Catch me if you can

Venkateshwaran Thamilselvan ‡
University of Luxembourg
113-6, Avenue du swing
L-4367, Belvaux, Luxembourg
+352 661364956
venkateshwaran.thamilselvan.001@student.uni.lu

Barbara Koduzi §
University of Luxembourg
Maison Kazerech,
7a Rue De Reckange, L-3943
+352 691222274
barbara.koduzi.001@student.uni.lu

Vanitha Vardharajan ‡
University of Luxembourg
8 Avenue , G.D.-Charlotte
L-3440 Dudelange, Luxembourg
+352 661364892
vanitha.varadharajan.001@student.uni.lu

Emiliana Pali §
University of Luxembourg
Maison Kazerech,
7a Rue De Reckange, L-3943
+355 691222272
emiliana.pali.001@student.uni.lu

ABSTRACT
The development of Unmanned Aircraft Vehicles, commonly known as UAVs or Drones have impacted our lives very much. Particularly in the recent times of COVID-19, it has been used in many surveillance applications. Moreover, UAVs are used across the world for commercial, civilian as well as military applications and government owned projects. Deciding upon the mobility of the UAVs have been a big issue in most of the applications. Here we consider upon the problem where we have two types of UAVs, one called defenders and the other intruders. The motive of the defender is to catch the intruder and of the intruder is to escape from the defender. The main objective of the defender is to minimize the catching time of intruders whereas intruder has to maximize the catching time. In this paper, we have developed and analyzed the mobility algorithms for both defender and intruder to move optimally in a confined space.

Categories and Subject Descriptors
G.1.6 - [**Mathematics of Computing**]: Optimization - *Unconstrained optimization*.

General Terms
Algorithms, performance, theory

Keywords
UAV's mobility, defender and intruder mobility, A and B's optimised algorithm

INTRODUCTION
As the importance of UAVs has been increasing nowadays, several new strategies or developments need to be integrated with it in order to increase the performance. To implement our problem statement, different strategies have been proposed for the defender (aka. 'A') to catch the intruder (aka. 'B') as well as for the intruder to escape from the defender. For that purpose, different zones or regions have been introduced surrounding A and B. In each regions, A and B performs different motions in order to increase its performance. According to that, there are four different algorithms for Defender and two different algorithms for Intruder. So, we have performed some statistical tests and finally we select the algorithm which has the lowest upper limit of p99 value as a best algorithm for A as well as for B for a single initial location. In the same way, for 100 initial locations we will have 100 best algorithms for A and B. The algorithm which has a higher vote wins according to the majority voting rule. For the purpose of simulation, we have to discretize time, δ_{time} is the discretization quantum of time.

1. OPTIMISATION OF DEFENDER'S MOBILITY
First, we will try to optimize the strategies of the A by keeping the type of motion B can perform as constant. By doing so, we will get the optimized strategy for A with which we can be confident that we can catch B within a specific upper limit of catching time. Here we also assume that both A and B has same capability such as velocity and acceleration which we will discuss further.

---

‡ Venkateshwaran Thamilselvan and Vanitha Varadharajan worked on mobility algorithms for Defenders

§ Barbara Koduzi and Emiliana Pali worked on mobility algorithms for Intruders

1.1 Velocity decomposition

For our convenience, we decompose the velocity to the corresponding three axes velocities.

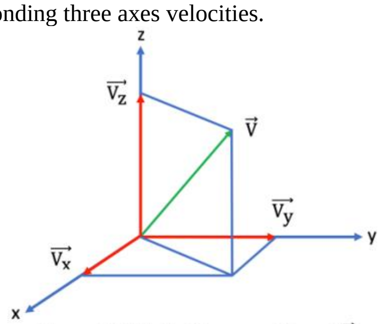

Figure 1. Velocity Decomposition of $\vec{V}$

$$\vec{V} = \vec{V_x} + \vec{V_y} + \vec{V_z} \tag{1}$$

Here the velocity of A is obtained by vector addition of the individual velocity component.

$$V = \sqrt{|\vec{V_x}|^2 + |\vec{V_y}|^2 + |\vec{V_z}|^2} \tag{2}$$

The magnitude of the velocity can be obtained from (2).

