Auto-Tuning Structured Light by Optical Stochastic Gradient Descent: Supplemental Document

Wenzheng Chen[1,2*] Parsa Mirdehghan[1*] Sanja Fidler[1,2,3] Kiriakos N. Kutulakos[1]
University of Toronto[1] Vector Institute[2] NVIDIA[3]
Toronto, Canada
{wenzheng,parsa,fidler,kyros}@cs.toronto.edu

A. Additional Details on Optical SGD

In this section, we provide more details on implementing Optical Stochastic Gradient Descent.

A.1. Illustration of Optical Differentiation Algorithm

To compute the gradients of error vector $\mathsf{err}(\mathbf{d},\mathbf{g})$ with respect to the control vector $\mathbf{c}$ (Eq. (8)), we must estimate the Jacobian optically. Eq. (9) of the paper and its following optical-domain procedure provides a method for capturing the Jacobian by the use of directional derivatives $\mathrm{D}_{\mathbf{a}}\mathsf{img}(\mathbf{c},\mathcal{S})$ along an adjustment vector $\mathbf{a}$. Figure A1 (third column) illustrates one sample adjustment vector and its corresponding image difference, where the adjustment vector has N/B equally-spaced non-zero elements (Section 4.1). We empirically set $B = 7$ and fix the adjustment magnitude h to 0.15. This adjustment magnitude provides enough change in the image while being small enough to approximate the directional derivatives.

[*] Authors contributed equally

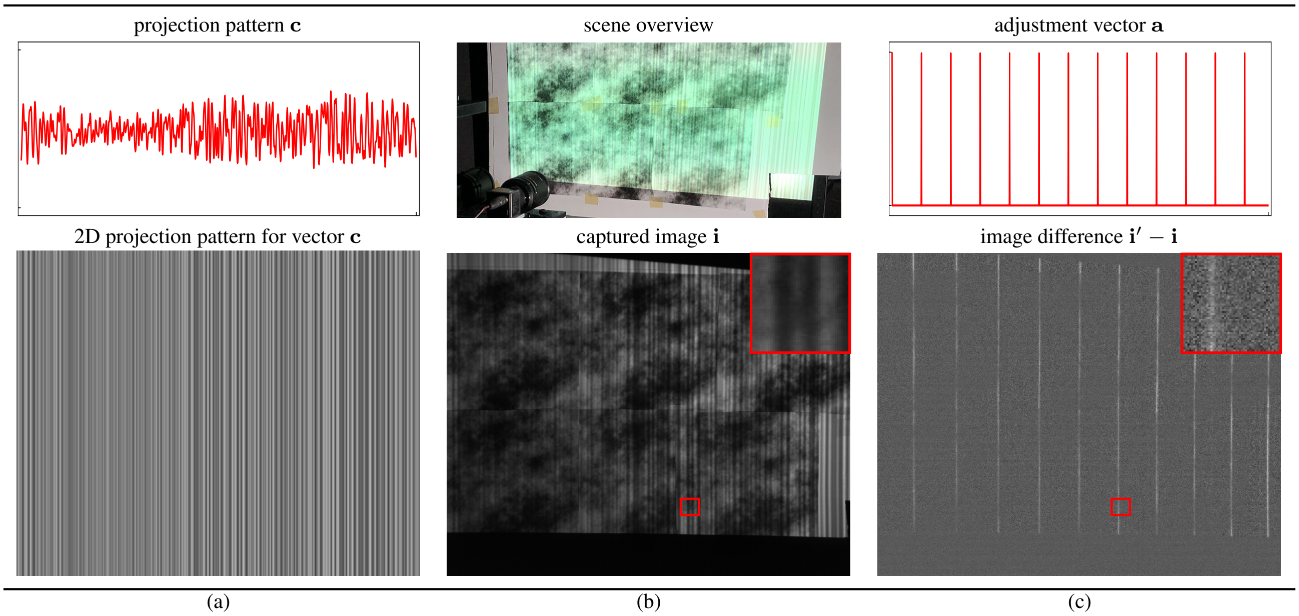

projection pattern $\mathbf{c}$
scene overview
adjustment vector $\mathbf{a}$
2D projection pattern for vector $\mathbf{c}$
captured image $\mathbf{i}$
image difference $\mathbf{i}' - \mathbf{i}$
(a)
(b)
(c)

Figure A1: **Optical-domain differentiation on an actual projector-camera system.** **(a)** 1D plot of a sample projection pattern $\mathbf{c}$ and its corresponding 2D projection pattern. **(b)** a photo of the experimental setup and the corresponding camera image of the scene under the projection pattern depicted in (a). **(c)** adjustment vector and its corresponding image difference.

A.2. Ground Truth Acquisition

For both optimization and evaluating the experimental results, capturing an accurate ground truth is required. We acquire the ground truth with the following routine:

1. Project 30 a la carte (0-tolerance) [1] patterns onto the scene, and use the ZNCC decoder to compute a pixel-to-column correspondence map.
2. Repeat the previous step 10 times to capture 10 maps.
3. Fuse the captured maps into a single map $\mathbf{g}_1$ by majority vote.

To verify the captured ground truth, we compare it with another map acquired with a different family of patterns:

1. Project 160 shifted cosine patterns (10 shifts for frequencies in the range of [16-31]) along with 11 shifted cosine patterns of frequency $\frac{N}{11}$, where N is the number of projector columns.
2. Compute a sub-pixel map [2].
3. Round the fractional correspondences to the nearest integer, to obtain an integer pixel-to-column correspondence map $\mathbf{g}_2$.

We say that the pixel-to-column correspondence map $\mathbf{g}_1$ captured by the first routine is an *ϵ-tolerance ground truth* if no pixel in $\mathbf{g}_1$ and $\mathbf{g}_2$ differs by more than ϵ; *i.e.* for each image pixel, the column correspondences captured by these two routines are at most ϵ columns apart from each other.

In our experiments, we were able to acquire 0-tolerance ground truth for the training board and our test scenes with all the imaging systems used in this work except the indirect light experiment in Figure 11. For that experiment only 1-tolerance ground truth was possible.

Capturing ground truth during optical SGD. Although an accurate ground truth is required for evaluation of different systems and for comparison to prior art, we do not need the most accurate ground truth while auto-tuning a system. This is because small errors can introduce an additional source of stochasticity, and helps the optimization not to overfit. Therefore, the optical SGD (outlined in Figure 3) uses just one set of 30 a la carte patterns.

A.3. Frequency Constraint

We always enforce a weak max-frequency constraint to pattern optimization because we found that it leads to a more stable optimization. When auto-tuning on board, in particular, we band-limit patterns to frequency $2^{\lfloor \log_2 N/4 \rfloor}$, which is the closest power of 2 to half the Nyquist rate. This limit is fixed in all our experiments, with the exception of auto-tuning for indirect light (see Figure 11 and section F). Enforcing the max frequency constraint is done in the Fourier domain; at the end of each optical SGD iteration, we set Fourier coefficients higher than the specified max frequency to zero.

A.4. Optical SGD for Low-SNR Scenes

Auto-tuning a system for low-SNR conditions (*e.g.* low projector brightness, distant scene and/or high ambient light) introduces a major challenge: the image difference $\mathbf{i}' - \mathbf{i}$ due to adjustment (Figure A1(c)) may be below the image quantization level. This will lead to incorrect gradient estimates that can cause optical SGD to fail. To cope with this challenge, we do not auto-tune systems under these conditions. Instead we always capture images under conditions where the image difference $\mathbf{i}' - \mathbf{i}$ can be computed reliably, and then corrupt these images by (1) down-scaling them and adding zero-mean Gaussian noise to simulate reduced signal level, and (2) adding Poisson noise to simulate high ambient-light contributions. These unquantized images are then used to compute the gradients required by optical SGD.

