Bio-inspired techniques for pattern recognition

A DISSERTATION PRESENTED
BY
YATHINDU R. HETTIARACHCHIGE
TO
THE FACULTY OF INFORMATION TECHNOLOGY

MONASH UNIVERSITY
MELBOURNE, VICTORIA
APRIL 2019

SUPERVISED BY:
DR. ASAD KHAN
DR. SRINIVASAN BALA
DR. MUHAMMAD A. CHEEMA

©2014 – Yathindu R. Hettiarachchige
all rights reserved.

Bio-inspired techniques for pattern recognition

Abstract

Intelligence as it occurs in nature demands a level of energy efficiency, speed, accuracy, and generalisability that current algorithms are not readily capable of emulating independently. There are also application areas such as swarm robotics or wireless sensor networks that rely upon relatively simple computational units to carry out complex tasks. Such applications operate under similar constraints (energy efficiency, speed, accuracy and generalisability) to biological organisms and can thus benefit from the computational strategies employed by nature.

In biology the concept of sparse coding of sensory inputs has become popular with strong experimental evidence to verify it. However, there appears to be two interpretations of this concept in the literature. The more prominent interpretation is used in neural networks and involves representing information with a set of functions that are sparsely selected from a much larger set of possible functions. The other, less prominent interpretation, uses large high dimensional sparse vectors to represent information. One of the reasons the vector interpretation is less prominent is due to the difficulty of encoding raw inputs into large sparse high dimensional vectors.

This thesis addresses the problems outlined above by focusing on information from neuroscience and existing bio-inspired approaches. Particularly several approaches that use the vector based interpretation of sparse coding (the less prominent interpretation) were investigated. Additionally both interpretations of sparse coding were compared in a unique bio-inspired test bed. Through these investigations 4 major contributions are made:

First, the failure-case for a bio-inspired approach that uses sparse vector representations called Map Seeking Circuits (MSCs) was examined. Previous research and experimentation done in this thesis showed that accuracy of MSC is highly dependant on the sparsity of the input vectors. Through the examination of this failure case four encoding heuristics affect that accuracy where identified.

Second, a comparative study with a highly parallel bio-inspired pattern recognition approach that uses the vector based interpretation of sparse coding and a non-bio-inspired pattern recognition approach was carried out. This study showed that fine grain parallelism, parallelism resulting from processing sparse vector representations, allowed for gains in energy efficiency.

Third, an extended version of a vector based sparse coding approach called Sequential Hierarchical Graph Neuron (SHGN) is presented. SHGN is based on an approach known as Hierarchical Graph Neuron and makes performance improvements (improving efficiency from O(n) to O(1)) and enables the processing of temporal information. SHGN allows for much shorter training periods at the cost of a drop in accuracy. Making it ideal for use cases where the data rapidly changes and frequent retraining is necessary.

Finally, a comparison between the two interpretations of sparse coding was carried out via a unique bio-inspired test bed. An extended version of an existing neural network and the approach mentioned in the previous paragraph were used in the comparison. The test-bed was a tracking a swarm of simulated objects. The swarm used a behaviour seen in prey animals as predator confusion. Predator confusion has been shown to decrease the effectiveness of predators so it makes for a difficult computer vision task. In this difficult task the results showed that the neural network not only performed well but also maintained an average F1 score of 0.8 as the size of the swarm increased.

Contents

Listing of figures

List of Publications

Hettiarachchige, Y., Khan, A., & Barca, J. C. (2018). Multi-Object Tracking of Swarms with Active Target Avoidance. In 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV) (pp. 1204–1209). http://doi.org/10.1109/ICARCV.2018.8581176

Hettiarachchige, Y. R., Khan, A. I., & Barca, J. C. (2018). Improving energy consumption of pattern recognition by combining processor-centric and bio-inspired considerations. Biologically Inspired Cognitive Architectures, 23, 54–63. http://doi.org/10.1016/J.BICA.2018.01.004

Hettiarachchige, Y. R. (2018) SHGN source code (Version 1.0)[Source Code]. https://bitbucket.org/yathindu_rangana/shgn

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

.....
Yathindu Hettiarachchige
April 16, 2019

Acknowledgments

First of all, I'd like to thank my supervisors Dr. Asad Khan, Dr. Srinivasan Bala and Dr. Muhammad Cheema for their guidance throughout my candidature. I'd also like to thank Dr. Jan Carlo Barca for his guidance throughout my project. I'd like to thank my family for their love and support. And finally I'd like to thanks my office mates and friends at the Swarm Lab for the interesting discussions and fun times.

1

Introduction

1.1 Preamble

Small insects such as honey bees have been shown to be able to recognize human faces (Dyer et al., 2005) and even form abstract concepts (Avarguès-Weber et al., 2012, Giurfa et al., 2001) all using their miniature brains. Even other small relatively simple creatures such as molluscs show simple associative memory such as learning and avoiding unde-

sirable environments (Gelperin, 2013). Learning, recognizing and predicting patterns are evidently vital functions in animal intelligence. Merriam-Webster defines pattern as the natural or chance relative arrangement of parts or elements (Merriam-Webster.com, 2018). Therefore learning in the context of pattern recognition can be thought of as the assignment of identity to relative arrangements. Recognition, the correct association between a given arrangement and its corresponding identity. And prediction the association between arrangements. The act of learning in animals can be categorised into three types (Zentall et al., 2014):

- Similarity-based where stimuli are categorised based of some physical similarity.
- Relational, where stimuli are categorised with relative to other stimuli.
- Associative, where stimuli become interchangeable with one another due to association with other stimuli.

In computing there is a long history of taking inspirations from biology dating back as early as the 1940s (Turing, 1950, Neumann, 1966). Entire fields of study such as genetic algorithms which draw from biological evolution or emergent systems which draw from the behaviours of groups of organisms have emerged from researchers incorporating ideas from biology. Broadly the term "bio-inspired" is used to describe such approaches. However the term "bio-inspired" is imprecise in a subtle way: it fails to capture how contributions made can feedback into the study of biology. For example Neumann (1966) developed a self replicating machine called the universal constructor in the 1940s, about 10 years before Watson & Crick (1953) discovered the molecular structure of DNA.

1.2 Bio-inspired approaches

Bio-mimicry of various biological systems have become quite interesting because of the fact that these systems operate in an energy constrained domain and appear to be very effective in the tasks they have evolved to do. Neuromorphic computing is being investigated by several research groups (Navaridas et al., 2013, P. Merolla et al., 2011, Schemmel et al., 2010) with the goal of developing computing architectures that allow brain-like computation within or less than brain-like energy constraints. Neurogrid (Benjamin et al., 2014) is one such system which is able to simulate a million neurons in real-time consuming 2.7 Watts (W). To add context, a personal computer consumes a few hundred watts to simulate 2.5 million neurons but the brain has about 80,000 times more neurons than Neurogrid and consumes only about three times as much power (Benjamin et al., 2014). The sensory inputs connected to the brain have also been sources of inspiration for some researchers. Bartolozzi et al. (2011) present an asynchronous neuromorphic vision circuit inspired by "frame-less" nature of the mammalian vision. Instead of processing a sequence of video frames like a conventional camera would, they propose using an event driven processing system. The system responds to events such as motion, orientation and contrast. This reduces the time, computation and energy cost of processing each frame of video.

There are application areas such as swarm robotics or wireless sensor networks that rely upon relatively simple individuals to carry out complex tasks. In both cases energy

minimisation while maintaining processing speed and accuracy are important because these devices operate on a tight energy budget. This research project focuses on how information from neuroscience and existing bio-inspired approaches can produce approaches well suited to such applications.

1.3 Issues and Challenges in pattern recognition

Biology can also shine light on areas where current technology is lacking and where development is needed.

Certain use cases such as robots or wireless sensor networks operate under a constrictive energy budget. Due to energy constraints making frequent communications is undesirable. Therefore it is desirable to have some energy efficient pattern recognition on board to help reduce the need to communicate frequently. Of course savings made due to energy optimizations can scale up to use cases like data centres too. Current research into energy efficient implementations, such as neuromorphic computing, focus on drastic changes to underlying hardware (Qiao et al., 2015). Exploring other aspects of bio-inspired computing might yield energy efficiencies while maintaining compatibility with contemporary hardware and software.

Part of the reason brains are energy efficient might be due to the way information is represented (Attwell & Laughlin, 2001). In fact there are biological pressures to evolve accurate, flexible and energy efficient representations for information. Sparse-coding

as an important mechanism in the sensory processing of the brain is supported by evidence from a variety of experimental studies on an assortment of species (Vinje & Gallant, 2002, DeWeese et al., 2003, Brecht & Sakmann, 2002, Perez-Orive et al., 2002). The crux of this idea is that out of large populations of neurons only a relatively low number of neurons would represent a particular piece of sensory information. A good example is that of a neuron in the retina responds to any contrast while a neuron in the cortex responds to a particular edge orientation (Olshausen & Field, 2004). This example also demonstrates that information is represented more sparsely as it is transmitted further down the processing chain. Experiments with neural networks in computer vision have also showed that neural networks learn a sparse set of basis functions to represent an input (Olshausen & Field, 2004). A different interpretation of sparse coding utilizes properties of High Dimensional Vectors and is called Sparse Distributed Memories (SDM) (Kanerva, 1988, Gayler, 2003, Kanerva, 2014, Levy & Gayler, 2008, Osipov et al., 2014). However SDMs are rarely used for visual pattern recognition with only two approaches implicitly using the concept (D. W. Arathorn, 2002, Khan, 2002). Both D. W. Arathorn (2002) & Khan (2002) present methods that increase in accuracy when input is encoded into sparse vectors however they do not connect their approaches with SDMs. Further study on the SDM interpretation of sparse coding is required to better utilise approaches such as those presented by D. W. Arathorn (2002) & Khan (2002) and enable SDMs to be applied to more practical problems.

One of the most crucial functions that animal intelligence requires is the ability to

store and integrate information across time and modality (Lewicki et al., 2014). Current methods have very short memories compared to what we observe in animals. The problem of information representation is related to this as it is evident that the representation used by the brain does have a much greater capacity to integrate temporal information.

An area where energy efficiency, flexible information representation and effective use of temporal information all play an important role is swarm robotics. Swarm robotics involves the organisation of a group of robots into a swarm much like swarms of birds or schools of fish. Visually tracking such a swarm is a difficult task due a phenomenon known as predator confusion. Swarming prey animals have evolved behaviours to actively confuse the sensory systems of potential predators. This scenario makes a challenging test-bed for the bio-inspired pattern recognition systems described in this thesis.

1.4 Hypothesis

In order to tackle the challenges outlined above, the following is used to guide the work described in this thesis:

We believe that deriving information from neuroscience and investigating bio-inspired approaches will enable us to produce algorithms and computational frameworks that can:

- *Minimise energy usage while maintaining processing speed.*
- *Maintain or increase accuracy.*

- *Readily generalise to novel scenarios.*

Extant literature from both biological and computer science fields support this hypothesis and indicate that the three goals are achievable. The core biological evidence that lays the foundation for this hypothesis is the idea of sparse coding mentioned earlier. Using the SDM interpretation of sparse coding, highly parallel implantations are possible. Studies such as (Bartolozzi et al., 2011) have shown how asynchronous computation can be more energy efficient. There is also evidence that the brain has evolved under an energy constraint (Attwell & Laughlin, 2001) suggesting that sparse coding might be energy efficient.

1.5 Research questions

This thesis explores the problem space described in §1.3 with respect to the hypothesis described above. Below are the research questions being tackled in this project:

R1 *Is energy minimisation an inherent property of parallel-distributed systems?*

R2 *Are sparse representations of information as relevant to machines as they are in biology?*

R3 *How can temporal feedback be utilised in a pattern recognition system to improve accuracy and speed?*

Tackling R1, R2 and R3 provide the foundation to build more practical pattern recognition systems based on the principals of SDMs. To test whether this combination works, a unique biological problem was selected.

Recognising and tracking swarms of animals utilizing predator confusion is a challenging visual problem because the targets are actively attempting confuse the observer. Applying learnings from R1, R2 and R3 to track a swarm not only tests research outputs on a difficult test-bed but also sheds some light onto whether biological confusion tactics are effective on modern computer vision techniques:

R4 *Does the predator confusion affect computer vision in the same way it does biological predictors?*

1.6 Contributions

The contributions of this thesis are as follows:

- In order to answer R1, a comparative study with a highly parallel bio-inspired pattern recognition approach and a non-bio-inspired pattern recognition approach was carried out. This study showed that fine grain parallelism, parallelism resulting from numerous small tasks, allowed for significant gains in energy efficiency. These result are published in (Y. R. Hettiarachchige et al., 2018).
- As part of the implementation of the bio-inspired approach mentioned above several experiments were carried out exploring the conditions under which sparse coding is most effective. These experiments helped develop a set of heuristics that produce good sparse representations for the bio-inspired approach. The experiments also answer R2, the sparse representation enabled the massive parallelism in the bio-inspired approach which in turn enabled the previously mentioned energy efficiency (Y. R. Hettiarachchige et al., 2018).
- In order to answer R3 and R4, two 3D tracking approaches both sharing common attributes of being bio-inspired, parallelisable and utilising temporal information were developed. The key differentiator between the two approaches

was the method by which patterns were learnt: One was a recurrent deep neural network while the other was a one shot learning associative memory. These two approaches were tested on a novel testbed involving the challenging problem of tracking a swarm of objects taking advantage of predator confusion (Y. Hettiarachchige et al., 2018).

1.7 Thesis Organization

Chapter 2 presents the relevant literature covering topics such as computer vision, SDMs and Sparse Coding in depth. The advantages of bio-inspired approaches are described and challenges to do with energy efficiency, information representation and temporal information is highlighted.

Chapter 3 investigates the information representation in the form of sparse coding as it is using in a bio-inspired approach called Map Seeking Circuits. This is used to explain the results discussed in Chapter 4 and produce heuristics to improve sparse coding schemes.

Chapter 4 presents an experimental comparison between a bio-inspired approach and a non-bio-inspired approach (which we refer to as "algorithmic") to computer vision. The insights gained are used show how bio-inspired approaches lend themselves for greater energy efficiency via the use of sparse coding.

Chapter 5 applies learnings from the previous chapters regarding parallelism, energy efficiency and sparse coding to the difficult visual problem of tracking a swarm of an-

imals. Two novel bio-inspired approaches are presented and applied to this problem. The two approaches use different interpretations of sparse coding, so this chapter also presents a comparison between different interpretations of sparse coding.

Chapter 6 summarises the contribution of this thesis and discusses future research that can be done in the area.

2

Background

This project draws considerably from the behaviour of organisms and research investigating the biological mechanisms that create such behaviour. There is a wealth of pattern recognition approaches that either explicitly attempt to mimic biological processes or leverage core concepts from biological processes. On the other hand there are also many approaches that have developed equivalent functions with purely engi-

neering considerations. This chapter presents literature in pattern recognition broadly categorised into two main classes: bio-inspired and algorithmic or non-bio-inspired approaches. Within these two classes, subjects such as the encoding of sensory information, the role of temporal information, memory, pattern learning and the recognition are of special interest. §2.1 presents a broad overview of non-bio-inspired approaches will be presented first, to provide context for the need for bio-inspired approaches. From here on non-bio-inspired approaches will be referred to as algorithmic approaches. Then in §2.2 a broad selection of relevant bio-inspired approaches will be presented and gaps within current bio-inspired pattern recognition research will be highlighted. Finally, §2.3 presents the conclusions of the chapter.

2.1 Algorithmic approaches

There are many approaches to pattern recognition that are not bio-inspired. These are often designed for specific purposes and perform very well in those domains. This section goes through a broad range of such approaches initially. Support Vector Machines and Random Forests are presented because they form the basis for many non-bio-inspired approaches to vision. Non-bio-inspired visual pattern recognition will be covered towards the end, these approaches will provide contrast and context for the bio-inspired approaches that will be presented in §2.2.

2.1.1 Support Vector Machines

Support Vector Machines (Cortes & Vapnik, 1995) (SVMs) are a type of supervised binary classifier. The approach works by transforming input vectors non-linearly such that they map onto a high-dimensional feature space. Then a hyperplane is constructed such that it maximises the distance to the nearest training data point for each class (see Figure 2.1). Due to the flexibility of the approach SVMs have been applied to a very wide range of applications. From face detection (Osuna et al., 1997) to bio-informatics (Byvatov & Schneider, 2003). However in situations where multiple classes are concerned or where labelled training data is not available, SVMs are difficult to apply.

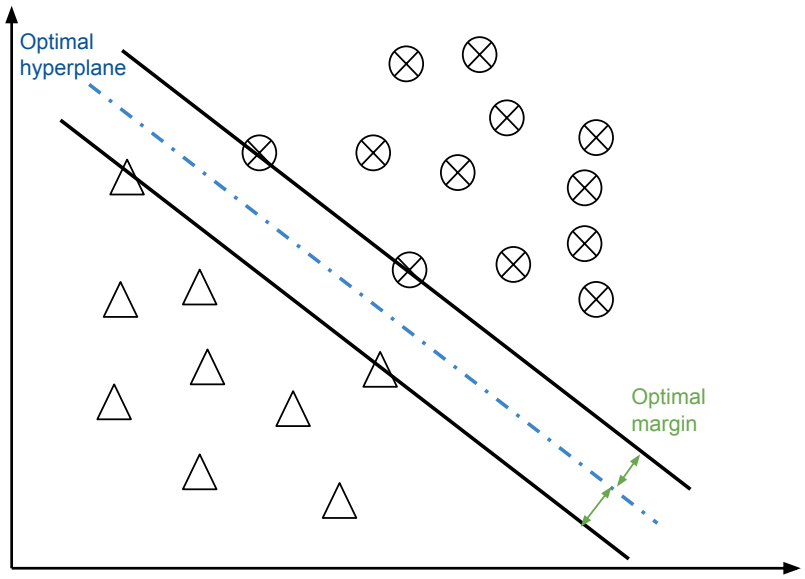


Figure 2.1: A visualization of a hyperplane separating data points into two classes in 2D space. The optimal hyperplane maximises the optimal margin.

2.1.2 Random Forests

Random forests (Breiman, 2001) are a type of ensemble method. The approach gets it's name from using the mean output of many decision trees. The approach is useful for use-cases where the data set is large and there are many potential features. Each tree separates the dataset differently so the ensemble approach reduce the risk of over-fitting to the training set. However in recent years deep neural network approaches have produced better accuracies (see §2.2.2.2).

2.1.3 Algorithmic approaches to vision

Approaches such as random forests and SVMs are general pattern recognition approaches that have been applied to vision tasks. There are other, more niche, approaches that build upon a myriad of other techniques to build vision systems. This subsection presents three non-bio-inspired approaches covering three important topics in computer vision: object detection, scene understanding and object tracking.

2.1.3.1 Deformable Parts Model

Deformable Parts Model (DPM), first introduced by Felzenszwalb et al. (2010), has gotten very good results in the PASCAL dataset. Before the surge of deep neural network based approaches, DPM was one the best performing object detection methods. DPM is built on the intuition that objects can be represented in a hierarchy of parts. The

parts location is constrained by the structure of the object. So detecting the parts in a valid configuration would mean that it is likely that the object has been found.

This hierarchy of parts is represented in DPM by a hierarchy of Dalal-Triggs detectors/filters, which use histogram of oriented gradients (HOG) to represent an object category. In (Felzenszwalb et al., 2010) this hierarchy is called the star model. A model at a particular scale and position within the image is given a score. This score determines if there is a recognition of a object in that scale and position.

2.1.3.2 Scene understanding

Zia et al. (2014a, 2013) present a trainable multiple object pose detector. Their publications describe its use for the detection of vehicles and they apply it to semantic scene understanding (Zia et al., 2014b). Their technique utilises 3D computer aided design (CAD) models as training data. Principal component analysis is applied to simplified versions of the CAD models to produce shape models. They also use the unsimplified cad models to train part detectors. When detecting an object, DPM is used to localise the object in 2D after which a sliding window applies the part detectors and generate evidence for each of the previously trained parts. Then the shape model and 3D pose that best explain this evidence is selected by exploring the possible object geometries and 3D poses using the shape models and random sampling.

The use of a separate step to localize an object in 2D before applying their method is a bottleneck in terms of accuracy; any false positive or false negative affects their overall

accuracy. In their tests the DPM 2D localiser only detected 52% of the objects. Out of that 52% their method detected 96% of objects and estimate the pose of the object to less than 1m 44% of the time. These results show that the 2D recognition of object is a limitation on accuracy. The next approach we review simultaneously localises in 2D and 3D.

2.1.3.3 Multiple Object Tracking

Pauwel et al. introduces a bi-ocular 3D object detector and pose estimator (Pauwels et al., 2014, 2015). Pauwels et al. (2013) present an approach, which is referred to as Phase Based Tracker (PBT), for use in a bi-ocular vision system. The system processes the two input images such that a stereo disparity map and a optical flow map is created. The disparity essentially gives each pixel a depth value. Since the pixels have been assigned depth values the image can now be thought of as a 3D point cloud. Like Zia et al., Pauwel et al. assume that 3D models of target objects are available. The approach has two interesting aspects: Firstly, the use of both sparse and dense information. Secondly, the feedback of results from the previous frame of video to initialise the tracker with an approximate transformation in the next frame of video. This use of feedback shrinks the search space of possible poses and positions the object can take. In the following paragraphs we will tersely describe the PBT approach, for more in depth information refer to (Pauwels et al., 2013) and (Pauwels et al., 2015).

