson [9] and its replacement by a probabilistic interaction (cf. behaviour that is most likely to occur) with the environment consequentially opposes empiricism and neuroscientific experimental designs proposing universal loci of particular functional mechanisms.

The translation of the situated cognition into an operational model has yet to be established, which inhibits applying situated cognition onto worldly situations. At the heart of this hiatus lies the development of a method storing information dynamically without a priori classification of shapes, elements of shapes or events. It must be clear that it would be insufficient to develop a system based upon hard-coded patterns (of either shapes, events or categories), since this would at maximum introduce merely a modification of empirically grounded theories. Developing a situated cognitive system requires a different architecture where dynamic and interactive cognition is constructed solely on low-level perceptual and relational inputs. In what follows we present the Associative Model (AM), which aims of initiating the core architecture of such a relational model and consequentially address some of the issues as described above.

The model presented here must be considered a hypothesis as to how an accumulation of knowledge might be produced without the need for hard-coded patterns or shapes. This model uses only relational entities for it to derive information and does not (directly) store any object or part of an object for it to achieve conceptual knowledge. In its simplest form, the AM is a basic pattern recognition tool, with the implementation of a non-object based framework. We have thus developed a model able to derive categories using a purely relational storage of perceptual stimuli. In what follows, we describe the concept behind the AM using simplified examples clarifying the framework we have developed. This framework will then be applied to an experimental build of the AM. The use case presented here is a visual pattern recognition task using simplified facial expressions, showing the practicality of the approach.

From an Object- to a Association-Based Model

Given that that the AM does not require prior knowledge of any object or its characteristics, yet is able to categorize objects solely based upon stored and

processed relations, the model can be considered as an embodied model using interaction rather than prior categorization. For this to become apparent, we need to clearly describe our conceptual framework. In what follows, we will outline the basic principles and concepts of the model using three simple examples. These concepts form the backbone of the application, which will be described further below.

Example I: elements, relations, clusters and change

The core concepts of the AM are what we denote as elements, relations, clusters, states of clusters and change. Before explaining these concepts, consider the following example: a single point moves in a repetitive and consistent manner across a grid (see figure 1). Moving 1 square at the time, the black point moves to the right, down, to the left and finally up again landing in its original position.

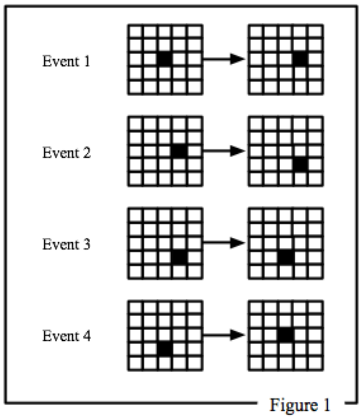

Figure 1

In this simple configuration, there are two elements: the grid and the point. At any given time, a particular relationship between these elements is in place (e.g. the point is at position '1' relative in the grid), in this case defined by the position of the point on the grid. Each of these snapshots ('the black point is at position 1') is what we call states (of a cluster), where the term 'cluster' (of relations) refers to the specific configuration of relationships between various elements (in this case defined solely by position). By changing the position of the point, that state (of a cluster) changes and a new cluster is formed ('cluster 2' in figure 1). The AM is thus defined by the elements (point and grid), their relations - forming clusters and state of clusters - and the possibility of *change*, which is the transition from one state to another.

Repetitive actions (cf. multiple sequences of figure 1) within the AM are then the motor of stable relationships. A continuous repetition of the various states in the example (e.g. a continuous sequence of cluster 1 to cluster 4) generates a stable cluster of relations. This is related to the temporal aspect of the AM, where we make a distinction between sequential and simultaneous states. If 'cluster 2' is always preceded by 'cluster 1', cluster 3 by cluster 2… the AM will consider these relations as a cluster of relations in itself, leading to stable states with high probability and predictability. The AM thus records either sequential or simultaneous changes of states, which are defined by the relations between the elements.

The AM is not recording the actual object, its boundaries, or any other meaning that is not introduced by a change in state. In the example provided here, the AM will not, for example, store the colour of the point if the colour does not change (undergoes a change of state). Given that the colour does not change e.g. from black to blue, no change has been detected and no record is made of the change, leading to no 'knowledge' about the change. Nor will it record how wide the lines are in the grid, nor how thick they are if they do not change either sequentially or simultaneously. In that sense, naivety is a core feature of the AM, being blind to consistency. The more consistent a particular feature is (e.g. the point that never changes its colour), the less relevant it becomes for the AM. Prior to knowledge in the AM is difference and change, where colour for example can only be used as a state if it is contrasted with a change of colour.

This does not mean that the AM is colour-blind. The system is equipped with a basic set of change detection, where low-level

features are recognized and differentiated. However, if occurring change does not stimulate these potential differentiators (e.g. colour), they will not evoke the perception of colour nor develop a concept of colour. This means that a deprivation of stimuli (e.g. change) of either one of these differentiators will make them less relevant for the given AM. In the case of our example, though colour can be detected, it is nevertheless not relevant to the system since there is no change in colour (apart from black and white). And since change is a transformation of one state into another, relations become the motor of knowledge production within the system.

Example II: association versus cause and effect

Building upon the previous example, consider the following: a system detects two states (of clusters), where change consists of both a change of position of the point to the right and a simultaneous rise of the system's internal temperature (cf. cluster 1 to cluster 2, adding a temperature variable to the system). Suppose a change of the position of the point to the left is also detected to consistently coincide with a simultaneous decline of temperature. As previously mentioned, a continuous repetition of these changes will build a stable cluster of relations and define the various states as well as their sequential and simultaneous temporal features. The system here will detect a strong relation between the sequence of positions of the point (having a constant repetition of cluster 1 and cluster 2), an equally strong relation of a sequential relation between the rise/decline of temperature and the simultaneous sequence of a change in position and a change in temperature.

