Combining Perception Considerations with Artificial Intelligence in Maritime Threat Detection Systems

Ian Chen
Plano West Senior High School
Plano, TX, USA
ianyc2004@gmail.com

Lucy Huang
Glendale High School
Springfield, MO, USA
hulucy1905@gmail.com

Jack Qiao
Walker Department of Mechanical Engineering
University of Texas at Austin
Austin, TX, USA
jackqiao2002@utexas.edu

Dan E. Tamir
Department of Computer Science
Texas State University
San Marcos, TX, USA
dt19@txstate.edu

Naphtali Rishe
Knight Foundation School of Computing and Information Sciences
Florida International University
Miami, FL, USA
rishen@cs.fiu.edu

2022 17th Annual System of Systems Engineering Conference (SOSE) | 978-1-6654-9623-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/SOSE55472.2022.9812640

***Abstract*—Over the past few years the need for early-warning maritime threat detection systems has dramatically increased. Our research aims to address this need by tackling three main problems: 1) classify boat activities into three categories: random walk, following, and chasing, 2) real-time classification of boat path trajectories, and 3) designing a novel perception-based framework for activity detection in the maritime context. We propose the implementation of an entropy-based detection algorithm, trained using synthetic data. We assess the viability of the proposed framework based on accuracy and the number of time steps required prior to identification. The synthetic data generated has the potential to spur other research efforts in the field of maritime detection.**

***Index Terms*—perception, Markov model, entropy, k-means, maritime threat detection**

I. Introduction

With the rise of illegal maritime activity [1], the need for artificial intelligence threat detection systems has dramatically increased. These systems use sensors, images, and videos to obtain large amounts of data, used to perform control operations via Computation with Words, as well as Computation with Numbers [2]. This approach is very promising as it has the ability to acquire and process enormous amounts of data. It allows the detection algorithm to behave autonomously. The main goal of this research is to explore the capability of the proposed detection algorithms to identify maritime threats in real-world situations. This goal implies two main problems: 1) given a time series of boat locations (i.e., a trajectory), identify boat activities such as random walk (representing a non-threatening activity) following, and chasing, and 2) classify trajectories under real-time constraints. The proposed solution is to design a perception-based framework that includes generating synthetic trajectories then implementing an entropy-based detection algorithm that employs history buffers on a given trajectory to find its entropy. Finally, we compare the entropy to a threshold in order to determine the boat activity. This method maintains a high level of accuracy while minimizing the number of time steps required. The evaluation of the detection algorithms is implemented via time-driven simulations governed by perception-based Markov models. For generating data involving movement patterns of "following" and "chasing", we utilize the Bresenham line-drawing algorithm [3]. The proposed AI-based system allows for an efficient method of detecting boat activities and maritime threats. The rest of the paper reviews background knowledge, provides the methodology and the experimental setup, presents results, and discusses applications for the research.

II. Background

A. Related Work

Obradovic et al., as well as Roy et al., analyze different ways to detect maritime movements [4-5]. These papers, however, focus on using neural networks to detect anomalous behavior, whereas our research aims to classify boat trajectories using an entropy-based system. Goodarzi et al. use the AIS satellite data to detect anomalous behavior [6]. In contrast, we use synthetic data instead of using AIS data. This is because the AIS data has low time resolution, with some data points occurring more than a month apart. Consequently, it is hard to accurately detect trajectories from the AIS data. Xin et al. use entropy and support vector machines [7]. This is a different method, introducing notable trade-offs compared to our method. Amir et al. propose a system for mining vessel trajectories from AIS data for illegal fishing detection [8]. In contrast, we use vessel trajectory to classify boats' activities. Fooladvandi et al., and Dabrowski et al. employ Bayesian networks to detect and synthesize boat activities [9-10]. In contrast, we use an entropy-based detection algorithm. Their method introduces trade-offs compared to our method.

B. Terminology

In this section, important terms and concepts used in this paper are defined and explained.

Bresenham Algorithm: The Bresenham algorithm is used to draw an approximation of a line in an integer coordinate system. Given the coordinates of the current point on a line, this algorithm determines the coordinates of the following points by selecting points that form the closest approximation of a line, adjusting the error with each step [3]. Our work utilizes the Bresenham algorithm to plot trajectories in the simulations of chasing and following.

