Evol. Intel. (2008) 1:145–157
DOI 10.1007/s12065-008-0011-y

RESEARCH PAPER

# Frequency analysis for dendritic cell population tuning

Robert Oates · Graham Kendall · Jonathan M. Garibaldi

Received: 11 December 2007 / Revised: 11 March 2008 / Accepted: 13 March 2008 / Published online: 15 April 2008

**Abstract** The dendritic cell algorithm (DCA) has been applied successfully to a diverse range of applications. These applications are related by the inherent uncertainty associated with sensing the application environment. The DCA has performed well using unfiltered signals from each environment as inputs. In this paper we demonstrate that the DCA has an emergent filtering mechanism caused by the manner in which the cell accumulates its internal variables. Furthermore we demonstrate a relationship between the migration threshold of the cells and the transfer function of the algorithm. A tuning methodology is proposed and a robotic application published previously is revisited using the new tuning technique.

**Keywords** Dendritic cell · Robotics

## 1 Introduction

The field of 'artificial immune systems' (AISs) is an area of computer science concerning the development and use of algorithms based on the mechanisms underlying biological immune systems. The immune system stands out as a potential source of inspiration for problem-solving algorithms as the information processing is inherently distributed (as with all cell-based systems) and the input data is noisy. A review of common algorithms within the field can be found in [5, 10]. AIS algorithms are diverse and have contributed to the fields of optimisation [2], classification [22] and anomaly detection [20]. The dendritic cell algorithm (DCA) is an emerging AIS algorithm based on biological dendritic cells (DCs). DCs form part of the innate immune system. The key responsibilities of a DC are the identification of threats to the body and play a part in the activation of an immune response to those threats. The authors of this paper view the DCA as a decision making algorithm, using a population of non-homogeneous agents to vote on a binary choice. This has applications within both the fields of heuristic-based classification [15] and anomaly detection [9].

Analysis of the DCA has shown that appropriate selection of the migration thresholds is crucial to the performance of the algorithm [6]. Currently no techniques exist for tuning the DCAs migration thresholds. Generally experiments are repeated using a wide selection of migration thresholds to identify one distribution which outperforms the others. If a migration threshold is too low, a cell will migrate too quickly and will not be able to gather a representative sample of the input signals. If a migration threshold is too high, the cell will migrate too slowly and will misclassify the gathered antigen. A balance between these two extremes is found in part by the population-based nature of the algorithm. However, this is still dependant on a suitable selection of migration thresholds within the population.

Despite an increasing amount of research being carried out on the DCA, its behaviour within the time and frequency domain are not well understood. Whilst it is apparent to its users that it behaves differently when exposed to signals of differing frequency, there is no

R. Oates (✉) · G. Kendall · J. M. Garibaldi
School of Computer Science, University of Nottingham,
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
e-mail: rxo@cs.nott.ac.uk

G. Kendall
e-mail: gxk@cs.nott.ac.uk

J. M. Garibaldi
e-mail: jmg@cs.nott.ac.uk

formal understanding of the relationship between the algorithm's input parameters and its behaviour in the frequency domain. This paper identifies a mathematical model of a single DC which can be used to identify the frequency response of the system as a function of the input parameters to the DCA. This information allows us to estimate the response of the algorithm to its input signals and make informed choices about the migration threshold.

This paper is organized as follows. Section 2 explains the workings of the DCA. Section 3 goes on to derive an optimised version of the DCA and demonstrates equivalence to known filters with characterised transfer functions. In Sect. 4 a tuning methodology is suggested based on the frequency response of the DCA and in Sect. 5 the robotic application from [15] is revisited. The latest results from the updated robotic architecture are presented and a range of migration thresholds are compared with the threshold suggested by the tuning methodology. Finally, the ramifications of these results are discussed in Sect. 6.

## 2 An overview of the dendritic cell algorithm

### 2.1 The biological dendritic cell

The authors of this paper are not biologists so we refer the interested reader to [6, 9] for a more formal treatment of the biological inspiration of the dendritic cell algorithm. The outline provided here is a summary of the relevant information within [6, 9]. For general information about biological dendritic cells see [13].

The functionality of the immune system that the DCA attempts to emulate is one of the underlying mechanisms for the body's ability to recognise threats such as viruses and bacteria. There are differing explanations within the biological community on how these mechanisms are realised. The DCA is based on one such paradigm known as danger theory. This model, first put forward in [14], is a framework underpinned by the concept that the immune system does not simply respond to antigen because it is identified as foreign or "non-self", rather it responds to evidence of "danger". For the purposes of the DCA, danger can be viewed simply as unexpected cell damage. The information presented here will be from the perspective of the danger theory model.

The first stage of a dendritic cell's life that is relevant to the DCA is termed the 'immature' phase. In this phase dendritic cells are found throughout the body's tissue and within the blood stream. Here they ingest antigen and absorb molecular patterns and signals. The molecular patterns that the cells absorb are 'pathogen-associated molecular patterns', or PAMPs. These molecular patterns are released into the tissue by pathogens (viruses, bacteria etc.), so their presence is indicative that the body has been invaded by a potentially harmful organism. In the danger theory model, the signals absorbed by the dendritic cells are termed 'danger' and 'safe'. Danger signals are evidence of unexpected cell damage. When cells die naturally (a process termed 'apoptosis'), the contents of the cell are destroyed internally before the cell walls break down. If cells die due to damage or infection (termed 'necrosis') the contents of the cell are released into the tissue. These chemical markers are danger signals. Safe signals are chemicals released by cells during apoptosis and can be viewed as evidence of 'normality' within the system.

When dendritic cells have been sufficiently exposed to PAMPs, danger or safe signals, they migrate to the lymph nodes of the body. There the cells become either 'mature' or 'semi-mature' cells. A cell will become a mature cell if it has been predominantly exposed to PAMPs or danger signals. Mature cells present their antigen to T-cells within the lymph node in conjunction with a compound called 'interleukin 12' (IL-12). This stimulates the T-cell and causes it to effect an immune response. A cell will become a semi-mature cell if it is predominantly exposed to safe signals. Semi-mature cells present their antigen to T-cells within the lymph node in conjunction with a compound call 'interleukin 10', (IL-10). This suppresses the T-cell and prevents an immune response being launched against the presented antigen.

This capacity to either stimulate or suppress cells that provide an immune response within an environment that contains noisy and potentially conflicting data is why dendritic cells are viewed as a source for decision making algorithms.

### 2.2 The dendritic cell algorithm

In this section we present the DCA. The aim is to provide an engineering-centred model of the algorithm and to remove the biological terms from its presentation where possible. For a full description of the original algorithm please refer to [7].

The DCA accepts four streams of data as input, three time-varying signals and an application-specific list of symbols. The time-varying signals are sourced from application-specific heuristics and are termed "PAMP", "Danger" and "Safe".

- The PAMP heuristic provides a signal which increases proportionally to the presence of data with a strong correlation to a positive or 'anomalous' situation.
- The Danger heuristic provides a signal which increases proportionally to the presence of data with a weaker correlation to a positive or 'anomalous' situation. A weaker correlation typically means that all 'anomalous'

or positive situations cause danger signals, but it is possible for some 'normal' or negative situations to also cause the same effect.

- The Safe heuristic provides a signal which increases proportionally to the presence of data with a strong correlation to a negative or 'normal' situation.
- The list of application-specific enumerations, termed "antigen", acts as a cyclic buffer, storing the symbols which describe the current environment for the algorithm.

