A Modern Chase Algorithm

Piyush Kaushik, Shrisha Rao, *Member, IEEE*,

***Abstract*—Pursuit and evasion tends to be incorporated in human nature from a very long time with a very huge range of activities. Here, we are going to create an intelligent chase algorithm, which uses two basic approaches. After which we will use the newly created equations to simulate both approaches and provide graphical results. This analysis is based upon the fact that in modern days we can estimate the speed of a moving object and then chase it down depending upon its speed. We will also be taking the accuracy of such estimation in picture and depending upon a certain accuracy and other inputs the simulation will provide the result and performance of both the approaches.**

***Index Terms*—Pursuit-Evasion, Escape-Chase, pursuit graphs, speed estimation, chase algorithm.**

I. Introduction

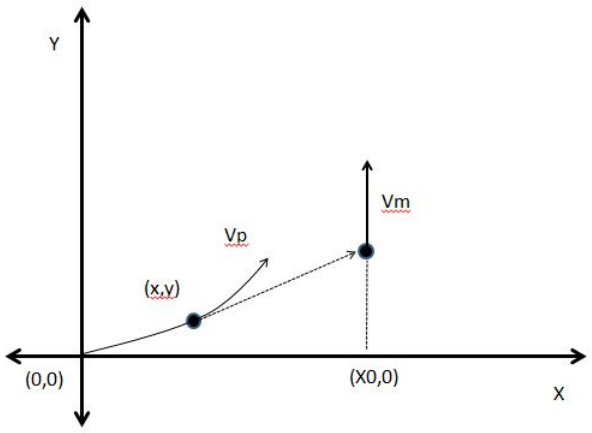

Fig. 1. Classic Ship Analysis

The above figure is the basis of Pierre Bouguer's pirate ship analysis. It was one for the very first analysis in the field of pursuit-evasion. In this, Bouguer considered a pirate ship chasing down a merchant ship.[**?**] Here, at time t = 0, there is a pirate ship at (0,0) which is chasing down a merchant ship at x_0 distance. The speed of both ships is v_p and v_m respectively and any given time the pirate ship moves towards the merchant ships position only. Thus a tangent at the path of pirate ship will always meet the merchant ship. At time t if the pirate ship is at (x,y) then the equation of y(x) can be calculated as,[1]

$$\begin{aligned} y(x) &= (n/(1-n^2)) * x_0 + [1/2 * (x_0 - x)] * \\ &\quad [(1-(x-x/x_0)^n/(1+n) - (1-x/x_0)^{-n}/(1-n)] \\ &\quad n = v_m/v_p \end{aligned} \tag{1}$$

S. Rao is with the International Institute of Information Technology

Here, differential model of pursuit evasion is used to fetch a saddle equilibrium and generate the capture point if exists. But the issue with differential pursuit evasion is that there might be more than one method possible to solve the problem. It involves the determination of optimal strategies. There are few concerns with the above equation, first is, it assumes that pirate ship is situated perpendicular to the motion of merchant ship which might not always be possible and second is that it doesn't address the possibility of both ships having different starting Y-axis co-ordinates. Another issue is with the rule of pursuit which is being followed is optimal or not. As we know in modern days, there is speed estimation technology exists for ships as well as other pursuers.[4] So we will be creating new derivations to accommodate these concerns and provide an analysis for which of the two approaches is going to result in better outcome, given certain inputs.

There has been some work in the field of creating and simulating a chase as [3] analyzes a simple robot with local sensors that moves in an unknown polygonal environment. The robot can execute wall-following motions and can traverse the interior of the environment only when following parallel to an edge. Also [2] a virtual avatar is programmed to pursue or evade the participant in an ambulatory virtual environment. Both of these researches have been highly useful in the robotics and self driven chases. But when it comes to implementation for mobile devices or machines with less processing power, they prove to be less practical. So here we are proposing a simple yet powerful chase algorithm, which will take Bouguer's analysis as basis implement modern estimation approach in it as well.

II. Theory and Concept

To create an efficient model, we need to take every case into our consideration that might occur during a chase, be it angular or linear deviation or both. Now, in our model we are using two basic approaches of chase-

1. When pursuer is always moving towards the evader. In this case, if at any point of time if we construct a tangent across the path of pursuer then it will definitely concatenate evader's position at that moment.
This approach is named traditional, as it follows the traditional concept of moving towards your goal.

