Proceedings of the 2008 IEEE
International Conference on Robotics and Biomimetics
Bangkok, Thailand, February 21 - 26, 2009

Sparse Distributed Memory for Experience-Based Robot Manipulation*

Sascha Jockel, Felix Lindner and Jianwei Zhang
CINACS Int. Research Training Group & Technical Aspects of Multimodal Systems
Department of Informatics, University of Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany
{*jockel,2lindner,zhang*}*@informatik.uni-hamburg.de*

***Abstract*—Sparse distributed memory (SDM) is a mathematical technique based on the properties of high-dimensional space for storing and retrieving large binary patterns. This model has been proposed for cerebellar functions, and has been used in simple visual and linguistic applications to date. This paper presents an SDM for robotic applications, especially for storing and recognising mobile manipulation actions of a 6-DOF robot arm. Sequences of events are stored as subjective experiences and are later used to guide robot arm behaviour based on its memory content. Several simple manipulation tasks, such as lift and place a wastebin from and on the floor, push an object aside on a table-top, and draw shapes in the air are analysed under different operation modes. The robot system shows good reproduction abilities of task-dependent arm trajectories based on sparse distributed memory. Moreover, the content-addressable, associative memory even predicts the residual arm trajectory of a task if the arm is placed somewhere close to a learnt trajectory.**

***Index Terms*—Sparse distributed memory, cognition, service robotics, mobile manipulation, sensorimotor learning, associative memory, motion prediction.**

I. Introduction

If the human mind stores each item—e.g. abstract concept, attitude, *etc.*—in a single memory location, the retrieval of a past (subjective) experience from memory triggered by current sensings needs to be identical to those sensings of the memorised experience. This kind of organisation would be useless, since the space of all possible experiences is so vast that no two experiences are ever exactly identical. The human mind would have to possess more memory locations than the number of particles in the universe to store all the permutations of sound, colour, and so forth that the senses are capable of detecting [1]. The underlying principle of the brain involved in sensory information processing is that information is represented by a relatively small number of simultaneously active neurons out of a large population and is commonly referred to as *sparse coding* [2]. Cognitive models accomplish the above-mentioned principle by distributing the storage of an item across many memory locations [3]–[6]. Likewise, each location participates in the storage of many items. By constraining the memory distribution, the interferences among memory items that would occur in a fully distributed memory have been overcome. Content-addressability is another important aspect of memory that would be desirable for computer systems and robots in particular.

Sparse distributed memory (SDM) [6] is an interesting form of *associative memory*, popular in both computer science and psychology [7]. It represents both an interesting storage device and plausible mathematical model of human long-term memory [6], [8], [9] that meets all above-mentioned demands. Until now, research on SDM for robotics is rare. Since it shares characteristics with human long-term memory, it should be of special interest for far more researchers that are in pursuit of biologically inspired and intelligent robots. These characteristics are: it is content-addressable, storage locations are gradually removed, it degrades smoothly, information is widely distributed, it is massively parallel, it can handle noisy or corrupt data, it processes high-dimensional data, and each memory location encodes for a multiple stored data pattern. The few SDM investigations in the robotics domain mostly focus on camera-based scene recognition and acquisition for navigational issues [10]–[13]. At the time of preparing this paper, to the authors no work is known regarding robot arm manipulation based on sparse distributed memory. Many practical problems have to be solved.

In this paper, we propose a system that can predict manipulation sequences of a 6-DOF robot arm by a content-addressable memory. During a learning phase, the 6 joint angles of the robot arm, as well as the tool centre and orientation, are stored in the SDM along with some additional parameters. During subsequent autonomous executions, the robot is supposed to recognise its current joint constellation to navigate the robot arm driven by its memory of past experiences. This paper is a part of an ongoing investigation into the feasibility of using SDM for robot control and memory-based action prediction [14], [15].

The paper is structured as follows: Section II gives a brief review on Kanerva's SDM and related models. In section III, we outline different operation modes and the memory organisation of our implementation. We introduce the experimental platform for the SDM-based trajectory prediction, and present our results in section V. We conclude and discuss shortcomings of SDM-based robotics in section VI.