An artificial DC uses these input signals to produce three internal signals, termed '*CSM*', '*IL*-10' and '*IL*-12' after their biological counterparts. Each of these signals is the summation over time of a weighted sum of the input signals (see Fig. 1). The weights used are typically the same weights suggested in [7] and are listed in Table 1.

*CSM* accumulates throughout the cell's lifetime and rises proportionally to the cell's exposure to any input signals. PAMP and Safe signals typically have a greater impact on the accumulation of *CSM* than Danger. *IL*-10 is a cumulative value that rises proportionally to the cell's exposure to the Safe signal. *IL*-12 is a cumulative value that rises proportionally to the cell's exposure to PAMP and Danger, but can be decreased by exposing the cell to Safe signals. Each time a cell polls the environment to update its internal variables it also removes 'antigen' from the input buffer and stores these symbols locally.

At the end of each iteration of the algorithm the accumulated *CSM* value of each cell is compared to its migration threshold. For each cell, this threshold is set to a random value using an application-specific probability distribution. Typically this is a uniform distribution around a specified mid-point $M_i$ from $0.5M_i$ to $1.5M_i$. If the accumulation of *CSM* is greater than or equal to the threshold, the cell is said to 'migrate'. The migration process triggers a decision to be made based on the relative concentrations of *IL*-10 and *IL*-12. If the cell has accumulated more *IL*-12 than *IL*-10, its decision is positive, otherwise its decision is negative. When the decision is reported, the cell also presents all of its locally-stored antigen. All the sampled antigen for a cell are presented with the decision value. However, multiple cells can sample copies of the same antigen. The final decision for a given antigen is calculated by averaging the decision values over all cells which sampled it. When a cell has migrated, it is removed from the population and a new cell is put in its place. The migration threshold of the new cell is randomly assigned using the probability distribution. The pseudocode of the algorithm is presented in Fig. 2.

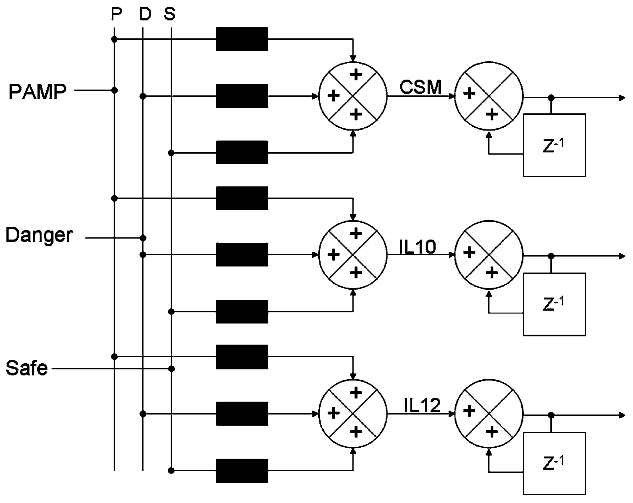

**Fig. 1** The block diagram of a dendritic cell. The variable $z^{-1}$ is used to denote a delay of one sampling iteration. The output from the summation of the *CSM* signal is compared against the migration threshold to determine the time of migration. At this time the relative size of the other two summations determines the output signal of the cell

**Table 1** The weights used to relate the input heuristic signals to the internal variables

| | PAMP | Danger | Safe |
|---|---|---|---|
| *CSM* | 2 | 1 | 2 |
| *IL*-10 | 0 | 0 | 3 |
| *IL*-12 | 2 | 1 | −3 |

This algorithm has demonstrated success with a variety of computer security applications [1, 7, 8]. Other application areas include fault detection within sensor networks [12] and threat classification within the field of robotic security [15]. All of these areas feature problem environments where there is a great deal of uncertainty associated with the information relevant to making the decision. In computer security, distinguishing which signals are sourced by legitimate use and which signals are sourced by malicious operations is an exceptionally difficult problem. In addition, for a robotic security application, the source data is often noisy and non-linear.

## 3 A simplified dendritic cell model

The first step in deriving a simplified model for the DCA is to identify how to minimise the processing performed by each unit. This was achieved by rearranging the block diagram in Fig. 1. The first step was to move any computation that does not require any persistent state from each cell into the 'tissue' (signal pre-processing). This reduces the overall algorithm processing as any computation that can be performed instantaneously from the input signals can be done once per iteration for the entire population, rather than once per cell, per iteration.

```
Function DCA(populationSize, populationDistribution)
   t = 0

   for (all cells in population)
      migration threshold = random(populationDistribution)
      Sum of CSM = 0
      Sum of IL10 = 0
      Sum of IL12 = 0
      Antigen List = blank
   end-for

   while(algorithm running)
      P = current PAMP signal
      D = current danger signal
      S = current safe signal

      for (all cells in population)
         Sum of CSM += CurrentCSM(P,D,S)
         Sum of IL10 += CurrentIL10(P,D,S)
         Sum of IL12 += CurrentIL12(P,D,S)
         Samples = randomly selected antigen from tissue
         Antigen List = [Antigen List, Samples]

         if (Sum of CSM > migration threshold)
            if (Sum of IL12 > Sum of IL10)
               output state = mature (positive)
            else
               output state = semi-mature (negative)
            end-if

            Copy Antigen List to output
            Signal output state
            Reset cell
         end-if
      end-for

      t++
   end-while
end-function
```

**Fig. 2** The pseudocode for the DCA

A comparison between two numbers can be expressed as the same comparison between the difference of those two numbers and zero. This can be applied to the comparison between *IL*-10 and *IL*-12. Rather than storing *IL*-10 and *IL*-12 separately, one can represent them both as the instantaneous difference between the two, termed *K*, i.e.:

$$K = IL{-}12 - \mathrm{IL}{-}10 \tag{1}$$

The sign of the summation of *K* can now be examined to determine the decision made by the cell. A positive value yields a positive outcome, while a zero or a negative value yields a negative outcome. Using the weights from Table 1, *IL*-10 and *IL*-12 can be expressed as the weighted sums by:

$$IL{-}10 = \mathrm{P(t)W_{pIL-10} + D(t)W_{DIL-10} + S(t)W_{SIL-10}} \tag{2}$$

and:

$$IL{-}12 = \mathrm{P(t)W_{pIL-12} + D(t)W_{DIL-12} + S(t)W_{SIL-12}} \tag{3}$$

where *P*(*t*) represents the output of the PAMP heuristic, *D*(*t*) represents the output of the danger heuristic, *S*(*t*) represents the output of the safe heuristic and $W_{\mathrm{XY}}$ is the weighting between input signal X and internal signal Y. Substituting Eqs. 2 and 3 into Eq. 1 yields:

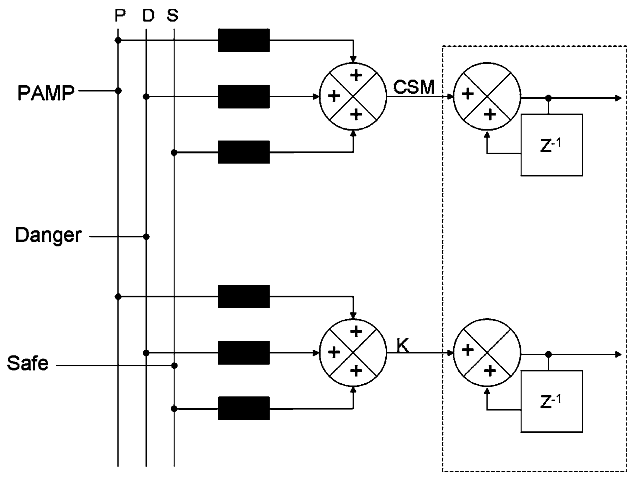

**Fig. 3** The optimised block diagram for a dendritic cell. Only processing contained within the dotted line needs to be carried out on a per cell basis. All other processing can be performed once per population

$$\begin{aligned} K = {} & P(t)(W_{pIL-12} - W_{pIL-10}) + D(t)(W_{dIL-12} - W_{dIL-10}) \\ & + S(t)(W_{sIL-12} - W_{sIL-10}) \end{aligned} \tag{4}$$

It is common to normalise the signals *P*(*t*), *D*(*t*) and *S*(*t*) to lie within 0 and 100%. Using the weights from Table 1 it is possible to identify that *K* is bounded to lie within the range −6 to + 3. The maximum value of *K* occurs when *P*(*t*) and *D*(*t*) are at a maximum and *S*(*t*) is at a minimum. The minimum value of *K* occurs when *P*(*t*) and *D*(*t*) are at a minimum and *S*(*t*) is at a maximum.