The system uses several kinds of information that is computed from the input im-

ages from the two cameras:

- Phase-based dense motion.
- Phase-based dense stereo.
- Scale Invariant Feature Transforms (SIFT)

All three sets for information is computed using GPU accelerated algorithms. We will briefly discuss the phase-based stereo and motion as they are specific to Pauwel et al.'s approach unlike SIFT.

Pauwels et al. refer to stereo disparity and motion information as "dense" meaning that this information is per pixel. Stereo disparity maps would have added depth information per pixel and optical flow maps would have added optical flow vectors per pixel.

Computing the disparity between the images from both cameras adds depth information to the other wise 2D representation essentially making it a 3D point cloud.

Motion information or optical flow is the apparent motion of objects in a scene relative to the observer, this is computed by a phased-based optical flow algorithm. This motion is represented as a vector field.

Pauwel et al. feeds back the pose estimates produced to the first step of the process, which is extracting stereo disparity and motion information. This is done by using OpenGL to render the objects and limiting the preprocessing to the pixels belonging to the object. Previous frame's pose estimate is also used as a starting pose for pose detection.

The three sets of information produced by prepossessing is used in pose detection and tracking. The algorithm essentially searches for the 3D transformations that transform the model as it was posed at frame $f - 1$ (the previous frame) to as it is posed in the current frame f:

$$m' = Rm + t \tag{2.1}$$

where m' is the model's pose at frame f, R is the 3D rotation matrix, m is the model's pose at frame $f - 1$ and t is the 3D translation vector. So identifying R and t is the problem here. Both stereo disparity and optical flow is used to jointly estimate the translation and rotation by minimising the error between the estimated pose and the observed pose. For more information see Equations 13 - 15 in (Pauwels et al., 2015).

Pose estimation for the stereo is done via a Iterative Closest Point (ICP) algorithm. ICP matches the point cloud created from the disparity map of the current frame with the pose estimate produced in the previous frame.

The use of motion information or optical flow is based on the concept of Structure from Motion (often abbreviated to SfM). As the name suggests, structure can be estimated given motion information. In PBT this notion is inverted, since the 3D memory models describe the structure of the object, motion (3D translation and rotation) is estimated based on structure. The current pose estimate is used to render the object into the current frame and compute what is called augmented reality (AR) flow. This is used to create error measure that represents the distances between the expected pixel

motion and observed optical flow. The AR optical flow is also used to create another error measure that is the distance between the expected pixel motion and the AR optical flow.

By using these three error functions (one for disparity, three for motion) the rotation and translation that produces the least error is found. To improve accuracy when faced with large rotations (this will cause inaccuracies in ICP) an iterative M-estimation scheme is used to gradually remove outliers from the estimation.

There is a second component to the pose estimation. A RANSAC-based pose estimator is run in parallel. This is called sparse pose detection while the pose estimation discussed above is called dense pose detection in Pauwels et al. (2013, 2015). The sparse detector does not use information from the previous frame and so is independent of previous pose estimates. The pose estimates produced in the sparse and dense detectors need to be compared and the most correct estimate selected. This is done based on which of the two estimates produce the largest proportion of valid AR optical flow when their pose estimates are fed back into the preprocessing phase.

As mentioned earlier, Pauwel et al.'s approach has very positive results. In terms of run time, parallelism is utilised to achieve 20Hz (20 frames of video per second or 50 milliseconds per frame) utilising a dual-core Nvidea GTX 590. In terms of accuracy it's minimum tracking success rate for noisy and occluded settings are 97% and 57% respectively. However, it should be noted that the control test, having the tracker always predict a static object located in the centre of the image, had an minimum accuracy of

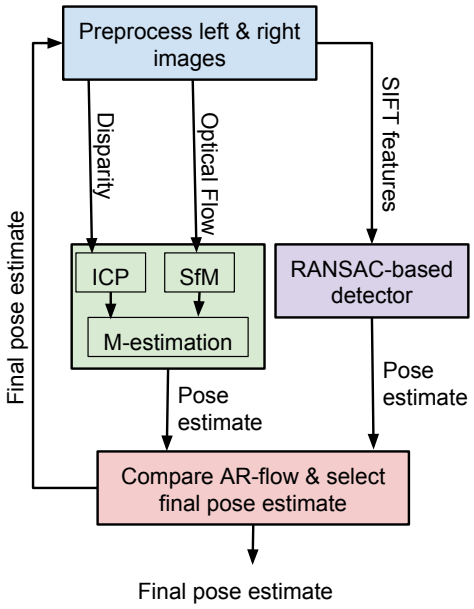


Figure 2.2: This is a high-level visualisation the approach described by Pauwels et al. (2015).

45% in both noisy and occluded tests.

2.1.4 Summery of algorithmic approaches

An overview of pattern recognition algorithms has been discussed. Various non-bio-inspired approaches were presented including approaches that are vision specific. Important aspects such as object detection, scene understanding, object tracking and utilising temporal information was covered.

There are also bio-inspired methods to achieve similar results. The next section will present approaches that tackle similar problems from a biological perspective.

2.2 Pattern recognition in biology

Pattern learning and recognition is a problem that is tackled by all animals. Small insects such as honey bees have been shown to be able to recognise human faces (Dyer et al., 2005) and even form abstract concepts (Avarguès-Weber et al., 2012, Giurfa et al., 2001). Dyer et al. (2005) showed that bees can discriminate between faces and by doing so they showed that the visual and neural systems are capable of learning to recognise patterns even if the organism has no evolutionary history for recognising that pattern. Avarguès-Weber et al. (2012) show that bees can be trained to form two abstract concepts and apply them to form rules. Giurfa et al. (2001) discuss similar results. Bees as well as other other animals have evolved this ability to carry out scene analysis.

In a recent paper, Lewicki et al. (2014) discuss the how a variety of animals carry out scene analysis using different senses. They discuss how the jumping spider uses it's vision system to hunt prey, locate mates and navigate the environment. They explain how songbirds analyse the acoustic scene to carry out functions such as mate attraction and territorial defence. They also review how a type of fish called the mormyrid uses electric sense to forage for food, navigate in darkness and to communicate. Lewicki et al. (2014) conclude their review by identifying four common principles that enable scene analysis in complex environments:

i The ability to utilise *a priori* knowledge to successfully extract scene properties from raw inputs when there is insufficient sensory data to arrive at a unique solution.

ii The ability to integrate and store information across different sensory modalities and time.

iii Efficient recovery and representation of 3D scene structure.

iv Motor actions that guide the acquisition of further sensory information required to achieve behavioural goals.

Notice that in many ways these principles represent problems that are being tackled in field of pattern recognition as discussed in the previous section.

Lewicki et al. (2014) also argue that problem of recognition has been defined too narrowly in the field of computer vision. Indeed much of the attention in computer vision has been to develop systems capable of assigning labels to pixels. But labels do not capture enough information to account for natural behaviour. Lewicki et al. (2014) cite the fact that many behaviours require the organism to have knowledge of things such as the object's 3D pose, location and geometric shape. Lewicki et al. (2014) go on to argue that treating recognition as a categorization problem causes the issue of representation to go unaddressed. They make the case that recognition in animals likely uses some representation that encodes 3D object structure in a viewpoint invariant form. They point out that current research that uses 3D models represent the 3D models in Euclidean space which is unlikely in nature.

2.2.1 Sensory information

Looking at nature we see a wide spectrum of different sensory inputs being utilised to aid the survival of different animals. We see that some senses that are common in

nature have evolved to a wide variety of sensitivities. For example Marshall & Arikawa (2014) compare the range of colour perceived by different species; humans being able to perceive wavelengths ranging 400nm-700nm while stomatopods(mantis shrimp) perceive wavelengths ranging 300nm-750nm. We also see senses familiar to us such as sight used in conjunction with senses such as magnetoception, which enables the animal to perceive magnetic fields. There is much literature investigating magnetoception used by homing pigeons to navigate (Schmidt-Koenig & Walcott, 1978, Keeton et al., 1974, Lednor & Walcott, 1983).

Studies focussing on a variety of sensory systems in different species has found similarities in neural activity suggesting a common mechanism for sensory coding. This common mechanism has been called *sparse coding*.

2.2.1.1 Sparse Coding

The theory of sparse coding of sensory inputs states that given a large population of neurons, information is represented by a relatively small number of simultaneously active neurons (Olshausen & Field, 2004). A often cited example of this is fact that the visual cortex of the cat has neurons tuned to specific edge orientations (Hubel & Wiesel, 1962). The inputs that stimulate such a tuned neuron is called it's *receptive field* (Olshausen & Field, 1996) in sparse coding literature. The following paragraphs will discuss evidence for sparse coding, and discuss how sparse coding is relevant to pattern recognition.

There are several reasons to believe that sparse coding plays a central role in sensory input processing. Firstly there are numerous experimental studies showing examples of sparse coding in a variety of sensory systems. We have already mentioned results from Hubel & Wiesel (1962) showing sparse coding in the visual cortex of the cat. In the auditory cortex of the rat, DeWeese et al. (2003) report highly reliable single spike response to tones at different frequencies. Vinje & Gallant (2002) present a study focused on the visual cortex of primates. Their results show that when shown sequences of images resembling those that occur in nature, the response is sparse over time. Suggesting that sparse code also plays a role in temporal information. Additionally, their results showed that when the same neurons were stimulated by just their receptive fields (input that when presented in isolation stimulates the neuron) the response was actually dense over time meaning that context actually sparsifies the response. Sparse coding is not limited to mammals, it has also been reported in insects. A recent study focussing on the olfactory system of fruit flies (Lin et al., 2014) demonstrated that not only was responses sparse but that when a particular feedback loop was disrupted response sparsity decreases and also inhibits the discrimination of similar but not dissimilar odours. What this shows is that the fruit flies ability to distinguish between similar odours is dependant on sparsity.

There are also several advantages of sparse coding which theoretically should bias evolution to converge upon it. These properties are also advantageous to the systems we aim to build. One of the most convincing is that sparse coding is energy efficient.

Sparse patterns expressed as sparse vectors have been demonstrated to be highly robust representations for the purposes of pattern recognition (Kanerva, 1988, Olshausen & Field, 1996). Attwell & Laughlin (2001) present a study in which they show that due to energy constraints only 1/50th of cortical neurons could afford to be active. They also show that sparse codings where less than 15% of neurons are simultaneously active reduces energy consumption and allows greater computational power.

Due to both the experimental evidence and the theoretical advantages presented above, sparse coding appears to be a plausible biological solution to represent sensory input. The same properties that make sparse coding advantageous in biology also make the it advantageous when applied to artificial pattern recognition.

Mathematically there appears to be two interpretations of sparse coding in the literature. The first and most prevelant is the interpretation presented by Olshausen & Field (1996). It is based on representing an image via a sparse selection of basis functions out of a large set of basis functions:

$$I(\overrightarrow{x}) = \sum_i a_i \phi_i(\overrightarrow{x}) \tag{2.2}$$

where I is a vector with the components $\overrightarrow{x}$ denoting a discrete spatial position in the image. The selection of basis functions is denoted by ϕ. Olshausen & Field (1996) refer to a as amplitude that needs to be computed per image. Viewing Equation 2.2 as a weighted superposition allows this to related to another bio-inspired approach pre-

sented in §2.2.2.6.

The second interpretation is under-represented in the literature and was first presented by Kanerva (1988). Kanerva (1988) present sparse coding as the encoding of information in large high dimensional binary vectors where only a few of the elements are 1 and most are 0. This is discussed in more detail in §2.2.2.3.

2.2.2 Bio-inspired pattern recognition

Many bio-inspired approaches to pattern recognition have been based upon the simplified model for biological neurons proposed by McCulloch & Pitts (1943). The McCulloch and Pitts model of neurons can have many binary inputs connected into it and one binary output. A weighted sum of these inputs is used to determine if the neuron fires, if the weighted sum is greater the neuron fires. These neurons can be connected together in various configurations. For example some configurations can reproduce logical functions. Such networks of neurons became known as Artificial Neural Networks (ANN).

Hebbian theory (Hebb, 1949) has also had a great influence on the progress made in ANNs. Hebb's theory is:

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased. (Hebb, 1949)

Hopfield networks (Hopfield, 1982) utilise both McCulloch and Pitt's ANNs and

Hebb's theory to create a trainable associative memory, i.e. a memory that can be accessed based on content. Hopfield networks are complete graphs with each neuron in the network connected to every other neuron. An important deviation from the McCulloch and Pitts model is that Hopfield neurons are binary but use -1 and 1 instead of 0 and 1. It works by using a energy function such that the networks overall energy decreases and eventually reaches a local minimum; e.i. a result. However, as the number of patterns committed to memory increases the accuracy of recall decreases Hopfield (1982). This accuracy decrease is analogous to forgetting in human memories, some refer to it as the "stability-plasticity dilemma" or "catastrophic forgetting" (Carpenter & Grossberg, 2010). Recent work in ANNs have incorporated more complex models of neurons and larger networks.

Spiking Neurons can be considered a more accurate model of a biological neuron. Spiking neurons model the membrane potential of a biological neuron by using assigning each neuron an potential (an activation level). A neurons potential at any point in time is the sum of excitatory post-synaptic potentials (EPSPs) and inhibitory post-synaptic potentials (IPSPs); which are caused by other connected neurons firing. When potential is greater than a threshold, the neuron fires. This is known as "integrate-and-fire". The model is inherently temporal (Maass, 1997). Instead of encoding information in the weights assigned to the inputs, information is encoded by sequences of activations or "spikes".

2.2.2.1 Reservoir Computing

Reservoir computing is a computational framework that utilises networks of randomly and recurrently connected neurons. There are two very similar independently developed approaches to reservoir computing: Liquid State Machines (LSM) (Maass et al., 2002) and Echo State Networks (ESL) (Jaeger & Haas, 2004). Both approaches share a common structure. Inputs are fed into the system via a randomly connected network of neurons called the reservoir layer. The dynamics of the reservoir layer maps the input into a higher dimension. Since the two approaches are very similar we will only discuss LSM is detail.

LSMs are an accepted model for brain-like computations (Maass et al., 2002). LSMs offer a computational model which does not require a central "clock", lending itself to algorithms with high levels of parallelism. An analogy Maass et al. (2002) use to explain the approach is puddle of water. External influences can cause perturbations with in the puddle, which in time weaken and disappear. At any point in time the state of the perturbed liquid encodes present and past states. The attenuation of perturbations can be thought of as a fading memory. In LSM, inputs generate perturbations which is used to map the desired output to the input. The anatomy of a LSM is quite simple it is composed of five components: the input stream $u(\cdot)$, the output is some predetermined function $y(\cdot)$ that carries out some useful work on $u(\cdot)$, the liquid filter L^M that takes $u(\cdot)$ as input, the liquid state $x^M(t)$ at time t created by the liquid filter and

lastly the readout filter f^M that transforms the liquid state $x^M(t)$ into $y(\cdot)$. The liquid filter is a randomly connected network of neurons whose recurrent nature causes the aforementioned perturbations. Unlike the liquid filter the readout filter is a population of integrate and fire neurons which has been trained for a specific purpose. Since many readout filters can operate on the same liquid filter and since the liquid state enables each separate readout filter to sample the liquid at any time, LSMs are highly asynchronous and parallel.

2.2.2.2 Deep Neural Networks

In recent years Deep Neural Networks (DNN) have become favourable. Inspired by biological sparse coding, they attempt to reduce complexity by using hierarchical compositions of low level components to form high level components. The biological analogue of this is the sparse-encoding discussed in previous sections. Indeed, the ability to automatically learn a hierarchy of composable features is one of the major advantages of DNNs.

Deep learning algorithms such as Convolutional Neural Networks (LeCun et al., 2010) (ConvNets) have been widely adopted by the computer vision community because it has produced very good results in object recognition tasks (Krizhevsky et al., 2012). Most of this progress has been driven by the commercial applicability of ConvNets and classification in general. Companies such as Google have used ConvNets to censor faces and licence plates to protect privacy (Frome et al., 2009). The ability to

label images at the pixel level also has many applications and has attracted the attention of researchers and companies interested in hardware implementations of ConvNets for real-time uses such as robots (Farabet et al., 2011), self-driving cars and smart phones (LeCun et al., 2010, 2015).

However DNNs remain expensive to train as they require many training iterations over a large dataset. The need for a large training set is lessened by data augmentation techniques (Lecun et al., 1998, Simard et al., 2003, Rebai et al., 2017) but may not be enough in certain scenarios.

2.2.2.3 Sparse Distributed Memories

Research done on sparse coding (see §2.2.1.1) is complemented well by Kanerva's Sparse Distributed Memories (SDMs) (Kanerva, 1988). SDMs were developed as a mathematical model for long term memory. SDMs take advantage of statistical properties of high dimensional (HD) binary spaces (Kanerva states that usually less than 10,000 dimensions is sufficient). Each point in a HD space is addressed by a HD vector. In SDM a piece of information is represented by a point within the binary HD space. The similarity between different pieces of information is measured by their Hamming distance, the minimum number of substitutions needed to change one binary HD vector to another binary HD vector. The "sparse" in "Sparse Distributed Memories" comes from the fact that it is assumed that the information points are distributed sparsely throughout the space. Due to the high number of dimensions, any one point within a HD

space would be relatively far from other unrelated points. In fact if two points were drawn randomly from such a space they are likely to be orthogonal. This also means that information can be encoded into the HD vector imprecisely because accidental overlap is unlikely. Features like imprecise information encoding an distributed representation make SDM a biologically plausible model for memory. For example the retina is very unlikely to receive the same inputs twice yet we are capable of identifying specific objects.

As associative memories SDM has been shown to be able to replicate rather complex behaviours exhibited by human memories such as the ability to "chain" different concepts via related concepts (an example from (Kanerva, 1988): "apple" → "red" → "firetruck") and "tip of the tongue" phenomenon described by Brogliato et al. (2014).

2.2.2.4 Vector Symbolic Architectures

A set of ideas very closely related the SDMs, that also utilise HD vectors, have been given the umbrella term Vector Symbolic Architectures (VSAs)(Gayler, 2003, Kanerva, 2014, Levy & Gayler, 2008, Osipov et al., 2014). VSAs primarily aim to model cognition, random binary HD vectors are used to represent concepts and vector operations such as addition and multiplication are used to compose lower level concepts into higher level concepts and form relationships between concepts. Because all VSAs are based on vectors and operations carried out on vectors they are inherently parallel. There are several different approaches to VSAs proposed. Binary Spatter

Codes(Kanerva, 1994), Holographic Reduced Representations(Plate, 1995) and MAP (Multiply, Add, Permute) Coding (Gayler, 2003). The main differences between these approaches are the values of the HD vectors and specific operations used. For example, HRRs appear to fairly different compared to MAP and BSC because HRRs utilise real valued vectors instead of binary vectors.

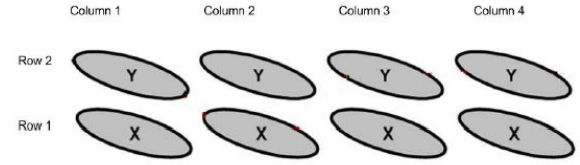
There has also been several papers proposing systems that utilise VSAs. Emruli et al. proposes a VSA based solution to interoperability between devices in the "internet of things" (Emruli et al., 2014). Their scheme uses BSC and SDMs to enable many different devices to communicate via HD vectors and learn to predict state changes within the network via an SDM. Osipov et al. (2014) propose a distributed associative memory named Holographic Graph Neuron (Holo-GN), an improvement to an existing distributed scheme called Hierarchical Graph Neuron(HGN) (Nasution & Khan, 2008). Holo-GN uses concepts from VSAs to lower the number of required computational nodes in HGN. Due to the fact that VSAs and SDMs rely on vectors and vector operations, they are very suitable for distributed systems such as wireless sensor networks, swarm robotics or the internet of things.