II. Sparse Distributed Memory

In this section, we review the SDM framework, introduce notions we will use later, and relate our work to the existing

*This work is funded by the DFG German Research Foundation (grant #1247) – International Research Training Group CINACS (Cross-modal Interactions in Natural and Artificial Cognitive Systems).

literature.

Kanerva has shown that widely accepted theoretical models on the cerebellum by Marr [16] and Albus [17] can both be represented by slightly modified SDM architectures [18]. The correspondence of the distance between concepts in human minds and the distance between points of a high-dimensional space, e.g. that any given point is relatively far from other points of interest, led to the principle idea of SDM.

The SDM associates two binary vectors $\vec{x}$ and $\vec{y}$ by projecting $\vec{x}$ into a very high-dimensional intermediate word-line vector $\vec{w}$, and then associating $\vec{x} \rightarrow \vec{w}$ and $\vec{w} \rightarrow \vec{y}$. Through its analogy with a conventional computer *random access memory*, the association of $\vec{x} \rightarrow \vec{w}$ is a kind of *address decoder memory*, and the association of $\vec{w} \rightarrow \vec{y}$ depicts the *data memory*. According to this, a write process consists of associating $\vec{x}$ to $\vec{w}$, and then associating $\vec{w}$ to $\vec{y}$. A read process is simply the presentation of a vector $\vec{x}$, resulting in a word-line vector $\vec{w}$, which is then used to read a value $\vec{y}$ out of the data memory.

The model is based on the crucial observation that if concepts or objects of interest are represented by high-dimensional vectors, they can benefit from the very favourable matching properties caused by the inherent tendency toward orthogonality in high-dimensional spaces. For example, consider the space $\{0,1\}^n$ for large n (with $n > 100$). If the Hamming distance is used as the distance metric between points in this space, then the number of points that are within a distance of D bits from an arbitrary point follows a binomial distribution which, for large n, can be approximated by the normal distribution with mean $\frac{n}{2}$ and standard deviation $\frac{\sqrt{n}}{2}$. In other words:

$$N(D) \simeq \Phi \frac{D - \frac{n}{2}}{\frac{n}{2}}, \tag{1}$$

where $\Phi(z)$ denotes the standard normal distribution function with zero mean and unit deviation. The number of different patterns that can be generated by a binary word of size $n, \rightarrow 2^n$ will form a n dimensional hypercube. Thus, an object of interest can be represented by a high-dimensional vector that can be subjected to considerable noise before it is confused with other objects. The same argument also applies to high-dimensional vectors whose components are non-binary [10], [19]. Whereupon other metrics than the Hamming distance may have to be applied respectively.

Kanerva's model is a type of neural *associative memory* consisting of a standard two-layer neural network design that could associate *data* patterns (weight of output units) with *address* patterns (weights of hidden units). The n-bit address, which correspond to a point in the memory space, and the data pattern together form a so-called *hard location* (HL). In contrast to related associative memory models, as the Willshaw network where the data is stored at all simultaneously active units, or the Hopfield networks where each data pattern is stored across the entire memory space, in SDM a data item is stored in many hard locations within a certain access radius related to a certain distance metric, e.g the Hamming distance. Thus, the memory is distributed in nature.

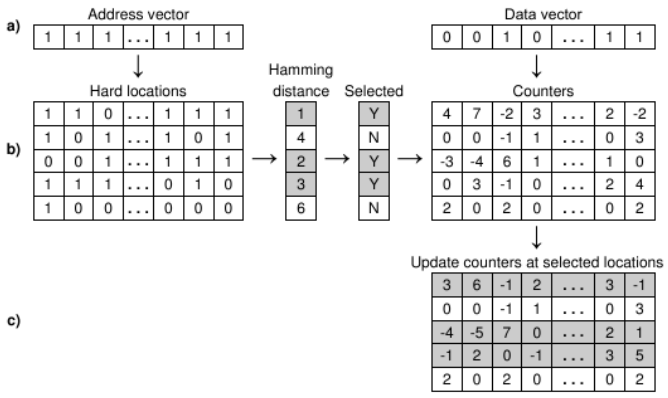

Fig. 1. Write a data-item into SDM: (a) the input address pattern is presented to the memory, (b) only locations with a Hamming distance $D \leq R$ are selected (shaded areas). The activation radius $R \leq 3$ was set *a priori*. (c) The data pattern is updated at the selected locations by adding 1 to each counter in the counter array corresponding to each 1, and subtracting 1 from each counter corresponding to each 0 in the pattern.

