Resolution Limit of Single-Photon LiDAR

Supplementary Material

Contents

6. Proof of Theorems

In this section, we present the proofs of the theorems. To clarify the contributions of this paper, we add comments to each proof to highlight whether this is based on existing theory or it is a new proof.

6.1. Axioms

We first state the three axioms for inhomogeneous Poisson processes [5].

Definition 1. *Axioms for inhomogeneous Poisson process*

- *The probability of one occurrence in an infinitesimal interval Δt is given by*

$$\mathbb{P}[1, \Delta t] = \lambda(t)\Delta t, \qquad \Delta t \to 0. \tag{31}$$

- *The probability of more than one occurrence in Δt is zero for $\Delta t \to 0$. Therefore, $\mathbb{P}[0, \Delta t]$ is the complement of $\mathbb{P}[1, \Delta t]$, which will give us*

$$\mathbb{P}[0, \Delta t] = 1 - \lambda(t)\Delta t, \qquad \Delta t \to 0. \tag{32}$$

- *The number of occurrences in any interval is independent of those in all other disjoint intervals.*

6.2. Proof of Theorem 1

✠ *Remark: This proof is adopted from Bar-David [1] with new elaborations provided for each step.*

Consider a set of time stamps $-T \le t_1 < t_2 < \ldots < t_M \le T$. For each timestamp t_j, we consider an infinitesimal width $\pm\Delta t_j/2$. The probability that one and only one photon falls in each of the intervals and none outside is

$$\begin{aligned}
&p(\mathbf{t}_M, M)\Delta t_1 \Delta t_2 \ldots \Delta t_M \\
&\quad = \mathbb{P}[\text{observing one and only one in each interval}] \\
&\quad = \mathbb{P}[0, (-T, t_1 - \tfrac{1}{2}\Delta t_1)] \\
&\qquad\qquad \times \mathbb{P}[1, (t_1 - \tfrac{1}{2}\Delta t_1, t_1 + \tfrac{1}{2}\Delta t_1)] \\
&\qquad\qquad \times \mathbb{P}[0, (t_1 + \tfrac{1}{2}\Delta t_1, t_2 - \tfrac{1}{2}\Delta t_2)] \\
&\qquad\qquad\qquad \ldots \\
&\qquad\qquad \times \mathbb{P}[1, (t_M - \tfrac{1}{2}\Delta t_M, t_M + \tfrac{1}{2}\Delta t_M)] \\
&\qquad\qquad \times \mathbb{P}[0, (t_M + \tfrac{1}{2}\Delta t_M, T)] \\
&\quad = \mathbb{P}[0, (-T, t_1 - \tfrac{1}{2}\Delta t_1)] \\
&\qquad\qquad \times \lambda(t_1)\Delta t_1 \times \mathbb{P}[0, (t_1 + \tfrac{1}{2}\Delta t_1, t_2 - \tfrac{1}{2}\Delta t_2)] \\
&\qquad\qquad\qquad \ldots \\
&\qquad\qquad \times \lambda(t_M)\Delta t_M \times \mathbb{P}[0, (t_M + \tfrac{1}{2}\Delta t_M, T)]
\end{aligned}$$

As $\Delta t_j \to 0$, the intervals merge to become $(t_1 + \frac{1}{2}\Delta t_1, t_2 - \frac{1}{2}\Delta t_2) \to (t_1, t_2)$. Therefore,

$$\begin{aligned}
&\mathbb{P}[0, (-T, t_1 - \tfrac{1}{2}\Delta t_1)] \times \ldots \times \mathbb{P}[0, (t_M + \tfrac{1}{2}\Delta t_M, T)] \\
&\;= \mathbb{P}[0, (-T, t_1)] \times \mathbb{P}[0, (t_1, t_2)] \ldots \times \mathbb{P}[0, (t_M, T)] \\
&\;= \mathbb{P}[0, (-T, T)] \\
&\;= \exp\left[-\int_{-T}^{T} \lambda(t)\, dt \right] = e^{-Q}.
\end{aligned}$$

Regrouping the terms, we can show that

$$p(\mathbf{t}_M, M) = p(\mathbf{t}_M) = e^{-Q} \prod_{j=1}^{M} \lambda(t_j).$$

6.3. Proof of Theorem 2

✠ *Remark: This proof is adopted from Bar-David [1].*

Let's assume that $F(\tau)$ is a twice differentiable function. Then, let $F_1 = \dot{F}(\tau_0)$, and decompose F_1 as $F_1 = \mu_1 + F_1'$, where F_1' is a zero-mean random variable. Similarly, decompose $F_2 = \ddot{F}(\tau_0)$ as $F_2 = \mu_2 + F_2'$. Then,

$$\epsilon = -\frac{\dot{F}(\tau_0)}{\ddot{F}(\tau_0)} = -\frac{F_1}{F_2} = -\frac{\mu_1 + F_1'}{\mu_2 + F_2'}.$$

Using Lemma 7, we know that $\mu_1 = 0$. Moreover, let's assume that $F_2' \ll \mu_2$. We can, hence, simplify the above as

$$\begin{aligned}
\epsilon^2 = \left(\frac{F_1'}{\mu_2 + F_2'} \right)^2 &= \left(\frac{F_1'}{\mu_2} \left[\frac{1}{1 + \frac{F_2'}{\mu_2}} \right] \right)^2 \\
&\approx \left(\frac{F_1'}{\mu_2} \left[1 - \frac{F_2'}{\mu_2} \right] \right)^2.
\end{aligned}$$

Taking the expectation will give

$$\mathbb{E}[\epsilon^2] \approx \frac{\mathbb{E}[F_1'^2]}{\mu_2^2} - 2\underbrace{\frac{\mathbb{E}[(F_1'^2 F_2']}{\mu_2^3}}_{\to 0} + \underbrace{\frac{\mathbb{E}[F_1'^2 F_2'^2]}{\mu_2^4}}_{\to 0}.$$

Using Lemma 7 and Lemma 8, and assuming that $\tau_0 = 0$ without loss of generality, it follows that

$$\begin{aligned}
\mathbb{E}[\epsilon^2] = \frac{\mathbb{E}[F_1'^2]}{\mu_2^2} &= \frac{\int_{-T}^{T} \frac{(\alpha \dot{s}(t))^2}{\alpha s(t) + \lambda_b}\, dt}{\left(\int_{-T}^{T} \frac{(\alpha \dot{s}(t))^2}{\alpha s(t) + \lambda_b}\, dt \right)^2} \\
&= \left[\int_{-T}^{T} \frac{(\alpha \dot{s}(t))^2}{\alpha s(t) + \lambda_b}\, dt \right]^{-1}.
\end{aligned}$$

6.4. Proof of Theorem 3

✠ *Remark: This is a new proof we make for this paper.*

We consider $s(x, t) = s(t - \tau(x))$. Using Approximation 1, we write

$$\begin{aligned}
s(x, t) = s(t - \tau(x)) &= \mathcal{N}(t \,|\, \tau(x), \sigma_t^2) \\
&= \mathcal{N}(t \,|\, \tau_n + c_n(x - x_n), \sigma_t^2).
\end{aligned}$$

By Approximation 3, we can show that

$$\begin{aligned}
&\widetilde{s}(x, t) \overset{\text{def}}{=} \phi(x) \circledast s(x, t) \\
&\quad = \mathcal{N}(x \,|\, 0, \sigma_x^2) \circledast \mathcal{N}(t \,|\, \tau_n + c_n(x - x_n), \sigma_t^2) \\
&\quad \overset{(a)}{=} \mathcal{N}(x \,|\, 0, \sigma_x^2) \circledast \frac{1}{c_n} \mathcal{N}\left(x \,\middle|\, \frac{t}{c_n} - \frac{\tau_n}{c_n} + x_n, \frac{\sigma_t^2}{c_n^2} \right) \\
&\quad \overset{(b)}{=} \frac{1}{c_n} \mathcal{N}\left(x \,\middle|\, \frac{t}{c_n} - \frac{\tau_n}{c_n} + x_n, \sigma_n^2 \right) \\
&\quad = \mathcal{N}(t \,|\, \tau_n + c_n(x - x_n), \sigma_n^2),
\end{aligned}$$

where (a) is based on a simple switch between the roles of x and t in a Gaussian distribution, and (b) is based on the fact that the convolution of two Gaussian probability density functions is equivalent to adding two Gaussian random variables.

If we restrict ourselves to $x = x_n$, then we will obtain $s_n(t) \overset{\text{def}}{=} \widetilde{s}(x_n, t) = \mathcal{N}(t \,|\, \tau_n, \sigma_n^2)$. Therefore, using the fact that $\phi(x) \circledast \lambda_b = \int_{-\infty}^{\infty} \phi(x) \lambda_b dx = \lambda_b$ since $\phi(x)$ integrates to 1, we can show that

$$\begin{aligned}
\lambda_n(t) &= \alpha \left\{ \phi(x) \circledast s(x, t) \right\} |_{x = x_n} + \lambda_b \\
&= \alpha \mathcal{N}(t \,|\, \tau_n, \sigma_n^2) + \lambda_b.
\end{aligned}$$

6.5. Proof of Theorem 4

In proving Theorem 4, we need a few lemmas. The first lemma is the classical bias-variance trade-off.

Lemma 2 (MSE decomposition)**.** *The MSE can be decomposed as*

$$\textit{MSE}(\widehat{\tau},\tau) \overset{def}{=} \textit{bias} + \textit{var}, \tag{33}$$

where

$$\textit{bias} = \int_0^1 \left(\overline{\tau}(x) - \tau(x)\right)^2 \, dx$$
$$\textit{var} = \mathbb{E}\left[\int_0^1 \left(\widehat{\tau}(x) - \overline{\tau}(x)\right)^2 \, dx\right].$$

✠ *Remark: The decomposition of the MSE into bias and variance can be found in most of the machine learning textbooks. For completeness and in the context of our paper, we provide the derivations here.*

Proof of Lemma 2. We start with the definition of the MSE:

$$\begin{aligned}\text{MSE}(\widehat{\tau},\tau) &\overset{\text{def}}{=} \mathbb{E}\left[\int_0^1 (\widehat{\tau}(x) - \tau(x))^2 \, dx\right] \\ &= \mathbb{E}\left[\int_0^1 \left(\widehat{\tau}(x) - \overline{\tau}(x) + \overline{\tau}(x) - \tau(x)\right)^2 \, dx\right]\end{aligned}$$

Expanding the terms, we can show that

$$\begin{aligned}\text{MSE}(\widehat{\tau},\tau) = &\underbrace{\mathbb{E}\left[\int_0^1 \left(\widehat{\tau}(x) - \overline{\tau}(x)\right)^2 \, dx\right]}_{\text{variance}} \\ &+ \underbrace{\mathbb{E}\left[\int_0^1 \left(\widehat{\tau}(x) - \overline{\tau}(x)\right)\left(\overline{\tau}(x) - \tau(x)\right) \, dx\right]}_{=0} \\ &+ \underbrace{\int_0^1 \left(\overline{\tau}(x) - \tau(x)\right)^2 \, dx.}_{\text{bias}}\end{aligned}$$

The second expectation in the above is zero because $\mathbb{E}[\widehat{\tau}(x)] = \overline{\tau}(x)$ for every x according to Lemma 7.

Lemma 3 (Bias Term)**.** *Let* $\overline{\tau}(x) = \sum_{n=0}^{N-1} \overline{\tau}_n \varphi(Nx - n)$*, and let* N *be the number of pixels in a unit space* $0 \le x \le 1$*. The bias is approximately*

$$\textit{bias} = \frac{c^2}{12N^2}, \tag{34}$$

where $c_n = \tau'(x_n)$ *is the slope of* τ *at* x_n*, and* $c^2 = \frac{1}{N}\sum_{n=0}^{N-1} c_n^2$*.*

✠ *Remark: This is a new proof we make for this paper.*

Proof of Lemma 3

Integrating over the unit space $[0,1]$, we can show that

$$\begin{aligned}&\int_0^1 \left(\overline{\tau}(x) - \tau(x)\right)^2 \, dx \\ &= \int_0^1 \left(\sum_{n=0}^{N-1} \tau_n \varphi(Nx - n) - \tau(x)\right)^2 \, dx \\ &= \sum_{n=0}^{N-1} \underbrace{\int_{n/N}^{(n+1)/N} \left(\tau_n - \tau(x)\right)^2 \, dx.}_{=e_n^2}\end{aligned}$$

Let's calculate the individual error e_n^2. Let x_n be the mid-point of n/N and $(n+1)/N$. That is,

$$x_n = \frac{1}{2}\left(\frac{n}{N} + \frac{n+1}{N}\right) = \frac{2n+1}{2N}.$$

Then, we can take the first order approximation of $\tau(x)$ around x_n:

$$\tau(x) = \tau(x_n) + \tau'(x_n)(x - x_n).$$

Since $\tau(x_n) = \tau_n$ by construction, it follows that

$$\begin{aligned}e_n^2 &= \int_{n/N}^{(n+1)/N} \left(\tau_n - \tau(x)\right)^2 \, dx \\ &= \int_{n/N}^{(n+1)/N} \Big(\tau_n - (\tau(x_n) + \tau'(x_n)(x - x_n))\Big)^2 \, dx \\ &= \int_{n/N}^{(n+1)/N} \Big(\tau_n - (\tau_n + \tau'(x_n)(x - x_n))\Big)^2 \, dx \\ &= \int_{n/N}^{(n+1)/N} \Big(\tau'(x_n)(x - x_n)\Big)^2 \, dx \\ &= [\tau'(x_n)]^2 \frac{(\frac{n+1}{N} - x_n)^3 - (\frac{n}{N} - x_n)^3}{3}, \qquad x_n = \frac{2n+1}{2N} \\ &= \frac{[\tau'(x_n)]^2}{12N^3}.\end{aligned}$$

Summing over all n's will give us

$$\int_0^1 \left(\overline{\tau}(x) - \tau(x)\right)^2 \, dx = \sum_{n=0}^{N-1} e_n^2 = \frac{c^2}{12N^2},$$

where $c^2 = \frac{1}{N}\sum_{n=0}^{N-1} c_n^2$.