2.2.2.5 Graph Neuron

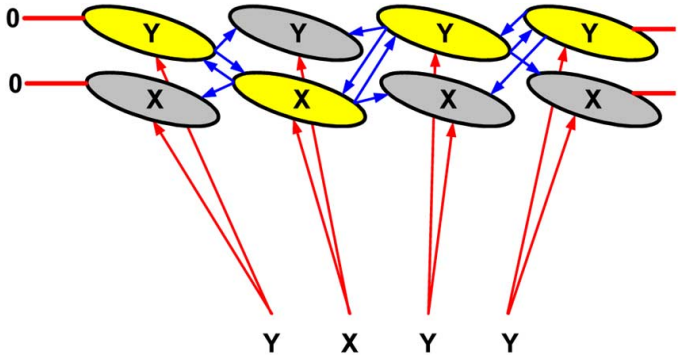
Graph Neuron (GN) is a distributed approach to associative memory and pattern recognition (Khan, 2002). A GN network is composed of simple neurons which activate in a binary fashion when input is received. Figure 2.3(a) visualises the network

and the way the neurons communicate to learn and recall patterns. The number of rows in an GN network is equal to the number of possible values per position and that the number of columns is equal to the pattern size. When a stimulus is received, for example the pattern $YXYY$, each value of the pattern is sent to their corresponding neuron column. Each neuron checks whether the incoming value is the same as their value; if it is the neuron becomes activated, if not it remains in it's idle state. Next, all the activated GNs would send their row indices to each adjacent GN. Figure 2.3(b) shows this process. By the end of the last step each neuron would have received the row indices of it's activated neighbours. These would be stored in a data structure known as the bias array (Khan, 2002, Nasution & Khan, 2008). Each element in the bias array contains the row indices of adjacent neurons activated in a particular pattern. Figure 2.3(c) shows an example of the bias array entries created when the network encounters the $YXYY$ pattern for th first time. Note that if the same pattern is encountered again no new bias array entries are required. This process is called collaborative comparison learning(Muhamad Amin & Khan, 2009).

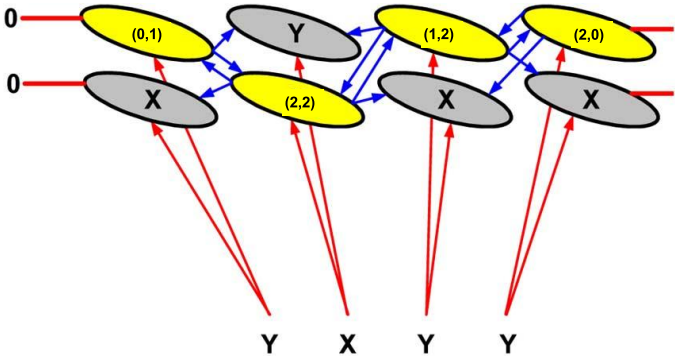
However the simple GN network presented above suffered from cross-talk between stored patterns and new incoming patterns. Cross-talk is when GN reaches the conclusion that it had encountered the pattern before when in fact it had not. The cross-talk problem arose because each neuron would only "know" about the states of it's immediate neighbours; therefore to solve the problem neurons would need way to become aware of the state of more neurons. This is what the Hierarchical Graph Neu-



(a) Basic structure



(b) Communication



(c) Bias array

Figure 2.3: Sub-figure (a) shows the structure of a GN network. Sub-figure (b) visualises the process of communication that enables the network to both learn new patterns as well as recall learnt patterns. Each neuron essentially learns the state of it's neighbours and associates that state with a particular patten in order to learn that pattern.

ron (HGN) (Nasution & Khan, 2008) does. HGN is composed of several simple GN networks connected in hierarchical layers. This way each layer has knowledge of the previous layer's state. This is visualised in Figure 2.4.

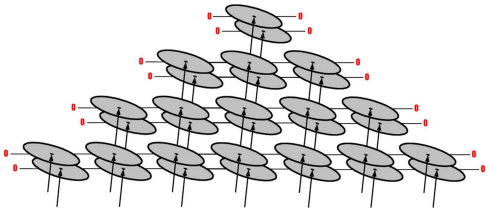


Figure 2.4: To solve the problem of cross talk present in GN a hierarchy of GN networks (HGN) was introduced.

Although HGN solved the cross talk problem, it hierarchical structure had several problems. Firstly, as input size increases the number of neurons required increases exponentially. Secondly, information storage is no longer completely distributed. The top layer of neurons would store all encounter patterns. Finally, if implemented in a WSN or swarm of robots it would not be able to take advantage of all individual devices in a swarm or WSN. To overcome these problems concepts from VSAs were used recently to overcome these problems. The Holographic GN (Osipov et al., 2014) enables the a flat GN network to have recall accuracy of HGN with out the hierarchy of neurons.

2.2.2.6 Map Seeking Circuits

Map Seeking Circuit (MSC)(D. W. Arathorn, 2002) is a bio-inspired approach to solve a broad range of problems that occur in nature called "transformation discovery prob-

lems". Important problems that animals need to solve can be viewed as transformation discovery problems such as limb inverse kinematics, path planning and vision (D. W. Arathorn, 2004). Looking at vision as an instance of a transformation discovery problem, the animal has 3D representations of objects in it's memory that when transformed by a certain sequence of transformations closely matches some object in the 2D projection of the world that the animal's eyes perceive. Notice how recognition, 3D pose and position estimation are all solved at once. The concept of transformation discovery can be abstracted to the search of correct mappings. Geometrical transformations are a type of mapping.

However seeking correct mappings is an expensive task; the number of possible mappings is often huge (conceder the visual recognition and localisation of a object). MSC tackles this problem by encoding inputs, memories and possible mappings as large sparse vectors (like SDM or VSA) and exploiting a property of such vectors called the ordering property of superpositions. The *Ordering Property of Superpositions* (or simply the ordering property) states that for a superposition $s = \sum_{i=1}^{n} v_i$ formed by the sum of sparse vectors $v_i \in V$ and another set of sparse vectors $v_k \notin V$, $v_i \bullet s > v_k \bullet s$. Intuitively, this can be shown as in Figure 2.5. Because the vectors are sparse, it is very unlikely that non-zero elements from several vectors will occupy the same position within the vector. So when the dot product of v_i and s is calculated, non-zero elements in v_i are guaranteed to multiply into corresponding non-zero elements in s. Any mismatched elements will simply be ignored as they would get multiplied

into zero elements in either s or v_i. If v_i is not sufficiently sparse or v_i is too numerous with respect to the size of s then the ordering property may not hold. This is called collusion and will be discussed later.

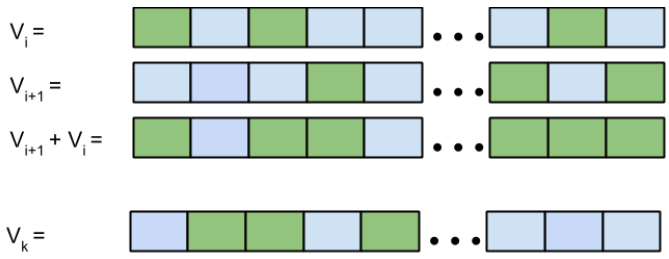


Figure 2.5: A simple visualisation of sparse vector superposition and the ordering property. The green elements are non-zeros while the blue elements are zero. Notice that: $(V_{i+1} + V_i) \bullet V_k < V_i \bullet V_k$. What a dot product would essentially do is give a measure of how much overlap there is between two pattern vectors. A superposition of vectors would have more overlap with vectors that formed the superposition than vectors that did not.

The ordering property is used along with competition between individual mappings to allow MSC to search the space of possible mappings efficiently.

MSC has been applied to rigid object recognition(Overman & Hart, 2012, Murphy et al., 2013), articulated object recognition(Arathorn, 2015), robot limb control (Arathorn, 2015), path planning and navigation (Snider & Arathorn, 2006, D. W. Arathorn, 2004), and even solving the Rubik's cube puzzle (Harker et al., 2007). There is also a proposed neuromorphic implementation (D. W. Arathorn, 2002).

2.2.3 Energy Efficiency

Energy efficient computation has received considerable attention in recent years due to the widespread use of mobile devices. Other applications such an robotics also require high performance yet low energy computation.

Bio-mimicry of various biological systems has become quite interesting due to the fact that these systems operate in a energy constrained domain. Neuromorphic computing is being investigated by several research groups (Navaridas et al., 2013, P. Merolla et al., 2011, Schemmel et al., 2010, Qiao et al., 2015) with the goal of developing computing architectures that allow brain-like computations within brain-like energy budget. Neurogrid (Benjamin et al., 2014) is one such system which is able to simulate 983 040 neurons in real-time consuming 2.7 W. In this context, a personal computer requires hundreds of watts to simulate 2.5 million neurons. A human brain has at least 80,000 times more neurons than Neurogrid and consumes only three times as much power. Neuromorphic computing focuses on the development of analogue circuits that incorporate neurological architectures to perform computations. A recent paper, Qiao et al. (2015), present a neuromorphic system capable of simulating 256 neurons operating approximately on 4mW. Another good example of specialized hardware for vendor specific neural networks such as the Tensor Processing Unit (Jouppi et al., 2017) which achieved both energy efficiency (about 30X - 80X more effiecient than contemporary GPUs and CPUs) and performance gains (about 15X - 30X faster). Algorithmic ap-

proaches to reduce the amount of computation is also a viable strategy to reduce energy usage as shown by Jiao et al. (2018) who use a kind of dynamic programming approach to avoid repeated computations for similar inputs when training neural networks. This strategy resulted in 47.5% energy saving while only costing a 1% accuracy drop.

However, neuromorphic computing would require an overhaul of current software engineering infrastructure. Applications such as robotics require both bio-inspired and algorithmic approaches to carry out all the functions required of them. Energy optimisation with conventional Von Neumann architectures has also been pushed by the prevalence of mobile devices. Studies at Intel (Grochowski & Annavaram, 2006, Shao & Brooks, 2013) showed that in terms of energy per instruction (EPI), using multi-core processors running at lower frequencies delivers high performance at an EPI that is approximately four times lower than an equally performing single core processor. Massive parallelism with low frequency processors might be a compromise between energy efficiency and staying compatible with existing software engineering infrastructure.

2.2.4 Summery of bio-inspired

In this section, pattern recognition with bio-inspired perspective was discussed. Sparse coding was highlighted showing it's prevalence in biology as well as the pattern recognition approaches that utilise mathematical interpretations of sparse coding to learn and store patterns. Table 2.1 summarises the approaches discussed in §2.2.2. Additionally the energy efficient properties of bio-inspired approaches were discussed.

Approach	Key Points
Reservoir Computing	Specialized hardware is required to exploit the parallel nature of spiking neurons. The reservoir layer is not trained, allowing for quicker training.
Deep Neural Networks	High accuracy across a number of domains. Requires a large training corpus.
Sparse Distributed Memories & Vector Symbolic Architectures	Easy to develop parallel implementations due to the use of vectors. Effective encoding schemes for images are difficult to develop.
Graph Neuron	Only requires a small training corpus. Low generalisation ability.
Map Seeking Circuit	Easily interpretable and fast. Accuracy is dependant on input encodings.

Table 2.1: The key points from §2.2.2

2.3 Conclusions

This chapter reviewed the literature in the two main areas that comprise our project: biology and pattern recognition. Throughout §2.2 the topic of sparse coding was revisited multiple times. The high dimensional vector based interpretation of sparse coding lends itself to highly parallel implementations such as MSC (§2.2.2.6) and GN (see §2.2.2.5). High parallel implementations may lead to energy efficiencies (See §2.2.3). From this, the first research question arises:

R1 Is energy minimisation an inherent property of parallel-distributed systems?

If sparse coding enables this parallelism, it is important to ensure that the vector based interpretation of sparse coding can be applied to practical problems because

approaches such as SDM (§2.2.2.3) and VSA (§2.2.2.4) are not applied to practical problems like computer vision. This is tackled by the second research question:

R2 *Are sparse representations of information as relevant to machines as they are in biology?*

As disccused in both §2.1 and §2.2 temporal information forms an important component of pattern recognition. The first of the principles of biological scene analysis listed by Lewicki et al. (2014) in §2.2 is the "ability to utilise *a priori* knowledge to successfully extract scene properties from raw inputs when there is insufficient sensory data to arrive at a unique solution". How a sparse vector based approach can utilise temporal information is explored by the third research question:

R3 *How can temporal feedback be utilised in a pattern recognition system to improve accuracy and speed?*

This chapter also discussed several use cases researchers have been using to test various pattern recognition approaches. The most challenging of these include detecting multiple objects such as Zia et al. (2014b) in §2.1.3.2, Pauwels et al. (2015) in §2.1.3.3 or Murphy et al. (2013) in §2.2.2.6. However looking at natural predictor prey interactions we observe a much more challenging test scenario: groups of prey actively attempting to confuse the predator via their movements. This is known at predator confusion and swarming animals often display it (Milinski & Heller, 1978, Jeschke & Tollrian, 2007). The final research question applies computer vision to the difficult problem of predator confusion:

R4 *Does the predator confusion affect computer vision in the same way it does biological predictors?*

The next chapter further explains the Map Seeking Circuit algorithm that introduced in §2.2.2.6 and how sparse encodings affect it's accuracy. This goes towards

3

Sparse Coding

3.1 Introduction

Sparse coding as a biologically plausible method to represent information was discussed in §2.2.1.1 and §2.2.2.3. Those two sections present two different interpretations of sparse coding. The first was introduced by Olshausen & Field (2004) and uses a sparse set of basis functions to represent information (see §2.2.1.1). The second was intro-

duced by Kanerva (1988) and uses high dimensional sparse vectors instead (see §2.2.2.3). In particular §2.2.2.3 discussed the second interpretation and listed a number of approaches that use large sparse vectors. One of the approaches listed was Map Seeking Circuits (MSC) (D. W. Arathorn, 2002). This chapter investigates sparse codings via MSC, examines it's performance and produces a set of heuristics that improve sparse encoding. Additionally this chapter lays the foundation for Chapter 4 which examines energy efficiency in highly parallel algorithms such as MSC.

3.2 Map Seeking Circuits

Map Seeking Circuits (MSC) (D. W. Arathorn, 2002) have been briefly discussed in §2.2.2.6. This chapter will examine MSC in detail, particularly how it relates to sparse coding.

As touched on in §2.2, Lewicki et al. (2014) present four common principles that enable scene analysis of complex environments in animals:

i The ability to utilise *a priori* knowledge to successfully extract scene properties from raw inputs when there is insufficient sensory data to arrive at a unique solution.

ii The ability to integrate and store information across different sensory modalities and time.

iii Efficient recovery and representation of 3D scene structure.

iv Motor actions that guide the acquisition of further sensory information required to achieve behavioural goals.

Lewicki et al. (2014) argues that problem of recognition has been defined too narrowly in the field of computer vision. For example, object recognition is seen primarily as mapping labels to pixels. In animals this insufficient information to drive behaviour. Animal behaviours require additional information such as object's 3D location, the object's context within the scene, it's geometric structure and properties needed to interact with the object. Lewicki et al. (2014) also point out how looking at recognition as a labelling or categorization problem means the problem of representation does not get tackled. Recent computer vision work (including the work presented in §5) uses 3D scanners to construct point cloud representations. However, biological representations are unlikely to use 3D point cloud representations. Sparse coding does offer a biologically plausible answer to the representation problem. The interpretation of sparse coding presented by Olshausen & Field (2004) has been applied to vision (Wright et al., 2010). However, the interpretation offered by Kanerva (1988) has not been applied to vision. An approach that implicitly uses the Kanerva interpretation of sparse encoding is Map Seeking Circuits.

Map Seeking Circuits (MSC) (D. W. Arathorn, 2002) addresses some of the problems Lewicki et al. had pointed out. MSCs solve a broad range of problems that occur in nature called "transformation discovery problems". Many problems that animals need to solve such as limb inverse kinematics, path planning and vision (D. W. Arathorn, 2004) can be viewed as transformation discovery problems. Looking at vision as an instance of a transformation discovery problem, the animal has 3D representations of

objects in it's memory that when transformed by a certain sequence of transformations closely matches some object in the 2D projection of the world that the animal's eyes perceive. By discovering the transformations required to map the 3D representation in memeory to the 2D projection from the eyes, recognition, 3D pose and position estimation are all solved at once. Furthermore, the concept of transformation discovery can be abstracted to the search of correct mappings. Geometrical transformations being one type of mapping.

One of the criticisms Lewicki et al. (2014) had was regarding representations. MSC is built around the properties of sparse vectors like SDM and VSA, which makes it biologically plausible. However in all current available work on MSC, information is encoded in an euclidean form and uses transformations that operate in Euclidean space so the representation problem remains partially unsolved.

3.2.1 MSC ALGORITHM

Assuming an input pattern v and memory pattern w the goal is to find the sequence of mappings $t \in \mathcal{T}$ ($\mathcal{T}$ are the possible mappings) that map v onto w and w onto v. The quality of a mapping is given by a correspondence function $C(t) = \langle t(v), w \rangle$ (we use the $\langle , \rangle$ operator to denote inner product). Transformation discovery can be formally defined as:

$$\arg \min_{t} C \tag{3.1}$$

In most problems $\mathcal{T}$ can be intractably large. MSC makes this computation tractable by encoding inputs, memories and possible mappings as large sparse vectors (like SDMs and VSAs) and exploiting a property of such vectors called the *Ordering Property of Superpositions* (or simply the ordering property). The ordering property states that for a superposition $s = \sum_{i=1}^{n} v_i$ formed by the sum of sparse vectors $v_i \in V$ and another set of sparse vectors $v_k \notin V$, $v_i \cdot s > v_k \cdot s$. Because the vectors are sparse, it is very unlikely that non-zero elements from several vectors will occupy the same position within the vector. When $v_i \cdot s$ is calculated, non-zero elements in v_i multiply into corresponding non-zero elements in s. Any mismatched elements will get multiplied into zero elements in either s or v_i (this is visualised in Figure 2.5). However, if v_i is not sufficiently sparse or the size of V is too large with respect to the size of s then the ordering property may not hold. This is called *collusion* in MSC literature and will be discussed later in §3.3. Revisiting Equation 3.1 we can see that the correspondence function C is a dot product. The dot product is most commonly used as C (D. Arathorn, 2001, D. W. Arathorn, 2002, Murphy et al., 2009, Martin et al., 2009) however dot products are not the only C possible, for example in a binary space Hamming distance would serve as an alternative.

The ordering property is used along with competition between individual mappings to allow MSC to search the space of possible mappings efficiently. The superposition is

created by a weighted sum of the individual mappings:

$$s = \sum_{i=1}^{|\mathcal{T}|} t_i(v) \times g_i \tag{3.2}$$

where s is a superposition, v is an input vector, $\mathcal{T}$ is a set of mappings, and $g_i \in g$, where g is a set of coefficients ranging from 0 to 1. The coefficients g_i represent the level of match between $t_i(v)$ and some stored memory vector w. The level of match is calculated by a dot product:

$$q_i = t_i(v) \cdot w \tag{3.3}$$

where $q_i \in q$, where q is a set of dot products of $t_i(v)$ and w. The dot products can be used in a competition function for the coefficients g_i:

$$g_i = \begin{cases} max(0, g_i - k(1 - \frac{q_i}{max(q)})) & , g_i \geq \theta \wedge q_i \geq \xi \\ 0 & , g_i < \theta \wedge q_i < \xi \end{cases} \tag{3.4}$$

where coefficient k controls the aggressiveness of the competition function, θ is the lower threshold of g_i, and ξ is the lower threshold for q_i.

Equations 3.2, 3.3, and 3.4 are applied iteratively until only one coefficient $g_i > 0$. This remaining g_i corresponds to the mapping t_i that maps input vector v to stored memory vector w.

MSC implementations organize mappings into several layers such that a sequence of

mappings from input to stored memory is found and vice versa. For example, Figure 3.2 shows block diagram of a MSC for visual object detection. The mappings in this case are geometric transformations. The possible transformations have been decomposed into four types of transformation, each with it's own layer: X/Y translation, 2D rotation, 2D scaling and 3D azimuth and elevation. Figure 3.1a visualizes a MSC layer. Each layer is connected to the next layer via two reciprocal pathways; a backward path and a forward path. The forward pathway searches for sequence of mappings $\mathcal{T}$ that maps the input v to the stored memory w while the backward path searches for the inverse mappings $\mathcal{T}^{-1}$ which maps w to v. Communication within the pathways is done via the superpositions (Equation 3.2). Each layer l has a backward input superposition b^l_{in}, a backward output superposition b^l_{out}, and corresponding pair of forward superpositions f^l_{in} and f^l_{out}. The layers are connected sequentially so $b^l_{in} = b^{l-1}_{out}$ and $f^l_{in} = f^{l+1}_{out}$. In terms of Equation 3.3, $v = f^l_{in}$ and $w = b^l_{in}$. Figure 3.1a also visualizes, at a high level, the parallelism of MSC. Each line can be run in parallel to the other lines. All the data flow lines, except those flowing from $Eq\ 3.2$ to g, actually represent vectors. There is fine grain parallelism embedded within the data flow due to the use of vectors. Each data flow line can have its own parallel data flow as shown by Figure 3.1b. Figure 3.1b shows that the expensive operations such as dot products and Equation 3.4 can be done in parallel. Also note that all the operations can be parallelized themselves at the per vector element/pixel level.

