Single-Photon Structured Light
Supplementary Material

Varun Sundar † Sizhuo Ma † Aswin C. Sankaranarayanan‡ Mohit Gupta †
† University of Wisconsin-Madison ‡ Carnegie Mellon University

Supplementary Note 1. Deriving Expected Error

In this supplementary note, we derive the decoding error probability of Gray codes and repetition codes subject to the bit-flip noise model (Sec. 3) of Single-Photon SL. We provide an analytic expression for error probability in BCH codes for a given decoding function and for the particular case of a bounded-error decoder. Additionally, we derive an upper-bound on the average decoding error (in terms of absolute deviation) for binary shifted patterns. Finally, we outline the Monte-Carlo simulation procedure to compare various strategies—such as BCH and repetition strategies in Fig. 6, and Hybrid and repetition strategies in Fig. 8.

1.1. Notation

Let $\mathcal{M} = \{\mathbf{m} \mid \mathbf{m} \in \{0,1\}^L\}$ denote the set of L-bit binary messages, e.g., Gray codes, that represent columns of a projector. For simplicity, we shall assume that the projector has 2^L columns, but the following analysis extends mutatis mutandis to the more general case. Further, let $\mathbf{1_m}$ and $\mathbf{0_m}$ denote the number of 1's and 0's in L-bit message $\mathbf{m}$ respectively.

1.2. Gray Codes

When transmitting a Gray code $\mathbf{m}$ an error occurs if any of the bits are flipped due to photon-noise, since no error-correcting strategy is employed. Assuming $\mathbf{y}$ denotes the received (possibly corrupted) codeword when $\mathbf{m}$ is projected, the expected error probability over the set of messages $\mathcal{M}$ is given as

$$
\begin{aligned}
\Pr\{\text{error}\} &= \sum_{\mathbf{m}\in\mathcal{M}} \Pr\{\mathbf{m}\}\Pr\{\text{error}\,|\,\mathbf{m}\} \\
&= \frac{1}{2^L}\sum_{\mathbf{m}\in\mathcal{M}} \Pr\{\mathbf{y}\neq\mathbf{m}\,|\,\mathbf{m}\} && \text{(Each message is equally likely to be transmitted)} \\
&= \frac{1}{2^L}\sum_{\mathbf{m}\in\mathcal{M}} 1-\Pr\{\mathbf{y}=\mathbf{m}\,|\,\mathbf{m}\} \\
&= \frac{1}{2^L}\sum_{\mathbf{m}\in\mathcal{M}} 1-\prod_{i=1}^{L}\Pr\{\mathbf{y}_i=\mathbf{m}_i\} \\
&= 1-\frac{1}{2^L}\sum_{\mathbf{m}\in\mathcal{M}} \Pr\{\mathbf{y}_i=1\,|\,\mathbf{m}_i=1\}^{\mathbf{1_m}}\Pr\{\mathbf{y}_i=0\,|\,\mathbf{m}_i=0\}^{\mathbf{0_m}} \\
&= 1-\frac{1}{2^L}\sum_{i=0}^{L}\binom{L}{i}\Pr\{\mathbf{y}_i=1\,|\,\mathbf{m}_i=1\}^{i}\Pr\{\mathbf{y}_i=0\,|\,\mathbf{m}_i=0\}^{L-i} \\
&= 1-\frac{1}{2^L}\left(\Pr\{\mathbf{y}_i=1\,|\,\mathbf{m}_i=1\}+\Pr\{\mathbf{y}_i=0\,|\,\mathbf{m}_i=0\}\right)^{L} \\
&= 1-\left(1-\frac{(\mathrm{P}_{\text{flip, bright}}+\mathrm{P}_{\text{flip, dark}}))}{2}\right)^{L} && \text{(S1)}
\end{aligned}
$$

where, we make use of Eq. (3) and Eq. (4) in the last step. Note that the decoding error is zero when $\mathrm{P}_{\text{flip, bright}}, \mathrm{P}_{\text{flip, dark}} \to 0$.

1.3. Repeated Gray Codes

Assume we repeat $\mathbf{m}$, the L-bit message, r times and decode it with a majority vote. Let $\mathbf{x}$ denote the received codeword and $\mathbf{y}$ the decoded message by using a majority vote. Then each bit y_i is decoded incorrectly if more than or equal $\lceil \frac{r-1}{2} \rceil$ bit-flips occur, with probability

$$\begin{aligned} \Pr\{\mathbf{y}_i \neq \mathbf{m}_i \,|\, \mathbf{m}_i = 1\} &= \prod_{j=\lceil \frac{r-1}{2} \rceil}^{r} \binom{r}{j} \mathrm{P}_{\text{flip, bright}}^{j} (1 - \mathrm{P}_{\text{flip, bright}})^{r-j} \\ &= g_r(\mathrm{P}_{\text{flip, bright}}) \end{aligned} \tag{S2}$$

where $g_r(p) = \prod_{j=\lfloor \frac{r-1}{2} \rfloor}^{r} p^r (1-p)^{r-j}$. Similarly, we can show that

$$\Pr\{\mathbf{y}_i \neq \mathbf{m}_i \,|\, \mathbf{m}_i = 0\} = g_r(\mathrm{P}_{\text{flip, dark}})$$

With this, we can evaluate decoding error probability for the repetition strategy by extending Eq. (S1) as

$$\Pr\{\text{error}\} = 1 - \left(1 - \left(\frac{g_r\left(\mathrm{P}_{\text{flip, bright}}\right) + g_r\left(\mathrm{P}_{\text{flip, dark}}\right)}{2}\right)\right)^{L} \tag{S3}$$

Note that, $g_r(p) \geq p \;\forall\; p \in [0, 1]$, so repetition always improves decoding reliability when $\mathrm{P}_{\text{flip, dark}}, \mathrm{P}_{\text{flip, bright}} < 0.5$.

1.4. BCH Codes

Construction. A BCH$(n, k, d) : \{0,1\}^k \to \{0,1\}^n$ encoder takes input messages of length k and produces output codewords of length n that are at least d-bits apart. In this paper, we consider primitive, narrow-sense BCH codes over the finite field GF(2). We present a brief summary of their construction here and refer readers to Roth [11] for a more detailed explanation.