Lemma 4 (Variance Term)**.** *Assume* $s(t)$ *is Gaussian and* $\lambda_b = 0$*. The variance is approximately*

$$\textit{var} = \frac{N}{\alpha_0}\left(c^2\sigma_x^2 + \sigma_t^2\right), \tag{35}$$

where $c_n = \tau'(x_n)$ *is the slope of* τ *at* x_n*, and* $c^2 = \frac{1}{N}\sum_{n=0}^{N-1} c_n^2$*.*

✠ *Remark: This is a new proof we make for this paper.*

Proof of Lemma 4.

Firstly, we use Theorem 3 and apply it to the single-pixel case in Theorem 2. Then, under the assumption that $\lambda_b = 0$, we can follow Example 3 to show that

$$\begin{aligned}\mathbb{E}[(\widehat{\tau}_n - \tau_n)^2] &= \frac{\sigma_n^2}{\alpha} = \frac{c_n^2\sigma_x^2 + \sigma_t^2}{\alpha} \\ &= \frac{N}{\alpha_0}\left(c_n^2\sigma_x^2 + \sigma_t^2\right).\end{aligned}$$

Therefore, for a unit length, it follows that

$$\begin{aligned}\text{var} &= \mathbb{E}\left[\int_0^1 \left(\widehat{\tau}(x) - \overline{\tau}(x)\right)^2 \, dx\right] \\ &= \mathbb{E}\left[\sum_{n=0}^{N-1}\int_{n/N}^{(n+1)/N} \left(\widehat{\tau}(x) - \overline{\tau}(x)\right)^2 \, dx\right] \\ &= \sum_{n=0}^{N-1}\int_{n/N}^{(n+1)/N} \mathbb{E}\left[\left(\widehat{\tau}(x) - \overline{\tau}(x)\right)^2\right] \, dx \\ &= \sum_{n=0}^{N-1}\int_{n/N}^{(n+1)/N} \mathbb{E}\left[\left(\widehat{\tau}_n - \overline{\tau}_n\right)^2\right] \, dx \\ &= \sum_{n=0}^{N-1}\int_{n/N}^{(n+1)/N} \frac{N}{\alpha_0}\left(c_n^2\sigma_x^2 + \sigma_t^2\right) \, dx \\ &= \frac{1}{N}\sum_{n=0}^{N-1}\frac{N}{\alpha_0}\left(c_n^2\sigma_x^2 + \sigma_t^2\right) \\ &= \frac{N}{\alpha_0}\left(c^2\sigma_x^2 + \sigma_t^2\right).\end{aligned}$$

7. Proof of Additional Results

7.1. Proof of Corollary 1

✠ *Remark: This proof is adopted from [1].*

The probability of observing M occurrence is

$$p(M) = \int_{-T}^{T} dt_1 \int_{t_1}^{T} dt_2 \ldots \int_{t_{M-1}}^{T} dt_M p(\mathbf{t}_M, M).$$

The integration limit comes from the fact that the time stamps follow the order $-T \leq t_1 < t_2 < \ldots < t_M \leq T$. The integral is equivalent to $1/M!$ of the hypercube $(-T, T)$. Thus,

$$\begin{aligned}p(M) &= \frac{1}{M!}\int_{-T}^{T} dt_1 \int_{-T}^{T} dt_2 \ldots \int_{-T}^{T} dt_M \quad e^{-Q}\prod_{j=1}^{M}\lambda(t_j) \\ &= \frac{e^{-Q}}{M!}\prod_{j=1}^{M}\int_{-T}^{T}\lambda(t_j)dt_j = \frac{e^{-Q}Q^M}{M!}.\end{aligned}$$

7.2. Proof of Corollary 2

✠ *Remark: Bar-David hinted the result in [1], but the proof was missing. We provide the proof here.*

The sum of the integrals can be written as follows.

$$\begin{aligned}&\sum_{m=0}^{M}\int_{\Omega_M} p(\mathbf{t}_M, M)\, d\mathbf{t}_M \\ &= \sum_{m=0}^{M}\int_{-T}^{T} dt_1 \int_{t_1}^{T} dt_2 \ldots \int_{t_{M-1}}^{T} dt_M \; p(\mathbf{t}_M, M).\end{aligned}$$

Since $\mathbf{t}_M$ is an ordered sequence of time stamps, we can rewrite the integration limits by taking into consideration the $M!$ combinations of the orders. This will give us

$$\begin{aligned}&\sum_{m=0}^{M}\int_{\Omega_M} p(\mathbf{t}_M, M)\, d\mathbf{t}_M \\ &= \sum_{m=0}^{M}\int_{-T}^{T} dt_1 \int_{-T}^{T} dt_2 \ldots \int_{-T}^{T} dt_M \, \frac{1}{M!}p(\mathbf{t}_M, M) \\ &= \sum_{m=0}^{M}\frac{e^{-Q}}{M!}\prod_{j=1}^{M}\left[\int_{-T}^{T}\lambda(t_j)d_{t_j}\right] = \sum_{m=0}^{M}\frac{e^{Q}}{M!}Q^M = 1.\end{aligned}$$

7.3. Proof of Lemma 1

✠ *Remark: This is a new proof we make for this paper*

The KL-divergence of the two distributions is

$$\begin{aligned}\text{KL}(\varphi\|\phi) &= \int_{-\infty}^{\infty}\varphi(x)\log\frac{\varphi(x)}{\phi(x)}dx \\ &= \int_{-\frac{W}{2}}^{\frac{W}{2}} -\frac{1}{W}\left[\log W + \log\left\{\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2}{2\sigma^2}}\right\}\right]dx \\ &= \int_{-\frac{W}{2}}^{\frac{W}{2}} -\frac{\log W}{W} + \frac{1}{W}\log(\sqrt{2\pi\sigma^2}) + \frac{x^2}{2\sigma^2 W}dx \\ &= -\log W + \log(\sqrt{2\pi\sigma^2}) + \frac{W^2}{24\sigma^2}.\end{aligned}$$

Taking the derivative with respect to σ^2 will yield

$$\frac{d}{d\sigma^2}\text{KL}(\varphi\|\phi) = \frac{1}{2\sigma^2} - \frac{W^2}{24\sigma^4}.$$

Equating this to zero and rearranging terms will give us

$$\sigma^2 = \frac{W^2}{12}. \tag{36}$$

7.4. Cramer Rao Lower Bound

Corollary 3 (Cramer Rao Lower Bound). *The Cramer-Rao lower bound for the ML estimate $\widehat{\tau}$ given $\mathcal{L}(\tau)$ is*

$$\begin{aligned}\mathbb{E}[(\widehat{\tau}-\tau_0)^2] &\geq \left(-\mathbb{E}\left[\frac{\partial^2\mathcal{L}}{\partial\tau^2}\right]\right)^{-1} \\ &= \left[\int_{-T}^{T}\frac{(\alpha\dot{s}(t))^2}{\alpha s(t)+\lambda_b}\,dt\right]^{-1}.\end{aligned}$$

✠ *Remark: The Cramer-Rao lower bound for single-photon LiDAR has been previously mentioned in [4]. We provide the proof here for completion.*

Proof of Corollary 3.

Let $\mathcal{L}(\tau)=\log[\alpha s(t-\tau)+\lambda_b]$. Then

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\tau} &= -\frac{\alpha\dot{s}(t-\tau)}{\alpha s(t-\tau)+\lambda_b} \\ \frac{\partial^2\mathcal{L}}{\partial\tau^2} &= -\frac{-(\alpha s(t-\tau)+\lambda_b)\ddot{s}(t-\tau)+(\alpha\dot{s}(t-\tau))^2}{(\alpha s(t-\tau)+\lambda_b)^2} \\ &= \frac{\ddot{s}(t-\tau)}{\alpha s(t-\tau)+\lambda_b}-\frac{(\alpha\dot{s}(t-\tau))^2}{(\alpha s(t-\tau)+\lambda_b)^2}\end{aligned}$$

Taking the expectation over t, we have

$$\begin{aligned}\mathbb{E}\left[-\frac{\partial^2\mathcal{L}}{\partial\tau^2}\right] &= -\int_{-T}^{T}\frac{\partial^2\mathcal{L}}{\partial\tau^2}\cdot[\alpha s(t-\tau)+\lambda_b]dt \\ &= \underbrace{-\int_{-T}^{T}\ddot{s}(t-\tau)\,dt}_{=0}+\int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau))^2}{\alpha s(t-\tau)+\lambda_b}\,dt \\ &= \int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau))^2}{\alpha s(t-\tau)+\lambda_b}\,dt.\end{aligned}$$

Without loss of generality, we set $\tau=0$. Taking the reciprocal completes the proof.

8. Auxiliary Results and Proofs

In this section, we present additional results and proofs that are essential to the development of our theories.

8.1. Lemma about Product of Functions

Lemma 5. *Consider a function $f(t)$ and define its product:*

$$f_\pi(\mathbf{t}_M)=\prod_{j=1}^{M}f(t_j). \tag{37}$$

Assume $\mathbf{t}_M\sim p(\mathbf{t}_M)$. The expectation of f_π is

$$\mathbb{E}[f_\pi(\mathbf{t}_M)]=e^{-Q+G(T)}, \tag{38}$$

where $G(T)=\int_{-T}^{T}f(t)\lambda(t)dt$.

✠ *Remark: This proof is adopted from Bar-David.*

Proof of Lemma 5.

The expectation can be shown as follows.

$$\begin{aligned}\mathbb{E}[f_\pi(\mathbf{t}_M)] &= \int_\Omega f_\pi(\mathbf{t}_M)p(\mathbf{t}_M)d\mathbf{t}_M \\ &= \sum_{M=0}^{\infty}\underbrace{\int_{\Omega_M}f_\pi(\mathbf{t}_M)p(\mathbf{t}_M)d\mathbf{t}_M}_{\overset{\text{def}}{=}\mathbb{E}_M(f_\pi)}.\end{aligned}$$

Substituting the definition of $f_\pi(\mathbf{t}_M)$ into the expression above, we can show that

$$\begin{aligned}\mathbb{E}_M(f_\pi) &= \int_{\Omega_M}f_\pi(\mathbf{t}_M)p(\mathbf{t}_M,M)\,d\mathbf{t}_M \\ &= e^{-Q}\int_{-T}^{T}dt_1\int_{t_1}^{T}dt_2\ldots\int_{t_{M-1}}^{T}dt_M\prod_{j=1}^{M}g(t_j),\end{aligned}$$

where $g(t)$ is defined as $g(t)=f(t)\lambda(t)$.

Now, consider a slightly different integration

$$\begin{aligned}&e^{-Q}\int_{-T}^{T}dt_1\int_{-T}^{T}dt_2\ldots\int_{-T}^{T}dt_M\prod_{j=1}^{M}g(t_j) \\ &= e^{-Q}\left[\int_{-T}^{T}g(t)dt\right]^M \\ &= e^{-Q}G^M(T),\end{aligned}$$

where $G(T)=\int_{-T}^{T}g(t)dt$. The difference between this integral and the previous integral is that the previous integral is based $-T\leq t_1\leq\ldots\leq t_M\leq T$. To account for the shape from this triangle to the full hypercube, we add the $M!$ factors. This will give us

$$\mathbb{E}_M(f_\pi)=\frac{e^QG^M(T)}{M!}.$$

Therefore,

$$\mathbb{E}[f_\pi(\mathbf{t}_M)]=e^{-Q}\sum_{j=1}^{M}\frac{G^M(T)}{M!}=e^{-Q+G(T)}.$$

8.2. Lemma about Characteristic Function

Lemma 6. *For the function $f(t;iu)$ defined as, where $i=\sqrt{-1}$,*

$$f(t;iu)=\exp\left\{iu\cdot\frac{\alpha\dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right\}. \qquad (39)$$

The function $G(T)=\int_{-T}^{T}f(t;iu)\lambda(t)dt$ has derivatives

$$G(T)\Big|_{iu=0}=\int_{-T}^{T}\lambda(t)\,dt=Q$$
$$\frac{d}{d(iu)}G(T)\Big|_{iu=0}=\int_{-T}^{T}\alpha\dot{s}(t-\tau_n)\,dt$$
$$\frac{d^2}{d(iu)^2}G(T)\Big|_{iu=0}=\int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b}\,dt$$

✠ *Remark: Bar-David mentioned the idea but we could not find the proof. Therefore, the proof here is new.*

Proof of Lemma 6.