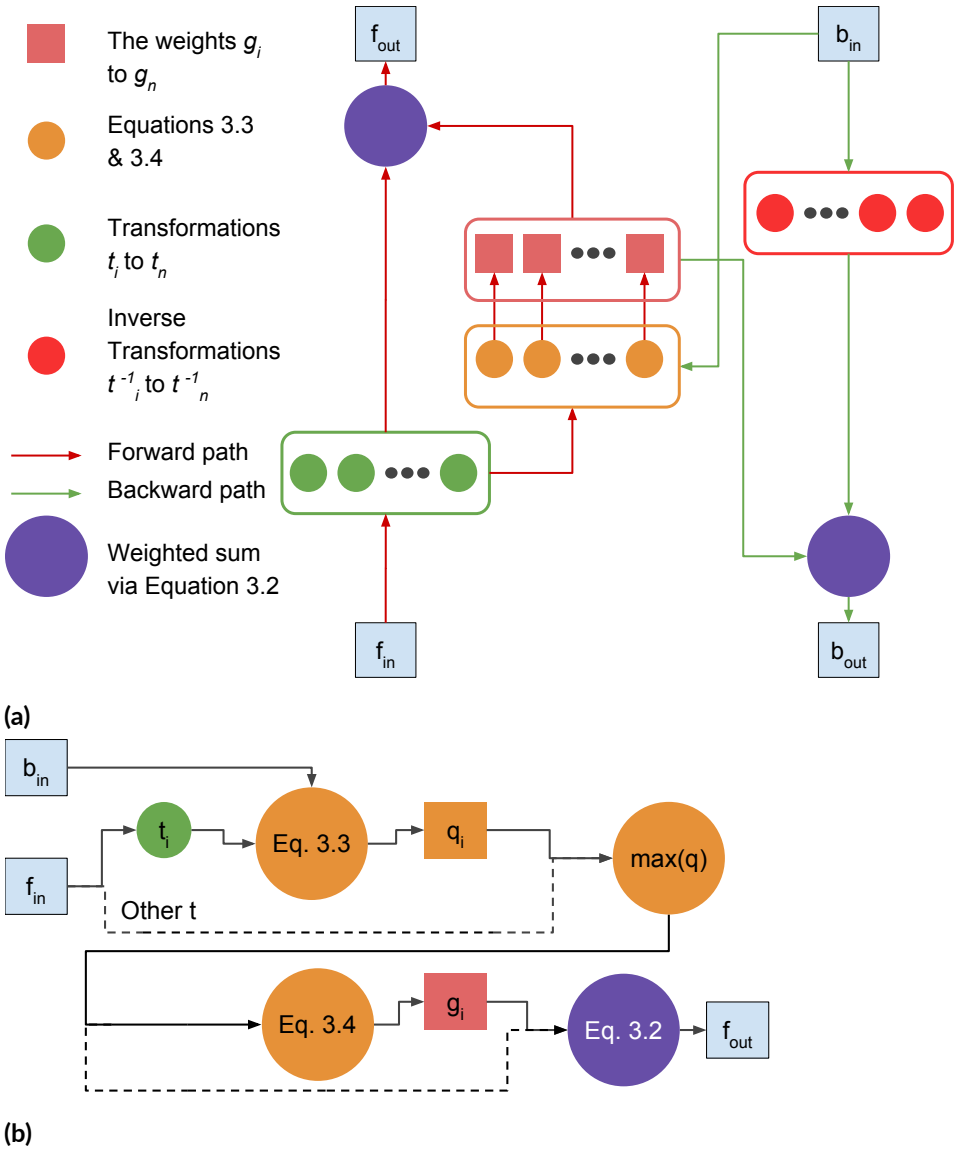


Figure 3.1: Diagram (a) visualizes the components of a typical MSC layer and their interactions. The sequential order to view this diagram is to start with the backward path (b_{in} to b_{out}) then the forward path (f_{in} to f_{out}). Diagram (b) shows the data flow of a single vector in a MSC layer, which can be implemented to run in parallel to other .

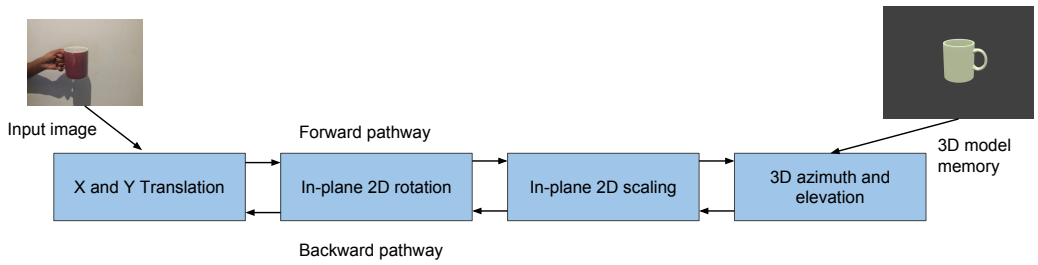


Figure 3.2: Four MSC layers are composed to form a visual MSC. Each layer searches for a specific type of transformation. The input is a 2D image while the memory is a 3D model. Since there is no inverse transformation for 3D to 2D projection, the 3D model is projected into 2D at various azimuth and elevation values and the resulting 2D images are used to create an azimuth/elevation layer that searches for the correct azimuth and elevation.

3.3 MSC Accuracy

The previous section discussed the ordering property of superpositions, how it is exploited in MSC and touched on the failure case known as collusion. To reiterate from the previous section, collusion occurs when vectors in a superposition produce dot products that are too similar and cause the algorithm to converge to incorrect mappings. Assuming the following:

- superposition s formed from a set of vectors V
- a vector $a \in V$ which can be considered a correct mapping.
- a vector $b \notin V$ which can be considered an incorrect mapping.

Collusion occurs when:

$$a \cdot s < b \cdot s \tag{3.5}$$

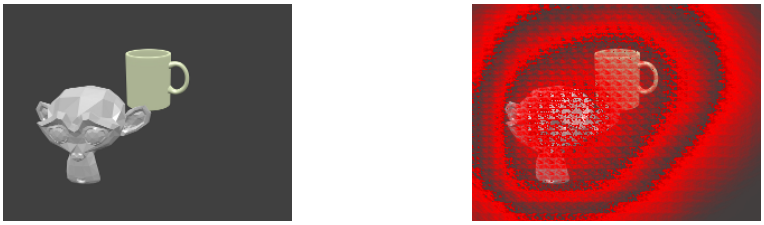
When the above is true the MSC algorithm will converge into the incorrect mappings.

Figure 3.3 illustrates a successful convergence of the MSC described in Figure 3.2 and Figure 3.4 shows the opposite, an incorrect convergence. Both examples show how initially the superposition is composed of a wide range of possible locations and orientations for the mug. Between iterations 1 and 3 the majority of potential mappings are eliminated. This can be seen by the red overlay being localised to around the mug and the monkey in both Figure 3.3 and Figure 3.4. Iteration 3 is when the two examples diverge. In Figure 3.3 the mappings positioning the mug over the monkey are eliminated after iteration 10 and the correct position and orientation of the mug is found eventually. In the incorrect case (Figure 3.4) by iteration 5 the mappings positioning the mug over the mug in the input image has been eliminated ensuring that the eventual convergence would be incorrect. This inaccuracy is caused by collusion.

3.4 Encoding Heuristics

The previous section defined collusion. This section will explore the factors that make collusion more likely and identify heuristics to avoid collusion.

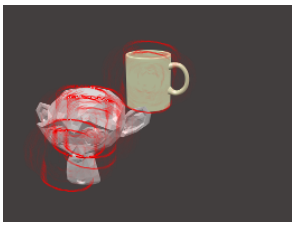
The ordering property requires sparse vectors therefore controlling the sparseness of the superpositions (b_{in}, b_{out}, f_{in} and f_{out} in Figure 3.1) as the MSC converges is crucial. However, it is also a trade off between speed and accuracy. The sparsity of a vector is the percentage of zeros in the vector. The sparseness of superpositions is influenced by two things. The number of transformations/mappings in each layer and the encoding



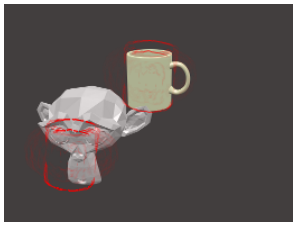
(a) Input image

(b) Iteration 1

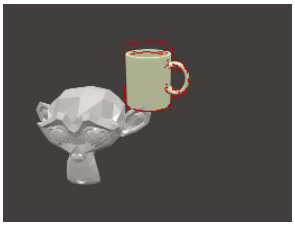
(c) Iteration 3



(d) Iteration 8



(e) Iteration 10

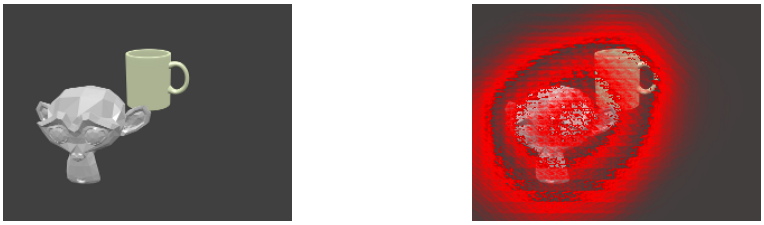


(f) Iteration 21

Figure 3.3: The images shown here are made by overlaying input image with the superposition produced by the backward pathway of the X and Y translation layer at several iterations of the MSC shown in Figure 3.2. The superposition is highlighted in red. This particular run took 21 iterations to converge.

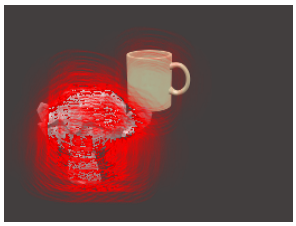
of the input and memory patterns. If the required level of sparseness is not met, the ordering property is likely to fail.

When collusion occurs a false transformation produces a dot product that is greater than the correct transformation. A superposition with low sparsity makes collusion more likely because a it is likely to falsely produce a high dot product with other vectors presented to it. Some simple tests were carried out with sparse vectors to help explore

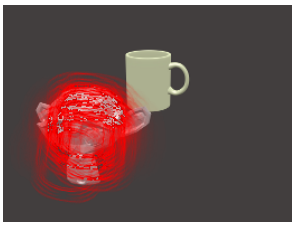


(a) Input image

(b) Iteration 1



(c) Iteration 3



(d) Iteration 5

(e) Iteration 8

(f) Iteration 18

Figure 3.4: The images shown here are made by overlaying input image with the superposition (in red) produced by the backward pathway of the X and Y translation layer at several iterations of the MSC shown in Figure 3.2. This particular run took 18 iterations to converge but it converged to an incorrect solution.

the requirements in terms of vector size, vector sparsity and the number of vectors participating in a superposition for the ordering property to remain true. The tests consisted of randomly generating a number of vectors with specific size and sparsity. The set of vectors is portioned into a set that will form the superposition (set A) and a set that will not form the superposition (set B). After the superposition has been created, dot products between the each vector in both sets and the superposition is

calculated. The percentage of dot products between the set A and the superposition that yield values greater than any dot product between set B and the superposition is the success rate of the ordering property. The results of this experiment is shown in Figures 3.5 and 3.6.

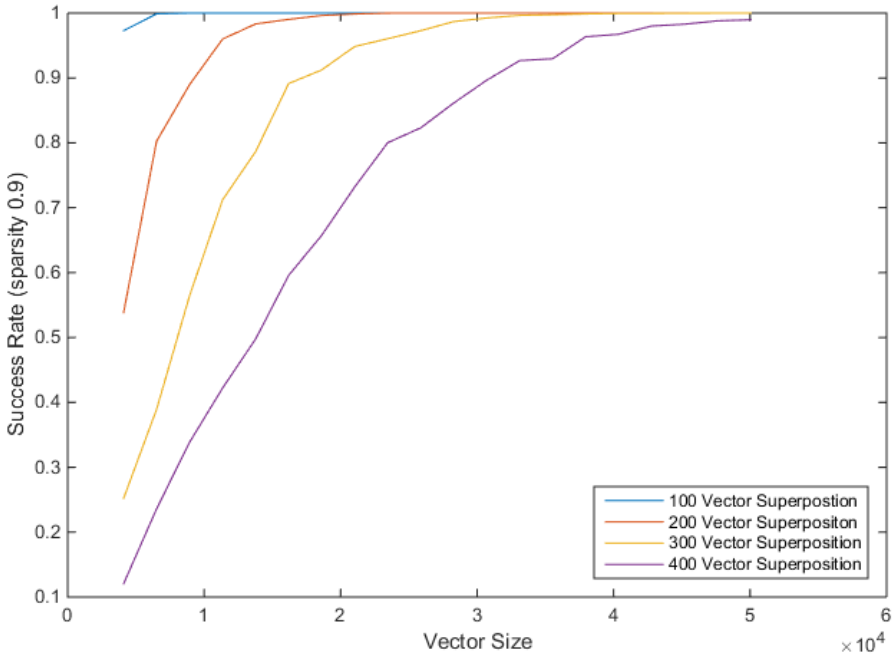


Figure 3.5: This graph show the success rate of the ordering property when increasing the number of vectors used to form the superposition. Vector with a sparsity of 90% were used. As the number of vectors forming the superposition increases, the larger the vector has to be to ensure that the ordering property is successful. The vector sizes vary from 4096 elements to 50052 elements.

The distribution of 1s and 0s within vectors also have an effect on the success of the ordering property. The vectors used in our test were generated randomly and the non-zero elements present within the vectors are uniformly distributed. This is ideal

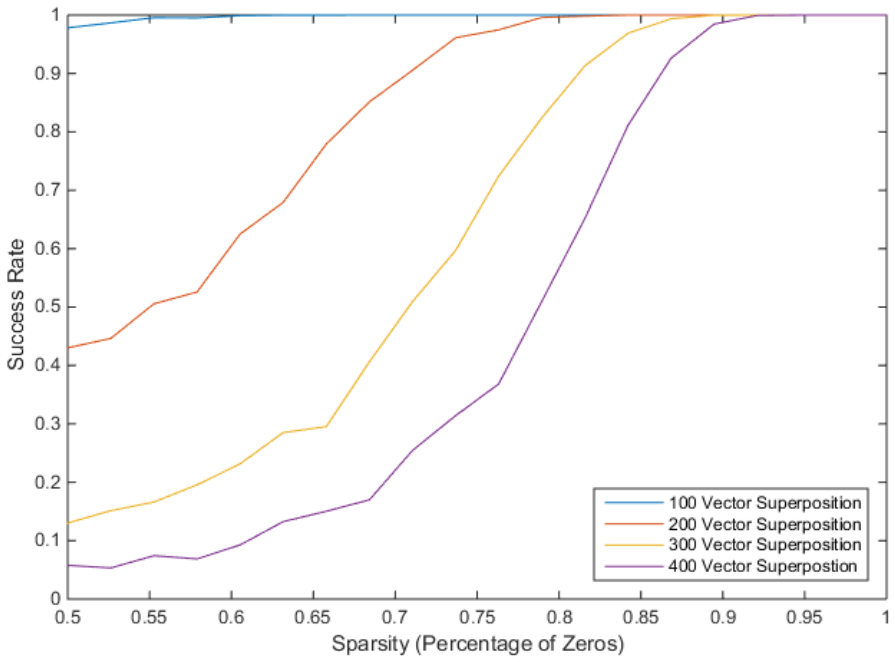


Figure 3.6: This shows the effect of decreasing the sparsity of the vectors (the percentage of zeros) and the number of vectors which form the superposition. The success rate decreases dramatically as sparsity decreases and as more vectors form the superposition. The vector size used for these tests is 50052 elements.

and much like the random vectors involved in VSAs. However it is clear that natural input images will not have non-zero elements uniformly distributed. Further tests that showed that ideally the sparse vector should be uniformly distributed (Figure 3.7).

From the above results four heuristics to follow when selecting sparse encodings can be derived:

- A high number of vectors in the superposition.
- A high cardinality of the vectors.
- A high sparsity of the vectors.

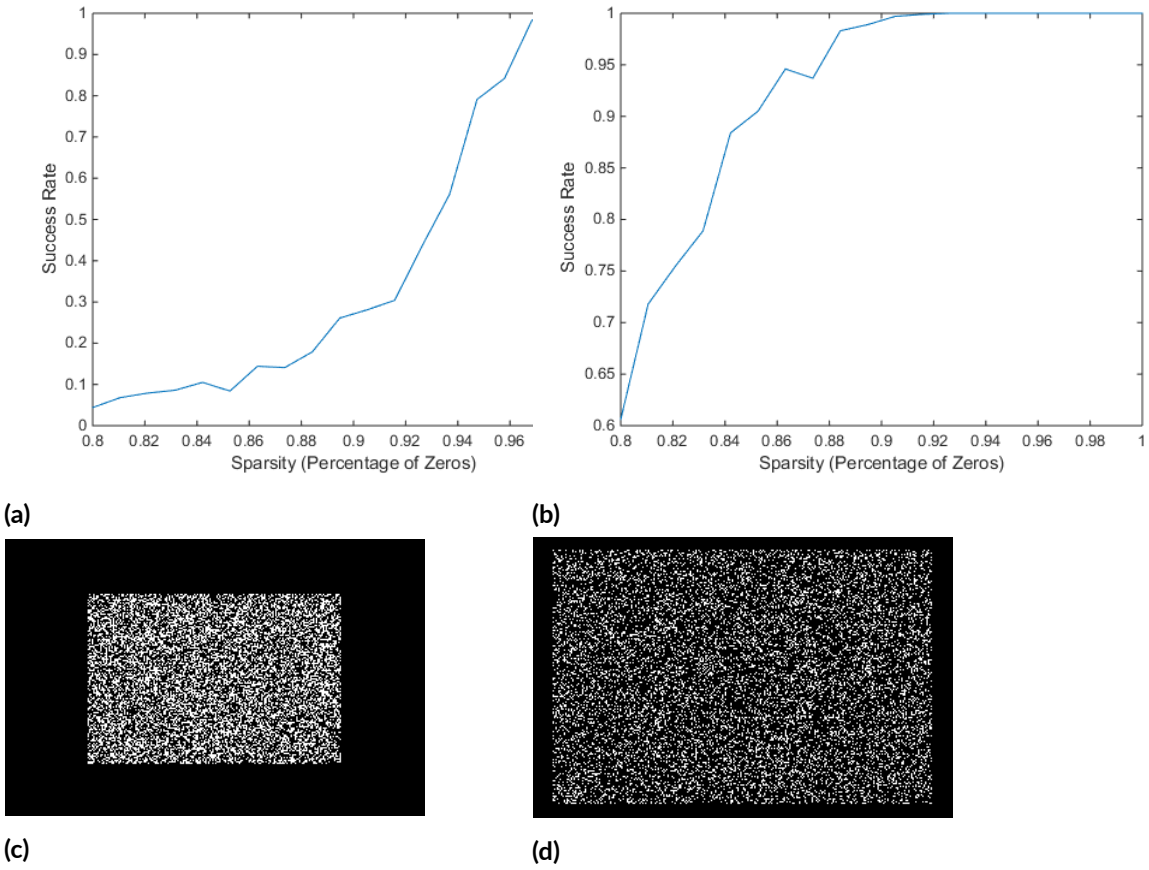


Figure 3.7: The distribution of non-zeros in (a) is such that all of the non-zero elements are distributed in 60% of the space as visualized in (c) while in (b) the non-zero elements are distributed in 90% of the space as visualized in (d). It is clear that the more uniformly distributed the non-zero elements are, the more likely it's that the Ordering Property is successful.

- Distribution of non-zeros in the vectors should be closer to uniform.

However creating an encoding that maximises all four factors is difficult due to the need to have coherent transformations and the nature of the input data. The next section presents ways to tune the MSC along side the above encoding heuristics.

3.5 Tuning MSC

The previous section produced useful information on how to ensure that the ordering property holds. There are many ways to tune MSC and the previous section's results can guide the tuning process. There are several aspects of MSC that can be tuned:

- Which types of layers are present.
- What mappings are in each layer.
- What the values for k and θ (see Equation 3.4) for each layer are. These two variables are well explored in (Gedeon & Arathorn, 2007).
- How to best encode inputs.

This section will describe the above tunable aspects of MSC in detail.

3.5.1 Tuning MSC layers

Before individual MSC layers can be tuned, the correct layers need to be selected for the task. For example for the task of detecting and locating a 3D object four layers are needed:

i 3D azimuth and elevation layer - Has a rendered 2D images of the object at a selection of different azimuth and elevation values.

ii In-plane 2D scaling layer - Has a selection of 2D scaling transformations.

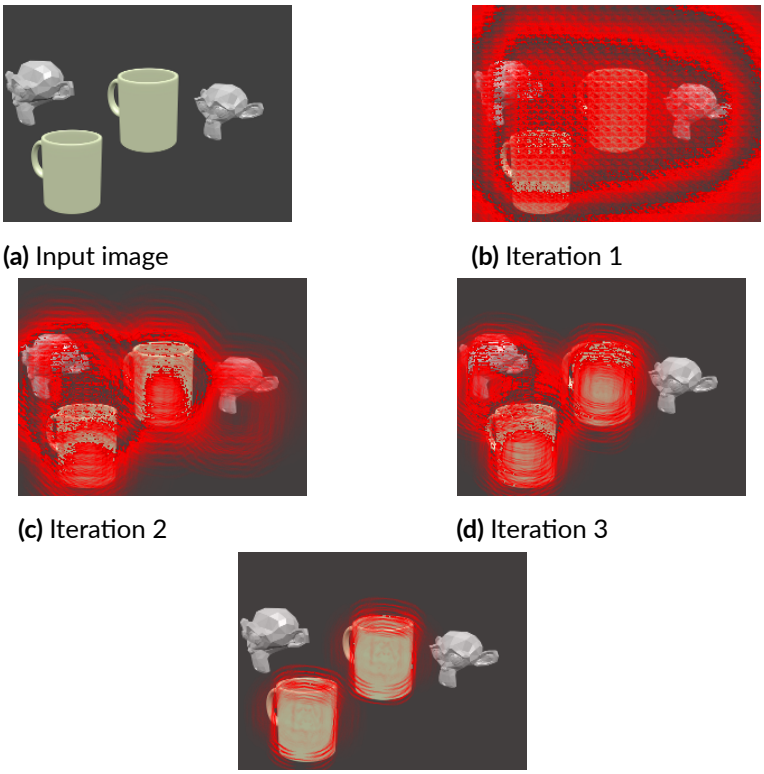
iii In-plane 2D rotation layer - Has a selection of rotation transformations.

iv X and Y translation layer - Has a selection of translations along both X and Y.