2. Whereas in a more modern approach, the pursuer moves towards the point where he thinks the evader will be at the time he will reach there.
In other words, the pursuer tries to guess the speed of evader then determine a capture point and move towards it.

Both of the two approaches are efficient for a chase, and it is obvious that if the pursuer is able to determine the speed of evader accurately, then the modern approach is always going to be the better of the both. But it is not the case as we have already discussed δ. So depending upon the value of delta anyone of the above two approaches may be better than the other.

In our model, we have created an intelligent chase algorithm which will consider both the approaches at any given situation and then it will determine which one to use. The results generated from such hybrid method are more efficient than both the above pure methods.

A. Terminology

Pursuer: At any given point of time, the coordinates of pursuer will be stored as (x_p, y_p), $\vec{v_p}$ is the velocity and the magnitude of his speed will be as v_p.

Evader: Similarly, At any given point of time, the coordinates of evader will be stored as (x_e, y_e), $\vec{v_e}$ is the velocity and the magnitude of his speed will be as v_e.

Other: (x_s, y_s) is going to be the final capture point for the given situation, and $n = v_e/v_p$ is the speed ratio of pursuer and evader, δ is the inaccuracy by which pursuer is determining the speed of evader.

B. Traditional Approach

As discussed above, the tradition pursuit analysis gives us the result based upon the concept that initially the pursuer is at (0,0) and evader is at $(0, x_m)$. But that might not be the usual case,

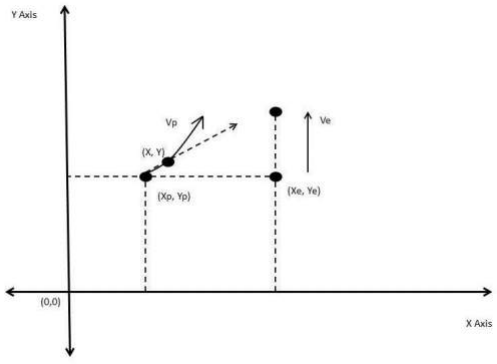

Fig. 2. A more Generic situation

As seen in the fig.2, Initially the pirate ship is considered at (x_p, y_p) and the merchant ship is at (x_e, y_e) with speeds $v_p and v_e$. Now the new equation for such scenario will be

$$y(x) = (n/(1-n^2)) * (x_e - x_p) + [1/2 * (x_e - x_p - x)] * [(1 - (x - x_p)/(x_e - x_p))^n/(1+n) - (1 - (x - x_p)/(x_e - x_p))^{-n}/(1-n)] \quad (2)$$

1) Distance Deviation: It is not compulsory for both the pursuer and evader to be in same line, i.e. $y_e = y_p = y_{'}$. We can compute the capture point for such case by reverse engineering the traditional formula.

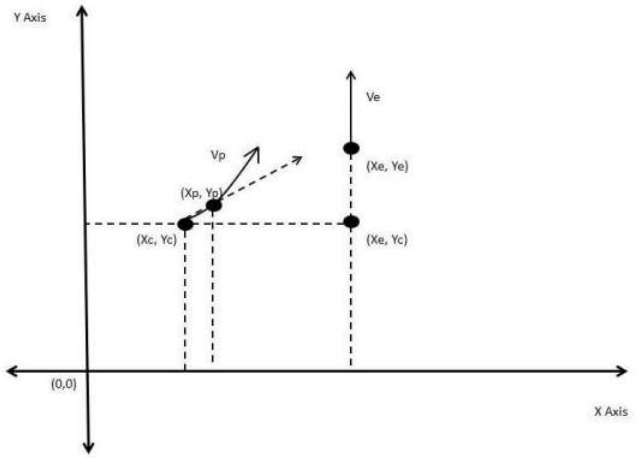

Fig. 3. Distance Deviation

Here in fig.3 we know that when evader is at (x_e, y_e) and pursuer is at (x_p, y^p) then we need to get the value of x_c in the graph and after that we will get the hypothetical starting position, which in turn can be used to calculate the capture point.

$$y_p = y_c + (n/(1-n^2)) * (x_e - x_c) + [1/2 * (x_e - x_c - x_p)] * [(1 - (x_p - x_c)/(x_e - x_c))^n/(1+n) - (1 - (x_p - x_c)/(x_e - x_c))^{-n}/(1-n)] \quad (3)$$

here (x_c, y_c) can be estimated, after which we can calculate the capture point for such case.

2) Angular Deviation: Apart from linear variation, there might be some angular deviation followed by merchant ship.

