SCALABLE ROBUST MATRIX RECOVERY: FRANK-WOLFE MEETS PROXIMAL METHODS

CUN MU*, YUQIAN ZHANG†, JOHN WRIGHT†, AND DONALD GOLDFARB*

Abstract. Recovering matrices from compressive and grossly corrupted observations is a fundamental problem in robust statistics, with rich applications in computer vision and machine learning. In theory, under certain conditions, this problem can be solved in polynomial time via a natural convex relaxation, known as Compressive Principal Component Pursuit (CPCP). However, many existing provably convergent algorithms for CPCP suffer from superlinear per-iteration cost, which severely limits their applicability to large-scale problems. In this paper, we propose provably convergent, scalable and efficient methods to solve CPCP with (essentially) linear per-iteration cost. Our method combines classical ideas from Frank-Wolfe and proximal methods. In each iteration, we mainly exploit Frank-Wolfe to update the low-rank component with rank-one SVD and exploit the proximal step for the sparse term. Convergence results and implementation details are discussed. We demonstrate the practicability and scalability of our approach with numerical experiments on visual data.

Key words. robust matrix recovery, compressive principal component pursuit, Frank-Wolfe, conditional gradient, proximal methods, scalability

AMS subject classifications. 90C06, 90C25, 90C52

1. Introduction.

Suppose that a matrix $\boldsymbol{M}_0 \in \mathbb{R}^{m\times n}$ is of the form $\boldsymbol{M}_0 = \boldsymbol{L}_0 + \boldsymbol{S}_0 + \boldsymbol{N}_0$, where $\boldsymbol{L}_0$ is a low-rank matrix, $\boldsymbol{S}_0$ is a sparse error matrix, and $\boldsymbol{N}_0$ is a dense noise matrix. Linear measurements

$$\boldsymbol{b} \;=\; \mathcal{A}[\boldsymbol{M}_0] \;=\; \left(\langle \boldsymbol{A}_1, \boldsymbol{M}_0\rangle, \langle \boldsymbol{A}_2, \boldsymbol{M}_0\rangle, \ldots, \langle \boldsymbol{A}_p, \boldsymbol{M}_0\rangle\right)^\top \in \mathbb{R}^p \tag{1.1}$$

are collected, where $\mathcal{A}:\ \mathbb{R}^{m\times n} \to \mathbb{R}^p$ is the sensing operator, $\boldsymbol{A}_k$ is the sensing matrix for the k-th measurement and $\langle \boldsymbol{A}_k, \boldsymbol{M}_0\rangle \doteq \mathrm{Tr}(\boldsymbol{M}_0^\top \boldsymbol{A}_k)$. *Can we, in a tractable way, recover $\boldsymbol{L}_0$ and $\boldsymbol{S}_0$ from $\boldsymbol{b}$, given $\mathcal{A}$?*

One natural approach is to solve the optimization combining the fidelity term and the structural terms:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \;\; \frac{1}{2}\left\|\boldsymbol{b} - \mathcal{A}[\boldsymbol{L}+\boldsymbol{S}]\right\|_2^2 + \lambda_L \mathrm{rank}(\boldsymbol{L}) + \lambda_S \left\|\boldsymbol{S}\right\|_0. \tag{1.2}$$

Here, λ_L and λ_S are regularization parameters, and $\|\boldsymbol{S}\|_0$ denotes the number of nonzero entries in $\boldsymbol{S}$.

Unfortunately, problem (1.2) is nonconvex, and hence is not directly tractable. However, by replacing the ℓ_0 norm $\|\boldsymbol{S}\|_0$ with the ℓ_1 norm $\|\boldsymbol{S}\|_1 \doteq \sum_{i=1}^m \sum_{j=1}^n |S_{ij}|$, and replacing the rank $\mathrm{rank}(\boldsymbol{L})$ with the nuclear norm $\|\boldsymbol{L}\|_*$ (defined as the sum of the singular values of $\boldsymbol{L}$), we obtain a natural, tractable, convex relaxation of (1.2),

$$\min_{\boldsymbol{L},\boldsymbol{S}} \;\; \frac{1}{2}\left\|\boldsymbol{b} - \mathcal{A}[\boldsymbol{L}+\boldsymbol{S}]\right\|_2^2 + \lambda_L \left\|\boldsymbol{L}\right\|_* + \lambda_S \left\|\boldsymbol{S}\right\|_1. \tag{1.3}$$

This convex surrogate is sometimes referred to as *compressive principal component pursuit (CPCP)* [1]. Equivalently, since

$$\left\{\; \boldsymbol{M} \in \mathbb{R}^{m\times n} \;\mid\; \boldsymbol{b} = \mathcal{A}[\boldsymbol{M}] \;\right\} \;=\; \left\{\; \boldsymbol{M} \in \mathbb{R}^{m\times n} \;\mid\; \mathcal{P}_Q[\boldsymbol{M}] = \mathcal{P}_Q[\boldsymbol{M}_0] \;\right\},$$

*Department of Industrial Engineering and Operations Research, Columbia University (cm3052@columbia.edu, goldfarb@columbia.edu). DG was funded by NSF Grants DMS-1016571 and CCF-1527809.

†Department of Electrical Engineering, Columbia University, (yq2409@cs.columbia.edu, johnwright@ee.columbia.edu). JW was funded by ONR-N00014-13-0492.

where $\mathcal{Q} \subseteq \mathbb{R}^{m\times n}$ is a linear subspace spanned by the set of sensing matrices $\{\boldsymbol{A}_i\}_{i=1}^p$, and $\mathcal{P}_Q$ denotes the projection operator onto that subspace, we can rewrite problem (1.3) in the (possibly) more compact form, *

$$\min_{\boldsymbol{L},\boldsymbol{S}} \ f(\boldsymbol{L},\boldsymbol{S}) \doteq \frac{1}{2}\left\|\mathcal{P}_Q[\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}_0]\right\|_F^2 + \lambda_L \left\|\boldsymbol{L}\right\|_* + \lambda_S \left\|\boldsymbol{S}\right\|_1 . \tag{1.4}$$

Recently, CPCP and its close variants have been studied for different sensing operators $\mathcal{A}$ (or equivalently different subspaces $\mathcal{Q}$). In specific, [2, 3, 4, 5, 6] consider the case where a subset $\Omega \subseteq \{1,2,\ldots,m\}\times\{1,2,\ldots,n\}$ of the entries of $\boldsymbol{M}_0$ is observed. Then CPCP can be reduced to

$$\min_{\boldsymbol{L},\boldsymbol{S}} \quad \frac{1}{2}\left\|\mathcal{P}_\Omega[\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}_0]\right\|_F^2 + \lambda_L \left\|\boldsymbol{L}\right\|_* + \lambda_S \left\|\boldsymbol{S}\right\|_1 , \tag{1.5}$$

where $\mathcal{P}_\Omega[\cdot]$ denotes the orthogonal projection onto the linear space of matrices supported on Ω, i.e., $\mathcal{P}_\Omega[\boldsymbol{M}_0](i,j) = (\boldsymbol{M}_0)_{ij}$ if $(i,j)\in\Omega$ and $\mathcal{P}_\Omega[\boldsymbol{M}_0](i,j) = 0$ otherwise. [1] studies the case where each $\mathcal{A}_k$ is an i.i.d. $\mathcal{N}(0,1)$ matrix, which is equivalent (in distribution) to saying that we choose a linear subspace $\mathcal{Q}$ uniformly at random from the set of all p-dimensional subspaces of $\mathbb{R}^{m\times n}$ and observe $\mathcal{P}_Q[\boldsymbol{M}_0]$. Accordingly, all the above provide theoretical guarantees for CPCP, under fairly mild conditions, to produce accurate estimates of $\boldsymbol{L}_0$ and $\mathcal{P}_\Omega[\boldsymbol{S}_0]$ (or $\boldsymbol{S}_0$), even when the number of measurements p is substantially less than mn.

Inspired by these theoretical results, researchers from different fields have leveraged CPCP to solve many practical problems, including video background modeling [3], batch image alignment [7], face verification [8], photometric stereo [9], dynamic MRI [10], topic modeling [11], latent variable graphical model learning [12] and outlier detection and robust Principal Component Analysis [3], to name just a few.

Living in the era of *big data*, most of these applications involve large datasets and high dimensional data spaces. Therefore, to fully realize the benefit of the theory, we need *provably convergent* and *scalable* algorithms for CPCP. This has motivated much research into the development of first-order methods for problem (1.4) and its variants; e.g see [13, 14, 15, 16, 17, 18]. These methods, in essence, all exploit a closed-form expression for the proximal operator of the nuclear norm, which involves the singular value decompsition (SVD). Hence, the dominant cost in each iteration is computing an SVD of the same size as the input data. This is substantially more scalable than off-the-shelf interior point solvers such as SDPT3 [19]. Nevertheless, the superlinear cost of each iteration has limited the practical applicability of these first-order methods to problems involving several thousands of data points and several thousands of dimensions. The need to compute a sequence of full or partial SVDs is a serious bottleneck for truly large-scale applications.

As a remedy, in this paper, we design more scalable algorithms to solve CPCP that compute only a rank-one SVD in each iteration. Our approach leverages two classical and widely studied ideas – Frank-Wolfe iterations to handle the nuclear norm, and proximal steps to handle the ℓ_1 norm. This turns out to be an ideal combination of techniques to solve large-scale CPCP problems. In particular, it yields algorithms that

*To transform problem (1.3) into problem (1.4), simple procedures like Gram-chmidt might be invoked. Despite being equivalent, one formulation might be preferred over the other in practice, depending on the specifications of the sensing operator $\mathcal{A}[\cdot]$. In this paper, we will mainly focus on solving problem (1.4) and its variants. Our methods, however, are not restrictive to (1.4) and can be easily extended to problem (1.3).

are substantially *more scalable* than prox-based first-order methods such as ISTA and FISTA [20], and converge *much faster* in practice than a straightforward application of Frank-Wolfe.

The remainder of this paper is organized as follows. Section 2 reviews the general properties of the Frank-Wolfe algorithm, and describes several basic building blocks that we will use in our algorithms. Section 3 and Section 4 respectively describe how to modify the Frank-Wolfe algorithm to solve CPCP's *norm constrained* version

$$\min_{\boldsymbol{L},\boldsymbol{S}} \; l(\boldsymbol{L},\boldsymbol{S}) \doteq \frac{1}{2}\left\|\mathcal{P}_Q[\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}_0]\right\|_F^2 \quad \text{s.t. } \|\boldsymbol{L}\|_* \le \tau_L,\ \|\boldsymbol{S}\|_1 \le \tau_S, \tag{1.6}$$

and the penalized version, i.e. problem (1.4), by incorporating proximal regularization to more effectively handle the ℓ_1 norm. Convergence results and our implementation details are also discussed. Section 5 presents numerical experiments on large datasets that demonstrate the scalability of our proposed algorithms. In Section 6, we summarize our contributions and discuss potential future works.