For the $f(t;iu)$ defined, we can show that

$$\frac{d}{d(iu)}f(t;iu)$$
$$=\frac{\alpha\dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\cdot\exp\left\{iu\cdot\frac{\alpha\dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right\}$$
$$=\frac{\alpha\dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\cdot f(t;iu)$$

If we restrict $iu=0$, then $f(t;iu)|_{iu=0}=1$.Therefore,

$$\frac{dG(T)}{d(iu)}\Big|_{iu=0}=\int_{-T}^{T}\frac{d}{d(iu)}f(t;iu)\cdot\lambda(t)\,dt\Big|_{iu=0}$$
$$=\int_{-T}^{T}f(t;iu)\cdot\frac{\alpha\dot{s}(t-\tau_n)}{\lambda(t)}\cdot\lambda(t)\,dt\Big|_{iu=0}$$
$$=\int_{-T}^{T}\alpha\dot{s}(t-\tau_n)dt=0,$$

where the last equality holds whenever $s(t)$ is a symmetric function.

$$\frac{d^2G(T)}{d(iu)^2}\Big|_{iu=0}$$
$$=\int_{-T}^{T}\frac{d}{d(iu)}f(t;iu)\cdot\frac{\alpha\dot{s}(t-\tau_n)}{\lambda(t)}\cdot\lambda(t)\,dt\Big|_{iu=0}$$
$$=\int_{-T}^{T}f(t;iu)\cdot\left(\frac{\alpha\dot{s}(t-\tau_n)}{\lambda(t)}\right)^2\cdot\lambda(t)\,dt\Big|_{iu=0}$$
$$=\int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b}\,dt.$$

8.3. Lemma for μ_1 and F_1

Lemma 7. *Let $F_1=\dot{F}(\tau_n)$, where the derivative is taken with respect to τ. Write $F_1=\mu_1+F_1'$. It holds that*

$$\mu_1=0,\qquad \mathbb{E}[F_1'^2]=\int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b}dt.$$

✠ *Remark: Similar to the previous Lemma, Bar-David mentioned the idea but we could not find the proof. Therefore, the proof here is new.*

Proof of Lemma 7.

Let the characteristic function of the random variable F_1 be

$$\Phi(iu)=\mathbb{E}\left[\exp\left\{ix\cdot\sum_{j=1}^{M}\frac{\alpha\dot{s}(t_j-\tau_n)}{\alpha s(t_j-\tau_n)+\lambda_b}\right\}\right]. \qquad (40)$$

Thus, by Lemma 5, we have $\Phi(iu)=\mathbb{E}[f_\pi(\mathbf{t}_M)]=e^{-Q+G(T)}$ with f_π defined by Lemma 6.

The first moment of F_1 is therefore

$$\mu_1=\frac{d}{d(iu)}\Phi(iu)\Big|_{iu=0}=\frac{d}{d(iu)}e^{-Q+G(T)}\Big|_{iu=0}$$
$$=e^{-Q+G(T)}\frac{d}{d(iu)}G(T)\Big|_{iu=0}=0,$$

where the last equality is due to Lemma 6.

For $\mathbb{E}[F_1'^2]$, we can show that

$$\mathbb{E}[F_1'^2]=\frac{d^2}{d(iu)^2}\Phi(iu)\Big|_{iu=0}$$
$$=e^{-Q}\frac{d}{d(iu)}\left[e^{G(T)}\frac{d}{d(iu)}G(T)\right]\Big|_{iu=0}$$
$$=e^{-Q}\left[e^{G(T)}\left(\frac{d}{d(iu)}G(T)\right)^2+e^{G(T)}\frac{d^2}{d(iu)^2}G(T)\right]\Big|_{iu=0}$$
$$=e^{-Q}\left[e^{Q}(0)^2+e^{Q}\int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau_n))^2}{\lambda(t)}dt\right]$$
$$=\int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau_n))^2}{\lambda(t)}dt=\int_{-T}^{T}\frac{(\alpha\dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b}dt.$$

Since the width of the pulse is much smaller than the interval $(-T, T)$, we can make $\tau_n = 0$ without loss of generality.

8.4. Lemma for μ_2 and F_2

Lemma 8. *Let $F_2 = \ddot{F}(\tau_n)$. Write $F_2 = \mu_2 + F_2'$. We can show that*

$$\mu_2 = \int_{-T}^{T} \frac{(\alpha \dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b} dt,$$

$$\mathbb{E}[F_2'^2] = \int_{-T}^{T} \left(\frac{d}{d\tau_n}\left(\frac{\alpha \dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right)\right)^2 \lambda(t)\, dt.$$

✠ *Remark: Similar to the previous Lemma, Bar-David mentioned the idea but we could not find the proof. Therefore, the proof here is new.*

Proof of Lemma 8.

The characteristic function of F_2 is

$$\Phi(iu) = \mathbb{E}\left[\exp\left\{iu \cdot \sum_{j=1}^{M} \frac{d}{d\tau_n}\frac{\alpha \dot{s}(t_j-\tau_n)}{\alpha s(t_j-\tau_n)+\lambda_b}\right\}\right]. \tag{41}$$

Similar to Lemma 7, we define

$$h(t) = \exp\left\{iu \cdot \frac{d}{d\tau_n}\frac{\alpha \dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right\}. \tag{42}$$

Then, the function $H(T) = \int_{-T}^{T} h(t)\lambda(t)dt$, where $\lambda(t) = \alpha s(t-\tau_n) + \lambda_b$, and its derivatives are

$$H(T)\Big|_{iu=0} = \int_{-T}^{T} [\alpha s(t-\tau_n)+\lambda_b] dt$$

$$\frac{d}{d(iu)}H(T)\Big|_{iu=0} = \int_{-T}^{T} \frac{d}{d\tau_n}\left(\frac{\alpha \dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right)\lambda(t)\, dt$$

$$\frac{d^2}{d(iu)^2}H(T)\Big|_{iu=0} = \int_{-T}^{T} \left(\frac{d}{d\tau_n}\left(\frac{\alpha \dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right)\right)^2\lambda(t)\, dt$$

With some simplifications, we can show that

$$\begin{aligned}
\mu_2 &= \frac{d}{d(iu)}H(T)\Big|_{iu=0} \\
&= \int_{-T}^{T} \frac{d}{d\tau_n}\left(\frac{\alpha \dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right)\lambda(t)\, dt \\
&= \int_{-T}^{T} \left(\frac{-\lambda(t)\alpha\ddot{s}(t-\tau_n) - \alpha\dot{s}(t-\tau_n)\dot{\lambda}(t)}{\lambda(t)^2}\right)\lambda(t)\, dt \\
&= \underbrace{-\int_{-T}^{T} \alpha\ddot{s}(t)\, dt}_{=0} + \int_{-T}^{T}\frac{(\alpha \dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b} dt.
\end{aligned}$$

Therefore,

$$\mu_2 = \int_{-T}^{T}\frac{(\alpha \dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b} dt.$$

Similarly, we can show that

$$\begin{aligned}
&\mathbb{E}[F_2'^2] = \frac{d^2}{d(iu)^2}\Phi(iu)\Big|_{iu=0} \\
&= e^{-Q}\left[e^{H(T)}\left(\frac{d}{d(iu)}H(T)\right)^2 + e^{H(T)}\frac{d^2}{d(iu)^2}H(T)\right]\Bigg|_{iu=0} \\
&= \left(\int_{-T}^{T}\frac{(\alpha \dot{s}(t-\tau_n))^2}{\alpha s(t-\tau_n)+\lambda_b} dt\right)^2 \\
&\quad + \int_{-T}^{T}\left(\frac{d}{d\tau_n}\left(\frac{\alpha \dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right)\right)^2\Big(\alpha s(t-\tau_n)+\lambda_b\Big)\, dt.
\end{aligned}$$

Subtracting the mean square will give us

$$\begin{aligned}
&\mathbb{E}[F_2'^2] = \mathbb{E}[F_2^2] - \mu_2^2 \\
&= \int_{-T}^{T}\left(\frac{d}{d\tau_n}\left(\frac{\alpha \dot{s}(t-\tau_n)}{\alpha s(t-\tau_n)+\lambda_b}\right)\right)^2\Big(\alpha s(t-\tau_n)+\lambda_b\Big) dt.
\end{aligned}$$

9. Detailed Setups of Experiments

For the experiments we presented in the main text, the parameters are configured as Table 1. The parameters here are unit-free in the sense they are picked to elaborate the theory.

Table 1. Parameters used in 1D experiments

N	number of pixels	
x	spatial coordinate	$0 \le x \le 1$
t	temporal duration	$0 \le t \le 10$
ΔN	width of each pixel	$1/N$
Δx	spatial grid spacing	1/2048
Δt	temporal grid spacing	1/256
σ_t	pulse width	0.5
α_0	overall scene flux	10000
λ_b	noise floor	0
σ_x	spatial Gaussian radius	$1/(\sqrt{12}N)$
c_n	nth pixel of $\tau'(x)$	use `gradient`

9.1. Ground Truth Time-of-Arrival Function

The true time of arrival function $\tau(x)$ can be any function defined on $0 \le x \le 1$. For example, in the main text, we consider a two-step scene with a smooth transition. Therefore, we can pick a sigmoid function of the following form.

$$\tau(x) = \frac{4}{1+e^{-20(x-0.5)}} + 4, \tag{43}$$

The shape for this particular model is shown in Fig. 11 below. We stress that this is just one of the many models we can pick.

To compute the gradient $\tau'(x)$, the computationally efficient way is to perform finite difference instead of the analytic form. In MATLAB, this can be done using the command `gradient`.

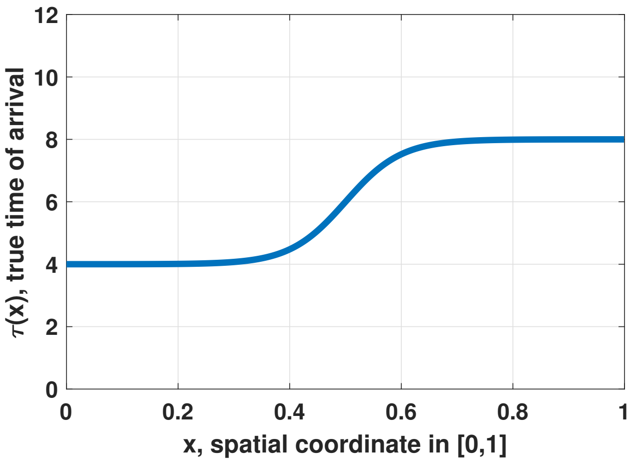

Figure 11. The ground truth time of arrival function $\tau(x)$ as a function of the coordinate $0 \leq x \leq 1$.

```
tau = 4./(1+exp(-20*(x-0.5)))+4;
tau_slope = gradient(tau)/dx;
```

The division by `dx` at the end is a reflection of the finite difference operation. For example, at x_k, the true gradient is

$$\begin{aligned}\tau'(x_k) &= \lim_{\Delta x \to 0} \frac{\tau(x_k + \Delta x) - \tau(x_k)}{\Delta x} \\ &\approx \frac{\tau(x_{k+1}) - \tau(x_k)}{\Delta x}.\end{aligned}$$

Since finite difference `gradient` only computes $\tau(x_{k+1}) - \tau(x_k)$, we need to divide the result by Δx to compensate for the spatial interval.

9.2. What if we use a different $\tau(x)$?

A critical observation of our theory is that the MSE can be decomposed into bias and variance. The variance has some subtle but important dependency on the shape of τ, but the bias term will experience more impact.

According to Lemma 3, the bias can be approximated by

$$\text{bias} = \int_0^1 [\tau(x) - \overline{\tau}(x)]^2 dx \approx \frac{c^2}{12N^2}, \qquad (44)$$

where $c^2 = \frac{1}{N}\sum_{n=0}^{N-1} c_n^2$, with $c_n = \tau'(x_n)$ being the gradient of τ. Therefore, as τ changes, c^2 will change accordingly. Figure 12 illustrates two examples. In the same figure, we also plot the theoretically predicted bias and the simulated result. When $\tau(x)$ is relatively smooth, the theory has an excellent match with the simulation.

Limitations. With no surprise, the piecewise constant approximation has limitations. (1) When the function $\tau(x)$ is intrinsically discontinuous, the theory will not be able to

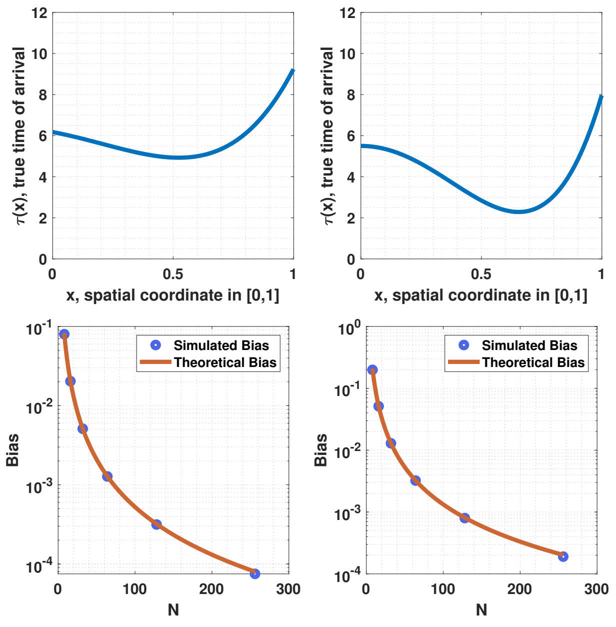

Figure 12. $\tau(x)$ and the bias.

keep track of the trend. (2) When the function $\tau(x)$ is noisy, then the gradient will be overestimated. Both will lead to degradation of the bias estimation, as illustrated in Fig. 13.