Intuitively, this approach leads to images with lower quantization error, so that very small image gradients can still be estimated. We use this method only for auto-tuning systems for far-field imaging (Figure 10, and section E.3). In all other cases, auto-tuning is performed without any image corruption.

A.5. Optical SGD for Scenes with Indirect Light

Auto-tuning a system for settings with significant indirect light requires specific considerations as well.

Capturing the ground truth. It is well-known that indirect light can corrupt shape estimates obtained by structured light [2, 3]. In such cases, simply increasing the number of structured light patterns does not guarantee that correct correspondences will be acquired. To mitigate the influence of indirect light as much as possible, we used the same method as described in subsection A.2 to obtain ground truth, but captured the images by operating EpiScan3D in epipolar-only imaging mode [4]. Note that this mode was employed only for acquiring the ground truth.

Optimization parameters. Indirect light leads to a less-sparse Jacobian. Consequently, we use larger spacing between non-zero elements of the adjustment vector. Furthermore, we observed that using a lower max-frequency for patterns yields a more stable optimization. We empirically set F to $\frac{1}{8}$ of Nyquist limit, and chose $B = 23$ for translucent training scene (refer to section F).

Hadamard multiplexing for image Jacobian acquisition. Indirect light reduces the SNR of direct surface reflections and therefore the SNR of optically-computed gradients. To improve the gradient estimation, we acquired the image Jacobian by Hadamard multiplexing [5] which is known to improve performance in low-SNR conditions. Specifically, instead of using adjustment vectors that have a single non-zero element, we use those defined by the S-matrix of appropriate size, and demultiplex the captured images after acquisition. This leads to higher-SNR gradients and improves the performance of optical SGD considerably (Figure 11 and section F).

A.6. Optical SGD for Color Structured Light Systems

To apply the optical auto-tuning framework to color structured light systems, we followed two approaches:

Approach 1: color structured light with demosaiced RGB images. First, we use demosaiced RGB images as the input to the algorithm [1]. For such a system running on K' color patterns, there are $K = 3K'$ control vectors, and $3K'$ corresponding images. With this modification, the same methods as Eq. (12-16), can be applied for both decoding and auto-tuning a system.

Approach 2: color structured light with raw single-channel images. As another option, we simply use raw single-channel Bayer image as the input. Operating on Bayer mosaic introduces a mismatch between the number of control vectors and the number of patterns: for K' RGB patterns, the number of captured images is K' but the number of control vectors is $K = 3K'$. This size mismatch between "image feature vector" and "projector feature vector" (as defined in section 4) prevents direct application of the ZNCC or ZNCC$_p$ decoders (Eq. (12-14)). Here we simply modify ZNCC-NN$_p$ for this purpose, by slightly changing the neural network architecture for image feature vector $\mathcal{F}()$. Specifically, we define $\mathcal{F}()$ to be a neural net with two fully connected layers of dimensions $(pK) \times (3pK) \times (3pK)$. Moreover, since each pixel in a 2×2 Bayer tile has a different spectral response and a different local neighbourhood, we use a distinct neural network for each pixel in a Bayer tile. This means the total number of learnable parameters for image features is $4(3p^2K^2 + 9p^2K^2)$. Eq. (16) then can be utilized to auto-tune the patterns and decoder.

B. Visualizations

Throughout the main paper and supplementary materials, we use several ways to visualize and analyze the results, and to showcase their differences. Here we provide more details on these visualizations and how to interpret them.

B.1. Disparity Maps

Given a pixel-to-column correspondence map $\mathbf{d}$, we define the *disparity* of a pixel (i, j) on the image plane to be $\mathbf{d}[i, j] - j$, where $\mathbf{d}[i, j]$ is its corresponding projector column. This definition is independent of intrinsic and extrinsic calibration of the projector-camera system and thus allows us visualize and evaluate performance in a calibration-invariant manner. Similiarly, *ground-truth disparity* of a pixel (i, j) is defined as $\mathbf{g}[i, j] - j$, where $\mathbf{g}[i, j]$ is its ground-truth corresponding projector column. It is straightforward to show disparity error at a pixel, is equal to its column-correspondence error $|\mathbf{d}[i, j] - \mathbf{g}[i, j]|$.

[1] We capture raw single-channel images with camera, and use standard OpenCV algorithm for demosaicing.

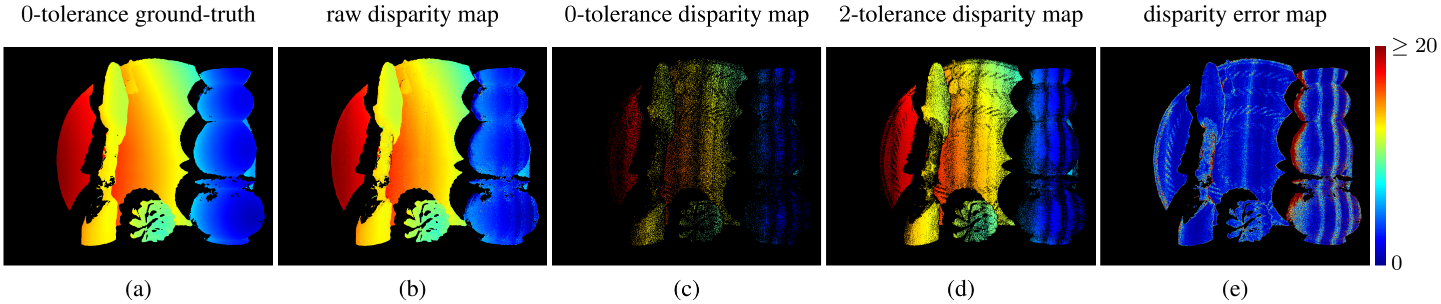


Figure B1: **Visualizing disparity maps.** Different methods of visualizing a sample disparity output. While for disparity maps (a-d), we use different color mapping depending on the scene, the color map for the disparity error map (e) is always fixed: dark blue corresponds to zero error and dark red shows error greater or equal than 20 pixels.

We use disparity map representation for all the results in the paper except Figure 1 (top row), where we calibrated the projector-smartphone pair to show metric depth instead.

To evaluate the results of different structured light systems, we use three distinct ways for visualizing the disparity maps:

Raw disparity maps. This visualization illustrates the disparities (with jet color map) for image pixels which have verified ground truth (Figure B1(b)) . We simply exclude the pixels for which ground truth cannot be acquired reliably. We set the color of those pixels to black.[2]

ϵ-tolerance disparity maps. The ϵ-tolerance disparity map excludes all the pixels whose disparity error is larger than ϵ. For example, Figure B1(c) shows all the pixels whose disparities were recovered perfectly (*i.e.* with zero error). This visualization is particularly useful for assessing results according to the ϵ-tolerance penalty: ϵ-tolerance disparity maps that contain more pixels signify a higher-quality reconstruction (*i.e.* higher number of pixels with disparity error less or equal than ϵ).

Disparity error maps. These maps show pixelwise disparity errors with a jet color map (Figure B1(e)). Since they show how errors are distributed over the whole image, we specifically use them to assess the system's performance according to the L_1 penalty.

