LenslessPiCam: A Hardware and Software Platform for Lensless Computational Imaging with a Raspberry Pi

Eric Bezzam [1], **Sepand Kashani** [1], **Martin Vetterli** [1], **and Matthieu Simeoni** [1]

1 École Polytechnique Fédérale de Lausanne (EPFL)

DOI: 10.21105/joss.04747

Software

- Review
- Repository
- Archive

Editor: Dana Solav

Reviewers:

- @raolivei13
- @siddiquesalman

Submitted: 21 March 2022
Published: 26 June 2023

Summary

Lensless imaging seeks to replace or remove the lens in a conventional imaging system. The earliest cameras were in fact lensless, relying on long exposure times to form images on the other end of a small aperture in a darkened room/container (*camera obscura*). The introduction of a lens allowed for more light throughput and therefore shorter exposure times, while retaining sharp focus. The incorporation of digital sensors readily enabled the use of computational imaging techniques to post-process and enhance raw images (e.g. via deblurring, inpainting, denoising, sharpening). Recently, imaging scientists have started leveraging computational imaging as an integral part of lensless imaging systems, allowing them to form viewable images from the highly multiplexed raw measurements of lensless cameras (see (Boominathan et al., 2022) and references therein for a comprehensive treatment of lensless imaging). This represents a real paradigm shift in camera system design as there is more flexibility to cater the hardware to the application at hand (e.g., lightweight or flat designs). This increased flexibility comes, however, at the price of a more demanding post-processing of the raw digital recordings and a tighter integration of sensing and computation, often difficult to achieve in practice due to inefficient interactions between the various communities of scientists involved. With `LenslessPiCam`, we provide an easily accessible hardware and software framework to enable researchers, hobbyists, and students to implement and explore practical and computational aspects of lensless imaging.

Statement of need

Being at the interface of hardware, software, and algorithm design, the field of lensless imaging necessitates a broad array of competences that might deter newcomers to the field. The purpose of `LenslessPiCam` is to provide a complete toolkit with low-cost, accessible hardware designs and open-source software, to quickly enable the exploration of novel ideas for hardware, software, and algorithm design.

The DiffuserCam tutorial (Biscarrat et al., 2018) served as a great starting point to the present toolkit as it demonstrates that a working lensless camera can be built with cheap hardware: a Raspberry Pi, the Camera Module 2, and double-sided tape. The authors also provide Python implementations of two image reconstruction algorithms: variants of gradient descent (GD) with a non-negativity constraint; and the alternating direction method of multipliers (ADMM) (Boyd et al., 2011) with an additional total variation (TV) prior.

The resolution and quality of the reconstructed images for the DiffuserCam tutorial is poor and the processing pipeline is limited to grayscale reconstruction. With `LenslessPiCam`, we improve the reconstruction by using the newer HQ camera as well as a more versatile and generic RGB computational imaging pipeline. See Figure 1 for a comparison between the two cameras.

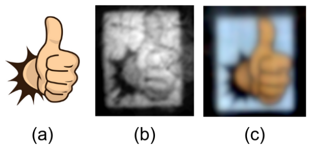

Figure 1: ADMM reconstruction of (a) an image of thumbs-up on a phone 40 cm away for (b) the original DiffuserCam tutorial (Biscarrat et al., 2018) and (c) our camera with RGB support.

Similar to (Biscarrat et al., 2018), the core image reconstruction functionality of `LenslessPiCam` depends on NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020). Moreover, `LenslessPiCam` also provides support for Pycsou (Simeoni, 2021), a universal and reusable software environment providing key computational imaging functionalities and tools with great modularity and interoperability. This results in a more flexible reconstruction workflow, allowing for the quick prototyping of advanced post-processing schemes with more sophisticated image priors. PyTorch (Paszke et al., 2017) support also enables the use of GPUs for faster reconstruction, and the use of deep learning for image reconstruction.

`LenslessPiCam` is designed to be used by researchers, hobbyists, and students. In the past, we have found such open-source hardware and software platforms to be a valuable resource for researchers (Bezzam et al., 2017) and students alike (Bezzam et al., 2019). Moreover, we have used `LenslessPiCam` in our own work for performing measurements, simulating data, and image reconstruction (Bezzam et al., 2022), and in our graduate-level signal processing course as a final project. Figure 2 demonstrates reconstructed images of our students using `LenslessPiCam` and images that were projected on a monitor 40cm away; the figure is adapted from this report.