1. Pick α a primitive element of GF(2^n).
2. Compute the generator polynomial $g(x) = \text{lcm}\,(q_1(x), ..., q_{d-1}(x))$, where $q_i(x)$ are minimal polynomials of α^i with coefficients in GF(2).
3. Each codeword is obtained by multiplying a polynomial representing the message $p(x)$ and the generator polynomial $g(x)$.
4. For systematic encoding, we set $p(x) = m(x)x^{n-k} - r(x)$, where $m(x)$ is the polynomial with the symbols of message $\mathbf{m}$ as coeffecients and $r(x) = m(x)x^{n-k} \bmod g(x)$.

From the construction, it can be seen that BCH codes are a subset of Reed-Solomon codes over the finite field GF(2^n) [6]. Specifically, $\mathcal{C}_{\text{BCH}}(n, k, d) = \mathcal{C}_{\text{RS}(n,k,d)} \cap \{0,1\}^n$, where RS$(n, k, d)$ represents the Reed-Solomon encoder with parameters (n, k, d) as before.

Decoding Error. Let $\mathbf{c}$ denote a codeword corresponding to BCH encoded message $\mathbf{m}$. As before, assume we receive (possibly corrupted) bits $\mathbf{x}$ which we decode to obtain $\mathbf{y}$. Depending on whether decoder gives up beyond the worst-case error limit (e.g. Berlekamp Massey [9]) or not (e.g. minimum distance decoding), we can derive two expressions. For decoders that work up to the worst-case error limit of $\lfloor \frac{d-1}{2} \rfloor$ bit-flips,

$$\begin{aligned}
\Pr\{\mathbf{y} \neq \mathbf{m}\} &= \Pr\left\{ d_H(\mathbf{x}, \mathbf{c}) > \frac{d-1}{2} \right\} && \text{(where } d_H \text{ is the Hamming distance)} \\
&= \Pr\left\{ \text{\#bit-flips bright + \#bit-flips dark} > \frac{d-1}{2} \right\} \\
&= \sum_{j=0}^{\frac{d-1}{2}} \Pr\{\text{\#bit-flips bright} = j\} \Pr\left\{ \text{\#bit-flips dark} > \frac{d-1}{2} - j \right\} \\
&= \sum_{j=0}^{\frac{d-1}{2}} \sum_{k=\frac{d-1}{2}-j}^{\mathbf{1}_c} \binom{\mathbf{0}_c}{j} \binom{\mathbf{1}_c}{k} \mathrm{P}_{\text{flip, dark}}^{j} (1 - \mathrm{P}_{\text{flip, dark}})^{\mathbf{0}_c - j} \\
&\quad \mathrm{P}_{\text{flip, bright}}^{k} (1 - \mathrm{P}_{\text{flip, bright}})^{\mathbf{1}_c - k} && \text{(S4)}
\end{aligned}$$

Whereas, if we use a minimum distance decoder,

$$\Pr\{\mathbf{y} \neq \mathbf{m}\} = \Pr\left\{ \min_{\mathbf{z} \in \mathcal{C}} d_H(x, z) \neq c \right\} \quad \text{(S5)}$$

$$\text{, where } \mathcal{C} = \{\, \mathbf{c} \mid \mathbf{c} = \mathcal{E}_{\text{BCH}(n,k,d)}(\mathbf{m}) \,\forall\, \mathbf{m} \in \mathcal{M} \,\}$$

Evaluating this probability expression in closed-form is not straightforward, but can be done by enumerating each codeword in an exhaustive manner. For this reason, we opt to use Monte-Carlo simulations when comparing BCH and repetition strategies.

1.5. Binary Shifting

Decoding binary shifted patterns is achieved by using a matched filter, where the template corresponds to a pattern with a burst of $2^{L_{\text{shift}}}$ ones followed by $2^{L_{\text{shift}}}$ zeros. In this section, we present the error analysis for binary shifting with $L_{\text{shift}} = 3$, corresponding to a temporal pattern length of $2^{L_{\text{shift}}+1} = 16$ frames. Without loss of generality, we consider the temporal sequence representing pixel location 0—comprising of 8 ones followed by 8 zeros. The expected error of other circularly shifted sequences are identical to this. Let s_t denote the value of the received signal (possibly corrupted) at the t-th time instant ($0 \leq t \leq 15$). Then, the probability of the absolute decoding error being exactly 1 pixel can be derived as

$$\begin{aligned}
\Pr\{|\text{error}| = 1\} &= \Pr\left\{ \{\sum_{t=1}^{8} s_t \geq \sum_{t=j}^{7+j} s_t \;\forall\; j\} \bigcup \{\sum_{t=-1}^{6} s_t \geq \sum_{i=j}^{7+j} s_i \;\forall\; j\} \right\} \\
&\leq 2\Pr\left\{ \{\sum_{t=1}^{8} s_t \geq \sum_{t=0}^{7} s_t\} \right\} && \text{(Union bound, equally likely events)} \\
&= 2\Pr\{s_8 \geq s_0\} \\
&= 2\Pr\{X \geq Y\} && (X \sim \text{Ber}(\mathrm{P}_{\text{flip, dark}}),\ Y \sim \text{Ber}(1 - \mathrm{P}_{\text{flip, bright}})) \quad \text{(S6)} \\
&= 2\left(\mathrm{P}_{\text{flip, dark}} + (1 - \mathrm{P}_{\text{flip, dark}})\mathrm{P}_{\text{flip, bright}}\right) && \text{(S7)}
\end{aligned}$$

where, $\text{Ber}(p)$ denotes the Bernoulli probability distribution. In the above expression, we consider time instances modulo 16, i.e., $t = -1$ is interpreted as $t = 15$. Next, the probability of the absolute decoding error being exactly 2 pixels can computed as

$$
\begin{aligned}
\Pr\{|\text{error}| = 2\} &= \Pr\left\{\{\sum_{t=2}^{9} s_t \geq \sum_{t=j}^{7+j} s_t \;\forall\; j\} \bigcup \{\sum_{t=-2}^{5} s_t \geq \sum_{t=j}^{7+j} s_t \;\forall\; j\}\right\} \\
&\leq 2\Pr\left\{\{\sum_{t=2}^{9} s_t \geq \sum_{t=0}^{7} s_t\}\right\} && \text{(Union bound)} \\
&= 2\Pr\{s_8 + s_9 \geq s_0 + s_1\} && \text{(S8)} \\
&= 2\Pr\{X \geq Y\} && (X \sim \text{Bin}(2, \text{P}_{\text{flip, dark}}),\ Y \sim \text{Bin}(2, 1 - \text{P}_{\text{flip, bright}}))
\end{aligned}
$$