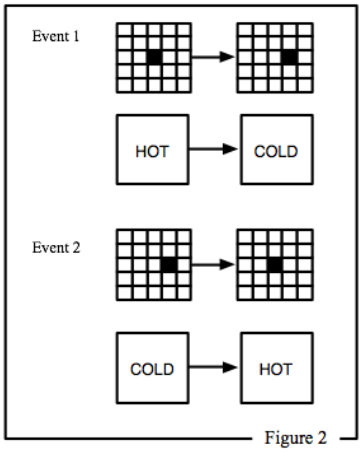

Figure 2

Within the AM this is the origin of causality. Causality is a stable cluster of relations, be it sequential or simultaneous, leading to an expectation about the occurrence of the various states. In the above example causality would be inferred when position was stable prior to any change in temperature. Assume that the system first learns the spatial change and once these clusters are well established detects a change in temperature, then as far as the system is concerned the position might be causing the temperature change. Such a hypothesis can be discarded when the opposite is shown and the temperature rises even when there is no occurrence of a spatial change. In that case, the relation between temperature and position is discontinued and causality is dropped. Falsification of change, change detection and temporal association is the motor behind what often is referred to as causality. Cause and effect can thus be expressed in change and change detection through the relations between the detected elements.

We have to be cautious, however, since for the system causality does not exist. Causality is the conceptualisation of stable clusters and robust states. The system does not deduce causality, nor does it introduce effects and origins. The AM only stores relations and change between relations as its motor for categorizations. Causality and effect being frozen concepts do no justice to an inherently dynamic model of association storage. The above examples are simplified to express the concepts of the AM, whereas terminology such as

cause and effect already introduce terms that can not be part of the system's core strategy, which is the storage of change. Or to put it differently: the AM does not require concepts such as cause and effect to expect effects, origins and causes.

Example III: associative predictability

Where the previous two examples were occurrences of singular elements, the same can be applied to patterns of elements. Take the example as shown in figure 3, where the triangle of points moves 1 square to the right in cluster 1 and back to its original position in cluster 2. Given that this is a continuous repetition of these states, the relation between the two states becomes stable.

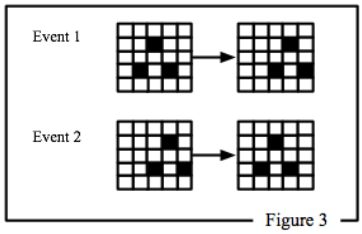

Figure 3

In this example, it should be obvious that the system does not require a concept of a 'triangle' to predict that the points will move to a certain direction. The system does not need to know whether the specific configuration of these points resembles what is understood as the term 'triangle'. The system will not even have a concept of a triangle if it is not confronted with a different configuration of points. However, it can 'behave' as if it has knowledge about a triangle, by using the specific relations between the elements to predict that the next state will indeed be a triangle, yet this time slightly more to the right or left relative to the grid. The system will thus apply the configuration as a stable set of relations, and use the triangular configuration as a way of predicting what the next position of the various elements will be.

With the point as an atom of the system's change detection, it will gradually and after sufficient repetition learn that the specific configuration is always present, thus forming a cluster of spatial associations. If the system has a highly stable spatial configuration through a consistent simultaneous appearance of these three points, in combination with a sequential change detection of a change in position that does not disrupt the internal spatial configuration, expectancy will be orientated towards more stable occurrences. Consider the following variation of our triangular example: a system detects over a long period of time cluster 1 and cluster 2. Suddenly, however, cluster 1 and cluster 2 do not occur any longer, yet cluster 1' and cluster 2' occur (figure 4).

With experience of previously highly stable clusters 1 and 2, the system expects that, based upon the information it has in cluster 1', cluster 2' will be 'triangular'. Not because it has recognized that there seems to be a triangle, but solely because it has learned that in the system's world, there used to be always three points spatially organized in a specific way which was stable over time (cf. simultaneous states). Unless there is evidence of a changed world (with e.g. squares), the system does not expect something other than a triangular spatial organization. Despite a change in perception (e.g. a missing point in particular states), its prediction therefore stays intact, or better unchanged.

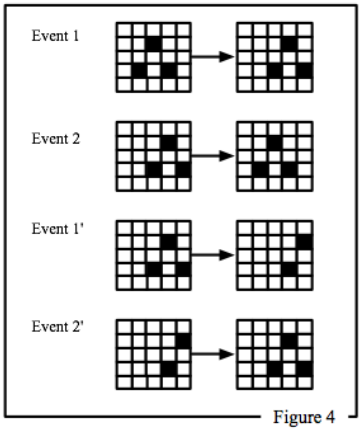

Figure 4

Thus, an AM uses association as a measure of stability and probability. Unknown situations are, despite their novelty, integrated, weighted and gradually engrained in the system's predictions until they form equally or

more stable associations, replacing previously detected associations. In other words: when the system is confronted with unexpected situations, it is able to detect novelty, incorporate the novel information and accumulate and rebuild previous stable clusters.

Rationale of the Associative Model

The associative model, as developed in the application, is a database design storing a minimum of information in order to derive or conceptualise information. This database design differs significantly from others, since it does not require any categorisation, nor connected tables for the structure of its stored data. The idea behind the design is to store the actual dynamism of learning processes as can be demonstrated in visual processing. The goal here is to store the accumulative process of visual perception, not by imposing a priori structures and objects used to reverse engineer data processing, but to let the system design its concepts based upon interaction with the environment. In other words, the database design aims to store the dynamic process of the interaction between the system (e.g. a person) and its environment (e.g. a working environment).