Figure 3 is the new block diagram for a DC. In this new model, *CSM* and *K* can be considered as inputs from the signal preprocessing to the cell population. This reduces the number of operations per cell, per iteration. For a population of 100 cells this represents a reduction from 180 operations per iteration (3 multiplications and 3 additions for three output signals for every cell) to 12 operations per iteration (three multiplications and three additions for two output signals for the entire population). These calculations are based on arithmetic operations and do not take into account the number of assignment operations.

## 3.1 Modelling the DCA as a filter

To analyse the flow of information through the DCA, we model a single cell as a low-pass filter. The transfer function of the cell should provide insight into the information that is used to make a decision and the information that is ignored. For the purposes of simplifying the model, we shall consider the signal being filtered to be the abstracted *K*(*t*). To model a cell as a filter, it was necessary to perform signal reconstruction from the output of the DCA. For the purposes of this model, this means that the signals used to make the decision will be explored rather than the decision itself. The standard technique for assessing the output from the DC population is to calculate the 'mature context antigen value' (MCAV) [7]. The MCAV is a symbol-specific calculation that identifies all of the cells that have voted for a given symbol within a fixed time-frame. An average is then calculated for that symbol. A value of one is attributed to cells with a positive vote and zero is attributed to cells with a negative vote. Thresholding the MCAV provides a final decision for the presented symbol. This technique has the advantages of being both computationally inexpensive and able to provide a measure of confidence. The further the result is from 0.5, the more confident the result.

However, this technique does not allow us to make detailed inferences about the input signal $K$. For this analysis, a different technique will be used to allow a reconstruction of the $K$ signal and thus allow inferences to be made about the information passed by the algorithm. Alternatives to the MCAV are also available. In [1] an alternative that takes into account the volume of antigen as well as the output signal is used to make more informed decisions about which antigen represents malicious code. In this case, we shall pass the time the cell spent accumulating signals (measured in sample steps) with the accumulated $K$ signal. By dividing the latter by the former we can estimate the average value of $K$ that the cell was exposed to. This is given by:

$$\hat{K} = \frac{\sum_{n=0}^{W_L-1} K[n]}{W_{\mathrm{L}}} \tag{5}$$

where $W_{\mathrm{L}}$ is the length of time the cell is accumulating signal. This technique not only allows an estimate of the $K$ signal to be constructed but the magnitude of $\hat{K}$ also provides information about how 'one-sided' a decision is. Large magnitudes indicate that the input signals were clearly indicating a decision, while low magnitudes indicate that the input signals were split.

For the purposes of this initial investigation we will simplify the model by assuming a constant *CSM* value of $C$. This will form the basis of future work where it is hoped to incorporate a probabilistic model of *CSM* into the model. It could be argued that a constant *CSM* is an oversimplification, as $K$ is likely to be coupled to the *CSM* signal. However, from the cell's point of view, it is only the sum of the *CSM* values that is processed. So while the constant *CSM* is a simplification, it is a valid representation of a range of values, with an average value of $C$. It is also of note that this model is intended to be instructive, given a suitably chosen estimate of the *CSM* value for a given application.

A constant *CSM* value allows the window length to be calculated by:

$$W_{\mathrm{L}} = \left\lceil \frac{M_i}{C} \right\rceil \tag{6}$$

where $M_i$ is the migration threshold of cell $i$ and $C$ is the constant value of *CSM*. The fraction is rounded up as the cell can only migrate after an integer number of steps and will only do so if the accumulated *CSM* is greater than $M_i$.

Equations 5 and 6 allow us to infer that the output of a single cell will be an average value of $K$, taken over $W_{\mathrm{L}}$ steps and reported every $W_{\mathrm{L}}$ steps.

### 3.2 Equivalence to other filters

The simplest technique for deriving the frequency response of a DC is to identify equivalence with established filters with known frequency responses. The description of the cell's output, given in Sect. 3.1, is similar to that of a moving-average filter with a length of $W_{\mathrm{L}}$. Equation 7 describes the behaviour of a moving average filter in the time domain (taken from [21]).

$$y[n] = \frac{\sum_{i=(n-(L-1))}^{n} x[i]}{L} \tag{7}$$

where $n$ is the current time step, $y$ is the output of the filter, $x$ is the input to the filter and $L$ is the length of the filter. Figure 4 compares the output of a moving average filter with the reconstructed output from a DC. A moving average filter continuously reports the average of the previous $L$ values for each value of $n$. In the DCA the population remains constant. As one cell migrates, it is reset, reassigned a new migration threshold using the relevant probability density function and put back into the tissue. This means that the output of a single DC, being repeatedly returned to the tissue with the same migration threshold will be a series of pulses, with a frequency of $1/W_{\mathrm{L}}$, where the magnitude of each pulse is the output of the moving average filter at that point in time. It is of note that Fig. 4 is for illustrative purposes only. The large distance between the filtered line and the original is a result of the input data being random and the window size being large relative to the number of points. This type of system can be realised using the transfer function expressed using the block diagram in Fig. 5.

The frequency domain transfer function of a moving average filter with length $L$ is given by:

$$Y(\omega) = \frac{\sum_{g=0}^{L-1} \mathrm{e}^{-jg\omega}}{L} \tag{8}$$

For this equation to describe the averaging process within a dendritic cell, we must substitute the window length $L$ for the number of samples that a dendritic cell samples over,

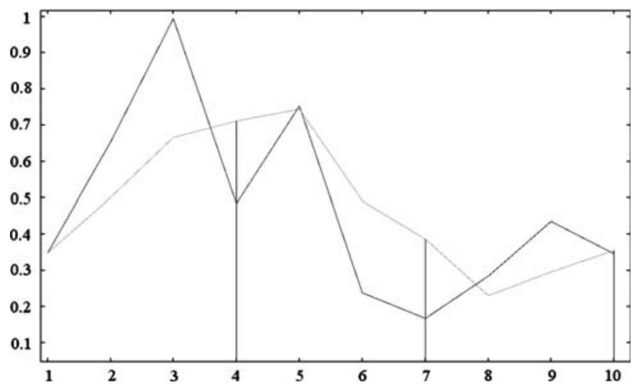

**Fig. 4** The output from a DC for constant $C$ and $M_i$. The $x$-axis is in sample steps and the $y$-axis is arbitrary. The output from the cell is represented as impulses occurring every three steps. The *dark line* is a randomly generated input signal. The *light line* is a moving average of that signal with a window size of 3

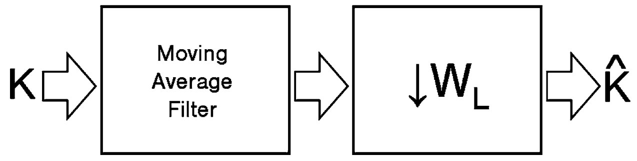

**Fig. 5** The equivalent to a dendritic cell implemented as known filters. The first box is a moving average filter with a window length of $W_L$ and the second box is a downsampler which reduces the sample rate of the input by a factor of $W_L$

$W_L$. In [18] the transfer function of a filter $X(\omega)$, downsampled by the integer $M$ is given by:

$$V(\omega) = \frac{\sum_{g=0}^{M-1} X(\omega + (2g\pi))}{M} \tag{9}$$

The downsampling factor of a DC is the number of input samples required before the DC produces an output, the window length $W_L$.