1.2 Constraint walls

We simulate inside a constrained environment, so we have to define the x, y and z axes constraints. For e.g. if we define the x, y and z constraints to be 100, 100 and 100, it means that our world is constrained to this 3D space and the A and B are never allowed to move outside this space in any scenario. Collision handling is explained in section 1.4.

1.3 Modular approach for defender

We assume that A has several types of sensors and each sensor are capable of sensing different information about the location of B. This capability to sense depends upon the distance between A and B.

1.3.1 Regions surrounding A

According to the sensing capabilities, we split the regions surrounding A to three categories, Line of Field View, Line of Sight and Line of Control.

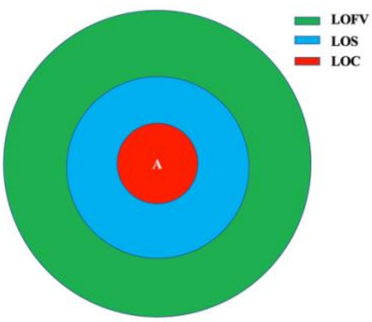

Figure 2. Regions surrounding Defender

Apart from these three regions, there is another region starting from outside of LOFV to the constraint wall. In this region, A can't detect any information about B.

1.3.1.1 Basic Assumptions on sensors

We have three main assumptions upon which we mainly rely on, they are-

i. When B is inside the LOFV, we assume that A can deduce the distance between A and B, and an angle θ which forms a spherical sector within which the B could possibly reside
ii. When B is inside the LOS, we assume that A can deduce the exact location of B
iii. When B enters inside LOC, A can catch B

1.3.1.2 Line of Field View (LOFV)

The d_{LOFV} is the LOFV distance, and this forms a sphere around A with d_{LOFV} as radius. This region between LOFV and LOS is called Field View region. Say we configure the d_{LOFV} as 30m, then if distance between A and B is reduced below 30m, according to our basic assumption (i), A can deduce both the distance and the angle θ.

1.3.1.3 Line of Sight (LOS)

The d_{LOS} is the LOS distance, and this forms a sphere around A with d_{LOS} as radius. This region between LOS and LOC is called clear sight region. If we fix the LOS to be around 10m and then distance between A and B is reduced below 10m, it means that B falls inside the LOS region. According to our basic assumption (ii), A can deduce the exact location of B.

1.3.1.4 Line of Control (LOC)

The d_{LOC} is the LOC distance, and this forms a sphere around A with d_{LOC} as radius. This region of sphere is called control region. Practically speaking, the LOC should be around 2m. When distance between A and B is reduced below 2m, A can announce that it has caught B and it records its catching time.

1.3.2 Strategies of defender

A has to decide upon its mobility based upon the data it has about B. We will look in detail how we can decide upon that.

1.3.2.1 Strategies when B is outside the LOFV

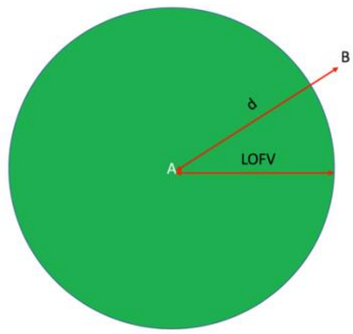

Figure 3. B is outside LOFV of A

Here, the distance between A and B is d.
When d > d_{LOFV}, we A cannot deduce any information about B. So, in this case we propose two types of mobility for A. They are Uniform motion and Random motion.

1.3.2.1.1 Random motion

In this type of motion, the new velocity is computed as the random permutation of the previous velocity. Let's say the original velocity is

$$v_{init} = [|\vec{Vx}|, |\vec{Vy}|, |\vec{Vz}|] \tag{3}$$

The new velocity is computed as

$$v_{new} = \text{random\_permutation(velocity)} \tag{4}$$

Here it is important to note that the magnitude of velocity doesn't change.

$$|v_{init}| = |v_{new}| \tag{5}$$

Let say the current position is (x_1, y_1, z_1). The new position (x_2, y_2, z_2) is given by

$$(x_2, y_2, z_2) = (x_1, y_1, z_1) + \delta_{time} * v_{new} \tag{6}$$

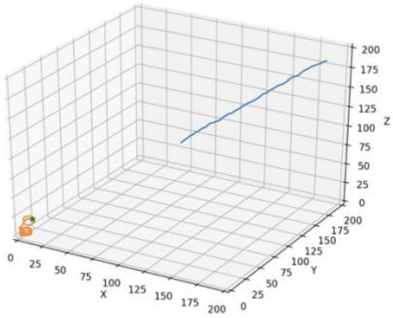

Figure 4. Random motion

In Figure 4., the blue color path represents the random motion performed by A.

