Modelling male fly courtship pursuit behaviour

Eugenia Chiappe Lab Rotation work*

Francisco Moreira de Azevedo

June 2023

1 Abstract

Studying animal behavior is challenging due to the inability to know the moment to moment state and goal of the animal. Decision making tasks constrain the animal's behaviour by reducing it into limited possible outcomes (correct, incorrect, miss, etc.) in order to understand how the animal might be transforming the information available to it. However the representations and algorithms the animal might be using are most often implicitly assumed. For example in any freely moving decision-making task there's a possibility that an animal might be using information about its own body to predict choice instead of transforming external inputs. An example would be a mouse using its own body movement, instead of the stimuli, to estimate elapsed time in a time categorization task.

In contrast during natural behaviours such as courtship, hunting and escape we know animals' explicit goals - mating, feeding, survival, respectively. This inherently reduces the behaviour to a specific immediate objective, amenable to our understanding, without the need for a task with inherent human biases concomitant with anthropocentrism.

In the case of fly courtship behavior, the male pursuits the female by minimizing the angle between its body and the back of the female, while also minimizing the distance. It achieves this by using its forward, sideways and rotational velocity. Despite clarity in the inputs and outputs of this behavior, the algorithm and it's mechanistic implementation are unclear. One added challenge is that according to data from Miguel Paço (unpublished) (also other papers?), flies react to changes faster than what's achievable by their visual processing (40-80ms) [1], hence suggesting a predictive or memory driven internal model.

To address this challenge, we used a control theory approach in which the current error angle is fed back, continuously updated and acted upon, to model fly courtship chase

*Final presentation **here**. Code and any other documentation housed in public repository **here** behavior. A similar approach has been used in which the authors controlled the error angle via angular velocity only, using a Proportional (P) and Derivative (D) controller [5]. Despite this, evidence from Miguel Paço (unpublished) shows that flies consistently use their sideways velocity to minimize the error angle. Taking this into account we used two controllers in parallel to control the error angle: one for the fly's angular velocity, another for its sideways velocity, keeping the forward velocity matched to that of the instantaneous velocity of the female. By comparing the model's behavior when using both controllers separately or together against real fly data we can understand which algorithms the fly might be using.

2 Problem statement

During chasing behavior the male approaches the female from its anterior part of the body. This chase can be seen geometrically in allocentric coordinates with γ defined as the angle between the external basis and the orientation of the fly, and λ between the same basis and the range vector (vector from the fly's centroid to the target). The objectives of the male are to:

1. minimize the **angle** $e = \lambda - \gamma$ between its orientation and the anterior part of the female (or target T);
2. minimize the **distance** d from the female.

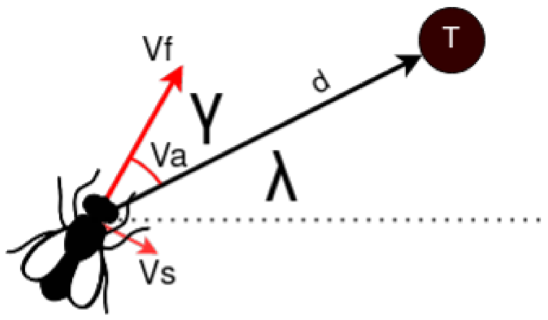


Figure 1: Geometric representation of inputs and outputs of fly chase

To achieve its objective the fly can move in two dimensions (decomposable into forward and sideways velocity, V_f and V_s, respectively) and rotate on itself (rotational velocity V_a). Throughout this work V_f has been chosen to match the target's instantaneous velocity, hence one of the objectives is not covered. In this work we also neglected the head's movement relative to the body as well as the eye movement which has been shown to occur in drosophila [3].

In this work we used a control theory approach to first simulate a virtual chase agent and secondly model real data of fly chasing an experimenter-controlled magnet.

