Journal of Experimental & Theoretical Artificial Intelligence

ISSN: 0952-813X (Print) 1362-3079 (Online) Journal homepage: http://www.tandfonline.com/loi/teta20

A computer model of context-dependent perception in a very simple world

Francisco Lara-Dammer, Douglas R. Hofstadter & Robert L. Goldstone

To cite this article: Francisco Lara-Dammer, Douglas R. Hofstadter & Robert L. Goldstone (2017): A computer model of context-dependent perception in a very simple world, Journal of Experimental & Theoretical Artificial Intelligence, DOI: 10.1080/0952813X.2017.1328463

To link to this article: http://dx.doi.org/10.1080/0952813X.2017.1328463

Published online: 27 May 2017.

JOURNAL OF EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE, 2017
https://doi.org/10.1080/0952813X.2017.1328463

A computer model of context-dependent perception in a very simple world

Francisco Lara-Dammer[a,b,c], Douglas R. Hofstadter[b] and Robert L. Goldstone[c]

[a]Facultad de Ciencias, Departamento de Matemática, Escuela Politécnica Nacional, Quito, Ecuador; [b]Center for Research on Concepts and Cognition, Indiana University, Bloomington, IN, USA; [c]Department of Psychological and Brain Sciences, Indiana University, Bloomington, IN, USA

ABSTRACT

We propose the foundations of a computer model of scientific discovery that takes into account certain psychological aspects of human observation of the world. To this end, we simulate two main components of such a system. The first is a dynamic microworld in which physical events take place, and the second is an observer that visually perceives entities and events in the microworld. For reason of space, this paper focuses only on the starting phase of discovery, which is the relatively simple visual inputs of objects and collisions.

ARTICLE HISTORY
Received 30 April 2017
Accepted 5 May 2017

KEYWORDS
Human perception; scientific discovery; computer model; visual observation; context

1. Introduction

This paper concerns the first part of a larger project of the modelling of scientific discovery with a computer. Although discovery is an area that has been explored for several decades (for instance, Klahr & Dunbar, 1987; Langley, Simon, Bradshaw, & Zytkow, 1987; Schmidt & Lipson, 2009), the current study offers a number of new ideas, the most important of which is the development of a computer model that is intended to emulate the *perceptual activity* occurring during the process of scientific thought.

The second part of our project, which will be presented in a different paper, focuses on scientific discovery as a consequence of the perceptual activity. This is in contrast with previous AI approaches, where discoveries are derived either from numerical tables (which contain predefined variables that are given hand-coded values) or from equation-solving (for instance, Bridewell, Sánchez, Langley, & Billman, 2006; Langley et al., 1987; Schmidt & Lipson, 2009). To achieve this goal (at least partially), our model incorporates, as much as possible, a number of current theories about perception and scientific discovery. The project is akin to projects such as Copycat (Mitchell, 1990) and Seqsee (Mahabal, 2009), in the sense that these are models that involve perception and discovery of regularities in a microworld.

Perception in the context of discovery is thus the focus of this paper, which is limited to describing the first part of our project. We propose that how a human perceives gives rise to interpretation. Therefore, two or more different discoveries may arise from observations of a single situation, depending on the way some aspect of the situation has been observed by the interpreter. We illustrate this claim with an example.

An observer sees two identical objects, *A* and *B*, approaching each other at the same constant speed, following the trajectories seen in Figure 1. The nature of the objects is unknown; they could be in the sky or on a computer screen, but they appear to the observer as dots and the observer can see the motion. The objects start out at the bottom of the diagram, approach each other and wind

CONTACT Francisco Lara-Dammer flaradam@indiana.edu

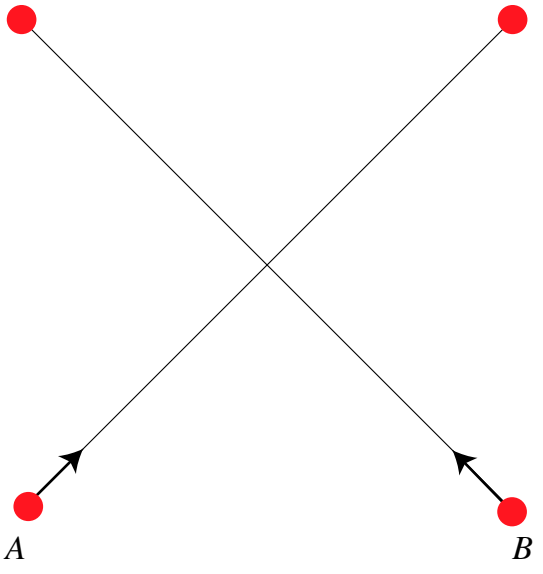


Figure 1. Trajectories of objects *A* and *B* initially approaching each other and then moving away from each other.

up as shown at the top. During this period, the observer watches them. At the end of this period, the observer wishes to know where object *A* is, and where object *B* is.

Using the formula *distance* = *velocity* · *time*, the observer can determine the pair of positions occupied by the two objects at any instant of time after their apparent crossing. What remains unknown, however, is which object, *A* or *B*, is on the top left and which is on the top right of the diagram. This depends on whether the objects collided when they met, or passed through each other (e.g. if the objects are made of gas). The observer's answer thus will depend on whether the objects were seen as colliding or as passing through each other. This project involves this type of ambiguous phenomenon.

Situations like these, where what is discovered depends on what is perceived, are not rare. For instance, Louis Pasteur (1822–1895) discovered that micro-organisms cause diseases. However, Claude Bernard (1813–1878) disagreed with Pasteur. According to Bernard's theory of the Internal Environment: 'The stability of the internal environment is the condition for free and independent life.' (To use an analogy, animals, including humans, evolved in such a way that we have a portion of the sea inside our bodies, and so we live in a similar environment to that in which our ancient predecessors used to live. If our inner sea's chemical structure is altered, we perish.)

Thus, for Bernard, the presence of micro-organisms in an animal is a consequence of its disrupted environment (i.e. this presence occurs after the natural chemical structure of the animal's inner fluids has been altered); for Pasteur, this environment is disrupted by micro-organisms. Our point here is not to question these French scientists' conclusions, but to illustrate the idea that interpretation plays a key role in discovery. According to Charles Gross (a neuroscientist at Princeton), Bernard's theory of the constancy of the internal environment eventually led to a theory of the development of (self-regulated) complex nervous systems, and subsequently, to cybernetics and the development of self-regulating machines (Gross, 1998). In contrast, Pasteur's theory seems akin to the development of systems (e.g. a traffic light) whose internal representations are created in response to external objects.

In essence, the phenomenon of perception in the case of ambiguous situations arises when there are two or more states of the world that are perceived by our senses, but the mechanisms that cause the transition from one state to the other are hidden to us. In the example of the two moving balls, the missing information is what happens between frames, and in the example of the disease, the missing information is what happens between the animal's wellness and the animal's illness. The two states are

Figure 2. A diagram representing the transition from one state of the world to another. Both states can be described qualitatively. The cloud in the middle represents missing information not present in the qualitative description.

illustrated in Figure 2, where the 'cloud' in the middle symbolises the fact that there is some missing information between the states. Note that the states are observed at one level of description (e.g. as seen by the naked eye), while what is in the cloud occurs at a lower level. What is on the first level of observation corresponds to what Forbus (1984) calls a 'qualitative state.'

Since visual perception is central to this project, part of the project focuses on the missing-information cloud at the visual level and how humans can make discoveries in spite of this limitation. The computer model at the heart of our study has two parts: the *world emulator* and the *interpreter*. The world emulator shows a situation to be observed by the interpreter. The interpreter simulates an observer perceiving the situation visually. The emulated world is populated by objects that are moving while they are being observed by the interpreter. Thus, one of the key tasks of the interpreter is to keep track of individual objects. It uses a mechanism of visual object-tracking (described later) based on theories of apparent motion proposed by Marr (1982) and Ullman (1979). Object-tracking, however, is only a starting point that helps the interpreter to find relations between the objects, and ultimately to make discoveries in the world.

Since perceptual activity is central to this project, we have striven to avoid, whenever possible, the style of the AI approaches briefly referred to above. Our model attempts to be more psychologically realistic. In summary, the interpreter part of our model was designed, at least theoretically, to do the following tasks:

- track objects in a dynamic environment;
- discover relations among the objects (such as collisions);
- make qualitative and quantitative estimates of aspects of the world, such as the volume of a container;
- derive 'laws of nature' from relations perceived by the system;
- make modifications of certain variables defining the world to see if the discoveries still hold in modified worlds (e.g. if *number of collisions* and *area of a container* are found to be inversely proportional in a circular container, is this still true in a square container?). This aspect of the programme can also be used for the purpose of generalisation.

This paper focuses on the first and second tasks. Our second paper will focus on the other tasks. Additionally, our model takes into account the crucial influence of context in perception, and thus in discovery. We hope that by incorporating these elements into our model, we can make a contribution to real-world problem-solving. In fact, the tasks just mentioned are related to quintessential activities involved in the making of scientific discoveries in the real world.

Of course, making a computer model of all the psychological aspects of human discovery is too ambitious a goal. One reason is the fact that many of the mental processes involved in perception and understanding are still unknown (especially when these processes involves deeper processing than just the initial layers of perception through the senses). Nonetheless, there is still some hope of making progress if we remember that much research has already been done, and many theories have been developed. This project resulted from our adapting and combining such research and theories. Our model was built in order to be able to deal with ambiguity in perception and interpretation. In contrast to models such as AM (Lenat, 1979), where ambiguity was explicitly avoided, we welcome ambiguity as an aspect of discovery that is truly human.

Figure 3. The model running a simple simulation. The right side shows a world consisting of three balls moving about in a circular container. The left side shows NINSUN's view of the world. Additional information (such as the number of collisions per minute) can be seen in other windows (not shown here).

2. Overview of the computer model

As was mentioned in the introduction, our computer model consists of two parts: the world emulator and the interpreter. The world emulator (called *Tricycle*) works independently of the interpreter, although the interpreter has the ability to manipulate certain parameters in Tricycle, thus modifying the simulated world. The simulated world is a collection of interacting objects with some similarity to real objects and their interactions. The interpreter, which is the *cognitive* part of our computer model, was designed to engage in observation of the world and to derive *relations* among the objects in it. We have been inspired by Michotte's theory of the perception of causality (Michotte, 1963) to emulate an approximation to the perception of physical properties of real objects.

The interpreter programme is called NINSUN, after the goddess in Sumerian mythology who interpreted her son's dreams. It can also be understood as an acronym for Novel INterpretations of Scientific UNderstanding. We will refer to the interpreter by this name.

Figure 3 shows our computer model at a particular moment during a simulation. In this figure, NINSUN (left side) is observing a simple world (right side) consisting of three moving balls. The arrows and dots on the interpreter side indicate its representation of motion over a short amount of time. This representation and more details about the interpreter's side will be explained more fully as we proceed.

For those who are interested in watching the programme in action, simulations can be run using a set of commands that can be typed in a command line placed on each side of the big window, shown at the bottom of Figure 3. For convenience, the set of commands for each specific simulation described in this paper is stored in a specific script file. These simulations will be referred to by the name of the appropriate script file.

There are two types of scripts: those that can be read by Tricycle and those that can be read by NINSUN. The former is a collection of Tricycle commands that create a world that can be observed by NINSUN with settings that can be specified by the user. The latter consists of a file with interpreter

commands that usually specify parameters that control how NINSUN observes the world. An example is the parameter *direction*, which reflects how much importance NINSUN gives to the direction of motion of an object that it is trying to track.

One way to refer to a simulation is by the name of the script. Another way to refer to a simulation is to watch a pre-recorded video of it. The videos have the advantage that they focus in on specific aspects of a simulation, which would take much longer to see if we were to run the full simulation. The recordings are available on the archives at the following website:

`http://www.indiana.edu/~pcl/2016/03/ninsun/`

and will be referred to throughout this paper. The scripts are available as part of the programme. For instance, the script that creates the world in Figure 3 is in the file 'threeSmallBalls.txt' and the video that shows the corresponding simulation is in the archives under the link 'Overview V1.'

2.1. The necessity of keeping the worlds simple

In this paper, *scientific thinking* can be described roughly as the mode of thought people adopt when they are trying to understand some aspect of the world. The thoughts leading someone to a scientific discovery and the cognitive activities a student may carry out while learning a specific topic in science are two examples of scientific thinking.

Much of scientific thinking involves condensing situations into a small number of simple concepts. The visual system and visual imagination play a fundamental role in this process. For example, moving objects often can be considered to be moving *dots*, allowing the thinker to ignore the shape of the object, when shape is irrelevant to the understanding of the situation. Diagrams and abstractions facilitate scientific thinking. When the thinking is productive in the sense of Wertheimer (1945), it has been recognised as being characterised by its *logical economy* (called the Efficiency Principle in Lara-Dammer (2009)) (Klahr & Simon, 1999; Lara-Dammer, 2009; Miller, 1987). This is one reason why we have striven for simplifying our microworlds as much as possible.

Static diagrams, although often useful, lack motion and they are not what we are modelling. Tricycle was created in order to show objects in genuine motion, not just still snapshots. It is, however, limited to two dimensions. Thus our system is a model of scientific perception and discovery in a 2-D environment.

The worlds built by Tricycle involve simple rules and simple shapes, such as lines, circles and triangles. There is set of tools that can turn a geometric shape into an object with physical properties, such as a circle into a ball, or a line segment into a wall. Certain physical properties, such as friction and attraction, can also be added to some of the objects.

The following sequence of Tricycle actions results in a circular container with three moving balls inside it, similar to the one in Figure 3.

(1) draw a large circle;
(2) make the circle a wall;
(3) create three small non-intersecting circles inside the container;
(4) make the small circles balls; and
(5) give random velocities to the balls.

Notice that even though a world is created by following a set of relatively simple rules, the world itself can become arbitrarily complex through repeated application of the rules.

2.2. NINSUN and Tricycle

NINSUN visually observes a world created by Tricycle. By 'visually observes,' we mean that NINSUN infers some spatial relations or properties that normally can be visually perceived by a human – for instance, proximity of two objects, the direction of motion of an object, the speed of an object and the size of an object.