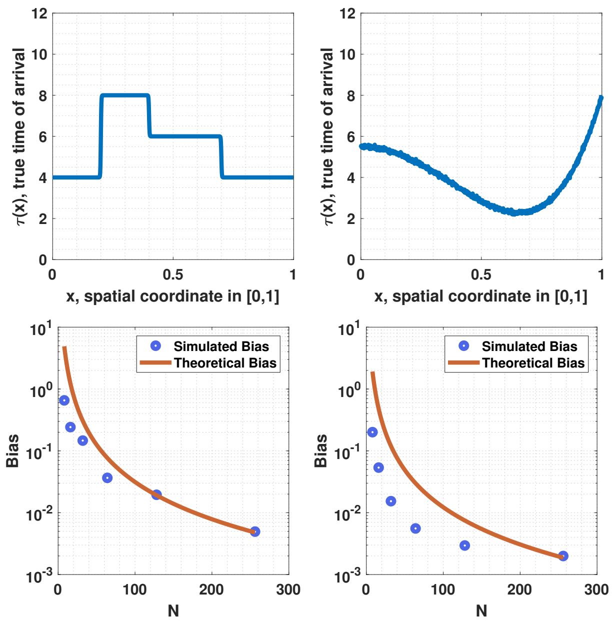

Figure 13. Limitations of the approximations we presented in this paper. In the presence of discontinuous edges, and/or in the presence of noise, the performance of the theoretical prediction degrades.

Not all limitations can be analytically mitigated. For example, mitigating the discontinuity requires us to know the location of the discontinuous points. In practice, the likely solution is to numerically integrate the error. However, we will not be able to retain a clean, interpretable, and elegant formula.

For the noise problem, we can model the true time of arrival function as

$$\tau(x) = \tau_0(x) + \eta(x),$$

where $\eta(x)$ is a random process denoting the noise, e.g., $\eta(x) \sim \mathcal{N}(0, \sigma_e^2)$. In this case, the bias will become

$$\text{bias} = \sum_{n=0}^{N-1} \int_{\frac{n}{N}}^{\frac{n+1}{N}} [\tau'(x_n)(x - x_n) + e(x)]^2 dx.$$

With some calculations, we can show that

$$\text{bias} = \frac{c^2}{12N^2} + \sigma_e^2,$$

where $c^2 = \frac{1}{N}\sum_{n=0}^{N-1} c_n^2$, with $c_n = \tau_0'(x_n)$ being the slope of the *clean* signal $\tau_0(x)$. Therefore, if we are given the noisy time of arrival $\tau(x)$, we first need to recover the clean $\tau_0(x)$ so that we can estimate the correct c^2. Then, the noise variance σ_e^2 is added to compensate for the noise term. The correction can be visualized in Fig. 14.

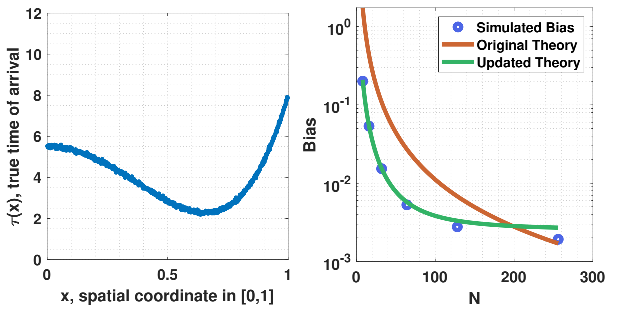

Figure 14. When the true time of arrival is noise, it is possible to obtain a better bias estimate by compensating for the noise. However, this is rather an *oracle* scenario because in practice, we never know the true noise-free $\tau(x)$ and we never know the probability distribution of noise.

10. Unit Conversion

For our theoretical analysis to match a realistic sensor, a conversion between the size of a physical pixel and the size of a point in the numerical grid needs to be established. In what follows, we present a specific example with some specific numbers. These can be easily translated to other configurations.

Suppose that the overall size of the SPAD array is 10mm as shown in Fig. 15 below. This 10mm would correspond to 1 unit space. Suppose that we use 1024 points in the numerical grid to measure the array. Then, the width of each pixel is

$$\Delta x \overset{\text{def}}{=} 1 \text{ pixel} = \frac{10\text{mm}}{1024} = 9.76\text{um} = \frac{1}{1024}\text{unit space}.$$

For this paper's analysis, let's assume that we group 32 points (for example) as one *super-pixel*. The width of each super-pixel is therefore

$$1 \text{ super-pixel} = 32 \text{ pixels} = 312.5\text{um} = \frac{32}{1024}\text{unit space}.$$

Now, if we want to use a Gaussian to approximate the boxcar kernel, then the standard deviation of the Gaussian should be

$$\begin{aligned}\sigma_x &= \frac{1 \text{ super-pixel}}{\sqrt{12}} \\ &= 9.2372 \text{ pixels} = 90.21\text{um} = \frac{32}{1024\sqrt{12}}\text{unit space}.\end{aligned}$$

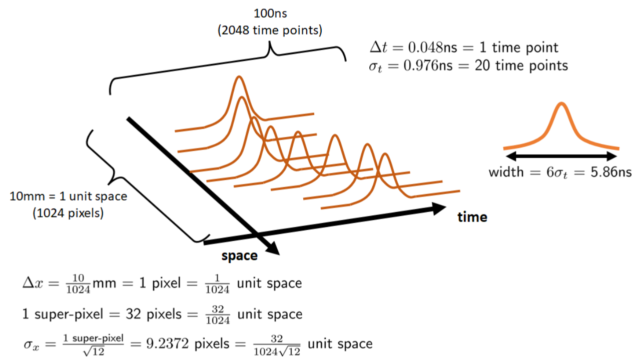
100ns
(2048 time points)
$\Delta t = 0.048$ns = 1 time point
$\sigma_t = 0.976$ns = 20 time points
width $= 6\sigma_t = 5.86$ns
10mm = 1 unit space
(1024 pixels)
time
space
$\Delta x = \frac{10}{1024}$mm = 1 pixel = $\frac{1}{1024}$ unit space
1 super-pixel = 32 pixels = $\frac{32}{1024}$ unit space
$\sigma_x = \frac{1 \text{ super-pixel}}{\sqrt{12}} = 9.2372$ pixels = $\frac{32}{1024\sqrt{12}}$ unit space

Figure 15. Conversion between the physical units to the number of grid points used on a computer.

For the temporal axis, we take a similar approach. Suppose that the total duration of the measurement is 100ns, and we use 2048 time points to measure the total duration. Then, each time point would correspond to

$$\Delta t = 1 \text{ time point} = \frac{100\text{ns}}{2048} = 0.048\text{ns}.$$

A typical pulse has a width of around 2ns-6ns. So, suppose that we define the pulse standard deviation as $\sigma_t = 20$ time points, then

$$\sigma_t = 0.976\text{ns} = 20 \text{ time points}.$$

Since $6\sigma_t$ of a Gaussian can capture 99% of the energy, we can safely say that the width of the pulse is around

$$\text{pulse width} = 6\sigma_t = 120 \text{ time points} = 5.86\text{ns},$$

which is reasonably inside the 2ns-6ns range.

11. Sampling Procedure

In the main text, we outlined a procedure to generate time stamps according to a Gaussian pulse. This section discusses how to extend the idea to arbitrary pulse shapes.

11.1. Inverse CDF Method

When the pulse $\lambda(t)$ has a completely arbitrary shape and when the noise floor is not a constant, we will not be able to draw samples from distributions with known formulae. In this case, the sampling can be done using the inverse cumulative distribution function (inverse CDF) technique [2, Ch.4.9].

The concept of sampling from a known PDF is to leverage the cumulative distribution function (CDF) and a uniform random variable. A classical result shows the following. Suppose that there is a random variable X generated from a distribution with a CDF F_X. If $U \sim \text{Uniform}[0, 1]$, and if we send U to the inverse CDF F_X^{-1}, then the transformed random variable $F_X^{-1}(U)$ will follow the distribution F_X.

The specific steps to perform the inverse CDF are as follows.

- Step 1: Compute the CDF $\Lambda(t) = \int_{-\infty}^{t} \lambda(r)dr$. Assuming that $\Lambda(t)$ is invertible, compute the inverse mapping Λ^{-1}. To obtain $\Lambda^{-1}(p)$ for a given p, we numerically build a lookup table.
- Step 2: Compute the number of samples $M \sim \text{Poisson}(Q)$. Then, generate random samples by sending uniform random variables: $t_j = \Lambda^{-1}(U_j)$, where $U_j \sim \text{Uniform}[0, 1]$, and $j = 1, \ldots, M$. Then $\{t_j\}_{j=1}^{M}$ will follow the probability distribution $\lambda(t)$.

11.2. Implementation and Demonstration

In terms of implementation, we can use the following MATLAB code. Suppose that we are given the received pulse $\lambda(t) = \alpha s(t - \tau) + \lambda_b$. Then, we can integrate $\lambda(t)$ to obtain the CDF $\Lambda(t) = \int_{-\infty}^{t} \lambda(r)dr$. On a computer, the command `cumsum` will serve the purpose of generating the CDF. Once the CDF is numerically determined, we send M uniform random variables to Λ^{-1}. The inversion is performed numerically by finding the closest t_j such that $t_j = \Lambda^{-1}(p)$ can give $\Lambda(t_j) = p$, where p is the uniform random variable.

```
function time_stamps = ...
    generate_time_stamps(lambda,t,M)

    c = cumsum(lambda)/sum(lambda);
    p = rand(1,M);
    c = repmat(c(:),[1,M]);
    [~,pos] = min(abs(c-p));
    time_stamps = t(pos);
end
```

Fig. 16 shows a demonstration where we generate M time stamps. The underlying pulse shape is Gaussian, but we also assume a nonlinear noise floor. Specifically, the noise floor mimics a scattering medium which is modeled by a Gamma distribution.

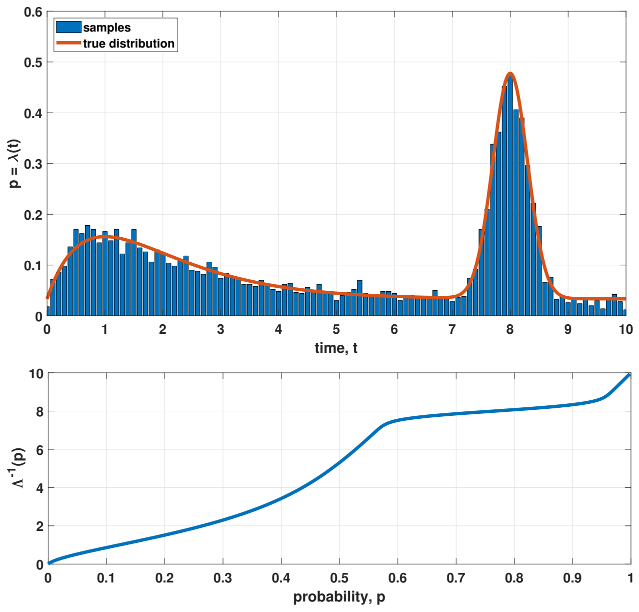

Figure 16. [Top] Random samples $\mathbf{t}_M$ drawn from a pulse $\lambda(t)$ which consists of a non-uniform noise floor and a Gaussian pulse. [Bottom] The inverse CDF $\Lambda^{-1}(p)$ is used to generate the samples from a uniform distribution. $\Lambda^{-1}(p)$ is generated numerically from the known equation $\lambda(t)$.

```
s1 = 0.2*pdf('normal', t, tau, sigma_t);
s2 = 0.2*pdf('gamma', t, 2, 1);
lambda_b = 0.02*ones(1,length(t));
lambda   = s1+s2+lambda_b;

lambda_pdf = lambda/sum(lambda*dt);
lambda_cdf = cumsum(lambda/sum(lambda));
dp = 1/500;
p  = 0:dp:1-dp;
c = repmat(lambda_cdf(:),[1,length(p)]);
[~,pos] = min(abs(c-p));
lambda_icdf = t(pos);
```

12. Non-zero Noise Floor

The goal of the main text is to derive a simple and informative MSE equation. This is achieved by assuming that the pulse is Gaussian and there is no noise. In this section, we discuss how the theoretical results can be derived for arbitrary pulses, by means of a numerical scheme.