In the previous section we described the core concepts. For the sake of simplicity, the application and consequences of the model were left out. In the following paragraphs, we will describe the principles behind the design of the AM, where we focus more on what the AM stores (and what it does not), leading to a grounded categorization of stimuli through change detection.

Relations are, objects are not

The AM therefore is the accumulation of relational structures between various elements. A subject perceiving a table is not the perception of the edges of the table, but the perception of the relation of those edges. The AM can be summarized as a relational accumulative storage of transforming relations between elements, where the change or consolidation of related elements is the sole source of accuracy. These elements are not stored as a label, but as a 'meaningless' element having 'a' relation with another meaningless element. A 'cup of tea' in the AM is not stored as 'a cup of tea', but as an element that has consolidated relations with other elements, which you can label as being 'cup', 'hot water', 'herbs' - which are in their turn complexes of elements and relations. In the AM the actual element is nothing but the cluster of relations as seen by the system. A 'cup of tea' therefore is the accumulation of storage occurrences in which the subject has noticed that the cup holds water, that it can be hot, that it might require some tea leaves, that it smelt like tea and so on. For now, we don't go into detail on how such is applied in a database design, yet will describe what we consider 'elements' and how they are active agents in the construction of perception.

The concept 'element' is an artificial term to denote the atom of the AM, though it's never perceivable as such. An element in the AM is never a singular, since it is always a collection of other elements and their relations. This means that an element (e.g. a human being) can either perceive (e.g. 'see a dog running towards him') another element (e.g. 'a dog'). The past (e.g. 'I was bitten by a mad dog') and present ('I see a dog running towards me') relations between the first element ('the human being') and the second element ('the dog'), can be simply perceived (e.g. 'I see a dog running towards me'), but also acted upon (e.g. 'I run away because I'm afraid of what my prior experience has led me to assume is a mad dog'). As one can see, the emphasis here lies on the relational structure storing perception and enabling action. The dynamics of perception in the previous example are not represented by an accurate classification of the 'object' (cf. secondary element), but by past and present relations between elements. This is significantly different from static perception, where classification is in essence a rigid structure with possible updates and revisions of a dominant class structure. The associative model does not implement events to

update relations, but assumes events to be clusters of relations integrating past and present relations between elements and are therefore always the (whether or not appropriate or accurate) integration of relations between elements given an ongoing stream of occurrences. Elements are therefore both atoms of the model as well as collections of other elements and relations.

Bottom-up Storage of Associations

An associative model records occurrences where a specific element is perceived in relation with a second element. An example would be the occurrence of a needle penetrating your finger and seeing a drop of blood on your finger. These two elements ('needle penetrating your finger' and 'seeing a drop of blood') are stored in a relation to one another ('when I saw the needle penetrate, I saw the drop of blood'). Note that the system does not categorize the elements (cf. this is a finger, which is part of my hand, which is part of my arm…). It only stores the occurrence of two elements. This means that the model in its initial empty shell 'knows' nothing, has stored nothing and can only record all the occurrences it encounters as time passes. Tightly interconnected elements then provide the abstract schema allowing the categorisation of novel elements or the derivation of common features among aggregated elements. This aggregation is the storage of change of the various elements, where change refers to a previously defined threshold of the system. When the threshold is reached, the change is stored and a subset of elements is created. Such a threshold can vary and serves as a guiding principle of the system, yet can be a low-level feature of the element. Examples could be a colour change in a bitmap or a luminosity difference in a bitmap.

The factual accumulation of relations between elements enables the potential and growing conceptual power of the model, where consistent confirmation of relations become stronger and contradiction of previously recorded relations become weaker. Do note that we deliberately do not use concepts such as 'human', 'table', 'cup of tea' to represent the elements, neither do we use meaningful descriptions for the relations (e.g. 'has_a', 'stands_before') to describe the relations. All of these elements and relations are information empty and, as previously noted, they can only become meaningful after a sufficient accumulation of elements and relations. In other words, the simplified end-result after occurrence 4 can therefore be a wide variety of concepts, from which without any further input nothing can be derived. As must be obvious, the relational structure is the source of information, yet it does not store any readable information.

As laid out above, the example situation already induces concepts ('a person' or even 'walking around'), which are not accepted in the AM. The AM in its purest form has no concepts in its initial phase, and consequently does not use such concepts to store data. However, the example given above facilitates a ready, if superficial, grasp of what the AM actually stores, despite the shortcomings and assumptions such an approach introduces. We would therefore like to distinguish the model from object-subject systems (or perceiver-perceived learning systems), where objects are considered to be distinct and separable, though potentially connected and dependent on other objects. Consider a 'subject' (e.g. a person) perceiving an 'object' (e.g. a table). The distinction between subject and object are often delineated through boundaries, salience differences, colour… Subjects are those objects that perceive other objects and are able to classify the perceived and interacting objects. This distinction between subject and object is rejected in the associative model, for the associative model can be defined as the storage of dynamic systems represented by the relation between objects-subjects, the physical boundaries of the system and the environment in which the system resides. Elements are defined not by their (physical or deduced) edges but by their relationship with other elements as they occur in events. Systems are then understood as being constituted by their cluster of

relations between elements as interacted with in the environment.