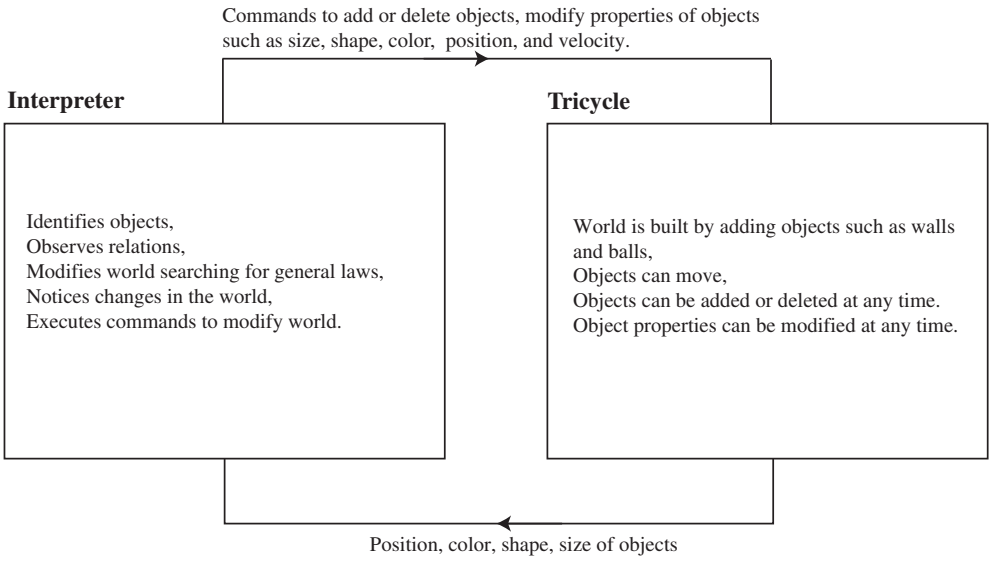


Figure 4. Connections between NINSUN and Tricycle.

NINSUN could have been created to receive visual input data from a camera, just as humans receive visual data through their eyes. However, the process of manipulating such complex images was computationally too demanding for this project. Instead, the coordinates, size, shape and colour of simulated objects are passed to NINSUN on a moment-to-moment basis. It is up to NINSUN to decide which coordinates correspond to which objects. In some cases, some attributes of objects, such as size and colour, are not given to NINSUN. This is because we have radio buttons corresponding to these attributes, some of which can be deselected, in which case they are not passed to NINSUN.

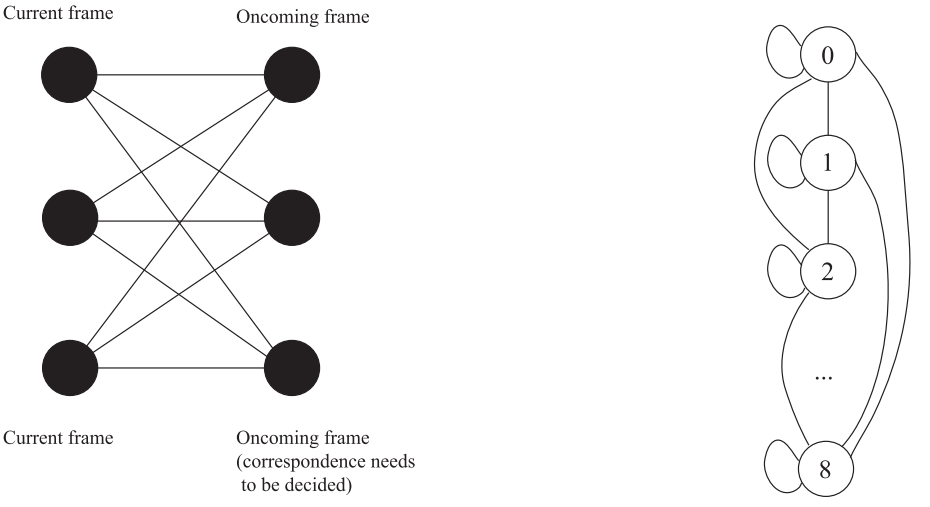
It is important to emphasise that the *identities* of objects created in Tricycle are not passed to NINSUN. A particularly difficult problem is therefore to distinguish and keep track of two objects that have the same (or close) coordinates. This topic is addressed in Section 3 (especially Section 3.4).

It is vitally important that NINSUN exploit raw data as much as possible. For example, if two objects are moving, NINSUN should be able to judge which of the two is faster based only on what it has been observing (in this case, the coordinates of the objects, supplemented by a short-term memory that remembers how an object was moving in the last milliseconds), without being helped by an external aid (such as having Tricycle calculate the objects' speeds and then pass them to NINSUN). A diagram of relationships between NINSUN and Tricycle is given in Figure 4.

3. Tracking of objects

The derivation of a mathematical law happens only after NINSUN has done several preliminary tasks, such as object tracking, relation detection and manipulation of the world. The first thing NINSUN needs to do is object-tracking. In a static world, this task is trivial, but in a dynamic world it is difficult. NINSUN tackles object-tracking using the motion correspondence problem (MCP) theory of Ullman (1979) and Dawson's implementation of an auto-associative connectionist network (Dawson, 1991).

Before we continue with the description of our connectionist approach to object-tracking, we would like to address briefly other approaches. Consider Milan, Roth, and Schindler (2016) or Nayak, Zhu, and Roth (2015). Although mathematically very sophisticated and even biologically inspired, these works focus on computer vision rather than on psychological aspects of human vision. These systems are more oriented towards practical applications than towards the modelling of human motion perception.



(a) Network tracking the positions of three moving balls

(b) The basic architecture of the auto-associative neural network to solve the MCP

Figure 5. (a) Three moving balls going from the positions on the left to the positions on the right. Each of the segments suggests a possible match and represents a unit for the network. (b) Another representation for the units with circles. Each circle represents a unit for the net and the lines connecting the units are connectionist weights. There are 9 units and 81 weights for this network. (For n balls, there are n^2 units and n^4 weights.) There is a square matrix of size 9×9 associated with this neural net, whose 81 values are the connectionist weights.

In contrast and in a nutshell, Dawson's model is a connectionist auto-associative neural network, which models human motion perception. The weights in this neural network are determined by consistency. Inconsistent nodes inhibit each other; consistent nodes excite each other. What it means for two objects to correspond to each other is that they are the same object over time.

To illustrate the basic architecture of Dawson's network, consider a small example of three small balls moving inside a container. Figure 5 summarises and shows the basic (non-extended) architecture and terminology for this network (for the three balls).

Dawson's model, although useful, needs to be extended to address certain tricky aspects of the generated worlds. We will explain the details of the proposed extension in the next section.

3.1. Extensions to Dawson's model

First, it is assumed that balls do not split or merge during the simulation. Of course, in real life merging or splitting might occur, such as with water drops, but NINSUN is built without such sophisticated knowledge.

Second, Dawson's model is based on exactly two frames: the current frame and the oncoming frame. Since our model needs to deal with more complicated situations, it is necessary to store more frames. The number of frames to be stored is a parameter that in principle depends on the degree of attention. (See below and Figure 7.) However, this number cannot be smaller than five (at least in the domain of bouncing balls) because this is the minimum required for collision detection. (See Section 4.1.) Most of this project has been carried out using exactly seven frames.

Third, in an auto-associative neural network to solve the MCP, the selection of the winner units is of paramount importance. In Dawson's simple examples, a threshold works fine. However, in a more complex world, a single threshold is not sufficient to decide on the winners, because the positions of the objects, the speeds of the objects, the sizes of the objects, etc. may change at any time, and a threshold that worked well at one time may not work at another time.

Fourth, it is necessary to specify the observables (called 'constraints' in Dawson (1991)) to be considered in the network. In addition to the three specified by Dawson (proximity, relative velocity, and integrity), NINSUN also uses the *direction* of a moving object. We call this the *direction observable*.

In most cases, direction is a very useful piece of information to help solve the MCP, since a ball's new direction is likely to be close to its most recent direction.

One might worry that in the case of a collision, NINSUN could get confused because the objects' directions change abruptly. This worry would be justified if NINSUN relied solely on direction, but in most cases other observables help NINSUN to keep track correctly of objects in motion.

The choice of direction as a relevant observable seems intuitive, since people have the illusion that a moving object leaves a 'line' behind it, like an airplane in the sky. There is also scientific evidence that 'direction' is a primitive concept not only in humans but in other animals. Certain populations of neurons respond to a stimulus associated with a specific direction (see Georgopoulos, 1988).

It should be mentioned that an object's *velocity* could be used as an observable for object-tracking, since the velocity vector includes the direction of the object, but speed (the magnitude of the velocity vector) is derivable from the distance between the object's current position and its previous positions (as stored in earlier frames), so velocity would be redundant information. (Actually, the velocity observable was temporarily implemented, and we learned that keeping track of the direction alone was slightly better than keeping track of the velocity vector for object-tracking.)

Color, shape, and size are also sources of information used in object-tracking. However, Dawson's model does not consider these features Dawson (1991), since they do not involve the motion and position of the objects. (According to certain researchers, position rather than appearance is the most relevant information for object-tracking Dawson (1991).) Whether or not appearance is less important than motion, NINSUN can use these aspects of appearance to help it track objects.

If the objects to be tracked have some appearance attribute, (say, colour) that is identical for them all, then this attribute will obviously play no role in object-tracking, since its contribution to the neural network will be the same for each object.

It should be mentioned that all observables are multiplied by *weights* that can be modified by humans or by NINSUN. A user can set these weights and thus help NINSUN find a way to identify objects. As the weight of an observable increases, that observable will have a larger impact on observed motion correspondences. For example, a green square would be more likely to be perceived as corresponding to a red square than a green circle if shape is given a greater weight than colour. In the rest of this paper, the term 'weight' will refer to the weight associated with an observable, not to the connectionist weights in Figure 5(b).

NINSUN, of course, takes advantage of situations where there is variation of a given appearance attribute. If all the objects have different colours, for example, the only attribute needed to identify an object would be its colour, and the rest of the weights could be set to zero without changing the behaviour of the network. In general, when at least one appearance attribute exhibits a range of different values, then there is less ambiguity, and so object-tracking is easier; NINSUN is more challenged when the appearance attributes are identical for all objects.

All in all, NINSUN currently can use seven observables to identify objects: *nearest neighbour*, *relative velocity*, *integrity, direction*, *colour*, *shape*, and *size*. These all contribute to the tracking of objects by the neural network. If, at some point during a run, some attribute is conjectured to be more important than others, the values of the weights can be modified easily. Moreover, if, during the evolution of the programme, it is conjectured that some other attribute of objects might be useful, a new corresponding observable can be added easily. Conversely, if, some attribute is seen to be unuseful, the corresponding observable can be removed easily.

Once NINSUN is able to track objects consistently, it is ready to tackle higher levels of perception, involving patterns and relations among the moving objects. These range from the detection of a collision all the way up to mathematical conjectures of laws holding in the microworld. Roughly speaking, we could say that the processing of information flows from one layer to the next. However, we will see that this is only a simplistic way to describe the model. The actual flow is not so simple,

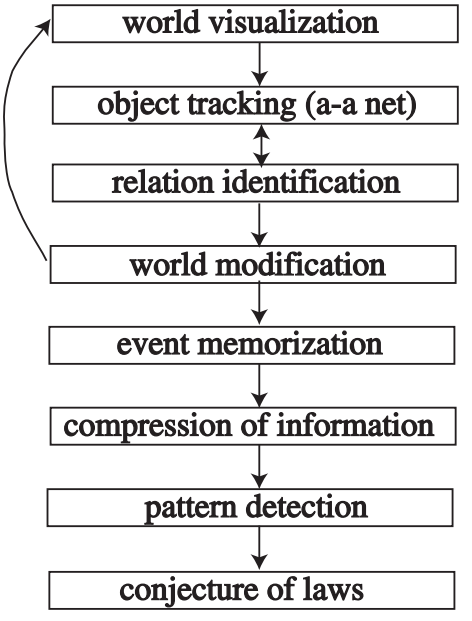


Figure 6. A broad overview of the flow of information in NINSUN, described as layers of processing information. Notice that there is feedback from the perception of relations (which is influenced by context) to the auto-associative neural network that allows object tracking.

as some processes occur in parallel with others and the flow can move bidirectionally, as is shown in Figure 6.

One thing that needs to be mentioned is that the machinery that does object-tracking has certain subtleties that go beyond Dawson's design. Without them, object-tracking simply would not work. One of these features is the updating of frames, and another is the calibration of weights. The current position is stored in a structure that we call 'frame 1.' When frame 1 needs to be updated, the information that was in frame 1 becomes frame 2, and the information that was in frame 2 becomes frame 3, and so on. What was in the oldest frame is lost. The number of frames can theoretically be adjusted at run time, but currently this number is set to the fixed value of seven. Therefore, the oldest frame is frame 7.

This is represented schematically in Figure 7, where four consecutive frame updates have been made while an object moves from position *A* towards a wall, and after bouncing off the wall, moves towards position *B*. Each circle shows the moving object at an instant. Frame 1 is at the top and contains the latest information; frame 7 is at the bottom and contains the oldest information. Currently, updates take place every 50 ms, but this parameter can be adjusted. In the leftmost snapshot, the circle that is highlighted indicates the most recent position (*A*) of the observed object. In the figure, this state of the world is replaced by a new state three times. The highlighted frame always has the same information but its number changes from 1 to 4. In the rightmost snapshot, frame 1 contains the information that the current position is *B*.

From this seven-frame data structure, information about the object can be derived, such as its velocity vector and its acceleration vector, by subtraction. In this way, this data structure holds information that can be used by NINSUN to draw conclusions and make conjectures about the motion of objects.

3.1.1. Calibration of weights

As was mentioned in Section 3.1, there are currently seven numerical weights (distance, relative velocity, integrity, direction, colour, shape and size) that are used to calculate the n^4 neural-net

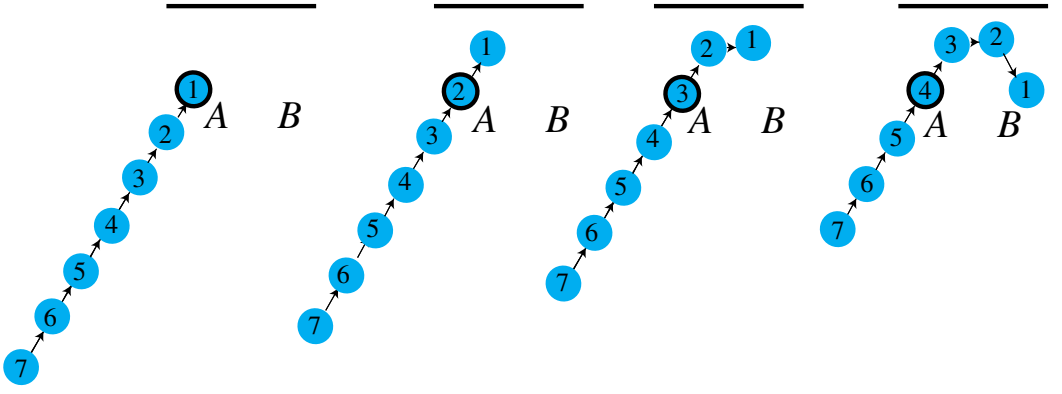


Figure 7. The seven-frame data structure belonging to an object that is moving from a position *A* towards a wall, and then, after bouncing off the wall, towards position *B*.

weights, and all seven of these lie in the interval [0, 1]. It is crucial to seek optimal settings of all these weights. This is a very difficult matter, since the number of possible combinations of weights is extremely large. Luckily, human intuition is pretty good at guessing effective settings for the weights. Occasionally, however, human intuition is misleading. For instance, it might seem reasonable to set all the weights to 1 (the maximum value), but NINSUN does a better job when certain weights are lower than that.