1.3.2.1.2 *Uniform motion*

In this type of motion, A moves linearly according to its original velocity.

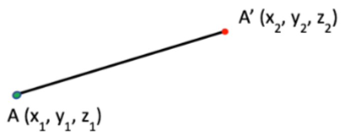

Figure 5. Next target location of A in uniform motion

If (x_1, y_1, z_1) is the initial location, the target location (x_2, y_2, z_2) is given by

$$(x_2, y_2, z_2) = (x_1, y_1, z_1) + \delta_{time} * v_{init} \tag{7}$$

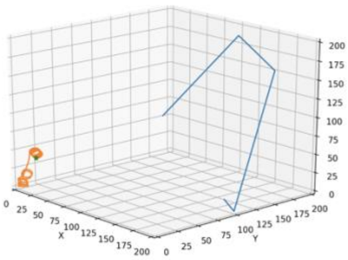

Figure 6. Uniform Motion

In Figure 6., A moves in uniform motion and when it collides with the walls it bounces back, we will see how we can handle collision in the later section.

1.3.2.2 *Strategies when B is inside the LOFV*

When B is inside the LOFV, according to our assumption (i), A knows the distance between A and B, and an angle θ, which forms a spherical sector around B, within which A knows that B resides. When this is the case, it is smart to move towards the feasible region of B. Nevertheless, moving randomly is always an option. So, we have proposed two types of motion, Field view motion and the previously seen random motion.

1.3.2.2.1 *Field view motion*

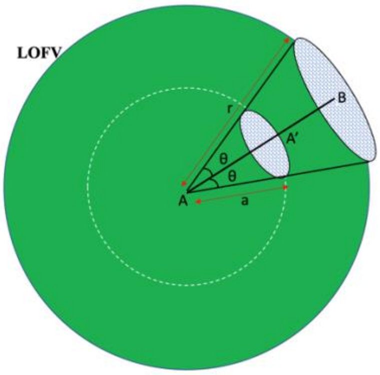

Figure 7. Field View Motion (FVM)

Here the distance between A and B is r.
When r < d_{LOFV}, A knows

i. Distance between A and B, i.e. r
ii. The field view angle θ

These two forms the spherical sector upon which B may reside, this region is shown in Figure 7. on the LOFV sphere.

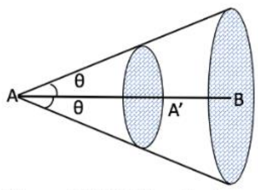

Figure 8. Field View Region

In this case A should move to anywhere in the inner shaded region shown in Figure 8. So, to compute the next target location A' we follow the below algorithm. This solution is inspired from [4]

Algorithm 1	Pseudocode for computing new velocity when B is inside LOFV
1.	$\theta \leftarrow 20°$
2.	$\vec{V_z} \leftarrow Random(-1, 1)$
3.	$a \leftarrow \lvert v_{init} \rvert$
4.	$\vec{V}_{x1} \leftarrow \sqrt{a^2 - \lvert\vec{V_z}\rvert^2} \cos\theta$
5.	$\vec{V}_{y1} \leftarrow \sqrt{a^2 - \lvert\vec{V_z}\rvert^2} \sin\theta$
6.	$\vec{V}_{z1} \leftarrow \vec{V_z}$

Here, when θ increases the total possible surface A can move will increase. Thus, the effectiveness of catching decreases. The maximum angle θ can attain is 180°, at maximum angle, the surface which can move is the whole sphere and thus here the movement of A will be random at maximum field view angle. Also, when θ is 0°, the A knows that the exact location of B and moves towards the B according to its velocity.