2. Preliminaries.

2.1. Frank-Wolfe method. The Frank-Wolfe (FW) method [21], also known as the conditional gradient method [22], applies to the general problem of minimizing a differentiable convex function h over a compact, convex domain $\mathcal{D} \subseteq \mathbb{R}^n$:

$$\text{minimize} \quad h(\boldsymbol{x}) \qquad \text{subject to} \quad \boldsymbol{x} \in \mathcal{D} \subseteq \mathbb{R}^n. \tag{2.1}$$

Here, ∇h is assumed to be L-Lipschitz:

$$\forall\, \boldsymbol{x},\, \boldsymbol{y} \in \mathcal{D}, \qquad \|\nabla h(\boldsymbol{x}) - \nabla h(\boldsymbol{y})\| \le L\,\|\boldsymbol{x}-\boldsymbol{y}\|\,. \tag{2.2}$$

Throughout, we let $D = \max_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{D}} \|\boldsymbol{x}-\boldsymbol{y}\|$ denote the diameter of the feasible set $\mathcal{D}$.

In its simplest form, the Frank-Wolfe algorithm proceeds as follows. At each iteration k, we linearize the objective function h about the current point $\boldsymbol{x}^k$:

$$h(\boldsymbol{v}) \approx h(\boldsymbol{x}^k) + \left\langle \nabla h(\boldsymbol{x}^k), \boldsymbol{v}-\boldsymbol{x}^k \right\rangle. \tag{2.3}$$

We minimize the linearization over the feasible set $\mathcal{D}$ to obtain

$$\boldsymbol{v}^k \in \arg\min_{\boldsymbol{v}\in\mathcal{D}} \left\langle \nabla h(\boldsymbol{x}^k), \boldsymbol{v} \right\rangle, \tag{2.4}$$

and then take a step in the feasible descent direction $\boldsymbol{v}^k - \boldsymbol{x}^k$:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \frac{2}{k+2}(\boldsymbol{v}^k - \boldsymbol{x}^k). \tag{2.5}$$

This yields a very simple procedure, which we summarize as Algorithm 1. The particular step size, $\frac{2}{k+2}$, comes from the convergence analysis of the algorithm, which we discuss in more details below.

First proposed in [21], FW-type methods have been frequently revisited in different fields. Recently, they have experienced a resurgence in statistics, machine learning and signal processing, due to their ability to yield highly scalable algorithms for optimization with structure-encouraging norms such as the ℓ_1 norm and nuclear norm. In particular, if $\boldsymbol{x}$ is a matrix and $\mathcal{D} = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_* \le \beta\}$ is a nuclear norm ball, the subproblem

$$\min_{\boldsymbol{v}\in\mathcal{D}} \; \langle \boldsymbol{v}, \nabla h(\boldsymbol{x}) \rangle \tag{2.6}$$

Algorithm 1 Frank-Wolfe method for problem (2.1)

Initialization: $\boldsymbol{x}^0 \in \mathcal{D}$;
for $k = 0,\ 1,\ 2,\ \ldots$ **do**
  $\boldsymbol{v}^k \in \operatorname{argmin}_{\boldsymbol{v}\in\mathcal{D}} \left\langle \boldsymbol{v}, \nabla h(\boldsymbol{x}^k)\right\rangle$;
  $\gamma = \frac{2}{k+2}$;
  $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \gamma(\boldsymbol{v}^k - \boldsymbol{x}^k)$;
end for

can be solved using only the singular vector pair corresponding to the single leading singular value of the matrix $\nabla h(\boldsymbol{x})$. Thus, at each iteration, we only have to compute a rank-one partial SVD. This is substantially cheaper than the full/partial SVD exploited in proximal methods [23, 24]. We recommend [25] as a comprehensive survey of the latest developments in FW-type methods.

Algorithm 2 Frank-Wolfe method for problem (2.1) with general updating scheme

Initialization: $\boldsymbol{x}^0 \in \mathcal{D}$;
for $k = 0,\ 1,\ 2,\ \ldots$ **do**
  $\boldsymbol{v}^k \in \operatorname{argmin}_{\boldsymbol{v}\in\mathcal{D}} \left\langle \boldsymbol{v}, \nabla h(\boldsymbol{x}^k)\right\rangle$;
  $\gamma = \frac{2}{k+2}$;
  Update $\boldsymbol{x}^{k+1}$ to some point in $\mathcal{D}$ such that $h(\boldsymbol{x}^{k+1}) \leq h(\boldsymbol{x}^k + \gamma(\boldsymbol{v}^k - \boldsymbol{x}^k))$;
end for

In the past five decades, numerous variants of Algorithm 1 have been proposed and implemented. Many modify Algorithm 1 by replacing the simple updating rule (2.5) with more sophisticated schemes, e.g.,

$$\boldsymbol{x}^{k+1} \ \in\ \arg\min_{\boldsymbol{x}}\ h(\boldsymbol{x}) \quad \text{s.t. } \boldsymbol{x} \in \operatorname{conv}\{\boldsymbol{x}^k,\ \boldsymbol{v}^k\} \tag{2.7}$$

or

$$\boldsymbol{x}^{k+1} \ \in\ \arg\min_{\boldsymbol{x}}\ h(\boldsymbol{x}) \quad \text{s.t. } \boldsymbol{x} \in \operatorname{conv}\{\boldsymbol{x}^k,\ \boldsymbol{v}^k,\ \boldsymbol{v}^{k-1},\ \ldots,\ \boldsymbol{v}^{k-j}\}. \tag{2.8}$$

The convergence of these schemes can be analyzed simultaneously, using the fact that they produce iterates $\boldsymbol{x}^{k+1}$ whose objective is no greater than that produced by the original Frank-Wolfe update scheme:

$$h(\boldsymbol{x}^{k+1}) \leq h(\boldsymbol{x}^k + \gamma(\boldsymbol{v}^k - \boldsymbol{x}^k)).$$

Algorithm 2 states a general version of Frank-Wolfe, whose update is only required to satisfy this relationship. It includes as special cases the updating rules (2.5), (2.7) and (2.8). This flexibility will be crucial for effectively handling the sparse structure in the CPCP problems (1.4) and (1.6).

The convergence of Algorithm 2 can be proved using well-established techniques [24, 25, 26, 27, 28, 29, 30, 31]. Using these ideas, one can show that it converges at a rate of $O(1/k)$ in function value:

THEOREM 2.1. *Let $\boldsymbol{x}^\star$ be an optimal solution to* (2.1). *For $\{\boldsymbol{x}^k\}$ generated by Algorithm* 2, *we have for $k = 0,\ 1,\ 2,\ \ldots,$*

$$h(\boldsymbol{x}^k) - h(\boldsymbol{x}^\star) \leq \frac{2LD^2}{k+2}. \tag{2.9}$$

Proof. For $k = 0,\ 1,\ 2,\ \ldots$, we have

$$\begin{aligned} h(\boldsymbol{x}^{k+1}) &\leq h(\boldsymbol{x}^k + \gamma(\boldsymbol{v}^k - \boldsymbol{x}^k)) \\ &\leq h(\boldsymbol{x}^k) + \gamma \left\langle \nabla h(\boldsymbol{x}^k), \boldsymbol{v}^k - \boldsymbol{x}^k \right\rangle + \frac{L\gamma^2}{2} \left\| \boldsymbol{v}^k - \boldsymbol{x}^k \right\|^2 \\ &\leq h(\boldsymbol{x}^k) + \gamma \left\langle \nabla h(\boldsymbol{x}^k), \boldsymbol{v}^k - \boldsymbol{x}^k \right\rangle + \frac{\gamma^2 L D^2}{2} \qquad (2.10) \\ &\leq h(\boldsymbol{x}^k) + \gamma \left\langle \nabla h(\boldsymbol{x}^k), \boldsymbol{x}^\star - \boldsymbol{x}^k \right\rangle + \frac{\gamma^2 L D^2}{2} \\ &\leq h(\boldsymbol{x}^k) + \gamma (h(\boldsymbol{x}^\star) - h(\boldsymbol{x}^k)) + \frac{\gamma^2 L D^2}{2}, \qquad (2.11) \end{aligned}$$

where the second inequality holds since $\nabla h(\cdot)$ is L-Lipschitz continuous; the third line follows because D is the diameter for the feasible set $\mathcal{D}$; the fourth inequality follows from $\boldsymbol{v}^k \in \operatorname{argmin}_{\boldsymbol{v} \in \mathcal{D}} \left\langle \boldsymbol{v}, \nabla h(\boldsymbol{x}^k) \right\rangle$ and $\boldsymbol{x}^\star \in \mathcal{D}$; the last one holds since $h(\cdot)$ is convex.

Rearranging terms in (2.11), one obtains that for $k = 0,\ 1,\ 2,\ \ldots$,

$$h(\boldsymbol{x}^{k+1}) - h(\boldsymbol{x}^\star) \leq (1-\gamma) \left(h(\boldsymbol{x}^k) - h(\boldsymbol{x}^\star) \right) + \frac{\gamma^2 L D^2}{2}. \qquad (2.12)$$

Therefore, by mathematical induction, it can be verified that

$$h(\boldsymbol{x}^k) - h(\boldsymbol{x}^\star) \leq \frac{2LD^2}{k+2}, \quad \text{for } \ k = 1,\ 2,\ 3,\ \ldots.$$

□

REMARK 1. *Note that the constant in the rate of convergence depends on the Lipschitz constant L of h and the diameter $\mathcal{D}$.*

While Theorem 2.1 guarantees that Algorithm 2 converges at a rate of $O(1/k)$, in practice it is useful to have a more precise bound on the suboptimality at iterate k. The surrogate duality gap

$$d(\boldsymbol{x}^k) = \left\langle \boldsymbol{x}^k - \boldsymbol{v}^k, \nabla h(\boldsymbol{x}^k) \right\rangle, \qquad (2.13)$$

provides a useful upper bound on the suboptimality $h(\boldsymbol{x}^k) - h(\boldsymbol{x}^\star)$:

$$\begin{aligned} h(\boldsymbol{x}^k) - h(\boldsymbol{x}^\star) &\leq - \left\langle \boldsymbol{x}^\star - \boldsymbol{x}^k, \nabla h(\boldsymbol{x}^k) \right\rangle \\ &\leq - \min_{\boldsymbol{v}} \left\langle \boldsymbol{v} - \boldsymbol{x}^k, \nabla h(\boldsymbol{x}^k) \right\rangle = \left\langle \boldsymbol{x}^k - \boldsymbol{v}^k, \nabla h(\boldsymbol{x}^k) \right\rangle = d(\boldsymbol{x}^k). \qquad (2.14) \end{aligned}$$

This was first proposed in [21] and later [25] showed that $d(\boldsymbol{x}^k) = O(1/k)$. Next, we provide a refinement of this result, using ideas from [25, 30]:

THEOREM 2.2. *Let $\{\boldsymbol{x}^k\}$ be the sequence generated by Algorithm 2. Then for any $K \geq 1$, there exists $1 \leq \tilde{k} \leq K$ such that*

$$d(\boldsymbol{x}^{\tilde{k}}) \leq \frac{6LD^2}{K+2}. \qquad (2.15)$$

Proof. For notational convenience, we denote $h^k \doteq h(\boldsymbol{x}^k)$, $\Delta^k \doteq h(\boldsymbol{x}^k) - h(\boldsymbol{x}^\star)$, $d^k \doteq d(\boldsymbol{x}^k)$, $C \doteq 2LD^2$, $B \doteq K+2$, $\hat{k} \doteq \lceil \frac{1}{2} B \rceil - 1$, $\mu \doteq \lceil \frac{1}{2} B \rceil / B$.