Substituting Eq. 8 for $X(\omega)$ in Eq. 9 allows us to calculate the transfer function of a moving average filter, downsampled by $M$. Further substituting $W_L$ for $M$ and $L$ gives the transfer function of a moving average filter with length $W_L$ being downsampled by $W_L$. This is the proposed model of a DC. The result is given in Eq. 10.

$$H(\omega) = \frac{\sum_{g=0}^{W_L-1} \sum_{b=0}^{W_L-1} e^{-jb((\omega+(2g\pi)))}}{W_L^2} \tag{10}$$

This model will provide the basis for our investigation into the frequency properties of the dendritic cell. This is an important derivation as it is common in engineering applications to precede a downsampling block with a filter [11]. Such a filter-downsampling pair is called a "decimator". A filter precedes the downsampler because if the original input signal contains any frequencies that satisfy the inequality in Eq. 11, the downsampling process will introduce harmful aliasing artefacts into the output.

$$f > \frac{f_s}{2M} \tag{11}$$

where $f_s$ is the sampling frequency and $M$ is the downsampling factor.

In engineering terms, the 'cut-off frequency' or 'corner frequency', $f_c$ of a filter is an important property. The gain for all frequencies greater than $f_c$ is considered to be negligible enough for these frequencies to be ignored. Any frequency below $f_c$ is said to be 'passed'. This frequency is generally considered to be the point where the gain has dropped to $1/\sqrt{2}$. For the moving average filter this can be calculated using:

$$f_c = \frac{0.443 f_s}{L} \tag{12}$$

taken from [21]. In the case of a DC, the downsampling factor and the filter window length are both equal to $W_L$. Rearranging Eqs. 11 and 12 reveals that the DC removes 88.6% of the frequencies that could cause aliasing problems. This implies that certain signal frequencies will be more prone to introducing incorrect classifications due to aliasing errors in the signal processing equation. However, the bulk of potentially dangerous signals are removed.

## 3.3 Verification of the model

In order to verify this model we can compare the predicted frequency response with the actual frequency response of a single DC with a constant *CSM* input and migration threshold. To do this we must use a modified version of the algorithm with an additional signal reconstruction stage at the output of the cell. This was generated by making $K$ equal to various sine waves at different frequencies and recording the magnitude of the resultant sine waves from both the model and the dendritic cell. This is a standard technique for calculating the frequency response of a system, as any periodic signal can be represented as a weighted sum of sinusoids. To fully verify the model this must be performed for a variety of *CSM* values and migration thresholds.

All of the experiments are performed using frequencies between 0 and the Nyquist frequency of the system. The Nyquist frequency, $f_n$, of a system is half the system's sampling rate. This is a valid test as the frequency response of a system from 0 to $f_n$ is exactly the same as the frequency response for the system from any $Xf_n$ to $(X + 1)f_n$ where $X$ is an even number. For frequency responses in regions $Yf_n$ to $(Y + 1)f_n$ where $Y$ is an odd number, the response is the mirror of the response from 0 to $f_n$. Thus, establishing that the model is accurate from 0 to the Nyquist frequency establishes that the model is accurate for any frequency. See pages 41–43 of [11] for a more detailed explanation.

**Table 2** The parameter values used for experiments 1–9

| Migration thresholds ($M_i$) | *CSM* signal values ($C$) | Calculated $W_L$ |
|---|---|---|
| 30 | 10 | 3 |
| 60 | 10 | 6 |
| 120 | 10 | 12 |
| 30 | 20 | 2 |
| 60 | 20 | 3 |
| 120 | 20 | 6 |
| 30 | 30 | 1 |
| 60 | 30 | 2 |
| 120 | 30 | 4 |

**Table 3** The parameter values used for experiments 10–24

| Migration thresholds ($M_i$) | Sampling frequencies($f_s$) |
|---|---|
| 30 ($W_L$ = 3) | 0.5 |
| 60 ($W_L$ = 6) | 1 |
| 120 ($W_L$ = 12) | 2 |
| | 4 |
| | 10 |

For each migration threshold, the calculated $W_L$ is also listed

Two sets of experiments were run. Firstly, nine runs of the experiment were performed keeping the sampling rate at 1 Hz, testing every migration threshold against every *CSM* signal value listed in Table 2. Secondly a further fifteen experiments were performed, keeping the *CSM* signal at 10 and using every value of the migration threshold with every sampling frequency in Table 3.

### 3.3.1 Results

In all cases the response followed a similar overall shape. The gain has an initial drop from 1 to 0 and in the higher frequencies, oscillations in the gain value can be observed. These oscillations are known as "ripple". The total, absolute error over the entire frequency range for the model can be defined as the Euclidean distance between its predictions and the actual response from 0 to the Nyquist frequency. In the worst case ($M_i$ = 120, $C$ = 30) the total, absolute error was approximately 0.69 and is shown in Fig. 6. This value represents that the model accurately follows the general shape of the DC response. This is an acceptable result.

It is clear from the plot that the error arises from circumstances where the actual DC gain transiently drops. This is not a serious concern for this investigation as we are merely interested in determining which frequency ranges are passed and which frequency ranges are rejected by the

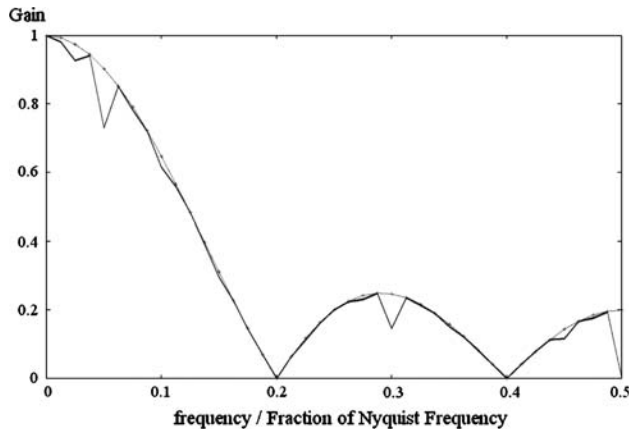

**Fig. 6** The worst performing model prediction ($M_i$ = 120 and *CSM* = 30). The *grey*, *dot-dashed line* represents the model's prediction and the *thick*, *solid line* represents the actual frequency response of the algorithm under those conditions

DCA. It is the general shape of the response that is important. In future work the cause of these transient drops will be investigated further.