During storage of a data item within SDM, all hard locations will be activated whose Hamming distance is less or equal than the activation radius to a reference address. Writing to an SDM is detailed in Fig. 1. Since the address at which a data item is stored depends on the data itself, SDM is a *content-addressable memory*.

To summerise, sparse distributed memory shares many features with human long-term memory at the conceptual level: it is content-addressable, storage locations are gradually removed, it degrades smoothly, information is widely distributed, it is massively parallel, it can handle noisy or corrupt data, it processes high-dimensional data, and each memory location participates in multiple data encoding. Kanerva's idea of sparse distributed memory mainly focus on four concepts [6]:

1) The space of $\{0,1\}^n$, where $100 < n$, that corresponds to above-mentioned intuitive notions between concepts.
2) Neurons with n inputs as ideal address decoders.
3) The unifying principle—Data and address space are the same (*content-addressable memory*).
4) Time can be traced in the memory as a function of where the data is stored.

For further relevant literature on SDM and its improvements see Kanerva [6], [18], Jaeckel [20], Kristoferson [21], Sjödin [22], and Anwar [23].

A. World model and its update within SDM

To represent a world model in a sparse distributed memory, we need to represent the sensory information at a particular moment as a long vector of features and let a sequence of such vectors represent the passage of time. The flow of states at a particular moment is represeted by a sequence of states and therefore describes the individual's (subjective) experience. The simplest way to build such a world model is to store the report of the senses in memory and to retrieve from there later. Information supplied by the senses and information supplied by the memory can produce the same subjective experience. It is reasonable that some common part of the architecture is

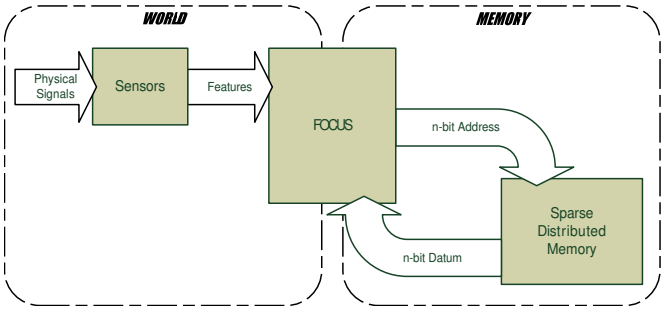

Fig. 2. Senses, memory and focus. Adaptet from [6]

responsible for the system's subjective experience about the world, and that both the senses and the memory feed into it. This is called the system's *focus*. A subjective experience is then represented by a sequence of patterns in the focus. By storing this sequence in memory, the memory can later recreate it in the focus [6]. This relation of the senses and the memory to the focus is shown in Fig. 2. Kanerva [6] further mentioned that in computer terms, the focus is a combined address-datum register. The memory is addressed by the focus, the contents are written into the memory, and the data from the memory feed into the focus. Through reassemble the past, the sensory data create a sequence in the focus that resembles a sequence. When this sensory sequence is used to address the memory, the memory responds with the same consequences as in the past.

III. Implementation

The bitwise implementation by Kanerva works with counters, and this is the reason for its limitations. Arrays of counters require a lot of unnecessary processing, which leads to an increase of processing time. Although the nature of the means (hardware counters and addresses) that have been used by Kanerva are not biological, SDM does not contradict the biological principles of brain functions, and at the same time it supports many properties and functions of the natural memory such as associativity, generalisation, abstraction, reproduction of a sequence, and even the ability to make mistakes [24, Ch. 1]. However, to reduce memory size and processing power, we replaced the bit counters of Kanerva's original model by a single bit to store one bit of information according to [8]. This implementation depicts the *bitwise operation mode*. Writing to memory in this model is done by replacing an old datum by the new one.