B.2. Empirical Column-Confusion Matrices

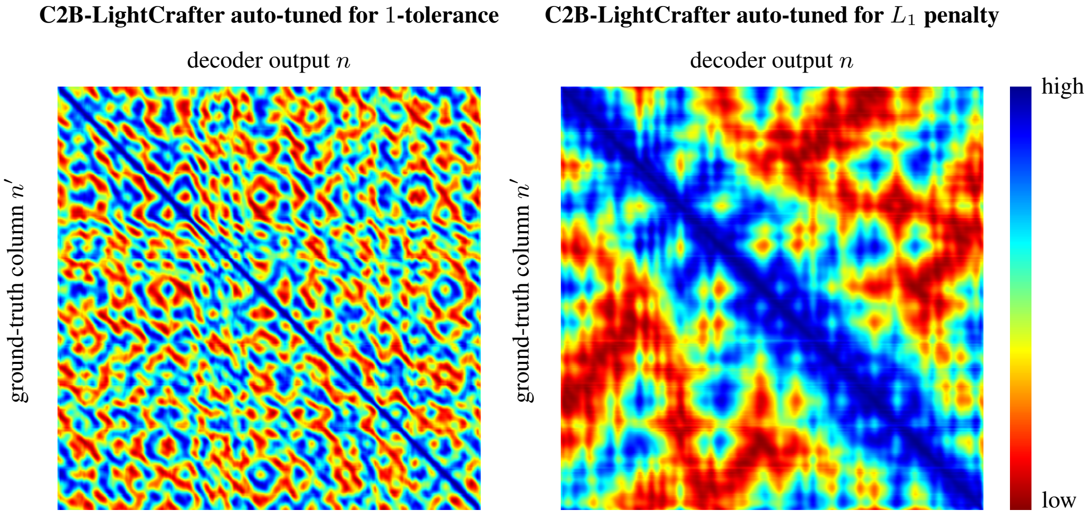
To assess how easily the decoder can confuse two projector columns, thereby leading to incorrect correspondences at a pixel, it is helpful to develop the notion of confusion. Here we consider an empirical measure of confusion that counts how many times the decoder outputs projector column n as the correspondence, when in fact the ground-truth correspondence was column n'. Given a particular pattern sequence and decoder, we compute these counts on the training board, once auto-tuning completes.

To visualize how easily two columns are confused, we visualize these counts as an $N \times N$ image, where pixel (n, n') indicates the frequency by which ground-truth column n' was decoded as column n (Figure B2 and supplementary video).

C. List of Devices

Table C1 lists all the devices we used in the paper and in the supplementary materials.

[2] Please refer to section A.2 for more details on acquiring the ground truth.



C2B-LightCrafter auto-tuned for 1-tolerance

C2B-LightCrafter auto-tuned for L_1 penalty

Figure B2: **Empirical column-confusion matrices.** Visualizing the column-confusion properties of the patterns and decoder in Figure 7 of the paper (top row, second and third column), obtained by auto-tuning the C2B-LightCrafter pair. Note how auto-tuning a system for L_1 penalty forces the easily-confused columns to lie near the diagonal, where disparity errors are low. In contrast, ϵ-tolerance created a drastically different confusion matrix that aims to maximize exact reconstruction.

projector	**projector response function**	**projector columns**	**pattern spatial frequency limit**	**camera**	**image exposure time**	**camera response function**
LG-PH550	non-linear	1280	256	IDS-UI324x	33msec	linear
LightCrafter	linear	400	64	C2B [6]	17msec	linear
Optoma 4K	non-linear	3840	512	Huawei P9	33msec	non-linear
LG-PH550	non-linear	1280	256	AVT-1920c	17msec	linear
LG-PH550	non-linear	1280	256	AVT-1920	17msec	linear
PicoPro	non-linear	1280	256	IDS-UI324x	17msec	linear

Table C1: **List of the experimental imaging systems.**

encoding scheme	**decoder**	**optimized for penalty**	**Reference**	**radiometrically calibrated projector?**	**auto-tuning applied?**
Hamiltonian	ZNCC	L_1	[7]	Yes	No
MPS	ZNCC	none	[2]	Yes	No
A la carte	ZNCC	0-tolerance	[1]	Yes	No
Hamiltonian	ZNCC$_5$	L_1	this paper	Yes	No
MPS	ZNCC$_5$	none	this paper	Yes	No
A la carte	ZNCC$_5$	0-tolerance	this paper	Yes	No
Hamiltonian	ZNCC-NN$_5$	0-tolerance & L_1	this paper	Yes	decoder only
MPS	ZNCC-NN$_5$	0-tolerance & L_1	this paper	Yes	decoder only
A la carte	ZNCC-NN$_5$	0-tolerance & L_1	this paper	Yes	decoder only
auto-tuned	ZNCC	0-tolerance & L_1	this paper	No	patterns only
auto-tuned	ZNCC$_5$	0-tolerance & L_1	this paper	No	patterns only
auto-tuned	ZNCC-NN$_5$	0-tolerance & L_1	this paper	No	patterns & decoder

Table D1: **List of the coding-decoding methods used for experiments.**

D. Quantitative Comparison with State-of-the-art Methods

Experimental Setup. We used the LG-IDS (Table C1) as our imaging system. The scene was placed approximately 1m from the system. We chose a scene with both high- and low-albedo surfaces, different geometries, texture, and depth discontinuities. An image of the scene under all-white projection pattern is shown both in Figures D1 and D2.
We used all the methods listed in Table D1 with $K = 4$ to compare their performance. Figures D1 and D2 show the comparisons for patterns auto-tuned with 0-tolerance and L_1 penalty, respectively. In Figure D1, we used percentage of pixels with no error as the evaluation metric, which is equivalent to 0-tolerance penalty. On the other hand, average disparity error is used for evaluation in Figure D2, which is the corresponding metric to L_1 penalty. In both figures, the improvements along the columns (marked with green) show the impact of our proposed decoding algorithms on existing encoding methods, while the last row showcases this work's contributions in both decoding and pattern optimization.

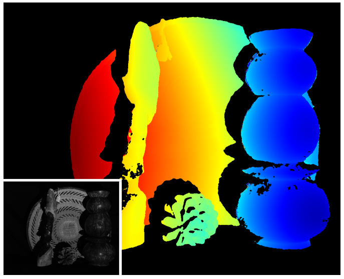
Neighbourhood decoding improves accuracy. Comparing the second columns of Figures D1 and D2 with their first columns indicates neighbourhood decoding has a drastic impact on the performance of system for all the patterns. For instance, it improves the percentage of zero-error pixels for MPS, Hamiltonian, and a la carte by 89%, 56%, and 86%, respectively. Same trend applies to average disparity error. Furthermore, the neural net decoder provides another level of improvement in the performance for auto-tuned patterns, but has almost no effect on patterns which are not optimized with auto-tuning. This suggests that the neural network decoder is most useful when it is optimized along with the patterns. It also justifies our default choice of ZNCC$_p$ decoder for state-of-the-art patterns used in Section 5.

The choice of error metric matters. A la carte patterns perform relatively well when evaluated by the percentage of zero-error pixels. However, their disparity maps result in high average disparity error. Hamiltonian patterns exhibit the opposite behavior: while they generate very smooth disparity maps with low average disparity error, they perform poorly when evaluated by the number of zero-error pixels. This suggests that optimality of a pattern, to a great extent, is related to the metric it will be evaluated by. Unlike other patterns that are blind to the choice of evaluation metric, our framework can tune the patterns for any desired metric by optimizing the equivalent error function.