Suppose on the contrary that

$$d^k > \frac{3C}{B}, \quad \text{for all } k \in \left\{ \lceil \frac{1}{2}B \rceil - 1,\ \lceil \frac{1}{2}B \rceil,\ \ldots,\ K \right\}. \tag{2.16}$$

From (2.10), we know that for any $k \geq 1$

$$\Delta^{k+1} \leq \Delta^k + \gamma \left\langle \nabla h(\boldsymbol{x}^k), \boldsymbol{v}^k - \boldsymbol{x}^k \right\rangle + \frac{\gamma^2 L D^2}{2} = \Delta^k - \frac{2d^k}{k+2} + \frac{C}{(k+2)^2}. \tag{2.17}$$

Therefore, by using (2.17) repeatedly, one has

$$\begin{aligned}
\Delta^{K+1} &\leq \Delta^{\hat{k}} - \sum_{k=\hat{k}}^{K} \frac{2d^k}{k+2} + \sum_{k=\hat{k}}^{K} \frac{C}{(k+2)^2} \\
&< \Delta^{\hat{k}} - \frac{6C}{B} \sum_{k=\hat{k}}^{K} \frac{1}{k+2} + C \sum_{k=\hat{k}}^{K} \frac{1}{(k+2)^2} \\
&= \Delta^{\hat{k}} - \frac{6C}{B} \sum_{k=\hat{k}+2}^{B} \frac{1}{k} + C \sum_{k=\hat{k}+2}^{B} \frac{1}{k^2} \\
&\leq \frac{C}{\mu B} - \frac{6C}{B} \cdot \frac{B - \hat{k} - 1}{B} + C \cdot \frac{B - \hat{k} - 1}{B(\hat{k}+1)} \\
&= \frac{C}{\mu B} - \frac{6C}{B}(1-\mu) + \frac{C}{B}\frac{1-\mu}{\mu} \\
&= \frac{C}{\mu B}\left(2 - 6\mu(1-\mu) - \mu\right) \qquad (2.18)
\end{aligned}$$

where the second line is due to our assumption (2.16); the fourth line holds since $\Delta^{\hat{k}} \leq \frac{C}{\hat{k}+2}$ by Theorem 1, and $\sum_{k=a}^{b} \frac{1}{k^2} \leq \frac{b-a+1}{b(a-1)}$ for any $b \geq a > 1$.

Now define $\phi(x) = 2 - 6x(1-x) - x$. Clearly $\phi(\cdot)$ is convex. Since $\phi(\frac{1}{2}) = \phi(\frac{2}{3}) = 0$, we have $\phi(x) \leq 0$ for any $x \in [\frac{1}{2}, \frac{2}{3}]$. As $\mu = \lceil \frac{1}{2}B \rceil / B \in [\frac{1}{2}, \frac{2}{3}]$, from (2.18), we have

$$\Delta^{K+1} = h(\boldsymbol{x}^{K+1}) - h(\boldsymbol{x}^\star) < \frac{C}{\mu B}\phi(\mu) \leq 0,$$

which is a contradiction. □

REMARK 2. *The convergence rate for the duality gap matches the one for $h(\boldsymbol{x}^k) - h(\boldsymbol{x}^\star)$ (see (2.9)), which suggests that the upper bound $d(\boldsymbol{x}^k)$ can serve as a practical stopping criterion.*

For our problem, the main computational burden in Algorithms 1 and 2 will be solving the linear subproblem $\min_{\boldsymbol{v} \in \mathcal{D}} \left\langle \boldsymbol{v}, \nabla h(\boldsymbol{x}^k) \right\rangle$, [†] i.e. minimizing linear functions over the unit balls for $\|\cdot\|_*$ and $\|\cdot\|_1$. Fortunately, both of these operations have simple closed-form solutions, which we will describe in the next section.

2.2. Optimization oracles. We now describe several optimization oracles involving the ℓ_1 norm and the nuclear norm, which serve as the main building blocks for our methods. These oracles have computational costs that are (essentially) linear in the size of the input.

[†] In some situations, we can significantly reduce this cost by solving this problem inexactly [27, 25]. Our algorithms and results can also tolerate inexact step calculations; we omit the discussion here for simplicity.

Minimizing a linear function over the nuclear norm ball. Since the dual norm of the nuclear norm is the operator norm, i.e., $\|\boldsymbol{Y}\| = \max_{\|\boldsymbol{X}\|_* \leq 1} \langle \boldsymbol{Y}, \boldsymbol{X} \rangle$, the optimization problem

$$\text{minimize}_{\boldsymbol{X}} \ \langle \boldsymbol{Y}, \boldsymbol{X} \rangle \qquad \text{subject to } \|\boldsymbol{X}\|_* \leq 1 \tag{2.19}$$

has optimal value $-\|\boldsymbol{Y}\|$. One minimizer is the rank-one matrix $\boldsymbol{X}^\star = -\boldsymbol{u}\boldsymbol{v}^\top$, where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the left- and right- singular vectors corresponding to the leading singular value of $\boldsymbol{Y}$, and can be efficiently computed (e.g. using power method).

Minimizing a linear function over the ℓ_1 ball. Since the dual norm of the ℓ_1 norm is the ℓ_∞ norm, i.e., $\|\boldsymbol{Y}\|_\infty := \max_{(i,j)} |Y_{ij}| = \max_{\|\boldsymbol{X}\|_1 \leq 1} \langle \boldsymbol{Y}, \boldsymbol{X} \rangle$, the optimization problem

$$\text{minimize}_{\boldsymbol{X}} \ \langle \boldsymbol{Y}, \boldsymbol{X} \rangle \qquad \text{subject to } \|\boldsymbol{X}\|_1 \leq 1 \tag{2.20}$$

has optimal value $-\|\boldsymbol{Y}\|_\infty$. One minimizer is the one-sparse matrix

$$\boldsymbol{X}^\star = -\text{sgn}(Y_{i^\star j^\star}) \boldsymbol{e}_{i^\star} \boldsymbol{e}_{j^\star}^\top,$$

where $(i^\star, j^\star) \in \arg\max_{(i,j)} |Y_{ij}|$; i.e. $\boldsymbol{X}^\star$ has exactly one nonzero element.

Projection onto the ℓ_1-ball. To effectively handle the sparse term in the norm constrained problem (1.6), we will need to modify the Frank-Wolfe algorithm by incorporating additional projection steps. For any $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$ and $\beta > 0$, the projection onto the ℓ_1-ball:

$$\mathcal{P}_{\|\cdot\|_1 \leq \beta}[\boldsymbol{Y}] = \arg\min_{\|\boldsymbol{X}\|_1 \leq \beta} \ \frac{1}{2} \|\boldsymbol{X} - \boldsymbol{Y}\|_F^2, \tag{2.21}$$

can be easily solved with $O(mn(\log m + \log n))$ cost [32]. Moreover, a divide and conquer algorithm, achieving linear cost in expectation to solve (2.21), has also been proposed in [32].

Proximal mapping of ℓ_1 norm. To effectively handle the sparse term arising in problem (1.4), we will need to modify the Frank-Wolfe algorithm by incorporating additional proximal steps. For any $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$ and $\lambda > 0$, the proximal mapping of ℓ_1 norm has the following closed-form expression

$$\mathcal{T}_\lambda[\boldsymbol{Y}] \ = \ \arg\min_{\boldsymbol{X} \in \mathbb{R}^{m \times n}} \ \frac{1}{2} \|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \lambda \|\boldsymbol{X}\|_1, \tag{2.22}$$

where $\mathcal{T}_\lambda : \mathbb{R} \to \mathbb{R}$ denotes the soft-thresholding operator $\mathcal{T}_\lambda(x) = \text{sgn}(x) \max\{|x| - \lambda, 0\}$, and extension to matrices is obtained by applying the scalar operator $\mathcal{T}_\lambda(\cdot)$ to each element.

3. FW-P Method for Norm Constrained Problem. In this section, we develop scalable algorithms for the norm-constrained compressive principal component pursuit problem,

$$\min_{\boldsymbol{L}, \boldsymbol{S}} \ l(\boldsymbol{L}, \boldsymbol{S}) = \frac{1}{2} \|\mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}]\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{L}\|_* \leq \tau_L, \ \|\boldsymbol{S}\|_1 \leq \tau_S. \tag{3.1}$$

We first describe a straightforward application of the Frank-Wolfe method to this problem. We will see that although it has relatively cheap iterations, it converges very slowly on typical numerical examples, because it only makes a one-sparse update to the sparse term $\boldsymbol{S}$ at a time. We will show how to remedy this problem by augmenting the FW iteration with an additional proximal step (essentially a projected gradient step) in each iteration, yielding a new algorithm which updates $\boldsymbol{S}$ much more efficiently. Because it combines Frank-Wolfe and projection steps, we will call this new algorithm Frank-Wolfe-Projection (FW-P).

Properties of the objective and constraints.. To apply Frank-Wolfe to (3.1), we first note that the objective $l(\boldsymbol{L}, \boldsymbol{S})$ in (3.1) is differentiable, with

$$\nabla_{\boldsymbol{L}} l(\boldsymbol{L}, \boldsymbol{S}) = \mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}] \tag{3.2}$$

$$\nabla_{\boldsymbol{S}} l(\boldsymbol{L}, \boldsymbol{S}) = \mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}]. \tag{3.3}$$

Moreover, the following lemma shows that the gradient map $\nabla l(\boldsymbol{L}, \boldsymbol{S}) = (\nabla_{\boldsymbol{L}} l, \nabla_{\boldsymbol{S}} l)$ is 2-Lipschitz:

LEMMA 3.1. *For all* $(\boldsymbol{L}, \boldsymbol{S})$ *and* $(\boldsymbol{L}', \boldsymbol{S}')$*, we have* $\|\nabla l(\boldsymbol{L}, \boldsymbol{S}) - \nabla l(\boldsymbol{L}', \boldsymbol{S}')\|_F \leq 2\|(\boldsymbol{L}, \boldsymbol{S}) - (\boldsymbol{L}', \boldsymbol{S}')\|_F$.

Proof. From (3.2) and (3.3), we have

$$\begin{aligned}
\|\nabla l(\boldsymbol{L}, \boldsymbol{S}) - \nabla l(\boldsymbol{L}', \boldsymbol{S}')\|_F^2 &= 2\|\mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}] - \mathcal{P}_Q[\boldsymbol{L}' + \boldsymbol{S}' - \boldsymbol{M}]\|_F^2 \\
&= 2\|\mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S}] - \mathcal{P}_Q[\boldsymbol{L}' + \boldsymbol{S}']\|_F^2 \\
&\leq 2\|\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{L}' - \boldsymbol{S}'\|_F^2 \\
&\leq 4\|\boldsymbol{L} - \boldsymbol{L}'\|_F^2 + 4\|\boldsymbol{S} - \boldsymbol{S}'\|_F^2 \\
&= 4\|(\boldsymbol{L}, \boldsymbol{S}) - (\boldsymbol{L}', \boldsymbol{S}')\|_F^2,
\end{aligned}$$

which implies the result. ∎

The feasible set in (3.1) is compact. The following lemma bounds its diameter D:

LEMMA 3.2. *The feasible set* $\mathcal{D} = \{(\boldsymbol{L}, \boldsymbol{S}) \mid \|\boldsymbol{L}\|_* \leq \tau_L,\ \|\boldsymbol{S}\|_1 \leq \tau_S\}$ *has diameter* $D \leq 2\sqrt{\tau_L^2 + \tau_S^2}$.

