Navigating a 2D world using a 1D compound eye

— project report for CNS 186: “Vision” —

Daniel Wagenaar, March 2002

An artificial fly is developed that navigates a two dimensional maze using a one dimensional eye to avoid obstacles. Biologically plausible spatiotemporal filters are constructed as the basis for optic flow estimation. Flow is subsequently calculated using a modified population vector algorithm. Equations are derived for optic flow in an infinite tunnel and when approaching an infinite wall. These equations are used to construct a cell that chooses appropriate evasive maneuvers based on estimated optic flow. The model can be extended easily to include goal seeking behavior, and the results of the first attempt at such an extension are included.

Introduction

“How on earth do they do it?” — This is a question that drives a lot of neuroscientific research in insects. True, an even larger body of research focuses on how *we* do it, but the relative simplicity of insect nervous systems make them an attractive subject. In this project, I’ll consider how simple biologically plausible control systems can navigate through a maze without crashing into the walls. I will not attempt to follow the architecture of any particular insect, but I will use only processing elements in my design that biology could easily implement.

To reduce the amount of CPU time needed for experiments I will use a one dimensional eye to navigate a two dimensional world, rather than attempting the more realistic 2D eye in 3D world case. This will simplify some of the mathematics, while still retaining much of the flavor of the problem. Most importantly, it will greatly simplify visualizing the motion of the creature — which I shall henceforth refer to as ‘fly’, even though it is obviously much simpler than a real fly. Also, I shall use the word ‘retina’, simply because it is so much shorter than the more correct ‘photoreceptor array’.

Engineering

From retinal input to flow

I’ll operate under the assumption that it is necessary to have an estimate of optic flow in most parts of the visual field in order to do navigation. This may not in fact be the case, and it would be interesting to study the performance impact of using (far) fewer processing elements.

The most mathematically straightforward way to estimate optic flow, is a gradient method: if $I = I(\theta)$ is the intensity of light impinging on the retina, we can estimate flow f as:

$$f(\theta) = \frac{\partial_t I}{\nabla I}.$$

(I will use θ exclusively to denote retinal coordinates.)

This method is not necessarily the most attractive one though. Computationally, stability issues occur in areas with little contrast, and physically, the underlying assumption — the intensity reflecting of an object staying constant — may not always be true.

An alternative class of flow estimators, which I shall use here, employs spatiotemporal filters that are tuned to a particular spatial and temporal frequencies k and ω. The outputs y_i of a collection of such filters can then be used to estimate flow using a population vector approach:

$$f(\theta) = \frac{\sum_i y_i(\theta)\omega_i/k_i}{\sum_i y_i(\theta)}.$$

(In fact, as I will show below, while this approach may work for narrowly tuned filters, a slightly more sophisticated combination of outputs is required for broadly tuned filters.)

In silicon, one could easily construct spatiotemporal Gabor-like kernels:

$$Gb_{\pm} = \frac{1}{N} \left\{ \begin{matrix} \cos \\ \sin \end{matrix} \right\} \Big(k\theta + \omega t \Big) e^{-\frac{1}{2}\frac{\theta^2}{\sigma^2} - \frac{1}{2}\frac{t^2}{\tau^2}},$$

which give beautiful spatiotemporal impulse responses (Figure 1).

However, implementing arbitrary spatiotemporal filters biologically is not easy. Luckily, it is possible to approximate these filters by cleverly combining simple spatial filters with temporal filters [Adelson and Bergen, 1985]. They observe that combining a pair of spatial filters such as sine and cosine patches with a pair of temporal filters of different time constant yields spatiotemporal filters much like the ones depicted above in a biologically plausible way. I shall follow that approach, though I will use slightly different filters.

Spatial filters

To avoid phase dependency, it is preferable to use a quadrature pair of filters: two orthogonal spatial kernels K_+ and K_- whose impulse response quadratically adds to a Gaussian:

$$K_+^2 + K_-^2 = G^2.$$

The most straightforward choice for a quadrature pair of spatial filters would be a sine and a cosine modulating a Gaussian:

$$K_+^0 = \frac{1}{\sqrt{2\pi}\sigma} \cos\left(\frac{\alpha x}{\sigma}\right) e^{\frac{-\frac{1}{2}x^2}{\sigma^2}} \qquad K_-^0 = \frac{1}{\sqrt{2\pi}\sigma} \sin\left(\frac{\alpha x}{\sigma}\right) e^{\frac{-\frac{1}{2}x^2}{\sigma^2}},$$

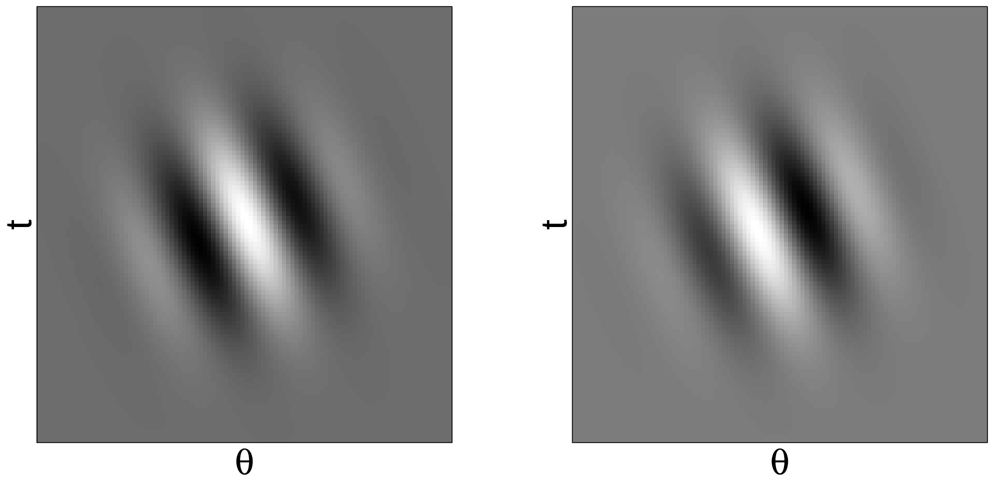


Figure 1: An array of gabor filters tuned for spatial and temporal frequency could be used to estimate local optic flow. These mathematically perfect space-time responses cannot be obtained in biology quite so easily though.

where $\alpha = 2 \ldots 5$ roughly defines how many positive and negative lobes the filter has.

However, these have the unwelcome property that K_+^0 responds to DC stimuli as well as to gratings, because $\int K_+ \mathrm{d}x \neq 0$. So, I decided to choose an alternative pair that does not suffer from this problem.

Consider the family of filters:

$$K_+^{(\beta)} = \frac{1}{\sqrt{2\pi}\sigma} \cos\left(\frac{\alpha x}{\sigma(1+\beta x^2)}\right) e^{\frac{-\frac{1}{2}x^2}{\sigma^2}} \qquad K_-^{(\beta)} = \frac{1}{\sqrt{2\pi}\sigma} \sin\left(\frac{\alpha x}{\sigma(1+\beta x^2)}\right) e^{\frac{-\frac{1}{2}x^2}{\sigma^2}}.$$

By varying β, it is possible to balance $\int K_+^{(\beta)} \mathrm{d}x = 0$, at least if $\alpha \gtrsim 2.5$. Results are shown in Figure 2.

As always, there is a price to be paid. In this case, the spatial frequency response curve of the filter is slightly broadened. This broadening is likely to reduce the S/N ratio of estimators based on these filters, but informal tests showed that the effect was negligible compared to the advantage of DC-tolerance.