3 Background on insect courtship behaviour

The first application of control theory to insect courtship behaviour was performed in the 70s by Land and Collet [6] on flying houseflies. By inspecting behavior they suggested that houseflies chase their mates using a PD controller, in which the proportional component responds irrespective of target's position on the retina, whereas the derivative component is only used when the target is on the front field of view (probably corresponding to the FOV in which there is binocular overlap). This is advantageous since otherwise the proportional term would be small when the error is small and the control of the error would be slower without the derivative component.

Focusing on the Drosophila melanogaster model in non-flying courtship, Robert Cook [2] did the first careful analysis showing that males chase females from their backs (Fig.8) using a proportional controller.

The most recent model [5] suggested for drosophila courthsip behavior is a I controller on the change in error angle. In this paper Sten et al sustain specific claims:

1. P1 neurons release and reflect a dynamic state of sexual arousal but decayed when animals transiently stopped courting despite still having future "enduring" arousal;

2. Furthermore, activating P1 neurons induces courtship similar to spontaneous natural chasing. It also induces LC10a activity similar to the ones evoked by natural chasing (Fig.3E);

3. LC10a's activity reflects changes of target on the retina - phase of fluctuations in LC10a match those of target. LC10a's activity gain increases when courting (but not when moving);

4. Optogenetic manipulation (Fig.2G-H) show LC10a's sufficiency for ipsilateral orienting behaviors. Anterograde tracing shows that downstream synpatic partners innervate lateral accessory lobe (LAL) neurons (Ext.Fig.9G-K).

5. During spontaneous courtship P1 activity leads LC10a's by 500ms (Ext.Fig.10A).

6. LC10a neurons respond identically to targets of different sizes Ext.Fig.8M-N, consistent with behavioral data [8].

7. Lastly, there are two plots that seemingly agree with the fly using sideways velocity. In Fig4F, the fly is overcompensating (slope of red line lower than 1), which could be due to an effect of the sideways velocity. In Fig4I (on the ball) the model has 3 distinct behaviors: for small angles there's almost no activity (due to binocular overlap and integration, imagine target coming from left to right, for the right RF, it cancels out), the plateau from 1-7 rad/s (due to Vs), and then linear.

There are also details about the model important to our work, namely:

1. Most importantly the model implements an **integral controller** to the change in the error angle, as a code snippet (Appendix I) from the **source code** of the paper shows.

2. This can also be seen in Fig.4B since a proportional controller to the **change** in angle would yield constant lines, but the model traces match those of the target.
3. The model is composed of LC10a left- and right-FOV tuned integrate and fire neurons. These neurons' receptive fields cover **each** around 10.5^{o} of the FOV of the fly, together they span the whole FOV. The output is the subtraction of one population's activity from another (a form of circuit competition).
4. There is a binocular overlap of 15^{o} at the center which is necessary for the model to exhibit "predictive" behavior (Ext.Fig.11H).
5. AOTu activity (serving as a proxy of LC10a axonal boutons) exhibit direction selectivity (Ext.Fig.11) which is vital for following a target in a time-locked manner, otherwise the model is delayed, phase wise, relative to real fly's data.
6. Including P1 activity as a scale of input current to LC10a neurons changes gain of model output yielding a better match to data (Fig.4I). P1 activity influences LC10as via a continuous-gain, not via threshold (Fig.4J).
7. P1 activity also changes the model's behaviour qualitatively from continuous control to one closer to discrete bouts, similar to the animal's behavior Ext.Fig.13)
8. Ext.Fig.3H-I show that V_a has a maximum of 15 rad/s (860^{o}/s) and V_s of 35 mm/s (although with different distributions) in agreement with data from Miguel Paço;

This paper has a model that is one order of derivation below the one we've been using during the rotation work. This has some implications since their model does not allow for an integral component (since the integral of an angle is not an interpretable quantity). This would imply the model is subject to accumulation of error.