Proof. For any $\boldsymbol{Z} = (\boldsymbol{L}, \boldsymbol{S})$ and $\boldsymbol{Z}' = (\boldsymbol{L}', \boldsymbol{S}') \in \mathcal{D}$,

$$\begin{aligned}
\|\boldsymbol{Z} - \boldsymbol{Z}'\|_F^2 = \|\boldsymbol{L} - \boldsymbol{L}'\|_F^2 + \|\boldsymbol{S} - \boldsymbol{S}'\|_F^2 &\leq (\|\boldsymbol{L}\|_F + \|\boldsymbol{L}'\|_F)^2 + (\|\boldsymbol{S}\|_F + \|\boldsymbol{S}'\|_F)^2 \\
&\leq (\|\boldsymbol{L}\|_* + \|\boldsymbol{L}'\|_*)^2 + (\|\boldsymbol{S}\|_1 + \|\boldsymbol{S}'\|_1)^2 \leq 4\tau_L^2 + 4\tau_S^2.
\end{aligned} \tag{3.4}$$

∎

3.1. Frank-Wolfe for problem (3.1). Since (3.1) asks us to minimize a convex, differentiable function with Lipschitz gradient over a compact convex domain, the Frank-Wolfe method in Algorithm 1 applies. It generates a sequence of iterates $\boldsymbol{x}^k = (\boldsymbol{L}^k, \boldsymbol{S}^k)$. Using the expression for the gradient in (3.2)-(3.3), at each iteration, the step direction $\boldsymbol{v}^k = (\boldsymbol{V}_L^k, \boldsymbol{V}_S^k)$ is generated by solving the linearized subproblem

$$\begin{pmatrix} \boldsymbol{V}_L^k \\ \boldsymbol{V}_S^k \end{pmatrix} \in \arg\min \left\langle \begin{pmatrix} \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}] \\ \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}] \end{pmatrix}, \begin{pmatrix} \boldsymbol{V}_L \\ \boldsymbol{V}_S \end{pmatrix} \right\rangle \quad \text{s.t.} \quad \|\boldsymbol{V}_L\|_* \leq \tau_L,\ \|\boldsymbol{V}_S\|_1 \leq \tau_S, \tag{3.5}$$

which decouples into two independent subproblems:

$$\boldsymbol{V}_L^k \in \arg\min_{\|\boldsymbol{V}_L\|_* \leq \tau_L} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{V}_L \rangle,$$

$$\boldsymbol{V}_S^k \in \arg\min_{\|\boldsymbol{V}_S\|_1 \leq \tau_S} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{V}_S \rangle.$$

These subproblems can be easily solved by exploiting the linear optimization oracles introduced in Section 2.2. In particular,

$$\boldsymbol{V}_L^k = -\tau_L \boldsymbol{u}^k (\boldsymbol{v}^k)^\top, \tag{3.6}$$

$$\boldsymbol{V}_S^k = -\tau_S \cdot \delta_{i^\star j^\star}^k \cdot \boldsymbol{e}_{i^\star}^k (\boldsymbol{e}_{j^\star}^k)^\top, \tag{3.7}$$

where $\boldsymbol{u}^k$ and $\boldsymbol{v}^k$ are leading left- and right- singular vectors of $\mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}]$ and $(i^\star, j^\star)$ is the of the largest element of $\mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}]$ in magnitude and $\delta_{ij}^k := \operatorname{sgn}\left[\left(\mathcal{P}_{\mathcal{Q}}\left[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}\right]\right)_{ij}\right]$. Algorithm 3 gives the Frank-Wolfe method specialized to problem (3.1).

Algorithm 3 Frank-Wolfe method for problem (3.1)

Initialization: $\boldsymbol{L}^0 = \boldsymbol{S}^0 = \boldsymbol{0}$;
for $k = 0,\ 1,\ 2,\ \cdots$ **do**
  $\boldsymbol{D}_L^k \in \arg\min_{\|\boldsymbol{D}_L\|_* \le 1} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_L\rangle$; $\boldsymbol{V}_L^k = \tau_L \boldsymbol{D}_L^k$;
  $\boldsymbol{D}_S^k \in \arg\min_{\|\boldsymbol{D}_S\|_1 \le 1} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_S\rangle$; $\boldsymbol{V}_S^k = \tau_S \boldsymbol{D}_S^k$;
  $\gamma = \frac{2}{k+2}$;
  $\boldsymbol{L}^{k+1} = \boldsymbol{L}^k + \gamma(\boldsymbol{V}_L^k - \boldsymbol{L}^k)$;
  $\boldsymbol{S}^{k+1} = \boldsymbol{S}^k + \gamma(\boldsymbol{V}_S^k - \boldsymbol{S}^k)$;
end for

The major advantage of Algorithm 3 lies in the simplicity of the update rules (3.6)-(3.7). Both have closed form, and both can be computed in time (essentially) linear in the size of the input. Because $\boldsymbol{V}_L^k$ is rank-one, the algorithm can be viewed as performing a sequence of rank one updates.

The major disadvantage of Algorithm 3 is that $\boldsymbol{S}$ has only a one-sparse update at each iteration, since $\boldsymbol{V}_S^k = -\tau_S \boldsymbol{e}_{i^\star}^k (\boldsymbol{e}_{j^\star}^k)^\top$ has only one nonzero entry. This is a significant disadvantage in practice, as the optimal $\boldsymbol{S}^\star$ may have a relatively large number of nonzero entries. Indeed, in theory, the CPCP relaxation works even when a constant fraction of the entries in $\boldsymbol{S}_0$ are nonzero. In applications such as foreground-background separation, the number of nonzero entries in the target sparse term can be quite large. The dashed curves in Figure 1 show the effect of this on the practical convergence of the algorithm, on a simulated example of size $1,000 \times 1,000$, in which about 1% of the entries in the target sparse matrix $\boldsymbol{S}_0$ are nonzero. As shown, the progress is quite slow.

3.2. FW-P algorithm: combining Frank-Wolfe and projected gradient. To overcome the drawback of the naive Frank-Wolfe algorithm described above, we propose incorporating an additional gradient projection step after each Frank-Wolfe update. This additional step updates the sparse term $\boldsymbol{S}$ only, with the goal of accelerating convergence in these variables. At iteration k, let $(\boldsymbol{L}^{k+1/2}, \boldsymbol{S}^{k+1/2})$ be the result produced by Frank-Wolfe. To produce the next iterate, we retain the low rank term $\boldsymbol{L}^{k+1/2}$, but set

$$\boldsymbol{S}^{k+1} = \mathcal{P}_{\|\cdot\|_1 \le \tau_S}\left[\boldsymbol{S}^{k+\frac{1}{2}} - \nabla_{\boldsymbol{S}} l(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}})\right] \tag{3.8}$$

$$= \mathcal{P}_{\|\cdot\|_1 \le \tau_S}\left[\boldsymbol{S}^{k+\frac{1}{2}} - \mathcal{P}_Q[\boldsymbol{L}^{k+\frac{1}{2}} + \boldsymbol{S}^{k+\frac{1}{2}} - \boldsymbol{M}]\right]; \tag{3.9}$$

i.e. we simply take an additional projected gradient step in the sparse term $\boldsymbol{S}$. The resulting algorithm is presented as Algorithm 4 below. We call this method the FW-P algorithm, as it combines Frank-Wolfe steps and projections. In Figure 1, we compare Algorithms 3 and 4 on synthetic data. In this example, the FW-P method is clearly more efficient in recovering $\boldsymbol{L}_0$ and $\boldsymbol{S}_0$.

The convergence of Algorithm 4 can be analyzed by recognizing it as a specific instance of the generalized Frank-Wolfe iteration in Algorithm 2. This projection step

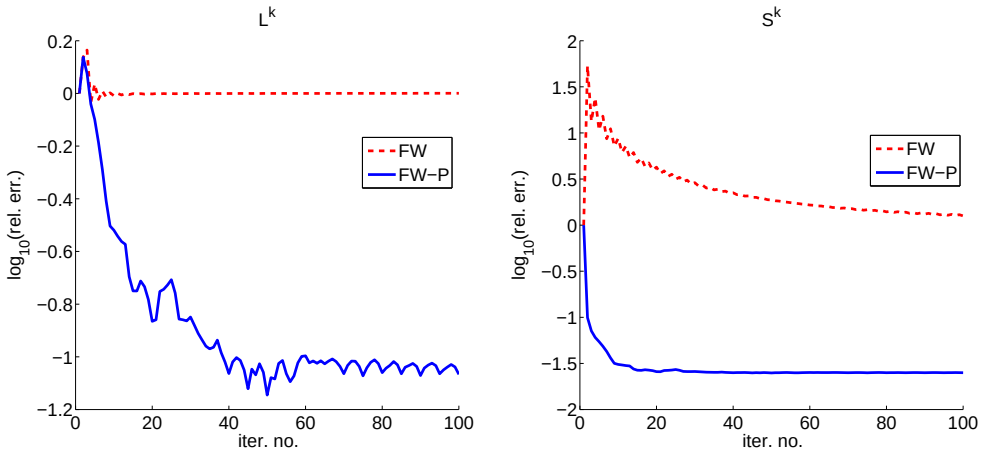

FIG. 1. ***Comparisons between Algorithms 3 and 4 for problem*** (3.1) ***on synthetic data.*** *The data are generated in Matlab as* `m = 1000; n = 1000; r = 5; L0 = randn(m,r) * randn(r,n); Omega = ones(m,n); S0 = 100 * randn(m,n). * (rand(m,n) < 0.01); M = L0 + S0 + randn(m,n); tauL = norm_nuc(L0); tauS = norm(vec(S0),1);` *The left figure plots* $\log_{10}(\|\boldsymbol{L}^k - \boldsymbol{L}_0\|_F / \|\boldsymbol{L}_0\|_F)$ *versus the iteration number* k. *The right figure plots* $\log_{10}(\|\boldsymbol{S}^k - \boldsymbol{S}_0\|_F / \|\boldsymbol{S}_0\|_F)$ *versus* k. *The FW-P method is clearly more efficient than the straightforward FW method in recovering* $\boldsymbol{L}_0$ *and* $\boldsymbol{S}_0$.