I used σ values of 1.5 to 6 photoreceptor distances.

Temporal filters

In keeping with my aim of biological plausibility, I explored the possibilities of filters that can be built simply from passive dendrites. If a dendrite is modelled as a sequence of low-pass RC filters,

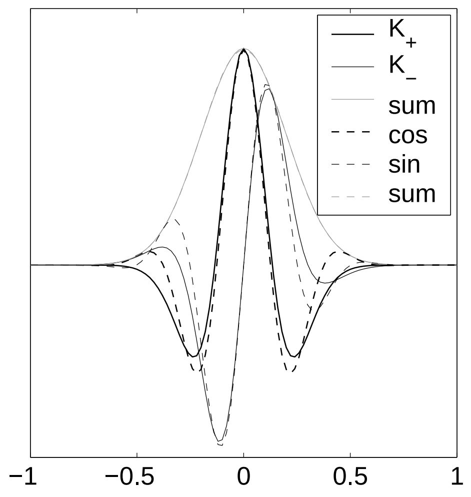


Figure 2: Impulse responses of the spatial filters K_+ and K_-. Also plotted is the quadrature sum $\sqrt{K_+^2 + K_-^2}$, showing that the envelope of the two filters is indeed a Gaussian. For reference, the unbalanced filters $\{\sin, \cos\}(\alpha x/\sigma)\exp\left(-x^2/2\sigma^2\right)$ are also shown (dashed curves).

the output y of a dendritic delay line obeys the following differential equation if driven by input x:

$$\left[1 + \left(\tau \frac{\mathrm{d}}{\mathrm{d}t}\right)^n\right] y = x,$$

where $\tau = RC$ and n is the number of elements. In the limit $n \to \infty$ this reduces to a perfect delay line, but I'll stick to low n. I wanted to have a pair of filters with impulse responses that produce a smooth curve without any local extrema when added in quadrature. Denoting the n-th order low pass filter with time constant τ as $K_n^{(\tau)}$, I found that the following pair fits the requirements:

$$K_S = K_6^{(\tau)} - K_{11}^{(\tau)} \qquad K_L = \sqrt{\frac{10}{6}} K_{10}^{(\tau)} - \frac{10}{15}\sqrt{\frac{10}{6}} K_{15}^{(\tau)},$$

where S and L refer to *short* and *long*. Figure 3 demonstrates that these filters produce a reasonably smooth envelope when added in quadrature.

In the following, τ values of 1.5 to 6 timesteps will be used, and a timestep will be taken to correspond to 1 ms. (Although this means that the simulation cannot be run in real-time, it is still preferable to larger timesteps which could easily lead to temporal aliasing when the fly moves through the world at reasonable velocities.)

Spatiotemporal filters

Multiplying $K_\pm$ against $K_{L,S}$ yields the separable spatiotemporal filters shown in Figure 4. By subtracting and adding pairs of these filters, a set of approximations to gabor patches is obtained (Figure 5). These are the basic velocity detecting filters I shall use in the following experiments.

Pre-filtering: spatial antialiasing

When images projected onto a real eye contain very high spatial frequencies, the result is a blur, because of refraction in the lens and eyeball. This is actually a very important feature, because if

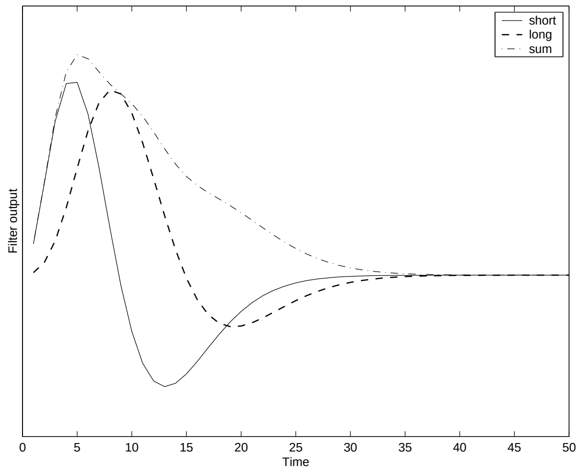


Figure 3: Impulse response of K_L and K_S. For reference, the quadrature sum $\sqrt{K_L^2 + K_S^2}$ is also plotted, showing that the filters produce a smooth envelope.

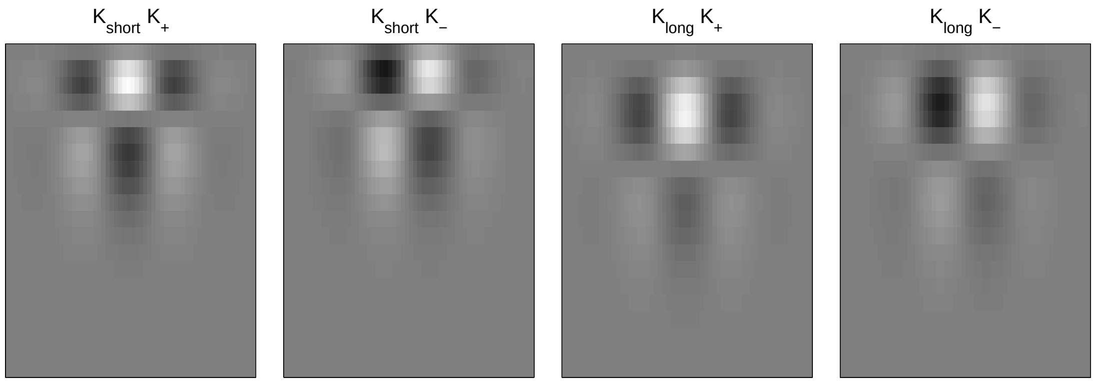


Figure 4: Combining the spatial filters $K_\pm(\theta)$ with the temporal filters $K_{L,S}(t)$ results in four separable spatiotemporal filters.

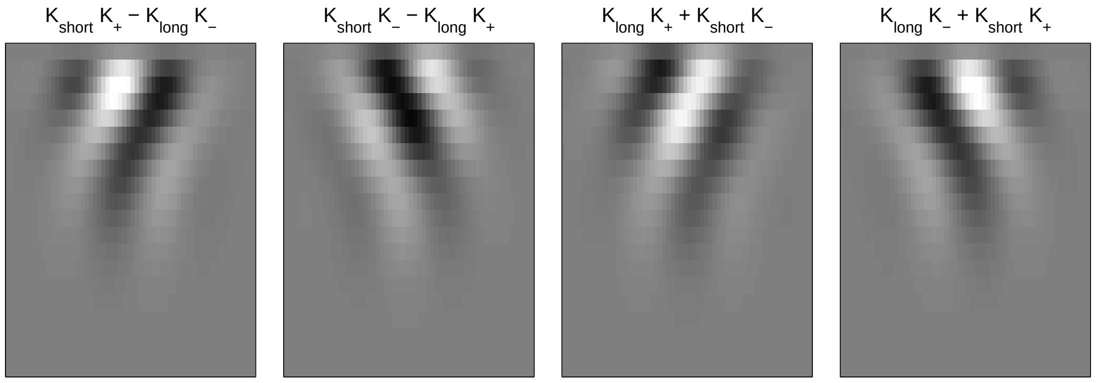


Figure 5: The separable filter responses shown in Figure 4 can be combined to yield velocity tuned responses. The left moving pair is in approximate quadrature as is the right moving pair.