1.3.2.3 *Strategy when B is inside LOS*

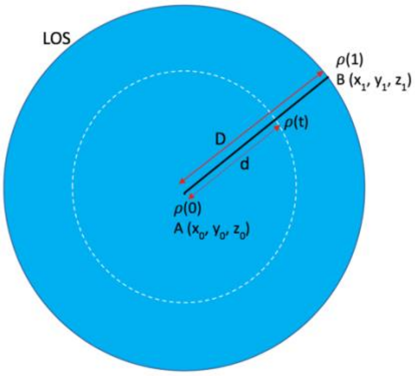

Figure 9. Line Of Sight

Here in Figure 9., D is the distance between A and B. When D < d_{LOS}, A knows the exact location of B according to our assumption (ii). The next target location in this situation can be computed from the below formula. Correctness of this formula can be seen in [13]

$$\rho\left(+\frac{d}{D}\right) = \left\{\left(1-\frac{d}{D}\right)(x_0, y_0, z_0) + \frac{d}{D}(x_1, y_1, z_1)\right\} \quad (8)$$

here d is given by

$$d = \delta_{time} * v_{init} \quad (9)$$

1.4 Collision handling with walls

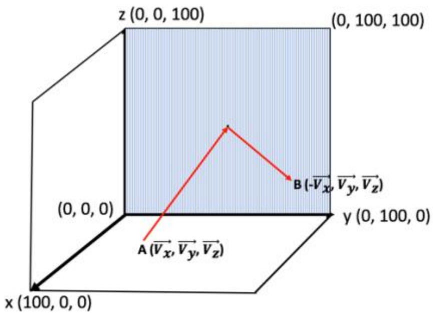

Figure 10. Collision Handling with walls

When A collides with YZ plane, it's because of the velocity component $\vec{V}_x$. So, we reverse the direction of $\vec{V}_x$ to avoid collision and to move A away from the walls. In the same way, if collision happens in XY plane, we reverse direction of $\vec{V}_z$ and if collision happens in XZ plane, we reverse the direction of $\vec{V}_y$.

1.5 Proposed possible-strategies of A

i. When B is outside LOFV, we have proposed two types of motion for A. They are-
   - Uniform motion
   - Random motion
ii. When B is inside LOFV, we have proposed two types of motion for A. They are-
   - Field view motion
   - Random motion
iii. When B is inside LOS, we have only one type of deterministic motion.
iv. When B enters LOC, A announces it has caught B and stops simulation.

From the above proposed motions, we can create 4 combinations of algorithm. They are-

i. UM_FVM
   - r_motion = UniformMotion
   - f_motion = FieldViewMotion
ii. UM_RM
   - r_motion = UniformMotion
   - f_motion = FieldViewMotion
iii. RM_FVM
   - r_motion = UniformMotion
   - f_motion = FieldViewMotion
iv. RM_RM
   - r_motion = UniformMotion
   - f_motion = FieldViewMotion

Here r_motion represents motion outside LOFV and f_motion represents motion inside LOFV.

1.6 Statistical test to choose the best strategy

1.6.1 *Test Procedure*

i. First, we take 100 random initial locations for both A and B inside our constraints
ii. For each initial location, we run the simulation 100 times for all the 4 algorithms independently and compute the catching time for each
iii. So now we have 100 catching time for each algorithm separately
iv. Let's say catching_time_1, catching_time_2, catching_time_3, catching_time_4 are the array of 100 catching times for algorithm A1, A2, A3, A4 respectively (for one initial location of A and B)
v. Then we compute the mean, standard deviation, variation, probability function and plot the distribution curve to analyze it
vi. The algorithm which has the lowest upper limit value of the p99 value is chosen as the best algorithm for one initial location
vii. We will have 100 best algorithms for 100 different initial location
viii. Finally, we select the algorithm with maximum votes as the best algorithm for A

1.6.2 *P99 value*

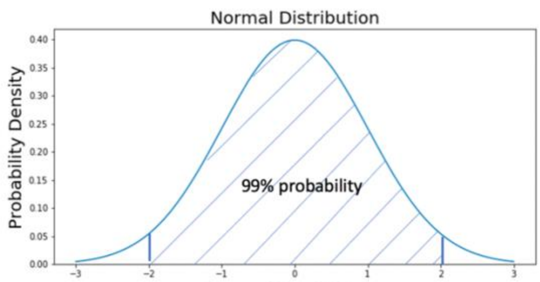

Figure 11. p99 Value

It is the range of catching times in which the area under the probability density function is 99% as that of the whole area. In other words, it the range of catching times in which we can catch B with 99% confidence.

1.6.3 *Experimental result*

Figure 12. shows that the algorithm UM_FVM has a significant difference when compared to others.