Algorithm 4 FW-P method for problem (3.1)

Initialization: $\boldsymbol{L}^0 = \boldsymbol{S}^0 = \boldsymbol{0}$;
for $k = 0,\ 1,\ 2,\ \cdots$ **do**
  $\boldsymbol{D}_L^k \in \arg\min_{\|\boldsymbol{D}_L\|_* \le 1} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_L\rangle$; $\boldsymbol{V}_L^k = \tau_L \boldsymbol{D}_L^k$;
  $\boldsymbol{D}_S^k \in \arg\min_{\|\boldsymbol{D}_S\|_1 \le 1} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_S\rangle$; $\boldsymbol{V}_S^k = \tau_S \boldsymbol{D}_S^k$;
  $\gamma = \frac{2}{k+2}$;
  $\boldsymbol{L}^{k+\frac{1}{2}} = \boldsymbol{L}^k + \gamma(\boldsymbol{V}_L^k - \boldsymbol{L}^k)$;
  $\boldsymbol{S}^{k+\frac{1}{2}} = \boldsymbol{S}^k + \gamma(\boldsymbol{V}_S^k - \boldsymbol{S}^k)$;
  $\boldsymbol{S}^{k+1} = \mathcal{P}_{\|\cdot\|_1 \le \tau_S}\left[\boldsymbol{S}^{k+\frac{1}{2}} - \mathcal{P}_Q[\boldsymbol{L}^{k+\frac{1}{2}} + \boldsymbol{S}^{k+\frac{1}{2}} - \boldsymbol{M}]\right]$;
  $\boldsymbol{L}^{k+1} = \boldsymbol{L}^{k+\frac{1}{2}}$;
end for

(3.9) can be regarded as a proximal step to set $\boldsymbol{S}^{k+1}$ as

$$\arg\min_{\|\boldsymbol{S}\|_1 \le \tau_S} \quad \hat{l}^{k+\frac{1}{2}}(\boldsymbol{S}) := l(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}) + \langle \nabla_{\boldsymbol{S}} l(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}), \boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}}\rangle + \frac{1}{2}\left\|\boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}}\right\|_F^2.$$

It can then be easily verified that

$$\hat{l}^{k+\frac{1}{2}}(\boldsymbol{S}^{k+\frac{1}{2}}) = l(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}), \quad \text{and} \quad \hat{l}^{k+\frac{1}{2}}(\boldsymbol{S}) \ge l(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}) \ \text{ for any } \boldsymbol{S}, \tag{3.10}$$

since $\nabla_{\boldsymbol{S}} l(\boldsymbol{L}, \boldsymbol{S})$ is 1-Lipschitz. This implies that the FW-P algorithm chooses a next iterate whose objective is no worse than that produced by the Frank-Wolfe step:

$$l(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}) = l(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+1}) \le \hat{l}^{k+\frac{1}{2}}(\boldsymbol{S}^{k+1}) \le \hat{l}^{k+\frac{1}{2}}(\boldsymbol{S}^{k+\frac{1}{2}}) = l(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}).$$

This is precisely the property that is required to invoke Algorithm 2 and Theorems 2.1 and 2.2. Using Lemmas 4.1 and 4.2 to estimate the Lipschitz constant of ∇l and the diameter of $\mathcal{D}$, we obtain the following result, which shows that FW-P retains the $O(1/k)$ convergence rate of the original FW method:

THEOREM 3.3. *Let $l^\star$ be the optimal value to problem* (3.1)*, $\boldsymbol{x}^k = (\boldsymbol{L}^k, \boldsymbol{S}^k)$ and $\boldsymbol{v}^k = (\boldsymbol{V}_L^k, \boldsymbol{V}_S^k)$ be the sequence produced by Algorithm 4. Then we have*

$$l(\boldsymbol{L}^k, \boldsymbol{S}^k) - l^\star \leq \frac{16(\tau_L^2 + \tau_S^2)}{k+2}. \tag{3.11}$$

Moreover, for any $K \geq 1$, there exists $1 \leq \tilde{k} \leq K$ such that the surrogate duality gap (defined in (2.13)*) satisfies*

$$d(\boldsymbol{x}^{\tilde{k}}) = \left\langle \boldsymbol{x}^{\tilde{k}} - \boldsymbol{v}^{\tilde{k}}, \nabla l(\boldsymbol{x}^{\tilde{k}}) \right\rangle \leq \frac{48(\tau_L^2 + \tau_S^2)}{K+2}. \tag{3.12}$$

Proof. Substituting $L = 2$ (Lemma 3.1) and $D \leq 2\sqrt{\tau_L^2 + \tau_S^2}$ (Lemma 3.2) into Theorems 2.1 and 2.2, we can easily obtain the above result. ∎

4. FW-T Method for Penalized Problem. In this section, we develop a scalable algorithm for the penalized version of the CPCP problem,

$$\min_{\boldsymbol{L},\boldsymbol{S}} \quad f(\boldsymbol{L}, \boldsymbol{S}) \doteq \frac{1}{2} \left\| \mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}] \right\|_F^2 + \lambda_L \left\| \boldsymbol{L} \right\|_* + \lambda_S \left\| \boldsymbol{S} \right\|_1. \tag{4.1}$$

In Section 4.1, we reformulate problem (4.1) into the form of (2.1) so that the Frank-Wolfe method can be applied. In Section 4.2, we apply the Frank-Wolfe method directly to the reformulated problem, achieving linear per-iteration cost and $O(1/k)$ convergence in function value. However, because it updates the sparse term one element at a time, it converges very slowly on typical numerical examples. In Section 4, we introduce our FW-T method, which resolves this issue. Our FW-T method essentially exploits the Frank-Wolfe step to handle the nuclear norm and a proximal gradient step to handle the ℓ_1-norm, while keeping iteration cost low and retaining convergence guarantees.

4.1. Reformulation as smooth, constrained optimization. Note that problem (4.1) has a non-differentiable objective function and an unbounded feasible set. To apply the Frank-Wolfe method, we exploit a two-step reformulation to transform (4.1) into the form of (2.1). First, we borrow ideas from [24] and work with the epigraph reformulation of (4.1),

$$\begin{aligned} \min \quad & g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) \doteq \frac{1}{2} \left\| \mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}] \right\|_F^2 + \lambda_L t_L + \lambda_S t_S \\ \text{s.t.} \quad & \left\| \boldsymbol{L} \right\|_* \leq t_L, \ \left\| \boldsymbol{S} \right\|_1 \leq t_S, \end{aligned} \tag{4.2}$$

obtained by introducing auxiliary variables t_L and t_S. Now the objective function $g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S)$ is differentiable, with

$$\nabla_L g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) = \nabla_S g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) = \mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}], \tag{4.3}$$

$$\nabla_{t_L} g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) = \lambda_L, \quad \nabla_{t_S} g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) = \lambda_S. \tag{4.4}$$

A calculation, which we summarize in the following lemma, shows that the gradient $\nabla g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) = (\nabla_L g, \nabla_S g, \nabla_{t_L} g, \nabla_{t_S} g)$ is 2-Lipschitz:

LEMMA 4.1. *For all* $(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S)$ *and* $(\boldsymbol{L}', \boldsymbol{S}', t_L', t_S')$ *feasible to* (4.2),

$$\|\nabla g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) - \nabla g(\boldsymbol{L}', \boldsymbol{S}', t_L', t_S')\|_F \leq 2 \|(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) - (\boldsymbol{L}', \boldsymbol{S}', t_L', t_S')\|_F . \tag{4.5}$$

Proof. Based on (4.3) and (4.4), it follows directly that

$$\begin{aligned}\|\nabla g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) - \nabla g(\boldsymbol{L}', \boldsymbol{S}', t_L', t_S')\|_F^2 &\leq 4 \|\boldsymbol{L} - \boldsymbol{L}'\|_F^2 + 4 \|\boldsymbol{S} - \boldsymbol{S}'\|_F^2 \\ &\leq 4 \|(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) - (\boldsymbol{L}', \boldsymbol{S}', t_L', t_S')\|_F^2 ,\end{aligned}$$

which implies the result. ∎

However, the Frank-Wolfe method still cannot deal with (4.2), since its feasible region is unbounded. If we could somehow obtain upper bounds on the optimal values of t_L and t_S: $U_L \geq t_L^\star$ and $U_S \geq t_S^\star$, then we could solve the equivalent problem

$$\begin{aligned} \min \quad & \frac{1}{2} \|\mathcal{P}_Q[\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{M}]\|_F^2 + \lambda_L t_L + \lambda_S t_S \\ \text{s.t.} \quad & \|\boldsymbol{L}\|_* \leq t_L \leq U_L, \ \|\boldsymbol{S}\|_1 \leq t_S \leq U_S, \end{aligned} \tag{4.6}$$

which now has a compact and convex feasible set. One simple way to obtain such U_L, U_S is as follows. One trivial feasible solution to problem (4.2) is $\boldsymbol{L} = \boldsymbol{0}$, $\boldsymbol{S} = \boldsymbol{0}$, $t_L = 0$, $t_S = 0$. This solution has objective value $\frac{1}{2} \|\mathcal{P}_Q[\boldsymbol{M}]\|_F^2$. Hence, the optimal objective value is no larger than this. This implies that for any optimal $t_L^\star, t_S^\star$,

$$t_L^\star \ \leq \ \frac{1}{2\lambda_L} \|\mathcal{P}_Q[\boldsymbol{M}]\|_F^2 , \qquad t_S^\star \ \leq \ \frac{1}{2\lambda_S} \|\mathcal{P}_Q[\boldsymbol{M}]\|_F^2 . \tag{4.7}$$

Hence, we can always choose

$$U_L = \frac{1}{2\lambda_L} \|\mathcal{P}_Q[\boldsymbol{M}]\|_F^2 , \quad U_S = \frac{1}{2\lambda_S} \|\mathcal{P}_Q[\boldsymbol{M}]\|_F^2 \tag{4.8}$$

to produce a valid, bounded feasible region. The following lemma bounds its diameter D:

LEMMA 4.2. *The feasible set* $\mathcal{D} = \{(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) \mid \|\boldsymbol{L}\|_* \leq t_L \leq U_L, \ \|\boldsymbol{S}\|_1 \leq t_S \leq U_S\}$ *has diameter* $D \leq \sqrt{5} \cdot \sqrt{U_L^2 + U_S^2}$.

Proof. Since for any $\boldsymbol{Z} = (\boldsymbol{L}, \boldsymbol{S}, t_L, t_S)$, $\boldsymbol{Z}' = (\boldsymbol{L}', \boldsymbol{S}', t_L', t_S') \in \mathcal{D}$, we have

$$\begin{aligned}\|\boldsymbol{Z} - \boldsymbol{Z}'\|_F^2 &= \|\boldsymbol{L} - \boldsymbol{L}'\|_F^2 + \|\boldsymbol{S} - \boldsymbol{S}'\|_F^2 + (t_L - t_L')^2 + (t_S - t_S')^2 \\ &\leq (\|\boldsymbol{L}\|_F + \|\boldsymbol{L}'\|_F)^2 + (\|\boldsymbol{S}\|_F + \|\boldsymbol{S}'\|_F)^2 + (t_L - t_L')^2 + (t_S - t_S')^2 \\ &\leq (\|\boldsymbol{L}\|_* + \|\boldsymbol{L}'\|_*)^2 + (\|\boldsymbol{S}\|_1 + \|\boldsymbol{S}'\|_1)^2 + (t_L - t_L')^2 + (t_S - t_S')^2 \\ &\leq (U_L + U_L)^2 + (U_S + U_S)^2 + U_L^2 + U_S^2 \\ &= 5(U_L^2 + U_S^2),\end{aligned}$$

which implies the result. ∎

With these modifications, we can apply Frank-Wolfe directly to obtain a solution $(\widehat{\boldsymbol{L}}, \widehat{\boldsymbol{S}}, \widehat{t_L}, \widehat{t_S})$ to (4.6), and hence to produce a solution $(\widehat{\boldsymbol{L}}, \widehat{\boldsymbol{S}})$ to the original problem (4.1). In subsection 4.2, we describe how to do this. Unfortunately, this straightforward solution has two main disadvantages. First, as in the norm constrained case, it produces only one-sparse updates to $\boldsymbol{S}$, which results in slow convergence. Second, the exact primal convergence rate in Theorem 2.1 depends on the diameter of the feasible set, which in turn depends on the accuracy of our (crude) upper bounds U_L and U_S. In subsection 4.3, we show how to remedy both issues, yielding a Frank-Wolfe-Thresholding method that performs significantly better in practice.