Exploring each weight on its own, with all the others set to zero, is a way to explore the effects of each weight on object-tracking. In a simple world set-up with three balls moving inside a container, if all weights are set to zero except the relative-velocity weight, identification works. This agrees with Dawson's claim that this parameter does well in most cases. Moreover, a low value of this parameter works just as well as a high value.

What about other weights? The distance weight all by itself behaves similarly to the relative-velocity weight. However, the integrity weight (the one that gives importance to the fact that objects do not merge or split) works only when it is in certain ranges (around 0.3 and above 0.7). Furthermore, the integrity weight does not work well when there are five or more balls. The direction weight alone works similarly to the relative-velocity weight, except (as expected) when the balls undergo collisions. The direction weight is very important, however, and will be discussed later in the paper. A few videos showing the influences of certain weights can be found in the archives under the links 'Object Identification V*i*' ($i = 1, \ldots, 4$). Interpreter scripts are also available. These are 'singleParameterTest.txt,' 'singleParameterTestSize.txt,' and 'singleParameterTestDirection.txt.')

Another interesting challenge involving the calibration of parameters arises when two or more objects have identical appearances and move in an identical manner. For instance, suppose three balls with identical appearance are moving with the same speed, one next to the other, and in parallel lines. In this case, the identical attributes of the objects do not aid NINSUN in identifying the objects. Instead, NINSUN must rely on those parameters that make a difference, and if these weights are set to zero or a very low value, the identification of the objects will be poor. A simulation of this kind of situation with poor object-tracking is found in the archives under the link 'Object Identification V5.' An interpreter script that allows this simulation to be run is found under the name 'identicalDirectionsTest.txt.'

3.2. A case study for the MCP: the world of walls and bouncing balls

The 2-D microdomain of walls and bouncing balls can be varied in many ways, and this richness makes the microdomain especially useful for testing the models of cognitive processes in the interpreter model. The controlling parameters can be easily modified for specific tests and purposes. For example, at any time the objects' sizes can be changed, an arbitrary number of objects can be inserted or

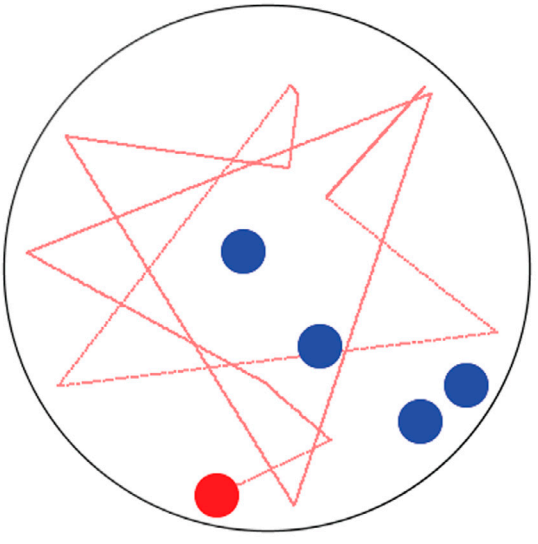


Figure 8. Trajectory of the red ball over 5–10 s.

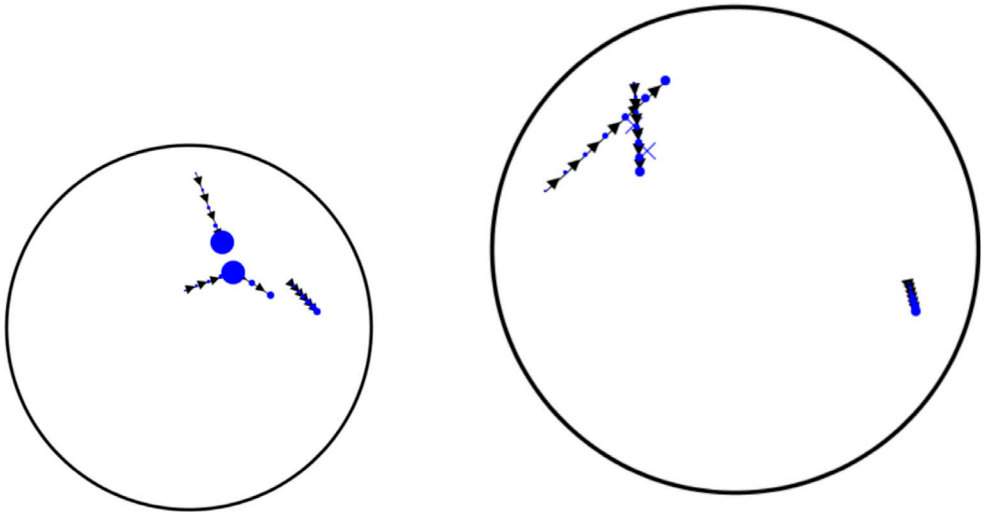
removed, and walls with specific angles and positions can be inserted. In this way, the world can be tuned to range from easy to hard to interpret.

The events in this microdomain look so random that the average person will think at first sight that there is no possible way to make scientific conjectures and discoveries. Figure 8 shows five identical balls, one of which is red, that are moving at medium speeds inside a circular container. The red ball has been tracked for about 5–10 s. In this relatively short amount of time, the lengths of the straight lines constituting its trajectory do not suggest a regular pattern. Yet careful attention allows a curious mind to discover regular patterns over longer periods of time. The very fact that this microdomain seems so random is what it makes it a highly appealing microdomain for study. We are also interested in other microdomains, but this one had sufficient richness to keep us busy for a long time.

Additionally, this microdomain allows us to make discoveries of relations such as that connecting the frequency of bouncing against a wall and the speeds of the balls. This type of discovery corresponds roughly to the discovery of relations in the field of ideal gases. In that case, the balls are very small and the number of balls is very large (simulating the fact that a gas is composed of billions of molecules). Hopefully, from these relations some of the laws of gases can be derived.

We distinguish two modes in the microdomain of bouncing balls and walls. In one mode, balls *collide* when they coincide in position (at least as perceived by the eye), and in the other mode, balls pass through each other. The first mode resembles a billiard table with balls rolling on it and occasionally hitting each other. The second mode resembles a TV screen showing a juggler performing with balls: occasionally some balls seem to pass through others from certain angles. (Of course, humans know that there is actually a third dimension and that some of the balls are in fact passing behind others.)

If all the objects are identical in appearance, the second mode is slightly more difficult for the object-tracking challenge. This is because there is a chance that the interval of time when the two balls are having contact (which includes overlapping) might occur during NINSUN's creation of the two corresponding consecutive frames. In a case like this, the distance between the centres of the objects remains smaller than the smallest of the distances between each of the objects' corresponding consecutive frames. (See Figure 9.) This fact is a cause of ambiguity, which is an interesting topic from the cognitive point of view. Essentially, in this case we need to deal with occlusion (one or more balls 'hiding' behind another). The detection of collision and pass-through-each-other events is explored in Section 4.1.



(a) Collision Event (b) Pass-through-each-other event

Figure 9. Three moving balls inside a circular container, seen from NINSUN's standpoint. (a) The two top balls have just collided with each other, and NINSUN is sending a signal of a collision event (bigger circles on the third frame). (b) The two top balls are passing through each other and NINSUN is sending a signal of a pass-through-each-other event (crossing lines). The distance between the objects can be smaller than the average distance between each object's consecutive frames. This makes object-tracking harder in the second case.

3.3. The selection of winners in the neural network

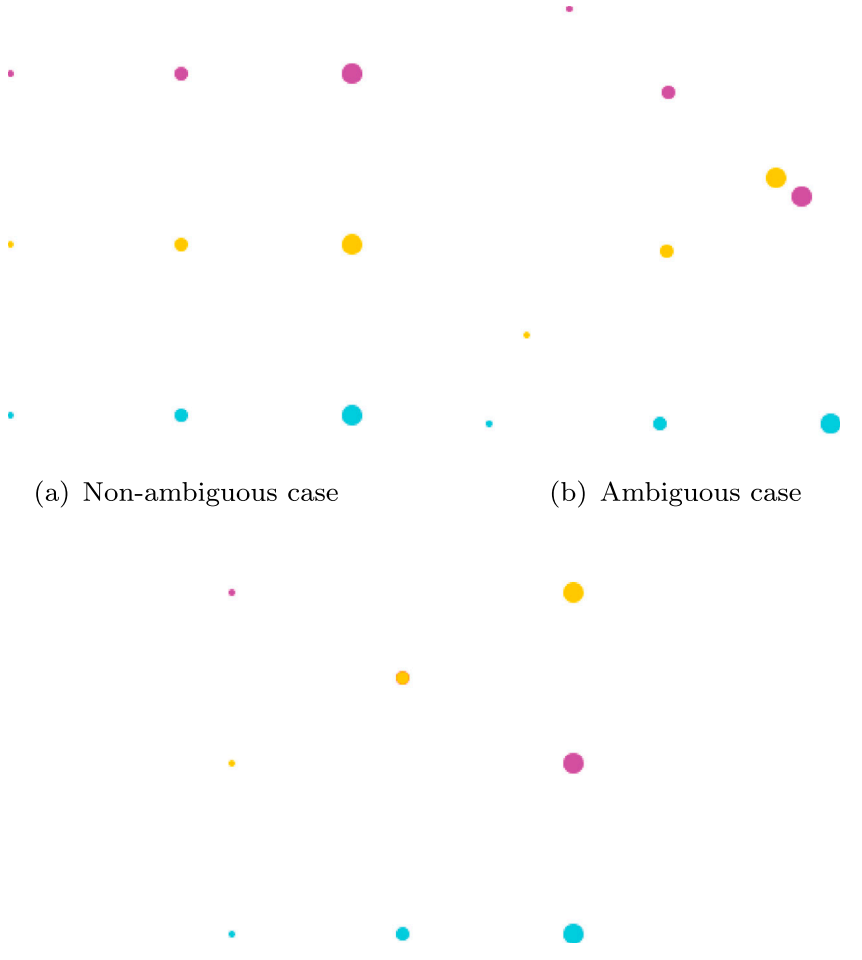
The neural network has a current frame with positions that are already identified and it is given a new frame with positions that are not yet identified. For each position in the current frame, the neural network has to select a position in the new frame to match this position. This selection is based on competition of the units emanating from a single position in the old frame. (See Figure 5(a).)

The winner of this competition (i.e. the unit with highest value) is the unit that is selected. A very important constraint in the bouncing-balls context is the Integrity Principle (see Dawson, 1991), which states that a 1–1 correspondence must be established between the positions in the old frame and the positions in the new frame. To help in tracking identification, NINSUN can make use of the knowledge that balls do not merge.

For the sake of clarity, we will explain the mechanism of the selection of the winner with numerical examples. Let us suppose there are three identical balls moving. NINSUN sees no difference among them except for their positions and velocities. In order to distinguish these balls, let us label them *A*, *B* and *C*. (These labels are not part of NINSUN's algorithms.) Each of the three cases in Figure 10 corresponds to three frames of three bouncing balls.

The three cases were generated by a separate programme that runs as part of Tricycle and that uses the neural network to solve the MCP. All weights in this neural network were set to 1, except for those of colour, shape and size, which were set to 0. Henceforth we will call the current frame 'old frame.' The smallest dots represent a frame that is older than old frame (we can call it 'older frame'), the medium-size dots represent old frame, and the largest dots represent new frame. (Colours are used in this diagram instead of labels. The magenta dots represent object *A*, the orange dots represent object *B* and the cyan dots represent object *C*. The auto-associative neural network knows neither the sizes nor the colours of the dots.)

Older frame can be used to make *direction* a member of the set of observables of this neural network; direction is not part of Dawson's model. (The direction weight, like any other weight, can be reset to zero, in which case direction would have no effect on the output of the neural network.)



(a) Non-ambiguous case

(b) Ambiguous case

(c) A hard case: Two points overlapping

Figure 10. Three Frames.

There are nine units in the network, one standing for each possible match. The units are labelled by the label of one ball in old frame followed by the label of one ball in new frame. For example, unit *BC* corresponds to the match identifying ball *B* in old frame with ball *C* in new frame. There is usually no ambiguity as long as the objects do not get too close, but problems can arise when some objects approach others. The cases illustrated in Figure 10 will be considered below.

In Figure 10(a), there is no ambiguity, and column (a) in Table 1 shows very strong winning matches. In Figure 10(b), there is ambiguity due to the proximity of two balls in new frame. Column (b) of Table 1 shows a winning match that was preferred by the network, although the matches $A \longrightarrow A$ and $B \longrightarrow B$ are also possible under the Principle of Integrity, if a threshold such as 0.13 were taken (as suggested in Dawson, 1991).

In Figure 10(c), two balls in old frame overlap and two balls in new frame are swapped. The corresponding unit values that were preferred by the network are shown in column (c) of Table 1. However, in this case the correspondence is not 1–1. This would be acceptable if splitting and merging of balls were possible, but not in the microdomain of walls and balls.

Table 1. Unit values of the three cases in Figure 10.

Unit values and winning matches of three cases in Figure 10						
	case (a)		case (b)		case (c)	
Unit	Value	Winning match	Value	Winning match	Value	Winning match
AA	0.5700		0.3345		0.2193	
AB	−0.0545	$A \longrightarrow A$	0.3986	$A \longrightarrow B$	0.3338	$A \longrightarrow B$
AC	−0.2610		−0.4067		−0.2799	
BA	−0.0432		0.2184		0.1189	
BB	0.4520	$B \longrightarrow B$	0.1506	$B \longrightarrow A$	0.4119	$B \longrightarrow B$
BC	−0.0432		0.0309		−0.2033	
CA	−0.2610		−0.2199		0.0235	
CB	−0.0545	$C \longrightarrow C$	−0.2306	$C \longrightarrow C$	−0.3560	$C \longrightarrow C$
CC	0.5700		0.6252		0.6400	

The fact that the unit values in case (c) of Table 1 are not suitable for our microdomain does not mean that the neural network is giving us the wrong values. They are correct, and they would be suitable in a domain where objects can merge or split, like water drops. In the microdomain of walls and balls, however, we have the extra knowledge that balls cannot split or merge, and we would like to use this knowledge.