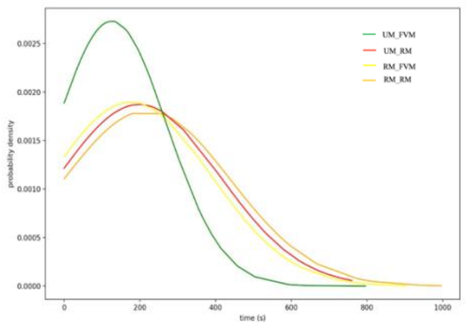

Figure 12. Obtained probability density

But we cannot decide that UM_FVM is the overall best algorithm with this result. So, we run the test for 100 random initial location of both A and B. We then get the 100 best algorithms corresponding to each initial location. The bar chart in figure 13. represents the vote count corresponding to each algorithm.

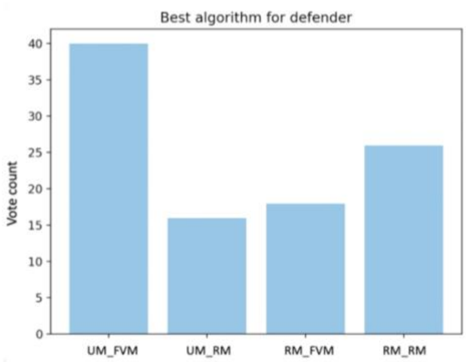

Figure 13. Barchart depicting the best possible best-algorithm for A

So, with 99% confidence we say that the algorithm UM_FVM is the best algorithm for A.

2. OPTIMIZATION OF INTRUDER'S MOBILITY

In the real world, for every predator, there is prey, for every defender, there is an intruder, for catching strategies there are escaping strategies. This kind of interaction is ubiquitous in nature and this is what we are trying to simulate.

In this part of our paper, we are going to talk about object B which is considered to be the intruder and explain how he maneuvers to "run away" from object A, the defender. Escaping is a complex interaction involving the sensory abilities, decision-making, and behavior of the intruder. Continuously monitoring its position in space relative to a goal, is the most essential task for object B that moves through the simulated environment.

2.1 Regions surrounding B

The model that we develop concerning the intruder, includes his responses to a threat from the defender, either when he starts moving, when he is in the mid of chase or totally in danger.

Based on the ability to sense danger coming from A being nearby, we define three different areas which conclude on three different cases that need to be taken care of. For each zone, our object changes its color into green, blue, red respectively for Safe Zone, Chase Zone and Danger Zone.

2.1.1 Strategies in Safe Zone

As long as the intruder does not have any trace of the defender, he is able to move freely. Even though in this scenario, there is no possibility of him being in danger, his movements should be in a way such that B can avoid any chance of being caught.

This zone is called Safe Zone and in this phase, B has to choose between two different kinds of motions. The first one is a Random Motion and the second one is a Helix Motion.

2.1.1.1 Random Motion

Even in real-life scenarios, it might happen for animals to move randomly in space with no specific intention. In our case, this idea consists of calculating a new random vector, which is an angle away from the current velocity vector. This will be the direction of the new acceleration vector.

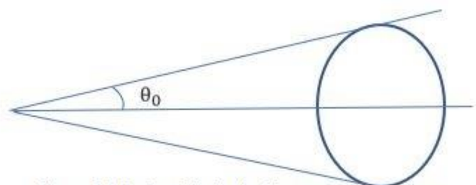

Figure 14. Random Motion inside cone.

To generate a random, uniformly distributed normalized vector, we follow the steps given below.

- Initially, we find a vector normal to the cone axis vector (by crossing the cone axis vector with the cardinal axis that corresponds with the cone axis vector component nearest to zero).
- Then, we find a second normal vector using a cross product.
- After this, we generate a random angle between [θ_{min},θ_{max}], where these θ_{min},θ_{max} are the angles between new acceleration and current velocity vector.
- Rotation uses the two normal vectors as a 2D coordinate system to create a new vector at the angle previously generated.
  So, now we have two mutually orthogonal unit vectors u, v. Both of them are orthogonal to the given axis a of the cone, where |a|=1. The random unit vector within the cone will be a vector x of the form:
  $$x = \sin\theta\ (\cos\phi u + \sin\phi v) + \cos\theta a$$
- Now we want the vectors x to be equidistributed on the spherical cap given by $0 \le \theta \le \theta_0$, where "equidistribution" refers to the area measure on the sphere S_2.
  This observation boils down to the following recipe: We choose z uniformly distributed on the interval [$\cos\theta_0$,1] and put $\theta := \arccos(z)$. Furthermore, let ϕ be uniformly distributed on [0,2π]. Then the point
  $$x = \sin\theta\ (\cos\phi u + \sin\phi v) + \cos\theta a$$
  will be uniformly distributed on the spherical cap.