4.2. Frank-Wolfe for problem (4.6). Applying the Frank-Wolfe method in Algorithm 1 generates a sequence of iterates $\boldsymbol{x}^k = (\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)$. Using the expressions for the gradient in (4.3) and (4.4), at each iteration, $\boldsymbol{v}^k = (\boldsymbol{V}_L^k, \boldsymbol{V}_S^k, V_{t_L}^k, V_{t_S}^k)$ is generated by solving the linearized subproblem

$$\boldsymbol{v}^k \in \arg\min_{\boldsymbol{v}\in\mathcal{D}} \quad \left\langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}], \boldsymbol{V}_L + \boldsymbol{V}_S \right\rangle + \lambda_L V_{t_L} + \lambda_S V_{t_S}, \tag{4.9}$$

which can be decoupled into two independent subproblems,

$$(\boldsymbol{V}_L^k, V_{t_L}^k) \in \arg\min_{\|\boldsymbol{V}_L\|_* \le V_{t_L} \le U_L} g_L(\boldsymbol{V}_L, V_{t_L}) \doteq \left\langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}], \boldsymbol{V}_L \right\rangle + \lambda_L V_{t_L} \tag{4.10}$$

$$(\boldsymbol{V}_S^k, V_{t_S}^k) \in \arg\min_{\|\boldsymbol{V}_S\|_1 \le V_{t_S} \le U_S} g_S(\boldsymbol{V}_S, V_{t_S}) \doteq \left\langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}], \boldsymbol{V}_S \right\rangle + \lambda_S V_{t_S}. \tag{4.11}$$

Let us consider problem (4.10) first. Set

$$\boldsymbol{D}_L^k \in \arg\min_{\|\boldsymbol{D}_L\|_* \le 1} \quad \hat{g}_L(\boldsymbol{D}_L) \doteq \left\langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}], \boldsymbol{D}_L \right\rangle + \lambda_L. \tag{4.12}$$

Because $g_L(\boldsymbol{V}_L, V_{t_L})$ is a homogeneous function, i.e., $g_L(\alpha \boldsymbol{V}_L, \alpha V_{t_L}) = \alpha g_L(\boldsymbol{V}_L, V_{t_L})$, for any $\alpha \in \mathbb{R}$, its optimal value $g(\boldsymbol{V}_L^k, V_{t_L}^k) = V_{t_L}^k \hat{g}_L(\boldsymbol{D}_L^k)$. Hence $V_{t_L}^k = U_L$ if $\hat{g}_L(\boldsymbol{D}_L^k) < 0$, and $V_{t_L}^k = 0$ if $\hat{g}_L(\boldsymbol{D}_L^k) > 0$. From this observation, it can be easily verified (see also [24, Lemma 1] for a more general result) that

$$(\boldsymbol{V}_L^k, V_{t_L}^k) \in \begin{cases} \{(\boldsymbol{0}, 0)\} & \text{if } \hat{g}_L(\boldsymbol{D}_L^k) > 0 \\ \operatorname{conv}\{(\boldsymbol{0}, 0), U_L(\boldsymbol{D}_L^k, 1)\} & \text{if } \hat{g}_L(\boldsymbol{D}_L^k) = 0 \\ \{U_L(\boldsymbol{D}_L^k, 1)\} & \text{if } \hat{g}_L(\boldsymbol{D}_L^k) < 0. \end{cases} \tag{4.13}$$

In a similar manner, we can update $(\boldsymbol{V}_S^k, V_{t_S}^k)$. This leads fairly directly to the implementation of the Frank-Wolfe method for problem (4.6), described in Algorithm 5. As a direct corollary of Theorem 2.1, using parameters calculated in Lemmas 4.1 and 4.2, we have

COROLLARY 4.3. *Let $\boldsymbol{x}^\star = (\boldsymbol{L}^\star, \boldsymbol{S}^\star, t_L^\star, t_S^\star)$ be an optimal solution to (4.6). For $\{\boldsymbol{x}^k\}$ generated by Algorithm 5, we have for $k = 0,\ 1,\ 2,\ \ldots,$*

$$g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star) \le \frac{20(U_L^2 + U_S^2)}{k+2}. \tag{4.14}$$

Proof. Applying Theorem 2.1 with parameters calculated in Lemmas 4.1 and 4.2, we directly have

$$g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star) \le \frac{2 \cdot 2 \cdot \left(\sqrt{5(U_L^2 + U_S^2)}\right)^2}{k+2} = \frac{20(U_L^2 + U_S^2)}{k+2}. \tag{4.15}$$

A more careful calculation below slightly improves the constant in (4.15).

$$\begin{aligned} g(\boldsymbol{x}^{k+1}) &= g(\boldsymbol{x}^k + \gamma(\boldsymbol{v}^k - \boldsymbol{x}^k)) \\ &\le g(\boldsymbol{x}^k) + \gamma \left\langle \nabla g(\boldsymbol{x}^k), \boldsymbol{v}^k - \boldsymbol{x}^k \right\rangle + \gamma^2 \left\| \boldsymbol{V}_L^k - \boldsymbol{L}^k \right\|_F^2 + \gamma^2 \left\| \boldsymbol{V}_S^k - \boldsymbol{S}^k \right\|_F^2 \\ &\le g(\boldsymbol{x}^k) + \gamma \left\langle \nabla g(\boldsymbol{x}^k), \boldsymbol{v}^k - \boldsymbol{x}^k \right\rangle + 4\gamma^2 (U_L^2 + U_S^2), \end{aligned} \tag{4.16}$$

Algorithm 5 Frank-Wolfe method for problem (4.6)

1: **Initialization:** $\boldsymbol{L}^0 = \boldsymbol{S}^0 = \boldsymbol{0}$; $t_L^0 = t_S^0 = 0$;
2: **for** $k = 0,\ 1,\ 2,\ \ldots$ **do**
3: $\quad \boldsymbol{D}_L^k \in \arg\min_{\|\boldsymbol{D}_L\|_* \le 1} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_L \rangle$;
4: $\quad \boldsymbol{D}_S^k \in \arg\min_{\|\boldsymbol{D}_S\|_1 \le 1} \langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_S \rangle$;
5: $\quad$ **if** $\lambda_L \ge -\langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_L^k \rangle$ **then**
6: $\quad\quad \boldsymbol{V}_L^k = \boldsymbol{0}$; $V_{t_L}^k = 0$
7: $\quad$ **else**
8: $\quad\quad \boldsymbol{V}_L^k = U_L \boldsymbol{D}_L^k$, $V_{t_L}^k = U_L$;
9: $\quad$ **end if**
10: $\quad$ **if** $\lambda_S \ge -\langle \mathcal{P}_Q[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}],\ \boldsymbol{D}_S^k \rangle$ **then**
11: $\quad\quad \boldsymbol{V}_S^k = \boldsymbol{0}$; $V_{t_S}^k = 0$;
12: $\quad$ **else**
13: $\quad\quad \boldsymbol{V}_S^k = U_S \boldsymbol{D}_S^k$, $V_{t_S}^k = U_S$;
14: $\quad$ **end if**
15: $\quad \gamma = \frac{2}{k+2}$;
16: $\quad \boldsymbol{L}^{k+1} = (1-\gamma)\boldsymbol{L}^k + \gamma \boldsymbol{V}_L^k$, $t_L^{k+1} = (1-\gamma) t_L^k + \gamma V_{t_L}^k$;
17: $\quad \boldsymbol{S}^{k+1} = (1-\gamma)\boldsymbol{S}^k + \gamma \boldsymbol{V}_S^k$, $t_S^{k+1} = (1-\gamma) t_S^k + \gamma V_{t_S}^k$;
18: **end for**

where the second line holds by noting that g is only linear in t_L and t_S; the last line holds as

$$\left\|\boldsymbol{V}_L^k - \boldsymbol{L}^k\right\|_F^2 \le (\left\|\boldsymbol{V}_L^k\right\|_F + \left\|\boldsymbol{L}^k\right\|_F)^2 \le (U_L + U_L)^2 = 4U_L^2, \quad \text{and}$$
$$\left\|\boldsymbol{V}_S^k - \boldsymbol{S}^k\right\|_F^2 \le (\left\|\boldsymbol{V}_S^k\right\|_F + \left\|\boldsymbol{S}^k\right\|_F)^2 \le (U_S + U_S)^2 = 4U_S^2.$$

Following the arguments in the proof of Theorem 1 with (2.10) replaced by (4.16), we can easily obtain that

$$g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star) \le \frac{16(U_L^2 + U_S^2)}{k+2}.$$

□

In addition to the above convergence result, another major advantage of Algorithm 5 is the simplicity of the update rules (lines 3-4 in Algorithm 5). Both have closed-form solutions that can be computed in time (essentially) linearly dependent on the size of the input.

However, two clear limitations substantially hinder Algorithm 5's efficiency. First, as in the norm constrained case, $\boldsymbol{V}_S^k$ has only one nonzero entry, so $\boldsymbol{S}$ has a one-sparse update in each iteration. Second, the exact rate of convergence relies on our (crude) guesses of U_L and U_S (Corollary 4.3). In the next subsection, we present remedies to resolve both issues.

4.3. FW-T algorithm: combining Frank-Wolfe and proximal methods. To alleviate the difficulties faced by Algorithm 5, we propose a new algorithm called Frank-Wolfe-Thresholding (FW-T) (Algorithm 6), that combines a modified FW step with a proximal gradient step. Below we highlight the key features of FW-T.

Proximal gradient step for $\boldsymbol{S}$. To update $\boldsymbol{S}$ in a more efficient way, we incorporate an additional proximal gradient step for $\boldsymbol{S}$. At iteration k, let $(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}})$

be the result produced by Frank-Wolfe step. To produce the next iterate, we retain the low-rank term $\boldsymbol{L}^{k+\frac{1}{2}}$, but execute a proximal gradient step for the function $f(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S})$ at the point $\boldsymbol{S}^{k+\frac{1}{2}}$, i.e.

$$\begin{aligned}\boldsymbol{S}^{k+1} &\in \arg\min_{\boldsymbol{S}} \left\langle \nabla_{\boldsymbol{S}} f(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}),\ \boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}} \right\rangle + \frac{1}{2}\left\|\boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}}\right\|_F^2 + \lambda_S \|\boldsymbol{S}\|_1 \\ &= \arg\min_{\boldsymbol{S}} \left\langle \mathcal{P}_Q[\boldsymbol{L}^{k+\frac{1}{2}} + \boldsymbol{S}^{k+\frac{1}{2}} - \boldsymbol{M}], \boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}} \right\rangle + \frac{1}{2}\left\|\boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}}\right\|_F^2 + \lambda_S \|\boldsymbol{S}\|_1\end{aligned} \tag{4.17}$$

which can be easily computed using the soft-thresholding operator:

$$\boldsymbol{S}^{k+1} = \mathcal{T}_{\lambda_S}\left[\boldsymbol{S}^{k+\frac{1}{2}} - \mathcal{P}_Q[\boldsymbol{L}^{k+\frac{1}{2}} + \boldsymbol{S}^{k+\frac{1}{2}} - \boldsymbol{M}]\right]. \tag{4.18}$$

Exact line search. For the Frank-Wolfe step, instead of choosing the fixed step length $\frac{2}{k+2}$, we implement an exact line search by solving a two-dimensional quadratic problem (4.20), as in [24]. This modification turns out to be crucial to achieve a primal convergence result that only weakly depends on the tightness of our guesses U_L and U_S.