In the proposal at hand, we are using an SDM model with hebbian learning inspired by B. Ratitch and P. Precup [25]. The authors propose a *randomised reallocation algorithm* for dynamic on-line allocation and adjustment of memory resources for SDMs, which eliminates the need for choosing the memory size and structure *a priori*. Their dynamic allocation algorithm starts with an empty memory, and locations are added based on the observed data. New locations are allocated randomly in the neighbourhood of a location when there is a new datum which cannot be stored in the existing ones anymore. The authors assume that the activation radii of the memory locations are uniform and fixed by the user.

B. Ratitch uses the following strategy to organise the storage of the locations: each dimension i is partitioned into intervals $I_1^i, ..., I_m^i$ of length δ. For each interval, the sets t_j^i of locations with centre in $\{\vec{y} : \vec{y}_i \in I_j^i\}$ are maintained. On input $\vec{x}$, we find the set of intervals $\{I_{j_i}^i\}_i^n = 1$ such that for all dimensions, $\vec{x}_i \in I_{j_i}^i$.

$$P := \bigcap_{i=1}^{n} (t_{j_i-1}^i \cup t_{j_i}^i \cup t_{j_i+1}^i) \supseteq H_{\vec{x}} \tag{2}$$

The set P of potentially active locations is usually much smaller than the whole set of hard locations H_α, this yields a relatively more efficient way of finding $H_{\vec{x}}$ than testing all the elements of H_α [26, p.10]. The work of Ratitch and Precup [25] provide a mechanism that builds the set of hard locations in a way that "important" regions of the memory space are more densely covered by hard locations [26] then others.

The above-mentioned variation by Ratitch is used to implement an *arithmetic operation mode* to our SDM. Learning is achieved using reinforcement learning and addressing is achieved by using an arithmetic distance instead of a Hamming distance. The vector values are updated by applying:

$$\Delta h^k = \alpha(x^k - h^k), \quad \text{with } \alpha \in \mathbb{R} \wedge 0 \leq \alpha \leq 1, \tag{3}$$

where h^k denotes the k^{th} 64 bits of the hard location, x^k is the corresponding value in the input vector x and the learning rate is denoted with α.

Since the Hamming distance is not proportional to the arithmetic distance. The Hamming distance sometimes even decreases when the arithmetic distance increases, e.g. the binary numbers `0100` and `0011`. This is one of the limitations of SDM. Mendes *et al.* [13] propose some initial ideas to improve the encoding problem of memory features. We are aware of the encoding problem as one of the biggest problems while using neuron like architectures, but this problem is not part of this paper.

The third mode consists of an extension of the dimensionality. Due to the above-mentioned problems with the Hamming distance we express a number y within an interval $[a, b] = \{y \in \mathbb{Z} | a \leq y \leq b\}$ with the number of 1-bits according to $sc(y) = y - a$. This representation is known as *Sum-Code*, a derivate of *1-from-n Code*.

IV. Experimental Platform

TASER is a service robot of human height (see Fig. 3). In this work SDM learning is constrained to the 6-degree-of-freedom MHI PA10-6C robot arm trajectory for manipulation tasks. For further specifications of the used robot system, see [14], [15]. Manipulation is based on a sequence of 3-D coordinates and roll, pitch, and yaw angles of the manipulation tool. The inverse kinematics to route from one point to another is computed by the Robot Control C Library (RCCL). During a learning phase the 6 joint angles[1] of the robot arm as well

[1]Each joint has a software limited operation range up to $\pm 255°$.

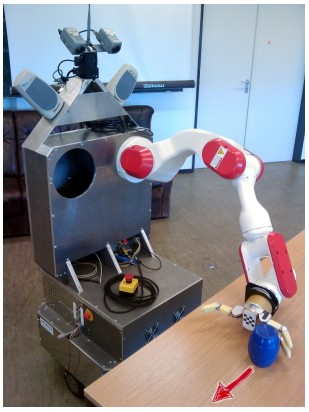

Fig. 3. The robot pushes an object aside (arrow). At first during this manipulation sequence, it reaches the hand closely above a table in front of it, then pushes a certain object aside by a lateral cartesian motion on a transversal plane namely the table and finally move the arm to a park position. The picture shows only the lateral transversal motion part of a much larger action sequence.

