Information Sciences 600 (2022) 323–341

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Cognitive decisions based on a rule-based fuzzy system

Xin Yuan, Michael John Liebelt, Peng Shi *, Braden J. Phillips

the School of Electrical and Electronic Engineering, the University of Adelaide, Adelaide, SA 5005, Australia

ARTICLE INFO

Article history:
Received 10 June 2021
Received in revised form 5 February 2022
Accepted 30 March 2022
Available online 4 April 2022

Keywords:
Artificial general intelligence
Fuzzy logic
Agent-based systems
Cognitive computation
Rule-based decisions

ABSTRACT

We develop an agent-based artificial general intelligent system that can be implemented in compact and power-efficient electronic hardware. The hardware under development is called the Street Engine, which is a hardware-based cognitive architecture for implementing agent-based artificial intelligence. In this paper, we introduce an agent-based system to replicate simple cognitive behaviours. In the processes of this system, numerical data are converted into fuzzy symbolic representations of the surrounding environment, and reasoning rules are included in a modified Fuzzy Inference System to support the cognitive decision-making. We use a case study example, the homing behaviour of the honey bee, to demonstrate constructing production rules and implementing the cognitive and reasoning capabilities of agents. The low level cognitive behaviour is converted into a rule-based fuzzy system, and hardware-based experiments have been conducted to verify the effectiveness of the proposed technique.

1. Introduction

For more than half a century, Artificial Intelligence (AI) research has made a great impact on a wide range of industries. Particularly in the recent decade, machine-learning (ML) approaches [1,43,5] have been successfully applied in practical commercial applications. In ML approaches, knowledge and information are represented as numerical data [23]. ML uses feature extraction and classification techniques to recognise patterns in the data and use these as the basis for classification or prediction future data inputs. This technique has been applied with great success in applications such as imaging processing [24,13] and big data analysis [28].

The "black box" processing of ML creates opaque algorithms and does not work so well in applications in which the input data cannot be characterised by frequently occurring patterns, for example when the machine enters an unfamiliar environment. In these circumstances, a rule-based approach such as implemented in a cognitive architecture shows advantages [22,14,48]. However, the principal difficulty with cognitive agents is that, as more advanced features are required, the amount of data acquired and the lifetime of the agents increase. The size of the real-time working memory increases rapidly with more advanced agents. Consequently, the lifetime and complexity of the agents are limited by the memory capacities and the performance of platforms on which they are run. To increase performance, the only practical options are to increase memory capacity or processor performance or more likely both. However, there are practical limits to what can be achieved this way, and the capacity of current agents falls short of human-level intelligence. Therefore, we are considering a new

* Corresponding author.
E-mail addresses: xin.yuan@adelaide.edu.au (X. Yuan), michael.liebelt@adelaide.edu.au (M.J. Liebelt), peng.shi@adelaide.edu.au (P. Shi), braden.phillips@adelaide.edu.au (B.J. Phillips).

https://doi.org/10.1016/j.ins.2022.03.089
0020-0255/

approach to the implementation of agents, with new processing methods and data structures, which are more amenable to implementation and scaling in microelectronic hardware.

We have devised a new type of computational processor with low power consumption, which we call the Street Engine [11]. This new processor is intended to implement a cognitive architecture and to be used for cognitive agents in autonomous devices. The agent is coded in a customised rule-based language based on OPS-5 [10], called the Street Language [11,33,47].

To demonstrate the feasibility of the Street Language/Street Engine approach to the implementation of cognitive agents, we have sought to replicate a simple form of cognitive behaviour found in nature. Neuroethological research [30] and training experiments of insects [15,35,7,45] show that cognitive behaviours are the integration of serial multiple natural reactions, which are lower levels of stimuli triggered by a part of a hard-wired neural system. Even with a simple neural system such as found in insects, advanced structures in their neural networks integrate the functions of different sections together to achieve a range of cognitive behaviours. We choose to build a Street agent-based system to replicate the homing behaviour of honey bees and use this project to demonstrate that it is feasible to build agents with simple cognitive behaviours using the Street rule-based cognitive architecture.

In this paper, Section 2 introduces basic principles and general logic of the honey bee cognitive behaviours, and Section 3 presents our novel processing architecture. The details of the implemented system architecture, data sensing and representation and cognitive logic are given in Section 4. To show the performance of the proposed new approach, the system behaviours are analysed and evaluated in Section 5.

2. Honey bee behaviour

Honey bees are believed to be one of the few types of insects that are capable of some low-level cognition [30,35]. The research of Fry et al. [12], Menzel et al. [32] and Labhart et al. [21] has shown that honey bees exhibit cognitive behaviours including homing behaviour using landmark references and exchanging knowledge of the paths to the food sources through what is known as the ‘‘waggle dance” during which honey bees move with special patterns to explain their experiences on the path to a food source [41]. It is thought that honey bees are able to distinguish significant visual characteristics from the background and record them as knowledge without recognition and understanding. The ‘‘waggle dance” generates coded messages that describe distances, directions and landmarks towards a food source, and shows that their knowledge representations are symbolic [41]. In this paper, we use the homing behaviour after foraging for food as a case to study on which to base the implementation of a simple cognitive behaviour with our Street system.

2.1. Sensors

Honey bees have multiple sensory inputs providing information about their environment, which they commit to memory [2]. The major sensors of a bee are their optical vision [7,37], olfactory sensors [38] and, according to some researchers, magnetoreceptors [42].

The visual inputs provide some capability of recognising specific objects [18] in the environment and registering [45] the position of the detected objects. However, identifying and measuring objects detected visually only provides basic knowledge about them. It appears that bees do not understand the nature of an object nor its exact location [29].

It is the olfactory sense of the honey bee that is used to identify a food resource. However, in our simulated system, we have chosen not to include an olfactory sensor, because we are focusing on reproducing the homing behaviour rather than foraging for and identifying the food.

Some biologists believe that an additional sensor in the honey bee’s abdomen exists, which they call the magnetoreceptor [42]. They hypothesise that the magnetoreceptor perceives changes of the magnetic field intensity and direction in the bee’s present location. Using this sensor to determine its bearings with respect to the geomagnetic field, the honey bee can recognise its direction of movement, and, by memorising this, have more information available to help it navigate to retrace its movements.

2.2. Foraging

Foraging behaviour is one of the essential and natural survival skills of most animals [8]. Foragers are challenged with investing time and energy into locating food. In the case of the honey bee, the worker bees explore the environment to find and harvest crops for the entire colony. During foraging, they fly in an unstructured pattern to explore the environment within a certain distance of their nest, to discover flowers with a rich stock of nectar and pollen [9]. With this technique, finding food becomes a random event with a low probability of success, and the individual investment in time and energy is high. However, in order to reduce the cost of the return trip and maximise the opportunity for other workers to locate the food source, the successful forager memorises the route of the foraging trip.

During free exploration, the forager uses its visual sensors to recognise objects in the environment and detect any significant changes to identify landmarks [32]. It memorises the sequence in which landmarks were encountered to create a chain of memories about the path taken [18,29,31]. The events of detecting landmarks are recorded as episodic memories, and the

sequence in which they appear links them together. The forager can use these episodic memories to return to the hive, and this increases the chance of returning home successfully. Also, knowledge of the path to the food can be communicated to other workers through the encoded symbolic message of waggle runs [45] to maximise their chance of also finding the food.

2.3. Homing

After engaging in the foraging behaviour introduced above, the honey bee recalls the recorded episodic memories to track back to its hive [9]. It initiates a small range of exploration from the current position to locate the last seen reference landmark. Until it receives the same visual inputs as a previous memory and matches a similar sense of the magnetic field, it starts to look for the next visualised landmark. By repeating these procedures, it can return to the hive along the same path, which it memorised during the free exploration foraging phase.

To summarise this behaviour from an engineering perspective, the foraging honey bee explores an unfamiliar environment and tries to find particular targets, usually food sources, by random motions. On this trip, it detects signature patterns at each reference position and memorises a sequence of them. All of this information is stored, presumably without understanding, and the information can be transmitted to other agents through symbolic communication, namely the waggle dance, so that others can use the data to find the same path.

3. The street engine

Our group is currently developing a new type of computer processor based on a parallel processing rule-based system. It is designed so that it can be implemented as a power-efficient processor to be embedded in autonomous hardware. The language that we use to express system behaviour is called the Street Language and the hardware architecture that supports execution of this language is called the Street Engine [11].

3.1. The street engine

The Street Engine [11] comprises multiple processing elements. All of these elements are tasked with evaluating production rules, and they communicate through a shared Working Memory (WM) using tokens.

3.2. The street language

Based on OPS-5 production rules [10], a Street Language program comprises a set of rules. Each rule consists of conditions and actions, which can be both positive and negative conditions. A positive condition is satisfied if the WM contains at least one element that matches the condition, and a negative condition is satisfied if WM does not contain an element that matches the condition. Whenever all of the conditions of a rule are satisfied, its list of actions is triggered. In our model of execution, all rule conditions are evaluated continuously and rules can be triggered concurrently if their conditions are satisfied at the same time.