During a talk at Cosyne 2023, Matthew Collie (Wilson lab, Harvard) et al extend the work by Sten et al via experiments. Since a proportional controller delayed with regards to reference is not enough to track a reference in a time-locked manner, they state that adding a derivative controller compensates for it. Importantly enough, on the plots shown, the derivative component precedes the proportional one; however how this information is presented looks misleading as the derivative term would at most precede by one timestep.

They go on to suggest AOTU019 is a neuron that might implement derivative control. To test this they suppress AOTU019's activity - as a result, flies are still able to engage in courtship, however it causes the pursuer to lag target motion, as if the derivative component were removed; hence AOTU019 are necessary for faithful pursuit. Furthermore AOTU019's activity scales with target speed, which means it could be coding for the derivative of the error angle. Lastly, activity peaks the most, closest to the midline, where the target is in the region of binocular convergence. Evidence from Nuno Rito (unpublished) shows that males cannot pursuit targets when the center of their eyes is coated with paint. Joining these two pieces of evidence plus Land and Collet's hypothesis that the derivative component is only active on small error angles, one can speculate that the derivative component is perturbed **the most** when the region of binocular convergence is coated with paint, moreover this should be reflected in AOTU019's activity.

4 Results - Virtual agent simulation

To simulate an agent we created a virtual playground where we could test our controllers. Besides the implementation of the controllers, which follow the generic PID formulation (Eq.1):

$$\theta' = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{1}$$

We can change variables such as:

1. Paths (fictive or from real data)
2. Which controller is active (either one or both)
3. The parameters in the controllers
4. The lead/lag of each controller separately compared to the path
5. The forward velocity compared to target instataneous velocity

For demonstration purposes, all the plots shown in the Results section were done using real data from the magnet rig, with limits on velocities: $V_s < 10mm/s$ and $V_a < 1000°/s$. The lead/lag was chosen to be zero for both controllers.

4.1 V_a controller

The controller is implemented following Eq.3, in which δe is the input to the controller which can then be modified by either a Proportional (P), Integral (I) or Derivative (D) component, each then is summed to yield the change in angle.

We simulated a P controller with different values of Kp, the gain that multiplies the error, on a circular path shown in Fig.2. Increasing the value of the proportional component determines how fast it converges to the path and how much steady state error remains, at the cost of a less smooth control, resulting in fast changes in the angular velocity. Despite the limit imposed on V_a, no limit is imposed on how fast it can change, hence it can have a unrealistic behavior compared to a real fly. A proportional controller tuned to the fly data should wield spectrograms of V_a identical to those of real flies.

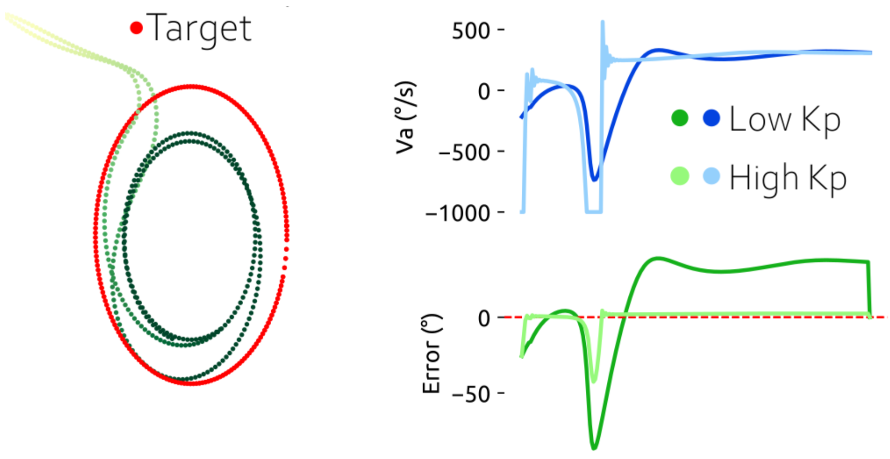


Figure 2: Effect of Kp when using P controller only for Kp=1.5 (”High”) and Kp=0.1 (”Low”)

The integral component have been traditionally (in engineering) added to controllers to eliminate steady state errors that remain either because 1) The proportional-only controller lags behind the process it’s trying to control or 2) There is a bias in the actuator (in our case the fly’s legs) or sensor (retina and visual pathway) that shifts the controller by a fixed constant. The integral is a memory component looking back into the past, the length of this window has an effect on the behavior of the controller.