% of pixels with no error

	ZNCC	ZNCC_5	ZNCC-NN_5
MPS	27.42	51.70	49.53
Hamiltonian	9.46	14.72	15.03
a la carte	33.14	61.78	61.76
auto-tuned	32.29	67.33	**72.24**

Ground Truth & Camera Image



visualization: 0-tolerance disparity maps overlaid with raw disparity

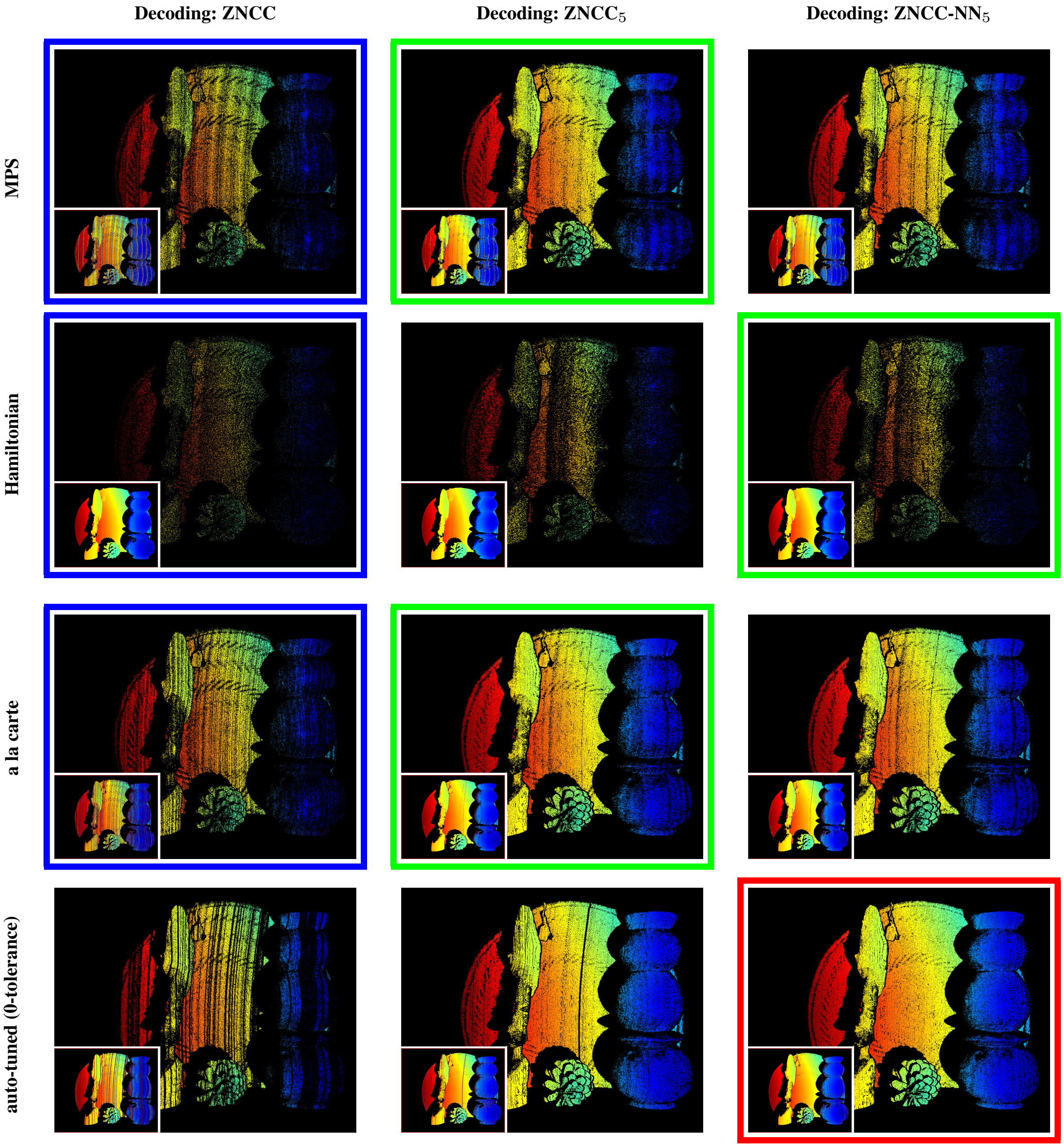
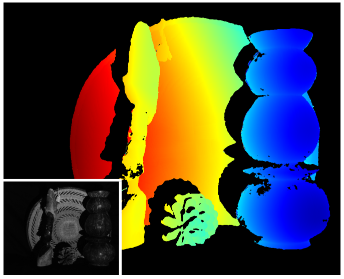


Figure D1: **Comparison of different coding-decoding methods based on percentage of zero-error pixels.** The table lists the no-error reconstruction rates (*i.e.* the percentage of pixels with zero error) of each pattern, decoded with all the decoding methods. Their corresponding 0-tolerance disparity maps are shown with the same layout as the table. The raw disparity maps are also shown as insets, for reference. Table entries and disparity maps marked as blue represent the current state of the art. Entries and disparity maps marked as green indicate the best-performing decoder for 3 previously-proposed pattern sequences (MPS, Hamiltonian, a la carte). Note that the best results of these patterns are obtained with decoders introduced in this paper. The best performance, shown in red, is obtained by auto-tuning with the ZNCC-NN$_5$ decoder.

average disparity error (pixels)

	ZNCC	ZNCC$_5$	ZNCC-NN$_5$
MPS	137.53	42.80	43.07
Hamiltonian	6.52	4.56	4.66
a la carte	173.67	39.84	41.30
auto-tuned	9.84	4.17	**3.65**