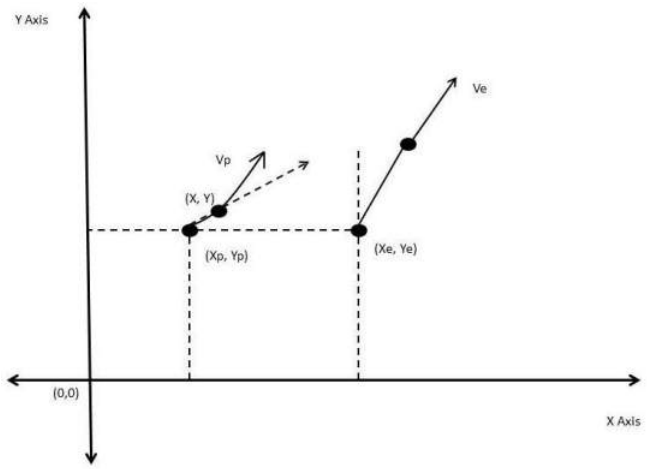

Fig. 4. Angular Deviation

As shown in the fig.4 there is a variation of θ. This situation can be solved by using the rotation of the axis w.r.t (0,0). The

co-ordinates for any point P(x,y) will be

$$x^{'} = x * \cos\theta + y * \sin\theta$$
$$y^{'} = -x * \sin\theta + y * \cos\theta \quad (4)$$

Then we will use the same equations (1 and 2) to get the capture point, and then convert the co-ordinates back to the original axis.

C. Modern Approach

In this, the chaser is aware of the speed of target, with certain accuracy. In the traditional example of pirate ship analysis, the pursuers will always be aware of the speed of merchant ship v_e with inaccuracy of δ
Using the above information, the pursuer can calculate the nearest capture point and move towards it in linear shortest path motion.

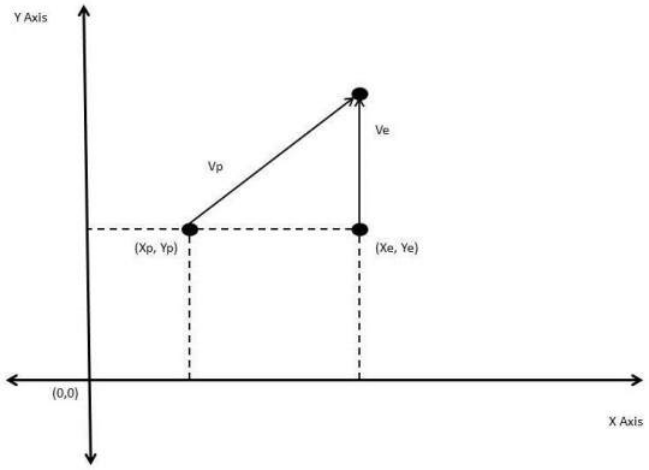

Fig. 5. Modern Approach

Here in fig.5, (x_e, y_s) is the capture point which can be generated as,
Case-1. δ = 0
In this case δ = 0, means the pursuer is able to measure the speed accurately.

$$(y_s - y_e)/v_e = ((x_e - x_p)^2 + (y_s - y_p)^2)^{1/2}/v_p \quad (5)$$

The only unknown in the above equation is y_s, and it can be solved as a quadratic equation. Out of which one solution will be $(< y_p)$ which can be ignored.

Case-2. δ_i is positive
In such case, the measured speed will be more than the actual speed of merchant ship. So the pirate ship will reach the capture point earlier than the merchant ship and it has to move towards merchant ship for some time.
Here the measured speed of merchant ship = v_e
and actual speed is $v_e - \delta_i * v_e$
So the total time here,

$$(y_s - y_e)/v_e + ((y_s - y_e)/v_e) * \delta_i * v_e * (v_p + v_e - \delta_i * v_e) \quad (6)$$

Case-3. δ_i is negative
Similarly as above case, after reaching the capture point, the pirate ship has to trail behind merchant ship.
Here the measured speed of merchant ship = v_e
and actual speed is $v_e + \delta_i * v_e$
So the total time here,

$$(y_s - y_e)/v_e + ((y_s - y_p)/v_e) * \delta_i * v_e * (v_p - v_e - \delta_i * v_e) \quad (7)$$

Distance Deviation in this approach won't require any special calculation, similarly *Angular Deviation* can be handled by the same axis rotation method.