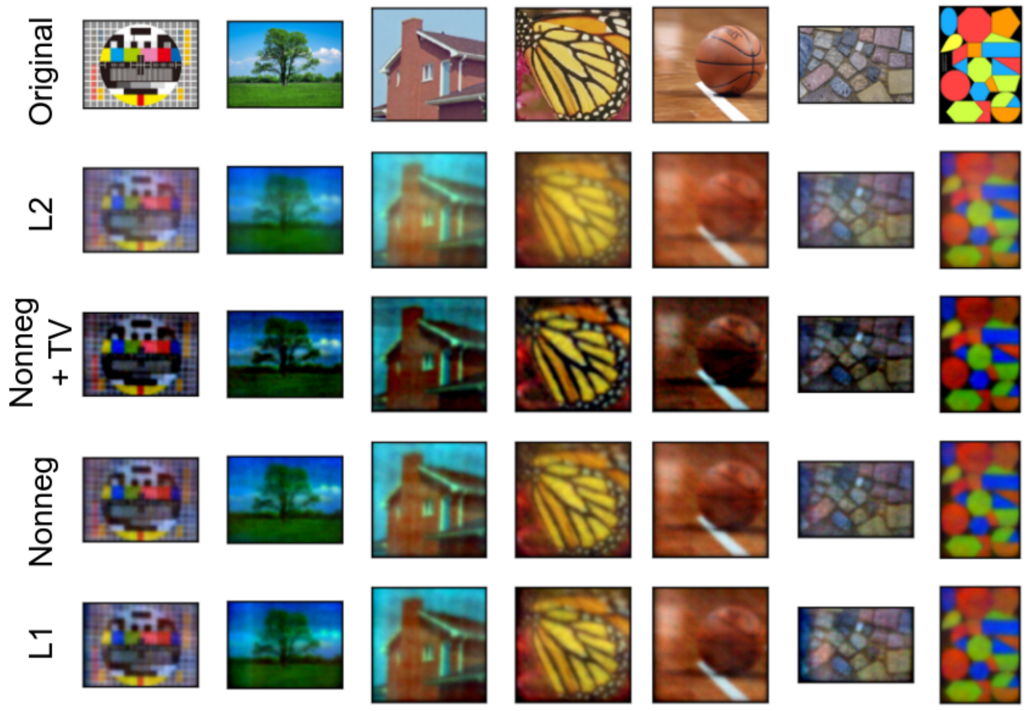

Figure 2: Reconstructions of images displayed on a monitor. "Original" is displayed on the monitor, and each subsequent row represents a reconstruction with (1) an L2 data fidelity between the measurement and the propagated image estimate (using a measured PSF) and (2) different priors/regularizers on the image estimate: L2 sparsity, non-negativity and total variation (TV), non-negativity, and L1 sparsity.

As opposed to Figure 1 and Figure 2, which show reconstructions of back-illuminated objects, Figure 3 demonstrates reconstructions of objects illuminated with an external source.

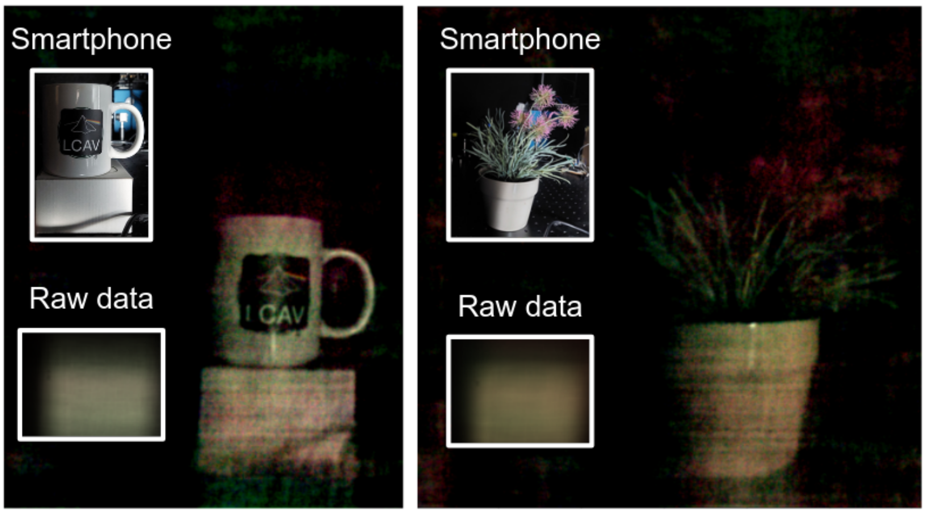

Figure 3: Reconstruction of objects 25cm away, illuminated with a lamp.