The order of the layers is important because transformations are uncommunicative. By placing the scaling and rotation layers before the translation layer, the superposition b_{in} in the translation layer contains all the different rotations and scaling. This enables the translation layer to search for the correct translation while the layers search for other transforms. The order of layers in the list above is from the perspective of the backward pathway (see Figure 3.1), the pathway that transform memory patterns to the input pattern. The order would be reversed for the backward pathway as the transforms would be inversed.

It is also possible to swap out some layers for other types of layers. The transformation layer for example can be split into two separate layers: X translation and Y translation. The major reason to add separate layers is to lower the number of mappings in a layer because (as shown by the results presented in §3.4) fewer mappings increase the chances of the ordering property holding. However, separating X/Y translation layers means that the X/Y translation is not simultaneously converged upon. When a layer that contains both X and Y translations converges onto a correct mapping it actually detects other similar objects in the scene simultaneously. Figure 3.8 shows such a scenario. The advantage of this is that multiple objects can be detected in parallel.

Another type of translation layer uses fast Fourier transforms (FFT)(Arathorn, 2018). Standard MSC translation layer has a number of transformations translating the input image along X and Y axis. For example, create a MSC translation layer that searches from 120px to -120px along the X axis and 100px to -100px along the Y axis.



(a) Input image

(b) Iteration 1

(c) Iteration 2

(d) Iteration 3

(e) Iteration 5

Figure 3.8: When presented with a scenario where two target objects are present a MSC with a X and Y translation layer (like in Figure 3.2) simultaneously detects multiple objects. Usually occurs in the first few iterations after which the X/Y translations corresponding to objects producing lower dot products would be eliminated.

This range of translations has to be discrete. So 5px increments can be used to discretise the range of transformations. The smaller the increment the more transformation the layer would have. It may seem that having a 1px increment would be perfect because all transformations possible within the range can be considered. However there are two disadvantages to doing so:

i The number of transformations required is very high. As these transformations are applied to each black pixel in the input image this slows down the MSC.

ii As the number of mappings in the layer would be very high, the ordering property has a high likelihood of failing (see §3.4).

The standard MSC translation layer can be modified to use FFT instead of having to compute many transformations. FFT is used in two places in the MSC layer. To calculate the dot product (q, see Equation 3.6) for each transformation and to create the superpositions of those transformations (see Equation 3.7).

$$q = ifft(fft(f_{in}) \times fft(b_{out})) \tag{3.6}$$

$$f_{out} = ifft(fft(f_{out}) \times fft(g)) \tag{3.7}$$

Using FFT to calculate q means that all the former dot product values can be calculated without having to do a large number of 2D translations. Figure 3.9 intuitively illustrates the how Equation 3.6 is applied to produce q.

(a) Original image

(b) Image translated by 50,50

(c) The result of Equation 3.6

Figure 3.9: These images show example f_{in}, b_{out} and q vectors. Notice the white dot circled in red in (c), this dot is at (50,50); the offset that was applied to the input image. This q vector is used as in the standard MSC to produce the gating coefficients g. The g are used to create the superposition using FFT as well as seen in Equation 3.7.

3.5.2 Encoding inputs

While MSC layers can be tuned, it is also necessary to pay attention to how to encode input patterns. This is a very difficult task because while maintaining the vector size, sparsity and number of vectors it is also necessary that the encoding supports meaningful transformations. So far the examples presented use an edge detection filter to produce an encoding that is both sparse and preserves the spatial information. However when using MSC layers with a high number of transformation such as the FFT based translation layer described above further thought on encoding is required.

Murphy et al. (2009) use MSC for vehicle detection in synthetic aperture radar imagery mention using oriented edges however they do not describe how those oriented edges are computed. To elaborate, each pixel is associated with one of four orientations. Each orientation put in it's own channel much like RGB colour channels in images. Each channel can be thought of as a vector and since each channel is processed separately the vector superpositions are sparser.

Since Murphy et al. (2009) did not describe how they computed edge orientations, a method based of the Canny filter (Canny, 1986) was developed.

3.5.2.1 Oriented edge encoding

The Canny edge detector can be modified to output oriented edges. There are four steps in the Canny edge detector:

i Smoothing – blurring the image to remove noise

ii Calculating gradients – pixels on edges have large magnitudes

iii Non-maximum suppression – Only local maximums are edges

iv Edge tracking by hysteresis - Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge

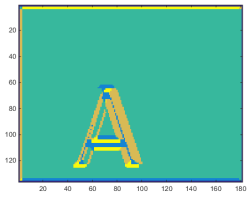
It is step 2 that is relevant because the gradient directions are calculated there. Two sobel filters, one for each axis, is used to produce two filtered images. Equation 3.8 is how Canny calculates the gradient direction at any pixel location.

$$\theta = arctan\left(\frac{|G_y|}{|G_x|}\right) \tag{3.8}$$

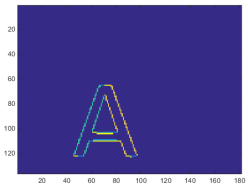
G_y and G_x are the values produced by the two Sobel filters at a pixel. To calculate θ, the magnitude of the two gradients at a particular pixel are treated like the opposite and adjacent sides of a triangle and θ is the angle between them.

In step 3 the θ values are rounded to the nearest 45° and used to select the pixels to compare for non-maximum suppression. The output of step 4 is a binary image where the edge pixels are 1. By step 4 all the information needed to create an oriented edge map is present. By multiplying θ and the binary image produces the gradient directions for the edge pixels and nothing else. Figure 3.10 visualises this process.

(a) Original image



(b) θ



(c) End result

Figure 3.10: These images illustrates the process of producing oriented edges. (b) visualises the values for θ, the output of step 2. (b) is multiplied by the binary image produced at step 4 to produce the oriented edges (c).

3.6 Conclusion

This chapter described the Map Seeking Circuit algorithm in detail. In §3.3 the failure case known as collusion and how it depends on the success of the ordering property was discussed. Exploring this failure case further, the experimental results presented in §3.4 produced four heuristics for MSC input encoding:

- Number of vectors in the superposition.
- Cardinality of the vectors.
- Sparsity of the vectors.
- Distribution of non-zeros in the vectors.

Motivated by these heuristics a methods to tune the MSC layers was presented in

§3.5. And finally, a encoding method derived from the Canny edge detection algorithm was presented in §3.5.2. This is the encoding scheme used in the experiments that will be presented in the next chapter.

4

Energy efficiency via bio-inspired parallelism

4.1 Introduction

In §2.2.3 research into using bio-inspired techniques to produce energy efficient pattern computation was reviewed. One of the characteristics of biological brains that lends to

it's energy efficiency might be the way information is represented (Attwell & Laughlin, 2001), in the form of sparse encodings. The previous chapter examined an approach called Map Seeking Circuits (MSC) that utilised sparse encodings and the highly parallel vector calculations that accompany it. This chapter compares the bio-inspired MSC to a non-bio-inspired approach. The non-bio-inspired approach, referred to as Phase Based Tracker (PBT) (Pauwels et al., 2013), was discussed in §2.1.3.3. PBT makes for a good comparison as it is also a tracking approach that is capable of tracking multiple objects and is implemented in a highly parallel manner; making use of Graphics Processing Units (GPUs). This chapter presents a comparison of a highly parallel bio-inspired approach and a non-bioinspired approach. The two approaches are compared across several aspects: speed, accuracy, generalisability and energy efficiency. The results regarding energy efficiency presented towards the end of this chapter is particularly interesting as it shows how massive parallelism could lead to brain like energy efficiency.

4.2 Comparison of bio-inspired and algorithmic approaches

There are several aspects through which the bio-inspired MSC and the algorithmic PBT can be compared. This section analyses the two approaches in terms of speed, accuracy, generalisability and energy efficiency.

4.2.1 Speed

Both MSC and PBT must take advantage of parallelism to achieve suitable speeds. Figure 3.1 and Figure 2.2 represent the flow of the two algorithms. It is clear that both algorithms have processes that can take advantage of task parallelism and that these tasks are data parallel (see Figure 3.1 for an example of this), forming a hierarchy of parallelism. The extent to which this hierarchy of parallelism can be exploited will have a significant impact on the overall run-time of the algorithm. For example, the implementation of MSC is completely sequential, and averages 6 seconds per frame when run against 196 frames of video from the test data provided by Pauwels et al. (2015). This is several orders of magnitude slower than PBT, which runs at 50 milliseconds per frame.

4.2.1.1 Task and Data Parallelism

In MSC each transform t or t^{-1} may be viewed as separate tasks until $max(q)$ needs to be computed where all the q_i values would need to be gathered form the parallel tasks. After $max(q)$ has been computed, updating the g vector (Equation 3.4) as well as the scaling of the transformed input superposition by g_i can be done in parallel. This process is visualized in Figure 3.1. What Figure 3.1 does not show is the data parallelism that exists within that task. Since the input is a vectorized image and all the operations are vector operations, the task represented in Figure 3.1 can take advantage of per pixel (per

vector element) parallelism; which modern GPUs are optimized to do. Some MSC layers such as the X/Y translation layer can have thousands of transformations. While theoretically all of these transformations may be considered in parallel, limitations of the GPU hardware would mean that very powerful GPUs are required to fully take advantage of parallelism in some MSCs.

PBT makes heavy use of data parallelism but does not have task parallelism to the extent MSC has. Task parallelism exists in the sparse detector (labeled RANSAC detector in Figure 2.2), which runs in a separate GPU. Data parallelism is used to speed up the most demanding portions of the dense tracker, which is in the preprocessing stage.

4.2.1.2 Multiple Target Detection

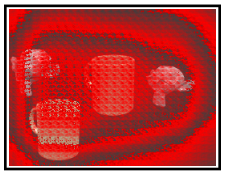
Both PBT and MSC have been extended to detect multiple targets. The more recent PBT paper (Pauwels et al., 2015) discusses multiple target tracking in detail. There is only one paper (Murphy et al., 2013) that describes a multiple object MSC.

In PBT the dense detector processes multiple objects in parallel and independently of each other (interactions are not considered) causing PBT to scale well with the number of objects considered (Pauwels et al., 2015). When the number of tracked objects was increased from one object to 160 objects, the largest change in run-time was approximately 10ms.

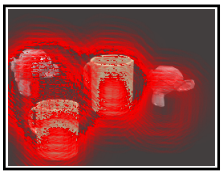
Murphy et al. (2013) detail the mechanism they used to detect multiple targets with MSC. They exploit the way a MSC with a combined X/Y translation layer (as opposed

to two separate layers handling X and Y translation) converges to a single target. Figure 4.1 shows the backward superpositions produced by the X/Y translation layer as it converges. The most likely X/Y coordinates are found within the first few iterations. In Figure 4.1, the MSC may be stopped at iteration 2 or even 1 and parallel "child" MSC searches commenced using a limited selection of transforms corresponding to the three highest g_i in the X/Y translation layer. In MSC the first few iterations are the most expensive, as all the transformations in each layer need to be computed so the "child" MSC searches can run quite quickly.

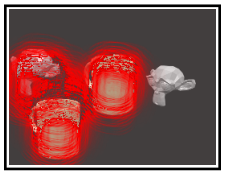
(a) Input



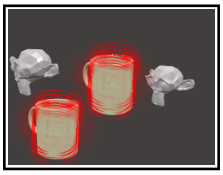
(b) Iteration 1



(c) Iteration 2



(d) Iteration 3



(e) Iteration 5

Figure 4.1: This shows the backward superposition for the X/Y translation layer for an input image with multiple objects. Notice that by iteration 2 the X/Y layer is considering all objects in the scene. It would be ideal at this point to stop the current MSC search and spawn parallel child MSC searches corresponding to a predetermined number of the highest g_i in the X/Y translation layer.

Due to the use of superpositions, MSC searches for multiple types of objects simultaneously without having parallel processes dedicated to each type of object being searched for. The MSC implementation by Overman & Hart (2012) is reported to scale linearly in run-time with increasing numbers of different models in memory. However it must be noted that as the number of objects considered is increased Overman et al.'s MSC scales quite poorly compared to PBT. When the number of objects is increased from 1 to 30 the difference in run-time was approximately 280ms. Overman et al. do not discuss implementation details so it is unclear to what extent the inherent parallelism, discussed in the previous section, was implemented.

One of the main reasons for the difference in amount of computation required between MSC and PBT is number of per pixel operations that needs to be performed. MSC requires much more computation per pixel. This is most prominent in the X/Y translation layer which has the most transformations to compute; the range of X/Y shifts needs to encompass the full width and height of the input image at a fine enough increment that objects are not missed. In PBT most per pixel operations occur in the preprocessing stage where low level dense visual cues are extracted. The preprocessing stage in PBT requires a total of 5010 instructions per pixel (Pauwels et al., 2012) while a count of the instructions for computing the X/Y translations in MSC yielded a total of 98 instructions per pixel per translation considered; the MSC implementation had 525 transformations so this would total 51450 instructions per pixel of input image.

Both MSC and PBT trade speed for accuracy. In MSC the number of transforma-

Table 4.1: The average error in 3D translation (e_T^x,e_T^y,e_T^z) and rotation (e_R^x,e_R^y,e_R^z) from ground truth in MM and Degrees.

	MSC	PBT
e_T^x	5.117	0.2
e_T^y	5.599	0.2
e_T^z	190.039	1.1
e_R^x	32.432038	0.3
e_R^y	39.426942	0.7
e_R^z	1.800000	0.6

tions that each layer has, controls this trade-off. In PBT the memory model complexity and the number of iterations of the M-estimation scheme controls this trade off.

4.2.2 Accuracy

PBT papers (Pauwels et al., 2013, 2015) show that PTB is very accurate in fairly cluttered environments. MSC papers (D. W. Arathorn, 2002, 2014, Overman & Hart, 2012, Murphy et al., 2013) show that MSC can perform well in noisy conditions. However the published MSC experiments are mainly targeted towards recognizing vehicles and it is assumed that the target would always be within certain limited azimuth and elevation values. These values are restricted such that full 6DOF is not considered. When applied to the 6DOF dataset used by Pauwels et al. (2013) tests showed that accuracy is quite low as Table 4.1, Figure 4.2 and Figure 4.3 shows.

To extend the restricted MSCs (D. W. Arathorn, 2002, 2014, Overman & Hart, 2012, Murphy et al., 2013) to handle 6DOF target movement the range of azimuth/elevation would need to be increased. The MSC azimuth/elevation layer does not have the set

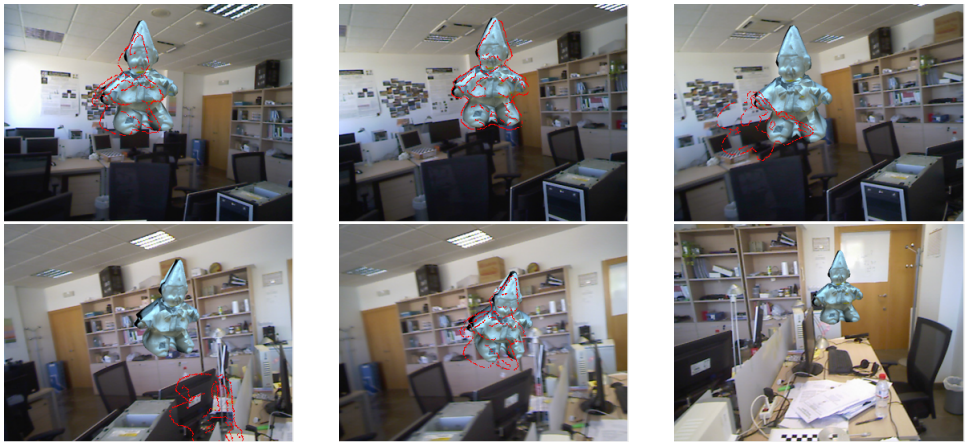


Figure 4.2: The MSC backward superposition outputs for part of the dataset provided by Pauwels et al.Pauwels et al. (2013). Notice that because there is no feedback between frames of MSC, nearby distractors often influence the convergence.

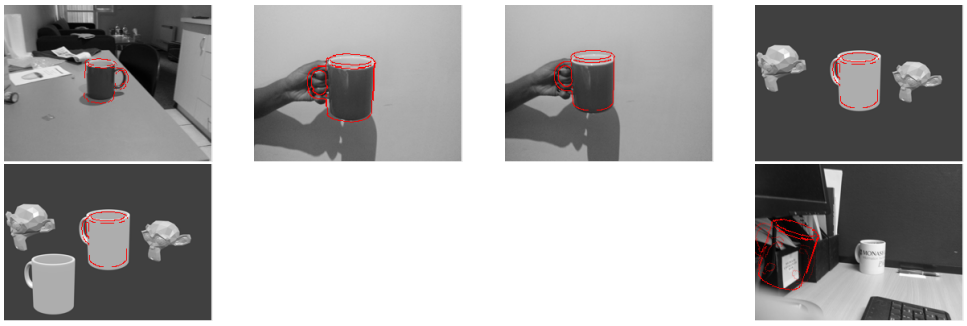


Figure 4.3: The MSC backward superposition outputs for a simpler data-set.

of transformations t that the other MSC layers have. Instead it has a mapping where the 3D memory model is rendered into 2D oriented edge preprocessed images. The increased number of mappings in the azimuth/elevation layer are the likely cause of these inaccuracies.

Another cause for the inaccuracy is the difficulty in developing an effective sparse encoding scheme as discussed in §3.4.

4.2.3 Generalisability

Bio-inspired algorithms are often designed with generalisability in mind. While PBT performs very well in the domain it is designed for, MSC can be applied to a wide variety of problems.

4.2.3.1 Object recognition

Object detection in 3D point-clouds is a very similar problem to the detection of objects in 2D images. PBT generalizes much better into this application area owing to the fact that when applied to 2D images PBT generates a point cloud through the stereo disparity. MSC has been applied to LIDAR data (Overman & Hart, 2012) which requires that MSC works completely in 3D space.

It is possible to adapt MSC to many different types of data. All that needs to be considered is a sparse super-imposable representation for the data and a set of MSC layers that decompose the set of possible mappings required to be searched. Overman

& Hart (2012) describe a MSC based sensor agnostic object recognition tool where the central MSC circuit operates in 3D and several encoders convert different types of sensory input into a form that the central MSC can operate on. They provide results for high range resolution (HRR) radar data (1D), electro optical (EO) imagery (2D) and LIDAR data (3D).

4.2.3.2 Articulated object recognition

Both MSC and PBT have been applied to detect articulated objects. The difference being that whilst PBT requires an extension, MSC only needs an additional MSC circuit.

Pauwels et al. (2014) extends the PBT algorithm for articulated objects. They achieve this by using structural constraints such as hinges or slides. They introduce a new component called the rigidization framework that attaches the object parts such that the constraints are satisfied. The template model is a set of sub-models of the objects parts and an associated kinematic structure.

MSC has been applied to detect people and estimate the pose of their limbs from 2D images. This is implemented by using two MSC circuits, a visual MSC and a inverse kinematic MSC. The visual MSC remains as previously described, except that the 3D memory model it is matching to is constrained by the outputs of the kinematic MSC. The kinematic MSC uses a kinematic structure as its memory model. This kinematic structure is essentially a "skeleton" as used in 3D animation, where each limb is defined by the set of connected vectors (or "bones") in 3D space. These vectors are organised

into several "chains", each arm would be a chain for example. The kinematic MSC (K-MSC) is composed of sub-K-MSCs for each of the chains in the model. The K-MSC for an arm, for example, would have three layers. An arm can be thought of as three connected vectors (upper arm, fore arm and hand) so three MSC layers would be used. Each layer searches for 3D transformations.

4.2.3.3 Motor controls

Articulated object detection is at its core is a inverse kinematics problem, the same type of problem that animals need to solve for limb control. K-MSC has been used to control a physical 6DOF robotic arm reaching a target location in the presence of obstacles (Arathorn, 2015).

K-MSC has also been applied to the motor control of a robotic snake exploring a previously unknown area (Snider & Arathorn, 2006). The novelty of the approach, described by Snider & Arathorn (2006) when comparing with other MSC papers, is that the input is used to adjust the transformations available to the MSC layers. The snake robot has tactile sensors along its body. These sensors allow the robot to detect obstacles whilst touching them. When a obstacle is detected, the set of transformations in each MSC layer is biased so that transformations that take the limb into unobstructed space are ignored.