Rule conditions are expressed as pattern matches on the attributes of symbolic elements stored in WM, and the actions of the rules will, in general, make changes to the WM, which may trigger the execution of further rules. Specifically, actions will either create some new WM elements or remove some existing WM elements.

In the Street Engine, condition element matching is performed by hardware structures which we call “productors”. In principle, the number of attributes that are carried in an element is unlimited, and the capacity of the entire WM is unlimited. In practice, however, both are limited by the physical storage allocated. In this architecture (Fig. 1), the size of the processor only depends on the number of rules that the agent contains.

3.3. Working memory

The Street Engine uses a real-time updating WM to store information, and data are stored in the form of symbolic attributes in Working Memory Elements (WMEs). Fig. 2 shows an example, with pre-defined WM contents in the left-hand box and a Street language rule in the right-hand box. In this example, the WM initially contains 20 elements, listed in the left-hand column.

3.4. Working memory elements

A WME is a tuple containing an unlimited number of whitespace-separated symbols contained within parentheses. Symbols may contain alphanumeric characters, and purely numeric symbols are interpreted as integers. Fig. 2 shows some examples of WMEs, within the left-hand side box. WMEs are created by the actions of rules and are then available for matching with the conditions of the other rules. WMEs may also be deleted by the negative actions of rules.

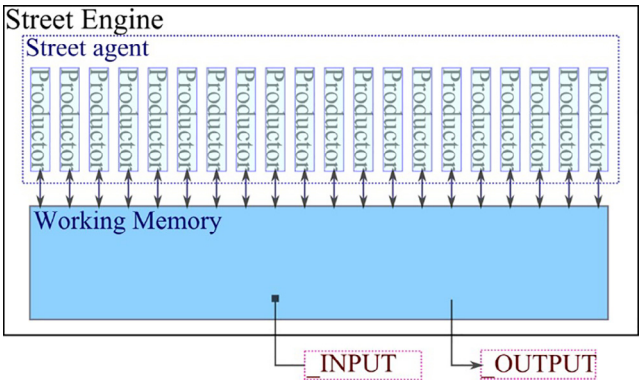

Fig. 1. The Street Engine.

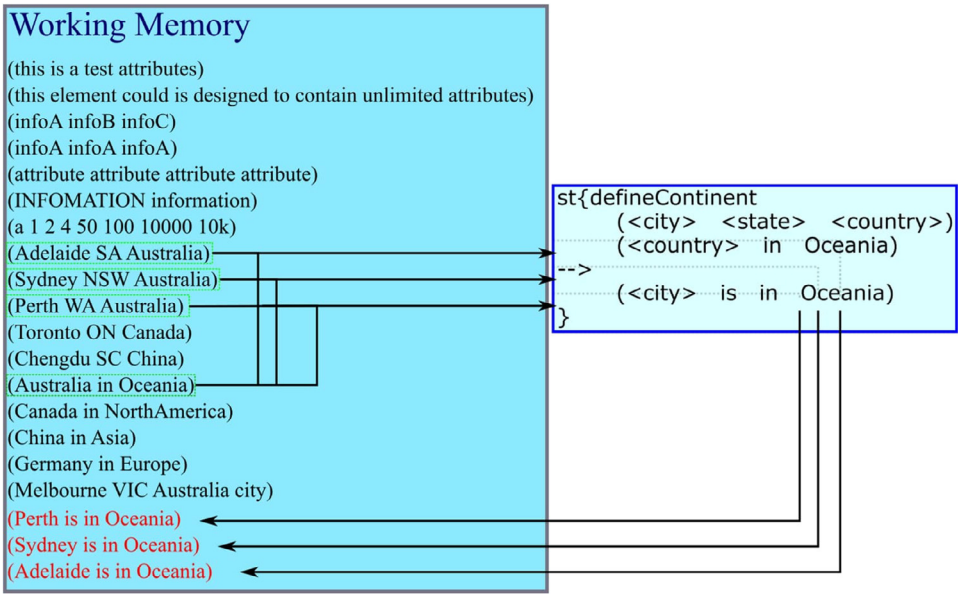

Fig. 2. A example of Working Memory.

3.5. Street language syntax and semantics

The smaller box on the right of Fig. 2 is an example production rule of Street. In Street Language, each rule begins with the reserved character sequence "*st{*" and ends with the matching closing brace, "}". The symbol following "*st{*" is the rule name, in this case, "*defineContinent*". This is followed by a sequence of conditions and a sequence of actions, separated by the reserved character sequence "−− >" (for rules that allow multiple triggering) or "∼∼>" (for single-triggered rules). Individual rules and actions are contained in parentheses. All conditions must be satisfied for the rule to trigger, and once a rule triggers all actions in the rule will be executed, which may (a) create new WMEs or (b) delete existing WMEs. It is important to note that the actions will not necessarily be executed in the order listed in the rule's action set.

In order for a rule to trigger, each of its conditions must match an existing WME. A rule will match a WME if it contains the same number of attributes as the element, and all literal symbols (those not contained in angle brackets, "<>") exactly match the attributes of the WME in the corresponding positions. A formal parameter, contained in angle brackets matches any attribute of a WME if all of the literal symbols match. Then in subsequent rules, the formal parameter will only match a WME attribute with the same value. That formal parameter will be assigned the same symbolic value in the rule's actions.

An action contained in parentheses containing a set of attributes (either literal symbolic values or formal parameters from the rule matching process) will result in the creation of a new WME with those attributes. However, if the left parenthesis is preceded by a "–" then the WME with those attributes will be deleted, if it already exists.

For example in Fig. 2, the production rule, *defineContinent*, contains two conditions. It will be triggered if, in the WM, there are two three-attribute WMEs in which both have the same attribute, < *country* >, as first or third attribute. The one with

< *country* > at the first place has last two attributes "*in*" and "*Oceania*" respectively, and the one with < *country* > at the third place has any two attributes that are represented as < *city* > and < *state* >. The action will create a new WME with the first attribute set to < *city* >, then, "*is*", "*in*" and "*Oceania*" as the remaining attributes.

In this WM example, the WME, (*Australia in Oceania*), together with (*Adelaide SA Australia*), (*Sydney NSW Australia*) and (*Perth WA Australia*) trigger the rule, *defineContinent*. Thus, new WMEs, (*Adelaide is in Oceania*) (*Sydney is in Oceania*) and (*Perth is in Oceania*) are created. Notice that the WME (*Melbourne VIC Australia city*) does not trigger the rule with (*Australia in Oceania*) even though it has attributes matching the condition. This is because the number of attributes in the element is not the same as the one in the rule. However, if this example rule changes from multiple triggering, with "−− >", to single triggering, with "∼∼>", it will only be triggered by one set of matching condition WMEs among all existing WMEs.

3.6. Street simulator

At this early stage of development on this new cognitive architecture, a Java Street Engine simulator has been developed and is used for agent development and processing methodology verification. This simulator has a GUI to enable loading agents and displaying elements in the WM after step-wise execution of rules.

4. System implementation

One main objective of this work is to demonstrate the feasibility and capacity of the proposed approach through the implementation of a case study. Therefore, we chose to implement a Street agent-based system on a two-wheel drive rover platform, and test it in an artificial environment with coloured balls as landmarks. The system is designed to reproduce the honey bee's homing behaviour, and to imitate the way a honey bee processes. In this simulation, electronic sensors are used to replicate a honey bee's detecting vision and magnetic field.

4.1. System structure

The system structure and relationships between the modules of the hardware platform are shown in Fig. 3.

The system receives numerical inputs representing information about detected objects from a colour-based object detection sensor and navigational angles from a magnetic sensor. These two inputs simulate the senses of vision and magnetoreception and provide all of the information that the rover has access to about its environment. The object processing module and the magnetic input processing module are run on an Arduino microcontroller, and they process the numerical inputs from the sensors and convert them into fuzzy symbolic representations. These two modules represent the honey bee's sensory organs and their processing lobes, which sense external stimuli from the real environment and transmit them into its nervous system in the form of chemical elements and electronic impulses [15]. The symbolic elements describing the environment are pushed into two different modules, which are both running on the Street Engine simulator but loaded with different agents. One of them memorises the information gathered in the foraging process, and the other determines the movement actions to be made. This multi-agent structure allows agents to develop their own WM without interaction or interference, which enables simpler implementation. After motion decisions have been made, the system outputs are sent to a motion controller, which is running on another Arduino microcontroller with motor drivers[1]. The motion controller receives mobility commands from the decision-making module (Command Agent) and converts them to numerical motion instructions. It drives two stepper motors with drivers to perform the required movement. The motion controller only has basic open loop controllers to drive the stepper motors. The rover is assembled on a 3D printed chassis, shown together with a diagram of the hardware architecture in Fig. 4.