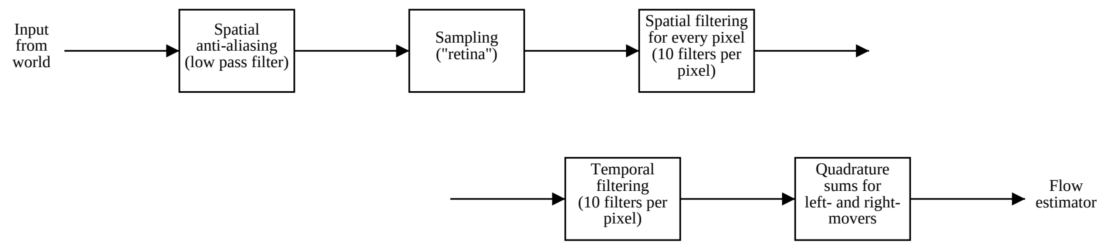

Input from world
Spatial anti-aliasing (low pass filter)
Sampling ("retina")
Spatial filtering for every pixel (10 filters per pixel)
Temporal filtering (10 filters per pixel)
Quadrature sums for left- and right-movers
Flow estimator

Figure 6: Schematic of optic path and early visual system. For a 180 pixel eye (1 pixel per degree, spanning 180°), 5 spatial filter pairs and 5 temporal filter pairs per pixel, there are a total of $180 \times 5 \times 2$ neurons involved in spatial filtering, followed by $180 \times 5 \times 5 \times 2$ neurons to calculate the quadrature sums after the temporal filters (which are implemented by the neurites between the spatial filters and the quadrature sum calculators).

frequencies higher than half the inverse separation between adjacent photoreceptors were allowed to reach the photoreceptor array, spatial aliasing would result, which cannot be removed in subsequent processing stages. The result would be confusion, because the aliased pattern might move with a very different velocity than the object generating it. I therefore apply a low pass filter to the image before feeding it into the retina.

The big picture

A schematic of the optical system presented above is given in Figure 6. The final result is a set of spatiotemporal filters as follows:

$$\begin{aligned}
K_{L1}(\theta, t) &= \big(K_S(t)K_+(\theta) - K_L(t)K_-(\theta)\big) * E(\theta), \\
K_{L2}(\theta, t) &= \big(K_S(t)K_-(\theta) + K_L(t)K_+(\theta)\big) * E(\theta), \\
K_{R1}(\theta, t) &= \big(K_S(t)K_+(\theta) - K_L(t)K_-(\theta)\big) * E(\theta), \\
K_{R2}(\theta, t) &= \big(K_S(t)K_+(\theta) - K_L(t)K_-(\theta)\big) * E(\theta),
\end{aligned}$$

where E refers to the spatial filter implemented by the eye: anti-aliasing and sampling, and L and R refer to *left*- and *right*-moving.

To obtain phase-free outputs from these, I take low-pass filtered the image I and compute the quadrature sums:

$$y^{(L)} = \big(K_{L1}I\big)^2 + \big(K_{L2}I\big)^2; \qquad y^{(R)} = \big(K_{R1}I\big)^2 + \big(K_{R2}I\big)^2.$$

Estimating flow from filter outputs

Of course there isn't just a single $y^{(L,R)}$ pair for each position in the visual field; there are several, one each for a range of spatial and temporal wavelength σ and τ. To make the system sensitive to a wide range of spatiotemporal frequencies, I chose a logarithmically distributed set of values:

$$\begin{aligned}
\sigma_i &= 1.5 \times 2^{i/2} \qquad \text{for } i = 0..4; \\
\tau_j &= 1.5 \times 2^{j/2} \qquad \text{for } j = 0..4.
\end{aligned}$$

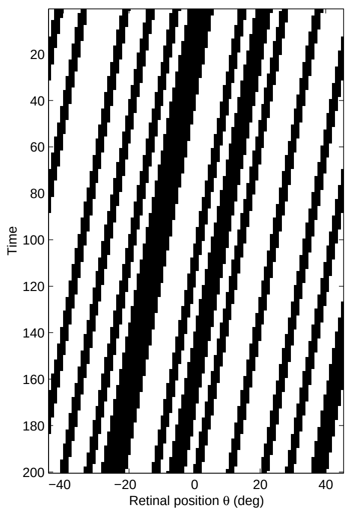


Figure 7: Example training movie. Notice the course angular resolution.

I experimentally determined the spatial frequency k_{ij} and temporal frequency ω_{ij} of the peak of each filter $y_{i,j}$ and calculated the peak velocity $v_{ij} = \frac{\omega_{ij}}{k_{ij}}$, rather than relying on the zeroth order estimate $v_{ij} \sim \frac{\sigma_i}{\tau_j}$.

To do this, I generated a large number of training stimulus movies such as the one shown in Figure 7, with a fixed and well-known spatal and temporal frequencies k, ω. I generated 10 such movies at each of 29 values of ω and 21 values of k. The result of training is shown in Figure 8.

A simple population vector approach for estimating flow at a given retinal position θ would now be:

$$f_0(\theta) = \frac{\sum_{ij}\left(y_i^{(R)}j - y_{ij}^{(L)}\right) v_ij}{\sum_{ij}\left(y_i^{(R)}j + y_{ij}^{(L)}\right)}.$$

However, this turns out to be strongly bias in favor of intermediate values of flow, since each filter is rather broadly tuned. To counter this bias, I attempt to find a better estimator in the family:

$$f(\theta) = \alpha\big[f_0(\theta)\big]^{\beta}.$$

(This particular equation works well to stretch the range back to reality because the individual filters are positioned logarithmically in velocity space.)

The system was calibrated using the same set of stimuli used for determining the peak locations, resulting in

$$\alpha = 6.47 \pm 1.42, \qquad \beta = 2.44 \pm 0.12.$$

(Quoted uncertainties are based on splitting the training set in 10 different ways.)

The results of calibration are shown in Figure 9.

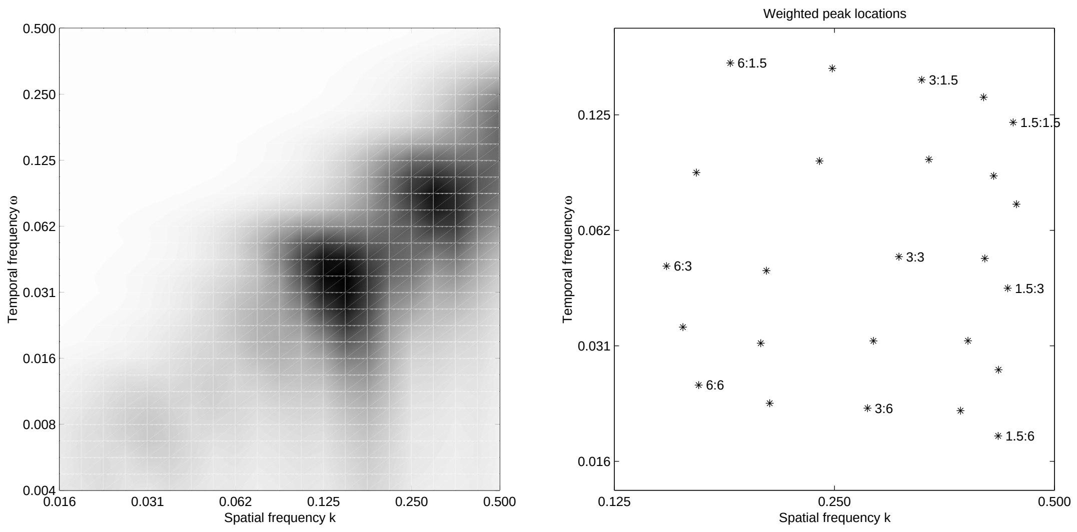


Figure 8: The tuning curve for one of the filters (left; black is strongest response) and the peak locations of all 5×5 filters (right). Selected filters have been labeled by their parameters in the form $\sigma : \tau$, measured in photoreceptor distances and timesteps respectively. (The white trangles are an artifact introduced jointly by matlab and my printer. Sorry.)