Figure 7 shows the effects of changing the *CSM* value for a constant value of $M_i$ (60). As the *CSM* value is increased the initial drop-off slope becomes slower. The ripple in the cut-band becomes larger with a lower *CSM*. Both of these aspects are generally considered to be negative features in a filter. A gradual cut-off slope means that more of the unwanted frequencies are close enough to the cut-off frequency to have an effect on the output signal. A high magnitude of ripple means that some higher frequencies have a disproportionately high effect on the output signal. This reinforces the idea that migration thresholds which are low relative to the *CSM* signal can lead to erroneous and noisy data being presented as part of

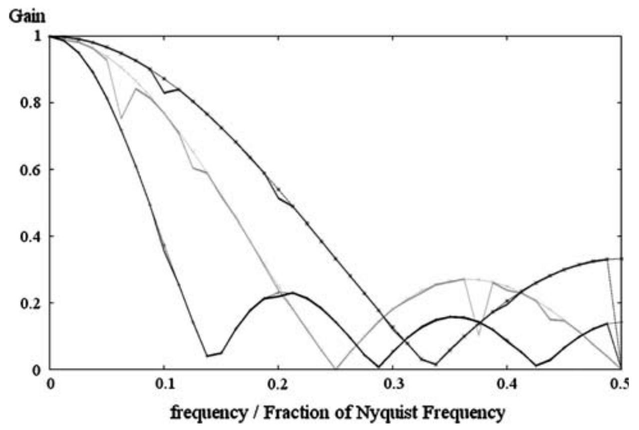

**Fig. 7** The effects of varying the *CSM* value. For these experiments the value of the migration threshold is set to 60 and the *CSM* value is 10, (*black line*) 20, (*light grey line*) and 30, (*dark grey line*). In each case the *dot-dashed line* is the model prediction and the *thick*, *solid line* is the actual response

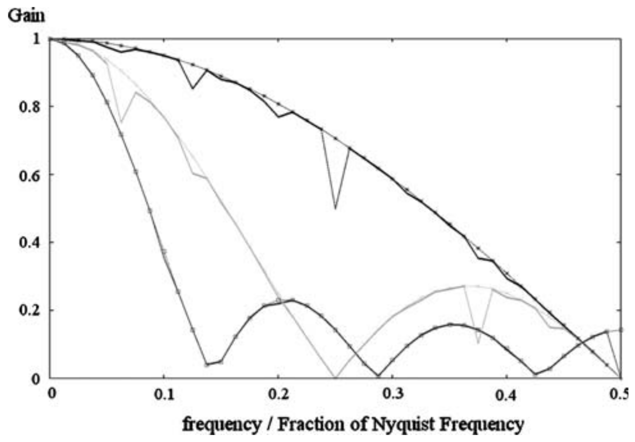

**Fig. 8** The effects of varying the migration threshold. For these experiments the value of the *CSM* is held at 20 and the migration threshold is 30 (*black*), 60 (*light grey*) and 120 (*dark grey*). In each case the *dot-dashed line* is the model prediction and the *thick-solid line* is the actual response

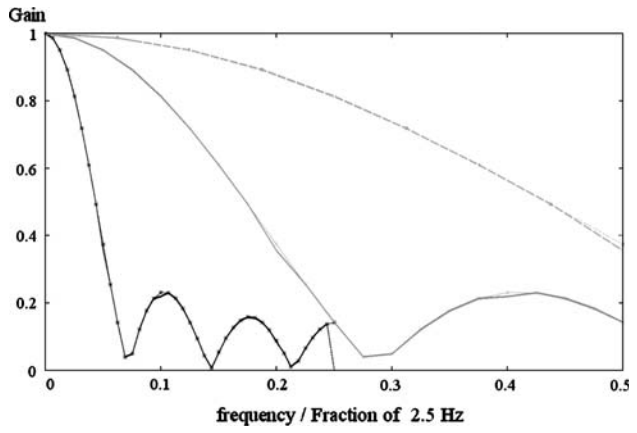

**Fig. 9** The effects of varying the sampling frequency. The *CSM* was held at 10, the migration threshold was held at 60 and the sampling frequency was changed from 0.5 Hz (*black*), 2 Hz (*solid grey*) and 5 Hz (*dashed grey*). The *x*-axis is scaled to the Nyquist frequency of a system with a sampling rate of 5 Hz (2.5 Hz)

the output. High values of the migration threshold relative to *CSM* cause a fast drop-off in the cut band, but this means that if the useful information is in the cut band, its information will be lost. In all of these cases the model predicts the performance of the algorithm excellently.

Increasing the migration threshold predictably had the inverse effect to increasing the *CSM* value. This supports previously seen experimental behaviour of the algorithm. Again the model's predictions were accurate across the entire range. These results can be seen in Fig. 8.

For the next set of experiments the sampling frequency of the algorithm was altered. For all experiments the effect was to simply scale the same shaped frequency response between 0 and the new Nyquist frequency. Three of the results can be seen in Fig. 9. In all cases the error rate was identical for all migration threshold/*CSM* pairs for all sampling frequencies. This is intuitive, given a stationary *CSM*, a larger sampling rate will proportionally increase the rate at which the cells migrate.

### *3.3.2 Analysis of results*

For all parameters, the predictions from the model were reasonably accurate. The significant effect of changing the ratio between the migration threshold and the *CSM* signal highlights the importance of correctly parameterising the algorithm. This further emphasises the need for a more informed way of selecting the migration threshold. Figures 7, 8 and 9 clearly demonstrate that the correct selection of the migration threshold will significantly alter which information is passed and which information is cut. The shapes of frequency responses are all characteristic of low-pass filters. This suggests that when used with applications where the useful information has a higher frequency than the noise, that the DCA would fail. A possible solution to this could be to use the input heuristics to cut any irrelevant low frequency information.

The scaling effects of varying the sampling frequency had no impact on the accuracy of the model. The scaling effect is an important result as it demonstrates that selecting the correct sampling frequency for a given application is crucial. In some microprocessor-based sampling systems the sampling rate can vary in times of high processor loading. Such systems could potentially be incompatible with an algorithm whose performance has a high sensitivity to the sampling frequency. However, it is likely that this is rectified by the population-based nature of the algorithm, so long as the spectrum of migration thresholds is chosen appropriately.

The construction of an accurate frequency domain model of a DC has an impact on using the DCA in the future. With representative samples of the *CSM* and *K* signals, for a given application, it may now be possible to tune the probability density function for the migration thresholds of the population to provide an optimum response for the application. By selecting a migration threshold that rejects data that could be potentially misleading to the decision making process our hypothesis is that the error rate of the algorithm could be significantly reduced. This of course assumes that the irrelevant information has a higher frequency than the useful information. This assumption is true for many complex problems. In robotics the misleading information comes from noisy sensors injecting high-frequency noise into the input data. The current model lacks the capacity to be parameterised using probabilistic methods, though such a development is intended as part of the future work. As a result of the model's current state it is not possible to tune either the shape nor the width of the probability density function. However, it may be possible to use the model to provide an insight into an appropriate mid point for the standard uniform distribution used in previous applications.

## 4 Tuning the DCA using the frequency domain model

It is hypothesised that the model proposed in Sect. 3.1 can be used to identify an appropriate choice of $M_i$, the midpoint for the migration threshold distribution, for the original DCA algorithm. The tuning methodology that we have devised can be broken down into six distinct stages.

Step One Take an indicative sample of the typical values of *K* and *CSM* that the system will encounter for the application.