Markov Models: Markov models are stochastic models that assume that future system's states depend only on k previous states. The use of these models provides a method for analyzing dynamic systems, in this case boats, since the predicted future positional boats' states depend on the past positional states [11]. In our work, Markov models are used to model boat activities, in a way that reflects human perception of boat activities, and to generate synthetic trajectory data.

Entropy: In Information Theory, first-order entropy is defined as follows [12]:

$$H = -\sum_{i} P(i) \log_2 P(i) \tag{1}$$

In our work, the entropy, H, is calculated and aggregated for many trajectories. Then, an entropy threshold is calculated for classification of the trajectory activities. The maximum entropy implies a random trajectory, while the minimal entropy implies a deterministic trajectory.

History Buffer: The detection algorithm uses multiple history buffers, which store a set amount of data points for calculating the entropy through a sliding-window.

K-means Clustering: Traditionally, a cluster is a collection of data points close together presumably due to key similarities; and the term "centroid" refers to the best representation of that cluster as a point. "K-means clustering" is an algorithm which partitions data points into discrete clusters based on mean-distance calculations [13]. The k-means algorithm randomly assigns k centroids and iteratively updates the centers and the points assignments in a way that minimizes the sum of the distances between the points and their corresponding centroid. In this paper, K-means clustering is used to determine the unsupervised entropy thresholds.

III. Experiments

The experimental setup involves a variety of data synthesis and detection experiments evaluated for accuracy.

A. Setup

For optimized performance, we synthesized and trained the classifier in C++ and visualized the data with MATLAB and Python. Detection and writing data files are implemented with Python. We divided the experiments into two parts: synthesis of data to train and execute the detection algorithm, and detection methodologies.

In this experimental setup, we consider two boats: u, the malicious boat, and boat v, which might be chased or followed by u. To ensure that boat u catches v at some point in time, boat u is capable of having a higher speed than boat v. We assume that the boats travel in a discrete coordinate system.

B. Data Synthesis Experiments

To train the classifier, we synthesize data for three categories.

Random Walk (Static Simulation): For each move of boat u, the simulation randomly selects one of the nine possible moves representing the current coordinate of boat u and the eight nearest neighbors of the current coordinate of boat u. For example, the cardinal direction East - and the non-cardinal direction Northeast - represent possible nearest neighbor points. Since the simulation is static, v does not move. We regard this as a trivial case for the remainder of the paper.

Dynamic and Static Chasing: In the static chasing simulation, boat v is static, and boat u moves according to the Bresenham algorithm towards boat v. In the dynamic chasing simulation, boat v moves East by one discrete unit every m time steps, where m is the speed parameter for boat v. Boat u moves every time step according to the Bresenham algorithm.

Following: Boat u ensures that it is always behind boat v while randomly choosing one of three movement types: zig-zag, chasing, and random-walk at random intervals. Boat v moves in a random predetermined cardinal direction.

C. Detection Experiments

After training the classifier with synthetic data, we utilize the following methods to find q means, denoted as μ_i, to characterize a given data set. From these means, we determine thresholds, finalizing the training process.

1) Finding Means: We find μ_i in both supervised and unsupervised methods.

Supervised Means: This is a method of calculating μ_i that best represents each trajectory type. Knowing the trajectory type of a given entropy data, we find the entropy mean of the points of that trajectory type. Given q trajectory types to analyze, we obtain q means for each kind of trajectory.

Unsupervised K-means Clustering: With k-means, we find representative centroids of the data for a preset parameter- k, the number of clusters. In this context, $k = q$ to find q means that best represent the entropy data.

