# SOLVING PTYCHOGRAPHY WITH A CONVEX RELAXATION: SUPPLEMENTARY MATERIAL

ROARKE HORSTMEYER, RICHARD Y. CHEN, XIAOZE OU, BRENDAN AMES, JOEL A. TROPP, AND CHANGHUEI YANG

## 1. The Burer–Monteiro Algorithm

As detailed in the main text, the Burer–Monteiro method, which defines our low-rank ptychography (LRP) solver, constructs the following augmented Lagrangian function:

$$L(\boldsymbol{R}, \boldsymbol{y}, \sigma) = \operatorname{tr}(\boldsymbol{R}\boldsymbol{R}^T) - \sum\nolimits_i y_i \cdot \big(\operatorname{tr}(\boldsymbol{D}_i \boldsymbol{R}\boldsymbol{R}^T) - b_i\big) + \frac{\sigma}{2} \cdot \sum\nolimits_i \big(\operatorname{tr}(\boldsymbol{D}_i \boldsymbol{R}\boldsymbol{R}^T) - b_i\big)^2, \tag{1.1}$$

where the variables $\boldsymbol{R} \in \mathbb{R}^{n \times r}$ and $\boldsymbol{y} \in \mathbb{R}^{q \cdot m}$ are unrestricted and the parameter $\sigma \in \mathbb{R}$ is positive. Each $b_i$ is the intensity measured by one image sensor pixel. The LRP solver then attempts to iteratively minimize the function $L$ by sequentially updating $\boldsymbol{R}$ and the parameters $\boldsymbol{y}$ and $\sigma$. We summarize this iterative update process in Algorithm 1.

**Algorithm 1** Burer–Monteiro Method (LRP)

**Require:** Algorithm parameters $\gamma > 1$ and $\eta < 1$, an auxiliary scalar $v_k$, iteration counter iter $= 0$, and the maximum number of iterations maxiter.

**Ensure:** A critical point

Initiate $\boldsymbol{y}^0$ and $\sigma_0$.
**repeat**
  Given $\boldsymbol{y}^k$ and $\sigma_k$, solve (1.1) to find the minimizer $\boldsymbol{R}_k$ via the LBFGS algorithm.
  Compute $v = \sum_i \big(\operatorname{tr}(\boldsymbol{D}_i \boldsymbol{R}_k \boldsymbol{R}_k^T) - b_i\big)^2$.
  **if** $v < \eta v_k$ **then**
    $y_i^{k+1} = y_i^k - \sigma_k \cdot \big(\operatorname{tr}(\boldsymbol{D}_i \boldsymbol{R}_k \boldsymbol{R}_k^T) - b_i\big)$ for all $i$,
    $\sigma_{k+1} = \sigma_k$,
    $v_{k+1} = v$;
  **else**
    $y_i^{k+1} = y_i^k$ for all $i$,
    $\sigma_{k+1} = \gamma \cdot \sigma_k$,
    $v_{k+1} = v_k$.
  iter = iter + 1
**until** iter = maxiter

## 2. Computational Cost

Here, we show the LRP algorithm's computational complexity for recovering 2D images is near-linear with image size. Suppose the desired complex image reconstruction contains $n$ pixels. For example, it might be a square image that is $\sqrt{n} \times \sqrt{n}$ pixels in each dimension. Furthermore, suppose the detected low-resolution magnitude-only images each contain $m$ pixels. Neighboring detected images overlap with one another in Fourier space. We assign the total number of pixels that each discretized Fourier spectrum overlaps with one of its immediate neighbors as $s = \alpha m$. We call $\alpha$ the overlap ratio between neighboring detected images. It is labeled as *ol* in the main text. Assuming both the low and high-resolution images are square for simplicity, the total number of detected images is given by,

$$L = \frac{n-m}{m-s} + 1 = \mathcal{O}\left(\frac{n}{(1-\alpha)m}\right).$$

where $\mathcal{O}$ denotes big-O notation.

The computational complexity of Algorithm 1 is dominated by Step 3: solving Eq. (1.1) to find the minimizer $\boldsymbol{R}_k$. We implement this step using the LBFGS algorithm. The main computational cost of Step 3 is the calculation of the augmented Lagrangian function gradient with respect to $\boldsymbol{R}$ in Eq. (1.1), which is

$$\mathcal{O}\big(L \cdot r \cdot m \log(\sqrt{m})\big) = \mathcal{O}\left(\frac{r \log(\sqrt{m})}{(1-\alpha)} \cdot n\right),$$

where $m \log(\sqrt{m})$ is the cost of a 2D fast Fourier transform applied to a square 2D image containing a total of $m$ pixels, $L$ is the number of low-resolution magnitude images, $r$ is the number of columns of the matrix $\boldsymbol{R}$. On the right hand side, we have substituted in for $L$ to find that each gradient calculation scales linearly with the number of pixels in the desired complex image reconstruction, $n$. The total computational cost is dominated by the cost of gradient evaluation multiplied by the number of gradient evaluations $C(n)$, which scales slowly as a function of $n$. Thus, the total computational cost of LRP is

$$\mathcal{O}\left(\frac{r \log(\sqrt{m})}{(1-\alpha)} \cdot n \cdot C(n)\right).$$

Experimentally, $C(n)$ scales approximately as $\log(n)$, thus justifying our claim that the computational complexity of the LRP algorithm is near-linear with the size of the reconstruction output.