Step Two Identify the median period between “interesting events”. The definition of “interesting” will vary from application to application. However, if the application-specific heuristics generated as part of the algorithm’s implementation process are appropriately selected, these events will correlate with peaks in the *CSM* and *K* values. An interesting event should cause a positive peak in *CSM* and either a positive or a negative peak in *K*. Using the median period is likely to remove outliers.

Step Three Identify the length of the interesting event with the lowest *CSM* value associated with that event. Events of interest will generate a spike in the *CSM* value of the system. The smallest spike is indicative of the most difficult event to extract from the input data. Record the median value of the spike and its duration.

Step Four Calculate the target corner frequency. The target corner frequency is the inverse of the period calculated in step two. As the filtering process has a very gradual cut-off (see Sect. 3.3.1) the fact that this is an approximation should not prevent any useful information from passing through to the decision making process.

Step Five Calculate the constant *CSM* value for the model. The median value of the smallest spike divided by the duration of that spike provides a good approximation of the value of *C* (*C* is the constant *CSM* value to use in the model). Note that this value is not intended to represent the entire range of the *CSM* signal for the application, but instead represents those events that contain useful information, but would be most vulnerable to over-aggressive filtering. Dividing the magnitude by the length of the spike ensures that the summation of *C* over the length of this spike alone will cause a cell migration. This should become the smallest artefact within the signal to be used in the decision making process.

Step Six The transfer function is used to search the space of available frequency responses to find one with a corner frequency which is approximately equal to the target corner frequency. A search is used rather than a calculation as it is possible that a range of possible solutions could exist for the same search parameters. It is also possible that no solution provides the target corner frequency and that a value of $M_i$ needs to be selected that approximates the target. For the purposes of this investigation, all integer values of $M_i$ were tried in the range 1–240. The upper limit of this search will obviously be effected by the application.

Figure 10 is a graphical representation of the tuning methodology outlined. In the following section we apply this methodology to a case study based on a previous

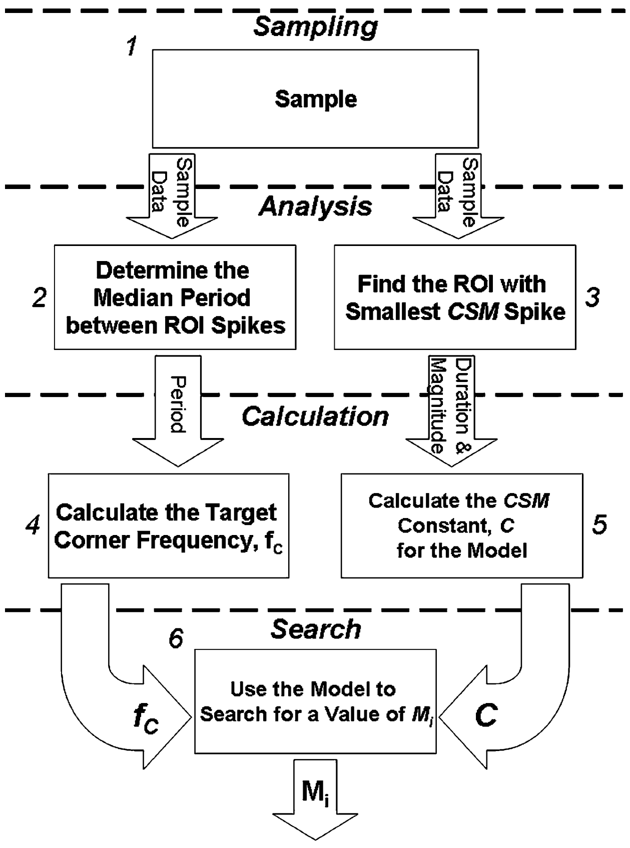
Sampling
1
Sample
Sample Data
Sample Data
Analysis
2
Determine the Median Period between ROI Spikes
Find the ROI with Smallest CSM Spike
3
Period
Duration & Magnitude
Calculation
4
Calculate the Target Corner Frequency, f_C
Calculate the CSM Constant, C for the Model
5
Search
6
Use the Model to Search for a Value of M_i
f_C
C
M_i

**Fig. 10** A flowchart representation of the tuning methodology. *Numbers* in italics relate flowchart stages to methodology steps

application of the DCA. The tuned migration threshold is evaluated to see how it performs when compared to the arbitrarily picked migration thresholds.

## 5 Revisiting the robotic classification problem

In [15] a prototype security system was proposed. The original DCA was used to assess sensor readings to decide if the current situation was of interest to a human operator or not. This style of security system, where the robot(s) act(s) as intelligent, mobile sensor-platforms, capable of drawing the attention of a security professional to an 'anomalous' or dangerous situation, has been explored within the field of robotics before using a variety of different decision making techniques [3, 4, 16, 17]. The robot used was a Pioneer manufactured by MobileRobots Inc. (http://www.mobilerobots.com/). It was decided to revisit this application as it represents a problem that the original DCA has been tested on before, with a degree of success. The use of unfiltered sensor data as the basis for the input heuristics guarantees a noisy, uncertain decision making environment.

In [15] the robot is placed in an environment as described by Fig. 11. It is of note that space restrictions prevented a pen of the same size as [15] being used. The inputs to the DCA are fed directly from normalised sensor readings. The danger signal is connected to the sonar array, the safe signal is connected to the laser range finder, and the PAMP signal is connected to the output of an image processing algorithm acting on the camera. The ranged sensors are normalised using a look-up table. This relates signal strength to the measured distance away from the closest object within the sensor scanning range. Table 4 shows the look-up table values. The signal values for distances lying between the table values are calculated using linear interpolation.

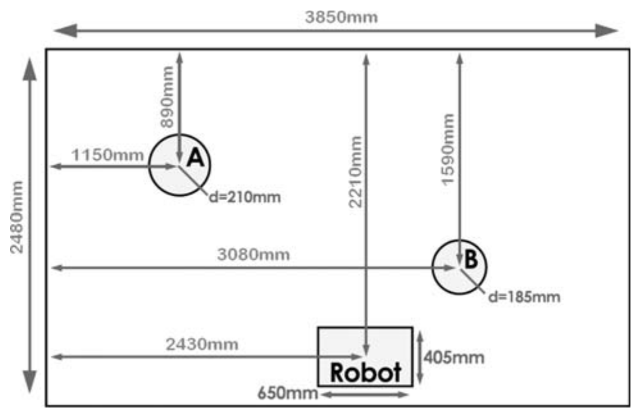

**Fig. 11** The robot pen layout for the experiments (not to scale). The shorter cylinder is the cylinder with a diameter of 210 mm. An updated version of the diagram taken from [15]

**Table 4** The look-up table used to normalise ranged sensor readings

| Distance (mm) | Signal strength |
|---|---|
| 0 | 100 |
| 300 | 90 |
| 600 | 50 |
| 900 | 20 |
| 1,200 | 0 |

The image processing unit uses histogram back-projection [19] to identify clusters of pixels within each captured frame that have similar colour characteristics to a target object. The algorithm is trained using a single image of a pink cylinder placed within the robot's environment. The output from the algorithm is the volume, in pixels, of the largest cluster with similar characteristics to the training image. This is normalised by the PAMP heuristic into a percentage of 6798.5 pixels. This value is the half-way point between the median output and the maximum output of the algorithm after a seven minute random walk around the robot's environment.