4.2.3.4 High level path planning

The previous section touched on the ability to bias the transformations considered by MSC with additional information, as a way to fuse information from different sources. D. W. Arathorn (2004) examines how a small pack of wolves stalking a herd of elk. An MSC solution to this problem is quite similar to the robot snake experiment described by Snider & Arathorn (2006). The internal representation of the space being navigated is biased by the perceived probability of detection by the elk. So transformations that take the wolf through a gully, for example, would be converged upon.

4.2.4 Energy analysis

One of the main differences between bio-inspired and processor centric approaches is the high level of parallelism present in bio-inspired approaches. To examine the differences in energy consumption, how energy efficiency varies with parallelism was examined. Focusing on the most time-consuming components of each algorithm. In PBT this is the preprocessing stage with 5010 instructions per pixel (Pauwels et al., 2015). In the MSC implementation the most time consuming stage is computing the transformations in the X/Y transformation layer with 51450 instructions per pixel (98 instructions per pixel per translation with 525 translations).

To model the relationship between number of instructions and energy efficiency a measure for energy per instruction (EPI) is needed. Shao & Brooks (2013) and Gro-

chowski & Annavaram (2006) introduce such a measure:

$$EPI = \frac{p \times \frac{c}{f}}{N} \tag{4.1}$$

where p is the power consumed, c is the number of cycles taken, f is the processor frequency and N is the number of instructions. We simplify this further by deriving the time taken t from $\frac{c}{f}$:

$$EPI = \frac{p \times t}{N} \tag{4.2}$$

When computing the total power consumed for parallel computations, the sum of the p for each sub-process is taken. The simulations show that the most expensive stage of MSC is more energy efficient than the most expensive stage in PBT (Figure 4.4). It also confirmed that MSC performs worse in terms of speed due to the number of per pixel per transformation operations that are required (Figure 4.5). For these simulations both approaches had a equal and constant EPI. N and t depended on the percentage of parallel instructions.

The experimental evidence shown in Figure 4.4 relies upon a property of Equation 4.2:

Property 4.1. *Given two processors, where the time to process N instructions is greater for one processor than the other ($t_1 < t_2$) and the EPI for the faster processor is greater ($EPI_1 > EPI_2$) the slower processor will require less power ($p_2 < p_1$).*

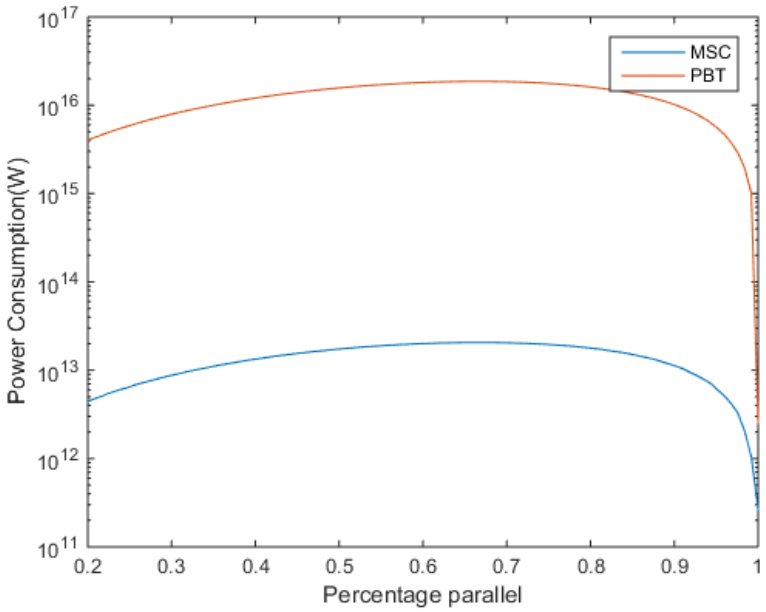
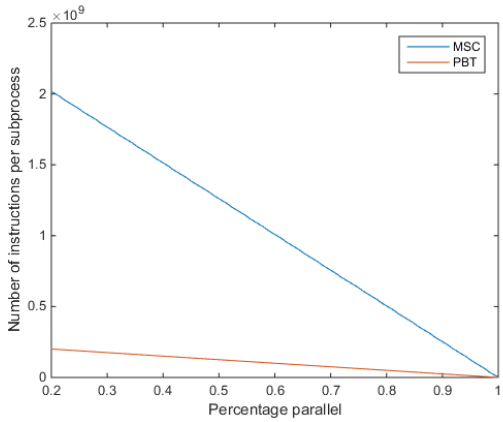
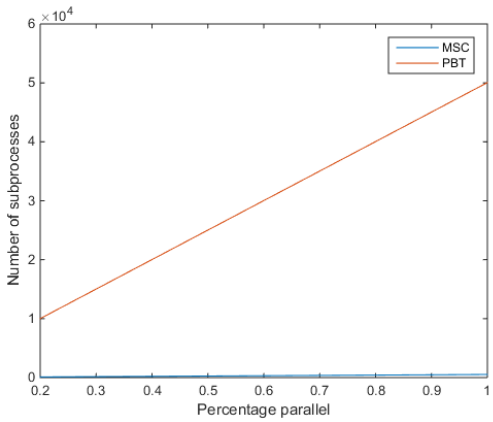


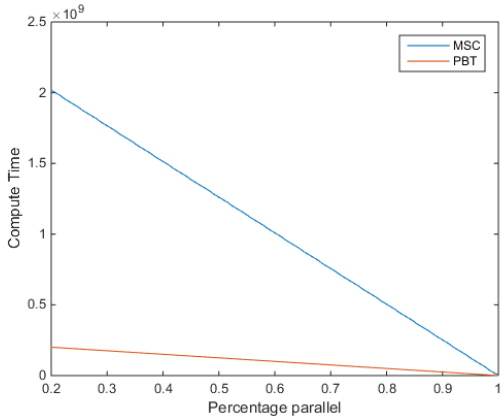
Figure 4.4: MSC consumes several orders of magnitude less energy than PBT. Figure 4.5 supports this result by showing that as parallelism increases, the number of sub-processes increases and the number of instructions per sub-process decreases. The percentage of parallel code in the two approaches has different meanings. In (Pauwels et al., 2015) this meant the number of pixels that were processed in parallel. In MSC it meant the number of transformations that were processed in parallel. Either way a high level of parallelism meant that each parallel process had less instructions to run than lower levels of parallelism.



(a) Number of instructions per subprocess



(b) Number of subprocesses



(c) Compute Time

Figure 4.5: Sub-figures (a) and (b) support Figure 4.4 by showing that as parallelism increases the number of sub-processes increases and the number of instructions per sub-process decreases. Sub-figure (c) shows that when power is kept constant and percentage parallelism is varied MSC is slower than PBT.

Proof of property 4.1. To prove the following: If processor 1 is faster than processor 2, the slower processor will require less power. Assume that both processors will run N instructions, that $EPI_1 > EPI_2$ and $t_1 < t_2$. These assumptions are based on the information reported in Grochowski & Annavaram (2006). Processor 1 can be described by:

$$EPI_1 = \frac{p_1 \times t_1}{N}$$

Processor 2 can be described by:

$$EPI_2 = \frac{p_2 \times t_2}{N}$$

$$\frac{p_1 \times t_1}{EPI_1} = \frac{p_2 \times t_2}{EPI_2}$$

$$\frac{p_2 t_2}{p_1 t_1} = \frac{EPI_2}{EPI_1}$$

Since we assume $EPI_1 > EPI_2$:

$$\frac{p_2 t_2}{p_1 t_1} < 1, \quad \frac{p_2}{p_1} < \frac{t_1}{t_2}$$

Since we assume $t_1 < t_2$:

$$\frac{p_2}{p_1} < 1, \quad p_2 < p_1$$

The power required by processor 2, the slower processor, is less than the faster proces-

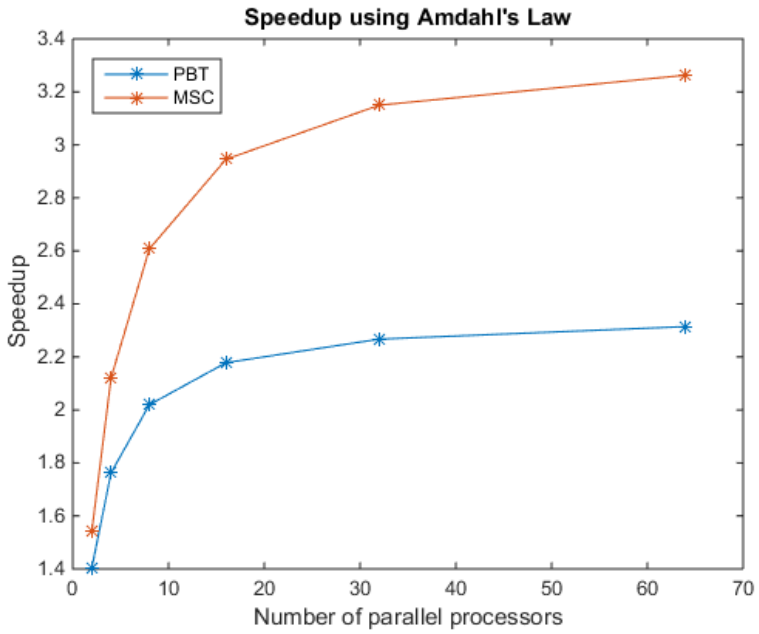


Figure 4.6: Due to the fine grain parallelism in bio-inspired techniques, such as MSC, they scale better with increasing parallelism.

sor 1. □

Amdahl's law (Amdahl, 1967) can be used to show that numerous low power low frequency processors are ideal to run massively parallel applications, such as bio-inspired algorithms. Using pseudo-code representations of both algorithms to estimate the proportion of parallelizable operations, Amdahl's law can be applied to show that the fine grain parallelism present in the bio-inspired MSC allows for greater scalability (Figure 4.6).

However, the scalability that bio-inspired approaches provide may not be attainable with conventional hardware. A study on the energy characteristics of the Intel Xeon

Phi processor (Shao & Brooks, 2013) showed that when varying the number of parallel cores from 1 to a maximum of 60 there is little energy efficiency gains after 10 cores. This was due to the energy overhead of the additional memory accesses.

Non-Von Neumann architectures like neuromorphic processors have been designed specifically for bio-inspired approaches operate at very high levels of parallelism very energy efficiently. IBM TrueNorth, for example, consumes 63mW run visual object detection and classification at 30 frames per second (P. A. Merolla et al., 2014). Further research into neuromorphic architectures would also expose more the generalisability inherent in bio-inspired approaches.

4.3 Conclusions

This chapter shed some light onto why in vivo computations, such as those occurring in the brain, are highly energy efficient while maintaining their ability to generalise to a wide variety of problems. The comparison has shown that although the bio-inspired approach has much more inherent parallelism, it generally runs slower than the algorithmic approach owing to not all inherent parallelism being exploited. The bio-inspired approaches such as MSC are composed of loosely coupled low-complexity sub-processes, which makes it attuned to massive parallelism. Such a parallel implementation will have high CPU-based scalability. The study is also supported by previous studies which showed that in terms of energy per instruction (EPI) using multi-core

processors running at lower frequencies delivers higher performance at an EPI that is approximately four times lower than an equally performing single core processor. Since the complexity of each sub-process in the bio-inspired algorithm is low, many low frequency processors would provide a fast implementation in an energy efficient manner. Other bio-inspired schemes such as LSM (Maass et al., 2002) show that an *on-demand* model of computation facilitates much higher levels of parallelism and lowers complexity, by providing a way to re-use previously computed solutions on-demand.

This chapter also shows that bio-inspired approaches yield better prospects for adapting computation to changing environmental requirements. For example, MSC may be applied to a variety of different problems by replicating circuits and changing the mappings within the circuits.

One important difference between PBT and MSC is how PBT uses temporal information. In fact, it is one of the reasons why MSC accuracy is lower than PBT. Temporal information is also important in animal intelligence. The next chapter develops two approaches to tackle temporal data as well as other problems highlighted in this chapter such as encoding. Additionally, the approaches introduced in the next chapter are applied to a difficult bio-inspired test-bed.

5

Exploiting temporal information for multi-object tracking

5.1 Introduction

In nature many animals such as zebra, herring and starling form swarms. Swarming behaviour has evolved because of the advantages it provides to important functions

such as energy efficient foraging and defence against predators. The defensive aspect of swarming falls into two categories (Jeschke & Tollrian, 2007), namely the dilution effect and the confusion effect. The dilution effect is essentially the idea of "safety in numbers". The risk of predation to the swarming animals is diluted because a predator only attacks a limited number of animals in a group. The confusion effect hinders a predator's ability to select individual targets out of a dense swarm of moving prey by overwhelming the predator's sensory capabilities (Milinski & Heller, 1978, Jeschke & Tollrian, 2007). The confusion effect presents a challenging problem to tackle with computer vision because the swarm is actively trying to decrease tracking accuracy.

There are important practical problems that require tracking solutions that tackle the confusion effect. For example, the use of swarms of Micro Air Vehicles (MAVs) is highly desirable in many scenarios such as search and rescue or exploration. There is a need to detect and track swarms of MAVs for the purposes of determining the swarms topology. This swarm topology can then be used for such tasks as determining "holes" in the swarm's coverage of an area or determining the swarm's underlying communication structure. However to arrive at a swarm topology, the relative positions of all the visible individuals needs to be determined. Accurate tracking is difficult for this task as swarm members can go behind each other, in and out of the view of the camera or the camera it self may get occluded for some time.

This chapter incorporates the learnings from previous chapters such as the sparse-encoding heuristics presented in Chapter 3, the need to maximise parallelism to exploit

the energy efficiency gains described in Chapter 4 and the importance of temporal information as shown by the comparison in Chapter 4. These learnings are applied to the bio-inspired test-bed provided by the task of tracking swarms. Two potential solutions to the problem were developed. Both utilise sparse encoding but each uses the two different interpretations of sparse coding that was discussed in §2.2.1.1, §2.2.2.3 and §3. This provides an interesting comparison between the two interpretations. Both approaches are highly parallel so the energy efficiency gains highlighted in Chapter 4 can be exploited. Additionally both approaches can be used for temporal tasks such as tracking.

5.2 Swarms and Predator Confusion

Biologists have been studying the various aspects of swarming on predator-prey relationships. Predator confusion in nature is composed of four factors:

i Number of individuals in a swarm (Ruxton et al., 2007, Jeschke & Tollrian, 2007).

ii Homogeneity of individuals also known as the oddity effect (Ruxton et al., 2007, Landeau & Terborgh, 1986).

iii Swarm density, it is unclear from the literature whether this is a factor. One experiment supports this (Olson et al., 2013) while another concluded that density is not a factor (Ruxton et al., 2007).

iv Prey agility (Jeschke & Tollrian, 2007).

Out of these four factors the number of individuals in a swarm appears to affect the predator the most (Ruxton et al., 2007, Jeschke & Tollrian, 2007, Olson et al., 2013).

For example, Landeau & Terborgh (1986) showed that when bass hunt minnows the percentage of trails that end in the bass capturing a minnow increases from 11% to 100% when the number of minnow is decreased from 15 to 1. Therefore, experiments should focus on the effect of swarm size on accuracy.

One factor that is universal in any tracking problem even instances simpler than a swarm utilising predator confusion is the temporal nature of tracking. The next two sections introduce two approaches that are temporal and can track a swarm.

5.3 Recurrent Neural Network Tracker

Ondrúška & Posner (2016) introduced a deep recurrent neural network (RNN) based system for tracking multiple objects under occlusion from raw laser scanner data. In this section a 3D RNN derived from the 2D RNN presented by Ondrúška & Posner (2016) is proposed. The network aims to model:

$$P(y_t|x_{1:t}) \tag{5.1}$$

where $y_t \in \mathbb{R}^N$ is a discrete time stochastic process modelling the world around the camera and $x_t \subseteq y_t$ are sensor readings of the world visible to the camera. Ondruska et al. developed their solution for the scenario where a robot equipped with a planar laser scanner is in a busy area with people moving all around it. The network takes a 2D occupancy grid of the scene at t as input and predicts a 2D occupancy grid for

$t + 1$. Since the ground truth for input at t is the frame at $t + 1$, training can be fully unsupervised. The RNN is composed of four 2D convolutional layers. The third layer is kept between timesteps .

To apply this RNN to the task of tracking a swarm in 3D the original 2D network needs to be changed to process a 3D occupancy volume instead of a 2D occupancy grid. Figure 5.1 shows the 3D RNN that was developed for this purpose. Like the 2D version it has four layers with the third layer being retained between timesteps. The number feature maps in the second and third layers was halved due to the increased computational cost of 3D convolutions.

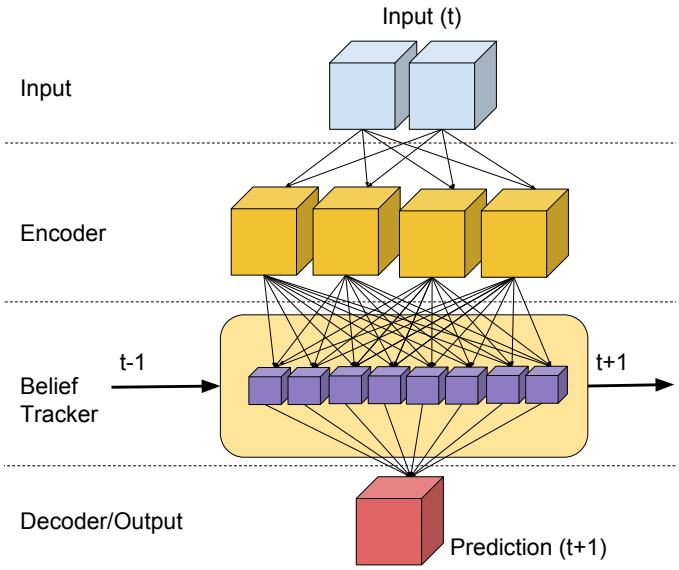


Figure 5.1: The 4-layer recurrent neural network. Layer 3 is used to transfer information to $t + 1$. Each cube represents a feature map. The "Input" layer has two feature maps because the input volume is split into two channels. A channel containing grid cells that are visible to the observer and a channel containing grid cells that are occluded to the observer.

In §2.2.1.1 & §2.2.1.1 the two different interpretations of sparse coding were introduced. A neural networks such as RNN represents the second interpretation of sparse coding. The one presented by Olshausen & Field (1996), where information is represented by a sparse set of basis functions. The next section introduces an approach that uses the other interpretation of sparse coding and also has a completely different method of training to a neural network.

5.4 Sequential Hierarchical Graph Neuron

This section describes two novel improvements to HGN (see §2.2.2.5) that enable processing of sequential data. This new HGN is referred to as Sequential HGN (SHGN). Firstly, a novel training procedure that allows for sequences of memories is presented. Secondly, an optimization to the bias arrays that improves both time complexity and memory usage is presented.

5.4.1 One-shot Sequence Training

Given a sequence of data S, it can be assumed that S_n is associated to S_{n+1} if the data is temporal. Since SHGN is an associative memory the memory of S_n can be associated with S_{n+1}. That way sequences of patterns can be recalled. Figure 5.2 illustrates this new approach.

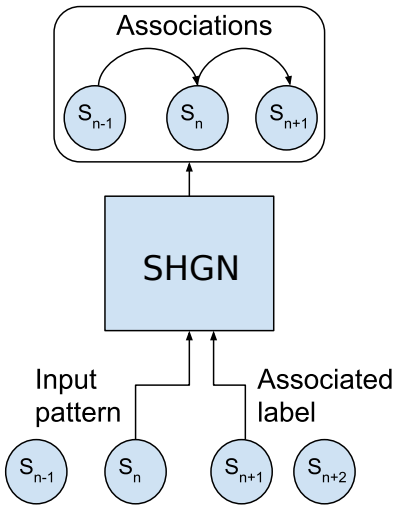


Figure 5.2: The new training procedure associates the frame S_n with the frame S_{n+1} creating a chain of association in its memory.

5.4.2 Improving HGN speed and memory usage

The original HGN is composed of rows of neurons structured such that one row is required for each unique input symbol. When implemented as a multi-threaded program that structure is expensive and has redundant neurons. In the original version Nasution & Khan (2008) the bias array is a linked list which would have a complexity of $O(n)$ to search. A hash based method can be introduced to improve both these aspects of HGN. Several changes to the HGN algorithm were required:

The first change is instead of having one row of GNs in each layer for each unique input symbol, only a single row is ever used. The number of neurons in each layer is equal to the size of the input pattern for that layer.

The second change is instead of sending/receiving the row numbers of activated

GNs the input symbols of neighbouring GNs is sent/received.

The third change is an improved implementation of the bias array. Using a Hash Map as the data structure for the bias array is at the core of this improvement. The key to the map is the hash of the nodes input symbol and the input symbols of the each neighbour. The value is the current size of the hash map minus one. By using a hash map the bias array look up is improved from $O(n)$ to $O(1)$. Additionally using a hash of each nodes symbol and neighbouring symbols eliminates the need for additional rows of GN neurons. Furthermore this approach enables the introduction of additional input symbols without needing to change the SHGN structure. The new algorithm for each GN is described in Algorithm 5.1.