4.2. Sensing inputs

In this work, the system is limited through the number and types of inputs to be just enough to enable the agent to make basic decisions. This work does not focus on the optimisation of sensors or low-level sensor processing, so the system is implemented with devices with pre-processing functions. We use the Pixy object detection camera[2], which can detect signature objects with predefined colours in a frame and output the coordinates and lengths of each detected object (in pixels) to the processing module together with their colour code. The LM303 compass module provides the angle between the rover's current heading direction and magnetic North, in degrees. All raw data are inputted to the Arduino (Micro) in real-time, which reformats the information as follows, for further processing. For each signature object, the colour code, such as "*RED*" and "*YELLOW*", is followed by the XY coordinates and height and width of the object, in pixels. The X coordinate is the distance in pixels to the left edge of the frame, and the Y coordinate is the distance to the top of the frame. Following the data for all detected objects, a navigational angle provided by the compass module is added. All inputs are updated in real-time, reformatted into

[1] An A4899 stepper motor driver is connected to drive each stepper motor.

[2] The Pixy object detection camera is also known as a PixyCam, which uses the contrast of colours to extract particular signature colours from the background.

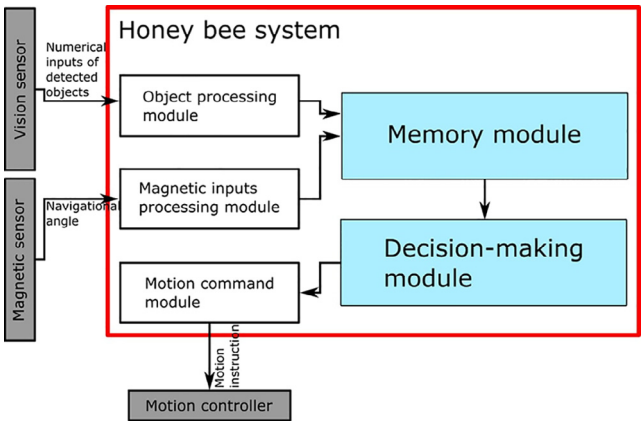

Fig. 3. System structure.

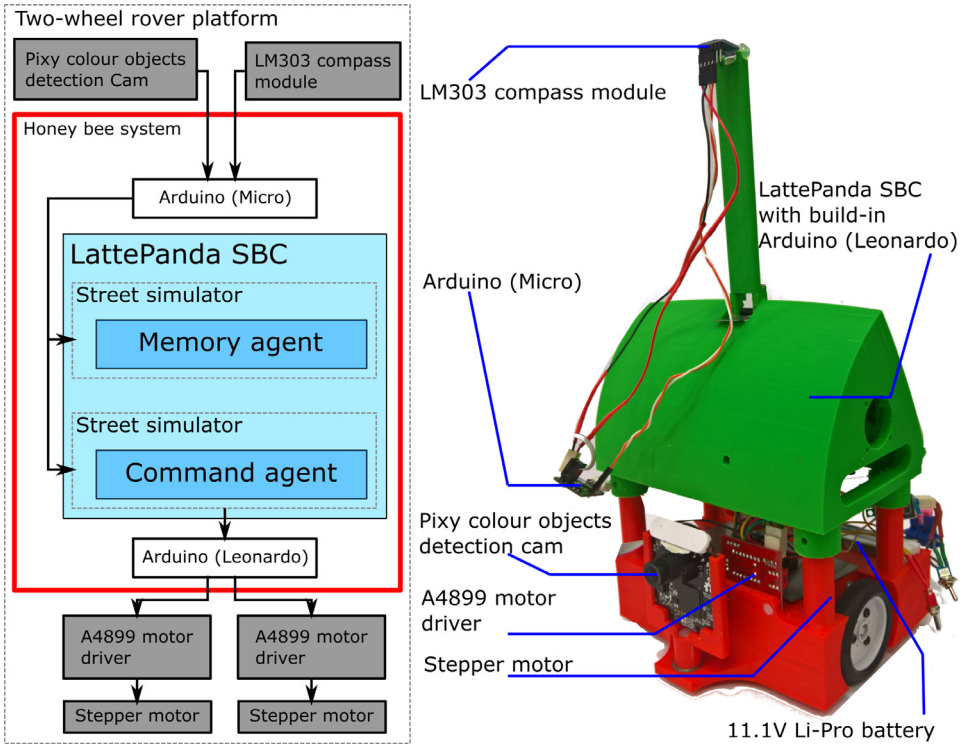

Fig. 4. Two-wheel rover platform.

this numerical data packet and pushed into the system as a variable length ASCII encoded string. The number of objects detected in the same frame is not limited. The numerical inputs are formatted as:

($signature_1$ X_1 Y_1 $Height_1$ $Width_1$ $signature_2$ X_2 Y_2 $Height_2$ $Width_2$ $angle$).

An example of two objects detected in the same frame is shown in Fig. 5.

4.3. Fuzzy logic

Our understanding of cognitive biological creatures is that they do not process information in the precise numerical form [20] that is available from most digital sensors. To the best of our knowledge, biological creatures only recognise imprecise information so we restrict ourselves to use processing representations of information in an imprecise form. For this reason, for cognitive processing and decision making, we have decided to represent information presented to and managed by the agents in fuzzy form. Fuzzy logic was first introduced by Zadeh [49], and has been applied to a wide range of research prob-

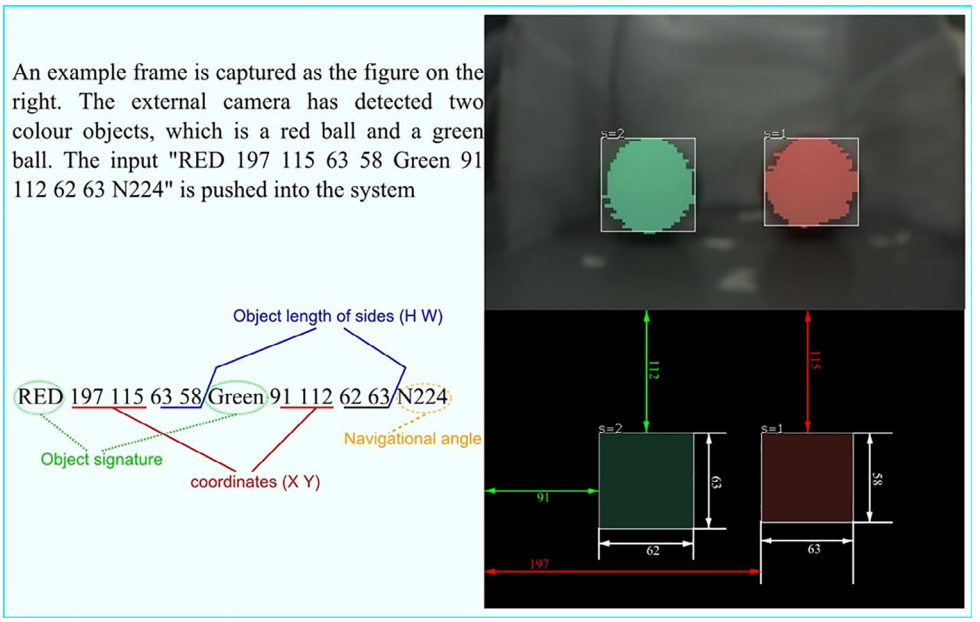

Fig. 5. Sensing inputs example.

lems [27,3,46,26,25,40,4]. Rather than following the formalism of fuzzy logic we adopt a fuzzy representation of the raw information obtained from the sensors and apply our rule-based processing to it.

Thus, all inputs are converted into the fuzzy symbolic elements by the processing modules running on the Arduino and are then processed by the agents. After information is processed by the agents and high-level decisions are made about the next direction of travel, the motion decision elements are decoded into digital signals to instruct the platform to execute a movement in that direction of travel. A fuzzy processing architecture is used with a cognitive agent at its core, as illustrated in Fig. 6. It is refined from the Fuzzy Inference System (FIS) [19,16,17,34]. In FIS, a number of fuzzy IF-THEN rules, [6,44] together with a predefined database, comprise the knowledge base for making decisions. In our structure, the knowledge base is replaced by the structure of decision-making rules, and reasoning rules trigger a series of reactions and create elements that can lead to motion decisions.

Compared with the honey bee's system structure (Fig. 3)) the processing modules are used for fuzzification and convert numerical inputs into a fuzzy symbolic representation. The motion module converts the fuzzy motion instructions into digital control signals for the motors based on the decisions made, which acts as a defuzzification process. The Command Agent makes the decisions to drive the platform, and these decisions are based on knowledge recorded in the Memory Agent.

4.4. Fuzzy symbolic representation

4.4.1. Visualise information

Before any information is processed through the Street agent, the system is required to convert numerical data into fuzzy symbolic form. That is, the characteristics of detected objects must be converted into words. [36] Therefore, the input processing module translates the digital signal from sensors into numerical values and then converts them into fuzzy symbolic elements and introduces them to our system as Working Memory Elements.

