A Quantitative Platform for Non-Line-of-Sight Imaging Problems

Jonathan Klein[1, 2]
kleinj@cs.uni-bonn.de
Martin Laurenzis[2]
martin.laurenzis@isl.eu
Dominik L. Michels[3]
dominik.michels@kaust.edu.sa
Matthias B. Hullin[1]
hullin@cs.uni-bonn.de

[1] University of Bonn
[2] French-German Research Institute of Saint-Louis (ISL)
[3] King Abdullah University of Science and Technology (KAUST)

This document contains supplementary information for *A Quantitative Platform for Non-Line-of-Sight Imaging Problems*. Sections are self-contained and they are not necessarily meant to be read in order.

Contents

1 Challenge design

1.1 Setup size and geometry

Our setup uses arbitrary units, but nevertheless we have to decide on the proportions of the individual parts. As the setups in the previously published work vary greatly (see Table 1), we focused on three main quantities: The size of the reflector, the distance between object and reflector and the size of the object itself (see Figure 1).

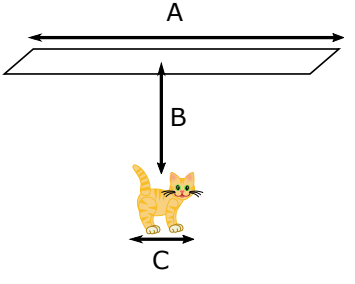


Figure 1: We categorize existing setups by the ratios of the reflector (A), distance to the object (B) and the size of the object(C).

We chose the size of our setup (shown in Figure 3 in the paper) by taking the geometric mean of the individual values and applied some manual adjustment (e.g., particularly difficult proportions were weighted less, if the reported results on them are inferior to the average).

Other interesting quantities of setups are its temporal resolution T, as well as the number and distribution of directly visible scene locations that are illuminated (N_i) and observed (N_o) over the course of the measurement routine. They are also shown in Table 1. We oriented our temporal resolution towards the resolution of streak cameras [12], as they offer the highest resolution and use a single illumination point with a regular grid of observation points.

Table 1: Key specifications for various setups reported in literature: temporal resolution T (histogram bin size or point spread function, whichever is greater); numbers of observed (N_o) and illuminated (N_i) locations; scene dimensions A, B and C as illustrated in Figure 1; and ratios of these dimensions. All values are approximate; those in parentheses have been estimated by authors of this paper from information provided in the respective works. Entries marked AMCW or ∞ denote amplitude modulated correlation sensors and steady-state intensity imagers, respectively.

Ref.	T [ps]	N_o	N_i	A [cm]	B [cm]	C [cm]	A/B	B/C
[7]	250	1	1	(2.5)	(4.3)	(3)	0.6	1.4
[9]	1.6	(672×512)	> 1	25	(15)	(1–1.5)	1.6	12
[12]	15	672	> 1	40×25	25	1.5×8.2	1.4	5
[5]	AMCW	160×120	1	200	150	(80)	1.3	1.8
[2]	30	1	185	100×80	150	40	0.6	3.8
[1]	AMCW	176×144	1	200×100	100	4	1.5	25
[8]	∞	320×240	1	130	(60)	80×30	2.1	1.2
[4]	110	32×32	1	15×30	45	30×10	0.5	2.6
[3]	64	3	1	30	50	15×15	0.6	3.3

2 Datasets

All datasets are rendered using the scene arrangement illustrated in Figure 3 in the main paper. Data is provided as transient images $I(u,v,\tau)$, where the image coordinates $(u,v) \in [0,1,\ldots,255]^2$ address square-shaped wall elements ("pixels") of size 0.002^2 in the X-Z plane, and the τ dimension is discretized in 1600 bins of size $d\tau = 0.001$ starting at $\tau = 0$ (τ is a measure of the path length and must be divided by the speed of light to retrieve the travel time). The exchange format for transient images is specified in Section 4. Extents and discretization of the temporal dimension are specified within each dataset.

The datasets are available for download at our website `https://nlos.cs.uni-bonn.de/`.

2.1 Geometry reconstruction

We consider geometry reconstruction the most important challenge, as it is not only the focus of most of the previous work, but also the most general problem with the highest number of degrees of freedom. In most scenarios, once a full geometric scene model has been reconstructed, derivative information such as object positions or classes can be obtained more easily from 3D geometry than from from raw transient images.