An output like case (c) in Table 1 is rare but it can happen. Such cases seriously confuse NINSUN. In a worst-case scenario, two objects would be seen as having the same position and they would thus become merged. Since the next frame depends on the current frame, the next values assigned to these objects have a high probability of being once again nearly identical, and thus a chain of wrong assignments can be generated, until the network recovers, either by changing parameters settings (this can be done by a human) or by an auxiliary mechanism of NINSUN. However, while the chain of wrong output lasts, NINSUN does not see three objects, but just two, or possibly even one. A human observing the interpreter's and Tricycle's windows would see NINSUN seeing objects as overlapping while at the same time the actual balls in the world of Tricycle are not overlapping. This kind of behaviour of the model of course has to be avoided because when humans see three non-overlapping objects, they do not get confused and think they are seeing two objects.

Theoretically, the problem could be solved by increasing the weight attached to integrity. However, in practice this is not easily achieved at run time, as will be explained in Section 3.4. Once two rows in the connection matrix (See Figure 5.) are nearly identical, they need some time to become distinct again. In order to solve this problem, we need to find a disambiguation strategy that will guarantee efficient 1–1 identification. NINSUN can be set to choose among several paradigms to select the winner whenever an ambiguous situation is presented. The paradigms are briefly described below.

3.4. One-to-one policies

It was mentioned before that if the objects do not split or merge, NINSUN can make use of this knowledge to improve its perception of the world. Humans too use heuristic or intuitive reasoning when they know that objects conserve their integrity but when one or more objects seem to be missing. In our system, we may need to add further strategies to the neural network in order to produce reliable object-tracking, but for the time being the strategies below are what we use.

3.4.1. Adjustment of the integrity weight

Theoretically, if the integrity weight is well adjusted, NINSUN will not see one object when there are really two (or three). The problem is that in general it is not easy to find the 'right' value, as this value depends on the other weights. If the integrity weight is set too low, NINSUN will be prone to guessing that objects have merged. If it is too high, it might enforce one-to-one–ness by creating what looks, to an external observer, like a random assignment of positions to the objects An intermediate setting is needed.

Furthermore, certain settings of this weight might work well in one world, but changes made to that world might make those values completely inappropriate. Of course, the weight can be readjusted at run time, but this might take too long. In short, relying solely on this parameter is not reasonable. In some worlds it might suffice for object-tracking, but in others it will not.

3.4.2. Adding noise to the positions of the objects

If random noise is added to the positions of the objects, then two objects that are very close to each other might get some 'separation'. As a result, those objects would have less chance of being confused. A disadvantage of this method is that there is no guarantee that merging will not occur, and of course the objects will be seen at slightly different positions from their true positions, and of course, the word 'slightly' has some amount of blur. In practice, though, this method seems to work fairly well, and can be combined with the previous policy for better results.

3.4.3. Dependent-selection algorithm

In this method, nothing is modified in the network. In the case shown in Figure 10(c) (also in Table 1), the network did not give up the second-place winner, because the balls are very close, almost on top of each other. As a consequence, they have essentially the same coordinates, and more than one winner is eligible for a single unit emanating from the same position in old frame.

In such cases, the strategy is to pick one of the winners at random. However, once one winner has been chosen for a given object, it is necessary to make sure that the winners for other objects do not conflict with each other (i.e. never select a winner that was previously selected).

In the example of Figure 10(c), either unit *AA* or unit *AB* is eligible for the first correspondence. Suppose *AB* is selected. It will then be necessary to choose from the remaining set of unit winners {*BA*, *BB*} in such a way that we do not end up choosing the same extension of two different positions in old frame. The only possibility is thus to select unit *BA*, as *BB* conflicts with the previous selection *AB*. Choosing *BA* leads to the correct solution: $A \longrightarrow B$, $B \longrightarrow A$, and $C \longrightarrow C$. The choice *BB* leads to the inconsistent solution shown in column (c) of Table 1.

In developing this model, we observed a pattern in the units every time that the values of two or more units in the network. (see Figure 5(a)), starting at the same position in old frame, exceeded a threshold value. Under this condition, the same number of units corresponding to matches starting at another position in old frame had values that exceeded the threshold too. (This was a consequence of underlying symmetries in the network.) For example, the threshold could be taken as 0.1; two corresponding units would be *AA*, *AB*, and the other two corresponding units would be *BA*, *BB* in the column labelled 'value' of case (c) of Table 1. If this symmetry did not exist, this method could fail.

The case we have considered contains only one possible conflict, but when there are more balls in the world, there are situations where several conflicts might appear. In such cases, every new selection must be made as a function of previous selections.

It is interesting to note that if all the weights are set to zero in the neural network, the algorithm creates a one-to-one mapping between objects in old frame and new frame. Unfortunately though, this mapping is only one mapping among many possible ones, and is not guaranteed to be consistent with the observations.

In short, when the network offers two or more winners for a set of units corresponding to matches emanating from the same position in old frame, the first selection is made randomly, and the remaining selections depend on the previous selections.

3.4.4. Occlusion algorithm

For this strategy, as in the just-described dependent-selection algorithm, nothing is modified in the network. The strategy is based on the assumption that if two objects, say *A* and *B*, have very close coordinates, then it does not matter which coordinates correspond to *A* and which coordinates correspond to *B*, for the purpose of identifying the objects.

More generally, when two or more objects overlap or are very close, NINSUN knows that they are not one object, and it creates an equivalence relation of occlusion. At such times, all the objects that are on top of each other (or are very close to each other) are identified as a single class.

In order to disambiguate, select the position in old frame corresponding to one object in the occlusion class. All the values of the units corresponding to matches emanating from the *selected* object in old frame are taken into consideration. The values of *other* units corresponding to positions in old frame that are in the same class as the selected one are ignored. Each unit corresponding to a match emanating from a position in old frame that belongs to the occlusion class is given the values of the units emanating from the selected position in old frame in descending order.

It is interesting to note that if all the weights are set to zero in the neural network, this algorithm makes NINSUN see all objects as overlapping – in other words, it is as if all the objects in the world were hidden by other objects (except for one in the very front). This behaviour is the exact opposite of the dependent-selection algorithm.

3.4.5. Which algorithm?

In general, any algorithm could be used, but in certain cases, for simulations that pose special challenges, one has to choose between the occlusion algorithm and the dependent-selection algorithm.

4. Relation identification

Once objects have been identified in a dynamic world, NINSUN then has a chance to observe *relations* among them. A preliminary identification of the objects in a dynamic world will give NINSUN a chance to observe *relations* among them, which in turn can have an effect on their identification, in a retroactive way. Relations are of two types: local and global. The former refers to a relation that an object has with another object, or other objects. The latter refers to those relations that involve the environment as a whole. For example, the relation 'hits' is local because an object *P* can hit another object *Q*. By contrast, the relation 'number of balls in world' is global.

Collisions and pass-through-each-other (henceforth, 'pass-through') events are the only local relations that the programme currently supports, but a future version is planned to include others, such as attraction, repulsion, stickiness, splitting and fusing. This section will explain how NINSUN detects collisions and pass-through events.

4.1. Collision detection

To simplify, let us assume that all balls move with constant velocity and in a straight line until a collision occurs. Two types of collisions can occur: collisions with walls and collisions with other balls. From a computational standpoint, the two types have similarities but they also have subtle differences.

Collision detection takes into account conditions that always occur in collisions (but that are not sufficient to guarantee that a collision occurred). These conditions should be selected in such a way as to be as independent of each other as possible. Among these are *abrupt change of the velocity vector* (which we refer to as *acv*, and which includes the direction of a moving ball), *proximity to another object*, a *'v-shape' pattern that arises in the sequence of frames*, *angle of reflection*, and *interchange of momentum*. An explanation of how the system handles each of these conditions is given below.

The collision-detection mechanism of our system was designed in such a way that the larger the number of conditions that are satisfied, the greater is the likelihood that an actual collision occurred. Specifically, each of the conditions satisfied adds a contribution to a running total specified for each object involved in the collision. We call this contribution a *reward* or a *punishment*, depending on whether it is positive or negative, respectively. (The number can also be zero, in which case there is neither reward nor punishment). A reward increases the likelihood of a collision detection, while a punishment decreases it. However, it is worth mentioning that this is not the only influence. The object correspondence influences collision detections and conversely, collisions detections influence the object correspondence.

If, for example, most of the conditions are satisfied and one or two are not satisfied, the running total may well exceed a certain specified threshold. In that case, the system concludes that it has seen a collision between two objects. If the total is below the threshold, then the system concludes that no collision took place. The threshold value was chosen to be 0.5.

There was no fundamental reason behind the choice of the value 0.5. That value was simply chosen as an intuitive reference value. The idea was that if the collision detection algorithm uses four conditions, and if 1.0 is the maximum value that the running total can take, then each of the conditions can provide a maximum reward of 0.25 (assuming all conditions are equally weighted). Therefore, the threshold value 0.5 was chosen so that it is equally likely that a collision happened or not. More information about the threshold is included in the section about the conditions.

We close this section by suggesting the complexity of collision detection in the microworld of balls. If n balls can collide with m walls, then there are nm possible collisions. Hence, at each moment, nm wall–ball collision variables have to be updated. In a similar fashion, if n balls can collide with each other, then there are $n(n-1)/2$ relations to be observed (due to the symmetry of the ball–ball relation). Hence, at each moment, $n(n-1)/2$ ball–ball running totals have to be updated.

4.2. Our method of collision detection

Our solution to the collision-detection problem involves the idea of rather than giving *fixed* values to each of the conditions, we allow *variable* values. Let us clarify this with a concrete case. Perhaps the most intuitive condition for the perception of a collision is the presence of a *short-distance* observation. (We discuss this idea in greater depth in Section 4.5.) If two objects are far from each other, the running total involving this pair will receive a large negative value, which will inhibit the observation of a collision involving them. As before, we call this inhibition a 'punishment' to the collision relation of the two objects.

This punishment will fight against the effect of other positive values, if there were some conditions favouring the observation of a collision. For objects that are closer, the punishment is milder and milder, but for objects that are sufficiently close, there is no more punishment. If the objects get even closer, the opposite phenomenon occurs. That is, the collision relation of these two objects will receive a positive value (a 'reward') that increases the likelihood of the observation of a collision. However, no matter how close the objects are, the reward cannot be so large that other conditions can be ignored, because of course two objects can zip right by each other without colliding. This gives rise to the idea of a *reward/punishment function*, an example of which is illustrated in Figure 11. Many mathematical functions can model the biological principle just described as reward/punishment for the distance observation, all of them having a shape similar to the one illustrated in Figure 11. Further explanation, along with the actual function used in NINSUN, can be found in Section 4.5.

As was suggested in Section 4, the final decision for collision detection is a function of the sum of all the reward/punishment values. If this sum exceeds a certain threshold, then NINSUN decides it has seen a collision; otherwise, it sees no collision. The reward/punishment functions and the threshold constitute a delicate system that depends on weights, all of which must be adjusted to work properly. There is a fine equilibrium among all these weights and the threshold, and, in general, all of them are sensitive to small changes.

In order to realise this type of collision detection, our strategy was to start with the threshold set at 0.5 and then to design reward/punishment functions taking this threshold into account. The maximum value that the sum can take is a relatively small number (e.g. 2.5 for wall–ball collisions), suggesting an ideal observation of a collision. The sum does not have a lower bound, and can take negative values of any magnitude. The idea is that of strongly punishing absurd misperceptions, such as a collision of two objects when the objects are very far away from each other, or a collision with a wall where a ball bounces off at an angle that is extremely different from the angle of incidence. Also, for collision detection, it is desirable that the sum of the reward/punishment functions should stay as far as possible from the critical value. Figure 12 summarises these principles about the sum-threshold system.

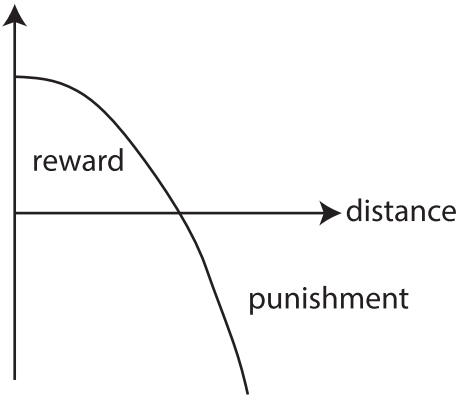


Figure 11. A model of a reward/punishment function based on the distance observed between two objects. The actual function used in the model is presented in Section 4.5.

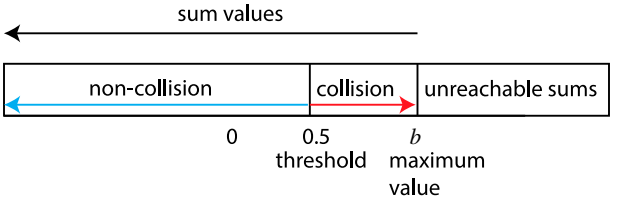


Figure 12. Scheme for the sum-and-threshold system for collision detection. The red arrow indicates reward values, and the blue arrow indicates punishment values. The value *b* is an upper bound for the sum, which is a small positive number (2.5 for wall–ball collisions).