Using flow for collision avoidance

I want to use the estimated optic flow to make the fly avoid hitting walls. Therefore, it is necessary to know how moving past simple shapes generates flow. That information can then be used to construct decision-making cells that integrate (part of) the flow field against a suitable kernel.

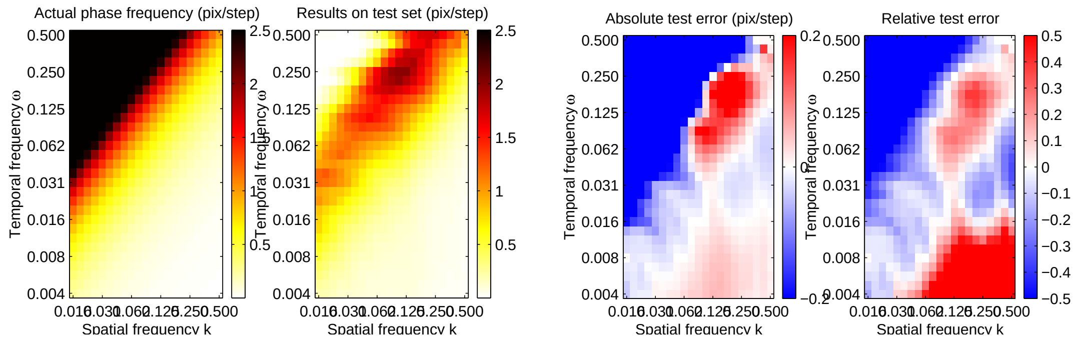


Figure 9: Results of SVM calibration. Leftmost panel shows actual phase velocities (ω/k). Second panel shows the phase velocities extracted from test movies at various values of ω and k. These movies were not in the training set. The final two panels show the calibration errors, in absolute terms (third panel), and as a fraction of the real velocities (rightmost panel). Calibration is seen to work well for velocities up to two pixels per timestep.

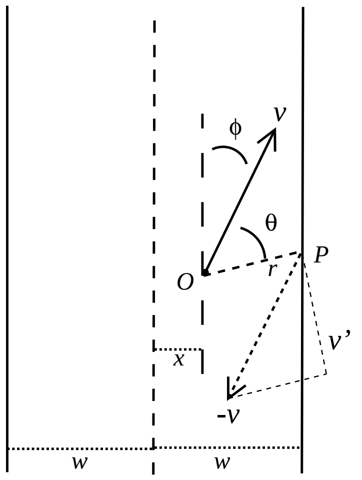


Figure 10: Moving through a straight tunnel.

Moving through a straight tunnel

Assume the fly is moving through an infinite tunnel, as shown in Figure 10. The motion is not necessarily parallel to the tunnel, nor is the fly located in the center of the tunnel.

Noticing that for parts of the eye that view the right wall:

$$\frac{w-x}{r} = \cos(90^\circ - \phi - \theta) = \sin(\phi + \theta),$$

one finds:

$$f(\theta) = \frac{v'}{r} = \frac{v}{w-x}\sin(\phi+\theta)\sin\theta \qquad \text{for } \theta > -\phi.$$

Similarly, one obtains:

$$f(\theta) = -\frac{v}{w+x}\sin(\phi+\theta)\sin\theta \qquad \text{for } \theta < -\phi.$$

The aim of the game is to obtain a robust estimate of x and ϕ from the observed $f(\theta)$, or rather, an estimate of $w - x - v\tau\sin\phi$, the estimated distance from the wall some time τ in the future. There doesn't seem to be a direct way of doing this, so I tried several integration kernels to search for a useful indicator. Specifically, I considered:

$$I_g = \int_{-\alpha}^{-\beta} fg\mathrm{d}\theta + \int_{\beta}^{\alpha} fg\mathrm{d}\theta$$

for $g = 1$, $g = \sin\theta$ and $g = \cos\theta$ and various values of $\alpha = 45^\circ \ldots 135^\circ$, and $\beta = 0^\circ \ldots \alpha$.

I found that a robust signal that determines which wall is most immediately dangerous can be obtained using $g = 1$, $\alpha = 90^\circ$, $\beta = 45^\circ$. Figure 11 shows that I_1 is generally positive when crashing into the right wall is imminent, or negative if crashing into the left wall is imminent.

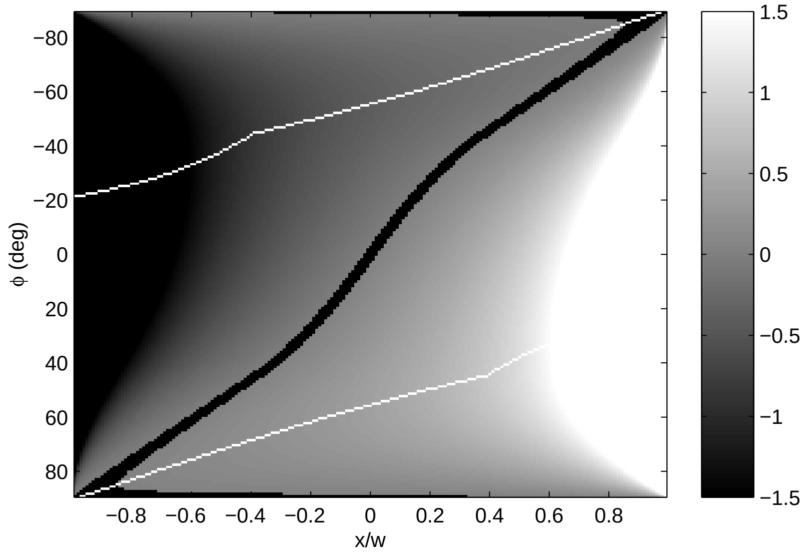


Figure 11: The value of the danger detector I_1 as a function of position x/w and heading ϕ. The black line marks the 'safe' heading for a given position. The white lines mark the edge of the region where the correction signal grows monotonically with heading. Only when approaching one of the walls almost head-on does I_1 give a false sense of security.

Moving towards a flat wall

Since the tunnel following signal becomes unreliable at large deviations ϕ, another signal must be computed to deal with looming walls. In the situation depicted in Figure 12, the flow as a function of retinal position is:

$$f(\theta) = \frac{v \sin\theta}{r} \frac{\cos(\theta - \phi)}{\cos\phi}.$$

In this situation it is useful to calculate

$$J_g = \int_{-\alpha}^{\alpha} f g \mathrm{d}\theta,$$

with kernels g as before.

For any value of α, one finds

$$J_1 \propto r^{-1} \tan\phi \qquad J_{\mathrm{sin}} \propto r^{-1}.$$

Thus, J_{sin} is a useful indicator of impending head-on collisions, and the sign of J_1 or I_1 may be used to choose which way to turn to avoid that collision. It seems reasonable to pick $\alpha \sim 45^\circ$, because that way the estimate is not too sensitive to walls on the side, and the interference between tunnel and head-on warning detectors is minimal. (Much smaller values would be bad, because optic flow near the center of expansion is very small leading to very noise estimates.)