The image sensor provides a description of each object detected in its visual field in the form of its colour and X and Y coordinates in the frame and Width and Height of the detected object. The Input Processing Module calculates a numerical estimate of the distance (D_n) from the bottom of the view to the centre of the nth objects, in pixels, with,

$$D_n = Frame_{height} - Y_n - \frac{Height_n}{2}$$

and the side distance (S_n),

$$S_n = X_n - \frac{Width_n}{2}$$

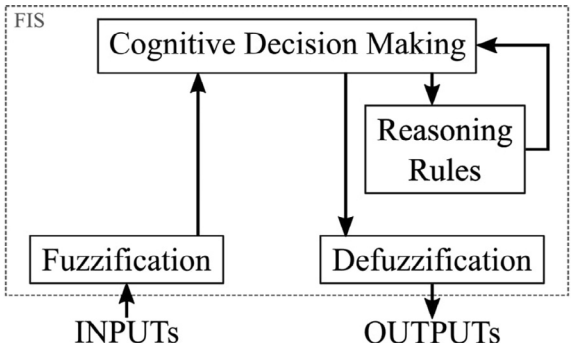

Fig. 6. Fuzzy decision-making architecture.

which provides a distance from the left side of the frame to the centre of the nth object. In these two equations, X_n and Y_n are the coordinates of the nth object in the frame, and Frame_ height is the height of the current frame. Then, these values are converted into symbolic fuzzy descriptors of the position of the object [3]

$$D_{fuzzy} = \begin{cases} is\_far, D \geqslant Frame_{height} \times 30\% \\ is\_close, Frame_{height} \times 30\% > D > Frame_{height} \times 20\% \\ is\_nextTo, D < Frame_{height} \times 10\% \end{cases}$$

$$S_{fuzzy} = \begin{cases} on\_the\_Left, D \leqslant Frame_{width} \times 20\% \\ in\_the\_Middle, Frame_{width} \times 20\% < D < Frame_{width} \times 80\% \\ on\_the\_Right, D \geqslant Frame_{width} \times 80\%, \end{cases}$$

where D_{fuzzy} and S_{fuzzy} are the symbolic attributes representing the fuzzy distance to the objects and the fuzzy distance to the side of the frame. The range of each fuzzy set is determined by the width ($Frame_{width}$) and height ($Frame_{height}$) of the frame, which are constants defined at the set-up.

To represent a frame of an image, we use a set of elements describing objects in the visual field. These elements include the fuzzy distances to the objects detected and their fuzzy lateral positions in the frame, with unique object IDs, colour signature elements and a count for the total number of detected objects in the same frame. For example, the set of WMEs describing the objects detected in the image in Fig. 7 would be:

$< s >$	*distance*	*_ID*1	*is_far*
$< s >$	*distance*	*_ID*2	*is_close*
$< s >$	*side*	*_ID*1	*in_the_Middle*
$< s >$	*side*	*_ID*2	*on_the_Right*
$< s >$	*colour*	*_ID*1	*GREEN*
$< s >$	*colour*	*_ID*2	*YELLOW*
$< s >$	*blocksCount*	2	

In this example, each row is a WME, in which $< s >$ represents a common reference ID for linking information about the same frame. The *_IDx* attribute provides a unique reference code for each object, and the colour signature of objects are also defined in WMEs. For each object detected in the frame there will be one distance WME, one side WME and one colour WME, with a final WME giving the total number of objects detected in the frame.

4.4.2. Navigation angle

To reproduce the honey bees' magnetoreception capability [42], a magnetic sensor is attached, which provides an angle (in degrees) of the clockwise bearing of the rover's current direction of travel from magnetic North, such as N224 in Fig. 5. This angle is converted into fuzzy symbolic form in the magnetic input processing module before being pushed into the WM of both agents.

The conventional description of geographical directions is North, East, South and West. However, this interpretation has been found to be too coarse to indicate directions in this system. We decided to use eight directions, because it was the smallest number that gave sufficient angular resolution. A 2-dimensional 360-degree compass is divided into eight major directions, N, A, E, B, S, C, W and D, as illustrated in Fig. 8.

[3] The percentages are determined based on experiments.

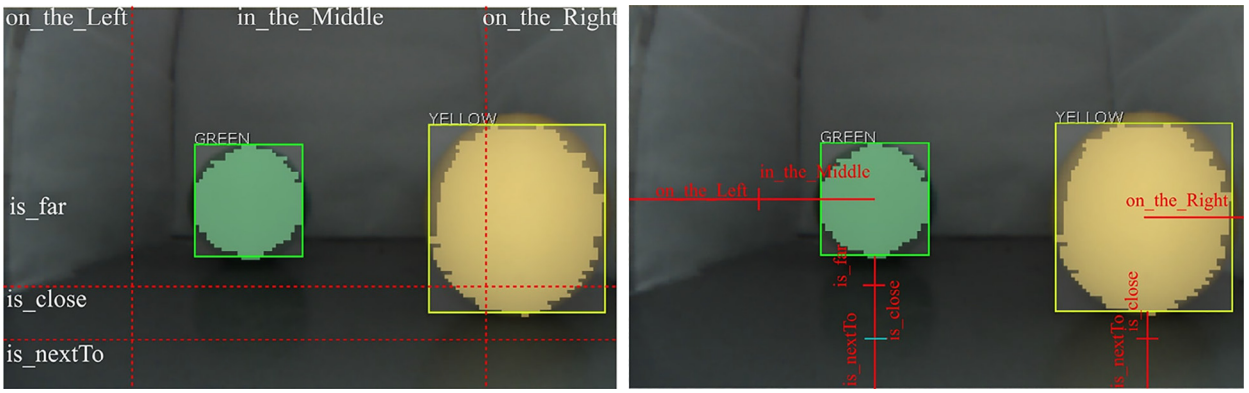

Fig. 7. The fuzzy representation distances and sides.

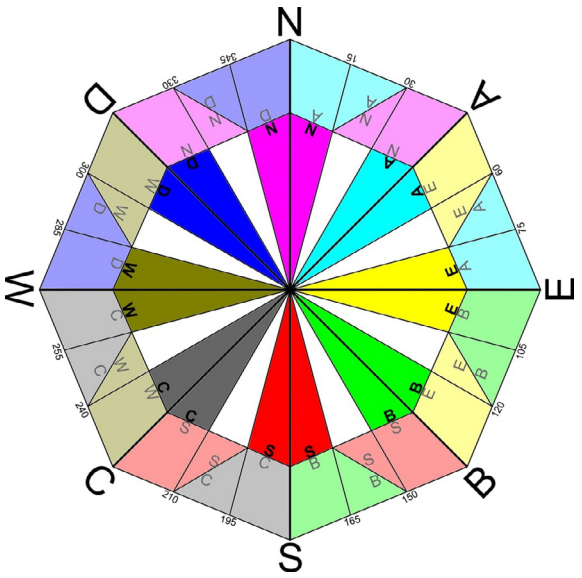

Fig. 8. The fuzzy compass.

The system is designed to avoid using the conventional compass point representations of NE, NW, SW and SE for brevity. For the purposes of fuzzification, for each of these compass points, there is a region either side in which the bearing is considered to be strongly aligned to the compass point (on the inner circle with darker colours) and two regions that are weakly aligned (on the outer circle with dimmer colours). If the input angle is significantly close to one of the eight compass points, the intensity element of that direction will be *STRONG*. Conversely, if it is further away, it will be designated *WEAK*. In some cases, if the input angle happens to be approximately equidistant from two compass points, the magnetic intensity elements will both be *WEAK*. Our approach is to simulate the magnetoreception of honey bees which is a sense of feeling towards a magnetic direction. Therefore, each inputted angle is converted to two WMEs describing the degree of alignment to the directions on either side. The format is ($< s >$ compass $<$ *direction* $>$ $<$ *intensity* $>$). For example, an input angle of 10° magnetic WME in the agent will be,

$< s >$ *compass N STRONG*
$< s >$ *compass A WEAK*.

This representation avoids motion being triggered with non-significant differences, such as the difference between 1° and 355°. A 1° angle is converted to:

$< s >$ *compass N STRONG*
$< s >$ *compass A WEAK*,

and a 355° angle is converted to:

$< s >$ *compass N STRONG*

$< s >$ *compass D WEAK*.

So, both are classified as being *N STRONG*.

For the implementation of the Command Agent, when making decisions about motion, only when both direction elements are different will a motion suggestion be triggered. Therefore, 1° and 355° angles are not considered to be significantly different angles.

4.4.3. Master frame

We call a set of WMEs describing all objects detected and the navigational direction of the platform at a reference position a Master Frame (MF). All WMEs in the MF are characterised with a common identifier, as described above. As a special case, an input MF describes the visually and magnetically perceived environment at the present instant in time. It is the integration of the elements containing object IDs with a fuzzy indication of their vertical and lateral positions in the visual field, colour signature, the total number count of detected objects and the geomagnetic field direction at a particular moment. As inputs, all WMEs in the MF are generated with the same reference ID symbol, *_INPUT*. An example of an input MF containing two objects in the middle of the frame, where a green ball is in the far distance, and a red ball is close, and where the magnetic angle is 79°, is:

_INPUT	*distance*	*_ID1is_far*
_INPUT	*distance*	*_ID2is_close*
_INPUT	*side*	*_ID1in_the_Middle*
_INPUT	*side*	*_ID2on_the_Right*
_INPUT	*colour*	*_ID1GREEN*
_INPUT	*colour*	*_ID2YELLOW*
_INPUT	*blocksCount*	2
_INPUT	*compass*	*AWEAK*
_INPUT	*compass*	*ESTRONG*

Therefore, in this work, symbolic elements in MFs store important characteristics of past situations experienced. In an MF, all the necessary information is put together to recreate a memory of a particular reference position. Therefore, each recorded MF is an episodic memory piece, and, by creating connections between them, these memories are linked together.