We split our test scenes into several categories that test different capabilities of solvers. Each scene contains a single object which has to be reconstructed. We define five categories of objects:

Cat. 1 2D shapes. This category contains two-dimensional objects of a certain thickness perpendicular to the wall. This category is closely related to the texture reconstruction challenge (Section 2.4).

Cat. 2 Simple geometric shapes. Objects with simple mathematical descriptions without additional surface details.

Cat. 3 Simple objects. Everyday objects from the real world, with limited geometric detail. Objects in this category are not easily approximated by the shapes of the previous category.

Cat. 4 Complex objects. Highly non-convex objects with complex shape but without fine surface details.

Cat. 5 Difficult objects. Objects with thin elements, fine structures and complex topology (e.g. many holes).

A full listing of datasets is given in Table 2. To facilitate the development and refinement of reconstruction techniques, ground truth geometry in `.obj` format is provided for some of the datasets; for all others, the true geometry remains unknown.

2.2 Position and orientation tracking

For tracking, three different rigid objects are used: a sphere, a golem figurine and an airplane (see Table 3). For each object, there are a total of four challenges:

1. movement along the three main axes without rotation,

Table 2: Objects and their categories for the geometry reconstruction challenge.

Category	Dataset name	Material	Ground truth provided
1	`LetterK`	diffuse	yes
1	`LetterQ`	diffuse	no
2	`Box`	diffuse	yes
2	`Cone`	diffuse	no
3	`StanfordBunny`	diffuse	yes
3	`UtahTeapot`	diffuse	yes
3	`Ax`	diffuse	no
3	`Hammer`	diffuse	no
3	`Cup`	specular	no
4	`StanfordDragon`	specular	yes
4	`Dinosaur`	diffuse	no
4	`FlyingDragon`	diffuse	no
4	`IndoorPlant`	diffuse	no
5	`Chair`	diffuse	no
5	`Bike`	specular	no
5	`Greenhouse`	specular	no

2. rotation at a fixed position along the three main axes,
3. movement along a complex path with constant orientation and
4. movement along a complex path while changing orientation.

Due to its symmetry, the challenges involving rotations are not included for the sphere dataset. The paths are different for each object and challenge, e.g. the golem moves along one path with constant orientation and along another path for the combined orientation and translation.

The object origins lie in the center of mass for each object. This way, the object position is uniquely defined for an ideal reconstruction of the scene, however we do expect a certain bias in the position in realistic scenarios. Therefore we consider the residual RMS to be the most important metric.

Each path forms a loop and contains 40 frames. The axes movement and static rotation consist of 30 frames, with 10 frames for each axis. The movements and rotations around the axes are designed to be easy to reconstruct and to make systematic errors and missed frames obvious.

The exact geometry of the golem is given as an `.obj` file and can be used to improve the reconstruction (e.g., by fitting it into a partial geometric reconstruction of each frame). The shape of the airplane is not revealed to test tracking of unknown objects.

2.3 Object classification

The goal of the object classification task is to assign the transient image to one of eleven object geometries, as listed in Table 4 and shown in Figure 4a in the main paper. All object shapes are given as `.obj` files and should be used to perform the classification.

The objects were scaled to equal surface area to prevent classifying by the size of the object, i.e. amount of reflected light. However, this approach is not necessarily perfect as very compact or concave objects appear smaller when scaled by their surface area.

Table 3: Overview of the object tracking datasets.

Dataset name	Material	Shape known	Position	Rotation
SphereAxesPos	diffuse	yes	yes	no
SpherePathPos	diffuse	yes	yes	no
GolemAxesPos	diffuse	yes	yes	no
GolemAxesRot	diffuse	yes	no	yes
GolemPathPos	diffuse	yes	yes	no
GolemPathRot	diffuse	yes	yes	yes
AirplaneAxesPos	specular	no	yes	no
AirplaneAxesRot	specular	no	no	yes
AirplanePathPos	specular	no	yes	no
AirplanePathRot	specular	no	yes	yes

Table 4: Overview of the object classification dataset.