as the tool centre point (TCP) and the tool orientation are stored to the SDM with a sampling rate of 6-10Hz[2]. However, these different parameters that are expressing the same, are stored for redundancy. During an autonomous execution, the robot is supposed to recognise its current joint constellation and follows the learnt trajectory from its memory. The input and output vector consist of arrays of doubles as shown in Eq. 4.

$$\vec{x}_i =< j_1, j_2, j_3, j_4, j_5, j_6, x, y, z, \chi, \psi, \omega, seq\_id, i >, \quad (4)$$

where each j_n is the angle of the corresponding arm joint. The 3D coordinates of the tool centre mounted at the end of the robot arm in relation to the robot coordinate system are depicted with x, y, z, and χ, ψ, ω describe the roll, pitch, yaw tool orientation. Each of the above mentioned parameters are represented as 8-byte double values. Finally, the seq_id is a unique 4-byte id for each sequence, and i is unique for each vector.

Addressing the memory is done only by presenting a single arm position by its corresponding 6-joint angles. During an autonomous execution phase, the SDM will predict the j_n from corresponding j_{n-1}. The 6-joint-angles at a particular moment correspond to the arrow denoted "Sensory features" within Fig. 2. They are transmitted through the focus as an n-bit address to a datum within the sparse distributed memory—please note that content is used for addressing. Then $\vec{x}_i$ as in Eq. 4 is returned under consideration of further hard locations within the predefined activation radius to the focus as denoted by the arrow "n-bit datum" in Fig. 2. This process goes round and round and guides the prediction. More formally a prediction is a sequence of sub-predictions $pred_i$ such that:

$$pred_0^k = \vec{x}^{\,k}, \quad (5)$$

$$\mathtt{NB} = \{\overrightarrow{data}(hl) \mid dist(hl, pred_{i-1}) \leq r\}, \quad (6)$$

$$pred_i^k = \frac{1}{|\mathtt{NB}|} \sum_{j}^{|\mathtt{NB}|} \vec{y_j}^{\,k}, \vec{y_j} \in \mathtt{NB}, \quad (7)$$

with the neighbourhood NB and activation radius r.

[2]Due to problems in timing at sampling phase, the rate ranges from 6-10Hz.

V. Results

The biological inspired SDM model works fine for mobile manipulation prediction and is supported by some results. To verify the different approaches mentioned in section III, namely binary, arithmetic and bitwise Sum-Code mode, we performed several tests on the SDM with the robot platform mentioned in section IV. Two stages have to be passed by the memory module. First the memory learns an action sequence carried out with the robot arm. For details on how data was acquired, see section IV. The three investigated sequences—described in some more detail below—differ in their length. Secondly, during an autonomous stage the arm is placed at an initial position and the memory predicts the next position that has to be reach by the arm. Since the data was memorised at a rate of 6-10Hz during learning, each predicted movement is quite small. The results are summerise in Tab. I. Also we examined the systems behaviour on how changes in activation radius, and sampling rate influence learning. The particular values we invastigated and present in Tab. I are:

Operation mode We made experiments with three different representational modes:

- **Arithmetic mode** Values are represented in $\mathbb{R}^n$. The Euclidian distance is used to define a metric in order to calculate distances between hard locations.
- **Binary mode** The data is represented in natural binary manner. We employed the Hamming distance as a distance metric for this mode.
- **Sum-Code mode** We chose a binary representation in this mode as well, in a way described in section III.

Task The robot learnt three different tasks: A) reaching the robot hand closely above a table in front of the robot, and push a certain object aside by a lateral cartesian motion on a transversal plane, B) lifting and placing a wastebin from and on the floor, and C) drawing a U-like shape in the air.

Distance to closest hard location Describes the distance to the closest hard location to a given n-bit address (cf. Fig. 2).

Distance to 2^{nd} closest hard location Describes the distance to the second closest hard location.

Average distance to all hard locations The average distance is defined as follows:

$$avgDist(\vec{x}) = \frac{1}{|HL|} \sum_{j=0}^{|HL|} H(\vec{x}, \overrightarrow{address}(hl_j)) \quad (8)$$

Increment The increments in the distance between the closest and the 2^{nd} closest hard location are shown as well as from the closest hard location to the average distance of all hard locations.