4.5. Cognitive processing architecture

The cognitive processing for reasoning and making decisions is performed by two cooperating Street agents: the Memory Agent and the Command Agent (see Fig. 3)). The system starts in a foraging mode, in which the Command Agent instructs the platform to explore the surrounding environment by generating a series of pseudo-random movements at controller level. As it proceeds, the Memory Agent determines significant changes in the surrounding environment to recognise new reference positions and records the sequence of these reference positions and their characteristics using MFs. While homing, the Command Agent makes decisions for motions based on comparisons between current situations and memorised situations and drives the platform to return using the same path.

4.5.1. Memory agent

The Memory Agent records the sequence of *_INPUT* MFs received and outputs each WME of a target MF to the Command Agent, when requested to do so. When the platform is in foraging mode, the Memory Agent determines any difference between the WMEs of the current *_INPUT* MF and those of the last recorded MF. If the new MF differs, the system is considered to be experiencing a new reference position. The Memory Agent then duplicates the *_INPUT* MF and stores it into its WM as a set of episodic memories [39] with a new unique reference ID $< s >$. It also produces a list of WMEs to link these reference IDs to track the order of MFs that appear by creating an element of the form (*_IDlast followedBy _IDnew*) in its WM.

4.5.2. Command agent

The Command Agent is the key module controlling the system. The primary function of this agent is determined by whether it is in free discovery foraging mode or homing mode. This is determined by whether the target object has been detected.

The Command Agent initialises into foraging mode. In foraging mode, the agent generates random movements while receiving inputs from the sensory modules and checking if any element of the current *_INPUT* MF matches a pre-defined target WME, which, for the purposes of this experiment, contains a set unique colour attribute. This simulates the discovery of a food source and causes the Command Agent to switch to homing mode.

Once the agent enters homing mode, it finds the last recorded MF for a reference position and adopts it as the target MF to find. It then compares the target MF with the elements of the current *_INPUT* MF and, as a result of any difference in attri-

butes, the rules suggest motion requirements and the Command Agent determines a high-level decision for motion based on these requirements. It thus begins its first step in retracing its path back to the initial location. The general processing structure is illustrated in Fig. 9.

More specifically, the Command Agent takes the function of the Reasoning Rules shown in Fig. 6. While comparing the target MF and the current *_INPUT* MFs, the rules in the Command Agent trigger the creation of WMEs suggesting movement requirements. This procedure provides WMEs describing a resolving movement for every difference detected by comparing the target and *_INPUT* MF elements representing the same property. Therefore, each MF comparison can result in several, possibly inconsistent, motion suggestions. These are resolved at a higher level, as described below.

The Command Agent's rules are written such that the agent compares each WME in the current MF and target MF describing longitude distances, lateral positions and the heading direction, and creates WMEs representing the detected differences. The general rule format is:

$$
\begin{aligned}
&st\{compareDifferences\\
&\quad (homing)\\
&\quad (\_IDtarget < type >< ID1 >< condition1 >)\\
&\quad (\_IDcurrent < type >< ID2 >< condition2 >)\\
&\quad (\_IDtargetcolour < ID1 >< colour >)\\
&\quad (\_IDcurrentcolour < ID2 >< colour >)\\
&\quad -(\_IDtargetcolour < ID2 >< colour >)\\
&-->
&\quad (difference < type >< ID1 >< condition1 >< condition2 >)\\
&\quad -(\_IDcurrent < type >< ID1 >< condition2 >)\\
&\}
\end{aligned}
$$

in which the WME starting with *difference* indicates a detected difference, and $< type >$ illustrates the type of difference (longitude distance, lateral position or the heading direction).

Then, these differences will trigger a motion suggestion. The longitude comparison compares the difference in distances for each detected object in the MF and generates a preferred longitudinal motion. It may suggest a reverse or forward movement depending on the difference, such as:

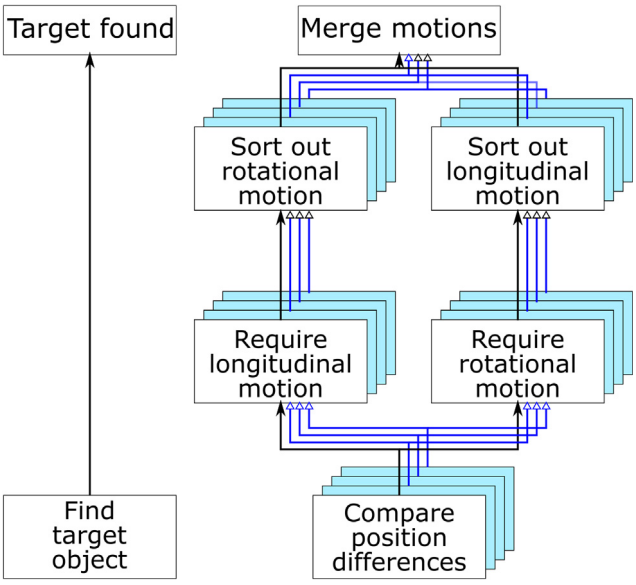

Fig. 9. Decision-making architecture of the Command Agent.

$st\{requireLongitudinal$
$(homing)$
$(difference\ distance < ID > is\_closeis\_far)$
$-->$
$(require\ longitudinal < ID > short\ forward)$
$-(difference\ distance < ID > is\_close\ is\_far)$
$\}$

For the lateral position, the agent's rules trigger lateral motion requests (left or right) if differences appear. The agent also compares the WMEs, which represent the current bearing. The agent creates an element to require a rotation if the heading direction elements are significantly different. As noted previously, in this comparison, a movement suggestion is only triggered if both elements representing the heading direction are different.

As an example, if the current MF contains:

$\_IDcurrent \quad compass \quad N \quad STRONG$
$\_IDcurrent \quad compass \quad A \quad WEAK,$

and the target MF contains,

$\_IDtarget \quad compass \quad N \quad WEAK$
$\_IDtarget \quad compass \quad A \quad WEAK,$

this does not trigger any action since this is not considered to be a significant difference. Therefore, a rotational motion requirement is only generated when two WMEs indicating a compass point difference exist in the WM. The rule format is in the form:

$st\{requireRotational$
$(homing)$
$(difference\ compass < direction1 >< intensityA >< direction2 >< intensityA >)$
$(difference\ compass < direction3 >< intensityB >< direction4 >< intensityB >)$
$-->$
$(require\ rotation < ID >< amplitude >< motion >)$
$-(difference\ compass < direction1 >< intensityA >< direction2 >< intensityA >)$
$-(difference\ compass < direction3 >< intensityB >< direction4 >< intensityB >)$
$\}.$

The Command Agent now has a reason for selecting a motion requirement to resolve the position difference between the target and current MFs. However, multiple motion requirements may have been generated as a result of multiple MF differences and the agent cannot necessarily act on all of these motions. Some of them may be conflicting. Therefore, for each property, only one WME suggesting a motion in a particular direction will be generated.

In principle, we select the WME containing the larger scale of motion, if all elements for the same property require a motion in the same direction:

$st\{sortMotion1$
$(homing)$
$(require < type >< ID1 > long < direction >)$
$(require < type >< ID2 > short < direction >)$
$-->$
$-(require < type >< ID1 > short < direction >)$
$\}.$

However, if there are motion requirement WMEs of a particular type with conflicting directions, the system will only pick the one with the larger scale of motion. An example rule for selecting from conflicting directions is:

$st\{sortMotion2$
$(homing)$
$(require < type >< ID1 > short < direction1 >)$
$(require < type >< ID2 > long < direction2 >)$
$-(require < type >< ID1 > short < direction2 >)$
$-->$
$-(require < type >< ID2 > long < direction2 >)$
$\}.$

If opposite directions appear in WMEs with the same scale, one of them will be selected by the rule for deletion, effectively picking one of the two directions at random:

$st\{sortMotion3$
$(homing)$
$(require < type >< ID1 >< scale >< direction1 >)$
$(require < type >< ID2 >< scale >< direction2 >)$
$-->$
$-(require < type >< ID2 >< scale >< direction2 >)$
$\}.$

After a single motion for each property has been selected, these selected motion requirement WMEs provide the next level of reasoning, and the agent will then aggregate them into a single high-level movement:

$st\{mergeMotions$
$(require\ longitudinal < ID1 >< scale1 >< direction1 >)$
$(require\ transverse < ID2 >< scale2 >< direction2 >)$
$(require\ rotation < scale3 >< direction3 >)$
$-->$
$(\_OUTPUTmotion\ decision < scale1 >< direction1 >< scale2 >< direction2 >$
$< scale3 >< direction3 >)$
$-(require\ longitudinal < ID1 >< scale1 >< direction1 >)$
$-(require\ transverse < ID2 >< scale2 >< direction2 >)$
$-(require\ rotation < scale3 >< direction3 >)$
$\}.$

5. System behaviour and analysis

5.1. Foraging mode

The system starts in foraging mode, in which the Command Agent instructs the platform to explore the surrounding environment by generating a series of pseudo-random movements. As it does so, the Memory Agent determines significant changes in the surrounding environment to recognise new reference positions and records the sequence of these reference positions and their characteristics, using MFs.