where, $\text{Bin}(n, p)$ denotes the binomial probability distribution. Similarly, we can upper bound the probability of the absolute decoding error being $1 \leq r \leq 15$ pixels as $2\Pr\{X_r \geq Y_r\}$, where $X_r \sim \text{Bin}(r, \text{P}_{\text{flip, dark}})$, $Y_r \sim \text{Bin}(r, 1 - \text{P}_{\text{flip, bright}})$. With this, the expected absolute decoding error is given by:

$$
\begin{aligned}
\mathbb{E}[|\text{error}|] &= \sum_{r=0}^{15} r\Pr\{|\text{error}| = r\} \\
&\leq \sum_{r=1}^{15} 2r\Pr\{X_r \geq Y_r\} && \text{(S9)}
\end{aligned}
$$

Using this expression, the upper bound on the expected decoding error for the dark room condition (using flux values from Fig. 3(a)) is 1.2 pixels.

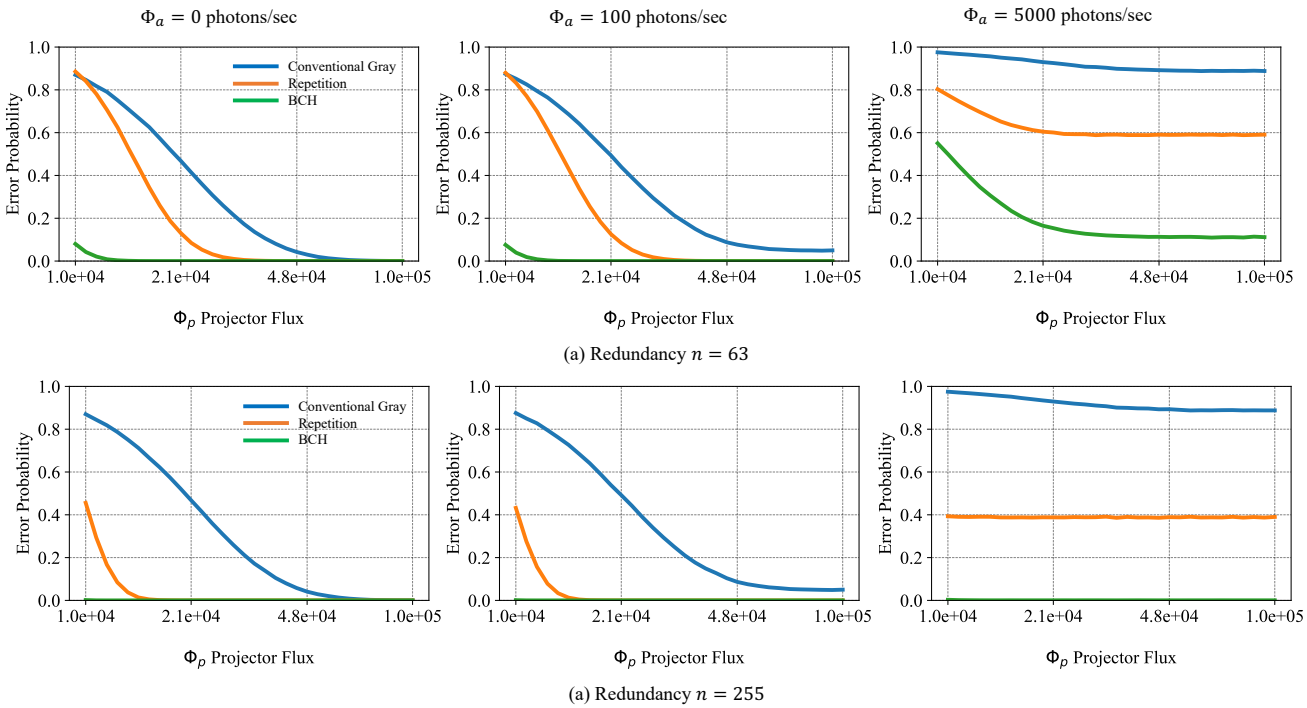
1.6. Monte-Carlo Simulation Procedure

Algorithm 1 Monte-Carlo Simulation Procedure

Require: Bit-flips probabilities $\mathrm{P}_{\text{flip, bright}}, \mathrm{P}_{\text{flip, dark}}$
Message set $\mathcal{M}$ with corresponding codeword set $\mathcal{C}$
error metric $\mathcal{L}$
Monte-Carlo iterations n_{iter}
procedure MONTE-CARLO-SIMULATION($\mathrm{P}_{\text{flip, bright}}$, $\mathrm{P}_{\text{flip, dark}}$, $\mathcal{M}$, $\mathcal{C}$, $\mathcal{L}$ n_{iter})
  error $\leftarrow 0$
  for $\mathbf{m} \in \mathcal{M}$ **do**
    Pick corresponding $\mathbf{c} \in \mathcal{M}$
    $\text{error}_{\mathbf{m}} \leftarrow 0$
    for $1 \leq i \leq n_{\text{iter}}$ **do**
      Randomly flip $\mathbf{1_c}$ and $\mathbf{0_c}$ with probability $\mathrm{P}_{\text{flip, bright}}$ and $\mathrm{P}_{\text{flip, dark}}$ respectively to obtain $\mathbf{x}$
      Decode $\mathbf{x}$ to obtain $\mathbf{y}$
      Compute $\mathcal{L}(\mathbf{m}, \mathbf{y})$
      $\text{error}_{\mathbf{m}} = \text{error}_{\mathbf{m}} + \mathcal{L}(\mathbf{m}, \mathbf{y})$
    end for
    $\text{error}_{\mathbf{m}} = \text{error}_{\mathbf{m}} / n_{\text{iter}}$
    $\text{error} = \text{error} + \text{error}_{\mathbf{m}}$
  end for
  $\text{error} = \text{error} / |\mathcal{M}|$
return error
end procedure

We describe the simulation procedure in Algorithm 1. The error metrics we consider are exact error ($\sum_{i=1}^{N} \frac{1}{N} \mathbb{I}(x_i, y_i)$, where $\mathbb{I}$ denotes the indicator function) and root mean square error (RMSE, $\sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2}$). Figure 6 and Figure 8 both use $n_{\text{iter}} = 100$ (Monte-Carlo iterations).