12.1. Maximum Likelihood Estimation

Assuming that the transmitted pulse is $s(t)$ and the received pulse is $\lambda(x, t) = \alpha s(t - \tau(x)) + \lambda_b$. Here, we assume that

$s(t)$ can take an arbitrary shape and $\lambda_b > 0$. We also assume that there is a spatial kernel $\phi(x)$ which will be applied to $\lambda(x,t)$ to yield the resulting space-time function $\widetilde{\lambda}(x,t)$:

$$\widetilde{\lambda}(x,t) = \phi(x) \circledast \lambda(x,t).$$

The convolution $\circledast$ can be implemented numerically. In MATLAB, we can use the command `imfilter`:

```
lambda = alpha*s + lambda_b
h = ones(dN,1)/dN;
lambda_tilde = imfilter(lambda, h, ...
            replicate');
```

Following the same derivations as the main text, we let $x_n = (2n+1)/(2N)$ be the mid point of each spatial interval and define the effective pulse $\widetilde{\lambda}_n(t)$ as

$$\widetilde{\lambda}_n(t;\tau_n) = \widetilde{\lambda}(x_n, t; \tau_n),$$

where we emphasize that $\widetilde{\lambda}_n$ has an underlying parameter τ_n which needs to be estimated. Then, the time of arrivals $\mathbf{t}_M = [t_1, \ldots, t_M]$ will follow the distribution specified by the effective return pulse $\widetilde{\lambda}_n(t)$.

The questions we need to ask now are (1). how to estimate the time of arrival $\widehat{\tau}$, (2). what is the variance $\mathbb{E}[(\widehat{\tau}_n - \tau_n)^2]$?

For the purpose of deriving theoretical bounds, the estimator we use is the maximum likelihood estimator. The ML estimator is the one that maximizes the likelihood function. In our case, it is

$$\widehat{\tau}_n = \underset{\tau_n}{\text{argmax}} \; \underbrace{\sum_{j=1}^{M} \log \widetilde{\lambda}_n(t_j;\tau_n)}_{\mathcal{L}(\tau_n)}, \qquad (45)$$

The shape of the likelihood function $\mathcal{L}(\tau_n)$ as a function of the parameter τ_n is shown in Fig. 17, for a typical Gaussian pulse with a non-zero noise floor. The likelihood function has a clear maximum (minimum if we take the negative log likelihood) around the true parameter. For the specific example shown in Fig. 17, we use the following configurations: $\alpha = 100$, $\lambda_b = 30$, $s(t,\tau_0) = \mathcal{N}(t \,|\, \tau_0, \sigma_t^2)$ where $\tau_0 = 5$ and $\sigma_t = 0.5$. We randomly generate $K = 100$ random vectors $\mathbf{t}_M^{(k)}$ for $k = 1, \ldots, K$. We plot the negative log-likelihood $\mathcal{L}(\tau_n \,|\, \mathbf{t}_M^{(k)})$ as a function of the parameter τ_n. The negative log-likelihood functions are random because $\mathbf{t}_M^{(k)}$ are random. We take the expectation of these $K = 100$ random negative log-likelihood functions to visualize the mean function.

An equivalent alternative approach is to perform the estimation by solving a zero-finding problem. Taking the derivative of $\mathcal{L}$ with respect to τ_n, we can show that the solution $\widehat{\tau}_n$ must satisfy the equation

$$\left.\frac{d\mathcal{L}}{d\tau_n}\right|_{\tau_n=\widehat{\tau}_n} = \left.\sum_{j=1}^{M} \frac{\dot{\widetilde{\lambda}}_n(t_j;\tau_n)}{\widetilde{\lambda}_n(t_j;\tau_n)}\right|_{\tau_n=\widehat{\tau}_n} = 0. \qquad (46)$$

Then, the maximum likelihood estimate $\widehat{\tau}_n$ is determined by finding the zero-crossing of the derivative such that $\dot{\mathcal{L}}(\widehat{\tau}_n) = 0$.

Example 4. *Suppose that $\widetilde{\lambda}_n(t;\tau_n) = \alpha s(t-\tau_n) + \lambda_b$, then the likelihood function is*

$$\mathcal{L}(\tau_n) = \sum_{j=1}^{M} \log\left\{\alpha s(t_j - \tau_n) + \lambda_b\right\}.$$

The derivative of the likelihood is

$$\frac{d\mathcal{L}}{d\tau_n} = -\sum_{j=1}^{M} \frac{\alpha s(t_j - \tau_n)}{\alpha s(t_j-\tau_n) + \lambda_b}.$$

The shape of the likelihood derivative can be seen in Fig. 18, for a typical example outlined in the description of Fig. 17. The maximum likelihood estimate is the zero-crossing of the derivative. We emphasize that for arbitrary pulse shape and a non-zero floor noise, neither the zero-finding approach nor the matched filter approach would have an analytic solution. We will discuss the numerical implementation shortly.

12.2. MSE Calculation

Given the effective return pulse $\widetilde{\lambda}_n(t)$, the MSE follows from Theorem 2. To make our notations consistent, we as-

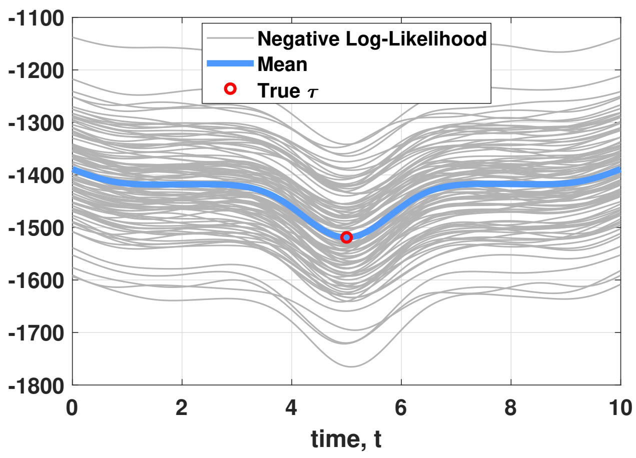

Figure 17. The shape of the negative log-likelihood function $\mathcal{L}(\tau)$ as a function of the unknown parameter τ.

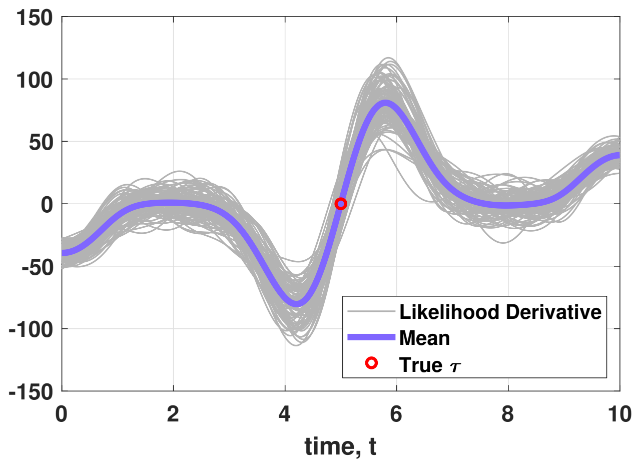

Figure 18. The shape of the negative log-likelihood function's derivative $\frac{d}{d\tau}\mathcal{L}(\tau)$ as a function of the unknown parameter τ.

sume that $\widetilde{\lambda}_n(t)$ takes the form

$$\widetilde{\lambda}_n(t) = \alpha s_n(t - \tau_n) + \lambda_b, \tag{47}$$

for some function $s_n(t)$. This assumption is valid whenever α and λ_b are independent of x so that any spatial convolution applied to construct $\widetilde{\lambda}(t)$ will not affect α. Under this notation, we follow Theorem 2 and write the variance of the estimator

$$\mathbb{E}[(\widehat{\tau}_n - \tau_n)^2] = \left[\int_{-T}^{T} \frac{(\alpha \dot{s}_n(t))^2}{\alpha s_n(t) + \lambda_b}\, dt\right]^{-1}, \tag{48}$$

where $\dot{s}_n(t)$ is the derivative of $s_n(t)$ with respect to t.

For arbitrary pulse shape, $s_n(t)$ does not have any analytic expression, so does $\dot{s}_n(t)$. Nevertheless, the integration in Eq. (49) can still be performed numerically. The MATLAB code below shows how we can do it numerically.

```
tau = 4./(1+exp(-20*(x-0.5)))+4;
tau_slope = gradient(tau)/dx;
for n=1:N
 c(n) = mean( tau_slope(dN*(n-1)+[1:dN]) );
 s_dot(n,:) = gradient(s(n,:))/dt;
 var_n(n) = 1/sum(((alpha*s_dot(n,:)).^2) ...
       ./(alpha*s(n,:)+lambda_b)*dt);
end
var_theory  = mean( var_n );
bias_theory = mean((c.^2)/(12*N^2));
mse_theory  = bias_theory + var_theory;
```

With Eq. (48) defined, the overall MSE as a function of the number of pixels is as follows.

Theorem 5. *For arbitrary pulse shape and noise floor, the MSE takes the form*

$$\mathit{MSE} = \frac{c^2}{12N^2} + \frac{N}{\alpha_0}\mathbb{E}[(\widehat{\tau}_n - \tau_n)^2]. \tag{49}$$

Here, we use the fact that $\mathbb{E}[(\widehat{\tau}_n - \tau_n)^2]$ has the same value for all n so it does not matter which n we use to calculate the variance. Although Eq. (49) is no longer a closed-form expression, it still preserves the shape of the resolution limit where the MSE decays quadratically with respect to $1/N^2$ and increases linearly with respect to N.

12.3. Solving the ML Estimation

In this subsection, we explain how to implement the maximum likelihood estimation for an arbitrary pulse and noise floor. There are three ways to do this.

Approach 1: Gradient-based Likelihood Maximization. For pulses with a known analytic expression, the most straightforward approach is to perform gradient descent with the gradient $\frac{d}{d\tau}\mathcal{L}$:

$$\tau^{(k+1)} = \tau^{(k)} - \gamma^{(k)}\frac{d}{d\tau}\mathcal{L}(\tau^{(k)}),$$

for some step size $\gamma^{(k)}$. Most numerical software such as MATLAB has built-in optimization packages to accomplish this step. For example, using `fminunc`, we will be able to find the maximum of the likelihood function.

The example below assumes a Gaussian pulse so that we have a closed-form expression for the likelihood function.

```
function L = myL(tau, tj, ...
          sigma_t, alpha, lambda_b)
L  = -sum(log(alpha * ...
          1/sqrt(2*pi*sigma_t^2) * ...
     * exp(-(tj-tau).^2/(2*sigma_t^2)) + ...
     + lambda_b));
```

To solve the actual maximization we do

```
fun = @(tau) myL(tau, tj, ...
    sigma_t, alpha, lambda_b);
tau_hat = fminunc(myL, tau0, option);
```

where `tau0` is the initial guess. For the purpose of deriving the "oracle" maximum likelihood estimate (so that we are not penalized by having a bad algorithm), we use the ground truth τ_0 as the initial guess. Readers may worry that this would return us the ground truth τ_0 as the maximum likelihood estimate, i.e., $\widehat{\tau} = \tau_0$. We note that this will never happen because the likelihood $\mathcal{L}$ needs to fit the measured time stamps $\mathbf{t}_M$ and so $\mathcal{L}$ is a random function. Since $\mathcal{L}$ is a random function, the maximum location is a random variable too. Therefore, while the expected value of the estimate $\widehat{\tau}$ is the true τ_0, for a particular realization $\widehat{\tau}$ will never be the same as τ_0.

Approach 2: Search-based Likelihood Maximization. For arbitrary pulse shape, the gradient-based approach is difficult to implement because the pulse $s(t - \tau)$ is numerically a different function when we use a different τ. The viable approach, as we mentioned in the previous subsection, is to run a matched filter. A matched filter requires us to shift the known pulse to the left or to the right until we see the best fit to the data. On computers, we need to perform two steps:

- Given a current estimate τ and a perturbation $\Delta\tau$, translate it to the time index of the numerical array $\widetilde{\lambda}_n(t;\tau)$.
- Shift the index based on the amount corresponding to $\Delta\tau$, and calculate $\widetilde{\lambda}_n(t;\tau+\Delta\tau)$.

In MATLAB, the shifting operation can be implemented using the commands below.

```
function L = myL(tau, tj, lambda_bar_n, ...
            t, tau0)
lengthT      = length(t);
[~,pos_new] = min(abs(t-tau));
[~,pos_ini] = min(abs(t-tau0));
lambda1_pad = [lambda_bar_n(1)...
                    *ones(1,lengthT), ...
               lambda_bar_n, ...
               lambda_bar_n(end) ...
                    *ones(1,lengthT)];
lambda2_pad = circshift(...
                lambda1_pad, ...
                pos_new-pos_ini);
lambda2      = lambda2_pad(...
                lengthT+1:2*lengthT);
L = -sum(log(interp1(t, ...
            lambda2, tj, 'spline')));
```

The padding is a band-aid for a MATLAB specific command `circshift` which circularly shifts the indices. If we do not pad the array appropriately, a pulse located near the end of the time axis will cause erroneous values to the likelihood function after they get circularly flipped to the beginning of time.