III. Model

A. Basic Algorithm

A basic algorithm, is the one that follows any one of the two approaches throughout the scenario. Which means that if it is set for the traditional approach then it will keep on following that until we have reached the capture point. Another assumption that is taken into consideration is that in the case of such basic algorithm, we are taking "delta" as zero when following the modern method. Which means that pursuer is accurately able to determine the speed of evader.

Inputs: $(x_p, y_p), (x_e, y_e), v_p, v_e, \theta$ at t=0 (Starting Values)

Stack: The stack is something which is going to play a pivotal role, it is going to reflect the changes in evader's speed and angle over the period of time. Each entry in the stack can be considered as t_d (Time when the deviation occurs), ang_d (angular deviation), vel_d (velocity deviation). The stack can be considered as the part of the input, if there is a smart algorithm which is written by the perspective of evader then there is going to be no need to keep the stack values as they will be automatically generated from the evader algorithm.

Output: $(x_s, y_s), T$ (Total time taken in the chase).

Here, once we have provided the valid inputs to the above algorithm, it runs the while loop until both evader and pursuer have not met. In the loop we are using a variable e_q which determines whether there is any distance deviation or not. It simply acts as a flag.

There are two different functions used in the above algorithm,

1.Move: Used to move both the pursuer and evader in their direction at a distance of unit time quantum.

2.Capture: Used to finish the algorithm, if there are no more runtime deviations and returns the capture point as well as total time taken.

Algorithm 3, details about the capture function, it applies the previous equations, based upon the requirement. Move function follows the same conditions, with a little change that

Algorithm 1 Basic Algorithm

Initialize θ *and* $t \leftarrow 0$
while $(x_p, y_p)! = (x_e, y_e)$ **do**
  if $y_p = y_e$ **then**
    $e_q \leftarrow NODEV$
  else
    $e_q \leftarrow DEV$
  if *Stack.top = null* **then**
    $capture(e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e)$
    $return(t, (x_p, y_p))$
  else
    if $Stack.top(t_d) = t$ **then**
      $move(e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e, ang_d, vel_d)$
      $Stack.pop()$
    else
      $move(e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e, 0, 0)$
  $t++$
return $(x_p, y_p), t$

instead of finishing the whole chase, it return the values of $(x_p, y_p), (x_e, y_e)$ after one time quantum only.

Algorithm 2 Move$(e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e, ang_d, vel_d)$

Update $v_e \leftarrow v_e + vel_d$
$\theta \leftarrow \theta + ang_d$
return $(x_p, y_p), (x_e, y_e)$ *with single quantam movement.*

Algorithm 3 Capture$(e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e)$

if $\theta = 0$ **then**
  No angular deviation
  if $e_q = 0$ **then**
    No distance deviation either.
    Use eq.2 and return (x_p, y_p)
  else
    Distance deviation exists.
    Use eq.3 and return (x_p, y_p)
else
  Angular deviation exists
  Use eq.4 to rotate axis over θ
  if $e_q = 0$ **then**
    No distance deviation either.
    Use eq.2 and rotate axis back.
    return (x_p, y_p)
  else
    Distance deviation exists.
    Use eq.3 and rotate axis back.
    return (x_p, y_p)

The difference between these two functions is that, if there are no more deviations in the speed or angle of evader in future then it is better to finish the algorithm. Whereas on the other hand if there are some future deviations then we must only move towards the next coordinates.

Now using above functions, algorithm first checks which one of the previous equations is applicable in existing situation broadly basic or deviation the rest of the classification is taken care inside move and capture functions. Once equation type is determined, it checks if the stack is empty. If true that means, there are no further deviations in the future and we can simply call the capture function and finish algorithm. Else it checks whether the current time is same as the time on which deviation is supposed to happen and depending upon that it calls the move function with certain parameters.

B. Delta

As discussed earlier, if pursuer is able to determine the speed of evader accurately then all we need is basic algorithm only which is going to run on modern approach. But that can't be normal case, there must always be some inaccuracy, it has been observed that even in latest radar speed estimations there is some inaccuracies.[5][6]
Here we will be using two different values of delta-

1) δ_p*:* This value is calculated with time and is known to the pursuer as his existing inaccuracy.