Dataset name	Material
Cat	diffuse
Icosphere	diffuse
LetterG	diffuse
Parallelepiped	diffuse
Plant	diffuse
SpoonDiffuse	diffuse
Whale	diffuse
Gramophone	specular
Headphones	specular
Pan	specular
SpoonSpecular	specular

2.4 Planar textures

In this challenge, a textured planar target of extent $(x,z) \in [-0.1,0.1]^2$ placed in front of the laser spot at $y = -0.3$ (see Figure 2) is the subject of the reconstruction. The texture, represented by a grayscale image of dimension $n \times n$, modulates the albedo (diffuse reflectance) of the surface. Each value $\rho_{s,t}$ within the texture covers a square-shape region of size $(0.2/n, 0.2/n)$ with a value in the range of $[0,1]$. We provide a variety of datasets that feature black-and-white and grayscale textures of different resolution (Table 5).

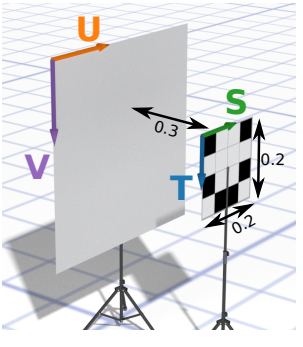


Figure 2: Texture reconstruction setup.

For the evaluation, results with arbitrary resolutions are supported. If the reconstruction T' has the same resolution as the reference, identical decomposition steps can be applied. Otherwise it is scaled to the closest power of 2 before decomposing it. If the pyramid of the reconstruction contains more layers (i.e. it had a higher input resolution), the highest layers are discarded; if it contains fewer layers, the missing ones are filled with zeros (as it did not contain any information about the higher frequencies). Therefore, reconstructions in the reference solution are preferred.

Table 5: Overview of the texture reconstruction datasets.

Dataset name	Resolution	Pixel depth
`Character`	4×4	$\{0,1\}$
`Digit`	4×4	$\{0,1\}$
`Letter`	4×4	$\{0,1\}$
`Smiley`	4×4	$\{0,1\}$
`House`	16×16	$\{0,0.25,0.5,0.75,1\}$
`Number`	16×16	$\{0,0.25,0.5,0.75,1\}$
`Pattern`	16×16	$\{0,0.25,0.5,0.75,1\}$
`Text`	16×16	$\{0,0.25,0.5,0.75,1\}$
`Books`	128×128	$[0,1]$
`Concert`	128×128	$[0,1]$
`Fan`	128×128	$[0,1]$
`Industrial`	128×128	$[0,1]$

3 Rendering

We used a modified version of `pbrt-v3`[□] to render the transient images.

3.1 Importance sampling

The transient images of our scenes contain only light from indirect reflections, which can make the rendering very inefficient if no special care is taken. Sampling from the wall towards the light source is futile, as the laser spot illuminates only the object, not the wall. Sampling the hemisphere over the wall is inefficient, as most objects of interest only cover a small solid angle on the hemisphere and are thus unlikely to be hit.

To improve the performance for this light transport scenario, we implemented a custom importance sampling that is heavily inspired from the area light source sampling already implemented in `pbrt-v3`. The triangles of the object are stored in a special list and both the wall and the object triangle are tagged with specific flags. During path tracing, everytime a ray hits the wall, we can sample directly into the direction of the object, as the wall can only receive light that was reflected from the object. These samples need to be normalized by considering the area, angle and distance of the triangle of the object to ensure the correct expected value of the sampling.

Our tests show that this custom importance sampling is two to three orders of magnitude more efficient than naive sampling. Physical correctness is preserved by only eliminating zero-radiance paths, e.g. interreflections on the object surface remain untouched by this optimization.

4 Transient image files

As of writing this paper, there is no standard format for storing transient images. It would seem like an canonical and attractive choice to resort to standard image formats for which suitable I/O libraries exist; however, such files would have to be accompanied by separate metadata specific to the dataset, and most image formats rigidly adhere to a Cartesian pixel arrangement. Custom formats have also been proposed, for example Arellano et al.'s `.float` and `.lasers` files [□], a format that would require thousands of individual files

to store a single dataset from our database, and in its current form is not expressive enough to cover new capture geometries such as O'Toole et al.'s confocal setting [10]. We therefore propose a new format that is compact (one file per dataset), easy to read and write, and at the same time flexible enough to cater to the needs of emerging research directions. Here we give a high-level overview over the file format. A detailed implementation guide comes as part of the SDK.

