Time Delay Estimation in Gravitationally Lensed Photon Stream Pairs

Michał Staniaszek
Supervisor: Peter Tiňo

June 21, 2013

this is the abstract

1 Introduction

- explain the project in layman's terms

2 Background

- Ideas behind the project
- what it's useful for
- what gravitational lensing and time delay are

3 Photon Stream Simulation

In the early stages of the project, we developed a subsystem which could be used to generate simulated photon stream data to use for the development and testing of the rest of the project. The only property of the photons which we are interested in is their arrival time at our capture device, so the simulator should produce some vector $\Phi = [\phi_0, \ldots, \phi_N], \phi_n \in \mathbb{R}$, where ϕ_n is the arrival time of the nth photon. In order to generate arrival times, we represent the source as some random variable X, which defines the average number of photons per unit time that arrive at the capture device, and whose varies according to the characteristic function of the source object.

The characteristic function of X is modelled as a non-homogeneous Poisson process (NHPP) with continuous function of time, $\lambda(t)$, known as the rate function. The rate function can be specified either by providing an expression which is a function of t, or by sampling from a randomly generated function. Random functions are constructed by uniformly distributing M Gaussians across the interval $[t_0, T]$ in which arrival times are to be generated. Each Gaussian g_i is defined by its mean μ_i, its width σ_i, and its weight w_i, which determines its height. The means of successive Gaussians are separated by some distance Δt, such that $\mu_{m+1} = \mu_m + \Delta t,$ where $\mu_0 = 0$. Greater variation in the functions is introduced by sampling the weights w_i from a uniform distribution $U(-1, 1)$ and scaling them by some multiplier. The value of the randomly generated function at some time t is computed by

$$\lambda(t) = \sum_{i=0}^{M} w_i \cdot e^{-(t-\mu_i)^2/2\sigma_i^2} \tag{1}$$

which is a weighted sum of Gaussians.

Having defined or constructed $\lambda(t)$, photon arrival times are generated from a homogeneous Poisson process (HPP) with constant rate λ, using inverse transform sampling. The waiting time to the next event in a Poisson process is [**1998art**]

$$t = -\frac{1}{\lambda}\log(U) \tag{2}$$

where $U \sim (-1, 1)$ Using this calculation, it is possible to generate a realisation of a HPP for an interval of any finite length. This provides a base which can be extended to generate events from NHPPs.

To generate events from an NHPP with a rate function $\lambda(t)$, we use a technique called thinning, where we generate a large number of values, and then throw some of them away based on some criterion. In the case of the NHPP, we generate events from a HPP with a rate parameter λ, where $\lambda > \lambda(t)$ for $0 <= t <= T$. In other words, the homogeneous lambda value must be larger than the value of the rate function for the whole time the process runs. First, two random values are independently sampled from a uniform distribution, $U_1, U_2 \sim U(0, 1)$. The first number, U_1, is used in (2) to find the next event time from the homogeneous process governed by λ. Using the time t generated from that, the value of $\lambda(t)$ is calculated. If $U_2 \leq \frac{\lambda(t)}{\lambda}$, then the event is kept. The closer $\lambda(t)$ is to λ, the more events will be kept, and so the number of events generated depends on the shape of the rate function.

Algorithm 1 Simulating T Time Units of a NHPP by Thinning

Require: $\lambda \geq \lambda(t), t_0 \leq t \leq T$

$\Phi = \emptyset$, $t = t_0$, $T =$ interval length
while $t < T$ **do**
  Generate $U_1 \sim U(0, 1)$
  $t = t - \frac{1}{\lambda}\ln(U_1)$
  Generate $U_2 \sim U(0, 1)$, independent of U_1
  if $U_2 \leq \frac{\lambda(t)}{\lambda}$ **then**
    $\Phi \leftarrow t$
  end if
end while
return Φ

4 Function Estimation

- Basic explanation of the IWLS estimator
- slightly more in-depth stuff for the development of our estimators

4.1 KDE

The second function estimation method implemented was a kernel density estimator, which use *kernels* to estimate the probability density of a random variable. A kernel is simply a weighting function, which affects how much a given sample is considered when constructing the function estimate. Since the photon stream data is assumed to be generated by a source whose variability is defined by some random variable, the event times are a sample drawn from the PDF of that variable. We use a Gaussian kernel

$$K(t, \mu) = e^{-(t-\mu)^2/2\sigma^2} \tag{3}$$

to estimate the PDF, centring a kernel at each photon arrival time a by setting $\mu = a$. The width of the kernel depends on some fixed value σ. We perform a Gauss transform on the N kernels, finding the contribution of all the kernels at M points in time, from which we get an estimate $\hat{\lambda}(t)$ of the characteristic function.

$$\hat{\lambda}(t_i) = \sum_{j=1}^{N} K(t_i, \mu_j), \quad i = 1, \ldots, M \tag{4}$$

Using a larger M gives a higher resolution. Depending on the value of σ used, $\hat{\lambda}(t)$ will be some multiple of the actual function $\lambda(t)$. Thus, the final step is to normalise $\hat{\lambda}(t)$. We split the stream data into B bins with midpoints b and calculate the bin count x for each. We start with the normalisation constant η at a low value, and gradually increase it to some threshold, finding

$$\sum_{i=1}^{B} \log\left(\frac{\phi^x e^{-\phi}}{x!}\right), \quad \phi = \eta \cdot \hat{\lambda}(b_i) \tag{5}$$

for each value of η. The value of η which maximises this sum of log Poisson PDFs is used to normalise $\hat{\lambda}(t)$ in subsequent computations. Figure **??** shows an example of a kernel density estimate, and displays a weakness in the estimator. As one moves towards the start or end of the interval, fewer Gaussians make a noticeable contribution to the function calculation, resulting in a drop-off of the estimate.