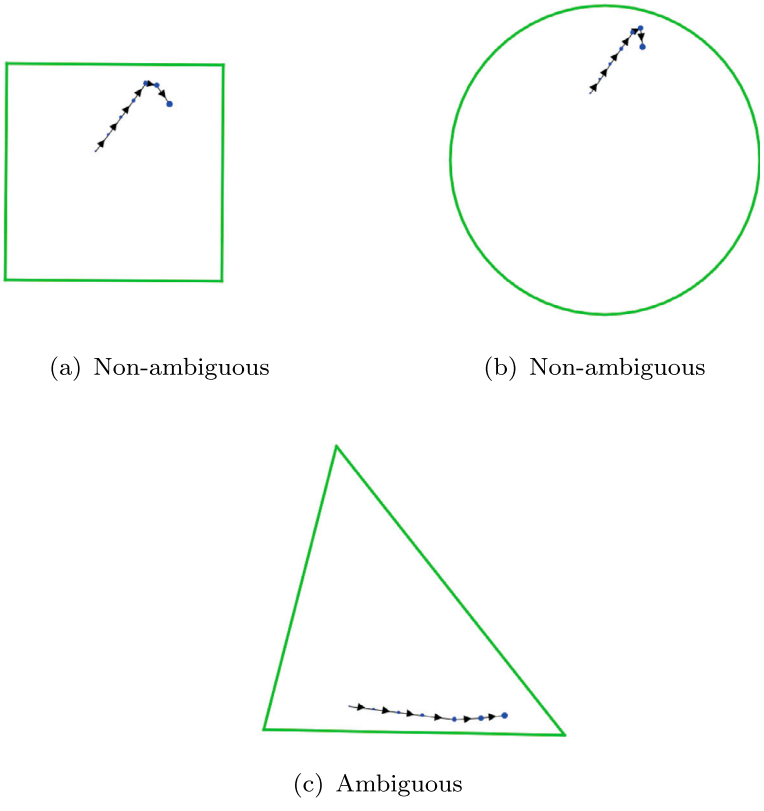
If the fine equilibrium of this system has not yet been reached (usually one of the weights causes excessive rewards for some condition), it is typical that when the system detects a collision, it detects yet another collision on the next frame update. This is due to the similar frame configuration shared by consecutive updates. If this happens, we say that the system is 'hyper-sensitive' to collisions. Similarly, when the fine equilibrium is broken (usually one of the weights causes excessive punishment for some condition), then the system often fails to detect a genuine collision. If this happens, we say that the system is 'hypo-sensitive' to collisions. It may even happen that a non-calibrated system is both hyper-sensitive and hypo-sensitive.

Homing in on a fine equilibrium state for the interpreter system requires a long and patient effort, as combinations of several parameters have to be tested. A fine equilibrium is reached when the system's collision detection is similar in sensitivity to that of a human, at least for unambiguous cases. A minimum test that the system needs to pass robustly is the reliable detection of collisions of a ball with a wall.

The next subsections are devoted to describing in greater detail the conditions that are used to detect collisions in the domain of balls and walls. All of these conditions have been implemented in the interpreter.

4.3. The V-shape

The *v-shape* condition applies to a single, usually small, moving object. If the trajectory of the moving object has this shape, the object may or may not have collided with something, but if it had a collision, there is a high probability that this shape will arise as part of its trajectory. A v-shape is likely to indicate a change of direction of a colliding object. Typical cases of simple v-shapes are illustrated in Figure 13.



(a) Non-ambiguous (b) Non-ambiguous

(c) Ambiguous

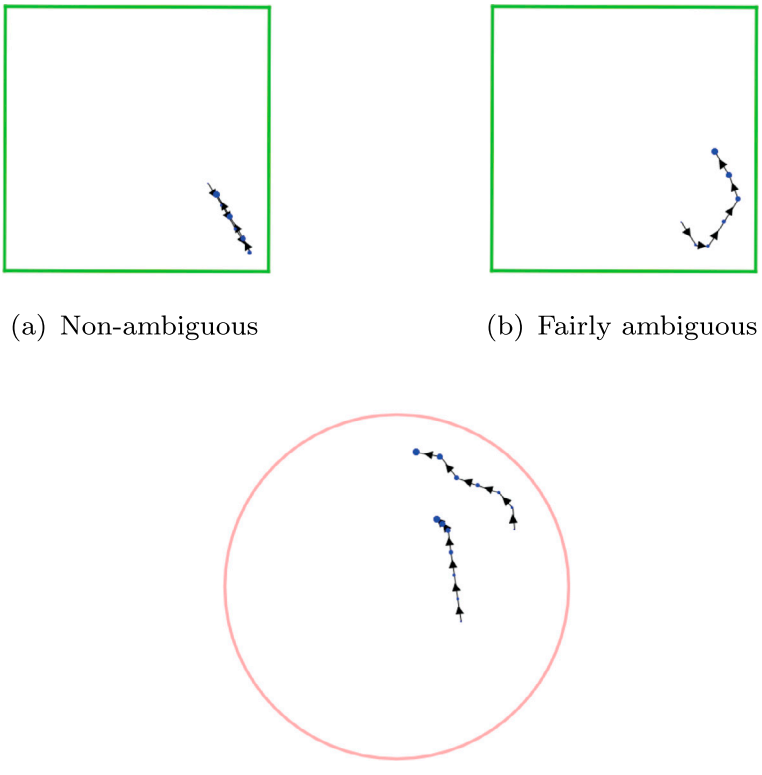
Figure 13. Typical v-shapes arising from the collision of a ball with a wall.

The cases in Figure 13(a) and (b) do not present ambiguity. (Notice that a v-shape arises independently of the shape of the wall.) However, the case in Figure 13(c), is ambiguous, since the 'v' angle is close to 180° and thus could easily be confused with a straight line.

Even though the term 'v-shape' suggests a specific visual form, some v-shapes are atypical in the sense that they may not look like a 'v.' Such atypical v-shapes may or may not be ambiguous in terms of indicating collisions. Some atypical v-shapes are illustrated in Figure 14. In Figure 14(a), the v-shape is not ambiguous (even though its interpretation could be ambiguous, since it is not clear how many collisions happened at the corner of the square). A situation like that in Figure 14(b) might produce an ambiguous v-shape if the two consecutive collisions against the horizontal and vertical walls happen too fast. One of the two v-shapes might be difficult for NINSUN to see. A situation like that in Figure 14(c) is hard to disambiguate, as it might not be clear how many times the ball close to the wall hit the wall after being hit by the other ball.

V-shapes produced when a ball hits a wall are multiplied by a different weight than those produced when a ball hits a ball. The reason for this is that when a collision occurs between balls, it is fairly frequent that at least one of the two resulting v-shapes is ambiguous. For example, this is the case if two balls are moving in almost the same direction, but the trailing ball is going faster than the leading one, and hits it from behind. In this case, neither of the balls makes a clear v-shape. In general, there are subtle differences between the psychological processes for detecting collisions with walls and those for detecting collisions with other balls.

In order to detect a v-shape, it is preferable to use as few frames as possible. This is because a ball can hit a wall and shortly thereafter a second wall (as in Figure 14(b)), or hit a wall and a ball almost



(a) Non-ambiguous (b) Fairly ambiguous

(c) Very ambiguous

Figure 14. Atypical v-shapes.

simultaneously, as in Figure 14(c). In cases like these, it is desirable that the v-shape be produced before the second bounce. If the bounces occur too close in time to each other, the observation of one of the v-shapes may fail to be noticed.

The current design uses five frames to determine whether an object's trajectory has a v-shape or not. The pattern that NINSUN detects is determined by the last five frames, labelled 1, 2, 3, 4 and 5 (1 being the most recent), with $P_1, \ldots, P_5$ being the object's positions in those frames. Specifically, for a typical v-shape (a non-degenerate case), the segments P_1P_2 and P_3P_4 do not lie in the same line, the angle $\angle P_2P_3P_4$ is a non-180-degree angle, and the angle $\angle P_3P_4P_5$ is a 180-degree angle. (See Figure 15(a).) For a degenerate case, such as when there is a collision perpendicular to the wall, the v-shape is determined taking into account the fact that the returning path of the ball is on the same line.

A v-shape seen in an object's trajectory also gives rise to a spatial point of interest – namely, the vertex of the 'v,' which is called the 'v-point,' and which is actually a small circle. A v-point is usually not one of the frame positions. It is a calculated or imagined point. If a ball hits a wall, the distance between the point of contact and the centre of the v-point would be the same as the radius of the ball. (See Figure 15(b).)

A v-point may be useful for certain computations, such as the determination of the distance between a ball at some instant and the ball's point of contact with a wall. For such purposes, the v-point gives a better estimate of this distance than an estimate using one of the frame positions. However, v-points are not always available and the system should not be too reliant on them, because

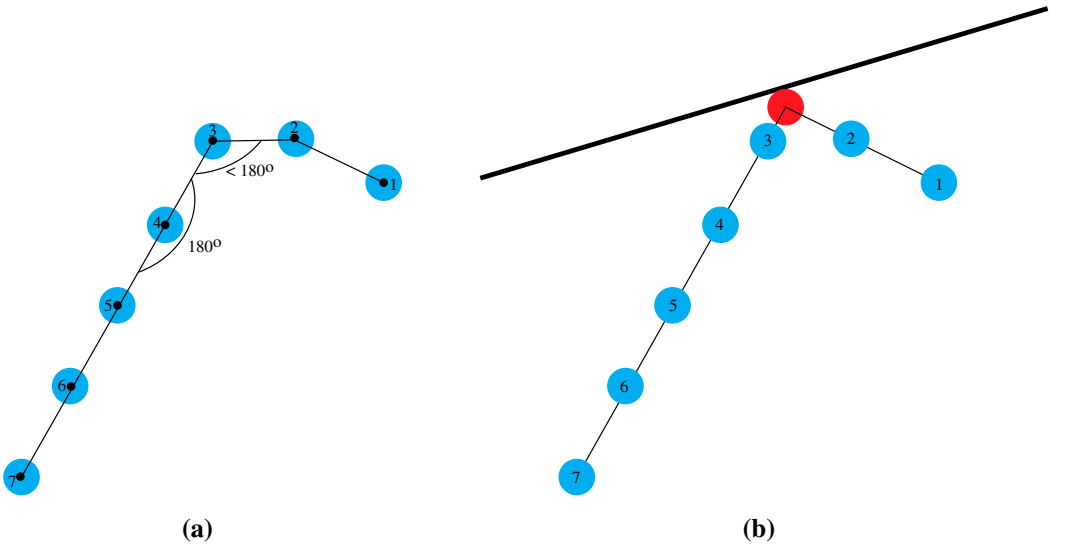


Figure 15. Patterns in the v-shape (Frame 1 is the latest). (a) $\angle P_2P_3P_4$ is a non-180-degree angle, and angle $\angle P_3P_4P_5$ is a 180-degree angle; (b) The v-point, represented in red, is imagined by the observer.

v-shapes are not always produced when collisions occur, and on occasion a v-shape will be 'seen' when in fact no change in direction occurred, which gives rise to the possibility of a false detection of a collision. Of course, v-points are most likely unavailable because of NINSUN's sampling rate, which makes it see the world in snapshots that in general do not include the vertex of the 'v.'

It is important to synchronise the observation of a v-shape with other observations, such as an abrupt change of velocity (see Section 4.4). They must be produced simultaneously. If this is not the case, the reward/punishment sum totals would be modified at different ticks of the clock, resulting in erroneous observation of a collision. Since the system currently uses seven frames, there are several v-shapes that can result from a collision. For non-degenerate cases, the current model creates a v-shape whenever angle $\angle P_2P_3P_4$ is less than 180° and angle $\angle P_3P_4P_5$ is a 180-degree angle (as was explained before). If another condition were observed at another instant, say for example, if angle $\angle P_2P_3P_4$ were less than 180°, then the criterion of simultaneity would not be met, and NINSUN would not see a v-shape, and as a result, no collision would be seen. Conversely, sometimes two collisions are detected when only one really took place.

In our model, if a v-shape is observed in the trajectory of one of the balls, a reward is added to the collision relations associated with that ball. If a v-shape is not observed, there is no punishment. This is because the production of a v-shape is an all-or-nothing phenomenon (unlike 'short distance,' which has a wide range of ambiguity). Also, the absence of punishment when no v-shape is produced gives a chance for those collisions that occur without a clear v-shape to be observed. For a wall–ball collision, the reward value is 0.5 (the same as the threshold). For a ball–ball collision, the reward value is 0.25 if a v-shape is observed for just one of the two balls, and 0.5 if v-shapes are observed for both balls.

4.4. *Abrupt change of velocity (acv)*

Like the v-shape condition, the *abrupt change of velocity* (*acv*) condition involves a single object. This means that NINSUN might observe a ball having *acv* without observing that it collided with a second ball or a wall. Since our model is stochastic, *acv* is occasionally observed when there is actually no change of velocity; and, for the same reason, sometime *acv* is not detected even though there was a change of velocity.

Given a sequence of ball positions $P_1, P_2, \ldots, P_n$, the ball's velocity vector v_i is given by the vector connecting two successive positions $\overrightarrow{P_{i+1}P_i}$ (its magnitude, $P_{i+1}P_i$, gives the ball's perceived speed). In order for a collision to be detected, the *acv* observation must be simultaneous with the v-shape observation. We compute the vector difference $\Delta v = v_1 - v_3$ (not $v_2 - v_3$) for the detection of a change of velocity.

Notice that for an accurate computation of the change of velocity, vector v_2 is less reliable than v_3, since v_2 is not parallel to either of the two lines of the 'v' shape, at the instant of observation of the v-shape. (See the positions of frames 2 and 3 in Figure 15(b).)

Intuitively, the perceived change of velocity Δv has to be a large vector if there is a collision. This statement can be formulated in terms of an inequality involving the average distance between two frames (to which, in this case, P_2P_1 gives a good approximation). Thus, the statement can be formulated by a condition of the form

$$||\Delta v|| = ||v_1 - v_3|| > \alpha \cdot P_2P_1, \quad \text{where } \alpha \text{ is a constant.}$$

However, in order to synchronise it with the v-shape, two more conditions are added. One is that the vector difference $v_3 - v_4$ should be close to zero and the other is that the vector difference $v_2 - v_3$ should *not* be close to zero. Once again, these conditions are expressed in terms of P_2P_1. They are:

$$||v_3 - v_4|| < \beta \cdot P_2P_1$$

and

$$||v_2 - v_3|| > \gamma \cdot P_2P_1$$

The constants α, β, and γ were determined by trial and error. Values close to 0.1, 0.3, and 0.8, respectively, work satisfactorily.

It is worth noticing that these computations require five frames. The computations above suggest that human beings must have a mechanism equivalent to that of storing several frames, and most likely at least five.

The reward/punishment values associated with *acv* are similar to those associated with v-shape. There is a reward when it is observed, and there is no punishment when it is not observed. For a wall–ball collision relation, the reward value is 0.5 (the same as the threshold). For a ball–ball collision relation, it is 0.25, even if both balls are observed as having undergone *acv*. Giving a reward value of 0.5 if both balls are observed as having undergone *acv* might seem reasonable, but it actually turns out that the system then becomes hyper-sensitive.

4.5. Proximity

If two objects collide, there is an instant at which the distance between them is zero. However, most likely NINSUN will miss this instant, as its input comes only from the few discrete frames that it can store, unless a frame were, by chance, created at the exact instant of contact of the two objects. Nonetheless, using the condition of *proximity*, NINSUN can 'cast a vote' in favour of or against a certain event as constituting a collision.

In this section, we are concerned with how close two separate objects are when at least one of them is moving, and how this influences NINSUN's perception of a collision. Let us assume that we have two balls that, after moving towards each other, collide, and then move away from each other. (See Figure 16.)

