A. Supplementary Materials

A.1. Unique Frequency Encoding

The simplest method of associating glints with their respective light sources is using a unique frequency per light. This is the method used in previous works on ALMs and is illustrated in Algorithm 1. We propose integrating the weight of each event transition w by a lowpass filter, in a similar manner to lowpass image reconstruction [22]. Essentially, each time a transition is registered, the previous value in the filter is decayed proportionally to the time since the last update, via the update formula

$$E(w_n) = E(w_{n-1})k + P(dt|f) \tag{3}$$

$$k = e^{-dt*f*\tau} \tag{4}$$

where τ is the time constant of the lowpass filter (cutoff frequency = $\frac{1}{\tau}$).

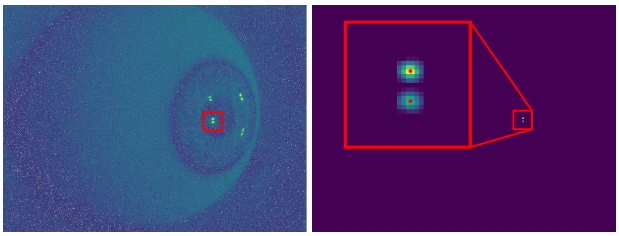
Frequency filter normalization In order to make our frequency filters more interpretable than a 'raw' expectation map as in [3], we can normalize the filter images using the ideal maximum value of the filter. Concretely, supposing that the event filter was observing an ideal pulse with frequency f, the update weight w for each pulse would be the mean likelihood of the sampling distribution $\mu_f = \mathcal{N}(0; 0, \sigma)$. Under this circumstance, the closed form definition for 3 is

$$E(n) = k^n E_0 + \frac{\mu_f(k^{n+1} - 1)}{k - 1}. \tag{5}$$

Algorithm 1: Frequency filtering algorithm (ec=event count, ts=timestamp, pol=polarity).

Input: Events $\mathcal{E}$, f, σ, λ_c, s_p
Output: I_f

```
forall e = {x, y, t, p} ∈ E do
    if p == curr_pol[x, y] then
        curr_ec[x, y] += 1;
    else
        next_ts[x, y] = t;
        next_pol[x, y] = p;
        next_ec[x, y] += 1;
    end
    if next_ec[x, y] > λc next_pol[x, y] == sp then
        dt=next_ts[x, y]- curr_ts[x, y];
        If[x, y] = N(dt; 1/2f, σ²);
        curr_pol[x, y] = p;
        curr_ec[x, y] = next_ec[x, y];
        curr_ts[x, y] = next_ts[x, y];
        next_pol[x, y], next_ec[x, y], next_ts[x, y] = 0;
    end
end
```



(a) Scene (b) Frequency Filter

Figure 12: In 12a four glint pairs flash at various frequencies. 12b shows the frequency filter response for the glints. A 2-mean GMM is applied to detect glint centers (red points).

This equation has a limit

$$\lim_{n\to\infty} E(n) = \frac{\mu_f}{1-k}, \tag{6}$$

which gives us the maximum value the filter can take. We divide by this maximum value to scale expectation maps to the range $[0, 1]$.

Glint centroiding Because the frequency filter doesn't rely on the synchronization pulse, it needs an additional step to distinguish the 2 glints within a glint pair (which operate at the same frequency, as they compensate each other). We use a 2-mean Gaussian Mixture Model (GMM) to find the centers of each glint pair in the expectation maps with subpixel accuracy (see Figure 12).

Operations per Event Note that the large majority of events never passes the `if` statement on line 9 of Algorithm 1. Only when an event of the *opposite* polarity to the current polarity is observed, is a transition registered and this condition triggered. For a single pulse of events, this should only occur *once* and only for those pixels observing the beacon stimulating the pulse. In our experiments, glints were always $\leq 120\,\mathrm{pix}$ in size. A typical pulse of events contained around $3500\,\mathrm{ev}$ at an upper bound of $500\,^\circ/\mathrm{s}$ ocular motion (see Figure 9, where an event-rate of $\approx 3.5\,\mathrm{Mev/s}$ is measured with 1000 pulses/s). Most of these events only require 3 FLOPs as they do not pass the second `if` statement, which is only passed $\approx$ 120 times. The second `if` statement requires 10 FLOPs, so on average $\approx \frac{3\times3400+10\times120}{3500} = 3.3$ FLOPs/ev

A.2. Binary Coded Glint Tracking Algorithm

A breakdown of the algorithm illustrated in 3.3 is presented in Algorithm 2.

Algorithm 2: Update algorithm for Binary Glint tracker (gm0=glint-map `0` , gm1=glint-map `1`).

Input: gm0, gm1, x, b, λ_p
if x == *none* **then**
  patch = `get_patch` (topleft=(0, 0), size=gm0.shape);
else
  patch = `get_patch` (center=x, size=(λ_p,λ_p));
end
if b == *none* **then**
  new_x_0, w0 = `find_glint` (gm0, patch);
  new_x_1, w1 = `find_glint` (gm1, patch);
  if *w0* > *w1* **then**
    return new_x_0;
  else
    return new_x_1;
  end
else
  if b == *0* **then**
    new_x, w = `find_glint` (gm0, patch);
    return new_x;
  else
    new_x, w = `find_glint` (gm1, patch);
    return new_x;
  end
end

A.3. Sensor Noise

We claim that our proposed method is resistant to sensor noise. To demonstrate this, we present a histogram of transition periods for a 60 s recording with the lens cap on (Figure 13). It is clear from this experiment that typical camera noise operates almost entirely in the 0-250 Hz range.