2) δ_i*:* Inaccuracy associated with ith iteration in stack.

$$\delta_p = \delta_p + |\delta_i|/(i+1) \tag{8}$$

C. Hybrid Algorithm

For these reason, there is a requirement of a hybrid algorithm which incorporates best of both methods. This algorithm compares both methods at each iteration and then pick the one which is best suitable for current values. It also keep on updating δ_p with each iteration.
This algorithm is almost similar to basic algorithm with a few changes-

1. There is a δ_i associated with each iteration of stack, which is unknown to pursuer until $t <= t_d$

2. There is a new function named compare, which actually takes both derivations of basic algorithms and runs them on the input. After which it provides which algorithm is better suited for current case.

3. There is a new variable called algo, which is used to hold the information or output of compare function.

Complexity

The complexity of both hybrid and basic algorithm, depends upon the iterations stored in the stack. If the input stack is empty, i.e. the speed and angle of evader is going to stay same through out the chase then capture point and total time

Algorithm 4 Hybrid Algorithm

Initialize θ, δ_p *and* $t \leftarrow 0$
while $(x_p, y_p)! = (x_e, y_e)$ **do**
  if $y_p = y_e$ **then**
    $e_q = BASIC$
  else
    $e_q = DEV$
  $compare(e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e, 0, 0)$
  Set algo
  if *Stack.top = null* **then**
    $capture(algo, e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e)$
    $return(t, (x_p, y_p))$
  else
    if $Stack.top(t_d) = t$ **then**
      $compare(e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e,$
      $ang_d, vel_d)$
      Set algo
      $move(algo, e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e,$
      $ang_d, vel_d)$
      $Stack.pop()$
      Update δ_p
    else
      $move(algo, e_q, (x_p, y_p), (x_e, y_e), \theta, t, v_p, v_e, 0, 0)$
  $t++$
return $(x_p, y_p), t$

can be calculated by using previous derived equations in a single step.
On the other hand, when there are some entries in stack, then the algorithm is going to run one time quantum at a time until the stack is empty.

- Best Case- O(1), Stack is empty.
- Worst case- O(MAX(t_d)), where MAX(t_d) is the time associated with the last iteration stored in stack.

Here as similar to the basic algorithm, we can see that first the equation type is determined in current situation. Once done, a compare function is called and value of $algo$ variable is set. This value is going to change only if there is some iteration on the top of stack for $t = t_d$. In such case compare function is called again, this time with the values of stack as well and accordingly $algo$ is set again.
After which move or capture functions are called, depending upon the situation.

IV. Future work

Both the algorithms have been implemented in C++ and are working perfectly.

1. As suggested, I am trying to implement these algorithms in higher dimensions.

2. I am also working upon creating an android chase game which will use the hybrid algorithm.

3. Obstacles can also be introduced with following types-

1) : Avoidable- These obstacles will only change the speeds of pursuers and evaders without changing their paths.

2) : Unavoidable- These obstacles will effect both speed and directions of agents.

3) : Application in mobile gaming as an intelligent chase algorithm.

V. Conclusion

Here, we have used one of the first pursuit-evasion equation and used it to derive a new modern chase algorithm. Which can be highly useful to provide basic chase intelligence in devices with processing constraints.

References

[1] Nahin, Paul J. Chases and escapes: the mathematics of pursuit and evasion. Princeton University Press, 2012.
[2] Warren, William, and Jonathan Cohen. "Perceiving pursuit and evasion by a virtual avatar." Journal of Vision 10, no. 7 (2010): 1041-1041.
[3] Katsev, Max, Anna Yershova, Benjamin Tovar, Robert Ghrist, and Steven M. LaValle. "Mapping and pursuit-evasion strategies for a simple wall-following robot." Robotics, IEEE Transactions on 27, no. 1 (2011): 113-128.
[4] Liu, F., F. Zhao, W. Yu, L. Shi, and R. Wang. "Ship detection and speed estimation based on azimuth scanning mode of synthetic aperture radar." Radar, Sonar —& Navigation, IET 6, no. 6 (2012): 425-431.
[5] Tunaley, James KE. "The estimation of ship velocity from SAR imagery." In Geoscience and Remote Sensing Symposium, 2003. IGARSS'03. Proceedings. 2003 IEEE International, vol. 1, pp. 191-193. IEEE, 2003.
[6] Shang, Shang, and Zhang Ning. "Low speed target detection with short CIT in HF surface wave radar." In Signal Processing Systems (ICSPS), 2010 2nd International Conference on, vol. 2, pp. V2-499. IEEE, 2010.
[7] Blinn, James F. "How to solve a quadratic equation." IEEE computer graphics and applications 25, no. 6 (2005): 76-79.