1.7. Empirical Comparison of BCH and Repetition Strategies at Various Φ_a



(a) Redundancy $n = 63$

(a) Redundancy $n = 255$

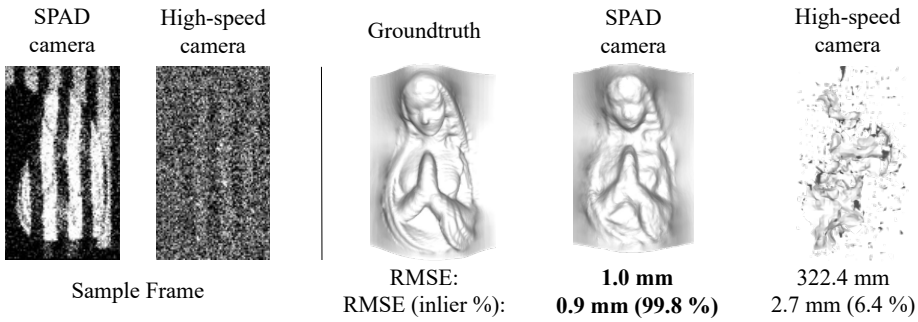
Suppl. Fig. 1. **Monte-Carlo evaluation of BCH and repetition strategies,** plotted at various ambient flux Φ_a. $\Phi_a = 0$ photons/sec (*left column*) represents a dark room, while $\Phi_a = 100$ photons/sec and $\Phi_a = 5000$ photons/sec *(middle and right columns)* represent indoor illumination and bright intensity matching the projector's flux respectively. Across ambient illumination levels and redundancy factors, BCH codes outperform repetition codes. Particularly at $n = 255$, BCH codes result in almost zero error everywhere.

Supplementary Note 2. Comparison to High-Speed Cameras

Read-noise of existing high-speed cameras is quite large and worsens with increasing read-out rates [2]. For instance, the Phantom v2640 [1] has a read noise of 18.8e- when capturing 640×480 frames at 28 kHz. This leads to an order of magnitude lower SNR than SPADs (Tab. 1), and hence extremely noisy, practically unusable, reconstructions (Fig. 2). This is because high-speed cameras must capture sufficient photons in each frame to beat the high read noise floor, which is not possible especially in high-speed motion scenarios considered in this paper.

Device	Dark room	Indoor lamp	Spot lamp
Phantom-v2640	0.07	0.08	0.13
SwissSPAD	0.81	0.84	1.10

Table 1. **SNR of a high-speed camera (Phantom-v2640) vs a SPAD camera (this work),** when observing a single bright pattern at 20kHz across various ambient conditions. No new experiment was conducted—incident flux data is obtained from Fig 3.

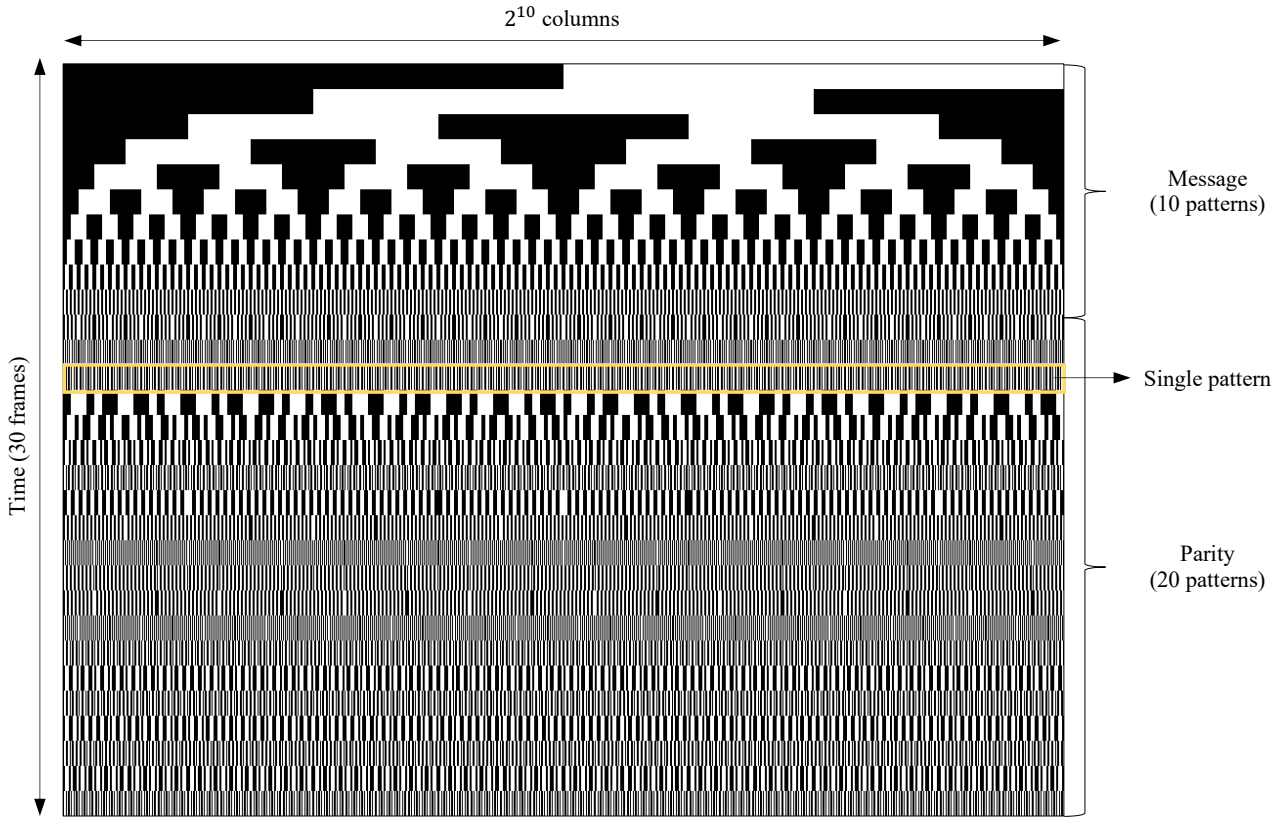


Suppl. Fig. 2. **3D reconstruction from the Phantom-v2640 vs the SPAD camera.** *(left)* We simulate high-speed capture using photon arrival rates. *(right)* Read-noise dominated frames of the high-speed camera lead to poor reconstructions. For a fair comparison, we use Hybrid-255 and Hybrid-127 with complementary frames as the coding scheme for the SPAD camera and the high-speed camera respectively—requiring similar acquisition times.