At any instant, NINSUN has seven frames, each of which stores the positions of the two balls. The distance between the objects is measured at each frame update. (Three of these distances are indicated by the three double-headed arrows, joining the two positions labelled '2,' those labelled '3,' and those labelled '6.') These distances are likely to influence the perception of a collision in situations

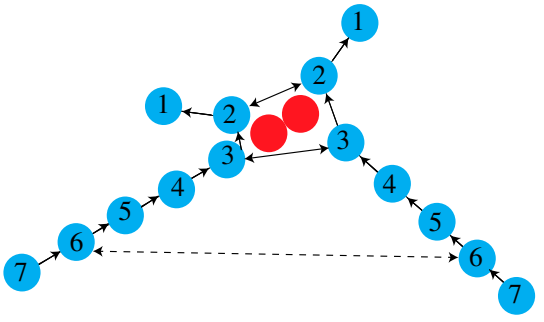


Figure 16. A diagram suggesting the proximity of two objects. The short distances in frame 2 and possibly frame 3 (solid double-arrowed lines) have a good chance of enhancing the perception of a collision, but not the distance in frame 6 (dashed double-arrowed line). The positions shown in red are not seen by NINSUN.

when the majority of other conditions are met. Additionally, when the distances are too large, these distances have an inhibitory effect on the perception of a collision, even if other conditions were met.

Given the fact that there is missing information between consecutive frames, most of the time the objects appear to have no contact, whether or not there was actual contact. Interestingly, sometimes we humans seem to perceive a zone or point of contact between the objects. However, the contact is most likely not directly observed but created in the brain. (See the red circles in Figure 16.) Other times, we do not have such a perception of contact, but we still judge that there was a collision based on the proximity of the objects. Therefore, ultimately, our perception of a collision is influenced by the proximity of the objects if their positions are very close for at least one instant.

The notion of proximity in NINSUN was designed to work in a way that resembles animal behaviour. In order to detect a collision, NINSUN tends to reward short distances and punishes non-short distances or long distances.

An analogy may help explain how proximity is intended to work in NINSUN. Consider a dog approaching a skunk starting from a very long distance. While the dog is sufficiently far away, the skunk behaves in its usual way (this is a zone of 'non-closeness' for the skunk). If the dog gets closer, the skunk becomes alert. (This is a zone of blurriness: not far but not close either.) If the dog gets even closer, the skunk prepares to spray by adopting a certain body position. At this point the skunk is in a tense state, and the tension might last while the dog approaches a little more. If the dog backs off, the skunk waits until the dog is sufficiently far away, and only then is its tension released, and it reverts to its usual shape. However, if the dog gets closer, the skunk will become tenser, making it highly likely to spray at any time. (At this point, the dog is in a zone of 'closeness' for the skunk.)

In our model, there is no sharp demarcation line between 'close' and 'far.' Instead, the reward/punishment function in NINSUN gets positive values (reward) for short distances, and negative values (punishment) for large distances.

The main difference between the mechanisms for observation of a v-shape (and also *acv*) and the mechanism for observation of distance is the nature of their reward/punishment functions. In the case of v-shapes and *acv*, a two-valued function is involved, whereas for proximity, a continuous function is used. Similar distances get similar rewards or punishments.

If the reward/punishment function for the observation of proximity were calculated using a two-valued function, it would give the same reward to any distance that was considered 'close.' (In the example of the skunk and dog, the skunk would spray only and always if the dog approached within a certain radius, which would result in a rigid system.) Animal behaviour is more flexible, in that it gives some chance to both prey and predator to avoid a fatal ending. The problem with treating proximity as a two-valued function is that a psychological sense of closeness depends on many factors, not just

precise distance. For example, a fast-approaching dog might create higher tension in the skunk than a slow-approaching dog at the same distance.

Keeping in mind this example, we now explain how proximity is implemented in NINSUN. Suppose, for simplicity, that an object is approaching a wall perpendicularly. (The same type of reasoning is valid for the more general situation of two objects approaching each other at random angles.) The judgement of proximity will depend on the Euclidean distance, but this number alone will not suffice. If the distance between positions in consecutive frames (which gives the speed of the moving object) is greater than the distance between the object and the wall (something that can happen when the object is moving fast), the object is perceived as very close to the wall. This is in agreement with the notion of proximity described in the dog–skunk situation, where a fast-approaching dog is more likely to raise the skunk's tension than a slow-approaching dog.

The proximity of two objects is used by NINSUN in deciding whether the objects collided or not. Since this judgement is based on a threshold (as was explained in Section 4.2), the distance needs to be normalised according to this threshold. Direct use of the Euclidean distance measured in units such as pixels is not sufficient.

The average distance between successive positions of the object plays the role of a basic unit of length that is used for the normalisation. The normalised distance between a moving object and a wall is defined to be the Euclidean distance between the wall and the object divided by the average distance between the object's successive positions. The normalised distance between two moving objects is computed similarly. It is defined to be the Euclidean distance between the objects divided by half the sum of the average distance between successive positions for each of the two objects. There are of course other sensible ways to define proximity, as long as they take into account not only the spatial separation of the objects but also how fast the objects are approaching each other.

Since the average distance between successive positions represents an average speed, proximity in NINSUN represents, in a certain sense, an approximation of how much time the object will need to reach the wall. This follows from the formula $speed = distance/time$. The validity of this approximation of the time to reach the wall is due to the fact that in a short amount of time, the velocity is approximately constant, especially in the domain of balls and walls.

Notice that this normalised distance of a ball from a wall depends on the ball's speed. The faster it is going, the greater the perceived proximity. Unlike the other two conditions for collision detection, which are unary relations, proximity is a binary relation.

An introductory idea for the observation of proximity was described at the start of Section 4.2. The reward/punishment function for proximity is modelled by a function with the shape illustrated in Figure 11. A function that behaves acceptably for wall–ball collisions is $f(x) = -x^2 + 1$, where x is the normalised Euclidean distance (or proximity) between the wall and the ball, and the constant 1 is twice the threshold. For ball–ball collisions, the function is $f(x) = -x^2 + 0.25$, where x is the normalised Euclidean distance (or proximity) between the balls. These functions have the advantage that if two objects are not close to each other, they strongly inhibit the detection of a collision, which, in some cases, would otherwise be very likely, thanks to the observation of other necessary conditions. (Our choice of these functions is of course not unique. Another function that was tried in the model and worked equally well was of exponential type.)

4.6. Equal angles

This is a condition for helping to decide whether there was a collision between a ball and a wall. It does not apply to collisions between balls. The main idea is that if the angle of incidence is the same as the angle of reflection (or if they are close), this favours the observation of a collision with the wall; and if the angles are not the same (or are not close), this discourages NINSUN from seeing a collision of the ball with the wall.

Since humans are sensitive to direction, a relatively small discrepancy between these two angles can affect the likelihood of perception of a collision of a ball with a wall. The larger the discrepancy

between the angles, the larger the likelihood that the event was not a collision of a ball with a wall. This condition is similar to the distance function, in that both involve continuous variables. In NINSUN, the angle function is modelled by a continuous quadratic function, similar to the function used for distance.

What about a situation where a ball hits a wall and moves away from the wall following exactly the same path, thus reversing its tracks? Such a situation does not involve equal angles of incidence and reflection, because the angle made by the returning pathway is not a 45-degree angle, but a 135-degree angle.

The function we chose is:

$$f(\theta) = 0.5 - \theta^2 ||P_1 - P_2||^2,$$

where θ is the difference between the angle of incidence and the angle of reflection, and P_1 and P_2 are consecutive frame positions. This function is used when there is a v-shape. When there is no v-shape, the value -2 is returned (which is enough to prevent the perception of a collision).

NINSUN uses the four most recent frames to estimate the angles of incidence and reflection. When there is a v-shape, NINSUN uses the v-point to determine the angles (recall that the v-point is most likely not in the four frames, but is 'imagined'). In the event that the v-point cannot be determined (e.g. when the first four frames make an angle close to 180°) NINSUN uses frame 3 as an approximation to the v-point.

If a ball is moving in a very narrow region surrounded by walls, then many bounces will occur in a short amount of time. In such a situation, it is difficult to detect the angles and the number of collisions. NINSUN cannot give reliable answers in cases like these because of the irregularity of the trajectory fragment.

We ran numerous simulations of collision detections when a ball bounced off a wall with expected and non-expected angles. (Simulations with the *wrong* angle of reflection were created with the help of the Tricycle command `blow`, which imitates the action of wind on a ball.) When the angle of incidence was too different from the angle of reflection, NINSUN did not recognise the event as a ball–wall collision. A video for this simulation can be found in the archives under the name 'Same Angle V1.' The file ready to run the simulation is `wrongAngle.txt`.

One would think that if most of the conditions for the observation of a collision are present, NINSUN should see a collision. However, this is not necessarily the case, since the conditions may not have been seen by NINSUN. An example of this is recorded in a video (see archives 'Same Angle V2') where there is a slight ambiguity as to whether the event involved a ball *rolling* along a circular wall or *colliding* with the wall at a very obtuse angle. In this video, NINSUN does not see any collisions.

4.7. Conservation of momentum

This condition is used only to detect ball–ball collisions. For detection of ball–wall collisions, we follow the 'same-angle' approach just described. This is an attempt at psychological realism. If two balls appear to interchange their momenta, this favours the observation of a collision. Psychologically, seeing a swap of momenta suggests that some part of the velocity vector of one ball was passed to the other ball, and vice versa.

The reward/punishment function for conservation of momentum is modelled by guessing or predicting the outcome of a collision, in a manner similar to how a billiards player tries to imagine the trajectories of the balls that are involved when the player is about to hit a ball with the cue. This idea is related to the ideas of dynamic mental representation (suggested in Freyd (1993)) and representational momentum (suggested in Perry, Smith, and Hockema (2008)). We check momentum conservation of two colliding balls by looking at the three last frames. Frames 2 and 3 are used to compute the velocities of the two colliding balls, and this information is then used to predict the velocity of the balls, based on the principle of momentum conservation. Once the new velocities have been guessed, they are matched against the actual velocities (as determined from frames 1 and 2). If the actual velocities are sufficiently close to the predicted velocities, NINSUN favours a collision vote.

If they are too far apart, NINSUN is discouraged from taking the event as a collision. In the former case, collision detection is rewarded by 0.25, whereas in the latter case, collision detection is punished by −0.25.

Why is it necessary to punish in such situations? Suppose two balls pass close to each other without colliding and both hit a wall at about the same time. Then both balls have good chances of having rewards for v-shape, short distance, and abrupt change of velocity. This might lead NINSUN to perceive a collision between the two balls. However, adding the possibility of punishment to the conservation-of-momentum condition prevents this from happening.

In general, when there is no collision, there is a high probability of getting punished by the conservation-of-momentum condition, and when there is a collision, there is a high probability of getting rewarded by the conservation-of-momentum condition. Therefore, one might well ask if this condition alone would be sufficient to detect collisions. The answer is no, although it is true that the condition alone takes care of many cases. An example of when conservation-of-momentum does not suffice is when two balls are moving parallel to each other with (nearly) identical velocity vectors. In this situation, it is the distance that allows NINSUN to recognise that the balls are not colliding.

One can also ask whether, if the conservation-of-momentum condition were only a punisher (i.e. never rewarding, even when there is an actual collision), the other conditions would be sufficient to identify a collision. The answer is once again negative. Since certain collisions are hard to detect (for example, some v-shapes are almost straight lines), and since the positions in frames are inexact, an incorrect collision detection has a non-zero probability when the conservation-of-momentum condition is ignored.

4.8. An overview of reward and punishment

In the preceding sections we gave descriptions of the mechanisms of reward and punishment provided by various conditions. However, decisions about collisions depend upon the result of adding the rewards and punishments coming from *all* the different conditions. This section is devoted to clarifying the reward/punishment mechanism as a whole. We will look at two examples.

Suppose a red ball and a blue ball are moving far from each other inside a container and both hit the wall simultaneously. NINSUN sees this event as in Figure 17(a). According to the reward/punishment assignments discussed above, their ball–ball collision sum totals are both likely to receive 0.5 for v-shape and 0.5 for *acv*, resulting in a sum total of 1.0 for each ball, which is the maximum score a ball can receive for collision detection. Should this situation be interpreted as a collision between the two balls? Not at all.

Luckily, the sum totals for *other* conditions keep this from happening. In fact, the distance condition's sum total has a large negative value which, when added to 1.0, still gives a negative number. If, in addition, the collisions with the wall occurred very near each other, then the distance condition might have been rewarded. However, this reward is likely to be a small positive value (less than 1). The momentum condition's sum total is also very likely to be a negative value, and that also will help to prevent a collision from being seen. Therefore, a collision is not likely to be observed when two balls by chance hit the wall simultaneously. It is worthwhile pointing out that the sum-total mechanism is effectively a form of abduction or reasoning from best explanation.

The second example involves a circular container that has a vertical wall passing through it. (See Figure 17(b).) Inside the container is a ball moving at constant speed, but an external agent (such as the wind) makes the ball change direction at a 45-degree angle when the ball is very close to the vertical wall. Therefore, the ball appears to collide with the wall but with an inappropriate angle of reflection. Does NINSUN see this event as a collision? It depends on the angle of incidence. If the angle of incidence is 45° (or close to it), the answer is yes. However, if the angle of incidence is very different from 45°, such as 90° (as is shown in Figure 17(b)), the answer is no.

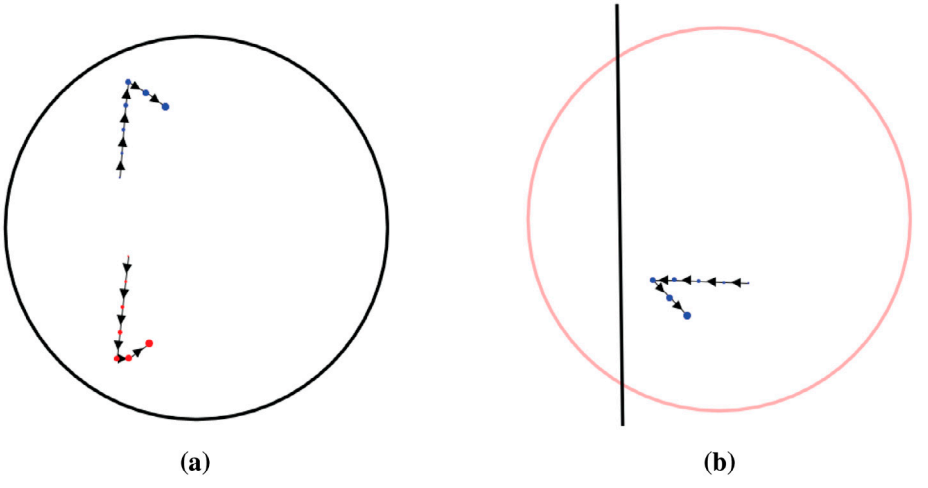


Figure 17. (a) A case that is not collision a ball–ball collision. (b) A case that is not a ball–wall collision.