Adaptive updates of U_L ***and*** U_S. We initialize U_L and U_S using the crude bound (4.8). Then, at the end of the k-iteration, we respectively update

$$U_L^{k+1} = g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_L, \quad U_S^{k+1} = g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_S. \tag{4.19}$$

This scheme maintains the property that $U_L^{k+1} \geq t_L^\star$ and $U_S^{k+1} \geq t_S^\star$. Moreover, we prove (Lemma 4.4) that g is non-increasing through our algorithm, and so this scheme produces a sequence of tighter upper bounds for $U_L^\star$ and $U_S^\star$. Although this dynamic scheme does not improve the theoretical convergence result, some acceleration is empirically exhibited.

Convergence analysis. Since both the FW step and the proximal gradient step do not increase the objective value, we can easily recognize FW-T method as a descent algorithm:

LEMMA 4.4. *Let* $\{(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)\}$ *be the sequence of iterates produced by the FW-T algorithm. For each* $k = 0, 1, 2\cdots,$

$$g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1}) \leq g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \leq g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k). \tag{4.21}$$

Proof. Since $(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)$ is always feasible to the quadratic program (4.20),

$$g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \leq g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k). \tag{4.22}$$

Based on (4.17), the threshold step (line 6 in Algorithm 3) can be written as

$$\begin{aligned}\boldsymbol{S}^{k+1} = \arg\min_{\boldsymbol{S}} \quad & \hat{g}^{k+\frac{1}{2}}(\boldsymbol{S}) \doteq \frac{1}{2}\left\|\mathcal{P}_Q[\boldsymbol{L}^{k+\frac{1}{2}} + \boldsymbol{S}^{k+\frac{1}{2}} - \boldsymbol{M}]\right\|_F^2 + \lambda_L t_L^{k+\frac{1}{2}} + \lambda_S \|\boldsymbol{S}\|_1 \\ & + \langle \mathcal{P}_Q[\boldsymbol{L}^{k+\frac{1}{2}} + \boldsymbol{S}^{k+\frac{1}{2}} - \boldsymbol{M}],\ \boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}} \rangle + \frac{1}{2}\left\|\boldsymbol{S} - \boldsymbol{S}^{k+\frac{1}{2}}\right\|_F^2.\end{aligned}$$

The following properties of $\hat{g}^{k+\frac{1}{2}}(\cdot)$ can be easily verified

$$\hat{g}^{k+\frac{1}{2}}(\boldsymbol{S}^{k+\frac{1}{2}}) = g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, \|\boldsymbol{S}^{k+\frac{1}{2}}\|_1) \leq g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}});$$

Algorithm 6 FW-T method for problem (4.1)

1: **Input:** data matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$; weights λ_L, $\lambda_S > 0$; max iteration number T;
2: **Initialization:** $\boldsymbol{L}^0 = \boldsymbol{S}^0 = \boldsymbol{0}$; $t_L^0 = t_S^0 = 0$; $U_L^0 = g(\boldsymbol{L}^0, \boldsymbol{S}^0, t_L^0, t_S^0)/\lambda_L$; $U_S^0 = g(\boldsymbol{L}^0, \boldsymbol{S}^0, t_L^0, t_S^0)/\lambda_S$;
3: **for** $k = 0,\ 1,\ 2,\ \cdots,\ T$ **do**
4: *same as lines 3-14 in Algorithm 5;*
5: $\left(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{K+\frac{1}{2}}\right)$ is computed as an optimizer to

$$\begin{aligned} \min \quad & \frac{1}{2}\left\|\mathcal{P}_Q[\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{M}]\right\|_F^2 + \lambda_L t_L + \lambda_S t_S \\ \text{s.t.} \quad & \begin{pmatrix} \boldsymbol{L} \\ t_L \end{pmatrix} \in \operatorname{conv}\left\{ \begin{pmatrix} \boldsymbol{L}^k \\ t_L^k \end{pmatrix}, \begin{pmatrix} \boldsymbol{V}_L^k \\ V_{t_L}^k \end{pmatrix} \right\} \\ & \begin{pmatrix} \boldsymbol{S} \\ t_S \end{pmatrix} \in \operatorname{conv}\left\{ \begin{pmatrix} \boldsymbol{S}^k \\ t_S^k \end{pmatrix}, \begin{pmatrix} \boldsymbol{V}_S^k \\ V_{t_S}^k \end{pmatrix} \right\}; \end{aligned} \tag{4.20}$$

6: $\boldsymbol{S}^{k+1} = \mathcal{T}\left[\boldsymbol{S}^{k+\frac{1}{2}} - \mathcal{P}_Q[\boldsymbol{L}^{k+\frac{1}{2}} + \boldsymbol{S}^{k+\frac{1}{2}} - \boldsymbol{M}], \lambda_S\right]$;
7: $\boldsymbol{L}^{k+1} = \boldsymbol{L}^{k+\frac{1}{2}}$, $t_L^{k+1} = t_L^{k+\frac{1}{2}}$; $t_S^{k+1} = \left\|\boldsymbol{S}^{k+1}\right\|_1$;
8: $U_L^{k+1} = g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_L$;
9: $U_S^{k+1} = g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_S$;
10: **end for**

$$\hat{g}^{k+\frac{1}{2}}(\boldsymbol{S}) \geq g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}, t_L^{k+\frac{1}{2}}, \|\boldsymbol{S}\|_1), \quad \text{for any } \boldsymbol{S}.$$

Therefore, we have

$$\begin{aligned} g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1}) &= g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+1}, t_L^{k+\frac{1}{2}}, t_S^{k+1}) \leq \hat{g}^{k+\frac{1}{2}}(\boldsymbol{S}^{k+1}) \\ &\leq \hat{g}^{k+\frac{1}{2}}(\boldsymbol{S}^{k+\frac{1}{2}}) \leq g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \end{aligned} \tag{4.23}$$

Combining (4.22) and (4.23), we obtain

$$g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1}) \leq g(\boldsymbol{L}^{k+\frac{1}{2}}, \boldsymbol{S}^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \leq g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k).$$

□

Moreover, we can establish primal convergence (almost) independent of U_L^0 and U_S^0:

THEOREM 4.5. *Let $r_L^\star$ and $r_S^\star$ be the smallest radii such that*

$$\left\{(\boldsymbol{L}, \boldsymbol{S}) \;\middle|\; f(\boldsymbol{L}, \boldsymbol{S}) \leq g(\boldsymbol{L}^0, \boldsymbol{S}^0, t_L^0, t_S^0) = \frac{1}{2}\left\|\mathcal{P}_Q[\boldsymbol{M}]\right\|_F^2\right\} \subseteq \overline{B(r_L^\star)} \times \overline{B(r_S^\star)}, \tag{4.24}$$

where $\overline{B(r)} \doteq \{\boldsymbol{X} \in \mathbb{R}^{m\times n} | \left\|\boldsymbol{X}\right\|_F \leq r\}$ for any $r \geq 0$.[‡] *Then for the sequence $\{(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)\}$ generated by Algorithm 6, we have*

$$\begin{aligned} & g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k) - g(\boldsymbol{L}^\star, \boldsymbol{S}^\star, t_L^\star, t_S^\star) \\ & \qquad \leq \frac{\min\{4(t_L^\star + r_L^\star)^2 + 4(t_S^\star + r_S^\star)^2,\ 16(U_L^0)^2 + 16(U_S^0)^2\}}{k+2}. \end{aligned} \tag{4.25}$$

[‡]Since the objective function in problem (4.1) is coercive, i.e. $\lim_{k\to+\infty} f(\boldsymbol{L}^k, \boldsymbol{S}^k) = +\infty$ for any sequence $(\boldsymbol{L}^k, \boldsymbol{S}^k)$ such that $\lim_{k\to+\infty} \left\|(\boldsymbol{L}^k, \boldsymbol{S}^k)\right\|_F = +\infty$, clearly $r_L^\star \geq 0$ and $r_S^\star \geq 0$ exist.

Proof. For notational convenience, we denote

$$\boldsymbol{x}^k = (\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k),\ \boldsymbol{x}^\star = (\boldsymbol{L}^\star, \boldsymbol{S}^\star, t_L^\star, t_S^\star) \text{ and } \boldsymbol{v}^k = (\boldsymbol{V}_L^k, \boldsymbol{V}_S^k, \boldsymbol{V}_{t_L}^k, \boldsymbol{V}_{t_S}^k).$$

For any point $\boldsymbol{x} = (\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \times \mathbb{R} \times \mathbb{R}$, we adopt the notation that $\boldsymbol{L}[\boldsymbol{x}] = \boldsymbol{L}$, $\boldsymbol{S}[\boldsymbol{x}] = \boldsymbol{S}$, $t_L[\boldsymbol{x}] = t_L$ and $t_S[\boldsymbol{x}] = t_S$.

Since $g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star) \le \frac{16(U_L^0)^2+16(U_S^0)^2}{k+2}$ can be easily established following the proof of Corollary 4.3, below we will focus on the other part that $g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star) \le \frac{4(t_L^\star+r_L^\star)^2+4(t_S^\star+r_S^\star)^2}{k+2}$.

Let us first make two simple observations.

Since $f(\boldsymbol{L}^\star, \boldsymbol{S}^\star) \le g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)$, we have

$$U_L^k = g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)/\lambda_L \ge t_L^\star \quad \text{and} \quad U_S^k = g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)/\lambda_S \ge t_S^\star. \tag{4.26}$$

Therefore, our U_L^k and U_S^k always bound $t_L^\star$ and $t_S^\star$ from above.

From Lemma 4.4, $g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)$ is non-increasing,

$$f(\boldsymbol{L}^k, \boldsymbol{S}^k) \le g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k) \le g(\boldsymbol{L}^0, \boldsymbol{S}^0, t_L^0, t_S^0),$$

which implies that $(\boldsymbol{L}^k, \boldsymbol{S}^k) \subseteq \overline{B(r_L^\star)} \times \overline{B(r_S^\star)}$, i.e. $\left\|\boldsymbol{L}^k\right\|_F \le r_L^\star$ and $\left\|\boldsymbol{S}^k\right\|_F \le r_S^\star$.