Supplementary Note 3. Code Look-Up-Tables

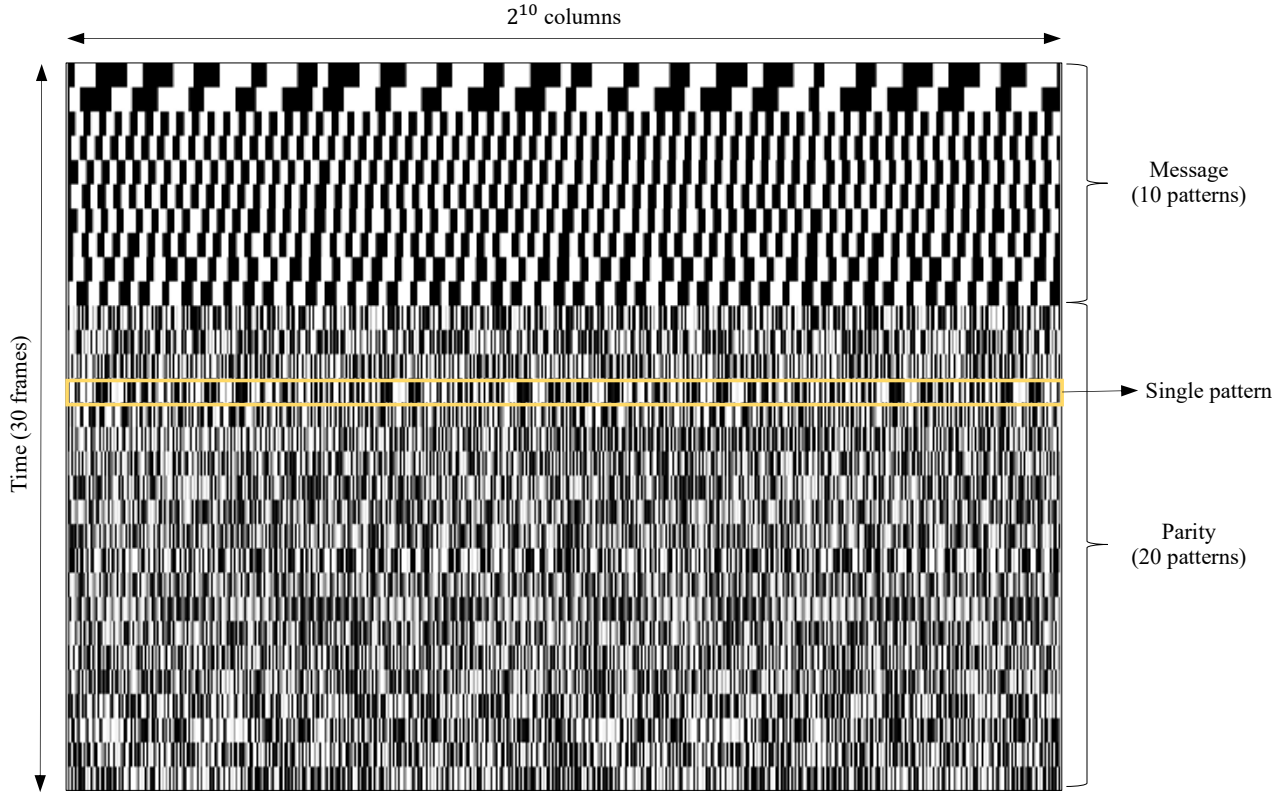
In this note, we describe the coding strategies developed such as BCH and Hybrid encoding, by means of their code look-up-tables (LUTs)—which depict the projected patterns row-wise across time.

3.1. BCH(31, 11, 11) Encoding of a 10-bit Conventional Gray Message



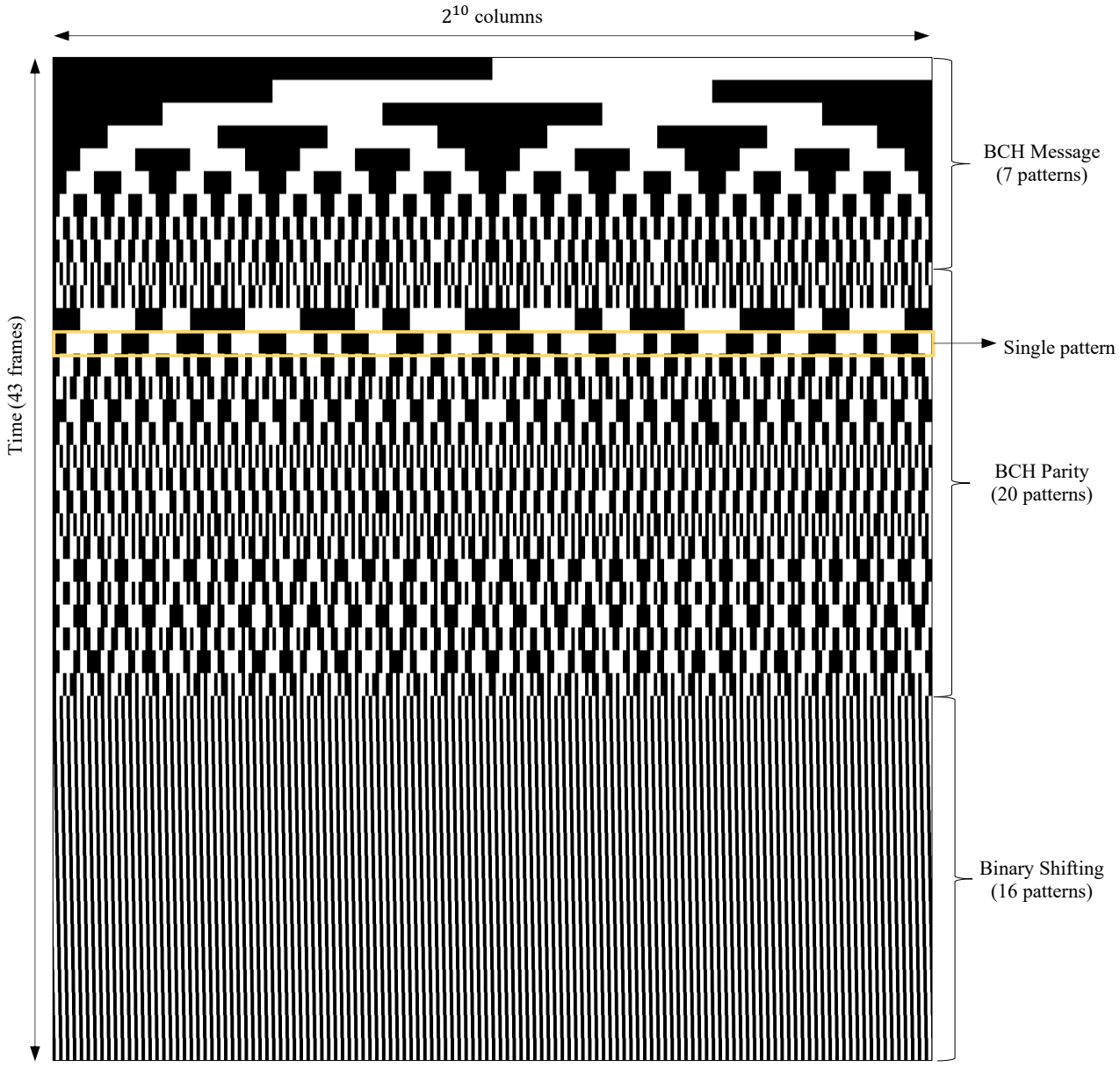
Suppl. Fig. 3. **Code Look-Up-Table describing the BCH(31, 11, 11) encoding of a 10-bit Conventional Gray message.** Following systematic encoding, the code LUT upto the first 10 frames is identical to a Gray Code LUT. Most of the parity frames feature high spatial-frequency, which is easily distorted by short-range effects in a SL system (Sec. 4.2). Zoom-in to see high-frequency details.