Because of the shifting operations defined above, the function call is significantly harder for automatic differentiation unless we customize the step size. The reason is that if the default step size is smaller than the temporal grid we used to define the pulse, two adjacent indices will remain the same. Thus the gradient will be zero. A workaround solution is to go with a non-gradient optimization. For MATLAB, we can use the package `fminsearch`.

```
fun = @(tau) myL(tau, tj, lambda_n, t, tau0);
tau_hat = fminsearch(fun, t0, option);
```

Approach 3: Zero-finding of the Likelihood Gradient. The third approach we can use is to directly solve for the maximum likelihood solution. Recall from Eq. (46) that the maximum likelihood estimate must be the zero-crossing point of Eq. (46), we can directly implement the derivative as a function call such as the example below.

```
function L = myL_dt(tau, tj, sigma_t, ...
                        alpha, lambda_b)
my_s     =  alpha ...
    * (1/sqrt(2*pi*sigma_t^2)) ...
    * exp( -(tj-tau).^2/(2*sigma_t^2) ) ...
    + lambda_b;
my_s_dot= -alpha*(1/sqrt(2*pi*sigma_t^2)) ...
    * exp( -(tj-tau).^2/(2*sigma_t^2) ) ...
    .* ((tj-tau)./(sigma_t^2));
L    = sum( my_s_dot./my_s );
```

Then, we can use `fzero` to find the zero-crossing point.

```
fun = @(tau) myL_dt(tau, tj, sigma_t, ...
                       alpha, lambda_b);
tau_hat= fzero(fun,tau0);
```

Which approach is faster? Since the goal of this paper is not to provide an algorithm, we skip a formal complexity analysis of the methods. To give readers a rough idea of the comparison between the three approaches, on a typical experiment using the same machine, the runtime is as follows.

	Method	Command	Runtime
1	Gradient descent	`fminunc`	0.064 sec
2	Matched filter	`fminsearch`	0.034 sec
3	Zero crossing	`fzero`	0.016 sec

We remark that all three methods give nearly identical solutions up to the precision of the numerical grid we set.

12.4. Experimental Results

To convince readers that Eq. (49) can be numerically implemented and matches with simulation, we show in Fig. 19 the comparison between the simulation and the theoretical prediction for a range of λ_b values. As is evident from the plot, the theoretical prediction matches very well with the simulation.

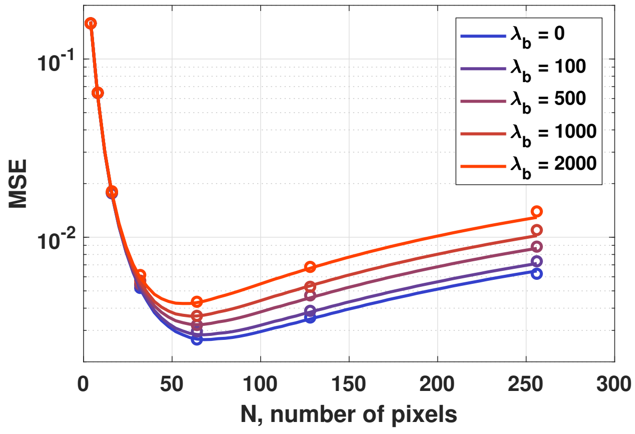

Figure 19. Excellent match between the theoretical bound and the simulation for various levels of ambient noise.

As we have demonstrated in this example, it is sometimes possible to numerically compute the bias and variance so that the theory will match with the simulation. However, by doing so, we will no longer be able to write down the resolution limit in a simple and interpretable closed-form expression. This is not a deficiency of our theory, it is just a sacrifice of clarity and interpretability in exchange for better theoretical precision.

13. Pile-up Effects

Pile up effects refer to the situation where the photo detector responds earlier than the actual arrival of the pulse signal, typically due to the presence of a strong background.

13.1. Distribution

For simplicity, we model the pile up effect as an exponentially decaying distribution in the background, that is,

$$\lambda(x,t) = \alpha s(t-\tau(x)) + \beta\lambda_p(t) + \lambda_b, \tag{50}$$

where $\lambda_p(t) = \gamma e^{-\gamma t}$ is an exponential function parameterized by the reflected laser event rate γ, and the background event rate λ_b [6]. In this case, the sampling of the time stamps will follow from three steps:

- Draw M_s = Poisson(α) samples. The distribution of these M_s samples is the shape of the pulse $s(x,t)$. The parameter α specifies the average number of incident signal photons.
- Draw M_p = Poisson($\beta\gamma$) samples. These M_p samples follow the distribution Exponential(γ) (whose mean is $1/\gamma$). The parameter $\beta\gamma$ specifies the average number of pile-up photons (coming from the background).
- Draw M_b = Poisson($\lambda_b T$) samples, assuming that $0 \leq t \leq T$. These M_b samples follow the distribution Uniform($\lambda_b T$).

An example of the time stamp histogram is shown in Fig. 20.

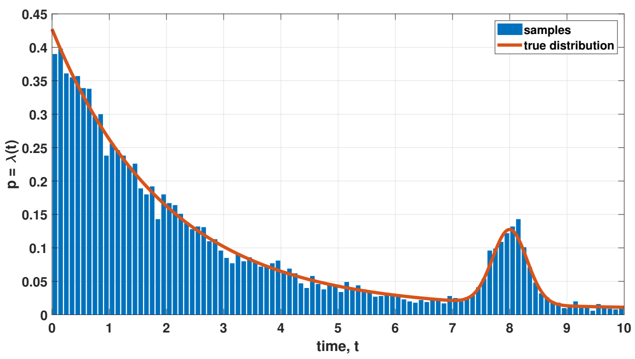

Figure 20. Pile up effect can be modeled as an exponential decaying distribution. Added to the signal pulse $s(t-\tau)$, pile up effect makes the ML estimation of the true delay significantly harder.

13.2. MLE

The maximum likelihood estimation in the presence of the pile up effect needs a separation of background and signal. Because of the simplified model we choose (for the purpose of theoretical illustration), we can exploit the oracle knowledge we know about the pile up and background. Given $\lambda(x,t)$, we know that the binning is such that

$$\begin{aligned}\lambda_n(t) &= \int_{\frac{n}{N}}^{\frac{n+1}{N}} \lambda(x,t)dx \\ &= \underbrace{\alpha\int_{\frac{n}{N}}^{\frac{n+1}{N}} s(t-\tau(x))dx}_{=s_n(t-\tau_n)} + \int_{\frac{n}{N}}^{\frac{n+1}{N}} [\beta\lambda_p(t)+\lambda_b]dx \\ &= \alpha s_n(t-\tau_n) + \frac{\beta\lambda_p(t)+\lambda_b}{N},\end{aligned}$$

Where we approximate the integral by a simplified function $s_n(t)$ delayed by a time constant τ_n.

Given this decomposition of $\lambda_n(t)$ as the signal $s_n(t)$ and the noise part $\beta\lambda_p(t) + \lambda_b$, we use Theorem 2 to estimate the variance of the nth pixel:

$$\mathbb{E}[(\widehat{\tau}_n - \tau_n)^2] = \int_{-T}^{T} \frac{[\alpha\dot{s}_n(t)]^2}{\lambda_n(t)}dt.$$

This integration needs to be evaluated numerically.

In terms of simulation, we need to run the ML estimation. The ML estimation requires us to search for an optimal τ_n such that the candidate distribution matches with the measurement. To this end, we follow the procedure outlined in Sec. 12.3 to numerically run a search scheme to pick up the ML estimate.

13.3. Theoretical MSE

A comparison between the simulated MSE and the theoretically predicted MSE is shown in Fig. 21. For this particular example, we set $\alpha = 1\times10^4$, $\beta = 1\times10^5$, and $\lambda_b = 100$. The pile up distribution has a constant $\gamma = 4$. Again, we emphasize that these units are unit-free, in the sense they are chosen to illustrate the validity of the theorem rather than matching any particular real sensor.

As we can see in Fig. 21, the theoretical prediction matches extremely well with the simulated MSE. The small deviation towards a larger N is due to the fact that the ML estimation faces difficulty when the total number of samples is few. As an example, when $N = 256$, the number of samples per pixel is around 400 whereas when $N = 8$, the number of pixels is around 28,000. When there is no pile up noise, the variance of ML estimate is basically limited by the number of samples. But in the presence of complicated noise (such as pile up), the ML estimation procedure has limitations where it cannot differentiate signal from noise. Therefore, when the number of samples is few, the simulation performance will be worse than what the theory predicts. Nevertheless, Fig. 21 still shows a high degree of match between the theory and the simulation.

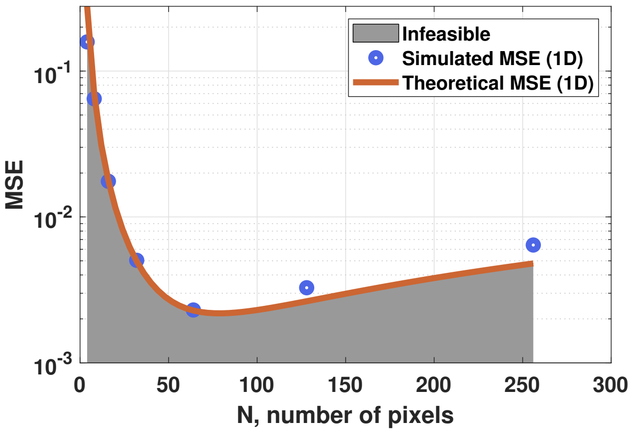

Figure 21. MSE comparison in the presence of pile up. The theoretically predicted MSE has a very good match with the simulated MSE.

14. Dark Count

The model we presented in the main text does not consider dark count. In this section, we briefly discuss how it can be included.

Dark count is a type of sensor noise caused by the random generation of electrons in the depletion region. These random electrons are present even when there is no photoelectric event. Therefore, one way to model dark count is to treat it as a uniform noise floor that does not change over time and does not depend on the scene flux. However, dark counts scale with the sensing area. So, if there are N pixels and if we assume that the sensing area scales linearly with N, then the dark counts will be reduced by N times on average. Therefore, a simple model in the context of this paper is to write

$$\lambda(t) = \frac{\alpha}{N} s(t-\tau) + \frac{\lambda_b}{N} + \frac{\lambda_{\text{dark}}}{N}. \tag{51}$$

Because of this simple addition to the background, we can treat λ_{dark} as part of the background noise. In a typical scenario when the dark count rate is in the order 10 per second while a SPAD can count up to 1 million photons per second (e.g., ThorLab's single-photon detectors), the impact of dark current is not visible unless we operate it in extremely low-light.

15. Extension to 2D

In this section, we briefly summarize how the results can be extended to 2D. We will derive the bias and the variance.

15.1. Bias

For the bias term, in 2D, we need to consider a 2D coordinate $\mathbf{x} \in [0,1] \times [0,1]$. The time of arrival function τ is denoted as $\tau(\mathbf{x})$. The average function $\overline{\tau}(\mathbf{x})$ is

$$\overline{\tau}(\mathbf{x}) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \tau_{m,n} \varphi(Mx-m)\varphi(Ny-n),$$

Where $\tau_{m,n}$ is the (m,n)th value defined as

$$\tau_{m,n} = \int_{\frac{m}{M}}^{\frac{m+1}{M}} \int_{\frac{n}{N}}^{\frac{n+1}{N}} \tau(x,y)\, dx\, dy.$$

The derivative of the time of arrival function, in the 2D case, will become the gradient. This means that

$$\nabla_{\mathbf{x}} \tau(\mathbf{x}) = \left[\frac{\partial \tau}{\partial x}, \frac{\partial \tau}{\partial y}\right]^T.$$

For notational simplicity, we write $\nabla_{\mathbf{x}}\tau(\mathbf{x}) = \mathbf{c} = [c^x, c^y]^T$. If we are interested in the gradient of the (m,n)th pixel, we write

$$\mathbf{c}_{m,n} = [c^x_{m,n}, c^y_{m,n}]^T = \nabla_{\mathbf{x}}\tau(\mathbf{x}_{m,n}).$$

The magnitude square of the gradient is denoted as $\|\mathbf{c}_{m,n}\|^2 = (c^x_{m,n})^2 + (c^y_{m,n})^2$, and the overall gradient summed over all the pixels is

$$\|\mathbf{c}\|^2 = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \|\mathbf{c}_{m,n}\|^2 \approx \int_0^1 \int_0^1 \|\nabla_{\mathbf{x}}\tau(\mathbf{x})\|^2 \, d\mathbf{x},$$

where the last approximation holds by assuming that (m,n) covers the unit squares.

The main result for the bias term in the 2D case is summarized in the lemma below.

Lemma 9. *Let $\tau(\mathbf{x})$ be a 2D time of arrival function over the unit square $\mathbf{x} \in [0,1] \times [0,1]$. Suppose that we use $M \times N$ pixels to approximate the unit square, and for simplicity assume that $M = N$. If we approximate $\tau(\mathbf{x})$ using a piecewise constant function $\overline{\tau}(\mathbf{x})$, the bias is given by*

$$\textit{Bias} = \frac{\|\mathbf{c}\|^2}{12N^2}. \tag{52}$$

Proof of Lemma 9.