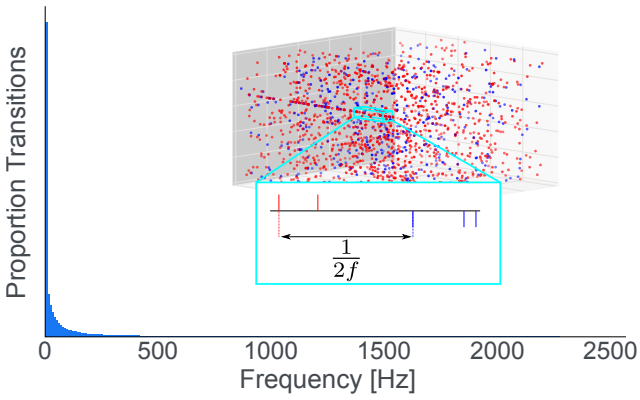


Figure 13: Dark blue bars show the histogram of the frequencies implied by the transitions of lens-cap noise events.

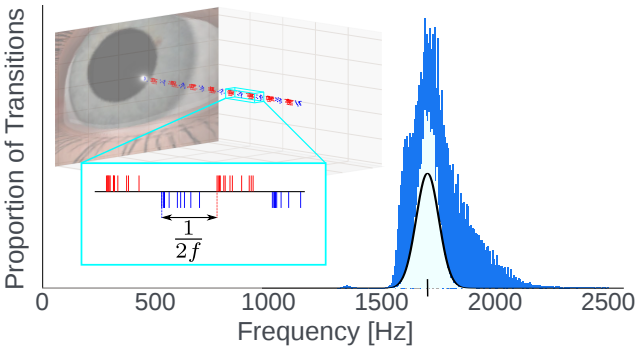
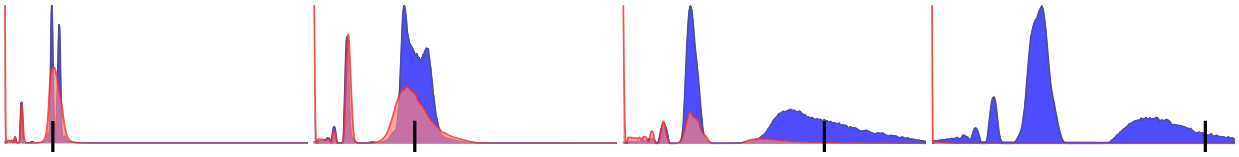


Figure 14: Dark blue bars show the histogram of the frequencies implied by the transitions of the event pulses of a light flashing at 1700 Hz for 10 s (mean $\mu_d = 1741$ Hz, stdev $\sigma_d = 129$ Hz). The sampling function (1) models this as a normal distribution $\mathcal{N}(\mu_s = 1700, \sigma_s = 80)$ (black curve). Multiplying the data with the sampling function reduces the bandwidth consumed (light blue), as $\sigma_s < \sigma_d$.

A.4. Event Camera Bandwidth

As identified in Section 4.5, the bandwidth required to robustly detect a beacon flashing at a fixed frequency is in the low hundreds of Hz for modern event sensors. This limits the number of beacons that can be robustly supported in a one-frequency-per-beacon encoding scheme to just ≈ 5. Figure 15 shows that event to achieve this, the target event sensors needs to be tuned for the task. The distribution of frequencies implied by the transition periods recorded observing a beacon flashing at a fixed frequency, shows that beyond 1 kHz the standard parameterisation fails entirely. Notice that even in our tuned camera, the peak of the distribution stalls at around 1800 Hz, implying that this is the limit of our camera's ability to accurately detect high frequency pulses. Also noteworthy is the smaller peak at the harmonic frequencies of the base frequency; since one 'missed' transition implies half the base frequency, and two 'missed' transitions imply on third the base frequency *etc*., there are peaks at these locations.

Effect of sampling function Since the frequency of the observed stimulus is estimated by inspection of the transition period between negative and positive events, variation in this period causes the recorded transitions to fall within a distribution D. This distribution is quite spread at higher frequencies, with σ in the low hundreds of Hz. Sampling with (1) is equal to a multiplication with D, giving a new distribution of weighted transitions: $W = \mathcal{N}(\frac{1}{2f}, \sigma_s) \times D$ (see Figure 14). Since σ_s is chosen to be less than the standard deviation of D, σ_d, the sampling distribution is the limiting factor that sets the bandwidth consumed by each



(a) 500 Hz (b) 1 kHz (c) 2 kHz (d) 3 kHz

Figure 15: Histograms of detected frequency (from 0 Hz to 3000 Hz) for light source flashing at 500 Hz, 1 kHz, 2 kHz, 3 kHz, with optimised biases (blue) and default biases (red). Black bars on the x axis denote the light frequency. Note the significant spike in detections at half of the target frequency - this occurs because a missed transition implies half of the frequency.

flashing stimulus, *i.e*. setting a small value for σ_s increases the available bandwidth. However, setting σ_s too small risks removing too much of D, which is the signal being measured. Therefore, there exists a tradeoff between available bandwidth and measurement accuracy.

The sampling distributions should not overlap much, since this introduces ambiguity, where the same transition can trigger a similar response in multiple frequency filters. This ultimately restricts the number of frequencies that can be supported on a given bandwidth.

References

[22] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. Continuous-time intensity estimation using event cameras. In *Asian Conference on Computer Vision (ACCV)*, pages 308–324, Dec. 2018.