Motor control

At every timestep, I_1 and J_{sin} are calculated. The results are average over 7 timesteps for added stability. Unless the average J_{sin} is larger than a critical value, I_1 is passed through a sigmoid to

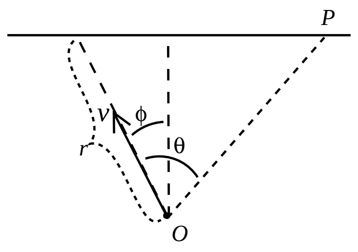


Figure 12: Moving towards a flat wall.

determine the desired correction to heading:

$$\delta\phi = \Phi \tanh(\alpha I_1),$$

where $\Phi = 35^\circ$ and α is manually tuned to obtain reasonable behavior. The correct is applied instantaneously, and for the next 30 timesteps the optic flow is ignored because it is known to be polluted by ego-rotation. (Correcting for this pollution would be a straightforward improvement of the algorithm.) The result is saccadic motion, not unlike a real fly's: straight stretches interspersed with sudden changes of heading. Of course, it would be more realistic to control angular acceleration rather than angle directly, but that would make the control harder, because ego-rotation cannot be ignored in that scheme.

If J_{sin} does exceed a critical value, the fly is apparently in immediate danger of crashing into a wall, and a much stronger escape maneuver is initiated:

$$\delta\phi = \Phi' \tanh(\alpha' I_1),$$

where $\Phi' = 45^\circ$ and $\alpha' = 4\alpha$.

Experiments

Maze navigation based on perfect flow estimation

Before applying the object avoidance functions I_1 and J_{sin} to the difficult problem of navigation based on estimated optic flow, I shall test them on a simpler problem: motion based on a perfect estimate of flow. Since my fly moves about in a computer generated world, presenting it with the true velocity map is trivial. Figure 13 shows motion through a typical maze. It is seen that the navigation is consistent and appropriate. Figure 14 demonstrates stability of trajectories.

To test robustness of the method, I added Gaussian noise to the velocity map with the noise power equal to the signal power. This barely affected the trajectory followed (Figure 15). Even adding Gaussian noise to the motor output ($\sigma = 10^\circ$ at every decision point) did not cause the fly to crash into walls, although the trajectory is now more erratic (Figure 16).

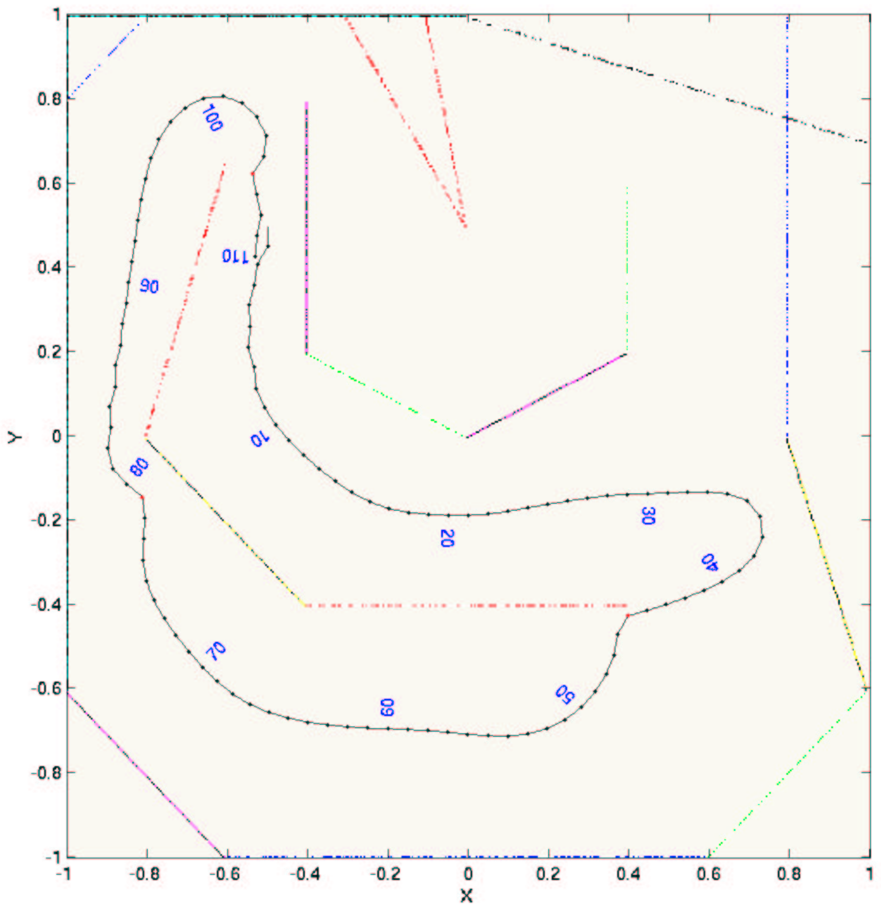


Figure 13: Maze navigation based on perfect flow estimation. Decision points are marked by black dots and enumerated by blue numbers. The fly is seen to follow the center of local tunnels quite well, except when a wall is approached from the sharp side (decision points 40–44). As observed by Abbott [1884], sharp edges are dangerous, because they are essentially invisible. As soon as the fly notices the sharp wall, a panic reaction (red dot at 45) is initiated, which takes it away from the wall. A number of other panic reactions also occurred (red dots). These are less obviously appropriate.

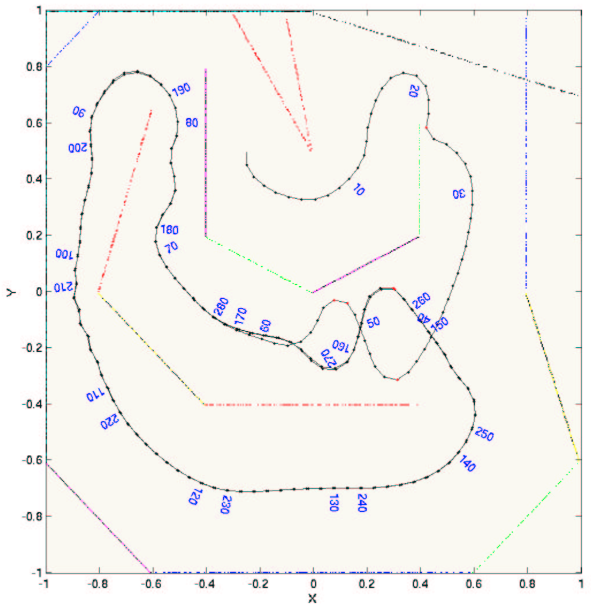


Figure 14: The trajectory resulting from another starting point. After a single loop through the world a stable limit cycle is reached.