The LRP algorithm is well-suited for implementation on a GPU. A number of key matrix operations in Step 3 of Algorithm 1 may be executed in parallel. Given the above computational analysis assumes a sequential solver, near-linear scaling of computational complexity with image size may be thought of as a worst-case scenario. In practice, we observe roughly sub-linear scaling when processing our experimental data on a GPU.

## 3. LRP algorithm parameters

The Burer–Monteiro method outlined in Algorithm 1 requires two parameters, $\gamma$ and $\eta$, which help guide its solution process. In most of the included experiments, we set $\gamma = 1.5$. The most observable consequence of selecting a different value of $\gamma$ is its influence on the number of required iterations for a desired level of performance. A larger value for $\gamma$ will cause a larger change within the augmented Lagrangian function each iteration, and thus a quicker progression to an optimized reconstruction.

Furthermore, we typically set $\eta = 0.5$. We observe that a different value of $\eta$ may also significantly alter the quality of reconstruction output with experimental input images (i.e., in the presence of noise). An example of this is shown in Fig. S1. Here, we use the same experimental setup that created the images in Fig. 6 in the main text of the manuscript. We capture 225 images of a stained red blood cell slide with varied LED illumination. Fig. S1 shows the resulting reconstructed phase using the AP (6 iterations) and LRP (15 iterations) algorithms. In this example, we set $\gamma = 1.5$ and $\eta = 0.5$ for the LRP algorithm, as usual. Note that the background of of both the AP and LRP reconstructions are not very uniform. However, close examination reveals that two adjacent red blood cells are completely "smoothed" together by the AP algorithm, while LRP is able to resolve each cell boundary (see insets). We note that a smaller value of $\eta$ (e.g., setting $\eta = 0.3$ and keeping all other algorithm parameters fixed) smooths out the LRP reconstructed phase such that it more closely resembles the smoother phase of AP-reconstructed cells. Unlike the AP result, the LRP background also remains uniform. In other words, $\eta$ appears to act as a global smoothing parameter, similar to the noise regularization variable $\lambda$ in our CLP program. AP completely lacks this type of parameter. Future work will examine how to select an optimal value of $\eta$ in LRP for a given background noise estimate.

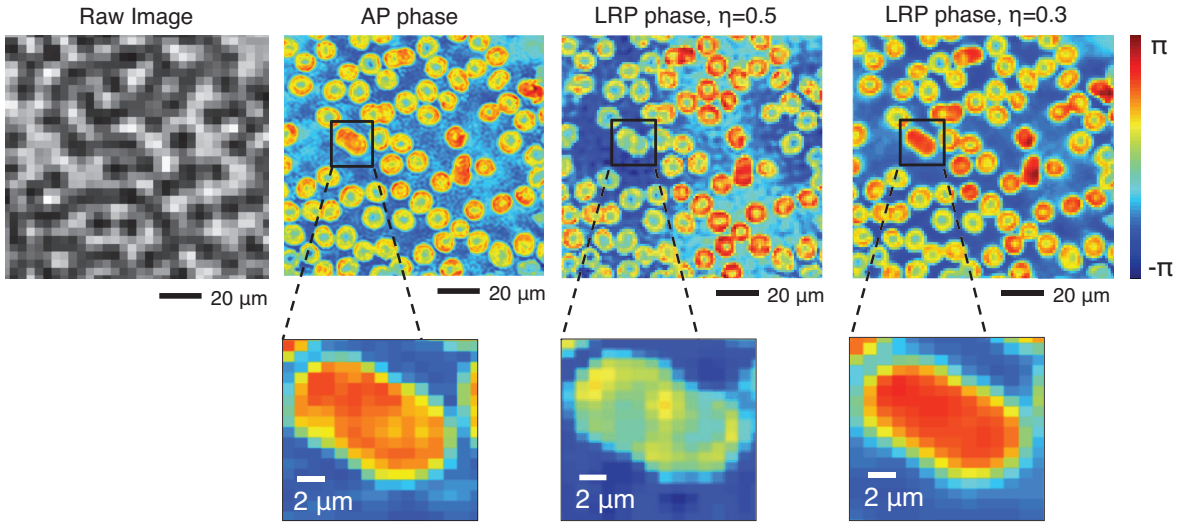


FIGURE S1. Comparing the reconstructed phase of red blood cells using the AP and LRP algorithm. Here, the LRP algorithm output exhibits a slightly non-uniform background when $\eta = 0.5$, but is able to resolve two adjacent cells (unlike the AP result). Changing $\eta$ to a smaller value (0.3) smooths out the LRP reconstruction to more closely resemble the AP reconstruction.