Ground Truth & Camera Image



visualization: disparity error maps overlaid with raw disparity

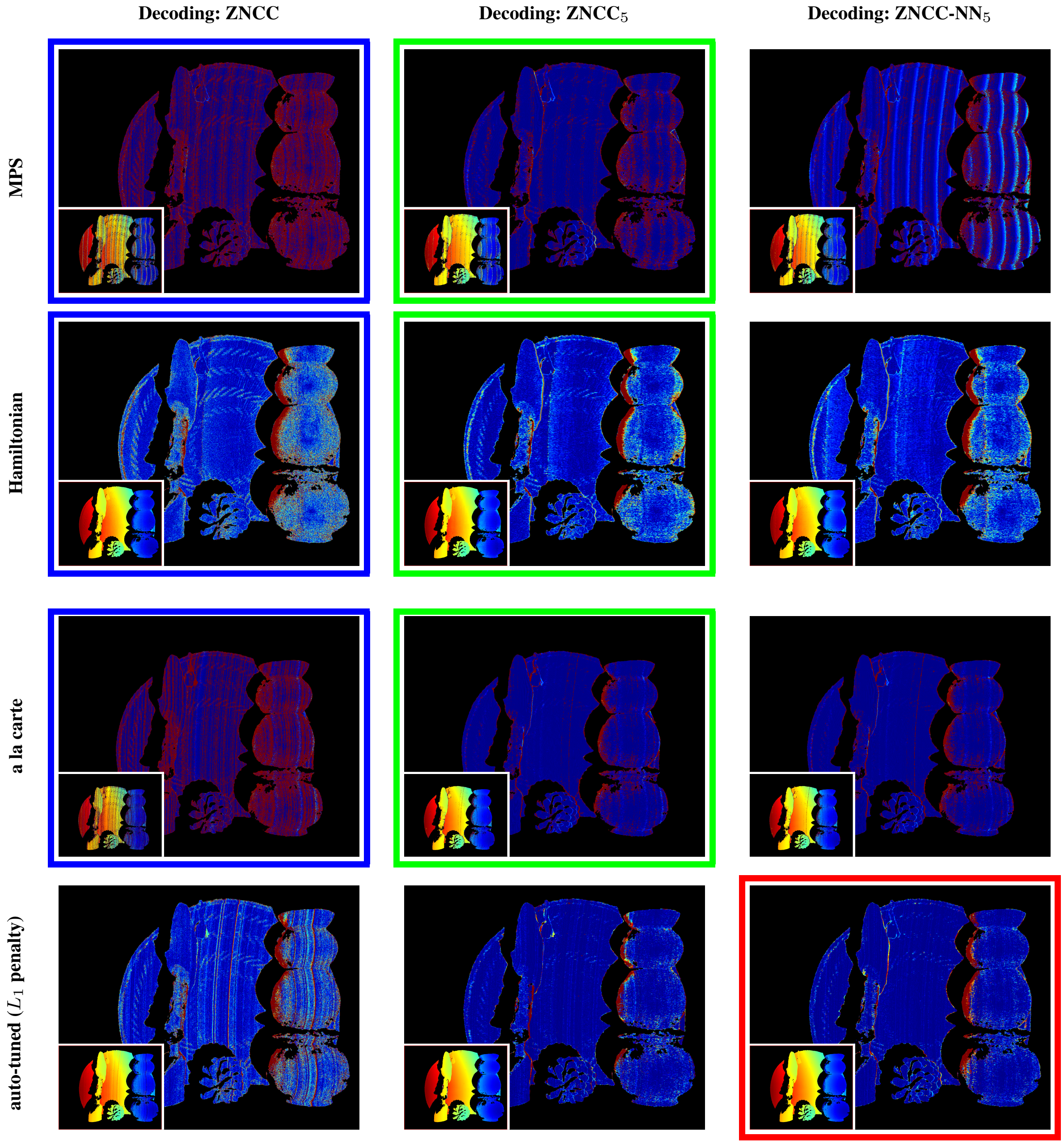


Figure D2: **Comparison of different coding-decoding methods based on average disparity error.** The table lists the average disparity error of each pattern, decoded with all the decoding methods. Their corresponding disparity error maps are shown with the same layout as the table. The raw disparity maps are also shown as insets, for reference. Table entries and disparity error maps marked as blue represent the current state of the art. Entries and disparity maps marked as green indicate the best-performing decoder for 3 previously-proposed pattern sequences (MPS, Hamiltonian, a la carte). Note that the best results of these patterns are obtained with decoders introduced in this paper. The best performance, shown in red, is obtained by auto-tuning with the ZNCC-NN$_5$ decoder.

E. Operating Range of an Auto-Tuned System

Experimental Setup. We used the LG-IDS system in Figure E1 to reconstruct the training board in geometrical arrangements that are different from those used for auto-tuning. We compare the performance of $K = 4$ sequences of MPS, Hamiltonian, a la carte patterns decoded with ZNCC$_5$ (which gives the best performance for the patterns), and patterns auto-tuned for 0-tolerance and L_1 penalties with ZNCC-NN$_5$ decoder (refer to Table D1 for details).

E.1. Varying Stereo Baseline

First, we analyze the performance of patterns on the same system but with different baselines. For this case, we fix the camera's and the training board's position and orientation, and move the projector to change the stereo baseline. The camera-to-board distance was approximately 80cm and we tested five baselines (12cm, 24cm, 36cm, and 48cm). We compare the performance of MPS, Hamiltonian, and a la carte with the performance of a system auto-tuned to one of two different baselines (12cm and 48cm). Figure E1 (left) shows the experimental setup, and patterns' performance according to the evaluation metrics corresponding to 0-tolerance and L_1 penalties.

Discussion. Three observations can be made about the results: First, by increasing the baseline the performance of all the patterns drops. Second, training on a particular baseline provides the best performance on that baseline (regardless of penalty function). Third, although optimizing on a particular baseline gives the best performance, auto-tuning a system on one baseline continues to perform well in other baselines as well, outperforming existing methods.

E.2. Varying Surface Orientation

Second, we analyze the effect of surface orientation on reconstruction performance. Here, we fix the baseline (24cm), and the training board's distance with respect to the system (0.8m); and rotate the board (0, 15, 30, and 45 degree). We auto-tune the system at one of two board orientations, and compare their performance with other existing patterns. Figure E1 (right) shows the experimental setup, and the patterns' performance at different orientations.

Discussion. The results show similar behavior to the case of varying baseline. First, almost all the methods' performance drops when the surface tilts away from the baseline; the only exception is the case of auto-tuning at 45 degree which performs better as tilt increases. Second, systems auto-tuned on a specific orientation perform best at that orientation. Third, a system tuned for zero tilt, generalizes well to other tilts, outperforming other existing methods at different angles. This does not always hold for a system tuned for 45 degree tilt. This suggests systems tuned with a board parallel to the baseline perform well for a larger range of orientations.

E.3. Varying Distance

Finally, we evaluate the influence of object-to-system distance on the performance of patterns. For this experiment, we used a room corner as the test scene (Figure 10 of the paper, top right). While keeping the baseline (20cm) and scene fixed, we translated the camera-projector pair backward from the corner (0.8m, 1.8m, 2.8m, 3.8m, 4.8m, and 5.8m) and auto-tune the system for 3 distances using the L_1 penalty: one for a training board at 0.8m, one for a training board at 5.8m, and one for the range of 0.8 to 5.8 meters (by drawing random distance samples). We used the approach in section A.4 to auto-tune for distances beyond 0.8m because the projection patterns are too dim at distances beyond 2m to compute gradients reliably. Figure 10 in the main paper shows an overview of the experimental setup, and the average disparity error as a function of depth.