We simulated a PI controller with different values of τ_I - the length of the integration window - shown in Fig.3. Smaller windows lead to larger oscillations in the error and angular velocity, since there each sample of error changes more the output of the component. A tuned PI controller is one where the window of integration τ_I and gain K_I are tuned to match the frequency of oscillations and trends of the target.

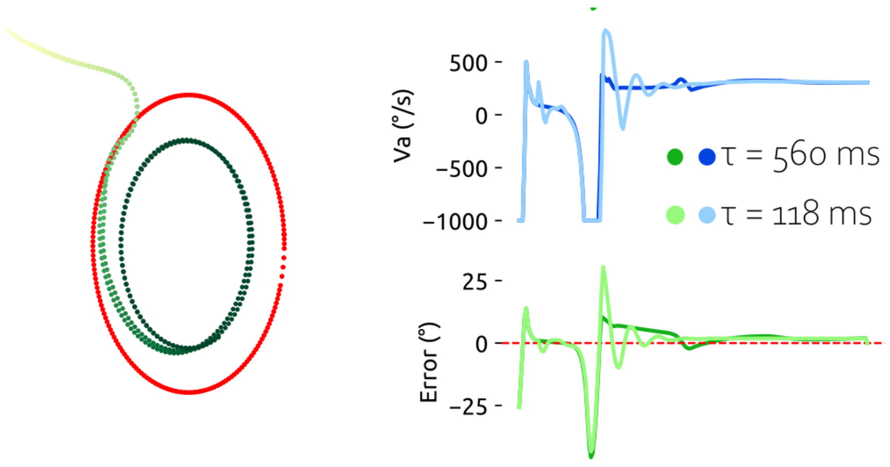


Figure 3: Effect of τ_I when including integral forming a PI controller

A common issue with integral components (a quick video explanation) arises when the controller's output reaches saturation (maximum angular velocity in our case) due to a large change in direction, and remains saturated for a sustained period of time. This leads to the integral error accumulating to compensate, however if the target changes direction, this error takes time to be depleted; whilst in the meanwhile it is counteracting the proportional controller leading to a sluggish, delayed response. This phenomena is called integral windup. As previously mentioned in the Background section, this is an issue already studied by neurophysiologists studying VOR in the eye ETC ETC

Lastly there is the derivative component. It is generally used to minimize fluctuations that may be induced by the integral component or inherent in the target's movement. Intuitively it is akin to a high proportional component in the sense that it creates sharp changes in the angular velocity that might not be realistic, as seen in Fig.4. The downside is that it is very sensitive to noise: since the derivative is predictive in its nature, if the prediction in one step is of opposite sign to the next, it will induce oscillations by overcompensating, yielding a control signal with high frequency oscillations known as "chatter" (deeper explanation here). This is very likely to happen if the measurement has a lot of white noise in it.