A transient image file consists of 4 blocks: The *file header* contains general information and the sizes of each remaining block. The *pixel data* block contains a linear array of transient pixels. The pixel interpretation block is an efficient representation of the illumination and observation points of each pixel. Finally, the *image properties* block contains arbitrary, JSON-encoded meta-data of the image.

For the *pixel interpretation* and *image properties* blocks we made some noteworthy design choices, which we will discuss in the following.

4.1 Pixel interpretation block

Traditionally, a single point on the reflector is illuminated while a regular grid of points on the reflector is observed. Due to the reciprocity of the light transport, this can be reversed, e.g. as done by Buttafava et al. [2]. In general, multiple illumination and observation points can be used (which may or may not be arranged in a regular grid), and in the extreme case of [10], a unique illumination and observation point is used for every pixel.

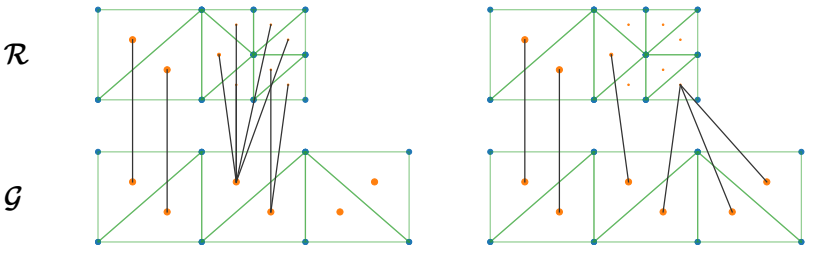
To support all these cases and still have an efficient representation of our data, the observation and illumination points can be stored in different modes. Mode 0 is the most general one and requires no structure in the observation or illumination points. They are stored for every pixel individually, however this introduces a certain overhead (which becomes negligible, if each pixel consists of a large number of bins). The SDK provides an upconverter to Mode 0 from the other, more specialized ones.

Mode 1 assumes a single illumination point and a regular grid of observation points, thus the transient image has a meaningful x and y resolution. Observation point positions are implicitly stored by the grid properties and their position in the linear *pixel data* array (as it is the case in raster graphic formats). Mode 2 is the reciprocal case, where the roles of observation and illumination points are reversed.

4.2 Image properties block

Image meta-data is widely used to store additional information such as the camera settings used to capture the image. In Transient image files they are stored as an UTF-8 encoded JSON string at the very end of the file. A number of standard fields are specified, however users are free to add their own ones.

This approach has multiple advantages: Meta-data can be read and written using a binary-compatible text editor, all fields are optional, new properties can easily be added, and readers and writers are quick to implement due to the wide availability of JSON en-/decoders.



(a) Top to bottom: $d(\mathcal{R},\mathcal{G})$ (b) Bottom to top: $d(\mathcal{G},\mathcal{R})$

Figure 3: Illustration of the correspondence selection for our surface comparison metric. For each triangle of the source, the closest triangle of the target is selected. The lower object is the ground-truth geometry $\mathcal{G}$, while the upper object is the reconstruction $\mathcal{R}$.

Table 6: Asymmetric reconstruction errors between a fast backprojection reconstruction (M1), a ground truth mesh (M2) and the same mesh after one level of Catmull-Clark subdivision (M3). In boldface, the distance $d(\mathcal{G},\mathcal{R})$ from ground truth to a test geometry measures the incompleteness of the reconstructed surface. As intended by design, this distance measure is significantly less sensitive to remeshing (third row) than it is to actual missing geometry (first and second rows).