4.2 Baseline

As mentioned in the previous section, the piecewise IWLS estimator gives us a piecewise disjoint estimate of the function, but we would like one which is piecewise continuous. In order to do this, the end of each interval estimate must meet the start of the next. The estimate returned by the piecewise estimator has several breakpoints—points where the start of one sub-interval and the end of another meet. If there are L lines that make up the estimate, there will be $R = L - 1$ breakpoints. At each of these breakpoints r, we calculate the value of the previous and subsequent function estimates f, and find their midpoint m with

$$m_i = \frac{f_i(r_i) + f_{i+1}(r_i)}{2}, \quad 0 \leq i < R \tag{6}$$

The value of m is calculated for each breakpoint. Midpoints are not calculated at time 0 and time T. Instead, the function values at those points are used. Each sub-interval is now represented by a point p at the start and q at the end, each with an x and y coordinate. With these points, we can recalculate each sub-interval estimate f of the form $y = \hat{a} + \hat{b}x$ by replacing y with p_y and x with p_x, and recalculating the gradient $\hat{b}$ and intercept $\hat{a}$ with

$$\hat{b} = \frac{q_y - p_y}{q_x - p_x} \tag{7}$$

$$\hat{a} = p_y - \hat{b} \cdot p_x \tag{8}$$

In this way, each sub-interval estimate links points p and q, giving us a piecewise continuous function estimate, and this step completes the first function estimation method. Figure **??** shows an example of a piecewise and baseline estimate.

5 Time Delay Estimation

5.1 PDF Method

The second method of estimation is using probability density functions. As before, we guess a value of Δ between $-\Delta_{\max}$ and $+\Delta_{\max}$ and shift $\hat{\lambda}_2$ by that amount. However, we know that there must be a single characteristic function, and we want to see how well our estimate of that matches the bin counts in each stream. We make an "average" function $\bar{\lambda}$ by combining the two function estimates we have, $\hat{\lambda}_1$ and $\hat{\lambda}_2$ (which is shifted by Δ).

$$\bar{\lambda}(t) = \frac{\hat{\lambda}_1(t) + \hat{\lambda}_2(t+\Delta)}{2} \tag{9}$$

The point on $\bar{\lambda}$ at time t is the midpoint between the values of the two estimates at that time. Once we have $\bar{\lambda}$, we can assign some score to the current estimate of the value of Δ.

$$\begin{aligned} \log P(S_A, S_B \mid \bar{\lambda}(t)) = \sum_{t=\Delta_{\max}}^{T-\Delta_{\max}} & \log P(S_A(t) \mid \bar{\lambda}(t)) \\ & + \log P(S_B(t+\Delta) \mid \bar{\lambda}(t)) \end{aligned} \tag{10}$$

Here, we calculate the probability that the function $\bar{\lambda}$ is the characteristic function of the two streams S_A and S_B. The streams are split into bins, and the log probability of the number of events in each bin given the value of λ calculated for that bin is computed and summed over all bins, as in Equation (??).

The calculation of λ is slightly more complicated than just taking its value at the midpoint of each bin. Since we are considering a number of events occurring in a given interval, we must consider the value of λ for the same interval. In order to do this, we use a discrete approximation of integrating $\lambda(t)$ over the interval.

$$\lambda_{a,b} = \int_a^b \lambda(t)\, dt \tag{11}$$

In the approximation t is incremented by some finite step for each successive value. The smaller the value of the step the more accurate the approximation of $\lambda_{a,b}$ becomes. As with the previous estimator, the estimate is made in two stages, first with a coarse pass over the values of delta to compute an initial estimate, and then a finer second pass around the first estimated value in order to refine the estimate.

5.2 Area Method

The first of the two methods uses a very simple metric to estimate the time delay. By taking the two function estimates, we can attempt to match up the two functions so that they "fit together" best. The goodness of fit can be determined by the area between the two functions $\hat{\lambda}_1$ and $\hat{\lambda}_2$, calculated by

$$\begin{aligned} d(\hat{\lambda}_1, \hat{\lambda}_2) &= \int (\hat{\lambda}_1(t) - \hat{\lambda}_2(t+\Delta))^2\, dt \\ &\approx \frac{1}{N} \sum_{i=1}^{N} (\hat{\lambda}_1(t) - \hat{\lambda}_2(t+\Delta))^2 \end{aligned} \tag{12}$$

for each value of Δ. Our estimate of Δ is set to the value at which $d(\hat{\lambda}_1, \hat{\lambda}_2)$ is minimised. Rather than using an integral to get the exact area between the functions, we use a less computationally expensive discrete approximation.

6 Experimental Results

- general explanation of the experiments performed
- how was model selection done
- what sort of data were experiments performed on

7 System

- very brief explanation of the system features

8 Conclusion

- some suggestions for extensions