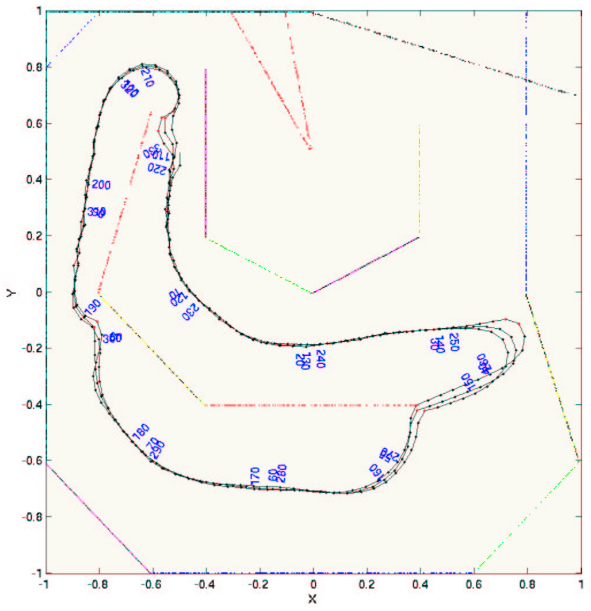


Figure 15: Maze navigation based on perfect flow estimation polluted by noise of equal power. The noise hardly affects the trajectory at all (cf Figure 13). This is probably the result of the integration involved in calculating I_1 as well as the averaging over several timesteps.

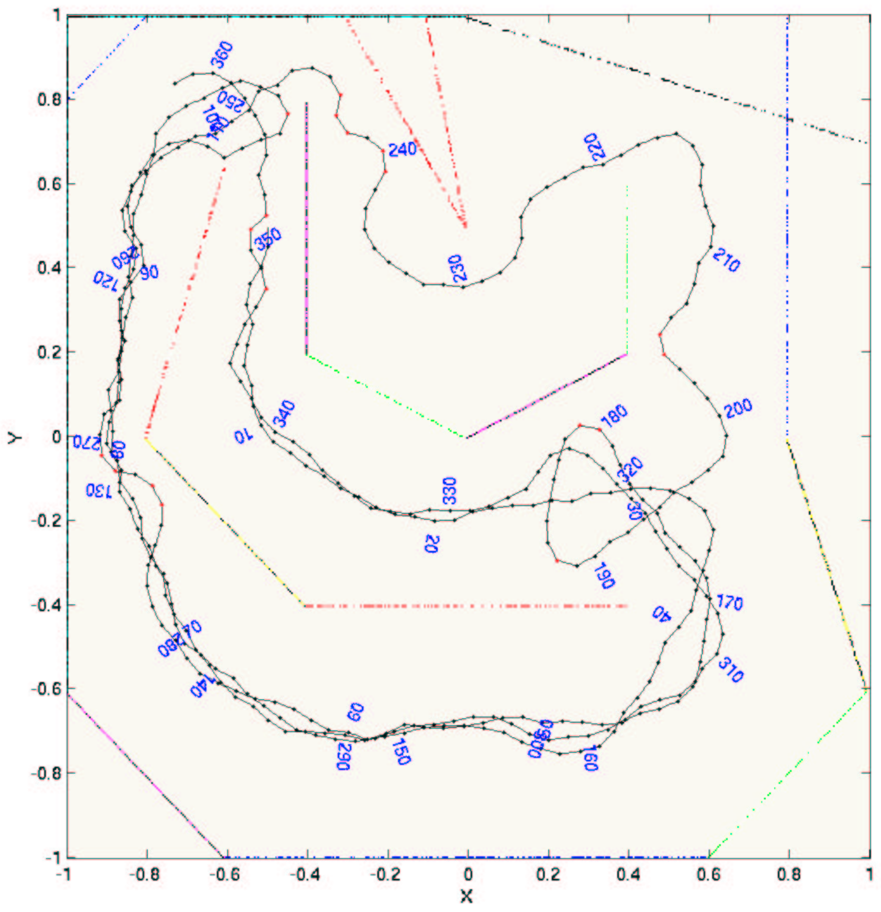


Figure 16: The same polluted flow estimation as in Figure 15, with added noise in the output stage: Adding 10° worth of Gaussian noise to the heading corrections $\delta\phi$ does affect the trajectory, but it doesn't prevent the fly from avoiding walls, although the number of panic reactions is significantly increased. Unsurprisingly, the trajectory is rather more erratic.

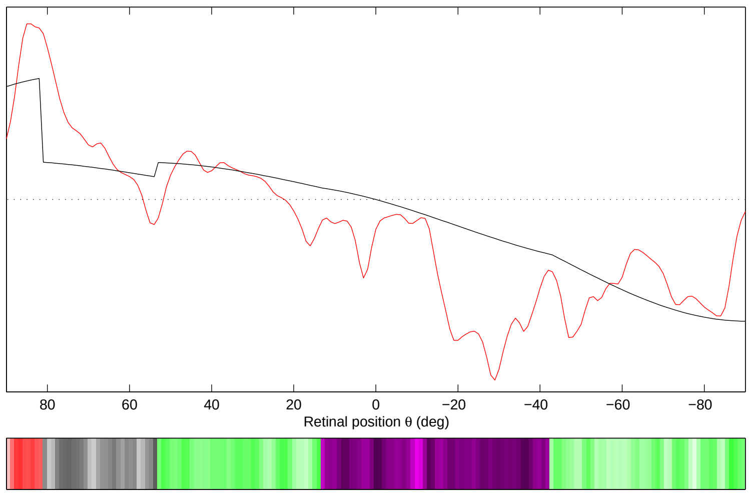


Figure 17: Estimated optic flow (top, red) and actual velocity field (black) while the fly navigates the maze. While the estimate roughly matches reality, it is clear that the noise is substantial. The retinal image at the time this estimate was made is shown below the graph. The result of anti-alias filtering is clearly visible in the blurring of the perceived patterns compared to Figure 7. Retinal coordinates are $+90°$ at the extreme left through $-90°$ at the extreme right.

A maze with randomly patterned walls

Finally, I let the fly loose in a world where it had to rely on its own flow estimation. The walls in the worlds used in Figures 17 sqq are decorated with the same kind of random block patterns used for training the detectors (Figure 7), with 250 blocks per line segment.

Unlike in the real world, flying through walls is not actively prevented by the physics of my simulation. It would perhaps be more realistic to have the fly bounce, like real flies do on windows, but I didn't want to waste CPU time on external collision detection. Thus, in the following runs, violations of walls happen occasionally.

Discussion

Overall, the fly manages to avoid walls reasonably well. In straight sections of tunnels, the flow based algorithm performs adequately. In narrow spots the fly tends to panic and fly through walls, and it also has a tendency to see sharp edges too late. In wide open areas the fly tends to lose its sense of direction. This is not as big a problem as the other two, because it can be attributed to the fly not having any goal in mind except avoiding walls. Buzzing around in circles is an acceptable response to that challenge.

Tests with God-given perfect flow values showed that my simple navigation algorithm is quite robust against (white) noise in the flow estimate, as well as in the motor output. Noise robustness was not tested with estimated flow, because the noise in flow data was already quite large (Figure 17).