Comparison	$d(\mathcal{R},\mathcal{G})$	$d(\mathcal{G},\mathcal{R})$
$\mathcal{R}=$ M1, $\mathcal{G}=$ M2	$5.255\cdot10^{-3}$	$\mathbf{1.819\cdot10^{-2}}$
$\mathcal{R}=$ M1, $\mathcal{G}=$ M3	$5.132\cdot10^{-3}$	$\mathbf{1.813\cdot10^{-2}}$
$\mathcal{R}=$ M3, $\mathcal{G}=$ M2	$4.729\cdot10^{-4}$	$\mathbf{6.438\cdot10^{-5}}$

5 Comparison metrics

Figure 3 shows how the closest points are selected during the evaluation of the asymmetric mesh-to-mesh distance (Equation 2 in the main paper). The reconstruction $\mathcal{R}$ (top) misses the right third of the surface that is known to exist in the ground-truth mesh $\mathcal{G}$ (bottom), and it has a finer tessellation in the middle segment. In the distance from reconstruction to ground truth, this results in more connections in the middle segment, compared to the left segment. Some of these connections are shorter and some are longer, but their average is roughly the same as in the case of equal tessellation. As they are weighted by the triangle area, the total cost for the middle part does not increase significantly by the additional connections. Overall, the cost is quite similar to the leftmost geometry segment.

The right part of $\mathcal{G}$ is missing in $\mathcal{R}$. In the distance from original to reconstruction ($d(\mathcal{G},\mathcal{R})$), this results in longer connections and thus in increased cost.

Additionally, the misalignment of the meshes increases the length of all connections and thus the overall distance.

The tessellation of an object does have a certain influence on the comparison result; however, it is small. To avoid bias from tesselation, all used models are tessellated finer than the maximal expected reconstruction resolution. Furthermore, trivial subdivision of the triangles can further reduce this bias if needed and does not require to render or reconstruct the scenes again.

In Table 6, we compare the reconstruction error introduced by a change in tessellation to that of a state-of-the-art reconstruction.

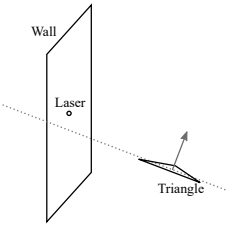


Figure 4: Culling of backfaces. Triangles that are facing away from the laser spot $(x,y,z) = (0,0,0)$ are removed from the comparison. All others are kept, even those whose normal vector points in positive z direction (away from the wall).

Table 7: Fast Backprojection Reconstruction Results. a) Geometry reconstruction: The greater number of the two (marked in bold) is the symmetric distance of reference and reconstruction (Equation 2 in the main paper). b) Position tracking: Columns from left to right: RMS distance (Equation 3 in the main paper), offset length, RMS residual after subtraction of offset, completeness of trajectory (percentage of recovered frames).

(a)

Scene	$d(\mathcal{R},\mathcal{G})$	$d(\mathcal{G},\mathcal{R})$
Axe	0.00376	**0.00645**
Bike	0.00675	**0.00916**
Chair	0.00429	**0.0137**
Cone	**0.0129**	0.00867
Cube	**0.0743**	0.00686
Cup	0.00809	**0.0283**
Dino	0.00500	**0.0172**
Dragon	**0.0128**	0.00453
Greenhouse	0.0133	**0.0200**
Hammer	**0.0108**	0.00876
IndoorPlant	0.00438	**0.0162**
K-Letter	**0.0200**	0.00734
Q-Letter	0.00625	**0.00631**
StanfordBunny	0.00356	**0.0155**
StanfordDragon	0.00359	**0.0202**
UtahTeapot	0.00341	**0.0362**

(b)

Scene	RMS dist.	‖Offset‖	RMS res.	Compl. [%]
Golem Axes	0.0238	0.0216	0.0101	100
Golem Path	0.0685	0.025	0.0638	100
Sphere Axes	0.0488	0.0485	0.0056	100
Sphere Path	0.0535	0.0504	0.018	100
Plane Axes	0.0269	0.0127	0.0237	100
Plane Path	0.05	0.0167	0.0472	100