2) Finding Thresholds: Next, we find $q - 1$ thresholds, which allows using simple inequality relations to classify a given trajectory, to finalize the training procedure for the classifier. We consider the following two methods:

Voronoi Diagram: Voronoi diagrams can be used to partition a multidimensional space for a given data set [14]. In our experiments, the Voronoi diagrams are used to partition 1-D space, which is entropy, using the midpoints between the means as follows:

$$T_i = \frac{\mu_i + \mu_{i+1}}{2} \tag{2}$$

Weighted Means: To find the thresholds/boundaries of two entropy data clusters we use the mean values, μ_i, of the entropy thresholds, and their standard deviations, σ_i, we find the entropy boundaries, T_i, between consecutive data clusters using the equation:

$$T_i = \mu_i + \frac{\sigma_i(\mu_{i+1} - \mu_i)}{\sigma_i + \sigma_{i+1}}, \quad 1 \leq i \leq q-1 \tag{3}$$

With two ways to find means and two ways to calculate T_i, we analyze and compare a total of four methods to calculate T_i.

Detection Procedures: In this experiment, the four methods of detection described above are employed.

1) Supervised Weighted: we train the detection algorithm with supervised means and weighted thresholds.
2) Supervised Voronoi: we train the detection algorithm with supervised means and thresholds found with the Voronoi diagrams.
3) Unsupervised Weighted: we train with k-means and weighted thresholds.
4) Unsupervised Voronoi: we train with k-means and thresholds found with the Voronoi diagrams.

IV. Technique

A. Synthesis

Random walk: The random walk simulation gives equal probability to all of the nine possible next locations. At the beginning of the simulation, boat u starts at $(1790, 1790)$ and boat v is static at $(2047, 2047)$. For every move of boat u, a pseudo-random number between 1 and 9 is generated. These nine numbers correspond to one of nine possible next locations. Boat u and boat v cannot move outside the bounds $(4096, 4096)$. The simulation ends either when boat u reaches boat v or after a set number of time steps.

Chasing: Both the static and dynamic variants utilize the Bresenham algorithm. For each move of boat u, the Bresenham algorithm uses the positions of both boats to calculate the approximate line between the two boat positions, and the first 20 points of the line constitute the next position taken by boat u.

Following: At the beginning of the simulation of following, boat u starts at $(2047, 1790)$, and boat v starts at $(2047, 2047)$. Boat v is preset to move one unit East. The algorithm for boat u's movement first determines whether or not boat u is behind boat v. This is done by drawing a line through boat v perpendicular to its last move. Checking in which side of the line boat u's position is, determines the next movement type. If boat u is not behind boat v, then boat u performs a Bresenham circle-drawing algorithm-based circling maneuver to travel behind boat v. The circling algorithm chooses the compass direction that takes boat u to a new position where the change in the distance between boat u and boat v is minimized. When boat u is behind boat v, a 3-state disguised chasing algorithm for boat u is performed, in which boat u alternates between zig-zag, chasing, or random walk patterns. If boat u is behind boat v, then boat u continues with the following procedure. The simulation of following ends once boat u reaches boat v.

B. Detection

The first step of the detection algorithm is finding the entropy of a given trajectory. To find the entropy mean of each movement, the detection algorithm is trained with a synthetic data set, which contains a large number of synthesized trajectories. Given a trajectory of size n and history buffer H_b of size b, we first calculate the empirical probabilities for each move. Next, we find the entropy of H by employing equation (1) we use $P'(i)$ to denote that this is an estimated probability. The calculation of entropy relies on the number of time steps in a given trajectory, where a time step is the time interval for which the simulation progresses to the next step.

When running the detection, the number of time steps, t, in a given trajectory should be minimized while maintaining high accuracy of the given detection, since the goal is to detect boat activities early in the trajectory.

V. Experiments' Results

A. Data Synthesis Experiments

Random Walk: In this experiment, we have implemented the following procedure:

1) Generate a pseudo-random integer $n \in [1, 9]$.
2) Choose the next movement according to the procedure for random walk described in section IV part A.
3) Repeat steps 1-2 until boat u catches up to boat v or until the designated number of time steps.

Figure 1 depicts one instance of random walk.

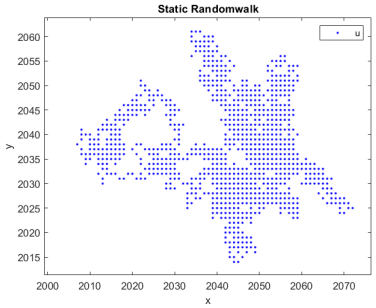

Figure 1. Static random walk, boat u is represented by the blue dots.