The reason for this is the punishment given when the difference between the two angles is large. Because this difference is far from zero for the angles in the example, the punishment is a large negative number, which cancels out all the rewards coming from other conditions.

5. The ambiguity of collisions as opposed to pass-through-each-other events

In Tricycle, balls can be created so that they can either pass through each other (like two shadows on a wall) or collide with each other. Moreover, a 2-D world can be created as if two layers of balls existed. The balls in a single layer may or may not collide with each other, but balls in *different* layers do *not* collide. This creates various sets of possibilities that can affect NINSUN in unexpected ways, since it is not informed about of the existence of the two layers. This type of world resembles an everyday situation where some objects are occluded by others or appear to pass by each other. In the simulated world, a human will observe two types of events (collision events and pass-through events).

Curiously enough, when two balls coming from certain directions meet at certain speeds, humans tend to confuse these two types of events. Humans tend to switch quite randomly between seeing collisions and pass-throughs, independently of what actually happened. This was verified in our experiments with human subjects which will be described in another article. The choice of which type of event is perceived is a result of neural competition between two types of perceptual activity fighting for the observation of one relation or the other, similar to the internal fight involved in the perception of the Necker cube. Just as only one interpretation at a time is perceived in the Necker cube illusion, only one event is perceived when two balls moving with certain velocities coincide in space (at least as perceived by the eye) and then move away from each other. The following section examines this type of internal perceptual competition.

5.1. Why two balls appear to collide or pass through each other

Suppose two balls are approaching each other with equal speeds from the upper and lower left corners of a square towards its centre. (See Figure 18.) Assume we do not know if they will collide or pass through each other when they meet. The critical moments are just before and just after the spatial coincidence of the two balls. Suppose that in these two instants we have recorded the two positions (before and after, but not between) of the two balls. We will call them Frame 2 and Frame 1, respectively.

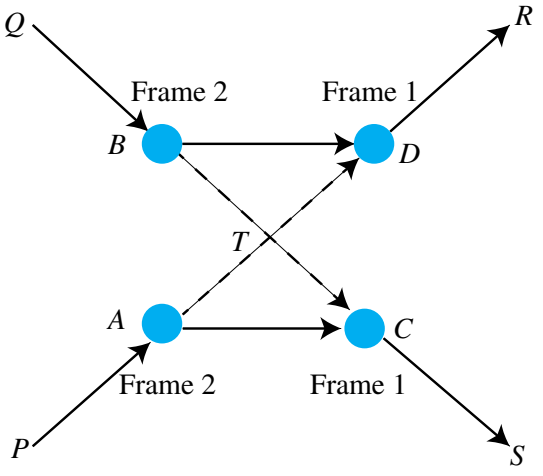


Figure 18. A diagram to explain the collision illusion. Circles *A* and *B* are in Frame 2, and circles *D* and *C* are in Frame 1. *A* and *B* represent the balls before they coincide in space, and *C* and *D* represent the balls after the coincidence. If the match given by the *solid* arrows is chosen, a collision will be perceived. If the match given by the *dashed* arrows is chosen, balls passing through each other will be perceived.

If NINSUN chooses the correspondence *AC* and *BD* from Frame 2 to Frame 1 (solid arrows), then the event has a very high likelihood of being seen as a collision. On the other hand, if NINSUN chooses the correspondences *AD* and *BC* (dashed arrows), then the event has a very high likelihood of being seen as a pass-through event. It all depends on how NINSUN winds up tracking the two balls – namely, either by selecting *AC* and *BD* or by selecting *AD* and *BC*. As we will see, context has a significant influence on which of these two ways of tracking the balls are more likely to be chosen.

In Figure 18, we see that if the event is perceived as a collision, the ball coming from point *P* is seen as following the v-shaped path *PATCS*, and the ball coming from point *Q* is seen as following the v-shaped path *QBTDR*, where *T* is imagined as the point of collision of the two balls. However, if the event is perceived as a pass-through, then the paths of the balls are seen as the straight lines *PATDR* and *QBTCS*. Between frames 2 and 1, there is critical missing information, and this missing information needs to be filled in by NINSUN to make sense of the situation and get rid of the ambiguity.

Two questions arise. (1) What would increase the likelihood of obtaining the missing information? (2) Are the collision and pass-through events the only two interpretations for a situation like that in Figure 18?

5.1.1. Increasing the probability of an illusion

Of course, sometimes the crucial information is available. Consider, for example, the situation of two objects that actually do pass through each other. In order for NINSUN to detect a pass-through event, the frames might provide crucial pieces of information about the moment of spatial coincidence. For instance, the objects might coincide in space for a relatively long time, thus giving NINSUN a chance to obtain key pieces of information directly from the frames. Ambiguity arises when such information is not available.

Certain conditions make the unavailability of the disambiguating information more likely. In the case of two objects passing through each other, if they move very fast or are very small, then chances are that the observer will not see the actual coincidence of the two objects. In this case, the event may well be perceived by NINSUN either as 'balls colliding' or as 'balls passing close to each other.'

We can use these facts to our advantage during certain simulations. We can temporarily modify the speeds of the objects to enhance the likelihood of interpretations that we know are more probable when the objects move at certain speeds.

5.1.2. More than just two types of perceived events

In the example of the two balls approaching each other (Figure 18), we mentioned only two interpretations, since Tricycle can only generate worlds in which two balls collide or pass through each other. However, it is conceivable that we might allow Tricycle to create other kinds of worlds featuring events that look like the one in Figure 18. For example, the objects might approach each other and suddenly change their trajectories without ever touching each other.

Although this project does not focus on such worlds, NINSUN can nonetheless perceive a two-ball event that is neither a collision nor a pass-through event. Moreover, there are times when even human interpreters are not sure of what happened, even though the event occurred right in front of their eyes. This means that we are left with a third possible interpretation: neither a collision nor a pass-through event.

Two short simulations of situations intended to show this third kind of interpretation are available. In them, two balls very close to each other move in the same direction and at the same speed in a single line. In the first simulation, after one ball hits the wall, the balls collide with each other. In the second simulation, after the ball hits the wall, the balls pass through each other. In both simulations, it looks to a human as if the balls are moving at an equal distance all the time and only one ball hits the wall. It is very difficult both for a human and for NINSUN to see this event as it actually happens. The simulations are shown in videos in the archives under the names 'Ambiguous Event V1/V2.' The world for the two simulations can be run from a Tricycle script named 'twoCloseSmallBalls.txt.'

5.2. The crucial role of the direction condition

When NINSUN is faced with a situation that might be either a pass-through event or collision, the neural competition described above is influenced by the *direction* condition. (See in Section 3.1.) This is because a telltale sign of a pass-through event is the lack of a change of direction, whereas most collisions involve a change of direction. In order to demonstrate this computationally, we set up a simple simulation.

Two balls were made to pass through each other at a 90° angle at the centre of a circular container, and they were given exactly the same speed, so that they periodically passed through each other at the centre after bouncing off of the circular wall. NINSUN was put in a mode in which it did not modify the world, and also the direction condition remained unchanged unless manually changed by a human. We explored various settings of the weight of the direction condition, in order to figure out what values would make NINSUN observe more pass-through events than collisions, and vice versa. A video of this simulation can be found in the archives under the link 'Direction Parameter V1.' In Section 5.4, we describe other simulations, in which we let the direction condition calibrate itself.

When the direction condition's weight is manually set to a high value, NINSUN tends to see pass-through events. This happens because when NINSUN relies heavily on this condition, it will tend to interpret ambiguous events as *not* involving a direction change, which means it will tend towards the pass-through interpretation. Conversely, when the direction condition's weight is manually set to a low value, changes in direction are of little relevance, and NINSUN relies mostly on other conditions. As a result, ambiguous events are more likely to be seen as collisions.

5.3. The influence of speed

Suppose a human is observing a Tricycle simulation of the two identical balls described in Section 5.2. Our experiments with human subjects show that if the balls are going at a medium speed, the human interpretation might vacillate between a collision and a pass-through event. However, if the speeds of the balls are reduced, humans will tend to favour pass-through-each-other events, and if the speeds are *very* low, the ambiguity vanishes totally, and humans will see only pass-through events. (Recall that the simulation is set up so that the balls are actually passing through each other.) Conversely, if the speeds of the balls are gradually increased, humans will start seeing collisions once again.

We now let NINSUN observe the same simulation. Suppose that NINSUN's weight of the direction condition has been set to an intermediate value. Interestingly, NINSUN acts similarly to a human. If we start out with the balls moving at a medium speed, NINSUN will vacillate between the two interpretations. If we gradually slow down both balls, NINSUN will start seeing more pass-through events than collisions. And conversely, when the speeds are gradually increased, NINSUN starts detecting mostly collisions. A Tricycle script to run this experiment is `testWithSquare2.txt`. A video of this simulation can be found in the archives under the link 'Direction Parameter V2.'

The simulation in this video has essentially four stages.

(1) We start with high speeds (0.8) and a relatively low setting of the weight of the direction condition (0.1). As expected, we see mostly collisions.
(2) We reduce the speeds of the balls to 0.1, keeping the weight of the direction condition constant. NINSUN does not see collisions here. (This was a surprise because we expected that a low value of the weight of the direction condition would bring about the perception of mostly collisions.)
(3) We raise the weight of the direction condition to 0.9 (a high value). Now NINSUN sees no more collisions.
(4) Finally, with the weight of the direction condition still at 0.9, we increase the speeds of the balls once again. Now NINSUN sees only pass-through events.

In addition to being an interesting cognitive phenomenon, reducing the speed to disambiguate ambiguous events can also be used as a strategy in certain simulations to counter the lingering effect of an earlier context. By reducing the speed of the balls, the effects of the old context are (at least partially) diminished, and if we then rapidly increase the speed from slow to fast, NINSUN will start seeing the desired effect more quickly.

5.4. Disambiguating by context

Context has a strong influence on human perception. The Ebbinghaus illusion (the perception of the size of a central circle surrounded by circles is influenced by the size of the surrounded circles), the Ponzo illusion (the perception of the size of an object is influenced by its background), and the moon illusion (the moon appears bigger when on the horizon than in other positions) illustrate the influence of context on the perception of size. Context is also a crucial aid in disambiguation. For example, a listener might hear either *I like the mall* or *I like them all* depending on the context. In a similar manner, context influences the perception of balls as colliding or passing through each other.

If NINSUN has seen m collisions and n pass-through-each-other events in a short interval of time, then the chance that a new event will be perceived as a collision or as a pass-through will depend on the values of m and n. The larger the number of collisions, the more likely the perception of a collision, and the larger the number of pass-through events, the more likely the perception of a pass-through event.

Figure 19 illustrates a simulation given to NINSUN (a video of this simulation can be found under 'Direction Parameter V3'). The two pictures represent different stages of the simulation. The red balls are set up to bounce simultaneously off of the circular wall, after which they pass through each other at a 90-degree angle at the centre of the container. This pattern repeats over and over again, like clockwork. At the same time, during the first part of the simulation, the blue balls are made to collide with each other. The blue balls are made larger in order to raise the probability of a collision with each other, but their larger size plays no role in object-tracking, since the weight of the size condition of the neural network is set to zero. Also, the weight of the colour condition is set to zero, so colour plays no role in NINSUN's tracking of the objects.

Initially, all balls are given the same medium speed. Under these circumstances, the blue balls undergo a large number of collisions, and their distribution looks random. Later on during the simulation, the blue balls are made to pass through each other, as shown in Figure 19(b). As a consequence, the number of collisions of blue balls that NINSUN 'sees' per minute is lower than

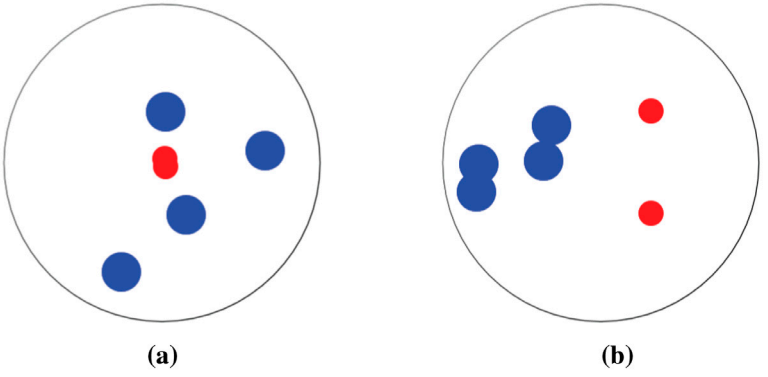


Figure 19. A set-up to manipulate the perceived frequency of collisions among blue balls from many collisions (a) to few collisions (b). In both cases the red balls are made to pass through each other at the centre of the circular container. In (a), the blue balls are made to collide with each other. In (b), the blue balls are made to pass through each other. In both cases the blue balls move in an apparently random manner.

in the first stage. Also, the number of perceived collisions per minute of *red* balls is lower than in the first stage. This is quite remarkable, since the properties of the red balls were not changed at all.

This simulation can be run by opening the script `directionParTest2.txt` and then letting NINSUN observe the world with the commands

```
autoModifyWorld false;
activateDirectionController true.
```

The first command ensures that NINSUN makes no changes to the world. The second command ensures that the weight of the direction condition is no longer controlled by a human, but is self-adjusted by NINSUN as a function of the context. (See Section 5.6 for more detail on context and self-adjustment.) By clicking on the network button, one can see the direction condition's weight changing often in value, as the numbers of perceived collisions and pass-through events change. (The direction condition's weight can also be seen in the text area on the interpreter side.) In the second stage, the world is manually set to disallow bouncing among the blue balls by executing the Tricycle command `setBallBouncing false`. (NINSUN does not know of this change.) The weight of the direction condition will then slowly drift upwards.

This simulation was designed in such a way that there would be a larger number of collisions among blue balls in the first stage than in the second stage. This was done by making the blue balls larger in size and by putting them in a different layer from the red balls. The blue balls thus do not interact with the red balls. This, of course, yields a large number of pass-through-each-other events in the first stage, since the red balls and blue balls pass through each other. However, in the first stage, even more collisions among the blue balls are perceived than in the second stage.

The simulation showed that there is a tendency for NINSUN to see the red balls as colliding with each other in the first stage. In the second stage, when the blue balls are made to pass through each other, the situation is dominated by pass-through events, and NINSUN's tendency is to see the red balls *also* as passing through each other. There are moments in this simulation where too many balls were close to the centre, resulting in many overlaps. This seemed to confuse the object-tracking process, and as a result, the red balls were occasionally seen as colliding with each other or even with the blue balls, even though no such event ever actually occurred.

