The Journal of Open Source Software

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.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 21 March 2022
Published: unpublished

Summary

Lensless imaging seeks to replace/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. We also provide detailed guides and exercises so that `LenslessPiCam` can be used as an educational resource, and point to results from our graduate-level signal processing course.

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 cheap, accessible hardware designs and open-source software, that also achieves satisfactory results in order to explore novel ideas for hardware, software, and algorithm design.

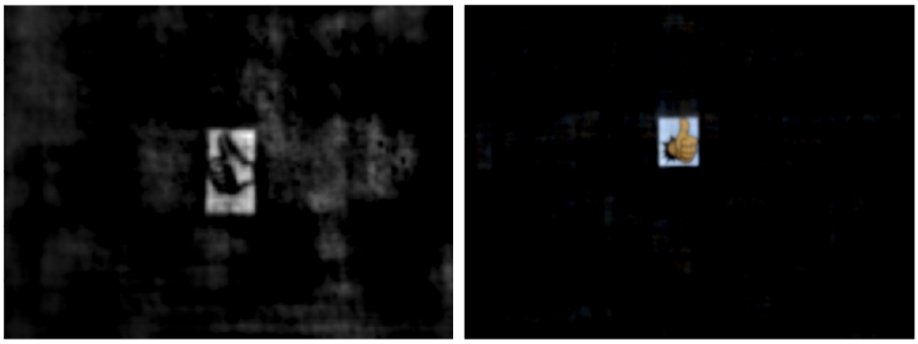
The DiffuserCam tutorial (Biscarrat et al., 2018) served as a great starting point to the present toolkit as it demonstrates that a working lensless camera can be built with cheap hardware: a Raspberry Pi, the Camera Module 2,[1] and a piece of tape. The authors also provide Python implementations of 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. Moreover, detailed guides explain how to build their camera and give intuition behind the reconstruction algorithms.

Unfortunately, the resolution of the reconstructed images is poor (see Figure 1a) and the

[1] www.raspberrypi.com/products/camera-module-v2.

processing pipeline is limited to grayscale reconstruction. With `LenslessPiCam`, we improve the resolution by using the newer HQ camera [2] as well as a more versatile and generic RGB computational imaging pipeline. The latter is built upon the Python library 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 and accurate reconstruction workflow, allowing for the quick prototyping of advanced post-processing schemes with more sophisticated image priors. See Figure 1b for an example image obtained with our lensless imaging framework.

`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).



(a) (b)

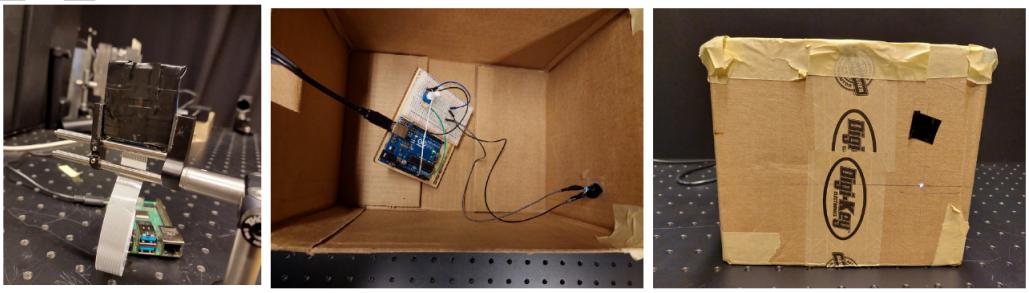
Figure 1: ADMM reconstruction of thumbs-up on a phone 40 cm away.

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 2, 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),
- 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) (b) (c)

Figure 2: (a) LenslessPiCam, (b) point source generator (inside), and (c) point source generator (outside).

[2] www.raspberrypi.com/products/raspberry-pi-high-quality-camera/.