From Figure 1, we can observe that random walk behaves as expected and resembles random walk reported in the literature [15].

Chasing: In this experiment we have implemented the traditional Bresenham line drawing algorithm as described in section B. Figure 2 depicts one instance of static chasing, and Figure 3 depicts one instance of dynamic chasing.

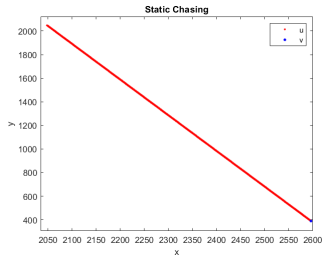

Figure 2. Static chasing: boat u, represented by the red line, starts at the top and goes towards boat v, which is static and represented by the black dot.

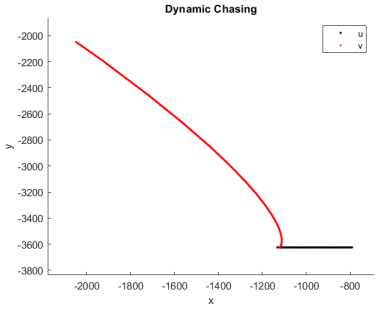

Figure 3. Dynamic chasing: boat u, represented by the red line, starts at the top and goes towards boat v, which is represented by the black line. Boat v starts from the right and moves left.

From Figure 2, we can observe that the simulation follows the Bresenham algorithm. Since boat v is static, the chaser (boat u) moves in a straight line towards boat v. A human sitting on boat v would most likely observe that v is being chased.

From Figure 3, we can observe that boat u follows a constantly updated Bresenham line drawing algorithm. Since boat v is moving, the lines drawn are updated in each time step, and only the first Bresenham step is taken. When the position of boat v changes, the algorithm redraws the line according to the Bresenham algorithm.

Following: In this experiment, we have implemented the following procedure:

1) Verify that boat u is behind boat v (as described in section A above).
2) When boat u is behind boat v, follow the 3-State Disguised Chasing algorithm.
3) The simulation ends when boat u catches boat v or when the designated number of time steps has been reached.

Figure 4 depicts one instance of the pattern of "following".

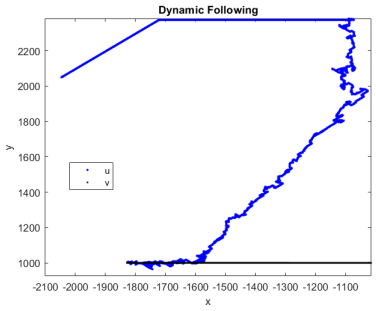

Figure 4. Dynamic following with boat v traveling right-to-left along the black segment and boat u traveling along the blue segment starting from the leftmost point.

From Figure 4, we can observe that boat u starts in front of boat v and performs the Bresenham circle drawing algorithm in order to get behind boat v. After boat u is behind boat v, it performs a 3-state procedure, where the trajectory alternates between Bresenham, random, walk, and zig-zag.

B. Detection Experiments

Supervised Means: Given a trajectory of size n and history buffer of size b, we:

1) Estimate the probabilities of each of the nine moves.
2) Calculate the entropy using a history buffer of size b.
3) Evaluate the average of $n-b$ buffer entropy data points to get the entropy value for the trajectory.
4) Using these means, identify thresholds as per equations (2) or (3).

Unsupervised Means: To find unsupervised means we:

1) Estimate the probabilities of each move.
2) Calculate the entropy using a history buffer of size b.
3) Employ k-means clustering to assign q centroids to the data set.
4) Using these means, identify thresholds as per equations (2) or (3).

Table 1 provides the means and centroids obtained by running the detection algorithm on a set of 200 synthetic trajectories.

Table I
MEANS AND CENTROIDS OBTAINED BY RUNNING THE DETECTION ALGORITHM ON A SET OF 200 SYNTHETIC TRAJECTORIES.