Invariance is repetitive Variance

Within the AM, invariance (or stability of clusters) is achieved through the detection of variance where specific states are either simultaneously or sequentially repeated over time. With variance being the change of a specific element, the system only stores *changes* of these elements, be it position, colour, size… Stability of perception then becomes repetitive perception of sequential or simultaneous states (of clusters), where invariance or stability is always relative and possibly subject to change. Invariance therefore is never final, assumed, discovered or realized. It's an ongoing dynamic state of clusters in itself, where its stability is only guaranteed by the repetitive nature of its sub-clusters. Invariance within the AM is thus a special kind of variance, leading to redefinitions of categorization within the system.

This implies that non-changing elements are not perceived and thus do not exist. These elements (e.g. the colour of the point in example 1) do not hold any relevant information, since they do not form any specific relation with other elements and are therefore ignored. The AM remains blind to factors that are not actively engaging the system's perception. Only perception and the change of perception provide useful chunks of information. Additionally, the AM refutes the existence of a priori stability, since stability and invariance are only achieved through the perceiver. Despite the possibility that a particular state can be initiated before being perceived, only the perceiver introduces stability and invariance when a consistent repetition of variances is stored within the system. The perceiver is the sole source of invariance definition, since there is no other information available regarding whether the invariance had been present before perception. Though various systems may be able to impose their invariance on others (e.g. exchange of information regarding invariant states and clusters), at the root of invariance still lies the initial variance. Invariance without variance is not perceived within the AM and in a sense delineates the boundaries in which the system moves.

Finally, invariance will 'counter' invariance if the perceived invariance in itself does not become as stable as the initial invariability. Invariability, or consistent variability, can only be broken down with a significant increase in changes in states, leading to the deconstruction of the invariance. Variability thus introduces invariability and has the tendency to neutralize itself through the construction of highly stable clusters. With an initial neutral state, the AM gradually produces biased systems, in which, after a sufficient amount of time, invariance is more present than variance. The AM therefore introduces a system moving towards stability through the use of clusters of variability and, though being built solely upon variability, over time the system attains such a level of stability that it is only impelled to reduce the number of stable clusters if confronted with radical and consistent changes in novel or already present states.

This stems directly from the claim that the system is relational and bottom-up, where only relations between elements are stored and object and prior features are left out. As previously described (cf. 'Relations are, objects are not'), gradual accumulation of clusters lead to a complex network of relations. Such a network is sufficient for the AM to produce categories of objects and actions.

Experimental Build: Simplified Facial Pattern Recognition

The following use case is an application of the previously described model interacting with a number of static images. This demonstration is intended to show how the AM grounds its information in relations and how it identifies similarities and differences between discrete experiences. The images used to derive similarities and differences based upon relational scans are created to be both similar and different

enough to allow the application to produce recognizable outputs after a relatively small number of scans. Sequences of states would be related differently in an embodied system to produce the equivalent of the event sequences in the use case.

Before describing the experimental application, we need to explain the built-in limitations of the application. Currently there is a 16-bitmap limit hard coded into the program. This could easily be changed if the queries were dynamically created at runtime. Additionally, the use case is a simulation of an intrinsically, largely parallel, processing system implemented on a serial processing machine, causing it to run slowly. Another limitation is the fact that event sequences are linked directly to filenames in the application. This is a simplification of the AM in that there ought to be an intermediate step corresponding to the unique and shared subsets discovered by the system. Finally, the application is currently also limited by the expedient linking of start points (bitmap canvas-relative coordinates) to the event sequences. This allows easy display of the visual output on a 128 x 128 bit bitmap but makes the system incapable of matching similar shapes situated in different positions relative to the bitmap canvas.

The System

The program is written in function-based Delphi code using Borland Developer Studio 2006 connecting to a Microsoft SQL Server relational database. A number of dimension values and types of relationships are loaded into the database on start up. These are used within the program to make the data readable to the user. A predefined workflow is implemented (figure 5), checking and storing the relational changes it detects during the scans. The entire workflow consists of two sub-workflows. The first sub-workflow (figure 5, step 1 to 3) is only used at the very beginning of the scans. Here the system starts scanning at a random point, where the sensor array (a 3x3 focus grid) moves in a random direction (figure 5, step 2). If the focus grid is told it has reached the edge of the base grid, it randomly changes direction and continues moving. Only when the focus grid detects a change in its 9xp pixel grid, it queries the database for an identical change detection (figure 5, step 3). If the sensor array is not present, it writes that particular sensor array state to the database. If the sensor array state is however present, it chooses another random direction.

After these three initial steps, the system then starts the second sub-workflow (figure 5, steps 4 to 8), with identical functions used during its scans. As with the initial start-up phase of the system, it goes through another cycle of moving and storing the sensor array (figure 5, steps 4 and 6). The system also stores (as opposed to the first sub-workflow) the movement made by the sensor array (figure 5, step 5). In case a specific scan sequence is then already present in the database (figure 5, step 7), that particular scan sequence is linked to the bitmap in which it occurred (figure 5, step 8). After these queries and array storages, this workflow (steps 4 to 8) are repeated until a total of 1000 scans is performed.