3.2. BCH(31, 11, 11) Encoding of a 10-bit Long-run Gray Message



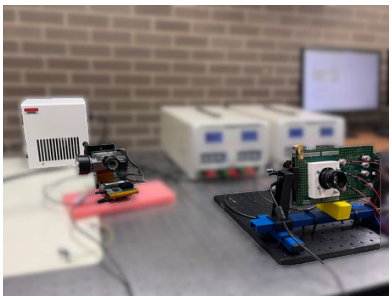
Suppl. Fig. 4. **Code Look-Up-Table describing the BCH(31, 11, 11) encoding of a 10-bit Long-run Gray message.** Owing to the maximal minimum stripe-widths of Long-run Gray codes, the first 10 frames of the code LUT are robust to short-range distortions such as projection defocus and/or resolution mismatch. However, the parity frames continue to comprise of high-spatial frequency patterns and are thus, easily distorted. Zoom-in to see high-frequency details.

3.3. Hybrid $(n = 31)$ Encoding of a 10-bit Gray Message

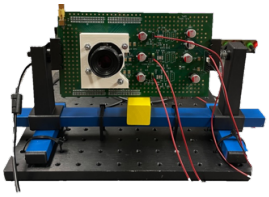


Suppl. Fig. 5. **Code Look-Up-Table describing the Hybrid$(n = 31)$ encoding of a 10-bit Conventional Gray message.** We encode $L_{\text{BCH}} = 7$ MSBs using the BCH(31, 11, 11) encoder and $L_{\text{shift}} = 3$ LSBs using binary shifting. Unlike the code LUTs of BCH encoded patterns (Fig. 3 and Fig. 4 which comprise of many high spatial-frequency patterns (typically featuring a minimum stripe-width of 1 or 2 pixels), the minimum stripe-width of these Hybrid $(n = 31)$ codes are at least 8 pixels by construction. This results in Hybrid codes offering robustness to both bit-flips arising from photon noise and other short-range effects arising from a combination of projector defocus and camera-projector resolution mismatch.

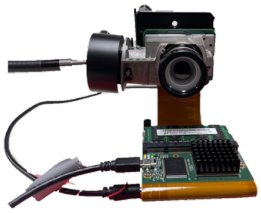
Supplementary Note 4. Single-Photon SL Hardware Prototype



(a) Single-Photon SL Prototype



(b) SwissSPAD2 array



(c) Vialux-7000 high-speed module

(d) SugarCUBE LED source

Suppl. Fig. 6. **Single-Photon SL hardware prototype.** The setup consists of a SwissSPAD2 array [12] and a Vialux-7000 development kit which is based on the Texas Instruments DLP6500 DMD. For an illumination source we use a SugarCUBE Ultra, a broadband white LED, featuring a light output of 4000 lumens.

Supp. Fig. 6 shows the various components of our Single-Photon SL prototype. The resolution of the SPAD array is 512×256 pixels, while the DLP projector has a resolution of 1024×768 pixels. We used a baseline of 14 cm between the camera and projector—with the intention of maximizing the field of view of the camera occupied by the objects of interest.

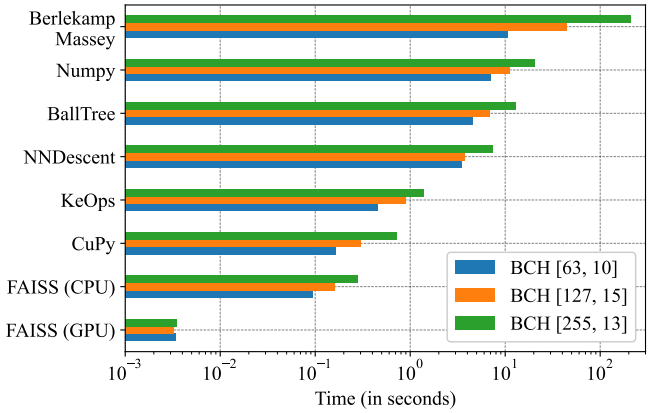
4.1. Calibration Procedure

Our calibration procedure is similar to Zhang et al. [13], where the projector is treated as an inverse camera. Specifically, we capture correspondence maps of a chessboard, repeated across 30 different positions. We then perform a calibration step mimicking a stereo setup based on the detected corners of the chessboard and their row and column correspondences.