Algorithm 5.1: The algorithm for each neuron in SHGN.

```
Data: input symbol
Result: the pattern index to be passed to the higher layer
Send input symbol to neighbours;
Receive neighbours input symbols;
key = Create a hash of the neighbouring symbols and input symbol;
index = access bias array with key;
if In training mode AND index == NULL then
    Create a new bias array entry with key and current size of bias array +
      1;
Send index to GN in next layer;
```

The last change pertains to the last step in the HGN algorithm, a method of counting the retrieved indices of all the GN's in the HGN to produce a final overall result. In previous work this was done via a simple voting scheme (Muhamad Amin & Khan,

2009) or simply trusting the highest level GN with a non-zero recall (Nasution & Khan, 2008). However those works apply HGN to the task of recognizing letters of the alphabet, the use case being explored here is very unlikely to produce data closely matching the training data. Therefore simply taking the recall of the highest level GN would produce false positives. A simple voting scheme is also unsatisfactory because the higher levels of HGN have a higher importance since they are processing higher level features. To facilitate fuzzy recalling a weighted voting scheme is required:

$$s = \sum_{l=1}^{L} \frac{1}{l_{numNeurons}} \tag{5.2}$$

Equation 5.2 is how a score is calculated for any particular recalled pattern index. In Equation 5.2, L is the total number of layers and $l_{numNeurons}$ is the number of graph neurons in the layer l. This ensures that recalls from neurons higher in the hierarchy contribute more to the total score. The indices recalled at different levels of the hierarchy can be scored and sorted by the score. This is particularly useful when not all GNs produce a recall, which is highly likely when tracking a swarm.

5.5 Experimental Results

This section will present the experiments done to evaluate both the 3D RNN and SHGN. Before covering the experiments and their results , the data generation process and training processes will be described.

5.5.1 Data generation

For the purposes of the experiments presented in this chapter the possibility of obtaining depth footage of swarming animals such as fish was investigated. However, no datasets were available and collection of depth footage is expensive. In lieu of real depth footage, simulations were used. In particular an aggrigation model (Couzin et al., 2002) that embodies key principles believed to underpin the confusion-effect such as high number of individuals Jeschke & Tollrian (2007) and visual homogeneity of individuals Landeau & Terborgh (1986) was used to produce both test data and training data.

The simulation is rendered at 23 frames per second as a point cloud and then converted into an 31x31x31 occupancy grid. The simulation can be run with varying number of swarm members. Figure 5.3 shows a few frames from this simulation. The simulation was restarted with a different starting state every 1000 frames to ensure that the complex behaviour created by the interacting swarm members was captured.

Figure 5.3: Several frames from the swarm simulation. It is colour coded by depth, white is far from the camera and black in close to the camera. The swarm converges onto a torus shape, this is documented behaviour in both simulated and natural swarms (Couzin et al., 2002).

5.5.2 Training Data

The simulation described above was used to produce 3D occupancy grids of various swarm sizes. Sizes ranging from 10 to 120 were used. The RNN consumes 100 frames at a time and 190 sequences (a little over 19,000 individual frames) were produced. The network was trained until the error plateaued which was about 200,000 iterations over the data.

The training data for SHGN is slightly different due to it being an associative memory and not in anyway generative like the RNN. Being generative allows the RNN to generalise to different swarm sizes better than SHGN. The swarm sizes ranged from 1000 to 10,000 which were the swarm sizes used in our evaluation data. The data consisted of 801 frames sampled from 8016 frames by taking every tenth frame; essentially reducing the frames per second. This sampling was done because experimentation showed that the Mean Squared Error (MSE) between each frame in our data was quite low and sampling improved SHGN's accuracy. SHGN consumed 10 frames at a time.

5.5.2.1 Training differences between RNN and SHGN

It is important to compare the differences in training procedure for the two approaches. The RNN required 200,000 iterations over the training data, taking 170 hours on a Nvidia K20 . Due to the one shot nature of SHGN it only requires one pass over the training data so training only took 3.6 hours on a Intel Xeon Phi.

5.5.3 Evaluation Data

Evaluation data for both approaches consisted of a range of swarm sizes: 1000 to 10,000 in increments of 1000 with 5010 frames for each swarm size. Every 1000 frames the simulation was restarted with a different random initial state. As mentioned above, for SHGN this data was sampled by taking every tenth frame.

5.5.4 Findings

Research studying biological swarms Jeschke & Tollrian (2007) suggested that increasing swarm size would lead to decreased tracking accuracy. The experiment's results showed that this is not true for RNNs while it is true for SHGNs. Swarm size does negatively affect accuracy but it does not have a great impact on the accuracy of the RNN. Figure 5.4 & Figure 5.5 shows the accuracy for RNN and SHGN as swarm size increases. The RNN is fairly resistant to increasing swarm size, maintaining a F1 score of approximately 0.8. It is clear from Figure 5.4 & Figure 5.5 that SHGN is heavily affected by increasing swarm size. An interesting trend can be seen in Figure 5.5 where the accuracy of SHGN mirrors the accuracy of natural predators as reported in Jeschke & Tollrian (2007)(Refer to Figure 1 in Jeschke & Tollrian (2007)).

The RNN's predictive accuracy is shown in Figure 5.6. The RNN is capable of maintaining an accuracy above 0.5 F1 upto 2 seconds into the future. The output generated by the RNN is visualized in Figure 5.7.

	RNN	SHGN
Training data size	190 sequences	80 sequences
Training iterations	200,000	1
Runtime per frame	0.03 sec (GPU)	0.07 sec (Xeon Phi)
Total training time	170 hours	3.6 hours

Table 5.1: Due to the temporal sub-sampling of data and one-shot learning SHGN in a much shorter time period and the run time of both approaches is comparable. However as Figure 5.4 & Figure 5.5 shows the RNN performs better in terms of accuracy.

In terms of speed both SHGN and RNN maintain constant runtime even as swarm size increases. SHGN was not implemented with GPU support so SHGN takes 0.07 seconds per frame of data on an Intel Xeon Phi. RNN takes 0.03 seconds per frame on a Nvidea K20 GPU. Table 5.1 summarises the differences in runtime and training time for the two approaches.

5.6 Conclusion

This chapter ties concepts touched on in previous chapters together. The two approaches examined use different interpretations of the idea of sparse coding that was discussed in chapter 3. Both approaches are highly parallel and take advantage of the energy efficiency discussed in chapter 4. This chapter makes four contributions: an existing recurrent neural network was extended for 3D data, performance improvements were made to an existing associative memory, that associative memory was extended to process sequences of data, and the two approaches were applied to a swarm using

the confusion effect. Experimentation showed that the while the associative memory trained much faster the neural network performed better in terms of accuracy. The experiments also showed that the method using the high dimensional vector interpretation of sparse coding (SHGN) had a similar reaction to increasing swarm sizes as real predators. This highlights the need for a strategy for robust tracking in situations where training data is lacking or the time cost of retraining is too high.

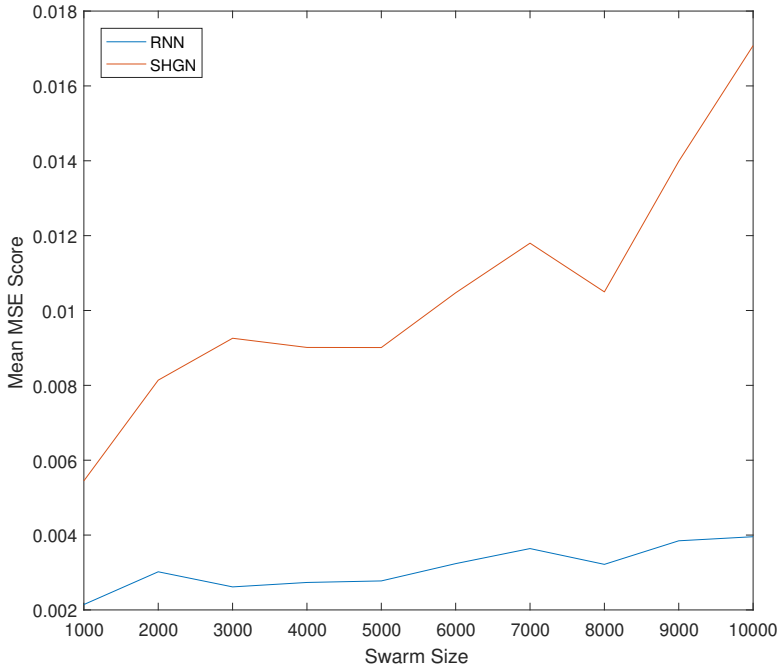


Figure 5.4: The RNN is resistant to increasing swarm sizes but requires 200,000 training iterations over 190 sequences of data while the SHGN required just 1 training iteration over 80 sequences of data. This graph uses the Mean Squared Error.

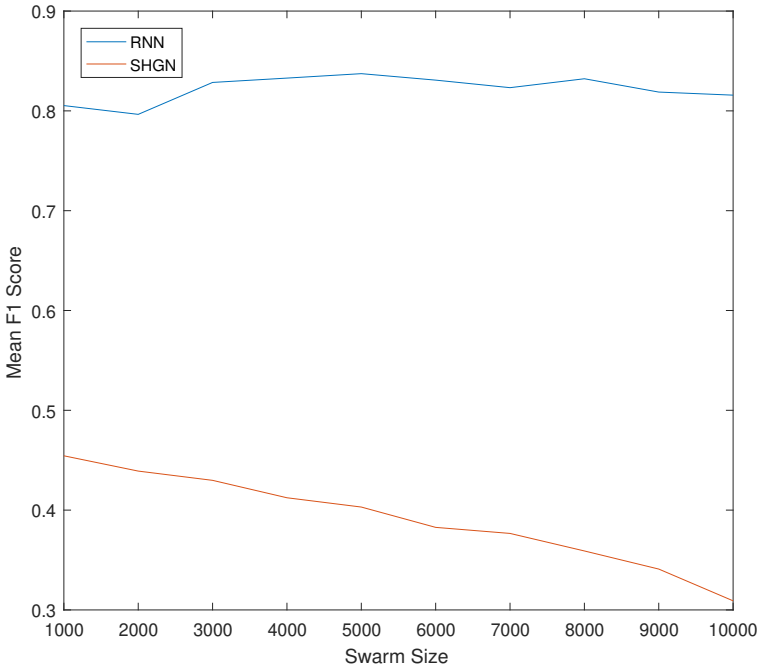


Figure 5.5: The RNN is resistant to increasing swarm sizes but requires 200,000 training iterations over 190 sequences of data while the SHGN required just 1 training iteration over 80 sequences of data. F1 for RNN and SHGN.

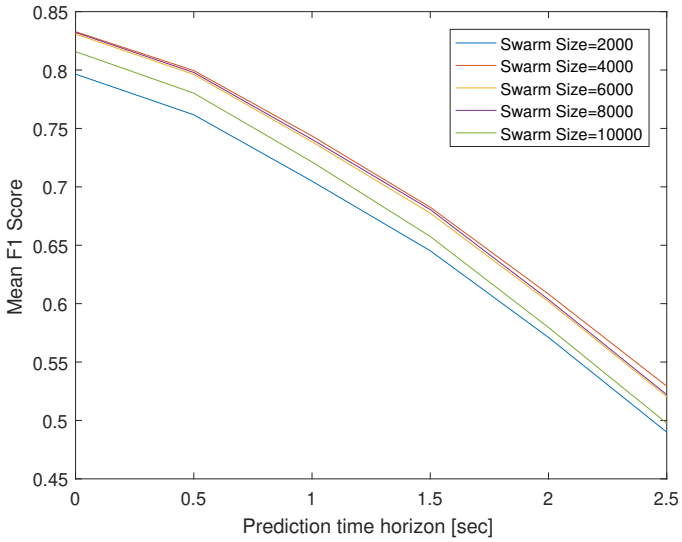
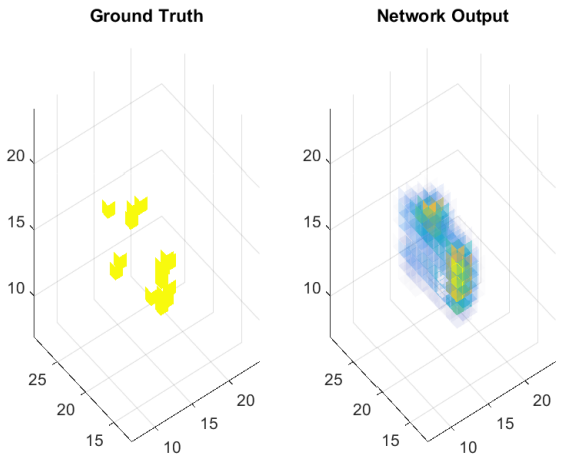
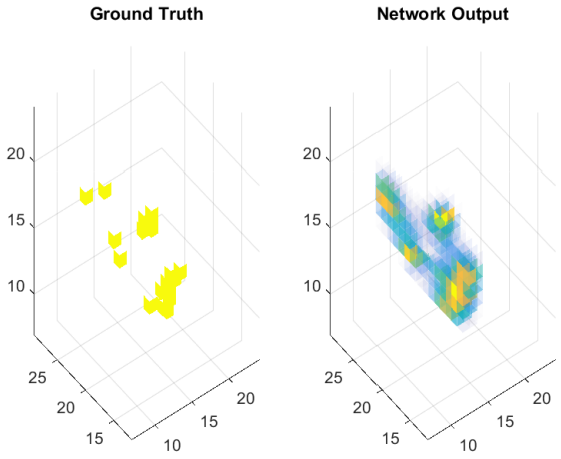


Figure 5.6: The neural network's accuracy at different time horizons. The network maintains an accuracy greater than 0.5 F1 for up to 2 seconds.



(a) Input and output from the RNN for frame 1.



(b) Input and output from the RNN for frame 14.

Figure 5.7: These are two frames of input and the respective output from the RNN for a swarm size of 10,000. The probability of a cell being occupied is color coded, bright yellow being high probability.

6

Conclusion

Looking at animal behaviour it can be observed that many complex actions are driven by a nuanced perception of the environment. This advanced perception is very reliant on how information is represented in the animal's brain. A type of representation that is considered biologically plausible is Sparse Coding (or Sparse Representation). There are two main interpretations of sparse coding in literature. Both interpretations rely on

the redundancy afforded by the numerous neurons in the brain. Both interpretations also lend themselves to highly parallel implementations.

A problem faced by many use cases is energy minimisation. For example in miniature devices such as Wireless Sensor Networks (WSNs), making frequent communications is undesirable due to the associated energy cost. Therefore it is desirable to have energy efficient pattern recognition onboard to help reduce the need to communicate frequently.

6.1 Contributions of the research

The contributions of this thesis are as follows:

- Chapter 3 examined a bio-inspired approach that uses sparse vector representations called Map Seeking Circuits (MSCs). Previous research and experimentation showed that accuracy of MSC is highly dependant on the sparsity of the input vectors. Through further exploration of this failure case four encoding heuristics affect that accuracy where identified.
- Chapter 4 presented a comparative study with a highly parallel bio-inspired pattern recognition approach and a non-bio-inspired pattern recognition approach. This study showed that fine grain parallelism, parallelism resulting from processing sparse vector representations, allowed for significant gains in energy efficiency. These result are published in (Y. R. Hettiarachchige et al., 2018).
- Chapter 5 applied the two interpretations of sparse coding to the bio-inspired problem of tracking a swarm of objects that are actively attempting to confuse the viewer through a phenomena known as the confusion effect. This work had five contributions:
  - Several improvements were made to the HGN approach making it faster while lowering the memory requirement.

- The improved HGN was extended to process sequences of data.
- An existing RNN was extended to handle 3D data.
- The work introduced swarms and the confusion effect as a challenging test-bed for computer vision. The chapter also examined how computer vision techniques fare in this test-bed. It showed that RNN's (which use the basis function interpretation of sparse coding) is fairly resistant to increasing swarm size, maintaining a F1 score of approximately 0.8. On the other hand HGNs (which use the sparse vector interpretation) actually decreased in accuracy with increasing swarm size in a similar way to what is observed in natural predators.
- The work also acts as a comparison between the two interpretations of sparse coding. In terms of training HGN proved to require much fewer training samples and so trained faster. However the neural network performed better in terms of accuracy.

6.2 Future work

The research discussed in this thesis focused on applying bio-inspired representations to pattern recognition problems and exploiting the parallelism afforded by using such representations. Through the course of the research several avenues for future work was also identified:

- The results from Chapter 3 showed that Map Seeking Circuits an approach that utilizes sparse vectors for information representation is highly reliant on how the input data is encoded. Chapter 3 also described 4 heuristics that an ideal encoding should follow. Future research should investigate the possibility of developing an automatic procedure to generate encoding schemes. This task

would benifit from a neural network encoding layer feeding input into a MSC. The MSC's accuracy on the resulting encoding can be the metric for trainig the encoding nerual network. The advantage of this approach would be the interpretability that MSC offers.

- The discussion regarding energy efficiency in Chapter 4 showed that the parallelism afforded by sparse coding actually leads to energy efficiency. Future research can focus on answering the question of whether implementing such a system in hardware is feasible.

- Chapter 5 applied two approaches that use sparse representations to the bio-inspired problem of tracking a swarm of objects that are actively attempting to confuse the viewer. The simulation used only varied the size of the swarm and didn't involve the predator interacting with the swarm. In the future more realistic experiments could be developed or, if possible, real data gathered.

- Building from the previous point, the RNNs approach discussed in Chapter 5 is perfectly suited to apply an adversarial training scheme. Exposing the parameters that control the swarm simulation to a separate trainable system would allow for further improvement of the detector as well as further exploration of the confusion effect.

- In addition to the item above, the results in Chapter 5 also highlighted how different the the two interpretations of sparse coding were. Future work should investigate methods to improve the accuracy of SHGN while maintaining the relatively small training data size. This is a challenging problem as it excludes straight forward solutions such as incorporating a neural network encoding layer that would optimise on the heuristics identified in Chapter 3. This is because a neural network encoding layer would require more training data.

References

Amdahl, G. M. (1967, Apr). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference on - AFIPS '67 (Spring)* (p. 483). New York, New York, USA: ACM Press. Retrieved from http://dl.acm.org.ezproxy.lib.monash.edu.au/citation.cfm?id=1465482.1465560 doi: 10.1145/1465482.1465560

Arathorn, D. (2001, February). Recognition under transformation using superposition ordering property. *Electronics Letters*, *37*(3), 164-165. Retrieved from http://ieeexplore.ieee.org.ezproxy.lib.monash.edu.au/articleDetails.jsp?arnumber=902790 doi: 10.1049/el:20010123

Arathorn, D. W. (2002). *Map-Seeking Circuits in Visual Cognition: A Computational Mechanism for Biological and Machine Vision*. Stanford University Press.

Arathorn, D. W. (2004). From Wolves Hunting Elk to Rubik's Cubes: Are the Cortices Composition/Decomposition Engines? *Proceedings AAAI Symposium on*

Connectionist Compositionality. Retrieved from http://www.aaai.org/Papers/Symposia/Fall/2004/FS-04-03/FS04-03-001.pdf

Arathorn, D. W. (2014). *Map-Seeking Circuit (MSC): A Computational Mechanism for Object Recognition under Transformation with Digital and Analog Implementations.* http://nice.sandia.gov/videos2014.html.

Arathorn, D. W. (2015, March). A System View of the Recognition and Interpretation of Observed Human Shape, Pose and Action. *arXiv e-prints*, arXiv:1503.08223.