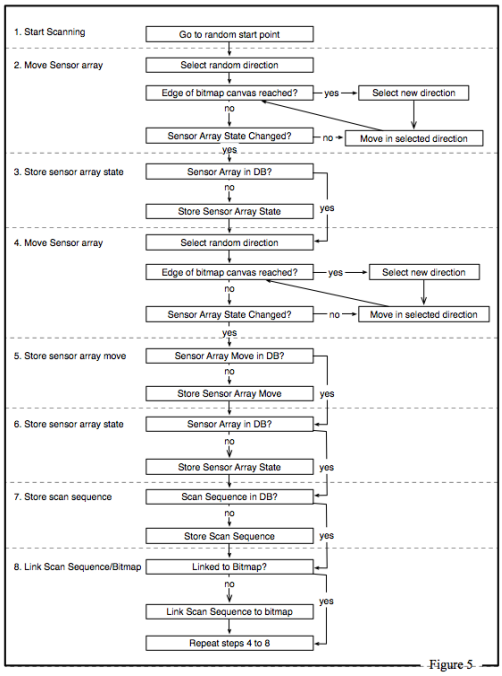

Figure 5

The above described workflow is essentially stored in 1 table, yet for

readability reasons two tables are implemented. The two tables implemented in the system are the Datum Table (tblID) and the Information Table (tblI). The datum table stores the explanatory text strings generated by the program in the v column, while the id column holds its associated unique identifier, the next unassigned integer in the table for each novel event or combination of events. The information table stores the elements and relationships discovered by the system while interacting with the bitmaps. The s column contains the element1 identifiers, the r column the relationship identifiers and the o column, the element identifiers.

All three columns are part of a concatenated primary key, making each record unique. These two tables are joined by three one-to-many relationships with the id from tblD linked to each of the s, r and o columns in tblI (see figure 6).

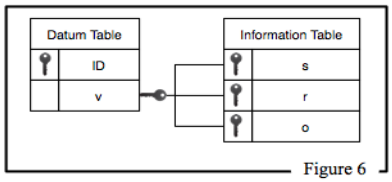

Figure 6

The datum table is created to make the data as stored in the information table readable. This is due to the fact that the information table solely stores numbers with no descriptive information attached to them, making it unreadable for the user. This is a logical consequence of the AM, storing only relations with no reference to previously labelled objects or object parts. An example can be found in figure 7, where the information table (figure 7 a) shows a selection of stored records of the use case.

Figure 7 (a, b) shows subsets of the data in the Information Table and the Datum Table from the use case has been reproduced. In the Datum Table the id column integers are tokens that are used in the Information Table in place of the text in the v column. At the base level in this example is the first record in the datum table, which has an id of 8 and a v of -. It is a direct and invariable representative used in place of an aspect of the structure of the use cases "environment". It occurs whenever a negative integer is input to or output from the use case to the computer that it runs on. In this virtual and limited version of the AM, the Datum Table becomes the interface between AM structured information and the programming environment.

The integer 8 is associated with the text character – when the program has a requirement for storage of negative polarity as a dimension of a complex informational entity. In this example u[-1], the table already had 7 records in it, hence the next available integer (8) was associated. Here, all information is defined through association with sets of base level interactions between the AM and its environment. The base level, simultaneous interactions that occur when the u[-1] vector is input or output, are the three aspects of a vector: dimension, magnitude and polarity (direction along its dimensional axis). These are assigned values which compromise the interaction, in this case u,1 and – respectively. Do note that the vector-based terminology was chosen for lack of a better alternative to make the tables more readable. The three aspects could just as validly be two aspects called *variable name* and *current value* and assigned the values u and -1.

The Datum Table furthermore contains records for the word polarity and for the vector descriptor u[-1], in our example the unique identifiers 14 and 24 respectively. These Datum Table records are not considered within the AM to hold any information. They are merely representative of interaction events. The unknown information that regulates the occurrence of base level interactions is stored out of reach in the inaccessible parts of the structure of the environment. What can be known within the AM is that certain interactions happened simultaneously and this information is recorded in the Information Table as a series of relationships. The Information Table in figure 7 contains a record that has 8 in the s column, 14 in the r column and 24 in the o column, one that has 3 in the s

column, 12 in the r column and 24 in the o column and one that has 27 in the s column, 13 in the r column and 24 in the o column. UPDATE TABLE 7. A loose interpretation of the information is that a polarity type of interaction occurred when the polarity sensor sent a – signal and it occurred in conjunction with a magnitude type interaction when the magnitude sensor sent a 1 signal and a dimension type interaction when the dimension sensor sent a u signal. The conjunction of these three interactions is recorded via the common o column value of 24 which is associated in the Datum Table with the text string u[-1].

Finally, the text string u[-1] has no intrinsic meaning in the AM, nor has the unique identifier 24, but the pattern of virtual connections represented by the Information Table records that the unique identifier 24 participates in has a direct correspondence to a unique combination of interactions; it, in effect, represents that pattern of occurrences.

(a) Information Table

s	r	o
8	14	24
24	15	36
3226	22	43
36	16	73
73	17	3225
3225	21	3226

(b) Datum Table

ID	v
8	-
14	polarity
15	coordinate
16	component
17	aspect
21	post-change state
22	event sequence
24	u[-1]
36	uv[-1,0]
43	{C:\FileBase\Faces\001.bmp}
73	{[-1,0]{255,255,255}}
3225	!State 2!
3226	at!ij[32,105]!!State 1!!xy[-1,0]!!State 2!

(c) sequentially linked structure

sv	rv	ov
!State 1!	!xy[-1,0]!	!State 2!

(d) Query example

sv	rv	ov
-	polarity	u[-1]
u	dimension	u[-1]
1	magnitude	u[-1]
u[-1]	coordinate	uv[-1,0]
uv[-1,0]	component	{[-1,0]{255,255,255}}
{[-1,0]{255,255,255}}	aspect	!State 2!
!xy[-1,0]!	change state	at!ij[32,105]!!State 1!!xy[-1,0]!!State 2!
!State 1!	pre-change state	at!ij[32,105]!!State 1!!xy[-1,0]!!State 2!
!ij[32,105]!	initial reference	at!ij[32,105]!!State 1!!xy[-1,0]!!State 2!
!State 2!	post-change state	at!ij[32,105]!!State 1!!xy[-1,0]!!State 2!
at!ij[32,105]!!State 1!!xy[-1,0]!!State 2!	event sequence	{C:\FileBase\Faces\001.bmp}

(e) variation from the pure AM

sv	rv	ov
!State 2!	post-change state	at!ij[32,105]!!State 1!!xy[-1,0]!!State 2!