A *_INPUT* MF is pushed into the WMs of both agents in real-time, containing a set of *_INPUT* WMEs representing the signature information of the current position. If any *_INPUT* WME is different from the last stored MF, the system is considered to be in a new reference position. The Memory Agent will then record this *_INPUT* MF for this new reference position on the foraging trail with a unique *_IDn*, and a WME to link the new MF to the preceding MF is generated, thereby creating a time-ordered list of MFs. An example scenario is presented in Fig. 10.

In this example, there are five reference locations, which are captured for five different MFs and are all stored in the WMs of the agents with unique reference IDs. Initially, the system detects two objects, a *GREEN* and a *RED*, both in an *is_far* and *in_the_middle* view distance, and the platform is aligned towards the top of the page. Then, the platform moves forward to get a closer to those two objects. A new MF, with the individual distances changed to *is_close*, is captured. One of the compass elements changes to *D WEAK* from *A WEAK*, yet the platform does not rotate. This change is caused by the drifting of the input from the magnetic sensor, and this minor difference will not trigger any actions in the comparisons later during the homing stage.

Then, the platform continues moving forward, and it moves even closer to the objects. The fuzzy symbolic attributes of the distances become *is_nextTo* for both objects. The RED object is *on_the_Right* since the viewing point is closer and farther

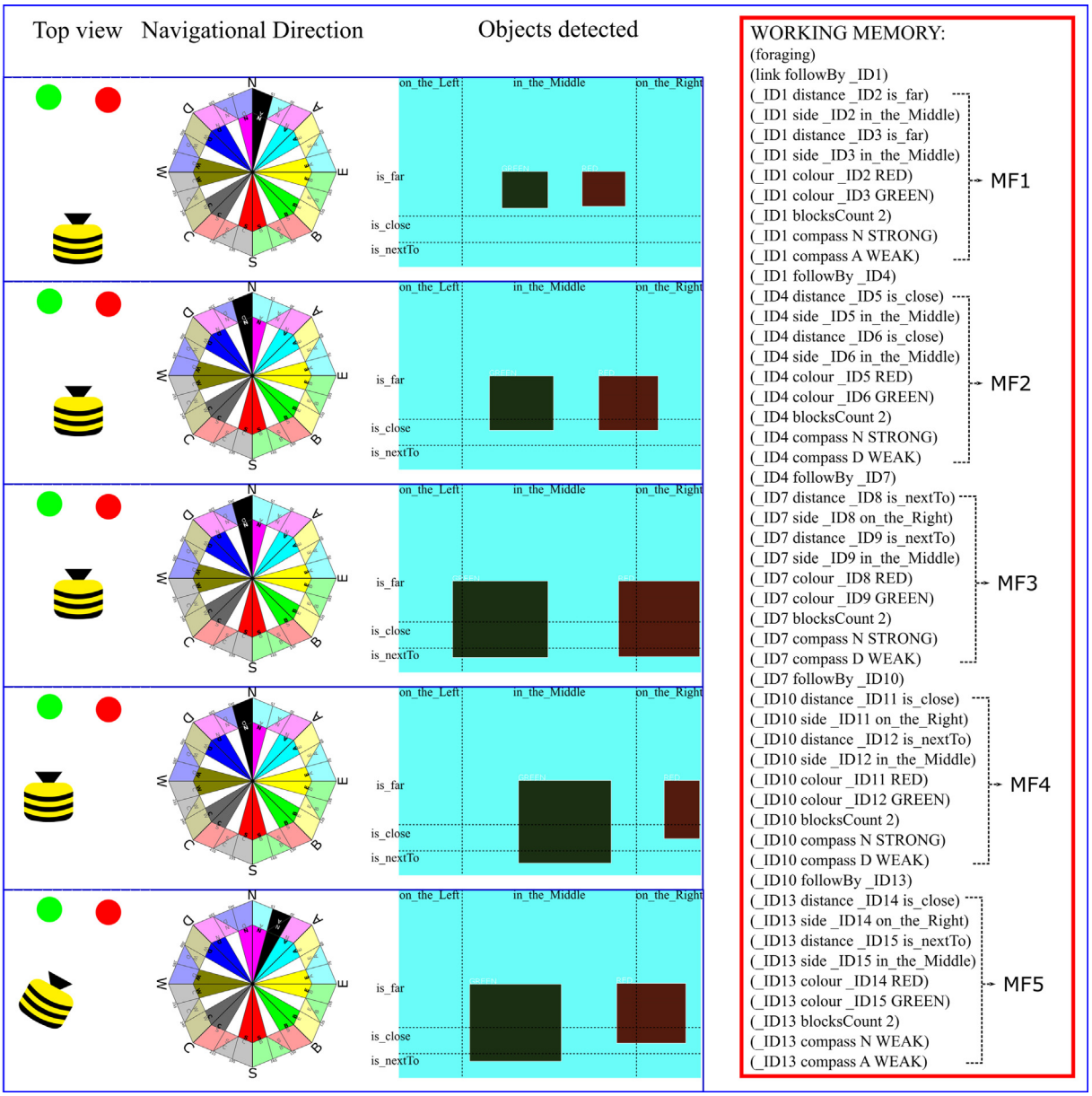

Fig. 10. Example of a foraging scenario.

from the viewing point of the *GREEN* object. Afterwards, the platform shifts to the left, in which the *RED* object looks farther away, which is *is_close*, and the fuzzy attributes about the *GREEN* object remain unchanged. Lastly, the platform rotates about 30 degrees clockwise, which changes the compass WMEs to *N WEAK* and *A WEAK*. All WMEs recorded by the Memory Agent during foraging stage of this example are detailed in Fig. 10.

5.2. Homing stage

5.2.1. Overview of behaviour during homing

Our main objective is to have the system to behave as a honey bee would. During the homing stage, the platform is expected to retreat to the approximate starting point by retracing the path it used to approach the target (food location) with the designed cognitive reasoning processes. To do this, the system returns to each of the reference positions indicated by a stored MF in the reverse order to that in which it visited them during the foraging phase, finishing at the first reference position.

Fig. 11 illustrates an example of our testing scenarios, in which the system has some pre-recorded MFs in the WM and uses them to make motion decisions to retreat to the initial position. After the mode changes from Foraging to Homing, the system begins to retrace its path by attempting to approach the most recent reference position. It adopts the MF of the last position as the Target Frame and compares it with all WMEs of the current *_INPUT* MF, as previously described. Based on this comparison, the Command Agent will decide to make a move that will position it closer to the position characterised by the target MF. When all WMEs of the target MF are matched with the current MF, the target reference position has reached, and the Command Agent will then request the next target MF from the WM of the Memory Agent. In general, the comparison will repeat until the initial position has been reached.

5.2.2. Resolving position differences

As in the example of Fig. 11, the last comparison is between a target MF,

_IDtarget	*distance*	*_ID1is_close*
_IDtarget	*side*	*_ID1on_the_Right*
_IDtarget	*distance*	*_ID2is_nextTo*
_IDtarget	*side*	*_ID2in_the_Middle*
_IDtarget	*colour*	*_ID1RED*
_IDtarget	*colour*	*_ID2GREEN*
_IDtarget	*blocksCount*	2
_IDtarget	*compass*	*NWEAK*
_IDtarget	*compass*	*AWEAK*.

and the current MF,

_IDcurrent	*distance*	*_ID3is_far*
_IDcurrent	*side*	*_ID3in_the_Middle*
_IDcurrent	*distance*	*_ID4is_far*
_IDcurrent	*side*	*_ID4in_the_Middle*
_IDcurrent	*colour*	*_ID3RED*
_IDcurrent	*colour*	*_ID4GREEN*
_IDcurrent	*blocksCount*	2
_IDcurrent	*compass*	*NSTRONG*
_IDcurrent	*compass*	*DWEAK*.

The rules determining the position differences are triggered, which create WMEs,

difference	*distance*	*_ID1is_closeis_far*
difference	*side*	*_ID1on_the_Rightin_the_Middle*
difference	*distance*	*_ID2is_nextTois_far*
difference	*compass*	*NWEAKNSTRONG*
difference	*compass*	*AWEAKDSTRONG*.

Based on these difference elements, motion requirement WMEs are created, which are:

require	*longitudinal*	*_ID1shortforward*
require	*transverse*	*_ID1shortleft*
require	*longitudinal*	*_ID1longforward*
require	*rotation*	*shortclockwise*

to suggest movements to resolve position differences. Then, the rule, *sortMotion1*, will be triggered by (*require longitudinal _ID*1 *short forward*) and (*require longitudinal _ID*1 *long forward*) to retain only the longer motion for the longitudinal movement. Then, the agent will make a decision by combining these movement requirements and output (*_OUTPUT motion decision long forward short left short clockwise*) to command the platform to move towards the target position. In Fig. 11, these processes have been repeated three times until the current MF matches the target MF.