With respect to reconstruction algorithms, we:

- provide significantly faster implementations of GD and ADMM, i.e. around 3x reduction in computation time,
- extend the above reconstructions to RGB,
- 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

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

  where F is a data-fidelity term between the observed and predicted measurements $\mathbf{y}$ and $\mathbf{G}\alpha$ 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 functionality to:

- remotely capture Bayer data with the proposed camera,
- convert Bayer data to RGB or grayscale,
- quantitavely evaluate the point spread function (PSF) of the lensless camera,
- remotely display data on an external monitor, which can be used to automate raw data measurements to, e.g., gather a dataset,
- evalute reconstructions on a variety of metrics: MSE, PSNR, SSIM, LPIPS (Zhang et al., 2018).

Finally, we have written a set of Medium articles to guide users through the process of building, using and/or teaching with the proposed lensless camera. An overview of these articles can be found here. The articles also include a set of solved exercises and problems for teaching purposes (solutions available to instructors on request).

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

High-level functionality

The core algorithmic component of `LenslessPiCam` is the abstract class `lensless.ReconstructionAlgorithm`. 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 `DifferentiableFunctional` or `ProximableFunctional`.

One advantage of deriving from `lensless.ReconstructionAlgorithm` is that functionality for iterating, saving, and visualization is already implemented. Consequently, 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/admm.py`):

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

A template for applying a reconstruction algorithm (including loading the data) can be found in `scripts/reconstruction_template.py`.

Efficient reconstruction

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

1. GD using FISTA and 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 Lenovo Thinkpad P15 with 16 GB RAM and a 2.70 GHz processor (6 cores, 12 threads), running Ubuntu 21.04.

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

	GD	ADMM
DiffuserCam	215 s	7.24 s
`LenslessPiCam`	67.9 s	2.76 s

From the above table, we observe a 3.1x reduction in computation time for GD and a 2.6x 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 FFT, which is possible since we are working with image intensities.

Figure 3 shows the corresponding grayscale reconstruction for FISTA and ADMM, which are equivalent for both DiffuserCam and `LenslessPiCam` implementations.

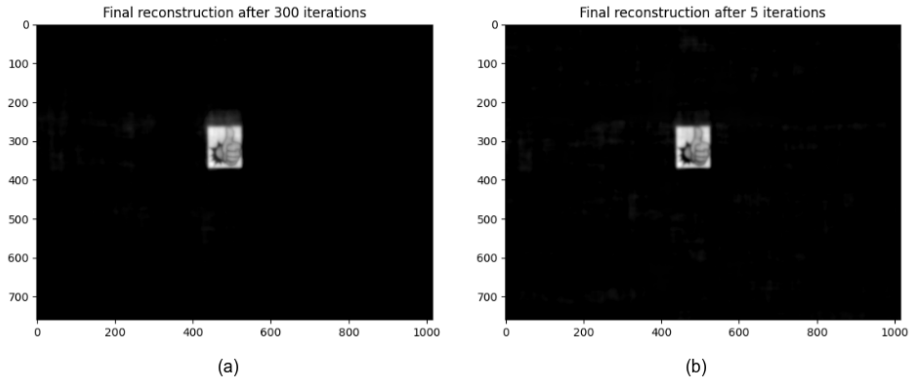


Figure 3: Grayscale reconstruction using FISTA (a) and ADMM (b).

Quantifying performance

In order 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),
- Peak signal-to-noise ratio (PSNR),
- Mean structural similarity (SSIM) index,
- Learned perceptual image patch similarity (LPIPS).

Below is an example of how a reconstruction can be evaluated against an original image.[3]

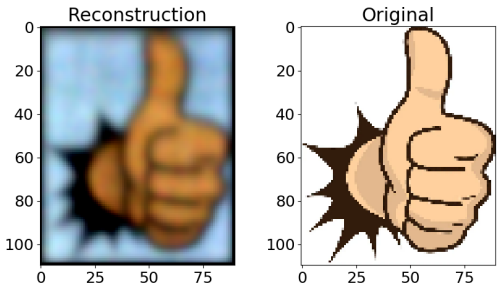


Figure 4: Extracting region from Figure 1b to quantify performance.

Table 2: Metrics for Figure 4.