Errors An error is defined as a non-intended prediction, e.g. a prediction of a state that is before or equal to the current state within the sequence. This would result in a flawed trajectory—the action sequence gets stuck or moves backwards.

Processing Time Average number of milliseconds needed to predict the next state within the sequence (e.g. next point of the trajectory in the 12-dimensional space).

TABLE I

THE DISTANCES TO THE CLOSEST AND 2^{nd}-CLOSEST HARD LOCATIONS (HL) ACCORDING TO THE OPERATION MODE AFTER 30 PREDICTIONS OF A LEARNT SEQUENCE. ALSO THE AVERAGE DISTANCE TO ALL REMAINING HARD LOCATIONS, THE USED ACTIVATION RADIUS AND THE SEQUENCE ERRORS ARE SHOWN. THE PERFORMED ACTIONS WERE A) REACHING THE ROBOT HAND CLOSELY ABOVE A TABLE IN FRONT OF THE ROBOT, AND PUSH A CERTAIN OBJECT ASIDE BY A LATERAL CARTESIAN MOTION ON A TRANSVERSAL PLANE, B) LIFTING AND PLACING A WASTEBIN FROM AND ON THE FLOOR, AND C) DRAWING A U-LIKE SHAPE IN THE AIR.

Task	Operation mode	Dist. to closest HL	Dist. to 2^{nd} closest HL	Inc. %	Aver. dist. to all HLs	Inc. %	Errors	Processing time (*ms*)
			Activation radius: $r = 10$					
A	Arithmetic	11.50	17.60	53.04	1142.67	9836.26	1	< 1
	Bitwise	1.23	4.67	278.38	58.89	4674.68	0	1
	Sum Code	8.37	8.37	0.00	1100.44	13052.69	0	41
B	Arithmetic	9.93	21.87	120.13	1293.37	12920.49	0	< 1
	Bitwise	2.90	7.77	167.82	62.65	2060.33	0	2
	Sum Code	7.87	7.87	0.00	1170.62	14780.74	0	50
C	Arithmetic	6.50	14.23	118.97	851.38	12998.15	0	< 1
	Bitwise	1.43	4.90	241.86	56.93	3871.58	0	2
	Sum Code	10.10	10.10	0.00	845.30	8269.32	0	21
			Activation radius: $r = 30$					
A	Arithmetic	24.97	32.30	29.37	1094.74	4282.80	11	< 1
	Bitwise	20.33	22.33	9.83	58.25	186.50	28	2
	Sum Code	16.77	16.77	0.00	1090.53	6404.14	0	34
B	Arithmetic	141.00	149.00	5.67	1841.13	1205.77	9	< 1
	Bitwise	18.53	24.10	30.04	60.47	226.26	22	2
	Sum Code	20.90	20.90	0.00	1144.38	5375.48	0	40
C	Arithmetic	13.97	21.13	51.31	898.46	6332.89	0	< 1
	Bitwise	21.87	24.30	11.13	53.65	145.36	9	2
	Sum Code	19.07	19.07	0.00	819.42	4197.64	0	19

Before any conclusions can be drawn from the results, one should note that due to the random characteristics of the SDM model, results cannot be reproduced absolutely exact. A deeper analysis of SDM behaviour is a non-trivial task. We found tendencies, however, that are worth to be discussed.

Activation radius—heavily influencing the degree of generalisation—turns out to be a crucial parameter when concerning the reliability of an SDM prediction. Since currently our SDM learns only one sequence at time, we are interested in the associative behaviour rather than the generalisation. As shown in Tab. I, a smaller activation radius leads to less error-prone predictions. Whereas prediction in both arithmetic mode and natural binary mode tend to get flawed when activation radius value is increased above 25, Sum-Code mode turns out to be very robust up to a value of 11000. This can be explained by the huge and redundant representation of the features.

Furthermore, the natural binary representation turns out to be the most error-prone among the tested ones. We explain this observation by the fact, that flipping bits (e.g. randomly during distribution while learning) can have vast impact on the values denoted by the representation. Also, the position of 1-bits within the bit-string—contrarily to the case of Sum-Code representation—carries important information. This might lead to wrong results during prediction phase where many hard locations are cumulated to a single result (cf. section II).

