`mitransient`: Transient light transport in Mitsuba 3[†]

Diego Royo[1⋆] Miguel Crespo[2⋆] Jorge García-Pueyo[1]

[1]Universidad de Zaragoza - I3A, Spain [2]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

⋆ Joint first authors.

† Part of this work is published in Computers & Graphics [1].

Transient imaging aims to capture and analyze how light propagates and interacts through a scene with an ultra-high temporal resolution, reaching the order of picoseconds. This increase of information about the temporal domain provides several benefits that have led to numerous applications, such as visualization of light in motion [2] (Fig. 1), vision through highly-scattering media [3], and inferring material properties from a distance. One of the most prominent applications of transient imaging is the reconstruction of hidden objects from captured indirect illumination (Fig. 2). This problem, known as non-line-of-sight (NLOS) imaging, is especially relevant to our work.

Unfortunately, most methods that rely on transient imaging require expensive hardware, including ultrafast cameras and pulsed illumination devices, which are difficult to calibrate and operate. *Transient light transport simulation* emerges as an alternative tool for developing, prototyping, and testing such systems, without hardware difficulties. However, there are no available state-of-the-art systems that fulfill the needs of the users, meaning that they are ad-hoc implementations for specific tasks (difficult to extend to other applications) or they lack novel optimized strategies (they are not as fast as they could be).

Our work fixes these issues by building a rendering tool that can simulate different sorts of time-resolved sensing devices (including time-gated and transient cameras), with easy-to-extend modules written in Python, which leverages the state-of-the-art technology of Mitsuba 3 [4]. Specifically, we can simulate the interactions of light with complex materials and participating media, run in both CPU and GPU by using vectorized JIT compiled code, compute ray tracing queries with optimized acceleration structures, and compute derivatives including the temporal domain.

We have published our code in GitHub[1] and as a PyPi package[2] for easy installation. In Sections 1 and 2 we showcase two use cases of our simulator using time-gated and transient cameras.

[1]`https://github.com/diegoroyo/mitsuba3-transient-nlos/`

[2]`https://pypi.org/project/mitransient/`

1 Transient path tracing

Our additions to Mitsuba 3 include extensions to steady-state path-tracing algorithms following the theory of the work of Jarabo et al. [2], including both surface `transient_path` and volumetric transport `transient_volpath`. Notice that our system is capable of simulating conventional, time-gated, and transient cameras, including effects such as time unwarping [5]. Furthermore, both RGB and spectral rendering are supported, covering the range of possible experiments that could be done.

Fig. 1 shows multiple results from our transient algorithms in both transport regimes: **(a)** We leverage the complex Staircase scene to demonstrate several computations that can be done with our system with light having multiple bounces inside of it. In addition, **(b)** presents a recreation of the iconic Bottle scene from Velten et al. [5], including a bottle containing water with some drops of milk for added scattering, which we model using a rough plastic material containing a participating medium with coefficients similar to the physical ones.

Transient derivatives. In Fig. 1 **(a)** we demonstrate the capabilities of our algorithm to compute derivatives with respect to scene parameters. Specifically, we show in the rightmost inset the transient forward derivatives of the material of the floor computed using `transient_path`.

Realistic hardware noise. Light transport simulations ignore many sources of noise that happen using real hardware (e.g. temporal jittering of the signal [6]). To help bridge the gap between perfect simulations and the real world, our system can also use measures from real hardware devices to process the signal, adding realistic noise.

2 Non-line-of-sight (NLOS) application

Non-line-of-sight imaging refers to the family of algorithms that reconstruct information from the non-directly visible parts of the scene from the sensor. Conventional path-tracing techniques are ill-suited for NLOS scenes. For this purpose, we

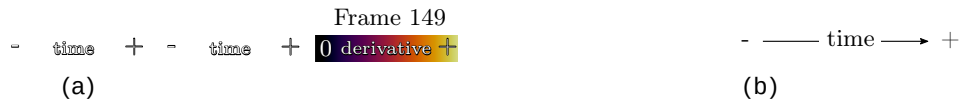

Figure 1: **(a)** STAIRCASE scene simulated using `transient_path` integrator. From left to right: conventional camera, time-gated camera, peak time visualization, warped peak time visualization, and transient forward derivatives (with respect to the material of the floor). **(b)** BOTTLE scene filled with water and milk, being illuminated by a laser from its bottom. Up: real scene and frames of the propagation of light through the bottle captured with a streak camera. Bottom: synthetic scene and frames computed by our `transient_volpath`.