Figure 4 demonstrates the field-of-view (FOV) of the proposed system with double-sided tape. Objects are 40cm away.

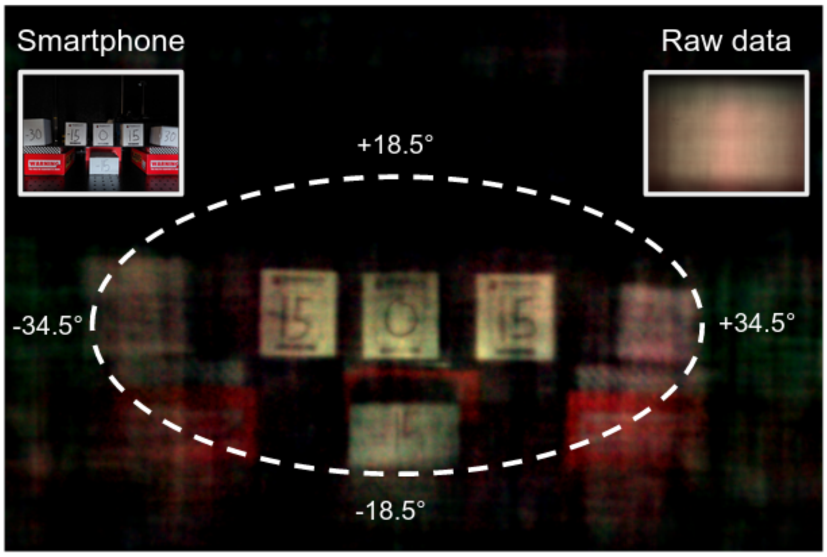

Figure 4: Experimental field-of-view (FOV) of tape-based LenslessPiCam.

Contributions

With respect to the DiffuserCam tutorial (Biscarrat et al., 2018), we have made the following contributions. In terms of hardware, as shown in Figure 5, we:

- make use of the HQ camera sensor (\$50): 4056 x 3040 pixels (12.3 MP) and 7.9 mm sensor diagonal, compared to 3280 × 2464 pixels (8.1 MP) and 4.6 mm sensor diagonal for the Camera Module 2 (\$30). A tutorial for building our proposed camera can be found on Medium;

- provide the design and firmware for a cheap point source generator (needed for calibration), which consists of an Arduino, a white LED, and a cardboard box. A tutorial for building this system can be found on Medium.

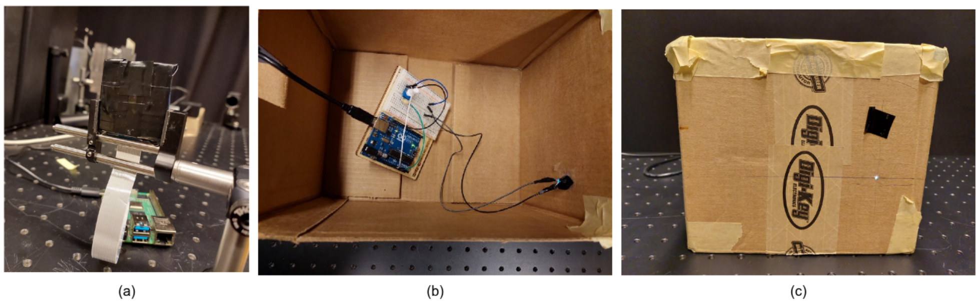

Figure 5: (a) LenslessPiCam, (b) point source generator (inside) and (c) (outside).

With respect to reconstruction algorithms, we:

- provide significantly faster implementations of GD and ADMM;
- extend the above reconstructions to RGB;
- provide PyTorch / GPU support;
- provide an object-oriented structure that is easy to extend for exploring new algorithms;
- provide an object-oriented interface to Pycsou for solving lensless imaging inverse problems. Pycsou is a Python package for solving inverse problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^N} F(\mathbf{y}, \mathbf{G}\mathbf{x}) \quad + \quad \lambda\mathcal{R}(\mathbf{x}), \tag{1}$$

  where F is a data-fidelity term between the observed and predicted measurements $\mathbf{y}$ and $\mathbf{G}\mathbf{x}$, respectively, $\mathcal{R}$ is a regularization component (could consist of more than one prior), and $\lambda > 0$ controls the amount of regularization.