This surprising result inspired us to explore other variations on the original experiment. One possibility was to invert the actual event, making the red balls collide with each other instead of passing through each other. Also, since colour plays no role, we made all balls the same colour (green).

Instead of using a 90-degree angle, we used a 180-degree angle, with the balls moving horizontally back and forth on the diameter of the circular container. We were interested in seeing how humans and NINSUN would perceive this event when the speeds of the balls were manipulated.

It turns out once again that the situation is perceived to consist mostly of collisions or of pass-through-each-other events, depending on the speed of the balls. In this variation, the balls that move along the diameter of the circle are made to collide with each other at the centre, while the others all pass through each other. The balls on the diameter are made to move very slowly for several seconds, and NINSUN sees them as passing through each other, under the influence of the many pass-through events generated by the other balls. Then the balls on the diameter are sped up. After a few seconds, NINSUN's perception starts to change. It starts seeing occasional collisions, and then the number of collisions increases very strongly. This rapid rise in the frequency of perceived collisions affects the weight of the direction condition, and soon NINSUN sees the balls on the diameter as colliding.

Notice that the high speed of the balls occasionally affects object-tracking. The fact that all balls are identical makes the identification of them rather challenging for NINSUN (and for a human, too). (A Tricycle script of this variation is found in the file `twoSmallBalls4.txt`. A video can be found in the archives under the link 'Direction Parameter V4.')

5.5. Other influences

If we greatly increase the size of the balls in the experiment described in Section 5.2, chances are that pass-through events will be observed if they are actually passing through each other. This is because in Figure 18, the radii of the balls can be greater than the distance between the positions in frame 2 and frame 1 (of a single ball), and NINSUN (and, of course, a human as well) can then detect the overlap of the two balls. In general, the size of the balls affects the likelihood of seeing a pass-through event or a collision. However, for the sake of simplicity, our research was oriented more towards the study of smaller balls, and since the balls are in general very small, their size can essentially be neglected.

We can also observe the effects of manipulating the size of the container. If we start with a large container with balls moving at a medium-low speed and actually passing through each other, we are likely to see balls passing through each other. If we then gradually decrease the size of the container, the frequency of collisions with the wall will increase. When the size of the balls is very small, the number of collisions with the wall can be so high that people might well perceive ball–ball collisions instead of pass-through events. (This hypothesis has not been tested with human subjects.) In any case, the frequency of perceived collisions with the wall may have an effect on the perception of the current situation. However, the influence of ball–ball collisions seems to be smaller than the influence of ball–wall collisions. Although they are very provocative, these kinds of perceptual effects, due largely to having a very small container, are not a major focus of our research.

5.6. Feedback from collision detection to object correspondence

This section concerns how context affects the likelihood of perception of a collision or pass-through event. We begin with an analogy to everyday life. Consider people's subjective estimates of the danger of certain environments. For example, most people will tend to imagine a church and a dark alley as presenting very different levels of danger. What matters would seem to be *how many* unpleasant experiences (e.g. assaults) have happened before in the given environments. However, what is even more important than the number is the relative *frequency* of unpleasant experiences in a given environment. Similar ideas hold for the detection of relations in the world of balls and walls. The number of perceived relations of one particular type should be compared to the number of relations of other types. This comparison yields a relative frequency.

One way to define this relative frequency is as follows:

$$r = \mathrm{ptpm} / (\,\mathrm{ptpm} + \mathrm{colpm}\,),$$

where 'ptpm' is the number of pass-through events per minute and 'colpm' is the number of collisions per minute. The denominator is of course the total number of events.

A slightly different approach to defining r involves the psychological impact that a collision has compared to that of a pass-through event. This can be done by attaching weights to the two event numbers. In this case r would be defined as follows:

$$r = \text{ptpm} / (\text{ptpm} + w\,\text{colpm}), \tag{1}$$

where w is a weight that we can assign to a collision relative to a pass-through event. NINSUN appeared to work best in the simulations described above by making $w = 10$. (This means that NINSUN gives a collision 10 times the importance that it gives to a pass-through event.)

We can try to justify the insertion of the factor w by returning to our analogy of danger assignment to various places. We could more accurately model a person's sense of a place's danger if we had a measure of the *emotional costs* of the experiences in various places (e.g. in a church or a dark alley). A single bad experience in a dark alley might far outweigh many tranquil experiences in a church.

The factor w can also be understood in a different way. Humans are more sensitive to an experience when it is recent. For instance, when a frightening event has just happened, people get scared and may well take precautions. After some time, however, when the event has receded sufficiently far into the past, most people gradually lose their fears and no longer take precautions. The hypersensitivity tends to fade.

In the case of people watching Tricycle simulations of balls moving inside a container, whenever there is a perception of a collision, this perception seems to stay in short-term memory for a short amount of time (say a couple of seconds), a little like the well-known auditory-reverberation phenomenon known as the 'phonological loop.' If another event (collision or pass-through) occurs during this brief period, it will most likely also be perceived as a collision. What happens in our model in such a situation? NINSUN keeps track of the number of events of each type in the last 5 s; after that, these numbers are simply merged with older statistics. All this information is used by NINSUN to build up a global sense for certain aspects of the simulation. By adding the factor w, we are simulating the psychological effect of a recent collision (meaning less than 5 s old).

Additionally, as was mentioned in Section 3, the object-tracking process contains several parameters that reflect the degree of importance or trust that NINSUN attaches to certain events. We earlier showed that one parameter that is critical in the detection of collisions is the weight associated with the direction condition. This is because at the instant of a collision there is usually an abrupt change in direction of motion (while the amount of proximity, for example, does not change much). The relative frequency r described above in equation 1 is a crucial factor in determining the importance of the weight of the direction condition. NINSUN updates the weight of its direction condition x (see Section 3.1) by making the assignment

$$x = 1 - r.$$

This assignment is carried out anew every 5 s, in parallel with other NINSUN's processes. Notice that r is a probability value. Therefore, the weight of the direction condition is related to this probability.

If the number of collisions is large compared to the number of pass-through events, then x will be small (close to zero). This makes NINSUN pay little attention to the direction condition, and it will thus tend to see more collisions. Conversely, if x is large, which happens when the number of pass-through events is large compared to the number of collisions, then NINSUN has a higher propensity to see pass-through events. In this fashion, the perception of events is biased so that ambiguous events will tend to be interpreted in a manner consistent with how previous recent events have been interpreted.

We close this section with a simulation featuring a world that consists of six balls moving in three horizontal lines inside a square container, with one pair of balls in each line. They all move with medium speed, and they all bounce back and forth between the container's left and right walls, as is shown in Figure 20.

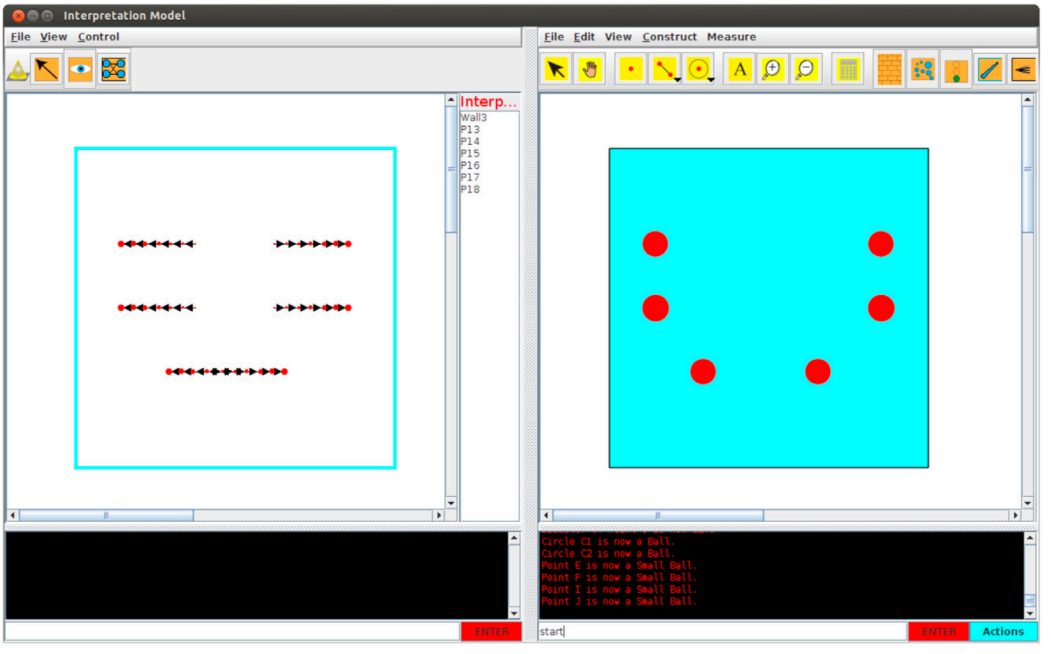


Figure 20. Three pairs of balls.

At first, all pairs of balls are made to pass through each other, and NINSUN sees pass-through events because of the large influence of this type of event. Then we change the top and bottom pairs to a colliding mode. Interestingly, just after the change has taken place, NINSUN still sees only pass-through events at first, as it is still under the influence of the dominant type of event in the first stage. It takes some seconds to before it starts to see collisions. Then the perception of collisions slowly rises for a few seconds. At the same time, the weight of the direction condition starts to fall. We also increase the speed of the balls to accelerate the process (it is very slow otherwise) and to increase the chance of collision perception. After some time, the upper and lower pairs are seen mainly as colliding, and the central pair of balls *also* starts to be seen as undergoing some collisions. Eventually, when the weight of the direction condition goes down even more, the dominant type of perceived event is just collisions. (A video is found under 'Direction Parameter V5,' and a Tricycle script that contains the world set-up for this simulation is found in the file 'twoBalls6.txt.')

6. Conclusion

We have created a novel model of scientific discovery that takes into account human perceptual activity. Our system's simulation of scientific discovery follows the pathway of perception, starting with the visualisation of the world, when certain objects catch the eye, to object tracking, to identification of relations (such as collisions), to making modifications of the world, to memorisation of events, to compression of information, to detection of patterns among the objects, all the way up to the conjecture of a mathematical law. Our simulation of these different stages of brain activity is based on selected cognitive theories of perception and scientific discovery. This paper concerns only the first four stages, leaving the remaining ones to be presented in another paper.

The world we live in is uncertain most of the time and our perception of it is context based. As long as science has existed, scientists have had to forge pathways of discovery under this constraint. And yet, they have been able to find patterns and make discoveries. Human perceptual activity is

simulated by the part of our computer model that we call NINSUN, which is able to interpret our very simple simulated world in a context-depending manner. The world is simulated in a second computer program that we call Tricycle. We deal with uncertainty and context by focusing mainly on the microworld of walls and balls, where balls move and collide with the walls and with each other (or pass through each other) in a way that has a high degree of randomness to a human perceiver.

Acknowledgements

The majority of this research was carried out at Indiana University, mostly in the Percepts and Concepts Laboratory of the Department of Psychological and Brain Sciences, but also partly at the Center for Research on Concepts and Cognition (CRCC). The most recent phases of the project were carried out at the Escuela Politécnica Nacional in Quito, Ecuador. Roughly 70% of the funding for this project came from National Science Foundation REESE grant 0910218, 20% from CRCC, and the remaining 10% from the Escuela Politécnica Nacional.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Science Foundation REESE [grant number 0910218]; CRCC; Escuela Politécnica Nacional.

References

Bridewell, W., Sánchez, J., Langley, P., & Billman, D. (2006). An Interactive environment for the modeling and discovery of scientific knowledge. *International Journal of Human-Computer Studies, 64*, 1099–1114.

Dawson, M. (1991). The how and why of what went where in apparent motion: Modeling solutions to the motion correspondence problem. *Psychological Review, 98*, 569–603.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence, 24*, 85–168.

Freyd, J. (1993). Five hunches about perceptual processes and dynamic representations. In D. Meyer & S. Kornblum (Eds.), *Attention and performance XIV: Synergies in experimental psychology, artificial intelligence, and cognitive neuroscience* (pp. 99–119). Cambridge: MIT Press.

Georgopoulos, A. (1988). Neural integration of movement: Role of motor cortex in reaching. *The FASEB Journal, 2*, 2849–2857.

Gross, C. (1998). Claude Bernard and the constancy of the internal environment. *The Neuroscientist, 4*, 380–385.

Klahr, D., & Dunbar, K. (1987). *Dual space search during scientific reasoning* (Tech. Rep. No. AIP-13). Pittsburgh, PA: Carnegie-Mellon University.

Klahr, D., & Simon, H. (1999). Studies of scientific discovery: Complementary approaches and convergent findings. *Psychological Bulletin, 125*, 524–543.

Langley, P., Simon, H., Bradshaw, G., & Zytkow, M. (1987). *Scientific discovery. Computational explorations of the creative processes*. Cambridge: The MIT Press.

Lara-Dammer, F. (2009). *Modeling human discoverativity in geometry* (Unpublished doctoral dissertation). Indiana University.

Lenat, D. (1979). On automated scientific theory formation. A case study using the AM program. In J. Hayes, D. Michie, & L. I. Mikulich (Eds.), *Machine intelligence* (pp. 251–283). Chichester: Hellis Horwood

Mahabal, A. (2009). *Seqsee: A concept-centered architecture for sequence perception* (Unpublished doctoral dissertation). Indiana University.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco: W.H. Freeman and Company.

Michotte, A. (1963). *The perception of causality*. London: Methuen.

Milan, A., Roth, S., & Schindler, K. (2016). Multi-target tracking by discrete-continuous energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*, 2054–2068.

Miller, A. (1987). *Imagery in scientific thought*. Cambridge: MIT Press.

Mitchell, M. (1990). *Copycat: A computer model of high-level perception and conceptual slippage in analogy-making* Unpublished doctoral dissertation. Universityof Michigan.

Nayak, N., Zhu, Y., & Roth, S. (2015). Hierarchical graphical models for simultaneous tracking and recognition in wide-area scenes. *IEEE Transactions on Image Processing, 24*, 2025–2036.

Perry, L., Smith, L., & Hockema, S. (2008). Representational momentum and children's sensori-motor representations of objects. *Developmental Science, 11*(3), F17–F23.
Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science, 324*, 81–85.
Ullman, S. (1979). *The interpretation of visual motion*. Cambridge: MIT Press.
Wertheimer, M. (1945). *Productive thinking*. Chicago: The University of Chicago Press.