2.1.1.2 Helix motion

In real-life situations, some animals use specific strategies to fit with the constraints of this world and also to prevent themselves and decrease the chances of being in hazardous situations.

During the time that B is in the safe zone, he does not have information about the whereabouts of A, and in this point, even if he has not other senses or is blind he can be secure using the Helix motion.

A unique characteristic of locomotion in organisms starting with Paramecium and evolving to other species including humans is the fact that in specific situations they move in spiral paths. R. Wicherman, in his book "The Biology of Paramecium", provides an explanation of the reasons that push these creatures to move in this kind of trajectory. He, himself, served as a subject for this experiment, recording his movement while blindfolded. He was directed to walk straight ahead toward a distant marker. The other senses were non-functioning, no noise, no wind, no scent. On completion of the experiment, an examination of the chart showed that he began walking in large spirals that became smaller and smaller circles. Other blindfolded subjects, all walked in spirals or series of circles, this for safety reasons. Based on the above assumptions, we created the algorithm for the intruder to move forward while following a circular path, with a fixed radius, around a center. By making this move, B has a clear view, causing him to be able to notice everything surrounding him but now in a larger diapason compared to a straight-forward movement.

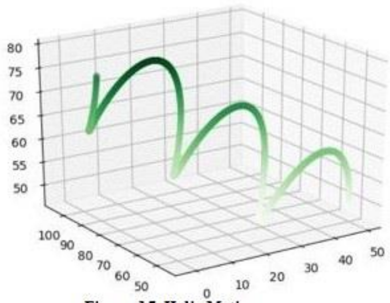

Figure 15. Helix Motion

2.1.2 Strategies in Chase Zone

In the paper "Principles of insect path integration", it is said that many arthropods, including a wide variety of insects, are known to use path integration strategy to return to their nest, hive, or burrow by the shortest possible route convoluted foraging trips. Path integration is a behavior of animals to regularly update their position relative to the point of departure, by measuring traveling speed and direction. This strategy is critical for survival since it reduces predatory pressure and allows it to minimize exposure to them. In principle, during path integration, animals continuously keep track of the distance and directions traveled.

Maneuverability can also be important for escaping from predators by reducing the predictability of the trajectories of movement. For instance, animals may choose to maximize either speed or maneuverability under different conditions to take advantage of the unpredictability of movement. In a study of leaf-cutter ants (*Atta sexdens*), Angilletta et al. (2008) found that individuals ran away from threats using straighter, more-predictable trajectories when they were faster, but when they were slower, they ran away using less-predictable trajectories.

Hence, for our model, there is a specific radius, inside which B can notice the exact position of A. This is called the Chase Zone. By having knowledge of the defender's coordinates, the intruder is capable of calculating the distance between them and the exact velocity that A has in that specific moment.

Since our space is 3D, we use Euclidian Formula as the metric to calculate the distance:

$$\mathbf{D_{A-B}} = \sqrt{(\boldsymbol{X_B} - \boldsymbol{X_A})^2 + (\boldsymbol{Y_B} - \boldsymbol{Y_A})^2 + (\boldsymbol{Z_B} - \boldsymbol{Z_A})^2}$$

For the velocity we used the following formula:

$$d = \{X = X_t - X_{t-k};\ Y_t - Y_{t-k};\ Z = Z_t - Z_{t-k}\}$$

where d is the distance travelled by A during the interval [t-k, t]

$$\mathbf{V} = \sqrt{\mathbf{X^2} + \mathbf{Y^2} + \boldsymbol{Z^2}}$$

Here the motion performed by the intruder is a uniform motion with a constant acceleration. B has the skills to predict the direction that he should follow to go further away from A by just comparing their coordinates. (e.g. $X_A < X_B$ & $Y_A > Y_B$ & $Z_A > Z_B$, so B chooses to move pursuing this X_B direction).