We also provide functionalities to:

- remotely display data on an external monitor (as done for Figure 2), which can be used to automate raw data measurements to, e.g., gather a dataset;
- simulate measurements, given a point spread function (PSF) of a lensless camera. A tutorial can be found on Medium;
- evaluate reconstructions on a variety of metrics: mean squared error (MSE), peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS). A tutorial can be found on Medium;
- quantitatively evaluate the PSF of the lensless camera.

As previous noted, we have written a set of Medium articles to guide users through the process of building and using the proposed lensless camera. An overview of these articles can be found here. A ReadTheDocs page also provides an overview of all the available features.

In the following sections, we describe some of these contributions, and quantify them (where appropriate).

High-level, modular functionality for reconstructions

The core algorithmic component of `LenslessPiCam` is the abstract class `lensless.ReconstructionAlgori`
The three reconstruction strategies available in `LenslessPiCam` derive from this class:

- `lensless.GradientDescient`: projected GD with a non-negativity constraint. Two accelerated approaches are also available: `lensless.NesterovGradientDescent` (Nesterov, 1983) and `lensless.FISTA` (Beck & Teboulle, 2009);

- `lensless.ADMM`: ADMM with a non-negativity constraint and a TV regularizer;
- `lensless.APGD`: accelerated proximal GD with Pycsou as a backend. Any differentiable or proximal operator can be used as long as it is compatible with Pycsou, namely derives from one of `DiffFunc` or `ProxFunc`.

One major advantage of deriving from `lensless.ReconstructionAlgorithm` is that code duplication across algorithm can be minimized as it handles most of the common functionality, i.e., efficiently computing 2D Fourier transforms (needed to solve Equation 1), iteration logic, saving intermediate outputs, and visualization. Using a reconstruction algorithm that derives from it boils down to three steps:

1. Creating an instance of the reconstruction algorithm.
2. Setting the data.
3. Applying the algorithm.

For example, for ADMM (full example in `scripts/recon/admm.py`):

```
recon = ADMM(psf)
recon.set_data(data)
res = recon.apply(n_iter=n_iter)
```

Example reconstruction scripts can be found in `scripts/recon`. The Hydra framework (Yadan, 2019) is used to configure the various reconstruction algorithms, with the default configuration (`defaults_recon.yaml`) and others located in the `configs` folder.

Efficient reconstruction

In Table 1, we compare the processing time of DiffuserCam's and `LenslessPiCam`'s implementations for grayscale reconstruction of:

1. GD using FISTA with a non-negativity constraint.
2. ADMM with a non-negativity constraint and a TV regularizer.

The DiffuserCam implementations can be found here, while `lensless.APGD` and `lensless.ADMM` are used for `LenslessPiCam`. The comparison is done on a Dell Precision 5820 Tower X-Series (08B1) machine with an Intel i9-10900X 3.70 GHz processor (10 cores, 20 threads), running Ubuntu 20.04.5 LTS and (when applicable) an NVIDIA RTX A5000 GPU.

Table 1: Benchmark grayscale reconstruction. 300 iterations for gradient descent (GD) and 5 iterations for alternating direction method of multipliers (ADMM).

	GD	ADMM
DiffuserCam	246 s	6.81 s
`LenslessPiCam` (`numpy`)	21.1 s	1.26 s
`LenslessPiCam` (`torch`, CPU)	4.32 s	272 ms
`LenslessPiCam` (`torch`, GPU)	274 ms	2.88 ms

In Table 1, we observe an 11.7x reduction in computation time for GD and a 2.4x reduction for ADMM. This comes from:

- our object-oriented implementation of the algorithms, which allocates all the necessary memory beforehand and pre-computes data-independent terms, such as forward operators from the point spread function (PSF);
- our use of the real-valued fast Fourier transform (FFT), which is possible since we are working with image intensities. Our convolver/deconvolver is implemented as an object - `RealFFTConvolve2D` - that pre-computes the FFT of the PSF and supports SciPy and PyTorch backends.

When using a GPU (through PyTorch), we observe a significant reduction in computation time: 898x and 2360x reduction for GD and ADMM, respectively.