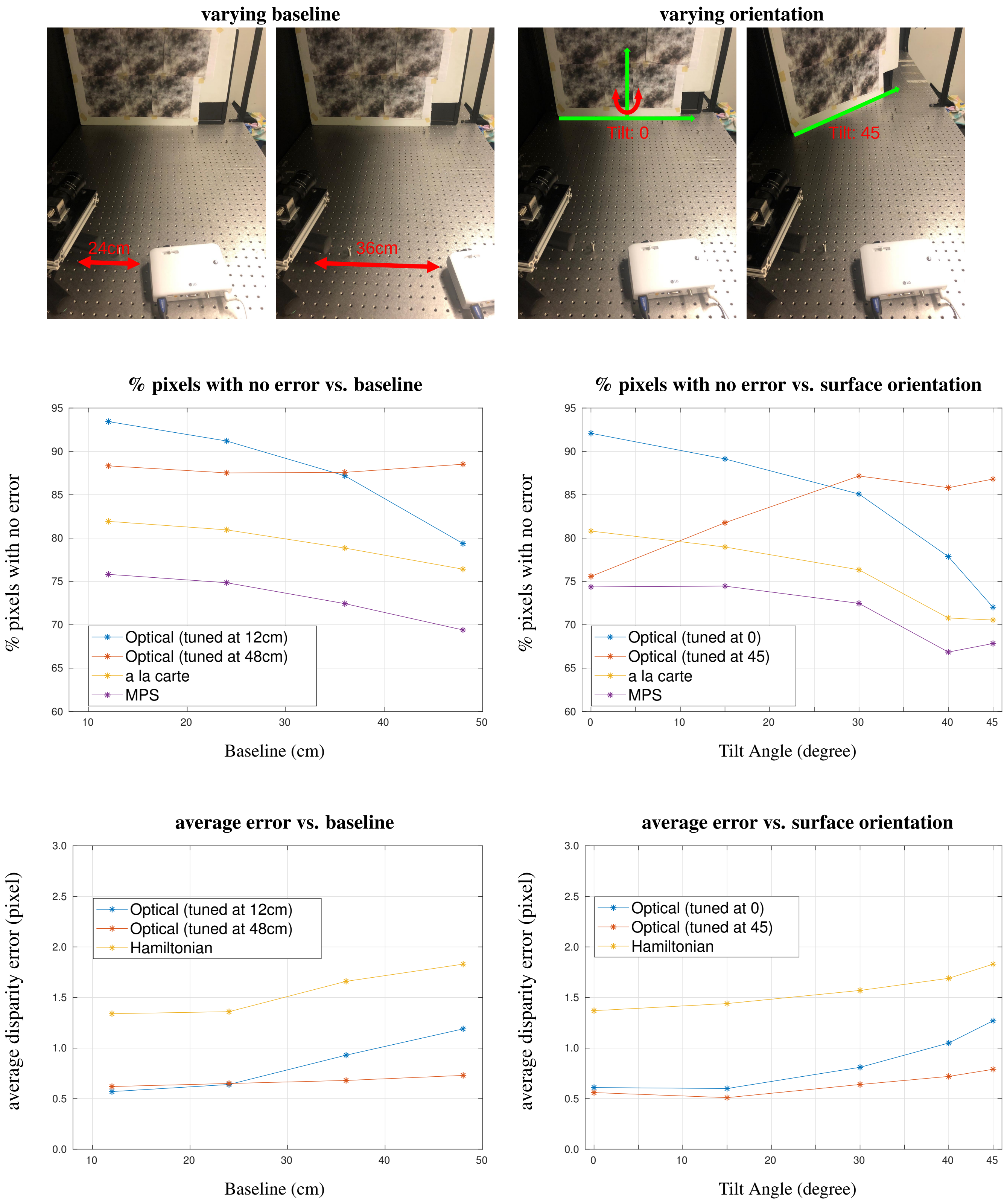
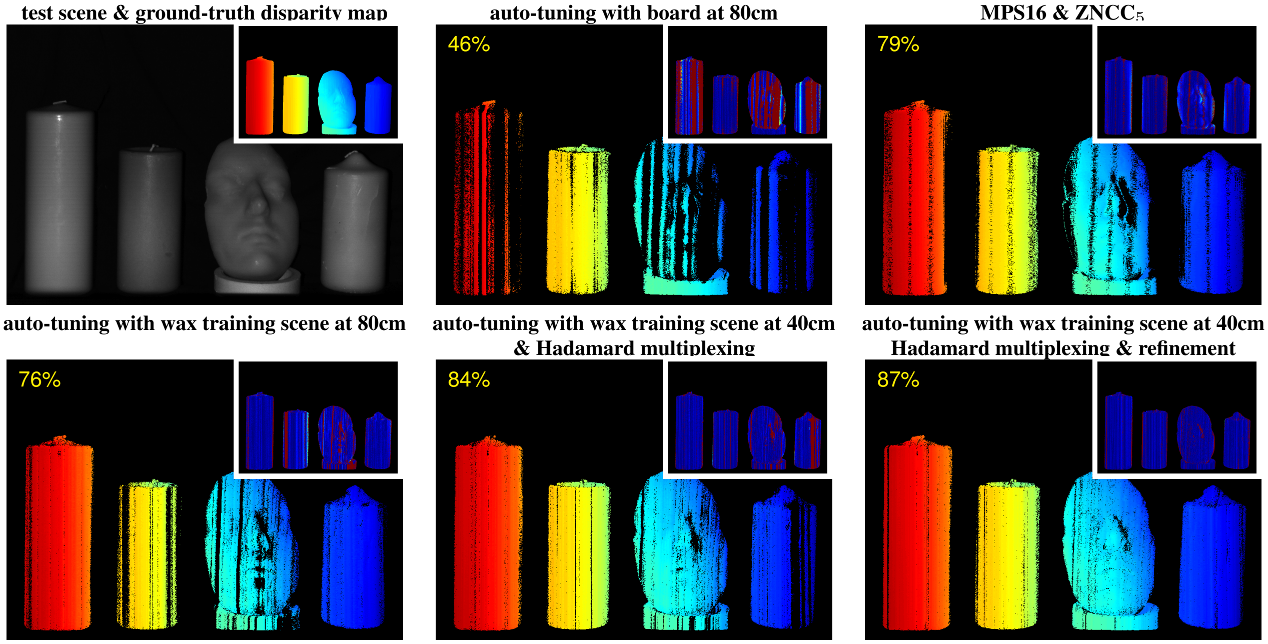


Figure E1: **Comparison of auto-tuned patterns optimized on different geometrical arrangements with existing patterns.**

F. Indirect Light Experiment — Additional Information

Ground truth for auto-tuning was acquired as described in sections A.2 and A.5. After running optical SGD for 2500 iterations with softmax temperature $\tau = 200$, we refine the patterns and the ZNCC-NN$_5$ decoder by executing an additional 1000 iterations with a much higher temperature, $\tau = 1000$. This provides an even tighter continuous approximation to the total penalty, $\|\mathsf{err}(\mathbf{d}, \mathbf{g})\|_1$, at the expense of a harder optimization landscape.

Figure F1 shows the performance improvements caused by individual modifications to our basic optical SGD procedure.


test scene & ground-truth disparity map

auto-tuning with board at 80cm

MPS16 & ZNCC$_5$

auto-tuning with wax training scene at 80cm

auto-tuning with wax training scene at 40cm & Hadamard multiplexing

auto-tuning with wax training scene at 40cm Hadamard multiplexing & refinement

Figure F1: **Auto-tuning for indirect light.** To better visualize reconstruction accuracy, only pixels with error ≤ 2 are shown in the disparity maps above. The total percentage of such pixels is indicated in yellow in the upper left, with the correspondence error map shown as an inset (darkest blue for error=0, darkest red for error ≥ 20).

G. Auto-tuning Structured Light for Different Imaging Systems

This section explores the applicability of optical auto-tuning framework to a broad range of structured light imaging systems.

G.1. Auto-Tuning for One-Shot Depth Estimation.

Experimental Setup. We applied auto-tuning framework to LightCrafter-C2B pair (Table C1) to optimize patterns for single-shot depth estimation (Figures 1 and 7). We set the baseline to 40cm, and place them approximately 80cm away from the scene being imaged (*i.e.* training board and test scenes). The camera field of view is much smaller than the projector's in this system, with only the central 400 projector columns in the camera's field of view. We auto-tune the system for $K = 4$ patterns, the ZNCC-NN$_5$ decoder, and the 1-tolerance and L_1 penalties.

Figure 7 compares auto-tuned systems with a la carte and Hamiltonian patterns, both decoded by ZNCC$_5$. It can be seen that patterns auto-tuned for both error penalties, perform better than other existing pattens (according to each penalty's equivalent evaluation metric) on this one-shot depth imaging system. We also included a video in supplementary materials, which shows raw disparity maps at around 5 Hz for several patterns.