With all the information that B holds, he can increase his own velocity but not beyond a given maximum. Therefore, the speed of movement, even during extreme situations like escaping the defender, should be based on a compromise between a range of factors. Speed choice is a complex trait constrained by energetic costs, a decrease of the control and accuracy of movement.

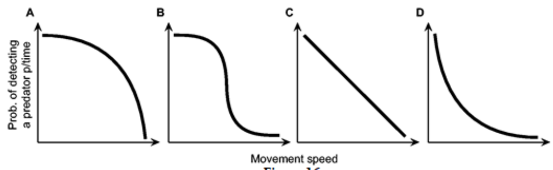

Figure 16

In figure 16, the function describing the relationship between intruder's speed and his probability of detecting a predator per unit time (p/time) will have implications for optimal speeds. Possible functions could show: (**A**) Individuals will have a high probability of spotting danger unless they are traveling at high speeds, (**B**) there is a threshold speed after which intruder is unlikely to detect a threat, (**C**) a linear decrease in the probability of detecting the defender, and (**D**) a rapid decrease in the probability of detecting a threat, even at low speeds of movement.

2.1.3 Strategies in Danger Zone

Danger Zone is the area where the probability of B being caught is way higher than the two other zones explained above. Since the risk is bigger, B has to find a new tactic in order to completely lose A from his sight. To develop this strategy, the whole idea is based on the paper "How moths escape bats: predicting outcomes of predator-prey interactions". This model assumes the intruder reacts immediately with a turn of its own and survives the encounter.

Turning when running requires an animal to change the main vector of motion and rotate its body to the new orientation (*Jindrich and Full 1999*), meaning that a turning animal must overcome its inertia and undergo angular motion (*Zollikofer 1994*). To do this, it must produce greater stabilizing forces when turning than when running in a straight line. An animal running at a particular speed (v) around a curve of radius r can only continue forward motion as it changes direction if the coefficient of friction of its feet with the ground is at least $v2 / rg$ (*Alexander 1982*). This means that an animal needs a larger coefficient of friction to turn at higher speeds or sharper angles and implies that speed should compromise turning ability.

When A expects less, suddenly B performs a deep turn. After that, depending on the new distance from A, if he still is very close to the defender, he follows the same strategy. Otherwise, if he is not in the Danger Zone but still can see A, he continues the strategy explained for the Chase Zone. If not, he is in the Safe Zone and can move peacefully.

2.2 Experimental approach

Based on the two different ways that B moves in the Safe Zone, we created two different algorithms denoted HelixB and RandomB. They differ only from the motion in the Safe Zone, whilst the motion of intruder is the two other zones remains the same. So, what we are going to do next, is deciding which algorithm is more efficient for B to follow.

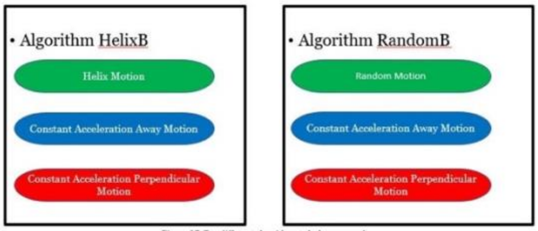

Figure 17. Two different algorithms to be be compared

2.2.1 Statistical test

- We take into account 100 random initial locations.
- We run the simulation 100 times, for the two algorithms independently.
- Then we compute the Standard Deviation, Mean Catching Time, Variance Catching Time and p99 value as given in the table below:

Table 1.

	HelixB Algorithm	RandomB Algorithm
σ	167.277	115.288
μ	41.92	26.19
Var	27981.873	13291.393
P99 value	(37.611, 46.228)	(23.220, 29.159)

2.2.2 Experimental results

To better decide the best algorithm, we used the probability distribution graph. As given in fig.18 the orange line shows the HelixB Algorithm and the red one shows RandomB Algorithm. When one line reaches the 0 value, it means that B is caught. From the graph, we notice that the catching time is significantly bigger when B follows HelixB Algorithm.