Variations on the learning rate did not have any remarkable impact on the prediction results. In our scenario this can, again, be explained by the fact that our SDM learnt only a single trajectory. We expect the learning rate to play a more important role in generalisation and forgetting when learning diverse tasks in one memory. The learning rate was set to 0.02 throughout all tests presented in this paper.

Another important parameter not listed in Tab. I is the memory size (e.g. the maximum number of HL the memory may contain). We chose a value of 10000, so that the trajectories learnt could entirely be represented by hard locations in the memory. Reducing the maximum number of hard locations results in worse associative ability and leads to faster forgetting. Future work will concentrate on how to deal with the tradeoff between association and generalisation.

Prediction time is mainly determined by search effort and metrical calculation. In the current implementation there is no search-space optimisation. For every prediction the algorithm has to check every hard location whether it is in the neighbourhood of the current sensory input in order to consider it in prediction output calculation or not. What can be seen from Tab. I, arithmetic mode and bitwise mode both have fast prediction times which is due to their compact representational form and the built-in operators for distance calculation (e.g. addition, potentiation and XOR, respectively). In Sum-Code mode, the address vector of a hard location is represented by 22200 bits. Concerning runtime, all our scenario modes produced results in an admissible time. It will be part of future work to investigate how SDM scales with an increasing amount of stored hard locations.

VI. CONCLUSION

An SDM that is capable of storing and retrieving robot arm trajectories for manipulation tasks has been presented. The analogy to the fact that the brain seems to *retrieve*

similar solutions to previous problems from memory rather than to compute new ones led to invastigations on mobile manipulation based on a plausible biological SDM model. The SDM model depends on subtle, nonintuitive properties of high-dimensional metric space and random distributed representations. The architecture was tested with three different manipulation actions executed with a 6-DOF robot arm mounted on the TAMS service robot TASER. The following action sequences were used: to push an object aside in a lateral-transversal manner on a table-top, to lift and place a wastebin from and on the floor, and to draw a U-like shape in the air. Three operation modes have been investigated, namely a bitwise mode based on a modification of Kanerva's original model according to [8] with Hamming distances, a memory-efficient arithmetic mode with a dynamic allocation algorithm of new hard locations according to [25], and a bitwise Sum-Code mode where a number $i \in N$ is represented by $n = N-1$ bits set to one, where N denotes the total amount of possible numbers within a certain interval.

The robot is able to learn and follow an arm motion trajectory. During learning, the associative memory stores joint angles and further parameters of a new and unknown manipulation task. In the retrieval phase, it compares joint angles from its current arm configuration to the closest matching one in its memory of past experiences. Since SDM is a content-addressable memory which corresponds to one of the most important human memory characteristics, the robot arm can be placed elsewhere in the manipulation sequence and will predict the next steps from its memory instead of getting lost.

The main limitations of SDM-based manipulation are recurring motion patterns and cross-sequences, e.g. beckoning to someone or drawing an "8". Especially as long as it only stores first-order (one-folded) transitions. To solve the problems of one-folded memory, we will extend our system to a *k-folded* memory which consists of a set of *j-step* memories with $j \in \{1, \ldots, k\}$. We will also use windowing functions to predict an item, e.g. E from ABCDE, based on composite sequences as ABCD or CD. A proper criterion for the length of a window has to develop, e.g. vigorous motion changes in specific joints or meta information. Moreover, we are going to extend our system—i.e. our vector—with further perceptions from laser range finders for improved mobility and depth images from stereo cameras for object and respective position detection, all for the purpose of biologically inspired memory prediction of robot actions.

The main contributions of this paper are as follows: This paper proposes the first implementation of a biologically inspired LTM model as an SDM in mobile robot manipulation. We have developed an experimental platform to investigate different SDM representation types in detail, and to test their usability in real-world service robotics domains. Finally, this paper exposes some challenges that arise when transferring theory to application, such as cross-sequences and, notably, the encoding problem. As we have outlined, our implementation constitutes significant first steps in SDM-based robot action learning and prediction. We see potential to extend our framework for application in more complex robot motion prediction.

REFERENCES

[1] L. Gabora. Cognitive mechanisms underlying the creative process. *4th Conf. on Creativity & Cognition*, pp. 126–133, NY, USA, 2002.