Parameter	*Random Walk*	*Chasing*	*Following*
Supervised Means	3.0629	1.0188	2.5743
K-means Centroids	3.0321	1.0195	2.5999
Standard Deviation	0.1548	0.4539	0.0058

From Table 1 we can see that the k-means centroids are relatively close to the supervised mean values, presumably due to the averaging nature of k-means. Because of this similarity, supervised and unsupervised mean values (and therefore the thresholds) are similar, signifying that, in this case, the unsupervised algorithm is almost as robust as

the supervised algorithm. Next, we find thresholds using equations (2) and (3). Table 2 provides the resultant threshold values:

Table II
THRESHOLD VALUES

	Supervised Weighted	*Supervised Voronoi*	*K-means Weighted*	*K-means Voronoi*
T_1	0	0	0	0
T_2	1.414	1.797	1.421	1.810
T_3	3.057	2.819	3.027	2.816

From Table 2, we can observe that the thresholds obtained through the different methods are quite similar.

Finally, we obtain the thresholds for the four detection experiments: supervised weighted, supervised Voronoi, unsupervised (k-means) weighted, and unsupervised Voronoi thresholds. These, as well as detection accuracy, are shown in Tables III-VI.

Detection Procedures

We run detection across the array P_i = $[0.1, 0.3, 0.5]$, where P_i is equal to the proportion of time-steps T used to classify. In this experiment and in all the others, $T = 3000$ time-steps. Thus, P_i equals 300, 900, or 1500 time-steps. For the classification, we use a new set (different than the training set) of trajectories with varying boat v speeds, positions, and trajectories.

The following paragraph provides details of the detection procedure.

1) Once the detection algorithm has been trained (by using either supervised or unsupervised means) and entropy thresholds have been found (by either using weighted or Voronoi), the algorithm calculates the entropy H_i of trajectory i via equation (1). This calculation is based on P_i .
2) The detection algorithm classifies boat trajectories by comparing H_i to the entropy thresholds. After calculating the entropy of a given boat trajectory, the algorithm finds the largest threshold that is less than H_i, which yields one of the three possible boat activities:
   a) If $H_i < T_1$, then the given trajectory is chasing.
   b) If $T_1 < H_i < T_2$, then the given trajectory is following.
   c) If $T_2 < H_i$, the the given trajectory is random walk.

Table III
SUPERVISED WEIGHTED THRESHOLD ACCURACY

300 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	123	77	61.5%
Following	160	40	80.0%
Random-walk	136	64	68.0%
900 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	129	71	64.5%
Following	137	63	68.5%
Random-walk	177	23	88.5%
1500 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	137	63	65.8%
Following	141	59	70.5%
Random-walk	196	4	98.0%

Table IV
SUPERVISED VORONOI THRESHOLD ACCURACY

300 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	200	0	100%
Following	92	108	46.0%
Random-walk	200	0	100%
900 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	200	0	100%
Following	83	117	41.5%
Random-walk	200	0	100%
1500 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	200	0	100%
Following	96	104	48.0%
Random-walk	200	0	100%

Table V
UNSUPERVISED WEIGHTED THRESHOLD ACCURACY

300 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	124	76	62.0%
Following	155	45	77.5%
Random-walk	172	28	86.0%
900 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	140	70	65.0%
Following	134	66	67.0%
Random-walk	199	1	99.5%
1500 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	138	62	69.0%
Following	139	61	69.5%
Random-walk	200	0	100%

Table VI
UNSUPERVISED VORONOI THRESHOLD ACCURACY

300 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	200	0	100%
Following	90	110	44.9%
Random-walk	200	0	100%
900 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	200	0	100%
Following	83	117	41.5%
Random-walk	200	0	100%
1500 steps	***True Positive***	***False Positive***	***Accuracy***
Chasing	200	0	100%
Following	95	105	47.5%
Random-walk	200	0	100%

VI. RESULTS EVALUATION

Based on Tables III to VI we can make the following observations.

- In general, the more time-steps used, the more accurately the classifier detects the trajectory.
  - We can see that this is not completely true when detecting following trajectories across all the four detection methods.
  - This is due to the difficulty in distinguishing between chasing and following as opposed to distinguishing between any of these two activities and random walk.
- Random walk has near 100% accuracy across the board.
- Detecting a following trajectory has the highest accuracy at 80% for small time-steps with the supervised weighted thresholds.
- Both of the Voronoi-based methods have perfect chasing detection; it is often because the chasing threshold is conservative. Hence, we should evaluate the performance of our methodology based on the accuracies of following and chasing. Moreover, following and chasing are the trajectories of interest.
- Both of the weighted threshold methods detect chasing and following trajectories with an accuracy of about 70% of the time. Nevertheless, often the unsupervised threshold method is better.