MSE	PSNR	SSIM	LPIPS
0.164	7.85	0.405	0.645

Sometimes it may be of interest to perform an exhaustive evaluation on a large dataset. While `LenslessPiCam` could be used for collecting such a dataset with the proposed camera,[4] the authors of (Monakhova et al., 2019) have already collected a dataset of 25'000 parallel measurements, namely 25'000 pairs of DiffuserCam and lensed camera images.[5] `LenslessPiCam` offers functionality to evaluate a reconstruction algorithm on this dataset, or a subset of it that we have prepared.[6] 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.[7]

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, namely how the reconstruction changes as the number of iterations increase.[8] The final reconstruction and outputed metrics are shown in Figure 5 and Table 4.

[3] Using `scripts/compute_metrics_from_original.py`.

[4] Using the remote display and capture scripts, i.e. `scripts/remote_display.py` and `scripts/remote_capture.py` respectively.

[5] waller-lab.github.io/LenslessLearning/dataset.html.

[6] Subset of DiffuserCam Lensless Mirflickr Dataset consists of 200 files (725 MB) as opposed to 25'000 files (100 GB) of the original dataset. The subset can be downloaded here.

[7] Using `scripts/evaluate_mirflickr_admm.py`.

[8] Using `scripts/apply_admm_single_mirflickr.py`.

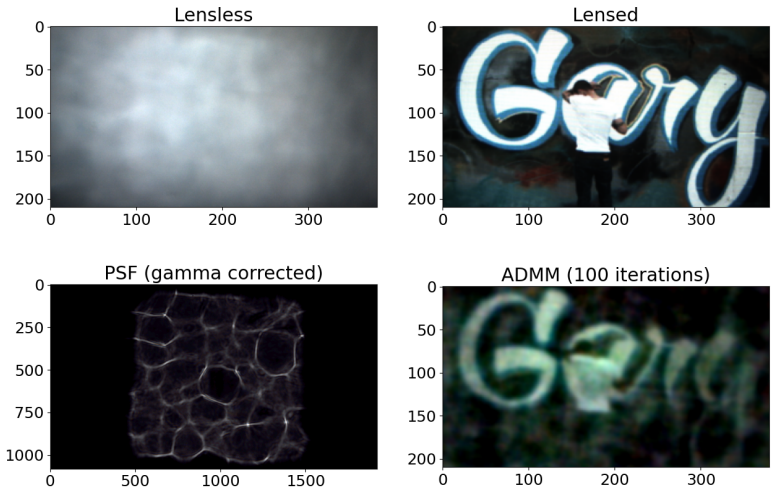


Figure 5: Visualizing performance of ADMM (100 iterations) on a single file of the DiffuserCam Lensless Mirflickr Dataset.

Table 4: Metrics for Figure 5.

MSE	PSNR	SSIM	LPIPS
0.0682	11.7	0.486	0.504

As an educational resource

As mentioned earlier, `LenslessPiCam` can serve as an educational resource. We have used it in our graduate-level signal processing course for providing experience in applying fundamental signal processing concepts and solving linear inverse problems. The work of our students can be found here.

As exercises in implementing key signal processing components, we have left some incomplete functions in `LenslessPiCam`:

- `lensless.autocorr.autocorr2d`: to compute a 2D autocorrelation in the frequency domain,
- `lensless.realfftconv.RealFFTConvolve2D`: to pre-compute the PSF's Fourier transform, perform a convolution in the frequency domain with the real-valued FFT, and vectorize operations for RGB.

We have also proposed a few reconstruction approaches to implement in this Medium article.

For the solutions to the above implementations, please request access to this folder detailing the intended use.

Conclusion

In summary, `LenslessPiCam` provides all the necessary hardware designs and software to build, use, and evaluate a lensless camera with cheap and accessible components. As we continue to use it as a research and educational platform, we hope to investigate and incorporate:

- computational refocusing,
- video reconstruction,
- on-device reconstruction,
- programmable masks,
- data-driven, machine learning reconstruction techniques.

Acknowledgements

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

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

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

Monakhova, K., Yurtsever, J., Kuo, G., Antipa, N., Yanny, K., & Waller, L. (2019). 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.

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

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018). 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