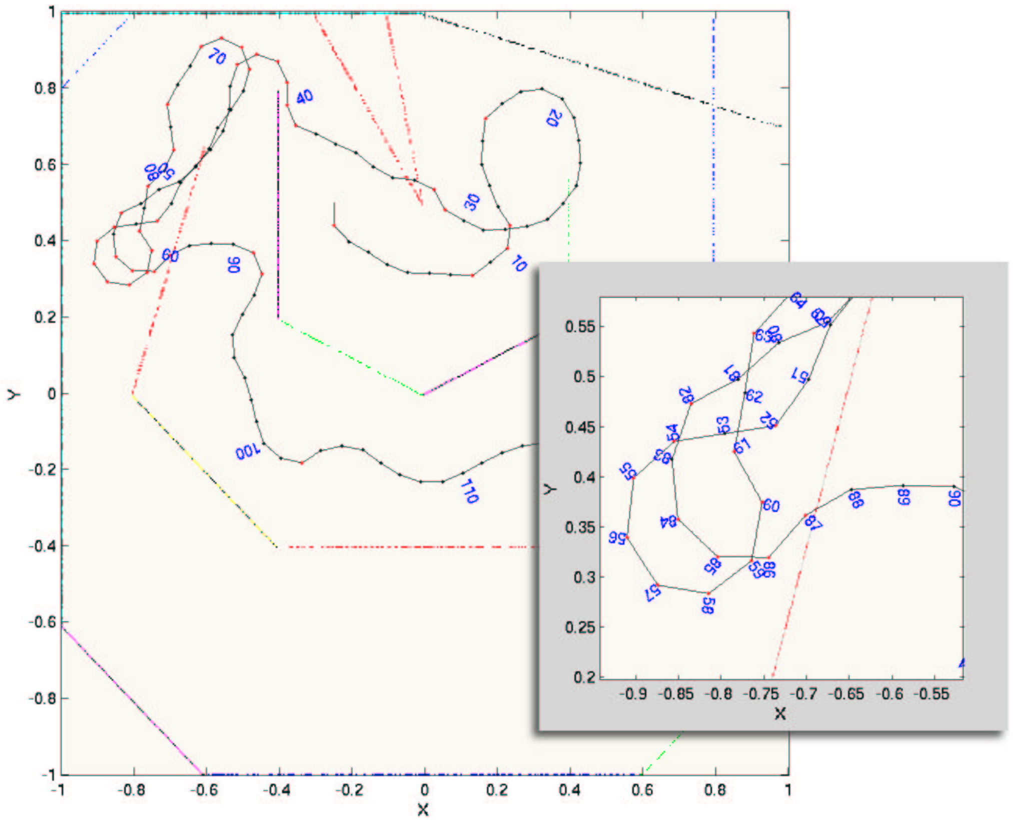


Figure 18: Maze navigation based on realistic flow estimation. The fly is considerably more panicky, and has more trouble with pointy edges. In fact, it is seen to disrespect walls more than once in this short sequence as a result of approaching a sharp angle. One other violation (near decision point 87, see overlay) can be understood by noticing that the immediately preceding panic rotation (86) was not strong enough to take the fly away from the wall. Since the wall's texture has no power at very high frequencies, it appears (nearly) uniform at extreme proximity, causing the velocity estimators to fail miserably.

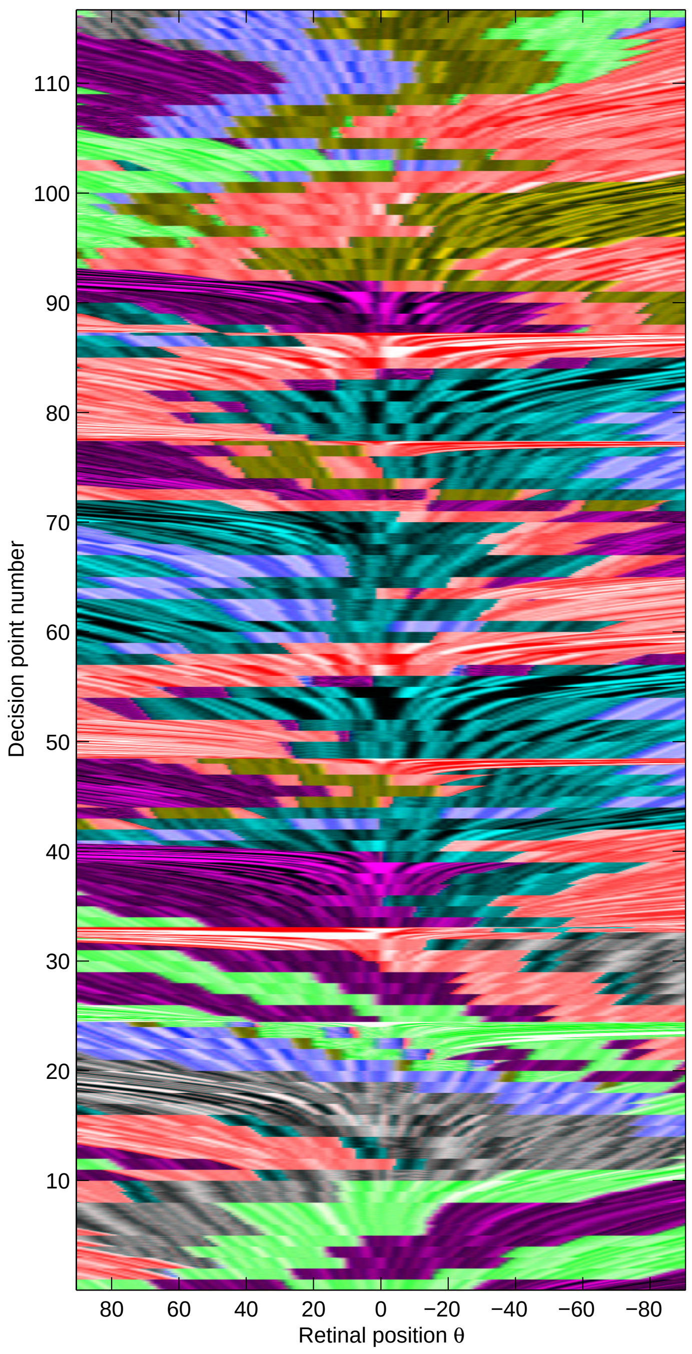


Figure 19: This is what the fly saw while performing the navigation depicted in Figure 18: slide a ruler from bottom to top to view this graph as the movie it really is. (Because we are used to seeing more flow below us than above us, I found it easier to view this "1D movie" with time flowing upwards, hence the change of convention.)

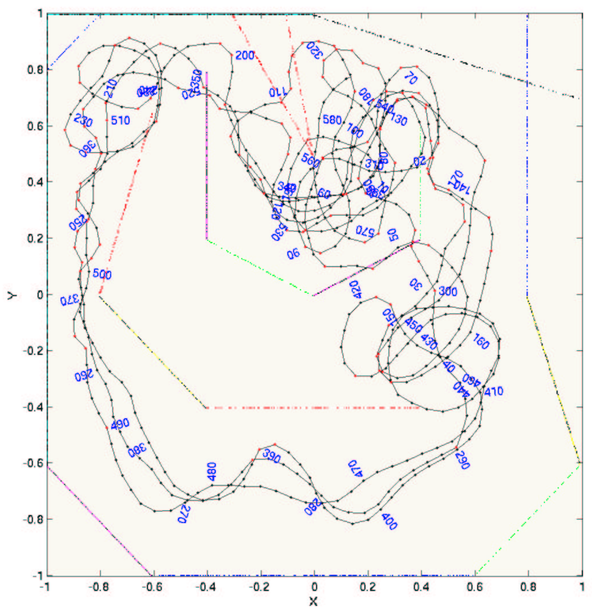


Figure 20: Another run with the same parameters (but differently patterned walls). In straight tunnels the fly does quite well, but it has severe trouble with sharp points and in narrow spots. In wide open areas the motion is quite erratic. If the fly had a goal in mind, its motion might become straighter