We start with the definition of the bias:

$$\begin{aligned} \text{Bias} &= \int_0^1 \int_0^1 \left(\overline{\tau}(\mathbf{x}) - \tau(\mathbf{x})\right)^2 d\mathbf{x} \\ &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \underbrace{\int_{\frac{m}{M}}^{\frac{m+1}{M}} \int_{\frac{n}{N}}^{\frac{n+1}{N}} \left(\tau_{m,n} - \tau(x,y)\right)^2 dx\, dy}_{=e^2_{m,n}}. \end{aligned}$$

Let $\mathbf{x}_{m,n}$ be the mid-point such that

$$\mathbf{x}_{m,n} = \left[\frac{2m+1}{2M}, \frac{2n+1}{2N}\right].$$

We can approximate the first order approximation to $\tau(\mathbf{x})$:

$$\tau(\mathbf{x}) = \tau(\mathbf{x}_{m,n}) + \mathbf{c}_{m,n}^T(\mathbf{x} - \mathbf{x}_{m,n}),$$

where $\mathbf{c}_{m,n} = \nabla\tau(\mathbf{x}_{m,n})$ is the gradient. Then, the error is

$$\begin{aligned} e_{m,n}^2 &= \int_{\frac{m}{M}}^{\frac{m+1}{M}} \int_{\frac{n}{N}}^{\frac{n+1}{N}} (\tau_{m,n} - \tau(\mathbf{x}))^2 \, d\mathbf{x} \\ &= \int_{\frac{m}{M}}^{\frac{m+1}{M}} \int_{\frac{n}{N}}^{\frac{n+1}{N}} \Big(\nabla\tau(\mathbf{x}_{m,n})^T(\mathbf{x} - \mathbf{x}_{m,n})\Big)^2 \, d\mathbf{x} \\ &= \left[c_{m,n}^x\right]^2 \frac{1}{12M^3N} + \left[c_{m,n}^y\right]^2 \frac{1}{12N^3M}. \end{aligned}$$

Assuming $M = N$, the above is simplified to

$$e_{m,n}^2 = \frac{\left[c_{m,n}^x\right]^2 + \left[c_{m,n}^y\right]^2}{12N^4} = \frac{\|\mathbf{c}_{m,n}\|^2}{12N^4}.$$

Summing over all (m,n)'s will give us

$$\begin{aligned} \text{Bias} &= \int_0^1 \int_0^1 (\overline{\tau}(\mathbf{x}) - \tau(\mathbf{x}))^2 \, d\mathbf{x} \\ &= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \frac{\|\mathbf{c}_{m,n}\|^2}{12N^4} = \frac{\|\mathbf{c}\|^2}{12N^2}. \end{aligned}$$

Visualization of the Bias. A visualization of the bias can be seen in Fig. 22. In this example, we use a real depth map as the ground truth $\tau(\mathbf{x})$. We scale the depth map so that the maximum value is 20, and the minimum value is 10. The full resolution is 512×512. To reduce the discontinuity of the depth map at object boundaries, we apply a spatial low-pass filter to smooth out the edges. The estimated gradient magnitude is $\|\mathbf{c}\|^2 = \int_0^1 \nabla_{\mathbf{x}}\tau(\mathbf{x})d\mathbf{x} = 1.42 \times 10^3$. We consider four different resolutions 8×8, 16×16, 32×32, and 128×128. For each resolution, we use the equation outlined in Lemma 9 to calculate the theoretically predicted bias. We compare this predicted bias with the simulation. As we can see from Fig. 22, the theoretical prediction offers an excellent match with the simulation.

15.2. Variance

For the variance term, we note that the derivation follows similarly to the 1D case:

$$\begin{aligned} \text{Var} &= \mathbb{E}\left[\int_0^1 \int_0^1 (\widehat{\tau}(\mathbf{x}) - \overline{\tau}(\mathbf{x}))^2 \, d\mathbf{x}\right] \\ &= \mathbb{E}\left[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \int_{\frac{m}{M}}^{\frac{m+1}{M}} \int_{\frac{n}{N}}^{\frac{n+1}{N}} (\widehat{\tau}(\mathbf{x}) - \overline{\tau}(\mathbf{x}))^2 \, d\mathbf{x}\right] \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \int_{\frac{m}{M}}^{\frac{m+1}{M}} \int_{\frac{n}{N}}^{\frac{n+1}{N}} \mathbb{E}\left[(\widehat{\tau}_{m,n} - \overline{\tau}_{m,n})^2\right] \, d\mathbf{x}. \end{aligned}$$

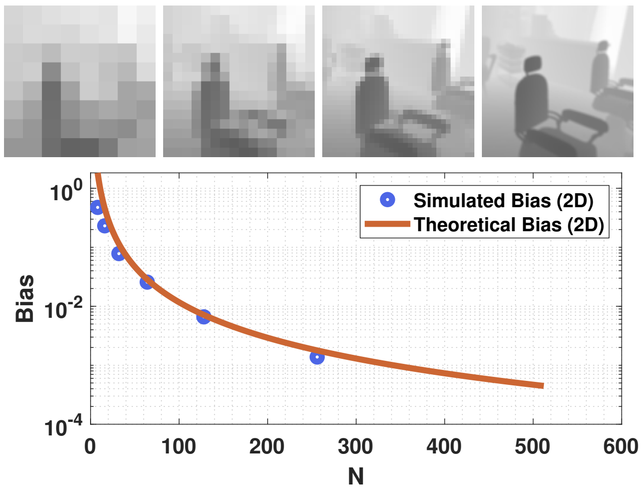

Figure 22. Visualization of the bias in 2D. The full resolution of the depth map is 512×512. We consider multiple resolutions to study the effect of the bias. This plot highlights the excellent match between the theoretical prediction and the simulation results.

Therefore, as long as we can calculate $\mathbb{E}\left[(\widehat{\tau}_{m,n} - \overline{\tau}_{m,n})^2\right]$, we will be able to determine the variance.

Based on the arguments we described before stating the lemma, we shall focus on deriving the variance for each individual pixel. To this end, we need to derive the effective return pulse $\lambda_{m,n}(\mathbf{x}, t)$. Following the steps of the proof of Theorem 3, the critical step is the convolution of a 2D spatial Gaussian and a 1D temporal Gaussian

$$\begin{aligned} \text{2D spatial Gaussian} &= \mathcal{N}(\mathbf{x} \,|\, 0, \sigma_s^2\mathbf{I}) \\ \text{1D temporal Gaussian} &= \mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x} - \mathbf{x}_0), \sigma_t^2). \end{aligned}$$

Here, $\sigma_s = 1/(\sqrt{12}N)$ is the spatial radius of the 2D Gaussian, which serves the same role as σ_x in the 1D case. The linear approximation $\tau_0 + \mathbf{c}_0^T(\mathbf{x} - \mathbf{x}_0)$ is the 2D version of the 1D linear approximation $\tau_0 + \tau'(x_0)(x - x_0)$, where $\mathbf{x}_0$ is the center pixel in each interval of the grid, and $\mathbf{c}_0$ is the gradient at $\mathbf{x}_0$.

Before we proceed to the proof, we state the main result.

Lemma 10. *Let $\tau(\mathbf{x})$ be a 2D time of arrival function over the unit square $\mathbf{x} \in [0,1] \times [0,1]$. Suppose that we use $M \times N$ pixels to approximate the unit square, and for simplicity assume that $M = N$. Let α_0 be the total flux over the unit square, then The variance is given by*

$$Var = \frac{N^2}{\alpha_0}\left(\|\mathbf{c}\|^2\sigma_s^2 + \sigma_t^2\right), \tag{53}$$

where $\sigma_s = 1/(\sqrt{12}N)$ is the spatial radius of the Gaussian approximation, and σ_t is the temporal spread of the pulse.

Proof of Lemma 10. The core derivation is the convolution between the 2D Gaussian in space and the 1D Gaussian in time. By separability of a 2D Gaussian, we can write the 2D Gaussian as the convolution of two orthogonal 1D Gaussian functions:

$$\mathcal{N}(\mathbf{x} \,|\, 0, \sigma_s^2 \mathbf{I}) = \mathcal{N}(x \,|\, 0, \sigma_s^2) \circledast \mathcal{N}(y \,|\, 0, \sigma_s^2).$$

The order of x versus y does not matter. Then, substituting it into the main convolution, we can perform two consecutive convolutions:

$$\begin{aligned}\mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x}-\mathbf{x}_0),\ \sigma_t^2) &\circledast \mathcal{N}(\mathbf{x} \,|\, 0, \sigma_s^2 \mathbf{I}) \\ &= \mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x}-\mathbf{x}_0),\ \sigma_t^2) \circledast \mathcal{N}(x \,|\, 0, \sigma_s^2) \\ &\qquad \circledast \mathcal{N}(y \,|\, 0, \sigma_s^2)\end{aligned}$$

The first convolution, following the derivation of Theorem 3, will give us

$$\begin{aligned}&\mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x}-\mathbf{x}_0),\ \sigma_t^2) \circledast \mathcal{N}(x \,|\, 0, \sigma_s^2) \\ &= \mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x}-\mathbf{x}_0),\ \sigma_t^2 + [c_0^x]^2\sigma_s^2),\end{aligned}$$

where c_0^x is the horizontal component of the gradient $\mathbf{c}_0 = [c_0^x, c_0^y]^T$.

The second convolution, which is applied after the first convolution, will give us

$$\begin{aligned}&\mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x}-\mathbf{x}_0),\ \sigma_t^2 + [c_0^x]^2\sigma_s^2) \circledast \mathcal{N}(y \,|\, 0, \sigma_s^2) \\ &= \mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x}-\mathbf{x}_0),\ \sigma_t^2 + [c_0^x]^2\sigma_s^2 + [c_0^y]^2\sigma_s^2) \\ &= \mathcal{N}(t \,|\, \tau_0 + \mathbf{c}_0^T(\mathbf{x}-\mathbf{x}_0),\ \sigma_t^2 + \|\mathbf{c}_0\|^2\sigma_s^2).\end{aligned}$$

Restricting ourselves to $\mathbf{x} = \mathbf{x}_0$, we can show that the effective return pulse is

$$\lambda_{m,n}(t) = \alpha \mathcal{N}(t \,|\, \tau_{m,n}, \|\mathbf{c}_{m,n}\|^2\sigma_s^2 + \sigma_t^2) + \lambda_b.$$

Finally, we recognize that since there are MN pixels in the unit space, (and assuming that $M = N$ for simplicity), then each pixel will see a total flux of α_0/N^2. Therefore,

$$\begin{aligned}\mathbb{E}\left[(\widehat{\tau}_{m,n} - \overline{\tau}_{m,n})^2\right] &= \frac{\text{variance of } \lambda_{m,n}(t)}{\alpha} \\ &= \frac{N^2}{\alpha_0}\left(\|\mathbf{c}_{m,n}\|^2\sigma_s^2 + \sigma_t^2\right).\end{aligned}$$

Summing over m and n will give us

$$\begin{aligned}\text{Var} &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\int_{\frac{m}{M}}^{\frac{m+1}{M}}\int_{\frac{n}{N}}^{\frac{n+1}{N}} \mathbb{E}\left[(\widehat{\tau}_{m,n} - \overline{\tau}_{m,n})^2\right] d\mathbf{x} \\ &= \frac{N^2}{\alpha_0}\left(\|\mathbf{c}\|^2\sigma_s^2 + \sigma_t^2\right).\end{aligned}$$

Visualizing the Variance. Fig. 23 shows the visualization of the variance as a function of N. The experimental configuration is identical to the one we used in the 2D bias analysis. For the purpose of analyzing the variance, we need to further set up a Monte Carlo simulation. To this end, we first set the spatial spacing of the grid as $\Delta x = 1/512$ along each direction. The total amount of flux is set as $\alpha_0 = 1e^6$. Suppose that we have N pixels on each side, the unit square will have N^2 pixels. Each pixel will therefore observe α_0/N^2 amount of flux. When $N = 8$, the per pixel flux is 15625 photons per pixel. When $N = 256$, the per pixel flux is 15.26 photons per pixel.

For simplicity, we assume that the pulse $s(t)$ is Gaussian. The width of the pulse is 2 arbitrary units, which is roughly 10 percent of the true depth value. We also assume that the noise floor is $\lambda_b = 0$. These two assumptions allow us to use a simple ensemble average as the maximum likelihood estimator. Otherwise, we need to numerically implement the matched filter.

In this particular experiment, we use $\sigma_s = 1/(\sqrt{12}N)$ as the spatial radius, and $\|\mathbf{c}\|^2 = 1.42 \times 10^3$. The variance is calculated for each pixel and then summed over the entire unit square. A total number of 1000 random trials are conducted to obtain the variance per pixel (and for the whole image).