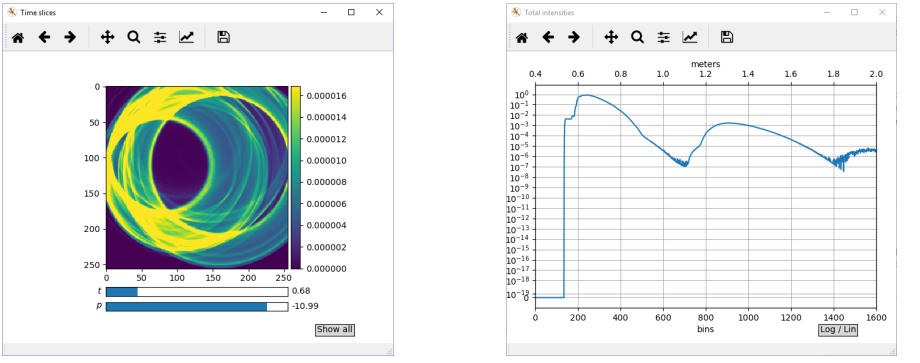
5.1 Backface culling

Under normal conditions, it cannot be expected that the backside of an object can be well reconstructed, as usually little to no information will reach the wall. Therefore triangles facing away from the reflector are discarded before the mesh distance is evaluated.

Figure 4 shows an example of a triangle that is pointing away from the reflector but still visible from the laser spot. We use this visibility of the laser spot as a culling criterion, instead of only checking whether the face normal is pointing away from the reflector. This is still not always correct, as global illumination can also allow light from culled triangles to reach the reflector (and likewise, triangles facing towards the reflector can be completely occluded), but taking all of these effects into account is not possible in a simple and transparent manner.

6 Tools

We provide a variety of tools for handling the transient images. At the core are loaders and writers for various programming languages including C++, Python and Matlab. Together



(a) Time slices (b) Histogram

Figure 5: Our viewer shows time slices and histograms of transient images.

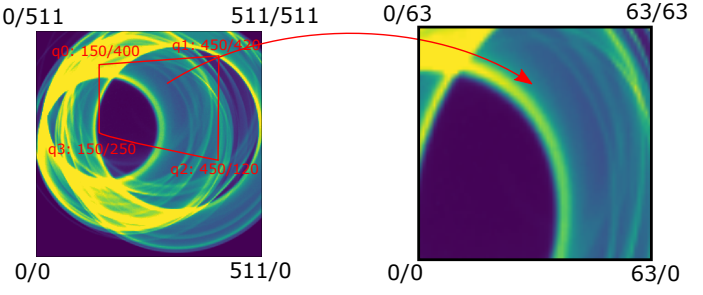


Figure 6: The camera converter uses a homography defined by four points pairs to resample a transient image.

with the file format description, they should allow a quick integration of our datasets in other frameworks. The tools can be found at our website https://nlos.cs.uni-bonn.de/.

6.1 Image viewer

We provide a simple viewer for transient images based on Python and Matplotlib. The user can scroll through time and adjust the intensity scale. A second view shows the transient image integrated over the spatial domain. The resulting histogram illustrates how much light arrived at what time, on either a linear and logarithmic scale. The viewer is shown in Figure 5.

6.2 Setup converter

Many setups seen in the real world have different illumination and viewing geometries. While a change in laser spot position would require re-rendering the scene, other camera placings and projections can be accommodated to make results more comparable. To this end, we offer a resampling tool to convert transient images to different camera positions.

At the heart of the resampling is a homography as depicted in Figure 6. The user defines

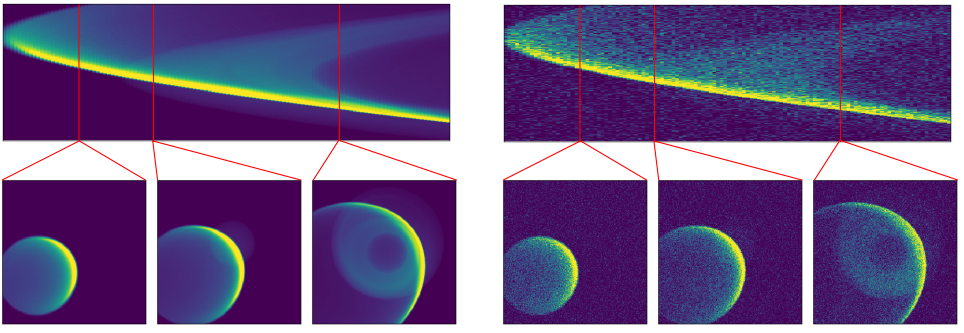


Figure 7: Transient image of the `Hammer` scene before (left) and after (right) applying the noise model `SPAD`.