5.2.3. Target master frame matches

If all WMEs in the current MF match the target MF, the system will realise that the target position has been reached. Then, the system replaces the target MF with the next recorded MF and repeats the comparisons and commanding procedures. The homing stage terminates when the last target, the first recorded MF created during the foraging phase, is matched.

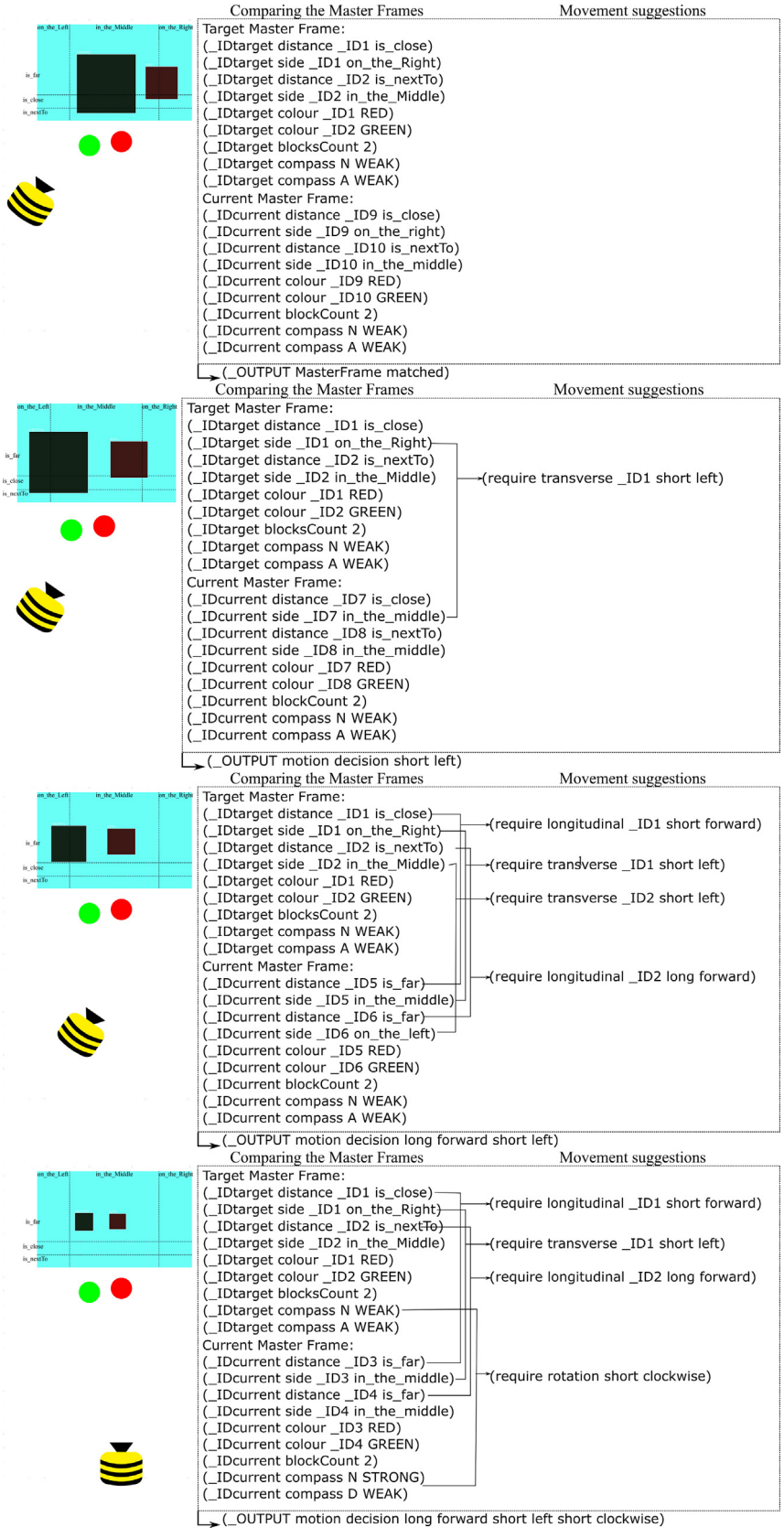

Fig. 11. Target Master Frame matching example.

Fig. 12 provides an example scenario of the agent homing to the starting point with a series of movements that map the current *_INPUT* MF to the target MF. In this example, the platform is driven along the path (black dash lines with direction arrows) and, eventually, reaches the target of the *YELLOW* object. Along this trip, about 10 MFs are recorded. When homing, the platform is driven based on the motion decisions and matches each recorded MF. The returning path does not entirely match with the foraging path, yet it still returns to close to the starting position.

5.3. Evaluation

To evaluate the effectiveness of the homing agent, we tested the system in an artificial environment, with coloured balls used as landmarks. The objective is to verify the ability of the system to retrace a path to its home location using only the visual landmarks provided by the coloured balls and the magnetic bearing. In each test, such as that illustrated in Fig. 12, the rover was moved manually through foraging paths of various lengths, exposing it to the landmarks in different sequences. These manual motions allowed us to track the foraging path, then compare it with the homing path. Tests were conducted with four different configurations of the balls and with foraging paths of varying lengths. As the system navigated the homing paths, the changes in WM of both agents and decision-making processes were monitored, and the accuracy of these input elements was checked by printing them out on an on-board screen.

Comparing with conventional path planning algorithms, in which motions are modelled directly based environment changes, our approach is entirely based on cognitive logic to make decisions with completely explainable reasoning processes to ensure the transparency of all decisions been made. The rules of the agents were debugged and augmented until the system was able to, in every experimental scenario, to reach each reference position in the homing path and eventually return to the starting position. With the fuzzy symbolic representation, the information elements do not provide sufficient details for the agent to allow it to position the platform at exactly the same position as it occupied in the foraging phase. Moreover, due to some inaccuracies in the motion control function, the platform can deviate from the desired path. Consequently, the system may not always make the optimal decisions about which direction to move, nor that it positions itself in exactly the same location with respect to the landmarks as it was in the foraging phase. However, it does move into approximately the original positions for each landmark in turn and eventually finish at the home location. This outcome means that the platform can make corrections at each reference position, and return on a paths similar to the foraging path.

6. Conclusion

This paper has detailed the conceptual design, development and testing of a system which is based on rule-based processing using fuzzy representation of the environment to reproduce honey bee homing behaviours. The system is able to approximately reproduce the homing behaviour of honey bees in an artificial environment. The same method and system can be

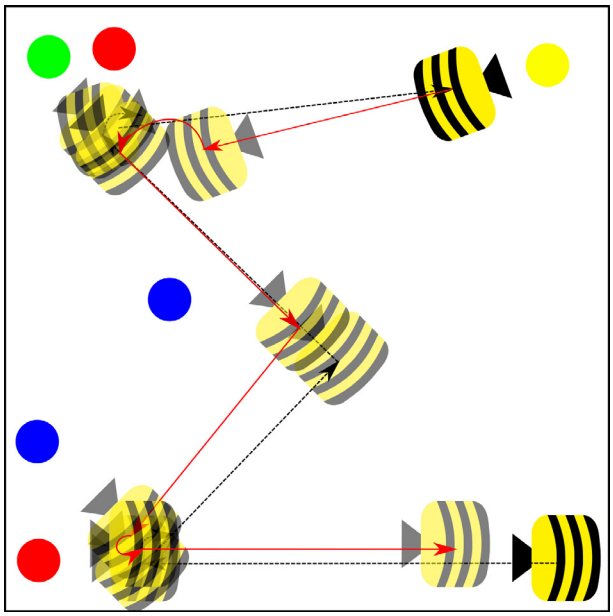

Fig. 12. Example homing scenario.

scaled up and used for duplicating human cognitive behaviours. This work has demonstrated the feasibility of using a rule-based approach for the implementation of limited Artificial General Intelligence.

Nonetheless, this work is limited by sensors, the fuzzification methods and decision-making structure. This system makes motion decisions based on matching colour objects, in which the colour signature is the only characteristic used to find matching objects. In our experiments, some repeated colours or detection of multiple objects with the same colour can lead to incorrect decisions being made because the wrong rules are triggered. To improve the effectiveness of the agents, more unique characteristics must be identified from the detected environmental information. Our future work will be the further refinement of both the decision-making structure and the fuzzy representation.