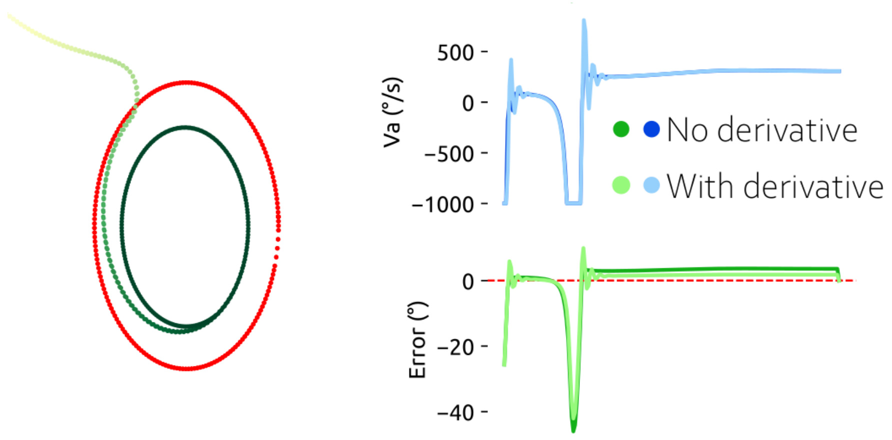


Figure 4: Effect of Kd when using a full PID controller

One last technical detail concerning the implementation of the derivative: by definition the derivative should be performed on the error signal, however if the error has a sharp (or non-continuous) transition, this will induce a large (theoretically infinite) change in the control signal, a phenomena known as "derivative kick" (more info here). To avoid this, the derivative is generally implemented on the manipulated variable (also known as the process variable), which is mathematically similar (as shown in the reference above) except for the avoidance of the derivative kick.

4.2 Parallel V_a and V_s controller

The controller is implemented following Eq.6 with a geometric interpretation in Fig.8, with derivation details in Appendix II.

Fig.5 shows a simulation (left) with only V_a (label incorrectly shows V_s) and a $V_a + V_s$ parallel controller. On the right are shown angular and sideways velocity as well as error of only the angular component. The V_a controller was a PD with $K_p = 1$, $K_d = 0.5$. The V_s controller was a PI with $K_p = 1$ and $K_i = 0.5$ with $\tau = 500ms$. In qualitative terms, when using both controllers the pursuer appears to intersect the target at its collision course, rather than following from behind, although this can be explained/influenced by other possible factors such as not normalized velocities, integral windup, zero lag between process and controllers, integration window size among others. Paradoxically it seems both controllers **together** yield a larger error and a more costly control, but that may be a cost worth to have from the animal side in order to adapt a strategy of parallel pursuit.

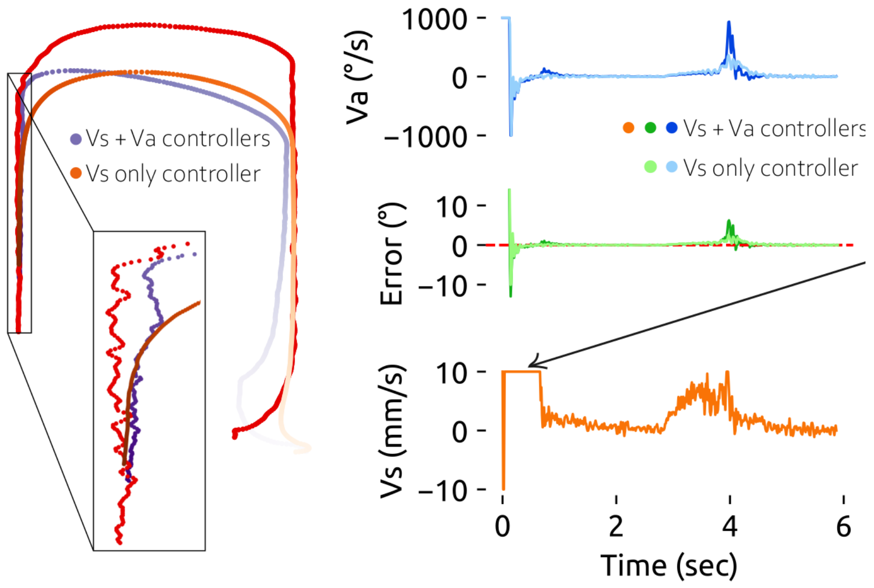


Figure 5: Simulations of V_s-only and $V_a + V_s$ parallel controller. Target is in red. Arrow on right bottom points to an example of integral windup.