The antigen in this application is a representation of the robot's position and orientation. The pen is divided into a grid with 30 cm × 30 cm spacing. Each grid square is further subdivided into twelve segments, representing direction. Each segment is assigned an integer identifier. These identifiers are used as antigen. It was decided to add an antigen multiplier to this system. This ensures that multiple copies sample the same antigen. For this application a dynamic multiplier is used. This means that the number of additional antigen copies made is not the same for all antigen. Equation 13 (taken from [15]) is the formula used to calculate the number of antigen copies to make. This equation provides a number between 2 and 102. It is composed of a variable range from 0 to 100 with an offset of 2. The offset ensures that 2 copies of every antigen are added. The variable range is inversely proportional to the speed of the robot. Locations which the robot passes through quickly, necessarily contribute less sensor data. This formula ensures that such locations also contribute less antigen. Translational velocity contributes more to the formula as it was observed that the rotational speed of the robot had a smaller impact on the robot's movement through the antigen enumerations.

$$W(v,\dot{\theta}) = 75\left(1 - \left|\frac{v}{v_{\max}}\right|\right) + 25\left(1 - \left|\frac{\dot{\theta}}{\dot{\theta}_{\max}}\right|\right) + 2 \quad (13)$$

In Eq. 13 $v$ is the velocity of the robot, $\dot{\theta}$ is the rotational velocity of the robot and $W$ is the amount of antigen added to the environment.

By using the laser as the inhibitory safe signal, it was theorised that the shorter cylinder, which is too small to reflect the laser from the sensor, should be recognised as an anomalous object. In contrast the taller cylinder and the walls of the pen would be ignored.

As a base-line for the experiment, a theoretically "perfect" response was calculated using a Java program to identify all of the enumerated segments which have line of sight to the anomalous object.

For every experiment a population of 100 cells was used. In keeping with the original DCA, the migration thresholds were generated using a uniform distribution. For the initial experiments a range of $M_i \pm (0.5M_i)$ where $M_i$ is the specified mid point for the migration thresholds. To gain a better understanding of the effectiveness of the tuning algorithm the experiments were repeated using a small distribution of $M_i \pm (0.1M_i)$.

The original results from the experiment, whilst promising, suffered from a continual degradation of performance over time. This was originally attributed to errors in the robot's tracking algorithm, which did not take into account wheel slippage. Over time the robot's concept of its own location within the pen became increasingly incorrect. As the experiment was validated using a location-based assessment, these errors seriously harmed the perceived false-negative and false-positive rates. In the experiments presented in this paper this limitation was overcome by providing the robot with a map identifying the shape of the outer walls of the pen. With this information a Monte-Carlo localisation algorithm corrected the odometry error. This generated higher quality results, with a more consistent performance over time.

## 5.1 Tuning the algorithm

A sample of the *CSM* and *K* values for a typical run of the experiment were taken. The sample rate used was 4 Hz. Figure 12 shows the first 400 samples covering a period of 10 s.

The estimated peaks from the sample in Fig. 12 are given in Table 5. The median distance of 30.5 samples gives a target corner frequency of approximately 0.131 Hz.

The 'interesting' event with the lowest *CSM* signal was identified as the small peak in *CSM* between samples 297 and 302. The positive *K* signal associated with this period implies that it was a transient inspection of the anomalous cylinder. The median *CSM* value between these points is approximately 22.9. This equates to a constant *CSM* value of approximately 4.5.

After searching the parameterised space of frequency responses a migration threshold of 54 was identified as an optimal result.

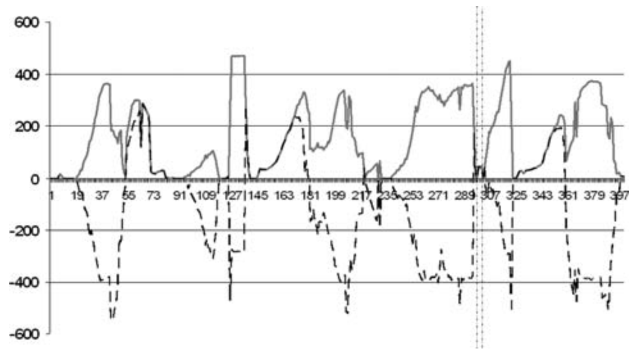

**Fig. 12** A plot of the magnitude of *K* and *CSM* against time (in sample steps). A 10 s sample from the robot application. The *dashed line* is the *K* signal and the *solid line* is the *CSM* value. The *vertical, dot-dashed lines* highlight the region of interest

**Table 5** The sample numbers of the peaks identified within Fig. 12

| Sample number | Distance from last peak |
|---|---|
| 42 | NA |
| 62 | 20 |
| 114 | 52 |
| 133 | 19 |
| 178 | 45 |
| 205 | 27 |
| 230 | 25 |
| 264 | 34 |
| 299 | 35 |
| 321 | 22 |
| 356 | 35 |
| 380 | 24 |

## 5.2 Results

Tables 6, 7 and 8 show the results from the experiments on the robot using a distribution of $\pm 0.5M_i$, $\pm 0.1M_i$ and $\pm 0$. The error rates presented are the average percentage of incorrect readings over 10 blocks of 12 s.

In Table 6, selecting the best performing value of $M_i$ is a non-trivial task. The value of 54 has by far the lowest false positive rate and the lowest standard deviation. The false negative rate for an $M_i$ of 15 is the lowest though this appears to be at the cost of the false positive rate. The false negative rates for 30, 54 and 60 are all comparable, though 60 has both the lowest value and the lowest standard deviation. The poor performance of the lowest and highest values of $M_i$ are unsurprising. A low value of $M_i$ will be prone to error as noise in the input signals will cause premature migration. Making decisions based on such little data is obviously more prone to noise. Conversely the large values will collect large numbers of antigen and apply the same decision to all. This is a particularly disastrous

Table 6 The results from the experiments on the robot using a migration threshold distribution of $M_i \pm (0.5M_i)$

| $M_i$ | Average false positive (%) | False positive deviation | Average false negative (%) | False negative deviation |
|---|---|---|---|---|
| 15 | 13.70 | 0.13 | 5.96 | 0.07 |
| 30 | 12.88 | 0.10 | 9.16 | 0.10 |
| 54 | 4.77 | 0.09 | 10.09 | 0.12 |
| 60 | 9.87 | 0.16 | 8.84 | 0.09 |
| 120 | 13.47 | 0.16 | 10.32 | 0.15 |
| 240 | 9.78 | 0.17 | 17.35 | 0.18 |

Table 7 The results from the experiments on the robot using a migration threshold distribution of $M_i \pm (0.1M_i)$

| $M_i$ | Average false positive (%) | False positive deviation | Average false negative (%) | False negative deviation |
|---|---|---|---|---|
| 15 | 7.05 | 0.13 | 7.84 | 0.09 |
| 30 | 9.11 | 0.10 | 7.41 | 0.09 |
| 54 | 15.01 | 0.19 | 7.71 | 0.09 |
| 60 | 5.00 | 0.13 | 6.83 | 0.10 |
| 120 | 11.70 | 0.10 | 12.31 | 0.14 |
| 240 | 2.50 | 0.08 | 9.69 | 0.12 |

Table 8 The results from the experiments on the robot using a migration threshold distribution of $M_i \pm 0$

| $M_i$ | Average false positive (%) | False positive deviation | Average false negative (%) | False negative deviation |
|---|---|---|---|---|
| 15 | 16.65 | 0.13 | 6.15 | 0.08 |
| 30 | 16.82 | 0.18 | 9.64 | 0.12 |
| 54 | 11.10 | 0.12 | 9.62 | 0.13 |
| 60 | 10.81 | 0.12 | 9.47 | 0.12 |
| 120 | 10.28 | 0.14 | 12.00 | 0.14 |
| 240 | 9.17 | 0.14 | 9.16 | 0.14 |

strategy for periods of low *CSM* where excessively large numbers of antigen could potentially be collected.