Quantifying performance

To methodically compare different reconstruction approaches, it is necessary to quantify the performance. To this end, `LenslessPiCam` provides functionality to extract regions of interest from the reconstruction and compare them with the original image via multiple metrics:

- Mean-squared error (MSE), where lower is better and the minimum value is 0;
- Peak signal-to-noise ratio (PSNR), where higher is better with values given in decibels (dB);
- Structural similarity index measure (SSIM), where values are within [-1, 1] and higher is better;
- Learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018a), where values are within [0, 1] and lower is better.

MSE, PSNR, and SSIM are computed using `skimage.metrics` (Van der Walt et al., 2014), while LPIPS is computed using `lpips` (Zhang et al., 2018b). MSE and PNSR compare images pixel-wise, while SSIM and LPIPS compare images patch-wise.

Figure 6 and Table 2 show how a reconstruction can be evaluated against an original image, using `scripts/compute_metrics_from_original.py`.

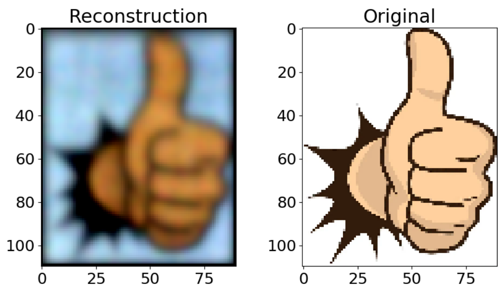

Figure 6: Comparing lensless reconstruction (left) with original image displayed on a screen (right).

Table 2: Metrics for Figure 6.

MSE	PSNR	SSIM	LPIPS
0.164	7.85	0.405	0.645

One limitation with comparing the reconstructed image (from measurements) directly with the original image is that the lighting during measurement can lead to rather poor results on the metrics, even though the content is visually similar (as in Figure 6). To mitigate this difference, we can compare the reconstructed image with the image displayed on the screen, but captured with a lensed camera. In the next section, we describe the functionalities `LenslessPiCam` provides for collecting such data and using existing datasets. Alternatively, simulation can be used to compare reconstruction algorithms without having to collect data.

Measured and simulated data

Sometimes it may be of interest to perform an exhaustive evaluation on a large dataset. `LenslessPiCam` could be used for collecting such a dataset with the proposed camera by using the remote display and capture scripts, i.e. `scripts/remote_display.py` and `scripts/remote_capture.py`, respectively.

Moreover, the authors of (Monakhova et al., 2019b) have already collected a dataset of 25,000 parallel measurements, namely 25,000 pairs of DiffuserCam and lensed camera images (Monakhova et al., 2019a). `LenslessPiCam` offers functionality to evaluate a reconstruction algorithm on the full dataset (100 GB), or a subset of 200 files (725 MB) that we have prepared. Note that this dataset is collected with a different lensless camera, but is nonetheless useful for exploring reconstruction techniques.

Table 3 shows the average metric results after applying 100 iterations of ADMM to the subset we have prepared, using `scripts/evaluate_mirflickr_admm.py`.

Table 3: Average metrics for 100 iterations of ADMM on a subset (200 files) of the DiffuserCam Lensless Mirflickr Dataset.

MSE	PSNR	SSIM	LPIPS
0.0797	12.7	0.535	0.585

One can also visualize the performance on a single file of the dataset, e.g., by using `scripts/apply_admm_single_mirflickr.py` to show how the reconstruction changes as the number of iterations increase.[^9] The final reconstruction and outputed metrics are shown in Figure 7 and Table 4.

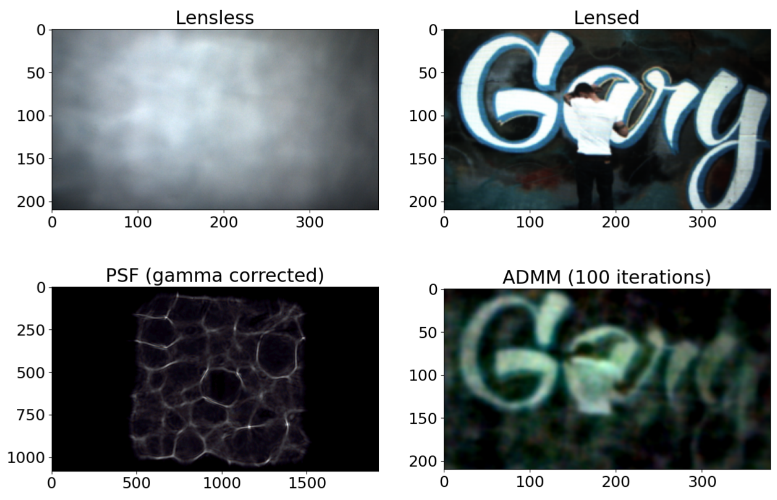

Figure 7: Visualizing performance of ADMM (100 iterations) on a single file of the DiffuserCam Lensless Mirflickr Dataset.