Arathorn, D. W. (2018). *General Intelligence Corp MSC implementation.* http://www.giclab.com/implementation.html. (Accessed: 2018-05-01)

Attwell, D., & Laughlin, S. B. (2001, October). An energy budget for signaling in the grey matter of the brain. *Journal of cerebral blood flow and metabolism : Official Journal of the International Society of Cerebral Blood Flow and Metabolism*, *21*(10), 1133–1145. Retrieved from http://dx.doi.org/10.1097/00004647-200110000-00001 doi: 10.1097/00004647-200110000-00001

Avarguès-Weber, A., Dyer, A. G., Combe, M., & Giurfa, M. (2012, May). Simultaneous mastering of two abstract concepts by the miniature brain of bees. *Proceedings of the National Academy of Sciences of the United States of America*, *109*(19), 7481–7486. Retrieved from http://www.pnas.org/content/109/19/7481 doi: 10.1073/pnas.1202576109

Bartolozzi, C., Rea, F., Clercq, C., Fasnacht, D. B., Indiveri, G., Hofstatter, M., & Metta, G. (2011, June). Embedded neuromorphic vision for humanoid robots. In *CVPR 2011 workshops* (pp. 129–135). IEEE. doi: 10.1109/CVPRW.2011.5981834

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J.-M., ... Boahen, K. (2014, May). Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations. *Proceedings of the IEEE*, *102*(5), 699–716. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6805187 doi: 10.1109/JPROC.2014.2313565

Brecht, M., & Sakmann, B. (2002, August). Dynamic representation of whisker deflection by synaptic potentials in spiny stellate and pyramidal cells in the barrels and septa of layer 4 rat somatosensory cortex. *The Journal of Physiology*, *543*(Pt 1), 49–70. Retrieved from http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2290465&tool=pmcentrez&rendertype=abstract

Breiman, L. (2001, Oct 01). Random forests. *Machine Learning*, *45*(1), 5–32. Retrieved from https://doi.org/10.1023/A:1010933404324 doi: 10.1023/A:1010933404324

Brogliato, M. S., Chada, D. M., & Linhares, A. (2014, January). Sparse distributed memory: understanding the speed and robustness of expert memory. *Frontiers in human neuroscience*, *8*, 222. Retrieved from http://www.pubmedcentral.nih.gov/

articlerender.fcgi?artid=4009432&tool=pmcentrez&rendertype=abstract doi: 10.3389/fnhum.2014.00222

Byvatov, E., & Schneider, G. (2003). Support vector machine applications in bioinformatics. *Applied bioinformatics*, *2*(2), 67–77. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/15130823

Canny, J. (1986, Nov). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698. doi: 10.1109/TPAMI.1986.4767851

Carpenter, G., & Grossberg, S. (2010). Adaptive resonance theory. *CAS/CNS Technical Report Series*, 1–23. Retrieved from http://digilib.bu.edu/ojs/index.php/trs/article/view/92

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297. Retrieved from http://link.springer.com/10.1023/A:1022627411411 doi: 10.1023/A:1022627411411

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., & Franks, N. R. (2002, Sep). Collective Memory and Spatial Sorting in Animal Groups. *Journal of Theoretical Biology*, *218*(1), 1–11. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0022519302930651 doi: 10.1006/jtbi.2002.3065

DeWeese, M. R., Wehr, M., & Zador, A. M. (2003, August). Binary Spiking in Auditory Cortex. *J. Neurosci.*, *23*(21), 7940–7949. Retrieved from http://www.jneurosci.org.ezproxy.lib.monash.edu.au/content/23/21/7940

Dyer, A. G., Neumeyer, C., & Chittka, L. (2005, December). Honeybee (Apis mellifera) vision can discriminate between and recognise images of human faces. *The Journal of Experimental Biology*, *208*(Pt 24), 4709–14. Retrieved from http://jeb.biologists.org/content/208/24/4709.abstract doi: 10.1242/jeb.01929

Emruli, B., Sandin, F., & Delsing, J. (2014, July). Vector space architecture for emergent interoperability of systems by learning from demonstration. *Biologically Inspired Cognitive Architectures*, *9*, 33–45. Retrieved from http://www.sciencedirect.com/science/article/pii/S2212683X14000474 doi: 10.1016/j.bica.2014.06.002

Farabet, C., Martini, B., Corda, B., Akselrod, P., Culurciello, E., & LeCun, Y. (2011, June). NeuFlow: A runtime reconfigurable dataflow processor for vision. In *CVPR 2011 workshops* (pp. 109–116). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5981829 doi: 10.1109/CVPRW.2011.5981829

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010, Septem-

ber). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/20634557 doi: 10.1109/TPAMI.2009.167

Frome, A., Cheung, G., Abdulkader, A., Zennaro, M., Bissacco, A., Adam, H., ... Vincent, L. (2009, September). Large-scale privacy protection in Google Street View. In *2009 IEEE 12th International Conference on Computer Vision* (pp. 2373–2380). IEEE. Retrieved from http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5459413 doi: 10.1109/ICCV.2009.5459413

Gayler, R. W. (2003, July). Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience. In *Proceedings of the ICCS/ASCS International Conference on Cognitive Science (13-17 July 2003)* (p. 133-138).

Gedeon, T., & Arathorn, D. (2007, Nov). Convergence of Map Seeking Circuits. *Journal of Mathematical Imaging and Vision*, *29*(2-3), 235–248. Retrieved from http://link.springer.com/10.1007/s10851-007-0028-3 doi: 10.1007/s10851-007-0028-3

Gelperin, A. (2013). Associative Memory Mechanisms in Terrestrial Slugs and Snails. In *Handbook of Behavioral Neuroscience* (Vol. 22, pp. 280–290). Elsevier. Retrieved from http://www.sciencedirect.com/science/article/pii/

B9780124158238000228http://linkinghub.elsevier.com/retrieve/pii/B9780124158238000228 doi: 10.1016/B978-0-12-415823-8.00022-8

Giurfa, M., Zhang, S., Jenett, A., Menzel, R., & Srinivasan, M. V. (2001, April). The concepts of 'sameness' and 'difference' in an insect. *Nature*, *410*(6831), 930–3. doi: 10.1038/35073582

Grochowski, E., & Annavaram, M. (2006). Energy per instruction trends in Intel microprocessors. *Technology@ Intel Magazine*. Retrieved from http://support.intel.co.jp/pressroom/kits/core2duo/pdf/epi-trends-final2.pdf

Harker, S. R., Vogel, C. R., & Gedeon, T. (2007, August). Analysis of Constrained Optimization Variants of the Map-Seeking Circuit Algorithm. *Journal of Mathematical Imaging and Vision*, *29*(1), 49–62. Retrieved from http://link.springer.com/10.1007/s10851-007-0024-7 doi: 10.1007/s10851-007-0024-7

Hebb, D. O. (1949). *The Organization of Behavior* (1st ed.). Psychology Press.

Hettiarachchige, Y., Khan, A., & Barca, J. C. (2018, Nov). Multi-Object Tracking of Swarms with Active Target Avoidance. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (pp. 1204–1209). doi: 10.1109/ICARCV.2018.8581176

Hettiarachchige, Y. R., Khan, A. I., & Barca, J. C. (2018, Jan). Improving energy consumption of pattern recognition by combining processor-centric and bio-

inspired considerations. *Biologically Inspired Cognitive Architectures*, *23*, 54–63. Retrieved from https://www.sciencedirect.com/science/article/pii/S2212683X17300361 doi: 10.1016/J.BICA.2018.01.004

Hopfield, J. J. (1982, April). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, *79*(8), 2554–2558. Retrieved from http://www.pnas.org.ezproxy.lib.monash.edu.au/content/79/8/2554.short doi: 10.1073/pnas.79.8.2554

Hubel, D. H., & Wiesel, T. N. (1962, January). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*(1), 106–154. Retrieved from http://doi.wiley.com/10.1113/jphysiol.1962.sp006837 doi: 10.1113/jphysiol.1962.sp006837

Jaeger, H., & Haas, H. (2004, April). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science (New York, N.Y.)*, *304*(5667), 78–80. Retrieved from http://www.sciencemag.org/content/304/5667/78 doi: 10.1126/science.1091277

Jeschke, J. M., & Tollrian, R. (2007). Prey swarming: which predators become confused and why? *Animal Behaviour*, *74*(3), 387–393. doi: 10.1016/j.anbehav.2006.08.020

Jiao, X., Akhlaghi, V., Jiang, Y., & Gupta, R. K. (2018, March). Energy-efficient neural networks using approximate computation reuse. In *2018 design, automation test in europe conference exhibition (date)* (p. 1223-1228). doi: 10.23919/DATE.2018.8342202

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... Yoon, D. H. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture* (pp. 1–12). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/3079856.3080246 doi: 10.1145/3079856.3080246

Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press.

Kanerva, P. (1994). The Spatter Code for Encoding Concepts at Many Levels. In M. Marinaro & P. Morasso (Eds.), *ICANN '94 SE - 52* (pp. 226–229). Springer London. Retrieved from http://dx.doi.org/10.1007/978-1-4471-2097-1_52 doi: 10.1007/978-1-4471-2097-1_52

Kanerva, P. (2014, September). Computing with 10,000-bit words. In *52nd Annual Allerton Conference onCommunication, Control, and Computing (Allerton)* (pp. 304–310). doi: 10.1109/ALLERTON.2014.7028470

Keeton, W. T., Larkin, T. S., & Windsor, D. M. (1974). Normal fluctuations in the earth's magnetic field influence pigeon orientation. *Journal of Comparative Phys-*

iology, *95*(2), 95–103. Retrieved from http://link.springer.com/10.1007/BF00610108 doi: 10.1007/BF00610108

Khan, A. (2002). A peer-to-peer associative memory network for intelligent information systems. In *ACIS 2002 Proceedings*. Melbourne. Retrieved from http://aisel.aisnet.org/acis2002/6/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Landeau, L., & Terborgh, J. (1986). Oddity and the 'confusion effect' in predation. *Animal Behaviour*, *34*(5), 1372–1380. doi: 10.1016/S0003-3472(86)80208-1

LeCun, Y., Bengio, Y., & Hinton, G. (2015, May). Deep learning. *Nature*, *521*(7553), 436–444. Retrieved from http://www.nature.com.ezproxy.lib.monash.edu.au/nature/journal/v521/n7553/full/nature14539.html doi: 10.1038/nature14539

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. doi:

10.1109/5.726791

LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 253–256). doi: 10.1109/ISCAS.2010.5537907

Lednor, A., & Walcott, C. (1983, January). Homing pigeon navigation: The effects of in-flight exposure to a varying magnetic field. *Comparative Biochemistry and Physiology Part A: Physiology*, *76*(4), 665–671. Retrieved from http://www.sciencedirect.com/science/article/pii/0300962983901275 doi: 10.1016/0300-9629(83)90127-5

Levy, S. D., & Gayler, R. W. (2008). Vector Symbolic Architectures: A New Building Material for Artificial General Intelligence. In *Proceedings of the First Conference on Artificial General Intelligence (AGI-08)* (pp. 414–418). Retrieved from www.cs.wlu.edu/~levy/pubs/agi_2008_levy_gayler.pdf

Lewicki, M. S., Olshausen, B. A., Surlykke, A., & Moss, C. F. (2014, April). Scene analysis in the natural environment. *Frontiers in Psychology*, *5*, 199. Retrieved from http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3978336&tool=pmcentrez&rendertype=abstract doi: 10.3389/fpsyg.2014.00199

Lin, A. C., Bygrave, A. M., de Calignon, A., Lee, T., & Miesenböck, G. (2014, April). Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature neuroscience*, *17*(4), 559–68. Retrieved from http://link.galegroup.com.ezproxy.lib.monash.edu.au/apps/doc/A366082643/AONE?sid=googlescholar&linkaccess=fulltext doi: 10.1038/nn.3660

Maass, W. (1997, December). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671. Retrieved from http://www.sciencedirect.com/science/article/pii/S0893608097000117 doi: 10.1016/S0893-6080(97)00011-7

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560. doi: 10.1162/089976602760407955

Marshall, J., & Arikawa, K. (2014, December). Unconventional colour vision. *Current Biology : CB*, *24*(24), R1150–4. Retrieved from http://www.sciencedirect.com/science/article/pii/S0960982214013013 doi: 10.1016/j.cub.2014.10.025

Martin, S., Rodriguez, P., & Murphy, P. (2009, October). The application of simulated annealing to a map seeking circuit. In *IEEE International Conference on*

Systems, Man and Cybernetics (pp. 3830–3835). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5346615 doi: 10.1109/ICSMC.2009.5346615

McCulloch, W. S., & Pitts, W. (1943, December). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. Retrieved from http://link.springer.com/10.1007/BF02478259 doi: 10.1007/BF02478259

Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., & Modha, D. S. (2011, September). A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In *IEEE Custom Integrated Circuits Conference (CICC)* (pp. 1–4). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6055294 doi: 10.1109/CICC.2011.6055294

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., ... Modha, D. S. (2014, Aug). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, *345*(6197), 668–673. Retrieved from http://science.sciencemag.org/content/345/6197/668.abstract

Merriam-Webster.com. (2018). *Definition for "pattern"*. https://www.merriam-webster.com/dictionary/pattern. (Accessed: 2018-05-01)

Milinski, M., & Heller, R. (1978, Oct). Influence of a predator on the optimal foraging behaviour of sticklebacks (Gasterosteus aculeatus L.). *Nature*, *275*(5681), 642–644. Retrieved from http://www.nature.com/doifinder/10.1038/275642a0 doi: 10.1038/275642a0

Muhamad Amin, A., & Khan, A. (2009). Collaborative-Comparison Learning for Complex Event Detection Using Distributed Hierarchical Graph Neuron (DHGN) Approach in Wireless Sensor Network. In A. Nicholson & X. Li (Eds.), *Ai 2009: Advances in artificial intelligence* (Vol. 5866, pp. 111–120). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-10439-8_12 doi: 10.1007/978-3-642-10439-8_12

Murphy, P. K., Rodriguez, P. A., & Martin, S. R. (2009, October). Detection and recognition of 3D targets in panchromatic gray scale imagery using a biologically-inspired algorithm. In *IEEE Applied Imagery Pattern Recognition Workshop (AIPR)* (pp. 1–6). IEEE. Retrieved from http://ieeexplore.ieee.org.ezproxy.lib.monash.edu.au/articleDetails.jsp?arnumber=5466310 doi: 10.1109/AIPR.2009.5466310

Murphy, P. K., Rodriguez, P. A., & Peterson, C. K. (2013). Detection and Recognition of 3-D Targets in Panchromatic and in Synthetic Aperture Radar Imagery Using a Map-Seeking Circuit Algorithm. *John Hopkins APL Technical Digest*, *31*(3),

234–253. Retrieved from http://www.jhuapl.edu/techdigest/TD/td3103/31_03-Murphy.pdf

Nasution, B. B., & Khan, A. I. (2008). A Hierarchical Graph Neuron Scheme for Real-Time Pattern Recognition. *IEEE Transactions on Neural Networks*, *19*(2), 212–229. doi: 10.1109/TNN.2007.905857

Navaridas, J., Furber, S., Garside, J., Jin, X., Khan, M., Lester, D., ... Yang, S. (2013, November). SpiNNaker: Fault tolerance in a power- and area- constrained large-scale neuromimetic architecture. *Parallel Computing*, *39*(11), 693–708. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167819113001051 doi: 10.1016/j.parco.2013.09.001

Neumann, J. V. (1966). *Theory of self-reproducing automata*. Champaign, IL, USA: University of Illinois Press.

Olshausen, B. A., & Field, D. J. (1996, June). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*(6583), 607–609. Retrieved from http://www.nature.com.ezproxy.lib.monash.edu.au/nature/journal/v381/n6583/abs/381607a0.html doi: 10.1038/381607a0

Olshausen, B. A., & Field, D. J. (2004, August). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, *14*(4), 481–7. doi: 10.1016/j.conb.2004.07.007

Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B., & Adami, C. (2013). Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, *10*(85).

Ondrúška, P., & Posner, I. (2016). Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 3361–3367). AAAI Press. Retrieved from http://dl.acm.org/citation.cfm?id=3016100.3016374

Osipov, E., Khan, A. I., & Amin, A. (2014, June). Holographic graph neuron. In *International Conference on Computer and Information Sciences (ICCOINS)* (pp. 1–6). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6868350 doi: 10.1109/ICCOINS.2014.6868350

Osuna, E., Freund, R., & Girosit, F. (1997, Jun). Training support vector machines: an application to face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (p. 130-136). doi: 10.1109/CVPR.1997.609310

Overman, T. L., & Hart, M. (2012, May). Sensor agnostic object recognition using a map seeking circuit. In *SPIE Defense, Security, and Sensing* (pp. 83910N–83910N–12). International Society for Optics and Photonics. Retrieved from http://

proceedings.spiedigitallibrary.org.ezproxy.lib.monash.edu.au/proceeding.aspx?articleid=1354632 doi: 10.1117/12.917640

Pauwels, K., Rubio, L., Diaz, J., & Ros, E. (2013, June). Real-Time Model-Based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2347–2354). IEEE. doi: 10.1109/CVPR.2013.304

Pauwels, K., Rubio, L., & Ros, E. (2014, June). Real-Time Model-Based Articulated Object Pose Detection and Tracking with Variable Rigidity Constraints. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3994–4001). IEEE. doi: 10.1109/CVPR.2014.510

Pauwels, K., Rubio, L., & Ros, E. (2015). Real-time Pose Detection and Tracking of Hundreds of Objects. *IEEE Transactions on Circuits and Systems for Video Technology*, *PP*(99), 1–1. doi: 10.1109/TCSVT.2015.2430652

Pauwels, K., Tomasi, M., Diaz Alonso, J., Ros, E., & Van Hulle, M. M. (2012, July). A Comparison of FPGA and GPU for Real-Time Phase-Based Optical Flow, Stereo, and Local Image Features. *IEEE Transactions on Computers*, *61*(7), 999–1012. doi: 10.1109/TC.2011.120

Perez-Orive, J., Mazor, O., Turner, G. C., Cassenaer, S., Wilson, R. I., & Laurent, G. (2002, July). Oscillations and sparsening of odor representations in the mushroom

body. *Science (New York, N.Y.)*, *297*(5580), 359–65. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/12130775 doi: 10.1126/science.1070502

Plate, T. A. (1995, May). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641. doi: 10.1109/72.377968

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., & Indiveri, G. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Frontiers in Neuroscience*, *9*(August). doi: 10.3389/fnins.2015.00141

Rebai, I., BenAyed, Y., Mahdi, W., & Lorré, J.-P. (2017, Jan). Improving speech recognition using data augmentation and acoustic model fusion. *Procedia Computer Science*, *112*, 316–322. Retrieved from https://www.sciencedirect.com/science/article/pii/S187705091731342X doi: 10.1016/J.PROCS.2017.08.003

Ruxton, G. D., Jackson, A. L., & Tosh, C. R. (2007, Mar). Confusion of predators does not rely on specialist coordinated behavior. *Behavioral Ecology*, *18*(3), 590–596. Retrieved from https://academic.oup.com/beheco/article-lookup/doi/10.1093/beheco/arm009 doi: 10.1093/beheco/arm009

Schemmel, J., Briiderle, D., Griibl, A., Hock, M., Meier, K., & Millner, S. (2010, May). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of IEEE International Symposium on Circuits and Systems* (pp. 1947–

1950). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5536970 doi: 10.1109/ISCAS.2010.5536970

Schmidt-Koenig, K., & Walcott, C. (1978, May). Tracks of pigeons homing with frosted lenses. *Animal Behaviour*, *26*, 480–486. Retrieved from http://www.sciencedirect.com/science/article/pii/0003347278900659 doi: 10.1016/0003-3472(78)90065-9

Shao, Y. S., & Brooks, D. (2013, September). Energy characterization and instruction-level energy model of Intel's Xeon Phi processor. In *International Symposium on Low Power Electronics and Design (ISLPED)* (pp. 389–394). IEEE Press. Retrieved from http://dl.acm.org.ezproxy.lib.monash.edu.au/citation.cfm?id=2648668.2648758

Simard, P., Steinkraus, D., & Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of Seventh International Conference on Document Analysis and Recognition* (Vol. 1, pp. 958–963). IEEE Comput. Soc. Retrieved from http://ieeexplore.ieee.org/document/1227801/ doi: 10.1109/ICDAR.2003.1227801

Snider, R. K., & Arathorn, D. W. (2006). Terrain discovery and navigation of a multi-articulated linear robot using map-seeking circuits. In *Proc. SPIE* (Vol. 6229,

pp. 62290H–62290H–11). Retrieved from http://dx.doi.org/10.1117/12.663721 doi: 10.1117/12.663721

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, *59*(236), pp. 433–460. Retrieved from http://www.jstor.org/stable/2251299

Vinje, W. E., & Gallant, J. L. (2002, April). Natural stimulation of the nonclassical receptive field increases information transmission efficiency in V1. *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience*, *22*(7), 2904–15. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/11923455 doi: 20026216

Watson, J. D., & Crick, F. H. C. (1953, Apr). Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature*, *171*, 737. Retrieved from http://dx.doi.org/10.1038/171737a0http://10.0.4.14/171737a0

Wright, J., Mairal, J., Sapiro, G., & Huang, T. S. (2010, Jun). Sparse Representation for Computer Vision and Pattern Recognition. *Proceedings of the IEEE*, *98*(6), 1031–1044. Retrieved from http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5456194 doi: 10.1109/JPROC.2010.2044470

Zentall, T. R., Wasserman, E. A., & Urcuioli, P. J. (2014, Jan). Associative concept learning in animals. *Journal of the Experimental Analysis of Behavior*, *101*(1), 130–

151. Retrieved from http://doi.wiley.com/10.1002/jeab.55 doi: 10.1002/jeab.55

Zia, M. Z., Stark, M., Schiele, B., & Schindler, K. (2013, November). Detailed 3D representations for object recognition and modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(11), 2608–2623. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/24051723 doi: 10.1109/TPAMI.2013.87

Zia, M. Z., Stark, M., & Schindler, K. (2014a, June). Are Cars Just 3D Boxes? Jointly Estimating the 3D Shape of Multiple Objects. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3678–3685). IEEE. Retrieved from http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=6909865 doi: 10.1109/CVPR.2014.470

Zia, M. Z., Stark, M., & Schindler, K. (2014b, Nov). Towards Scene Understanding with Detailed 3D Object Representations. *International Journal of Computer Vision*. Retrieved from http://link.springer.com/10.1007/s11263-014-0780-y doi: 10.1007/s11263-014-0780-y