In Table 7 the best performing value of $M_i$ is clearly 60, with the second best false positive rate and the best false negative rate. The false positive rates of all the values improve with the smaller distributions, with the exception of 54, the tuned value. The performance of an $M_i$ of 54 degrades heavily with a significantly worse false positive rate and a much worse standard deviation. The only false negative rate that was significantly altered by changing the distribution width was that of $M_i = 240$.

When the migration threshold was kept constant across the entire population $M_i = 60$ still outperforms the other migration threshold values. The performance of $M_i = 60$ with a large deviation is comparable with the performance of $M_i = 60$ with no deviation, whilst the mid-range performance is much better. For the experiments with constant $M_i$, the tuned value of 54 is one of the better performing values.

### 5.3 Conclusions

The initial experiments suggested that the tuned value was performing well. However, its severe degradation with a smaller distribution suggests that the tuning was ineffective and that the performance was a side effect of the large range touching on a better selection of migration thresholds. However, when the range was reduced to zero the performance of the tuned value, in comparison to the other values, is amongst the best. It was still out performed by one of the arbitrarily pick values.

The large changes in performance with different distributions suggest that the key to improving the algorithm's performance is not simply to identify a suitable mid point, but to also shape the distribution appropriately. It is possible that a mapping based on the distribution of *CSM* values or input frequencies could be used to calculate the appropriate shape. This is further supported by the degradation of the results for constant values of $M_i$. As the tuned value was calculated using data from the region of interest with the lowest *CSM* it is possible that this value is not a suitable midrange value, but in fact the lower bound of a suitable distribution.

## 6 Discussion

The DCA has been optimised in this paper to improve the speed of execution for future implementations of the algorithm. This optimised model has been evaluated mathematically to derive the transfer function for a single DC under the condition that the *CSM* value is held constant. The accuracy of this model was verified experimentally and found to be an excellent indicator for the performance of a DC with a constant *CSM* input. Most importantly, it has been demonstrated that there is a clear relationship between the frequency of the information presented by the input heuristics and the amount of influence that information has on the final decision made by the cells. Insight has been gained into the relationship between the *CSM* signal, the migration threshold, and the sampling frequency of the DCA. A low migration threshold has been shown to indicate a more gradual reduction in gain as the frequency is increased. This increase in the volume of information being used in the decision making process is intuitive. In contrast, a high migration threshold means that higher frequencies are cut from the decision making process. A tuning methodology, based on the transfer function equation, was put forward and investigated using a robotic application of the DCA. Whilst the tuning methodology was found to yield mixed results, this does not detract from the validity of the model which has been demonstrated to be accurate. Instead the results indicate that more work is required to identify a model capable of generating a good distribution of migration thresholds for a given application.

The derivation process demonstrated that a useful measure of the output of a DC is $\hat{K}$. This estimate of the overall signal that the cell was exposed to during its sampling phase provides an indication of how certain the cell is of the final decision. Cells with low magnitudes of $\hat{K}$ have been exposed to low signal values or approximately equal amounts of positive and negative signal. This could potentially be incorporated into future versions of the DCA to provide an alternative to the MCAV that is more resilient when cells are exposed to multiple, conflicting events during their sampling phase. A negative feature of this technique is that it requires some measure of 'age' in order to calculate the estimate, a measurement which is not found in the biological model.

In future work the transient drops in gain for certain frequencies in the normal DCA will be investigated. These are the main source of error between the model the actual algorithm. A more advanced model that can take into account the varying nature of the *CSM* would also potentially yield more insight into the workings of the DCA.

**Acknowledgments** The authors would like to thank Dr. Gordon Wyeth of the University of Queensland for his technical insight and support; William Wilson and Phil Birkin for their recommendations and suggestions throughout the derivation process and Heather Chapleo for her assistance during the practical experiments. This project is financially supported by MobileRobots Inc.

## References

1. Al-Hammadi Y, Aickelin U, Greensmith J (2008) DCA for Bot Detection, Submitted for The 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong
2. Campelo F, Guimares FG, Igarashi H, Ramirez JA, Noguchi S (2006) A modified immune network algorithm for multi-modal electromagnetic problems. IEEE Trans Magn 42:1111–1114
3. Castelnovi M, Miozzo M, Scalzo A, Piaggio M, Sgorbissa A, Zaccaria R (2003) Surveillance robotics: analysing scenes by colours analysis and clustering. CIRA
4. Everett H, Gilbreath G, Heath-Pastore T, Laird R (1994) Controlling multiple security robots in a warehouse environment. AIAA/NASA conference on intelligent robots
5. Freitas AA, Timmis J (2007) Revisiting the foundations of artificial immune systems for data mining. IEEE Trans Evol Comput 11(4):521–540
6. Greensmith J (2007) The dendritic cell algorithm. PhD Thesis. The University of Nottingham
7. Greensmith J, Aickelin U, Twycross J (2006) Articulation and clarification of the dendritic cell algorithm. ICARIS'06
8. Greensmith J, Twycross J, Aickelin U (2006) Dendritic cells for anomaly detection. Congress on evolutionary computation (CEC)
9. Greensmith J, Aickelin U, Cayzer S (2005) Introducing dendritic cells as a novel immune inspired algorithm for anomaly detection. ICARIS'05
10. Hart E, Timmis J (2008) Application areas of AIS: the past, the present and the future. Appl Soft Comput 8:191–201
11. Ifeachor EC, Jervis BW (2001) Digital signal processing: a practical approach. Prentice Hall, Englewood Cliffs, ISBN 978-0201596199
12. Kim J, Bentley PJ, Wallenta C, Ahmed M, Hailes S (2006) Danger is ubiquitous: detecting mis-behaving nodes in sensor networks using the dendritic cell algorithm. ICARIS '06
13. Lutz MB, Schuler G (2002) Immature, semi-mature and fully mature dendritic cells: which signals induce tolerance or immunity? Trends Immunol 23(9):991–1045
14. Matzinger P (1994) Tolerance danger and the extended family. Ann Rev Immunol 12:991–1045
15. Oates R, Greensmith J, Aickelin U, Garibaldi J, Kendall G (2007) The application of the dendritic cell algorithm to a robotic classifier. ICARIS'07
16. Pastore T, Everett H, Bonner K (1999) Mobile robots for outdoor security applications. ANS'99
17. Saitoh M, Takahashi Y, Sankaranarayanan A, Ohmachi H, Marukawa K (1995) A mobile robot testbed with manipulator for security guard application. IEEE international conference on robotics and automation
18. Strang G, Nguyen T (1996) Wavelets and filter banks. Wellesley-Cambridge Press, ISBN 978-0961408879
19. Swain M, Ballard D (1991) Color indexing. Int J Comput Vis 7(1)
20. Taylor D, Corne D (2003) An investigation into negative selection algorithm for fault detection in refrigeration systems. ICARIS'03. Springer, Heidelberg, pp 34–45
21. Young SS (2001) Computerized data acquisition and analysis for the life sciences. Cambridge University Press, Cambridge. ISBN 978-0521565707
22. Zhong Y, Zhang L, Huang B, Li P (2006) An unsupervised artificial immune classifier for multi/hyperspectral remote sensing imagery. IEEE Trans Geosci Remote Sens 44:420–431