Figure 7

Interface and Functions

The application's interface (figure 8) can be divided into roughly 4 sections: the graphic representation of the queries (figure 8 A), image list (figure 8 B), event set members (figure 8 C) and a display of the generated rows in the database (figure 8 D). The interpreted data display grids on the right hand side of the application window show stored information from the database hierarchically organised from top left to bottom right with lower level grids filtered to show only records associated with the selected record in the grid immediately above.

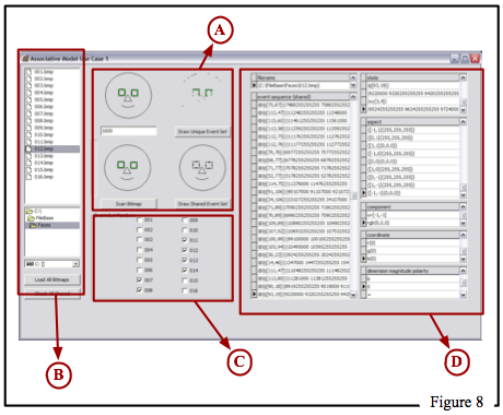

Figure 8

- *Scan Bitmap*

Clicking the Scan Bitmap (figure 8 A) button causes the program to start at a random point on the selected bitmap and to move in a constant, but initially randomly selected direction, comparing the red, green and blue values of each corresponding position in two 3x3 pixel sampling grids, one at the origin of the scan, the other at the current focus position until a difference in colour is detected. A difference in colour of any pixel is interpreted as being equivalent to a change in state of the sampling grid at its current position relative to its start position state. This first scan is not recorded in the system because the start point is not guaranteed to correspond to a point of difference in the bitmap. The next and all subsequent scans are made in random directions, each starting at the end point of its immediately previous scan. If a scan reaches an outer edge of the bitmap frame, it "bounces" back into the body of the bitmap.

The relationship between the start and end points of each scan is the difference between their coordinates. The system associates the change in position of the focus grid with the change from the grids initial state to that of its subsequent state. In the use case, the relationship or transformation is calculated, whereas in an embodied system the transformation information would be provided by changes in states of sensors, each monitoring a degree of freedom of, for instance, a camera mount moving relative to the body of the system. The components, coordinates, dimensions, magnitudes and polarities are stored to show that, even in this simplified, simulated system, information is firmly associatively grounded at the lowest level of interaction between the system and its environment.

- *Check All Shared*

Clicking the 'Check All Shared' button causes the system to look for each event sequence in the selected bitmap's unique event set in every other bitmap in the list of bitmaps available in the system. This is a series of directed scans. When a scan is repeated in a new bitmap, it is associated, or shared, with the new bitmap in addition to the original one. Next time the original bitmap's unique event set is queried any shared event sequences will not be included in the result set.

- *Draw Unique Event Set*

Clicking the Draw Unique Event Set button will filter the event sequence display grid to include only those records associated solely with the selected bitmap. The program will draw all those initial and subsequent states on a blank bitmap above the button.

- *Draw Shared Event Set*

Clicking the 'Draw Shared Event Set' button filters a query that then shows only those event sequences that are shared by the bitmaps selected in the checkbox panel on the application. The program will then draw all those initial and subsequent states on a blank bitmap above the button.

- *Use of Datum Table values to detect existing relations*

The program stores descriptive text in a database table named tblD to ascribe user interpretable meanings to events and relations. This is not a requirement of the AM, merely a convenient way to interpret the data. We have also used text descriptions to check the novelty of new input, to make the program easier to debug.

Stimuli

In this use case we used 16 simplified faces (figure 9), with differences in the colour of the 'nose' (e.g. a red nose of 010 and a green nose of 016), the colour of the eyes (e.g. red eyes for 004 and blue for 006), whether the faces were 'glasses' or not (e.g. face 007, 008, 013…) and whether the faces are smiling (e.g. 001, 004… but also 015), neutral (e.g. 002) or slightly sad (e.g. 003). The images are located in the exact same position within their 128-pixel square grid. For the sake of simplicity and process speed, the position relative to the grid remains constant despite the fact that this is not a requirement for the AM. If the position of the face varied the application would be forced to store additional relational elements, which is feasible but would slow the system down.

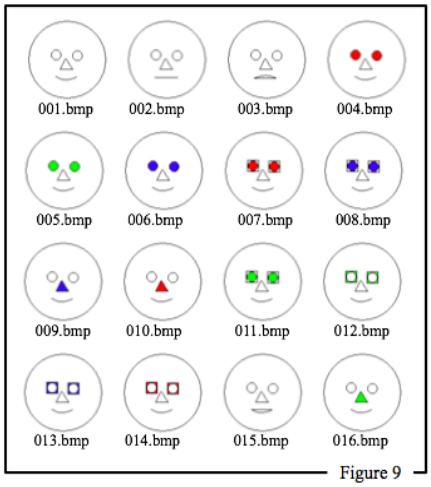

Figure 9

Scans and Results

Scanning a bitmap, against a fixed background grid, means to detect change

in colour between the starting point and the end point. A 3x3 pixel focus grid detects this change, where the 9-pixel square detects any change in any of the 9 pixels on the base grid. In figure 10 a, the yellow lines and the red dots represent change-detection. In this graphical representation of an archetype of a 1000-scan the red squares represent the change as detected by the application and the yellow lines the stored translations linking these changes. The yellow lines thus demonstrate the fact that the focus grid does not detect any change and thus moves on in the same direction until it detects a change (cf. figure 5). When a change is detected a new and randomly generated direction is chosen and continued until another change is detected. The process in this case is repeated 1000 times (see section 'The System').