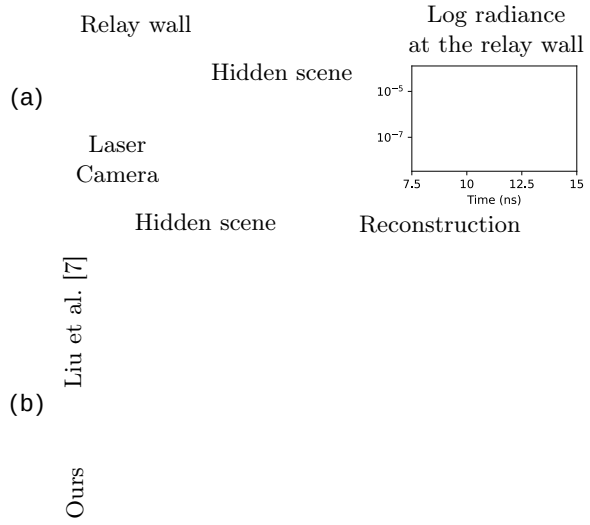

Figure 2: In order to showcase our system in a NLOS scenario, we have recreated the OFFICE scene from Liu et al. [7]. **(a)** The laser emits a pulse towards the relay wall, and the ultra-fast camera captures the response of the hidden scene. We show one pixel of this time-resolved response. **(b)** NLOS reconstructions of the captured and synthetic scenes.

have added custom sampling techniques tailored for NLOS scenes [1], resulting in an algorithm denoted as `transient_nlos_path`.

To showcase the potential of our work in this area, we have recreated the setup from Liu et al. [7], as seen in Fig. 2 **(a)**. The hidden scene consists of shelves, a wooden chair, and a cardboard box. Similar to the original experiment, we compute the impulse response of the hidden scene on a $\texttt{180} \times \texttt{130}$ grid of points on the relay wall, and as can be seen in Fig. 2 **(b)**, our system can be used for NLOS reconstructions.

Our sampling techniques greatly improve the convergence time: this experiment requires only four minutes of execution time on an Intel Xeon E5-2697 CPU using 500.000 samples per pixel, which would take hours otherwise. Nevertheless, using 5.000 samples (with four seconds of execution time) gives almost the same reconstruction results.

`y-tal`: A software toolkit for NLOS captures. We also make available a Python library and command-line utility, which offers a simple interface to interact with our system tailored for NLOS setups. It is publicly available on GitHub[3], and through PyPi[4].

References

[1] D. Royo, J. García, A. Muñoz, and A. Jarabo, "Non-line-of-sight transient rendering," *Computers & Graphics*, vol. 107, pp. 84–92, 2022.

[2] A. Jarabo, J. Marco, A. Muñoz, R. Buisan, W. Jarosz, and D. Gutierrez, "A framework for transient rendering," *ACM Trans. Graph.*, vol. 33, no. 6, 2014.

[3] P. Luesia, M. Crespo, A. Jarabo, and A. Redo-Sanchez, "Non-line-of-sight imaging in the presence of scattering media using phasor fields," *Opt. Lett.*, vol. 47, no. 15, pp. 3796–3799, Aug 2022.

[4] W. Jakob, S. Speierer, N. Roussel, and D. Vicini, "Dr.jit: A just-in-time compiler for differentiable rendering," *ACM Trans. Graph.*, vol. 41, no. 4, Jul. 2022.

[5] A. Velten, D. Wu, A. Jarabo, B. Masia, C. Barsi, C. Joshi, E. Lawson, M. Bawendi, D. Gutierrez, and R. Raskar, "Femto-photography: Capturing and visualizing the propagation of light," *ACM Trans. Graph.*, vol. 32, no. 4, 2013.

[6] Q. Hernandez, D. Gutierrez, and A. Jarabo, "A computational model of a single-photon avalanche diode sensor for transient imaging," 2017.

[7] X. Liu, I. Guillén, M. La Manna, J. H. Nam, S. A. Reza, T. H. Le, A. Jarabo, D. Gutierrez, and A. Velten, "Non-line-of-sight imaging using phasor fields virtual wave optics," *Nature*, 2019.

[3] `https://github.com/diegoroyo/tal/`
[4] `https://pypi.org/project/y-tal/`