CRediT authorship contribution statement

Xin Yuan: Investigation, Methodology, Software, Writing – original draft. **Michael John Liebelt:** Supervision, Writing – review & editing. **Peng Shi:** Supervision, Writing – review & editing. **Braden J. Phillips:** Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

aaa

References

[1] E. Alpaydin, Introduction to machine learning, MIT Press, 2020.
[2] N. Boeddeker, L. Dittmar, W. Stürzl, M. Egelhaaf, The fine structure of honeybee head and body yaw movements in a homing task, Proc. R. Soc. B 277 (1689) (2010) 1899–1906.
[3] O. Castillo, L. Amador-Angulo, A generalized type-2 fuzzy logic approach for dynamic parameter adaptation in bee colony optimization applied to fuzzy controller design, Inf. Sci. 460–461 (2018) 476–496.
[4] O. Castillo, L. Cervantes, J. Soria, M. Sanchez, J.R. Castro, A generalized type-2 fuzzy granular approach with applications to aerospace, Inf. Sci. 354 (2016) 165–177.
[5] P.K. Chan, F. Luo, Z. Chen, Y. Shu, D.S. Yeung, Transfer learning based countermeasure against label flipping poisoning attack, Inf. Sci. 548 (2021) 450–460.
[6] C.H. Chen, S.Y. Yang, Neural fuzzy inference systems with knowledge-based cultural differential evolution for nonlinear system control, Inf. Sci. 270 (2014) 154–171.
[7] L. Chittka, J. Tautz, The spectral input to honeybee visual odometry, J. Exp. Biol. 206 (14) (2003) 2393–2397.
[8] N. Courbin, T. Chinho, L. Pichegru, A. Verma-Grémillet, C. Péron, P.G. Ryan, D. Grémillet, The dance of the cape gannet may contain social information on foraging behaviour, Anim. Behav. 166 (2020) 95–108.
[9] M.J. Couvillon, F.C.R. Pearce, C. Accleton, K.A. Fensome, S.K. Quah, E.L. Taylor, F.L. Ratnieks, Honey bee foraging distance depends on month and forage type, Apidologie 46 (1) (2015) 61–70.
[10] C.L. Forgy, OPS5 user's manual (Technical report), Carnegie-Mellon Univ Pittsburgh PA Dept of Computer Science, 1981.
[11] J. Frost, M.W. Numan, M. Liebelt, B.J. Phillips, A new computer for cognitive computing, in: 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC), Beijing, China, 2015, pp. 33–38.
[12] S.N. Fry, R. Wehner, Honey bees store landmarks in an egocentric frame of reference, J. Comp. Physiol. A 187 (12) (2002) 1009–1016.
[13] W. Gao, M.D. McDonnell, Analysis of gradient degradation and feature map quality in deep all-convolutional neural networks compared to deep residual networks, in: International Conference on Neural Information Processing, Springer Verlag, Guangzhou; China, 2017, pp. 612–621.
[14] X. Geng, Y. Liang, L. Jiao, Earc: Evidential association rule-based classification, Inf. Sci. 547 (2021) 202–222.
[15] M. Giurfa, Behavioral and neural analysis of associative learning in the honeybee: a taste from the magic well, J. Comp. Physiol. A 193 (8) (2007) 801–824.
[16] X. Gu, P. Angelov, H.J. Rong, Local optimality of self-organising neuro-fuzzy inference systems, Inf. Sci. 503 (2019) 351–380.
[17] A. Hernandez-Aguila, M. García-Valdez, J.J. Merelo-Guervós, M. Castañón-Puga, O. Castillo, Using fuzzy inference systems for the creation of forex market predictive models, IEEE Access 9 (2021) 69391–69404.
[18] A. Horridge, Some labels that are recognized on landmarks by the honeybee (apis mellifera), J. Insect Physiol. 52 (11–12) (2006) 1254–1271.
[19] J.S.R. Jang, ANFIS: Adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (3) (1993) 665–685.
[20] B. Kenwright, Bio-inspired animated characters: A mechanistic & cognitive view, in: 2016 Future Technologies Conference (FTC). St San Francisco; United States, 2016, pp. 1079–1087..
[21] T. Labhart, E.P. Meyer, Neural mechanisms in insect navigation: polarization compass and odometer, Curr. Opin. Neurobiol. 12 (6) (2002) 707–714.
[22] J.E. Laird, The Soar cognitive architecture, MIT Press, 2012.
[23] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[24] Y. Li, L. Liu, C. Shen, A. Van Den Hengel, Mining mid-level visual patterns with deep cnn activations, Int. J. Comput. Vision 121 (3) (2017) 344–364.
[25] Z. Lian, Y. He, C.K. Zhang, M. Wu, Further robust stability analysis for uncertain takagi-sugeno fuzzy systems with time-varying delay via relaxed integral inequality, Inf. Sci. 409–410 (2017) 139–150.
[26] H. Liu, R.M. Rodríguez, A fuzzy envelope for hesitant fuzzy linguistic term set and its application to multicriteria decision making, Inf. Sci. 258 (2014) 220–238.
[27] Q. Lu, P. Shi, H.K. Lam, Y. Zhao, Interval type-2 fuzzy model predictive control of nonlinear networked control systems, IEEE Trans. Fuzzy Syst. 23 (6) (2015) 2317–2328.
[28] Y. Lv, Y. Duan, W. Kang, Z. Li, F.Y. Wang, Traffic flow prediction with big data: a deep learning approach, IEEE Trans. Intell. Transp. Syst. 16 (2) (2014) 865–873.
[29] R. Menzel, J. Fuchs, L. Nadler, B. Weiss, N. Kumbischinski, D. Adebiyi, S. Hartfil, U. Greggers, Dominance of the odometer over serial landmark learning in honeybee navigation, Naturwissenschaften 97 (8) (2010) 763–767.
[30] R. Menzel, M. Giurfa, Cognitive architecture of a mini-brain: the honeybee, Trends Cogn. Sci. 5 (2) (2001) 62–71.

[31] R. Menzel, U. Greggers, A. Smith, S. Berger, R. Brandt, S. Brunke, G. Bundrock, S. Hülse, T. Plümpe, F. Schaupp, et al, Honey bees navigate according to a map-like spatial memory, Proc. Natl. Acad. Sci. U.S.A. 102 (8) (2005) 3040–3045.
[32] R. Menzel, A. Kirbach, W.D. Haass, B. Fischer, J. Fuchs, M. Koblofsky, K. Lehmann, L. Reiter, H. Meyer, H. Nguyen, et al, A common frame of reference for learned and communicated vectors in honeybee navigation, Curr. Biol. 21 (8) (2011) 645–650.
[33] M.W. Numan, J. Frost, M. Liebelt, B.J. Phillips, A network-based communication platform for a cognitive computer, in: CEUR Workshop Proceedings, Turin, Italy, 2015, pp. 94–103.
[34] E. Ontiveros-Robles, P. Melin, O. Castillo, Comparative analysis of noise robustness of type 2 fuzzy logic controllers, Kybernetika 54 (1) (2018) 175–201.
[35] S. Stach, J. Benard, M. Giurfa, Local-feature assembling in visual pattern recognition and generalization in honeybees, Nature 429 (6993) (2004) 758–761.
[36] K. Štepánová, F.B. Klein, A. Cangelosi, M. Vavrečka, Mapping language to vision in a real-world robotic scenario, IEEE Trans. Cogn. Develop. Syst. 10 (3) (2018) 784–794.
[37] W. Stürzl, N. Böddeker, L. Dittmar, M. Egelhaaf, Mimicking honeybee eyes with a 280 field of view catadioptric imaging system, Bioinspiration & Biomimetics 5 (3) (2010) 036002.
[38] K. Tan, W. Chen, S. Dong, X. Liu, Y. Wang, J.C. Nieh, A neonicotinoid impairs olfactory learning in Asian honey bees (apis cerana) exposed as larvae or as adults, Scientific Rep. 5 (10989) (2015).
[39] H. Tang, R. Yan, K.C. Tan, Cognitive navigation by neuro-inspired localization, mapping, and episodic memory, IEEE Trans. Cogn. Develop. Syst. 10 (3) (2017) 751–761.
[40] A.C. Tolga, I.B. Parlak, O. Castillo, Finite-interval-valued type-2 gaussian fuzzy numbers applied to fuzzy todim in a healthcare problem, Eng. Appl. Artif. Intell. 87 (2020) 103352.
[41] K. Von Frisch, The dance language and orientation of bees, Harvard University Press, 1967.
[42] E. Wajnberg, D. Acosta-Avalos, O.C. Alves, J.F. de Oliveira, R.B. Srygley, D.M. Esquivel, Magnetoreception in eusocial insects: An update, J. R. Soc. Interface 7 (2) (2010) S207–S225.
[43] H. Wang, Y. Wu, G. Min, J. Xu, P. Tang, Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach, Inf. Sci. 498 (2019) 106–116.
[44] J. Wang, J. Liang, C.T. Zhang, Dissipativity analysis and synthesis for positive roesser systems under the switched mechanism and takagi-sugeno fuzzy rules, Inf. Sci. 546 (2021) 234–252.
[45] H. Wolf, Odometry and insect navigation, J. Exp. Biol. 214 (10) (2011) 1629–1641.
[46] Y. Yang, Z.S. Chen, R.M. Rodríguez, W. Pedrycz, K.S. Chin, Novel fusion strategies for continuous interval-valued q-rung orthopair fuzzy information: A case study in quality assessment of smartwatch appearance design, Int. J. Mach. Learn. Cybern. (2021).
[47] X. Yuan, M.J. Liebelt, B.J. Phillips, A cognitive approach for reproducing the homing behaviour of honey bees, in: Proceedings of the 9th International Conference on Agents and Artificial Intelligence, Porto, Portugal, 2017, pp. 543–550.
[48] X. Yuan, M.J. Liebelt, P. Shi, B.J. Phillips, Creating rule-based agents for artificial general intelligence using association rules mining, Int. J. Mach. Learn. Cybern. 12 (1) (2021) 223–230.
[49] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (1965) 338–353.