To disentangle some of the factors mentioned, simulations were performed where the relative lag between the Vs and Va controller was changed (Fig.6). It is clear that the pursuer is using a parallel pursuit trajectory, chasing the target from the side, being sustained by V_s. In this case the V_s controller was only an Integral but the behaviour does not change when it is a PI. Furthermore the integration window also does not yield significant differences (data not shown). As a negative control when the V_s lags V_a instead, the behavior is identical to that of just V_a working instead (data not shown).

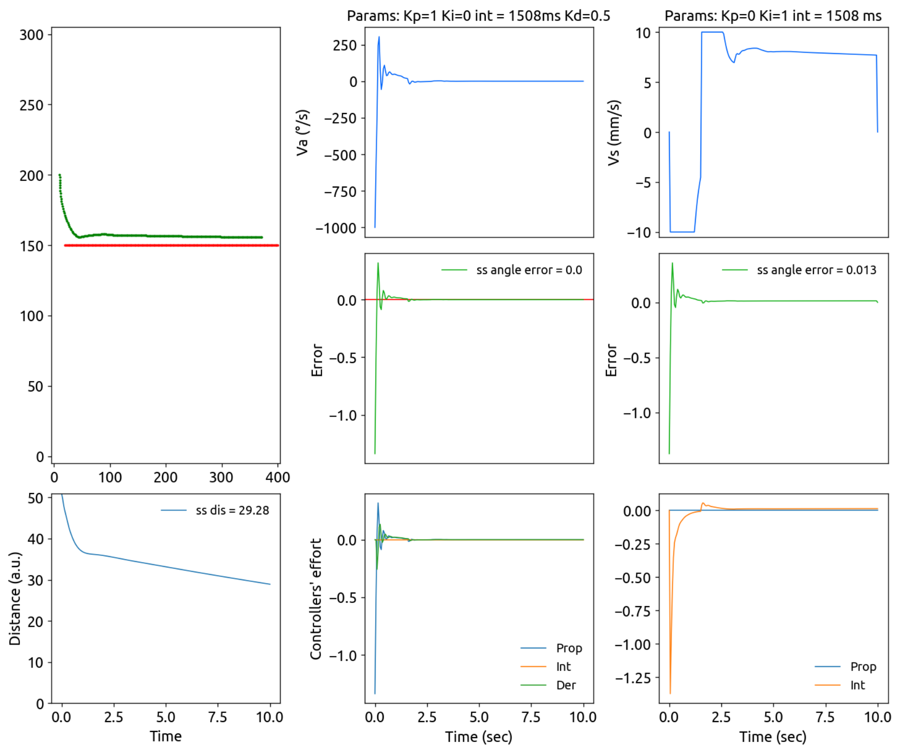


Figure 6: Simulations of $V_a + V_s$ parallel controller in which V_a is delayed by 50 ms compared to V_s

5 Future work

There are several directions for future work, from small implementation changes and simulations that can be performed in a short period of time, to others that would involve hypothesis and predictions from real data. A non-exhaustive list follow:

1. When simulating both controllers the net velocity ($V_f + V_s$) of the model is not normalized to match the net velocity of the target;
2. The controller's parameters should be tuned to the real fly's data in order to make any claims or interpretation with regard to the model's behavior compared to real flies. It is not trivial to decide which (and how much) data to choose from, in order to tune the model;
3. Furthermore only one of the controllers can be tuned when using both in parallel. Which controller to tune has scientific implications about which claims are allowed;
4. Simulate the agent with the derivative on the error instead of on the PV, specifically in environments with sudden changes in set point;
5. Implement Integral windup reset on V_s;

6. Condition the derivative component of V_a to activate only when $e < 15°$ to test Land and Collet's hypotheses;

7. Tune optimal window of integral for V_s via sideways velocity spectrum of the real fly in a straight path. If I controller integration window is exactly matched to the period of oscillations in target then pursuer chases target in straight line - however does not work if there is integral windup reset;

8. Test whether model recapitulates better data from head-fixed rather than normal flies to test if our assumption that head movement doesn't play a major role holds;

9. Lagged pursuer, but it uses a predictive linear model to judge the target's direction based on X past positions (would break on parallel cylinders path);

10. Some research has been performed showing that the distance matters in terms of courtship [4][7], hence it would be interesting, in order to have a complete model, to have a controller on the forward velocity too.