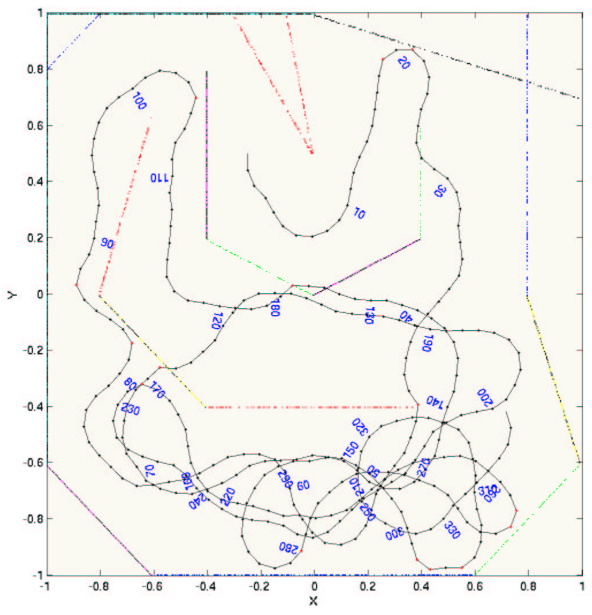


Figure 21: Increasing the threshold for panic behavior while increasing the high-frequency content of the wall patterns does not make for an obvious change in trajectories (but the number of panic responses is reduced signigificantly).

Stumbling blocks

When I first proposed this project, I had much higher levels of behavior in mind: I would have liked to replicate behaviors described by Kern et al. [1997], Srinivasan et al. [1999] and Srinivasan et al. [2000]. However, it turned out that even constructing a flow estimator good enough for navigation was a significant challenge, so I focused on that. More goal oriented behaviors would be an exciting follow-up. Below, I will highlight some of the problems I ran into.

Sinusoidal worlds

The first generation of flow detectors I constructed were trained on sinusoidal patterns. They looked extremely promising, exhibiting very linear responses over a wide range of velocities. However, when I tried them on a world with richer frequency contents, such as random patterns, I found that they failed miserably.

Gaussian patterns: 2D vs 3D

Before settling on sharply defined patterns with anti-aliasing filters on the fly, I considered walls patterns that were low-pass filtered white noise. These proved to be very hard to navigate, because at large distances spatial aliasing did kick in, while at short distances the walls showed too little texture for the filters to latch on to. In a way, this problem is exacerbated in 2D compared to 3D, since 1D gratings have a much larger risk of exhibiting low-energy patches than 2D patterns do. Thus, as for gradient methods, a 3D implementation might actually be easier to get to work than this 2D one.

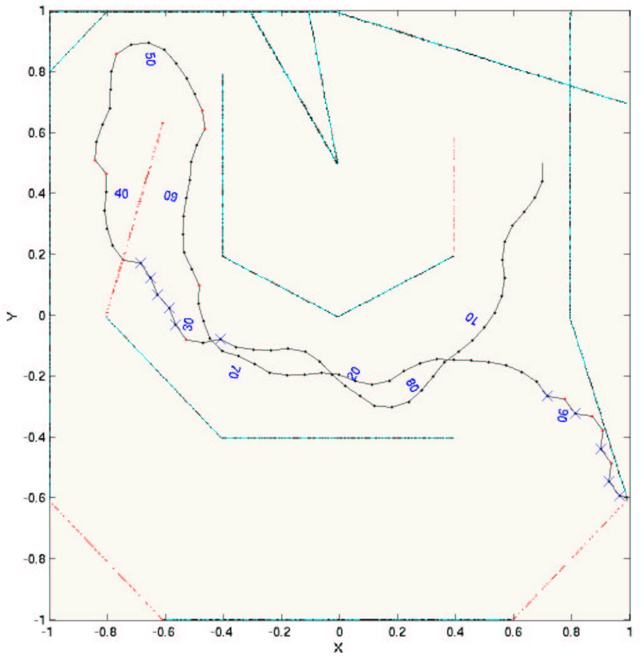
Directions and musings

Fewer processing elements

It is well known that real flies use rather few neurons in the highest stages of optic flow balancing, culminating in a single identified cell, H1. My construction seems extremely clunky in comparison, and yet the fly performs much better. It would be interesting to test whether having far fewer filters has a negative performance impact on object avoidance. The smoothness of the estimated flow depicted in Figure 17 suggests that this might not be the case.

Higher level behaviors

It would be attractive to add simple goal seeking to the present model. This could easily be done by modifying the panic response to looming stimuli: if the looming object is attractive, fly towards it rather than away, otherwise perform the evasive action currently implemented. A crude version of this approach works, as evidenced by the trajectory and retinal movie shown in Figure 22, where the fly has been given an innate liking for red things.



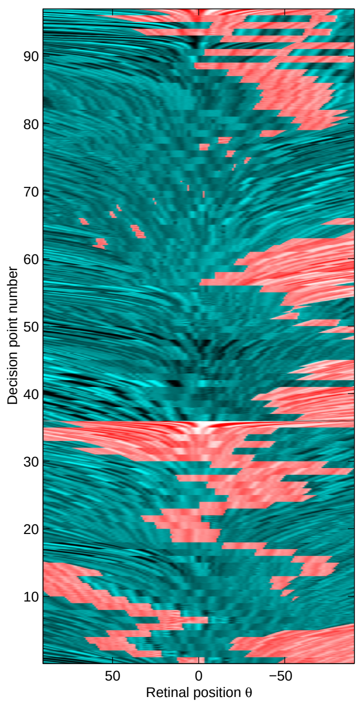


Figure 22: A trajectory of a fly with an innate liking for redness. It behaves the same as the flies studied before, except when the majority of the central part of its visual field is filled by a red object — in that case it turns towards the object rather than away from it. Blue crosses mark points along the trajectory where such goal seeking dominated the behavior. Since the fly was closer to a blue–green wall than to the target when the second target came within view, it lands right at the edge: whenever it is flying towards the red, it turns towards the nearest wall as per its simplistic goal seeking algorithm, and when it finds itself flying towards that wall, it panics and rotates away from it. Etc. The decision points at which the targets were reached (36 and 97)stand out clearly in the movie by the stretching of the pattern.

Biology vs software, simulation of physics

Once again, I had to learn the hard way that the world is very good at representing itself, and that simulating it takes a lot of effort. It turns out that the majority of CPU time taken by this project was used to do things that physics does automatically for real flies: spatial anti-aliasing and temporal delay filters. Studies of the sort described here would be a lot easier with hardware that takes care of physics more naturally. Efforts to equip robots with compound eyes ought to have a big impact on the modelling community.

References

E. A. Abbott. *Flatland*. 1884.

E.H. Adelson and J.R. Bergen. Spatio-temporal energy models for the perception of motion. *J. Opt. Soc. Am.*, A2:284–299, 1985.

R. Kern, M. Egelhaaf, and M.V. Srinivasan. Edge detection by landing honeybees: Behavioural analysis and model simulations of the underlying mechanism. *Vision Res.*, 37(15):2103–2117, 1997.

M.V. Srinivasan, S.W. Zhang, J. Berry, K. Cheng, and H. Zhu. Honeybee navigation: linear perception of short distances travelled. *J. Comp. Physiol. A*, 185(3):239–245, 1999.

M.V. Srinivasan, S.W. Zhang, J.S. Chahl, E. Barth, and S. Venkatesh. How honeybees make grazing landings on flat surfaces. *Biol. Cybern.*, 83(3):171–183, 2000.