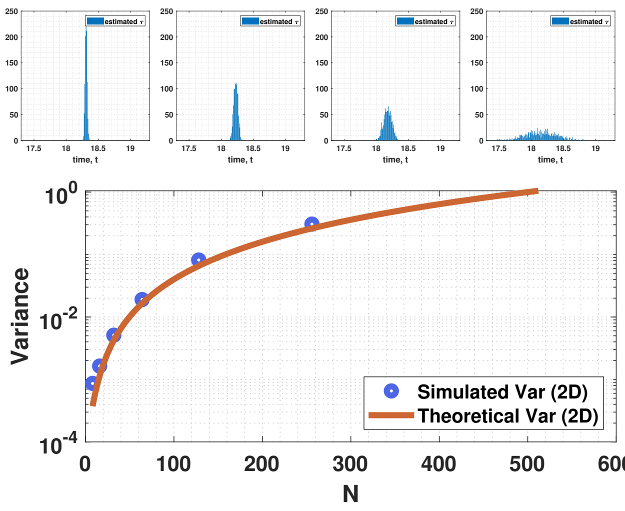

Figure 23. Visualization of the variance for a 2D example. For every pixel in the scene, we run a Monte Carlo simulation to compute the variance of the estimated depth values. The histograms shown in this figure correspond to the four resolutions we showed in the bias case. As we increase the spatial resolution (so that each pixel becomes smaller), the histograms become flat and so the variance increases. The plot at the bottom highlights the excellent match between the theoretically predicted variance and the simulated variance.

16. Real 2D Experiment

In this section, we provide additional detail on how real 2D experiments are conducted.

16.1. Snapshot of Real Data

As depicted in the main text, the real data is collected by a 192×128 SPAD array previously reported by Henderson et al. [3]. The system has a time-to-digital converter (TDC) timing resolution of 35 ps. For convenience, we crop the center 128×128 region for analysis. Fig. 24 shows a one-frame snapshot of the real data, and the 10,000-frame average. We observe that the 10,000-frame average is noisy, although the shape of the object is visible.

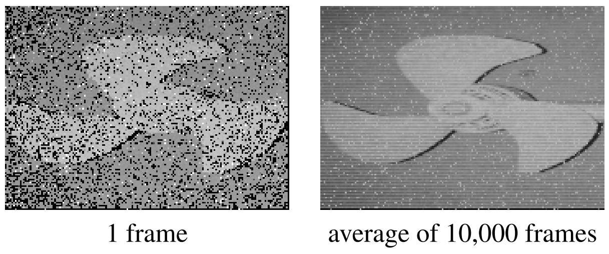

Figure 24. Raw data produced by the SPAD array.

Inspecting the source of the noise, we plot the histogram of one of the pixels of the data volume in Fig. 25. We recognize two issues in the data: (i) There is a secondary pulse, likely caused by secondary bounces from the background; (ii) there is a strong peak happening at the beginning (and sometimes at the end) of the sensing period, likely caused by failed detections of the SPAD. We did not show these samples in the histogram because it is just a strong spike in the histogram. The presence of these two issues makes the average of the time stamps problematic, hence generating noise.

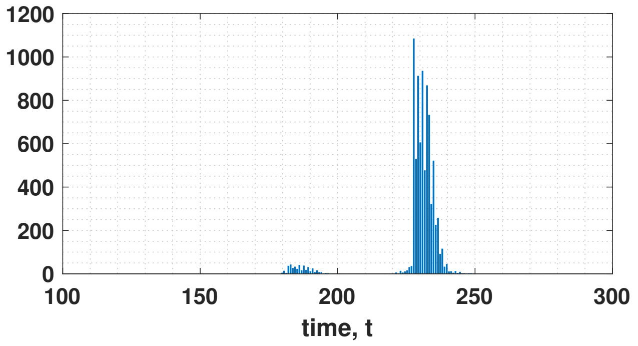

Figure 25. Histogram of one of the pixels in the SPAD data array.

16.2. Pseudo Ground Truth

For the purpose of MSE analysis, we need to construct a pseudo ground truth — a ground truth signal generated from all the available samples, hoping as noise free as possible. To achieve this goal, we need to reject the outliers so that we can retain only the primary pulse.

Our pre-processing step is done by identifying the center of the primary pulse. Once the center is identified, we crop around the center histogram with $\pm 3\sigma_t$ where σ_t is the standard deviation of the pulse.

Two remarks are worth mentioning: (i) The identification of the center of the primary pulse involves a few steps. The first step is to find a coarse estimate of the center. For this purpose, we compute the mean of the entire 10,000 data points and crop a large temporal window around the mean. We smooth out the histogram so that we can pick a more reliable peak. The smoothing step is done by adding Gaussian random variables. (ii) Once the peak is identified, we move to the second stage by putting a small temporal window around the peak. We reject all samples that fall outside this small temporal window, thus effectively removing the secondary pulse and the spike in the beginning and at the end.

After we have cleaned up the data, we can plot the resulting histogram as shown in Fig. 26. The pulse is significantly more concentrated around the true peak. The resulting depth map is slightly over-smoothed. However, since all our subsequent analysis is done based on the processed histogram (as we will be re-sampling with replacement from the processed histogram), computing the MSE with respect to this smoothed depth map will not cause issues to the analysis.

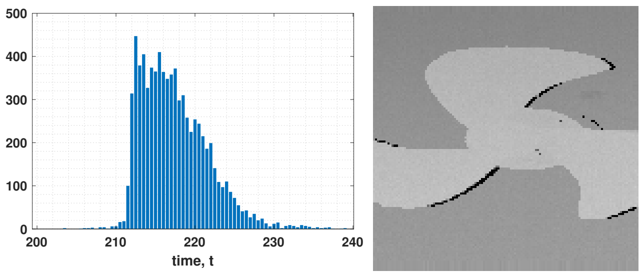

Figure 26. [Left] The processed histogram of one of the pixels and [Right] the resulting depth map.

16.3. Estimating σ_t and α_0

In the theoretical analysis of the variance, we need to know the pulse width σ_t and the total amount of flux α_0.

For σ_t, we use the histogram shown in Fig. 26 to compute the variance at every pixel location. Although the pulse is asymmetric, we treat it as a symmetric pulse and compute the standard deviation regardless. We repeat the process for every pixel, and we generate a map of σ_t as shown in Fig. 27. For simplicity, we use the average of the σ_t in this map as the true value for our theoretical analysis. Our estimated σ_t is $\sigma_t = 3.93$.

For α_0, we again use the histogram shown in Fig. 26 to count the number of elements in the histogram. Recall that

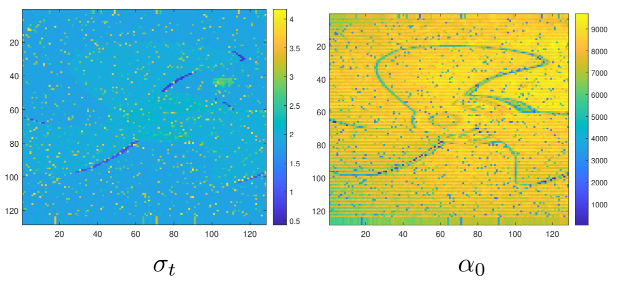

Figure 27. The values of σ_t and α_0 over the unit square.

during the pre-processing stage, we have already eliminated all the outliers from the raw data. This will leave us with a smaller set of samples compared to the original data array. The number of active samples will inform us about the number of photons arriving at the sensor. During our analysis, we will sample with replacement from the histogram. For each trial, we draw K samples from the available data points per pixel. Therefore, the total flux we have in the data is the sum of the values in Fig. 27, divided by 10,000 because we have 10,000 frames, and multiplied by K. Approximately our α_0 is 3.86×10^4.

16.4. Variance Estimation via Bootstrap

The main tool we use to evaluate the variance is bootstrap. The idea is that given a dataset of M data points, we sample with replacement K samples and calculate the estimate of interest. The variance of the estimate is called the bootstrapped variance. When we repeat the random trials for long enough, the bootstrapped variance will converge to the true variance in probability.

Following this idea, for every pixel, we sample with replacement $K = 3$ times to generate the samples. From these samples, we construct the ML estimate (for every pixel) where the ML estimation is done by taking the simple average of the samples. Here, we assume that the pulse is symmetric even though it is not. We find that this approximation does not cause too many issues. Once the ML estimate is obtained, we repeat the process 20 times to evaluate the variance.

For lower resolutions, we bin the samples to form a bigger pool. If we use a 2×2 bin, then the total number of samples to be generated for the bootstrap purpose is $K \times 2^2 = 12$ samples; If we use a 3×3 bin, then the total number of samples is $K \times 3^2 = 27$ samples. These samples will give us an estimate of the real variance of the data.

The theoretical formula of the variance is based on a simpler form $\sigma_t^2 N^2/\alpha_0$. We omitted the $\|\mathbf{c}\|^2\sigma_s^2$ because $\|\mathbf{c}\|^2\sigma_s^2$ is small compared to σ_t^2. (Recall that $\sigma_s = 1/(\sqrt{12}N)$).

16.5. Bias Estimation via Numerical Integration

The bias is computed numerically via integration. The reason is that the pseudo ground truth depth map is overall a piecewise constant function. As we explained in the limitation subsection of the Bias section, there is no simple analytic formula for calculating the bias for piecewise constant functions unless we perform the numerical integration. The numerical integration is done by downsampling and upsampling the cleaned raw data volume. For example, to calculate the bias caused by a 2×2 bin, we take the average over all the available samples in the $2 \times 2 \times T$ (where T is the number of available samples). After binning, we scale it back to the full resolution and compute the sum squared error with respect to the pseudo ground truth.

17. Q&A

In this section, we list a few questions and answers which might be of interest to readers.

Q1. Can you just use as many pixels as possible during acquisition, and then perform binning (of the time stamps) as post-processing?

Answer: Yes, this is completely possible. In fact, in our real data analysis, we see that when the pulse is short enough, the MSE decays monotonically with the N. Theoretically, the optimal N still exists but this N could be larger than what the physical resolution of the sensor can support. In this case, we should just maximize the resolution.

From a practical point of view, we also agree that post-processing of noisy time stamps can be cost-effective. Analog processing on the sensor front could also be a solution.

Q2. What if you denoise the estimated time of arrival map? Will it beat your MSE bound?

Answer: Yes, it will. The theory we presented here uses the maximum likelihood (ML) estimation. ML estimation allows us to say things concretely so that we can offer a simple and interpretable MSE estimate. If we do denoising, then we will be doing maximum a posteriori or minimum mean square estimation. In those cases, we need to specify the prior distribution for which no one has a formula. Even if we do pick a prior (e.g., total variation), the derivation of the MSE will be substantially harder if not intractable. So, we lose the capability of writing a simple and interpretable formula. In short, while we are almost certain that a well-defined post-processing will “beat” our MSE bound, this “victory” offers little to no theoretical benefit to the scope of this paper.

Q3. What is the utility of this paper?

Answer: The MSE we show in this paper is, in our honest opinion, simple, interpretable, and elegant. For the first time in the LiDAR literature, we provide the exact characterization of the resolution limit.

Q4. You need to show more comparisons by sending the raw time stamps through a state-of-the-art depth reconstruction neural network.

Answer: Beating a SOTA depth reconstruction neural network is not the purpose of this paper.

Q5. Is MSE the right metric?

Answer: Yes, if you want to derive equations, especially a simple equation to give you intuitions about the problem. No, if you are more interested in practical scenarios. There is no better or worse.

Q6. Your model is inaccurate. It made too many assumptions such as Gaussian pulse, single-bounce, no dark current, constant reflectivity, etc.

Answer: While we also want to be as accurate as possible, accuracy and simplicity are mutually exclusive in this paper. As we have explained in the supplementary material, all these situations can be handled *numerically* by integrations. But this will defeat the purpose of deriving a closed-form expression.

Q7. Some papers in the literature use the Markov chain/self-excitation process to model the photon arrivals. Why are you skipping all these?

Answer: We are just assuming that there is no dead time. If there is dead time, then you are correct that self-excitation processes are needed to provide a better model. However, this would be substantially more complicated than what we present here. By assuming no dead time, we can go back to the standard inhomogeneous Poisson process by exploiting independence. Even in this significantly simplified case, we see that the derivation of the MSE is non-trivial.

Q8. Can your model handle fog?

Answer: Yes, but it will be complicated. Scattering medium such as fog affects the reflectivity α_0 and causes additional background as shown in Figure 16. You will need numerical integration to evaluate the bound.

References

[1] Israel Bar-David. Communication under the poisson regime. *IEEE Transactions on Information Theory*, 15(1):31–37, 1969. 1, 2, 4

[2] Stanley H. Chan. *Introduction to Probability for Data Science*. Michigan Publishing, 2021. 10

[3] Robert K. Henderson, Nick Johnston, Francescopaolo Mattioli Della Rocca, Haochang Chen, David Day-Uei Li, Graham Hungerford, Richard Hirsch, David Mcloskey, Philip Yip, and David J. S. Birch. A 192 $\times$ 128 time correlated SPAD image sensor in 40-nm CMOS technology. *IEEE Journal of Solid-State Circuits*, 54(7):1907–1916, 2019. 18

[4] Dongeek Shin, Ahmed Kirmani, Vivek K Goyal, and Jeffrey H. Shapiro. Information in a photon: Relating entropy and maximum-likelihood range estimation using single-photon counting detectors. In *2013 IEEE International Conference on Image Processing*, pages 83–87, 2013. 5

[5] Donald L. Snyder and Michael I. Miller. *Random Point Processes in Time and Space*. Springer, 1991. 1

[6] Alessandro Tontini, Sonia Mazzucchi, Roberto Passerone, Nicolò Broseghini, and Leonardo Gasparini. Histogram-less LiDAR through SPAD response linearization. *IEEE Sensors Journal*, 24(4):4656–4669, 2024. 14