Any change in the red, green or blue values (of the bitmap) in any of nine pixels sampled in the focus grid will trigger a change event. The system then records or finds the corresponding grid state record (see section 'The System' steps 4 to 8). As can be seen in figure 10 (a), 1000 random scans do not locate all aspects of a pattern, but sufficient information is stored to differentiate the scanned bitmap from other bitmaps.

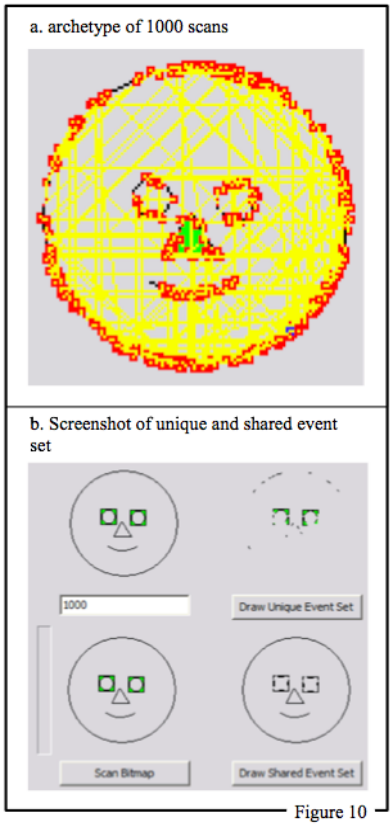

Figure 10

The current system detects and stores relations between changes in the patterns of colours (in a nine pixel reference grid moving in straight lines). The random direction of the focus grid movements relative to the base bitmap, as well as the changes, are stored, checked for repetition of those relations in new situations (other bitmaps) by means of data directed searches and then the program displays sets of commonly observed relational content for selected combinations of bitmaps. These could, according to the AM, theoretically be used as experientially grounded relational bitmap classifiers. Using the AM, the application does not store pictures or features as such, instead it stores sequences of system states determined by interactions between the system and each bitmap and then shares repeated experiences of sequence segments with other bitmaps.

Figure 10b shows the graphical translation of the scans and the relational storage as described in section 'The system'. The button 'Draw Shared Event Set' queries the system to report, based upon the relational matrix, what the differences are

in bitmap 012.bmp within the image group consisting of images 007.bmp, 008.bmp, 011.bmp, 012.bmp, 013.bmp and 014.bmp. The rationale behind this type of query is explained by the limitations of the current application. In a fully automated system, groups would be produced by the system itself. The application could e.g. query for shared elements and build groups by ranking them according to the amount of shared elements. In other words, a category would be created if a particular set of bitmaps has a high number of shared elements, which differ significantly from another collection of bitmaps sharing a high number of shared elements. Here, the use case runs a singular bitmap against a predefined collection of bitmaps. The collection of bitmaps are handpicked based upon obvious shared elements (in this case the fact that all wore glasses, had a nose, a smiling mouth and a contour), to show that the AM is able to deduce shared elements not based upon objects stored in the database, but solely based upon their relational and change-sensitive matrix (bottom right image of figure 10b).

The initially blank database is thus able to deduce shared elements after a total of 16000 scans and produce a highly detailed reproduction of what you could denote a 'categorization' of bitmap 012. The bitmap 012 is thus detached from its particularities, and by comparing the shared bitmaps across the collection of bitmaps the system produces its category description (here represented by a graphic of the category having glasses, a nose, a mouth and a contour). As said before, the hypothesis here was that these elements would be picked up, which is indeed the case.

The same applies for clicking the button 'Draw Unique Event Sets', where the query focuses on the particularities of bitmap 012 within the handpicked collection of bitmaps (cf. 007.bmp, 008.bmp, 011.bmp, 012.bmp, 013.bmp and 014.bmp). As opposed to the previous example, here we made sure that the bitmap had particularities that were not shared across the images, which in this case was the green glasses of bitmap 012. It is shown that this specific characteristic is indeed detected by the system, again based solely on the relation change-matrix. Do note that the unique elements, which are not detected (e.g. gaps as seen in the glasses), are to be attributed to the limited amount of scans of the system. With only 1000 scans for each bitmap, it is seen that not everything of the face is stored (figure 10a). Consequentially, when these changes are not stored, the system does not have any information about these undetected changes and they are thusly not represented as unique elements of the bitmap.

Do note that the produced unique and shared event sets are currently not stored within the database. Each time when the query is requested, the system does a fresh check and derives the shared and unique event sets. Though it is theoretically possible, and for an embodied system desirable, the current system does not store these creations. The generated categories and individuals are thus solely created to show the applicability of the system using the AM. Despite the fact that the system currently does not automatically produce categories, it is here clearly shown that it is able to do so, even within a limited set of scans starting from an empty database.

Conclusion

The current application as described above is able to deduce shared and unique elements based upon a single table in a database storing nothing but relational change-detection of low-level features (e.g. colour). This implies that the system does not require predefined geometrical shapes or any other information. It is capable of deducing categories on the basis of 16000 scans of 16 bitmaps and produce what we categorize as obvious similarities and differences. In other words, the application of the AM allows an emulation of our human visual categorization with limited application of an intrinsically very simple system.