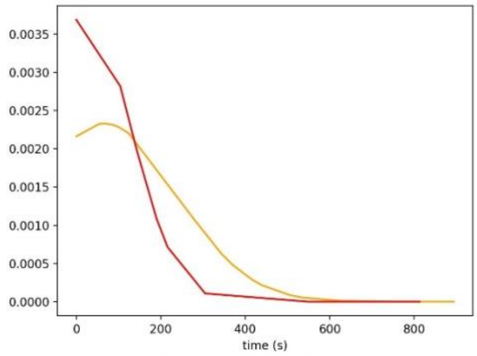

Figure 18. Probability distribution graph

The bar chart in fig.19 represents the best algorithm out of 100 tests. In more than 60 executions, B is caught faster by A when using Random Motion, whilst in less than 40 it is caught slower by A using Helix Motion.

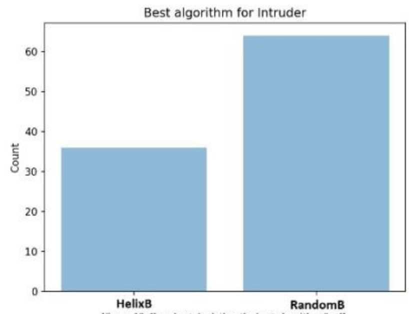

Figure 19. Bar chart depicting the best algorithm for B

From all the results that we got, the best algorithm for B to follow is HelixB as we want the catching time for B to be maximized.

3. CONCLUSION

The main goal of simulating this model of an object A trying to catch another object B, was to minimize the catching time when the problem is concerning A part and maximizing this time when the subject is B escaping from A. To do so we developed different strategies concluding in different algorithms for A and B specifically. Out of these algorithms, the most efficient ones were reached when B followed the HelixB Algorithm and A followed UM_FVM Algorithm (as explained above). This approach satisfied the main objectives, but there is still place for future improvements.

4. FUTURE WORK

What is left now, is defining the opportunity for this model to grow later on. This may include adding some more obstacles for our intruder and defender, giving them more sensors so they can be more unpredictable and specifying new strategies.

Some interesting points that can be taken into consideration are presented below:

- The algorithm can be optimized by removing the space constraints for nodes to move freely.
- The number of As and Bs can be increased.

- Defenders might chase only one specific B and create a strategy to pursue it or they just move around independently and catch the one that is closer or passes by.
- Bs can share information with each other, and they can distract A if any of the Bs is in danger.

5. REFERENCES

[1] Aaron J. Corcoran, William E. Conner, *How moths escape bats: predicting outcomes of predator-prey interactions.*

[2] Carlos De La Torre, Daniel Urda, El-Ghazali Talbi. *Optimization and Learning: Third International Conference*, OLA 2020.

[3] Ilkka Hanski, Otso Ovaskainen, *Modelling the movements of butterflies.*

[4] joriki (https://math.stackexchange.com/users/6622/joriki), Generate a random direction within a cone, URL (version: 2012-10-01): https://math.stackexchange.com/q/205589

[5] Keith T.Dillar, Laurence D. Picton, William J. Heitler ,*The Neuroethology of Predation and Escape.*

[6] Linda Blane, *Development of Psycho-motor Competence*.

[7] Mikko Kuussaari, Matias Saarinen, Eeva-Liisa Korpela, Juha Poyry, Terho Hyvonen, *Higher mobility of butterflies than moths connected to habitat suitability and body size in a release experiment*.

[8] R. Wichterman, *The Biology of Paramecium.*

[9] Robbie S. Wilson, Jerry F. Husak, Lewis G. Halsey, Christofer J. Clemente, Predicting the Movement Speeds of Animals in Natural Environments, *Integrative and Comparative Biology*, Volume 55, Issue 6, December 2015, Pages 1125–1141, https://doi.org/10.1093/icb/icv106

[10] Robbie S. Wilson, Jerry F. Husak, Lewis G. Halsey, J. Clemente, *Predicting the Movement Speeds of Animals in Natural Environments.*

[11] S. Ducatez, A. Humeau, M. Congretel, H. Freville, M. Baguette, *Butterfly species differing in mobility show different structures of dispersal-related syndromes in the same fragmented landscape.*

[12] Stanley Heinze, Ajay Narendra, Allen Cheung, *Principles of insect path integration.*

[13] steven gregory (https://math.stackexchange.com/users/75410/steven-gregory), Finding a point along a line a certain distance away from another point!, URL (version: 2019-08-12): https://math.stackexchange.com/q/3321112

[14] William E. Cooper, Jr. Daniel T. Blumstein. *Escaping From Predators: An Integrative View of Escape Decisions.*