VII. CONCLUSION & FURTHER RESEARCH

The reported experiments show that using entropy as a single classification feature has a high potential for accurate detection of random walk, chasing, and following maritime vessel activities. Hence, they can provide valuable information for maritime threat detection.

There are several potential improvements and other considerations to be made regarding this problem. For example, instead of using a 1-D analysis with entropy, it is possible to extend to higher dimensions using other relevant factors. In addition, it would be beneficial to train with both synthetic data and real data from maritime activities and develop a robust, practical detection algorithm.

ACKNOWLEDGEMENTS

This material is based in part upon work supported by the National Science Foundation under Grant Nos. MRI20 CNS-2018611, MRI CNS-1920182, and DHS E2055778.

REFERENCES

[1] Martínez-Zarzoso, Inmaculada; Bensassi, Sami (2010). "How costly is modern maritime piracy for the international community?" IAI Discussion Papers, No. 208, Georg-August-Universität Göttingen, Ibero-America Institute for Economic Research (IAI), Göttingen

[2] Tesic J., Tamir D., Neumann S., Rishe N., Kandel A. (2020). "Computing with Words in Maritime Piracy and Attack Detection Systems." In: Schmorrow D., Fidopiastis C. (eds) Augmented Cognition. Human Cognition and Behavior. HCII 2020. Lecture Notes in Computer Science, vol 12197. Springer, Cham.

[3] Koopman, P. "Bresenham line-drawing algorithm." Forth Dimensions 8.6 (1987): 12-16.

[4] Obradović, Ines, Mario Miličević, and Krunoslav Žubrinić (2014). "Machine learning approaches to maritime anomaly detection." Naše more: znanstveni časopis za more i pomorstvo 61.5-6 (2014): 96-101.

[5] Jean Roy (2008). "Anomaly detection in the maritime domain," Proc. SPIE 6945, Optics and Photonics in Global Homeland Security IV, 69450W (15 April 2008); DOI: 10.1117/12.776230

[6] Goodarzi, Mojtaba, and Mahdi Shaabani (2019). "Maritime Traffic Anomaly Detection from Spatio-Temporal AIS Data." The Second International Conference on Management and Fuzzy Systems (ICMFS Series).

[7] Li, Xin (2018). "Collaborative intrusion detection method for marine distributed network." Journal of Coastal Research 83 (10083) (2018): 57-61.

[8] Shahir, Amir Yaghoubi, et al. "Mining vessel trajectories for illegal fishing detection." 2019 IEEE International Conference on Big Data (Big Data). IEEE.

[9] Fooladvandi, Farzad, et al (2009). "Signature-based activity detection based on Bayesian networks acquired from expert knowledge." 2009 12th International Conference on Information Fusion. IEEE.

[10] Dabrowski, Joel Janek, and Johan Pieter De Villiers (2015). "Maritime piracy situation modelling with dynamic Bayesian networks." Information fusion 23 (2015): 116-130.

[11] L. Rabiner and B. Juang, "An introduction to hidden Markov models," in IEEE ASSP Magazine, vol. 3, no. 1, pp. 4-16, Jan 1986, DOI: 10.1109/MASSP.1986.1165342.

[12] Sayood, Khalid (2000). Introduction to Data Compression. 3rd ed., Morgan Kaufmann Publishers.

[13] Hartigan, J. A., and M. A. Wong (1979). "Algorithm AS 136: A K-Means Clustering Algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, [Wiley, Royal Statistical Society], 1979, pp. 100–08, DOI: 10.2307/2346830.

[14] Imai, Masao Iri, and Kazuo Murota (1985). "Voronoi Diagram in the Laguerre Geometry and Its Applications." SIAM Journal on Computing 14:1, 93-105.

[15] Codling Edward A, Plank Michael J and Benhamou Simon (2008). "Random walk models in biology." J. R. Soc. Interface, 5813–834.