Here, the AM shows its applicability of embodied and relational theories within a visual detection paradigm. The system further uses low-level change detection, which is an essential part of human perception. Though the system is limited in scope (fixed grid, identical pixel point detection and fixed bitmap positions), it shows its useability as a model for visual perception where no a priori knowledge is required.

Discussion

With the application we showed not only the theoretical but also the practical relevancy of the AM. The AM is conceived as a theoretical basis for a practical way to integrate all the disparate inputs and outputs required in an embodied system in such a way as to allow the system to store dynamic information about real-world interactions and use it to produce environment-appropriate behaviours. In the AM, content is relational and can exist independently from descriptive labelling. In fact, labels can only be introduced on top of the relational matrix. Content is thus an indirect consequence of relations, applied in the application above to visual perception. Once formed however, simply associating descriptive text strings entered by a user with the system-discovered features displayed would enable the system to be later queried by a user using text based descriptors in order to display a list of bitmaps sharing these named features.

Though the system as implemented in the experimental build is simplified to show its potential without becoming overly complicated (due to the exponential amount of stored data), we believe that it can be modified to record e.g. a parallel state that stores a Boolean value for each pixel in the grid based on identity of the colour of each grid pixel with the colour of the grid's centre pixel. Similar shapes of differing colours could then be correlated by the system as being the same in one way, but different in another. The finer-grained the shared information, the more refined the level of discernment of the system becomes.

Another potential application could be an embodied system requiring multifaceted sensory inputs (visual, auditory, tactile…) to correlate complex inputs. This clearly requires a more complex design storing highly complex data with a wide variety of relational grids stemming from multimodal stimuli. However, with an increased complexity in data storage a more refined relational matrix is constructed, leading to a more fine-grained output appropriate to the situation at hand. This could be correlated with e.g. behaviour where one integrates a very large body of stimuli, where repetitive related events build up towards an 'understanding' of the given situation.

Finally, an AM system need not be limited to a single agent. A server-based application could, in theory, wirelessly run multiple bodies simultaneously as long as each input from each sensor and each output to each actuator was unique within the AM system. Two such systems could also query each other using those same text strings. The two systems could then scan displays of each other's idiosyncratic version's content and thereby update their own definitions of the text string's agreed meaning. These two systems could also, theoretically, merge experiences if their base-level ID assignments were identical. The merging process would have to systematically identify functionally identical relational structures in the databases from the bottom up, then rationalise the IDs by globally replacing the various dissimilar but functionally identical IDs with single, common IDs.

The identity of a system therefore is not derived from the possible super-classes it belongs to, but from the relations it stores with elements as experienced in the environment. As such, identity within the associative model is not a subject, but a bundle of relations perceived or activated from a solipsistic point of view. In other words: the concept of 'subject' is the reference to the perceiver or activator relation bundle, where its uniqueness is

not the distinction between the particular element but its relations with the surrounding elements. This implies that the environment is not the collection of objects as perceived and interacted with by the subject, but the collection of relational elements as stored by the subject.

The AM therefore is the application of 'situated cognition', using the relational structure between perceiver and perceived as the core of understanding these interactions. In the AM there is thus no room for universals, but only for relational stable clusters. Subject, object, causality or universals are seen by the AM as semantic labels freezing content rather then freeing content. With the AM we thus present a possible model allowing for a continuously dynamic system effectively dissolving static terminology as strong clusters of variability. Or in the light of our use case: visual perception is action-based interaction.

Acknowledgements

We would like to thank Courtney Olivecrona for proof-reading the document and prof. Lorenzo Magnani for his most valued comments on the initial draft of this paper.

References

1. Ramnani, N., et al., *New approaches for exploring anatomical and functional connectivity in the human brain.* Biol Psychiatry, 2004. **56**(9): p. 613-9.
2. Fingelkurts, A.A. and S. Kahkonen, *Functional connectivity in the brain - is it an elusive concept?* Neuroscience and Biobehavioral Reviews, 2005. **28**(8): p. 827-836.
3. Lee, L., L.M. Harrison, and A. Mechelli, *A report of the functional connectivity workshop, Dusseldorf 2002.* Neuroimage, 2003. **19**(2): p. 457-465.
4. Bechtel, W., *The challenge of characterizing operations in the mechanisms underlying behavior.* Journal of the Experimental Analysis of Behavior, 2005. **84**(3): p. 313-325.
5. Bechtel, W., *Mental mechanisms : philosophical perspectives on cognitive neuroscience*. 2008, Hove: Psychology. xiii, 308 p.
6. Craver, C.F., *Explaining the brain : mechanisms and the mosaic unity of neuroscience*. 2007, Oxford: Clarendon. xx, 308 p.
7. Dotov, D.G., L. Nie, and A. Chemero, *A demonstration of the transition from ready-to-hand to unready-to-hand.* PLoS One, 2010. **5**(3): p. e9433.
8. Cardinali, L., et al., *Tool-use induces morphological updating of the body schema.* Curr Biol, 2009. **19**(12): p. R478-9.
9. Gibson, J.J., *The ecological approach to visual perception*. 1979, Dallas ; London: Houghton Mifflin. xv,332p.
10. Resnick, L.B., *Discourse, tools, and reasoning : essays on situated cognition*. 1997, Berlin ; London: Springer. 474p.
11. Robbins, P. and M. Aydede, *The Cambridge handbook of situated cognition*. 2009, Cambridge: Cambridge University Press. xi, 520 p.