[2] B. A. Olshausen and D. J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14:481–487, 2004.

[3] G. Palm. On associative memory. *Biological Cybernetics*, 36(1):19–31, 1980.

[4] D. J. Willshaw. *Parallel Models of Associative Memory*, ch. Holography, associative memory, and inductive generalization, pp. 83–104. Lawrence Earlbaum Associates, 1981.

[5] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In D. E. Rumelhart, J. L. McClelland, et al. (ed.), *Parallel Distributed Processing: Vol. 1: Foundations*, pp. 77–109. MIT Press, Cambridge, 1987.

[6] P. Kanerva. *Sparse Distributed Memory*. MIT Press, Cambridge, MA, USA, 1988.

[7] P. Suppes. Representations and models in psychology. *Annual Review of Psychology*, 45:517–544, 1994.

[8] S. B. Furber, G. Brown, J. Bose, J. M. Cumpstey, P. Marshall, and J. L. Shapiro. Sparse distributed memory using rank-order neural codes. *IEEE Trans. on Neural Networks*, 18(3):648–659, 2007.

[9] P. J. Denning. Sparse distributed memory. *American Scientist*, 77:333–335, 1989.

[10] R. P. N. Rao and O. Fuentes. Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. *Machine Learning*, 31(1-3):87–113, 1998.

[11] M. Watanabe, M. Furukawa, and Y. Kakazu. Intelligent agv driving toward an autonomous decentralized manufacturing system. *Robotics and Computer-Integrated Manufacturing*, 17(1-2):57–64, 2001.

[12] M. Mendes, A. P. Coimbra, and M. Crisóstomo. AI and memory: Studies towards equipping a robot with a sparse distributed memory. *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, pp. 1743–1750, Sanya, China, 2007.

[13] M. Mendes, M. Crisóstomo, and A. P. Coimbra. Robot navigation using a sparse distributed memory. *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Pasadene, CA, USA, 2008.

[14] S. Jockel, D. Westhoff, and J. Zhang. EPIROME—A novel framework to investigate high-level episodic robot memory. *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, pp. 1075–1080, Sanya, China, 2007.

[15] S. Jockel, M. Weser, D. Westhoff, and J. Zhang. Towards an episodic memory for cognitive robots. *6^{th} Cognitive Robotics Workshop at 18^{th} Euro. Conf. on Artificial Intelligence (ECAI)*, 2008.

[16] D. Marr. A theory of cerebellar cortex. *Journal of Physiology*, 202:437–470, 1969.

[17] J. Albus. A theory of cerebellar function. *Mathematical Biosciences*, 10(1/2):25–61, 1971.

[18] P. Kanerva. Sparse distributed memory and related models. *Associative Neural Memories: Theory and Implementation*, pp. 50–76, New York, 1993. Oxford University Press.

[19] T. A. Hely. Sparse distributed memory. *Encyclopedia of Cognitive Science*, pp. 101–108, 2006. University of Edinburgh, Edinburgh, UK.

[20] L. A. Jaeckel. A class of designs for a sparse distributed memory. Technical Report RIACS TR 89.30, Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989.

[21] J. Kristoferson. Some results on activation and scaling of sparse distributed memory. Technical Report R97-04, 22, 1997.

[22] G. Sjödin. The Sparchunk code: a method to build higher-level structures in a sparsely encoded SDM. *IEEE Int. Joint Conf. on Neural Networks*, 2:1410–1415, 1998.

[23] A. Anwar, D. Dasgupta, and S. Franklin. Using genetic algorithms for sparse distributed memory initialization. *1999 Cong. on Evolutionary Computation (CEC)*, Vol. 2, pp. 1043–1050, Washington, DC, 1999.

[24] V. G. Tul'chinskii, I. N. Pshonkovskaya, and S. V. Zaytseva. Fast retraining of SDM. *Cybernetics and Systems Analisys*, 35(4):543–552, 1999.

[25] B. Ratitch and D. Precup. Sparse distributed memories for on-line value-based reinforcement learning. *LNCS*, 3201:347–358, 2004.

[26] A. Bouchard-Côté. Sparse distributed memories in a bounded metric state space: Some theoretical and empirical results. Technical Report, McGill University, 2004.