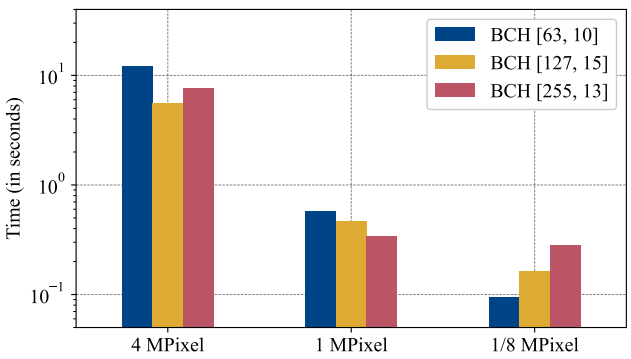
To obtain correspondences, we use projected patterns based on Gray codes and binary shifting—similar to our proposed Hybrid strategy, but with no added redundancy—to encode both column and row indices. This provides us precise row and column correspondence while simultaneously dealing with short-range effects arising from projector defocus / resolution-mismatch. To suppress the effects of photon-noise, we capture 5120 SPAD frames for each projected pattern—which are subsequently averaged and compared against acquired complementary frames to produce binary outputs.

Supplementary Note 5. Benchmarking Minimum Distance Decoding Methods

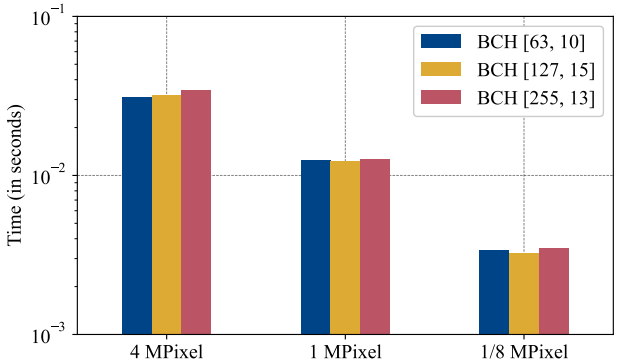
Supp. Fig. 7 shows that Minimum Distance Decoding (MDD) can be significantly faster than algebraic decoding in Single-Photon SL. Supp. Fig. 8 further shows that the most performant MDD method—FAISS using a GPU runtime—also scales reasonably to larger sized arrays including 1MPixel and 4MPixel sensors.



Suppl. Fig. 7. **Benchmark of various Minimum Distance Decoding (MDD) methods.** We query a 512×256 array of codewords each encoding a 10-bit message. We compare an algebraic method (Berkelamp-Massey [9]) to several MDD methods including those using exhaustive search (Numpy [7], CuPy [10]), graph-based search (BallTree [3], NNDescent [5]) symbolic-tensors (KeOps [4]) and batched queries (FAISS [8]). Benchmarks reported across various BCH encoders using a Intel-i7 8700K CPU and a NVIDIA 1080Ti GPU.



(a) FAISS CPU



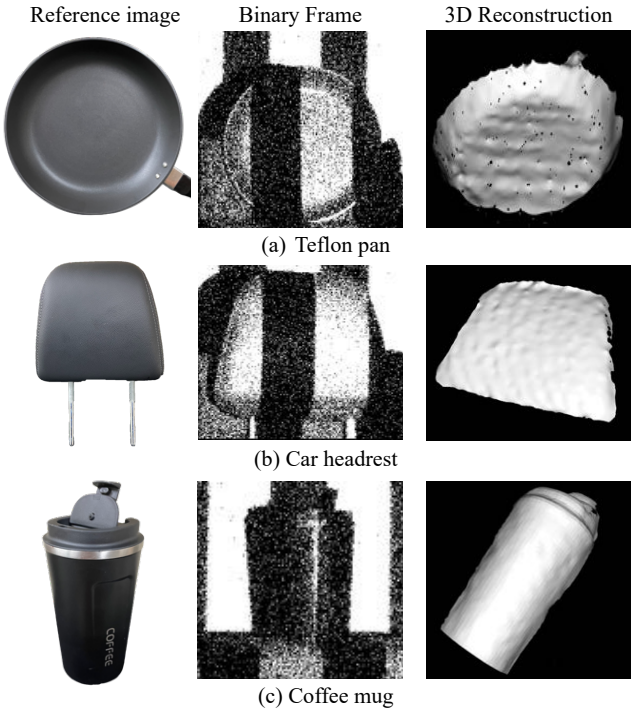
(a) FAISS GPU

Suppl. Fig. 8. **MDD methods scale to larger resolution arrays.** Our implementation based on FAISS requires about 4s and 15s on a CPU (Intel-i7); 12 ms and 30 ms for one and four megapixels respectively on a GPU device (GTX 1080Ti).

Supplementary Note 6. More Results

6.1. Low-albedo Objects

Supp. Fig. 9 includes additional results on low-albedo objects; demonstrating the capabilities of Single-Photon SL in 3D imaging commonplace dark objects—such as the teflon pan, coffee mug and car headrest.



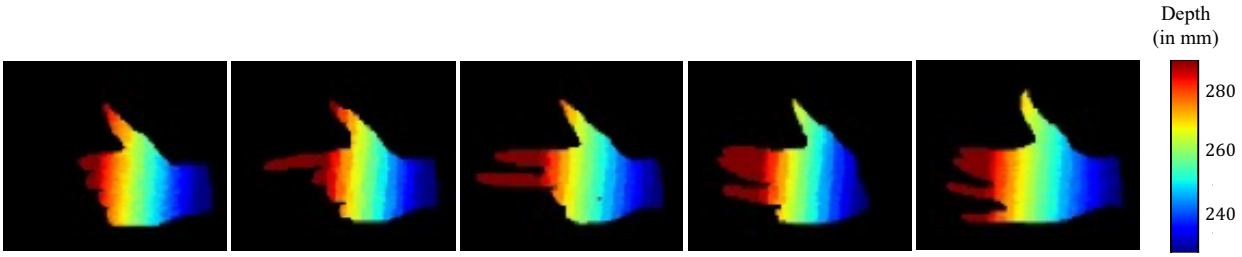
Suppl. Fig. 9. **More 3D reconstructions of low-albedo scenes,** using Hybrid $(n = 255)$ at 40 FPS.