References

[1] Rudy Behnia, Damon A Clark, Adam G Carter, Thomas R Clandinin, and Claude Desplan. Processing properties of on and off pathways for drosophila motion detection. *Nature*, 512(7515):427–430, 2014.

[2] Robert Cook. The courtship tracking of drosophila melanogaster. *Biological Cybernetics*, 34(2):91–106, 1979.

[3] Lisa M Fenk, Sofia C Avritzer, Jazz L Weisman, Aditya Nair, Lucas D Randt, Thomas L Mohren, Igor Siwanowicz, and Gaby Maimon. Muscles that move the retina augment compound eye vision in drosophila. *Nature*, pages 1–7, 2022.

[4] Andreas F Haselsteiner, Cole Gilbert, and Z Jane Wang. Tiger beetles pursue prey using a proportional control law with a delay of one half-stride. *Journal of the Royal Society Interface*, 11(95):20140216, 2014.

[5] Tom Hindmarsh Sten, Rufei Li, Adriane Otopalik, and Vanessa Ruta. Sexual arousal gates visual processing during drosophila courtship. *Nature*, 595(7868):549–553, 2021.

[6] Michael F Land and Thomas S Collett. Chasing behaviour of houseflies (fannia canicularis) a description and analysis. *Journal of comparative physiology*, 89:331–357, 1974.

[7] RM Noest and Z Jane Wang. A tiger beetle's pursuit of prey depends on distance. *Physical Biology*, 14(2):026004, 2017.

[8] Inês MA Ribeiro, Michael Drews, Armin Bahl, Christian Machacek, Alexander Borst, and Barry J Dickson. Visual projection neurons mediating directed courtship in drosophila. *Cell*, 174(3):607–621, 2018.

Appendix I - Code snippet from model by Sten et al [5]

In more detail, the model computes motion direction as the sign (not the value) of the difference of the position of the female in the male's retina. This is then used as an input to compute the spikes using an integrate and fire neuron model, however only neurons in which the RF covered the position of the target are able to fire. The script snippet shows that the heading is computed as the difference between all the left- and right-FOV tuned neurons. This corresponds to a proportional and not just a constant controller since the faster the target moves, the more RFs (hence more neurons) it will excite (second code snippet). However the first snippet shows that the neurons use temporal receptive fields (derived from [8]) meaning they integrate information about the change in angle over time, thence, in terms of its outputs the model is really an **Integral** controller on change in angle (which can also be viewed as a proportional controller to the angle itself, meaning the position on the retina).

```
% get input current to each LC10a neuron
if strcmpi(METHOD,'threshold')
    filtInput = temporalRF_LC10(inField,relativeTime,(tc(
       ImIdx)>threshold)*0.5);
elseif strcmpi(METHOD,'continous')
    filtInput = temporalRF_LC10(inField,relativeTime,tc(
       ImIdx));
elseif strcmpi(METHOD,'none')
    filtInput = temporalRF_LC10(inField,relativeTime,1);
end

% Transform spiking to turning and compare to behavior
binSize = 30; %ms
modelTimeStamp = 0:dt:t-dt;
deltaHeading = zeros(length(modelTimeStamp)-binSize,1);
for i = 1:length(modelTimeStamp)-binSize
    %compute spikes in right and left hemisphere; compare
    rightSpikes = sum(sum(V_exc(i:i+binSize,1:nLC/2) ==
       spikeMag));
    leftSpikes = sum(sum(V_exc(i:i+binSize,nLC/2+1:nLC) ==
       spikeMag));

    deltaHeading(i) = leftSpikes - rightSpikes;
end
```

Appendix II - Deriving controller equations

The control law for the angular controller is as follows:

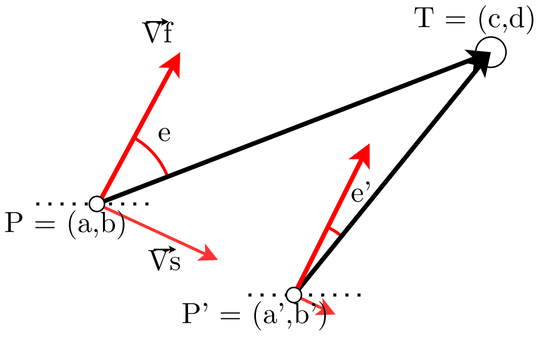
$$e = \lambda - \gamma \tag{2}$$

Fig. 1 shows how the angles are defined. Both γ and λ are defined in allocentric coordinates. The angle is updated following Eq. 3:

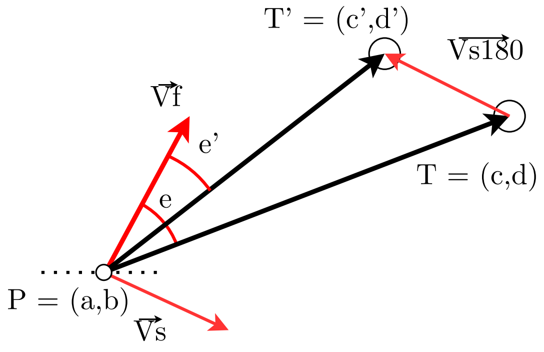
$$e' = e - \delta e \tag{3}$$

In this equation δe is the input to the controller. The output depends on the controller configuration and parameters.

For the sideways controller the equations implementing them derive from an insight shown in Fig.7. In order to control the angle e using the sideways we first compute it using Eq.2. To correct this angle the controller would suggest a change in angle (Eq. 3), the same as in the angular controller, represented in $^\circ/s$. In the case of sideways velocity, the challenge is converting this to units of mm/s, so the problem becomes the one exposed in Fig.7a: what is $|\overrightarrow{V_s}|$ such that e yields e' in the next step? (assuming the sideways controller is working in parallel to the angular one and independent to its effects). This can be solved by iteration over magnitudes until our desired angle is achieved, but we derived analytical solution to this problem.



(a) Diagram of update of position based on V_s



(b) Equivalent diagram to a) but in which the target moves instead of the pursuer

Figure 7: Geometric intuition for V_s controller set update in allocentric coordinates

Instead of viewing this problem in allocentric coordinates, one can formulate it via egocentric coordinates (Fig.8) and then transform them back. The idea being that if we update the angle δe, how much should $|V_s|$ change, expressed in Eq.6.

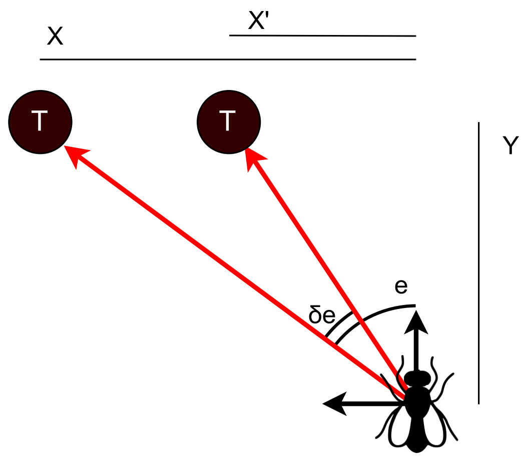


Figure 8: Geometric intuition for V_s controller set update in egocentric coordinates

$$tg(e) = X/Y \tag{4}$$

$$tg(e + \delta e) = X'/Y \tag{5}$$

$$\begin{aligned} |V_s| &= X - X' \\ |V_s| &= w(tg(e) + tg(e + \delta e) \\ |V_s| &= d(sin(e) - cos(e)tg(e + \delta e)) \end{aligned} \tag{6}$$