Table 4: Metrics for Figure 7.

MSE	PSNR	SSIM	LPIPS
0.0682	11.7	0.486	0.504

Scripts inside `scripts/sim` show how to simulate lensless camera measurements given a PSF, most notably with PyTorch compatibility for easy integration with machine learning tasks.

This approach can be used to conveniently compare reconstruction algorithms without having to collect data.

Conclusion

In summary, `LenslessPiCam` provides all the necessary hardware designs and software to build, use, and evaluate a lensless camera with low-cost and accessible components. Furthermore, a simulation framework allows users to experiment with lensless imaging without having to build the camera. As we continue to use `LenslessPiCam` as a research and educational platform, we hope to investigate and incorporate:

- computational refocusing and 3D imaging;
- video reconstruction;
- on-device reconstruction;
- programmable masks;
- data-driven, machine learning reconstruction techniques.

Acknowledgements and disclosure of funding

We acknowledge feedback from Julien Fageot and the students during the first iteration of this project in our graduate course.

This work was in part funded by the Swiss National Science Foundation (SNSF) under grants CRSII5 193826 "AstroSignals - A New Window on the Universe, with the New Generation of Large Radio-Astronomy Facilities" (M. Simeoni), 200 021 181 978/1 "SESAM - Sensing and Sampling: Theory and Algorithms" (E. Bezzam) and CRSII5 180232 "FemtoLippmann - Digital twin for multispectral imaging" (S. Kashani).

References

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202. https://doi.org/10.1137/080716542

Bezzam, E., Hoffet, A., & Prandoni, P. (2019). Teaching practical DSP with off-the-shelf hardware and free software. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7660–7664. https://doi.org/10.1109/ICASSP.2019.8682923

Bezzam, E., Scheibler, R., Azcarreta, J., Pan, H., Simeoni, M., Beuchat, R., Hurley, P., Bruneau, B., Ferry, C., & Kashani, S. (2017). Hardware and software for reproducible research in audio array signal processing. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6591–6592. https://doi.org/10.1109/ICASSP.2017.8005297

Bezzam, E., Vetterli, M., & Simeoni, M. (2022). Privacy-enhancing optical embeddings for lensless classification. *arXiv Preprint arXiv:2211.12864*.

Biscarrat, C., Parthasarathy, S., Kuo, G., & Antipa, N. (2018). *Build your own DiffuserCam: Tutorial*. https://waller-lab.github.io/DiffuserCam/tutorial.html

Boominathan, V., Robinson, J. T., Waller, L., & Veeraraghavan, A. (2022). Recent advances in lensless imaging. *Optica*, *9*(1), 1–16. https://doi.org/10.1364/OPTICA.431361

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, *3*(1), 1–122. https://doi.org/10.1561/2200000016

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

Monakhova, K., Yurtsever, J., Kuo, G., Antipa, N., Yanny, K., & Waller, L. (2019a). *DiffuserCam lensless Mirflickr dataset*. https://waller-lab.github.io/LenslessLearning/dataset.html

Monakhova, K., Yurtsever, J., Kuo, G., Antipa, N., Yanny, K., & Waller, L. (2019b). Learned reconstructions for practical mask-based lensless imaging. *Optics Express*, *27*(20), 28075. https://doi.org/10.1364/OE.27.028075

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. Akad. Nauk SSSR*, *269*, 543–547.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). *Automatic differentiation in pytorch*.

Simeoni, M. (2021). Pycsou. In *GitHub repository*. GitHub. https://github.com/matthieumeo/pycsou

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., & Yu, T. (2014). Scikit-image: Image processing in python. *PeerJ*, *2*, e453.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272.

Yadan, O. (2019). *Hydra - a framework for elegantly configuring complex applications*. Github. https://github.com/facebookresearch/hydra

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018a). The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 586–595. https://doi.org/10.1109/CVPR.2018.00068

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018b). *Perceptual similarity metric and dataset*. Github. https://github.com/richzhang/PerceptualSimilarity