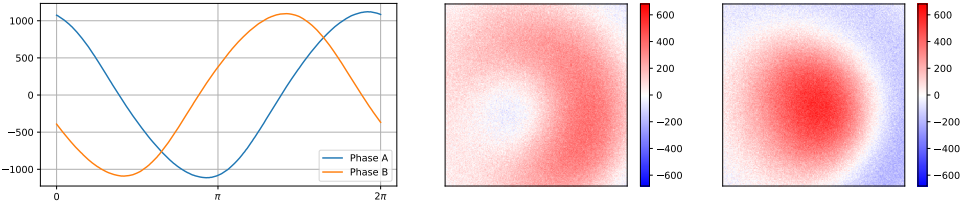


Figure 8: Synthetic AMCW measurements of the `Hammer` scene. Left: Measured demodulation functions from a *PMDTec CamBoard nano* AMCW camera. Middle/Right: The two phase images (0° and 90°) computed using the sensor model.

the four point pairs in the old and new image from which the homography is computed. Applying it to the image jointly crops, transforms and rescales the image. The user can also specify a camera and laser position as three-dimensional coordinates which are used to compute a temporal offset for each output pixel. Additional, the temporal window of the output image can be changed. For the resampling, a Mitchell-Netravali filter with customizable size for both spatial and temporal filtering is used. If no size is specified, reasonable filter sizes are computed from the homography.

The tool is written in C++ with no external dependencies. It is implemented as command line tool and thus ready for integration in batch processing.

6.3 Fast backprojection integration

All scripts that were used to export the transient images to the *fast backprojection* solver by Arellano et al. [1] are available on our website. This allows the user to set up a complete reconstruction pipeline and re-evaluate all results in the paper.

6.4 Sensor models / noise

The suite of scripts and tools contains two noisy sensor models to reflect the characteristic behavior of two important types of device: `AMCW`, a simple model of a correlation ToF sensor (4-tap near-sinusoidally modulated correlation time-of-flight measurement with Skellam-distributed shot noise) and `SPAD`, a single-photon counter with Poisson-distributed

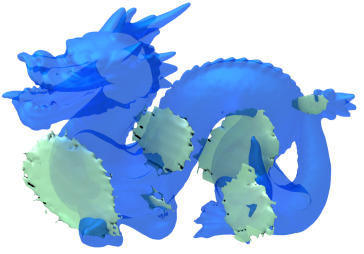


Figure 9: Reconstruction of the Stanford Dragon. The ground truth geometry is shown in blue, while the reconstructed geometry is green.

shot noise and dark counts. Figure 7 shows an example dataset before and after applying the `SPAD` model with standard settings. Figure 8 shows images for the same scene as seen through the `AMCW` sensor model.

7 Reconstruction results

Our evaluation metrics aim to make different reconstruction algorithms comparable by reducing their overall performance to a single number (that are shown in Table 7). However, these error terms arise from various characteristics of the algorithm which are interesting to study, as they increase the understanding of the behavior and point at possible improvements. Thus we now discuss some characteristics of the backprojection example used in the paper. It should be noted that we did not tune parameters to achieve the highest possible accuracy. Instead, these examples serve to illustrate the typical behavior of a reconstruction algorithm.

7.1 Geometry reconstruction

Figure 9 shows the reconstruction of the Stanford Dragon model. The reconstruction mainly consists of planar patches that are parallel to the wall which is very typical and seen in almost all reconstructions. The Laplacian filter used after backprojection to isolate surfaces that favors flat structures and thus struggles with surfaces that are curved or not aligned with the wall. The resulting reconstructions are often incomplete, low in detail, and they feature a distinctive “cloud-of-pancakes” look.

7.2 Position tracking

Our naive tracking position implementation reconstructs first the object geometry and then uses its center of mass as object position. Thus it is very vulnerable to incomplete geometry reconstructions.

Figure 10 shows two frames of the `AirplaneAxesPos` dataset, where the plane moves along the X axis. Both reconstructions are incomplete and favor geometry close to the laser spot, which lies in between both positions. When the center of masses are computed, the movement of the object thus appears to be smaller than it actually is (see Figure 11). A more sophisticated algorithm could be aware of this shortcoming and try to fit the given object geometry into its reconstruction to determine which part of the plane was reconstructed.