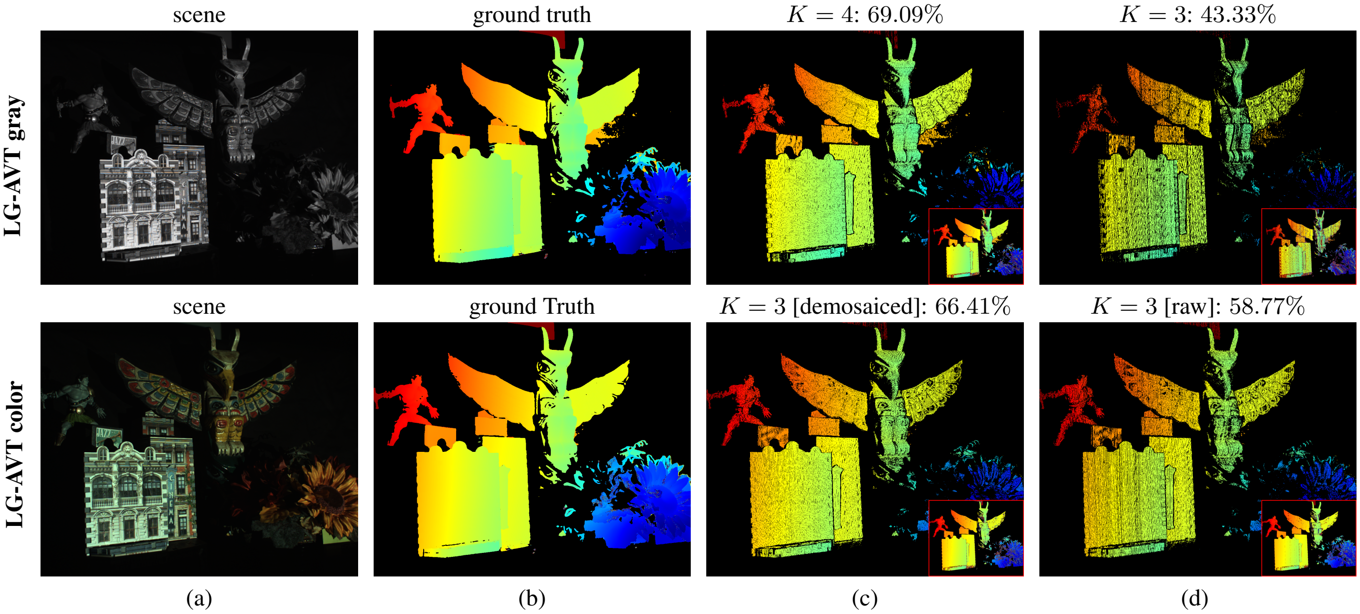


Figure G1: **Comparison of auto-tuned color patterns with gray patterns. Top**: Auto-tuning gray patterns for LG-AVT. **(a)** scene image captured by the monochrome camera. **(b)** 0-tolerance ground-truth. **(c-d)** 0-tolerance disparity map (overlaid with raw disparity map) for $K = 4$ and $K = 3$ with their percentages of zero-error pixels. **Bottom**: Auto-tuning color patterns for LG-AVT. **(a)** scene image captured by the color camera. **(b)** 0-tolerance ground-truth. **(c)** 0-tolerance disparity map (overlaid with raw disparity map) for $K = 3$ color pattern, auto-tuned on demosaiced images with its percentage of zero-error pixels. **(d)** 0-tolerance disparity map (overlaid with raw disparity map) for $K = 3$ color pattern, auto-tuned on raw images with its percentage of zero-error pixels.

G.2. Auto-Tuning Color Projector-Camera Systems

We used the methods described in section A.6 to auto-tune two different projector-camera systems for color structured light.

Color structured light with cellphone. The results are shown in Figure 1 (top row). In this experiment, we auto-tuned the system for $K = 5$ patterns, the ZNCC-NN$_5$ decoder, and 1-tolerance penalty function.

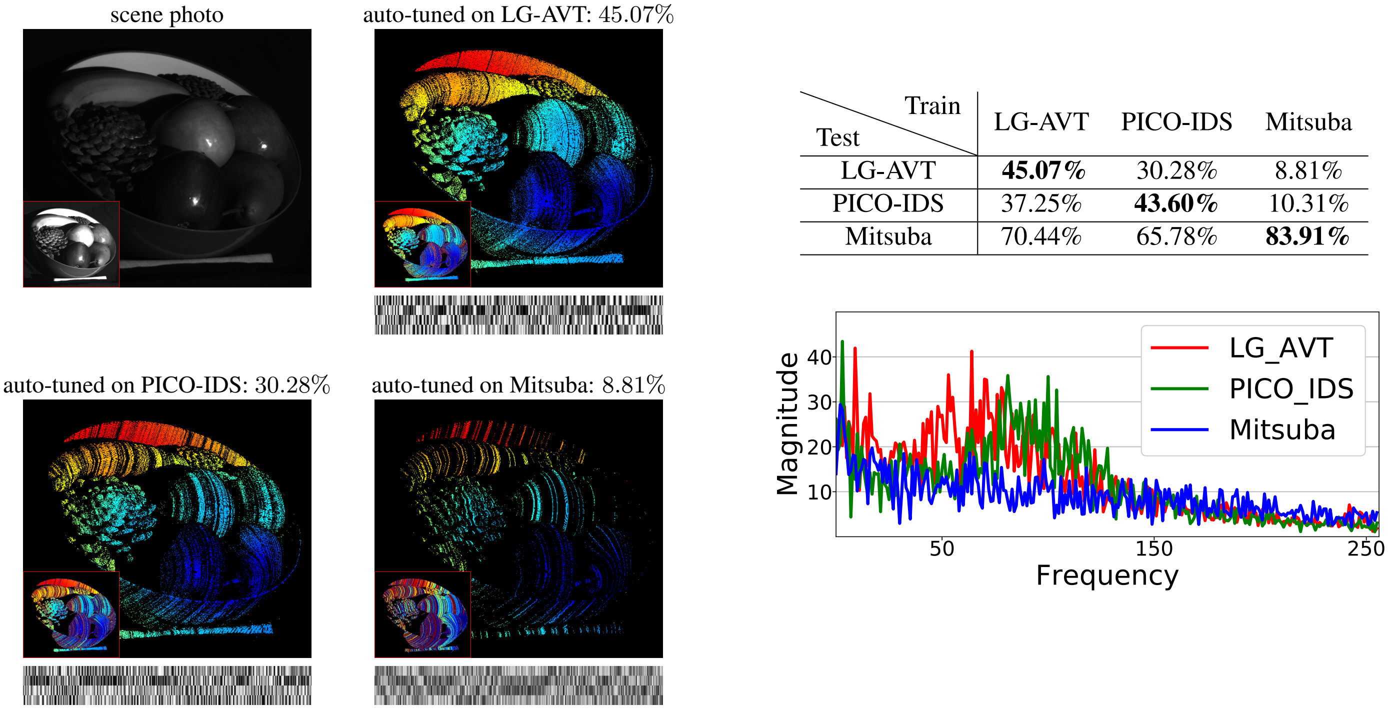
Color vs. monochrome structured light experiment. To do so, we used the same projector (LG) and base camera system (Prosilica AVT with Schneider lens), with the only difference being that the color version of the system had a Bayer filter fitted onto the sensor (models Prosilica 1920c and Prosilica 1920, respectively). Thus, the color camera provided color information at the expense of lower quantum efficiency for its pixels. For both systems, we auto-tune for the 0-tolerance penalty and the ZNCC-NN$_5$ decoder.

Figure G1 compares disparity maps obtained with color patterns and gray patterns, using the approach in Section A.6 to handle color. Two observations can be made about the results. First, despite their lower quantum efficiency and lower number of patterns, 3 color patterns perform comparably to 4 gray patterns. Second, auto-tuning on demosaiced images works better than using the raw images. Intuitively, auto-tuning on demosaiced images benefits from the inference taking place in the demosaicing procedure.

G.3. Transferring Auto-Tuning Results from One System onto Another

Lastly, we investigate how an auto-tuned pattern optimized on a particular imaging system behaves on other systems.