6.2. Dynamic Motion Sequences

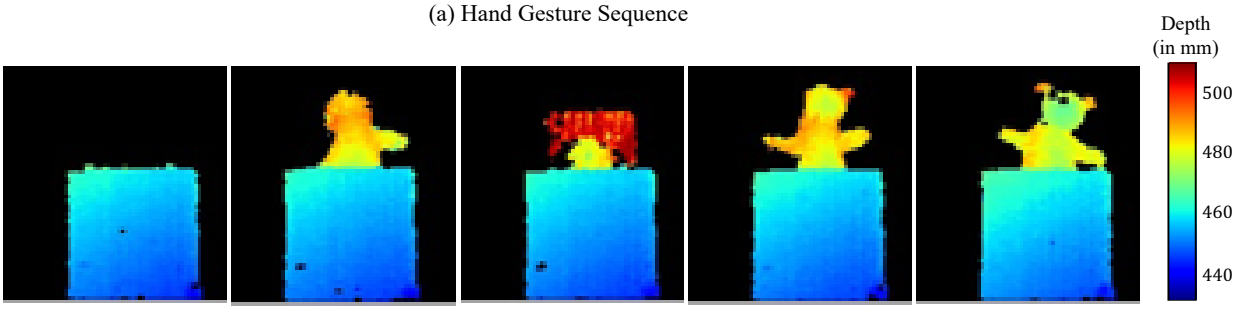
We include further results on dynamic scenes in Supp. Fig. 10. The curtain took an estimated 70 milliseconds to fall, while the Jack-in-the-box toy took around 40 milliseconds to spring up.

Going faster with pipelining. While we have decoded disjoint bursts of acquired frames to output depth-maps in Fig. 11 and Fig. 13, we can also use overlapping sequences. Assuming N patterns are projected, we can recover depth across time by decoding sequences $\{I_i\}_{i=1}^{N}, \{I_i\}_{i=N+1}^{2N}, \{I_i\}_{i=2N+1}^{3N}, ...$, or by considering sequences $\{I_i\}_{i=1}^{N}, \{I_i\}_{i=s+1}^{N+s}, \{I_i\}_{i=s+2}^{N+s+1}, ...$ i.e., using a sliding window of N patterns with stride $0 < s < N$. This strided or pipelined approach contains more temporal information than disjoint bursts. We utilize pipelining based frame-interpolation for the jack-in-the-box sequence to produce a high-speed 3D reconstruction video at 2000 FPS.

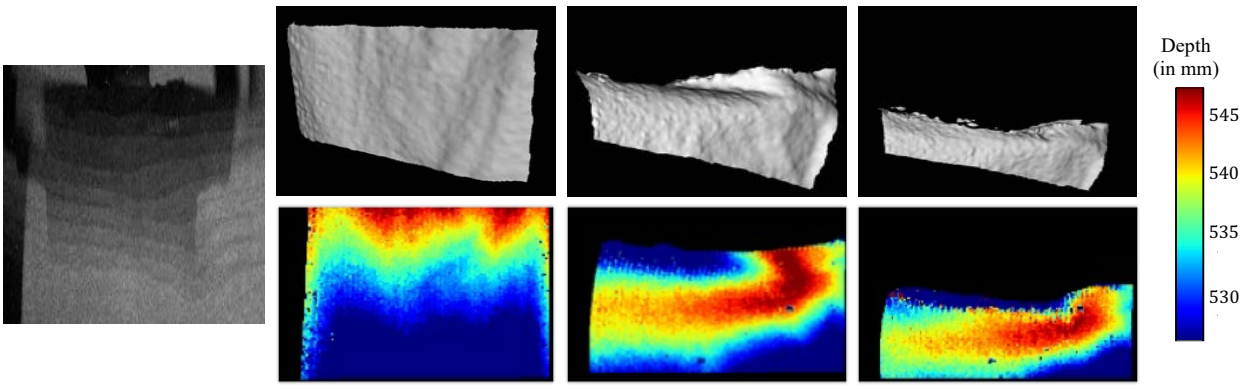
High-speed depth videos for the deforming cloth sequence and the Jack-in-the-box toy can be found on the project webpage. By using pipelining-based decoding, we output videos at 1000 FPS for the waving cloth and 2000 FPS for the Jack-in-the-box with a stride of 9 and 20 patterns respectively. We captured the Jack-in-the-box by placing it horizontally to maximise its occupied field-of-view. Both videos are also included in the teaser video which provides an overview of our paper.



(a) Hand Gesture Sequence



(b) Jack-in-the-box Sequence



(c) Falling Curtain Sequence

Suppl. Fig. 10. **Additional dynamic scenes imaged by Single-Photon SL.** The hand gesture sequence was captured using Hybrid ($n = 255$) at 75 FPS, while the Jack-in-the-box and falling curtain sequences were both acquired using Hybrid ($n = 31$) at 450 FPS.

References

[1] Phantom-v2640. phantomhighspeed.com/products/cameras/ultrahighspeed/v2640/. 7

[2] Assim Boukhayma, Arnaud Peizerat, and Christian Enz. A sub-0.5 electron read noise vga image sensor in a standard cmos process. *IEEE Journal of Solid-State Circuits*, 2016. 7

[3] Leonid Boytsov and Bilegsaikhan Naidan. Engineering efficient and effective non-metric space library. In *International Conference on Similarity Search and Applications*, 2013. 12

[4] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021. 12

[5] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the International Conference on World wide web*, 2011. 12

[6] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. *Draft available at http://www. cse. buffalo. edu/ atri/courses/coding-theory/book*, 2012. 2

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 12

[8] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019. 12

[9] J. Massey. Shift-register synthesis and bch decoding. *IEEE Transactions on Information Theory*, 15(1):122–127, 1969. 2, 12

[10] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman. Cupy : A numpy-compatible library for nvidia gpu calculations. 2017. 12

[11] Ron M Roth. Introduction to coding theory. *IET Communications*, 47:18–19, 2006. 2

[12] Arin Can Ulku, Claudio Bruschini, Ivan Michel Antolovic, Yung Kuo, Rinat Ankri, Shimon Weiss, Xavier Michalet, and Edoardo Charbon. A 512×512 SPAD Image Sensor With Integrated Gating for Widefield FLIM. *IEEE Journal of Selected Topics in Quantum Electronics*, 25(1):1–12, Jan. 2019. 11

[13] Song Zhang and Peisen S Huang. Novel method for structured light system calibration. *Optical Engineering*, 45(8):083601, 2006. 11