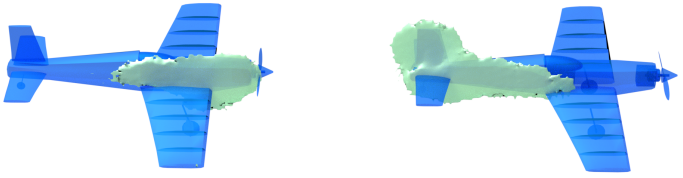


Figure 10: Reconstructed geometry as it is used during position tracking. The ground truth geometry is shown in blue, while the reconstructed geometry is green. In different frames, different parts of the plane were reconstructed, resulting in an error in the position reconstruction.

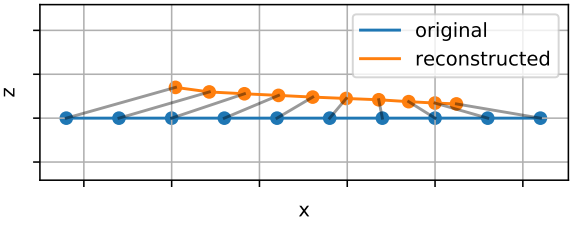


Figure 11: Reconstructed trajectory of the airplane for the movement along the X axis. Apart from the offset in Z direction, the reconstructed path is too short.

References

[1] Victor Arellano, Diego Gutierrez, and Adrian Jarabo. Fast back-projection for non-line of sight reconstruction. *Opt. Express*, 25(10):11574–11583, May 2017. doi: 10.1364/OE.25.011574. URL http://www.opticsexpress.org/abstract.cfm?URI=oe-25-10-11574.

[2] Mauro Buttafava, Jessica Zeman, Alberto Tosi, Kevin Eliceiri, and Andreas Velten. Non-line-of-sight imaging using a time-gated single photon avalanche diode. *Optics Express*, 23(16):20997–21011, 2015.

[3] S. Chan, R.E. Warburton, G. Gariepy, Y. Altmann, S. McLaughlin, J. Leach, and D. Faccio. Fast tracking of hidden objects with single-pixel detectors. *Electronics Letters*, 53(15):1005–1008, 7 2017. ISSN 0013-5194. doi: 10.1049/el.2017.0993.

[4] Genevieve Gariepy, Francesco Tonolini, Robert Henderson, Jonathan Leach, and Daniele Faccio. Detection and tracking of moving objects hidden from view. *Nature Photonics*, 10(1), 2016.

[5] Felix Heide, Lei Xiao, Wolfgang Heidrich, and Matthias B. Hullin. Diffuse mirrors: 3D reconstruction from diffuse indirect illumination using inexpensive time-of-flight sensors. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[6] Achuta Kadambi, Hang Zhao, Boxin Shi, and Ramesh Raskar. Occluded imaging with time-of-flight sensors. *ACM Transactions on Graphics*, 35(2):15:1–15:12, March 2016. ISSN 0730-0301. doi: 10.1145/2836164. URL http://doi.acm.org/10.1145/2836164.

[7] A. Kirmani, T. Hutchison, J. Davis, and R. Raskar. Looking around the corner using transient imaging. In *Proc. ICCV*, pages 159–166, 2009.

[8] Jonathan Klein, Christoph Peters, Jaime Martín, Martin Laurenzis, and Matthias B. Hullin. Tracking objects outside the line of sight using 2d intensity images. *Scientific Reports*, 6(32491), August 2016. doi: doi:10.1038/srep32491. URL http://www.nature.com/articles/srep32491.

[9] N. Naik, S. Zhao, A. Velten, R. Raskar, and K. Bala. Single view reflectance capture using multiplexed scattering and time-of-flight imaging. *ACM Trans. Graph.*, 30(6): 171, 2011.

[10] Matthew O'Toole, David B. Lindell, and Gordon Wetzstein. Confocal non-line-of-sight imaging based on the light-cone transform. *Nature*, 2018. doi: http://dx.doi.org/10.1038/nature25489.

[11] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Elsevier Science, 3rd edition, 2016. ISBN 9780128007099. URL https://books.google.de/books?id=iNMVBQAAQBAJ.

[12] A. Velten, T. Willwacher, O. Gupta, A. Veeraraghavan, M.G. Bawendi, and R. Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature Communications*, 3:745, 2012.