Experimental Setup. We conduct experiments with 3 different camera-projector pairs, where two of them are real devices (LG-AVT gray, and PicoPro-IDS), and the third one is a computational light transport renderer (MitsubaCLT [8]). To have a fair assessment, for MitsubaCLT renderer, we used the same number of projector columns (1280) as other projectors used in this experiments, and also used linear camera-projector model. We auto-tune $K = 4$ patterns for each system with the 0-tolerance penalty and the ZNCC-NN$_3$ decoder. For real devices, we train on the regular training board placed approximately



scene photo

auto-tuned on LG-AVT: 45.07%

auto-tuned on PICO-IDS: 30.28%

auto-tuned on Mitsuba: 8.81%

Test \ Train	LG-AVT	PICO-IDS	Mitsuba
LG-AVT	**45.07%**	30.28%	8.81%
PICO-IDS	37.25%	**43.60%**	10.31%
Mitsuba	70.44%	65.78%	**83.91%**

(a)

(b)

Figure G2: **Performance of system-specific auto-tuned patterns on other imaging systems.** Given a scene (dark fruit jumble) and system (LG-AVT), we compare the patterns and decoders that result from auto-tuning different systems. (a) Scene image as captured by camera is shown in top left (overlaid with its linear scaled version for reference). 0-tolerance disparity maps (overlaid with raw disparity maps) are shown for patterns auto-tuned on different imaging systems when used for reconstructing the scene with LG-AVT system. For each reconstruction, the percentage of zero-error pixels are also included. (b) Top Right: for each testing system, we used the patterns auto-tuned on different imaging systems to reconstruct the same object. We used the fruit jumble for the physical systems (PICO-IDS and LG-AVT). However, since recreating the same object in the rendering environment was infeasible, we used bunny model. Note that the same object was still used for evaluating the patterns on each system. Bottom Right: the mean-magnitude of FFT (Fast Fourier Transform) coefficients for each set of auto-tuned patterns.

80cm from the baseline; for MitsubaCLT we used a flat surface at the same distance as other systems, with the same albedo map as our physical training board (Figure 1, middle row).

We cross-evaluate the auto-tuned patterns and decoders on other systems and show the quantitative and qualitative results in Figure G2; the table provides the percentage of pixels with zero disparity error for all the cases. We observe that for each system we tested, auto-tuned patterns optimized on that particular system perform best. As further example, we used these auto-tuned patterns to reconstruct a scene with the LG-AVT system. Figure G2(a) shows the corresponding 0-tolerance disparity maps along with their percentage of zero-error pixels. This reiterates the fact that auto-tuned patterns are specifically optimized for the system they are trained on. Comparing the frequency content of the auto-tuned patterns (Figure G2(b)) makes clear that these patterns exhibit very distinct spatial structure, in response to the specific characteristics of imaging system for which they are optimized.

H. Proof of Eq. (16)

The total penalty of a correspondence map $\mathbf{d}$ with ground truth map $\mathbf{g}$ is defined as:

$$\|\mathsf{err}(\mathbf{d},\mathbf{g})\|_1 = \sum_{m=1}^{M} \rho(\mathbf{d}[m] - \mathbf{g}[m]) \tag{H1}$$

Using ZNCC-based decoders (Eq.(12-15)) for acquiring correspondence map $\mathbf{d}$, we have:

$$\mathsf{err}(\mathbf{d},\mathbf{g})[m] = \rho(\arg\max_{1\leq n\leq N} \mathbf{z}_m[n] - \mathbf{g}[m]) \tag{H2}$$

Now, we define a max-indicator vector $\mathbf{i}_m$ as follows:

$$\mathbf{i}_m[j] = \begin{cases} 1 & \text{if } j = \text{argmax}_{1\leq n\leq N}\, \mathbf{z}_m[n] \\ 0 & \text{otherwise} \end{cases}$$

Using $\mathbf{i}_m$, we can rewrite Eq. (H2):

$$\mathsf{err}(\mathbf{d},\mathbf{g})[m] = \rho((\mathbf{i}_m \cdot \mathbf{index}) - \mathbf{g}[m](\mathbf{i}_m \cdot \mathbf{1})) \tag{H3}$$

where $\mathbf{index}$ is a vector whose i-th element is equal to its index i; $\mathbf{1}$ represents an all-one vector; and $\cdot$ denotes the dot product.

We can now further simplify Eq. (H3) by factoring $\mathbf{i}_m$:

$$\mathsf{err}(\mathbf{d},\mathbf{g})[m] = \mathbf{i}_m \cdot [\rho(\mathbf{index}[0] - \mathbf{g}[m]), \ldots, \rho(\mathbf{index}[n] - \mathbf{g}[m])] \ . \tag{H4}$$

On the other hand, it is straightforward to show:

$$\lim_{\tau\to\infty} \mathsf{softmax}(\tau \mathbf{z}_m) = \mathbf{i}_m \tag{H5}$$

By combining Eq. (H4) and Eq. (H5), we get:

$$\mathsf{err}(\mathbf{d},\mathbf{g})[m] = \lim_{\tau\to\infty} \mathsf{softmax}(\tau \mathbf{z}_m) \cdot \underbrace{[\rho(\mathbf{index}[0] - \mathbf{g}[m]), \ldots, \rho(\mathbf{index}[n] - \mathbf{g}[m])]}_{\mathsf{err}(\mathbf{index}-\mathbf{g}[m],\mathbf{0})} \ , \tag{H6}$$

Therefore, for sufficiently large τ, we can conclude:

$$\|\mathsf{err}(\mathbf{d},\mathbf{g})\|_1 \ \approx \ \sum_{m=1}^{M} \mathsf{softmax}(\tau \mathbf{z}_m) \cdot \mathsf{err}(\mathbf{index}-\mathbf{g}[m], \mathbf{0}) \tag{H7}$$

QED. □

References

[1] P. Mirdehghan, W. Chen, and K. N. Kutulakos, “Optimal Structured Light à La Carte,” in *Proc. IEEE CVPR*, pp. 6248–6257, 2018. 2, 6

[2] M. Gupta and S. Nayar, “Micro Phase Shifting,” in *Proc. IEEE CVPR*, pp. 813–820, 2012. 2, 3, 6

[3] D. Moreno, K. Son, and G. Taubin, “Embedded phase shifting: Robust phase shifting with embedded signals,” in *Proc. IEEE CVPR*, pp. 2301–2309, IEEE, 2015. 3

[4] M. O’Toole, S. Achar, S. G. Narasimhan, and K. N. Kutulakos, “Homogeneous codes for energy-efficient illumination and imaging,” *ACM TOG (SIGGRAPH)*, vol. 34, no. 4, 2015. 3

[5] Y. Schechner, S. Nayar, and P. N. Belhumeur, “Multiplexing for Optimal Lighting,” *IEEE T-PAMI*, vol. 29, no. 8, pp. 1339–1354, 2007. 3

[6] M. Wei, N. Sarhangnejad, Z. Xia, N. Gusev, N. Katic, R. Genov, and K. N. Kutulakos, “Coded Two-Bucket Cameras for Computer Vision,” in *Proc. ECCV*, pp. 54–71, 2018. 5

[7] M. Gupta and N. Nakhate, “A Geometric Perspective on Structured Light Coding,” in *Proc. ECCV*, pp. 87–102, 2018. 6

[8] J. C. Sun and I. Gkioulekas, “Mitsuba clt renderer.” https://github.com/cmu-ci-lab/mitsuba_clt. 12