Let us now consider the k-th iteration. Similar to the proof in [24], we introduce the auxiliary point $\boldsymbol{v}_+^k = (\frac{t_L^\star}{U_L^k}\boldsymbol{V}_L^k, \frac{t_S^\star}{U_S^k}\boldsymbol{V}_S^k, \frac{t_L^\star}{U_L^k}\boldsymbol{V}_{t_L}^k, \frac{t_S^\star}{U_S^k}\boldsymbol{V}_{t_S}^k)$. Then based on our argument for (4.13), it can be easily verified that

$$(\boldsymbol{L}[\boldsymbol{v}_+^k], t_L[\boldsymbol{v}_+^k]) \in \arg\min_{\|\boldsymbol{V}_L\|_* \le V_{t_L} \le t_L^\star} g_L(\boldsymbol{V}_L, V_{t_L}) \tag{4.27}$$

$$(\boldsymbol{S}[\boldsymbol{v}_+^k], t_S[\boldsymbol{v}_+^k]) \in \arg\min_{\|\boldsymbol{V}_S\|_1 \le V_{t_S} \le t_S^\star} g_S(\boldsymbol{V}_S, V_{t_S}). \tag{4.28}$$

Recall $\gamma = \frac{2}{k+2}$. We have

$$\begin{aligned}
&g(\boldsymbol{x}^{k+\frac{1}{2}}) \\
&\le g(\boldsymbol{x}^k + \gamma(\boldsymbol{v}_+^k - \boldsymbol{x}^k)) \\
&\le g(\boldsymbol{x}^k) + \gamma\langle \nabla g(\boldsymbol{x}_k),\ \boldsymbol{v}_+^k - \boldsymbol{x}^k\rangle + \gamma^2\left(\left\|\boldsymbol{L}[\boldsymbol{v}_+^k] - \boldsymbol{L}[\boldsymbol{x}^k]\right\|_F^2 + \left\|\boldsymbol{S}[\boldsymbol{v}_+^k] - \boldsymbol{S}[\boldsymbol{x}^k]\right\|_F^2\right) \\
&\le g(\boldsymbol{x}^k) + \gamma\left(g_L(\boldsymbol{L}[\boldsymbol{v}_+^k - \boldsymbol{x}^k], t_L[\boldsymbol{v}_+^k - \boldsymbol{x}^k]) + g_S(\boldsymbol{S}[\boldsymbol{v}_+^k - \boldsymbol{x}^k], t_S[\boldsymbol{v}_+^k - \boldsymbol{x}^k])\right) \\
&\qquad\qquad +\gamma^2\left((t_L^\star + r_L^\star)^2 + (t_S^\star + r_S^\star)^2\right) \\
&\le g(\boldsymbol{x}^k) + \gamma\left(g_L(\boldsymbol{L}[\boldsymbol{x}^\star - \boldsymbol{x}^k], t_L[\boldsymbol{x}^\star - \boldsymbol{x}^k]) + g_S(\boldsymbol{S}[\boldsymbol{x}^\star - \boldsymbol{x}^k], t_S[\boldsymbol{x}^\star - \boldsymbol{x}^k])\right) \\
&\qquad\qquad +\gamma^2\left((t_L^\star + r_L^\star)^2 + (t_S^\star + r_S^\star)^2\right) \\
&= g(\boldsymbol{x}^k) + \gamma\langle \nabla g(\boldsymbol{x}^k),\ \boldsymbol{x}^\star - \boldsymbol{x}^k\rangle + \gamma^2\left((t_L^\star + r_L^\star)^2 + (t_S^\star + r_S^\star)^2\right) \\
&\le g(\boldsymbol{x}^k) + \gamma\left(g(\boldsymbol{x}^\star) - g(\boldsymbol{x}^k)\right) + \gamma^2\left((t_L^\star + r_L^\star)^2 + (t_S^\star + r_S^\star)^2\right),
\end{aligned}$$

where the first inequality holds since $\boldsymbol{x}^k + \gamma(\boldsymbol{v}_+^k - \boldsymbol{x}^k)$ is feasible to the quadratic program (4.20) while $\boldsymbol{x}^{k+\frac{1}{2}}$ minimizes it; the third inequality is due to the facts that

$$\begin{aligned}
&\left\|\boldsymbol{L}[\boldsymbol{v}_+^k] - \boldsymbol{L}[\boldsymbol{x}^k]\right\|_F \le \left\|\boldsymbol{L}[\boldsymbol{v}_+^k]\right\|_F + \left\|\boldsymbol{L}[\boldsymbol{x}^k]\right\|_F \le \left\|\boldsymbol{L}[\boldsymbol{v}_+^k]\right\|_* + \left\|\boldsymbol{L}[\boldsymbol{x}^k]\right\|_F \le t_L^\star + r_L^\star \\
&\left\|\boldsymbol{S}[\boldsymbol{v}_+^k] - \boldsymbol{S}[\boldsymbol{x}^k]\right\|_F \le \left\|\boldsymbol{S}[\boldsymbol{v}_+^k]\right\|_F + \left\|\boldsymbol{S}[\boldsymbol{x}^k]\right\|_F \le \left\|\boldsymbol{S}[\boldsymbol{v}_+^k]\right\|_1 + \left\|\boldsymbol{S}[\boldsymbol{x}^k]\right\|_F \le t_S^\star + r_S^\star;
\end{aligned}$$

the fourth inequality holds as $(\boldsymbol{L}[\boldsymbol{x}^\star], t_L[\boldsymbol{x}^\star])$ and $(\boldsymbol{S}[\boldsymbol{x}^\star], t_S[\boldsymbol{x}^\star])$ are respectively feasible to (4.27) and (4.28) while $(\boldsymbol{L}[\boldsymbol{v}_+^k], t_L[\boldsymbol{v}_+^k])$ and $(\boldsymbol{S}[\boldsymbol{v}_+^k], t_S[\boldsymbol{v}_+^k])$ respectively minimize (4.27) and (4.28);

Therefore, we obtain

$$g(\boldsymbol{x}^{k+\frac{1}{2}}) - g(\boldsymbol{x}^\star) \leq (1-\gamma)\left(g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star)\right) + \gamma^2\left((t_L^\star + r_L^\star)^2 + (t_S^\star + r_S^\star)^2\right).$$

Moreover, by Lemma 4.4, we have

$$g(\boldsymbol{x}^{k+1}) \leq g(\boldsymbol{x}^{k+\frac{1}{2}}).$$

Thus, we obtain the recurrence

$$g(\boldsymbol{x}^{k+1}) - g(\boldsymbol{x}^\star) \leq (1-\gamma)\left(g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star)\right) + \gamma^2\left((t_L^\star + r_L^\star)^2 + (t_S^\star + r_S^\star)^2\right).$$

Applying mathematical induction, one can easily obtain that

$$g(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k) - g(\boldsymbol{L}^\star, \boldsymbol{S}^\star, t_L^\star, t_S^\star) \leq \frac{4\left((t_L^\star + r_L^\star)^2 + (t_S^\star + r_S^\star)^2\right)}{k+2}.$$

□

Since U_L^0 and U_S^0 are quite crude upper bounds for $t_L^\star$ and $t_S^\star$, $16(U_L^0)^2 + 16(U_S^0)^2$ could be much larger than $4(t_L^\star + r_L^\star)^2 + 4(t_S^\star + r_S^\star)^2$. Therefore, this primal convergence results depend on U_L^0 and U_S^0 in a very weak manner.

However, the convergence result of the surrogate duality gap $d(\boldsymbol{x}^k)$ still hinges upon the upper bounds:

THEOREM 4.6. *Let $\boldsymbol{x}^k$ denote $(\boldsymbol{L}^k, \boldsymbol{S}^k, t_L^k, t_S^k)$ generated by Algorithm 6. Then for any $K \geq 1$, there exists $1 \leq \tilde{k} \leq K$ such that*

$$g(\boldsymbol{x}^{\tilde{k}}) - g(\boldsymbol{x}^\star) \leq d(\boldsymbol{x}^{\tilde{k}}) \leq \frac{48\left((U_L^0)^2 + (U_S^0)^2\right)}{K+2}. \tag{4.29}$$

Proof. Define $\Delta^k = g(\boldsymbol{x}^k) - g(\boldsymbol{x}^\star)$. Following (4.16), we have

$$\Delta^{k+1} \leq \Delta^k + \gamma\left\langle \nabla g(\boldsymbol{x}^k), \boldsymbol{v}^k - \boldsymbol{x}^k\right\rangle + 4\gamma^2\left((U_L^0)^2 + (U_S^0)^2\right). \tag{4.30}$$

Then following the arguments in the proof of Theorem 2 with (2.17) replaced by (4.30), we can easily obtain the result. □

Stopping criterion. Compared to the convergence of $g(\boldsymbol{x}^k)$ (Theorem 4.5), the convergence result for $d(\boldsymbol{x}^k)$ can be much slower (Theorem 4.6). Therefore, here the surrogate duality gap $d(\cdot)$ is not that suitable to serve as a stopping criterion. Consequently, in our implementation, we terminate Algorithm 6 if

$$\left|g(\boldsymbol{x}^{k+1}) - g(\boldsymbol{x}^k)\right| / g(\boldsymbol{x}^k) \leq \varepsilon, \tag{4.31}$$

for five consecutive iterations.

5. Numerical Experiments. In this section, we report numerical results obtained by applying our FW-T method (Algorithm 6) to problem (1.5) with real data arising from applications considered in [3]: *foreground/background separation in surveillance videos*, and *shadow and specularity removal from face images*.

Given observations $\{\boldsymbol{M}_0(i,j) \mid (i,j) \in \Omega\}$, where $\Omega \subseteq \{1,\ldots,m\} \times \{1,\ldots,n\}$ is the index set of the observable entries in $\boldsymbol{M}_0 \in \mathbb{R}^{m\times n}$, we assigned weights

$$\lambda_L = \delta\rho \left\|\mathcal{P}_\Omega[\boldsymbol{M}_0]\right\|_F \quad \text{and} \quad \lambda_S = \delta\sqrt{\rho}\left\|\mathcal{P}_\Omega[\boldsymbol{M}_0]\right\|_F / \sqrt{\max(m,n)}$$

to problem (1.5), [§] where $\rho = |\Omega|/mn$ and δ is chosen as 0.001 for the surveillance problem and 0.01 for the face problem.

We compared our FW-T method with the popular first-order methods *iterative soft-thresholding algorithm* (ISTA) and *fast iterative soft-thresholding algorithm* (FISTA) [20], both of whose implementations used partial *singular value decomposition* (SVD). In subsection 5.1, we provided detailed descriptions and implementations of ISTA and FISTA.

We set $\varepsilon = 10^{-3}$ in FW-T's stopping criterion (4.31),[¶] and terminated ISTA and FISTA whenever they reached the objective value returned by the FW-T method.[‖] All the experiments were conducted on a computer with Intel Xeon E5-2630 Processor (12 cores at 2.4 GHz), and 64GB RAM running MATLAB R2012b (64 bits).

5.1. ISTA & FISTA for problem (1.5)**.** *Iterative soft-thresholding algorithm* (ISTA), is an efficient way to tackle unconstrained nonsmooth optimization problem especially at large scale. ISTA follows the general idea by iteratively minimizing an upper bound of the original objective. In particular, when applied to problem (1.5) of our interest, ISTA updates $(\boldsymbol{L},\boldsymbol{S})$ for the k-th iteration by solving

(5.1)

$$(\boldsymbol{L}^{k+1},\boldsymbol{S}^{k+1}) = \arg\min_{\boldsymbol{L},\boldsymbol{S}} \left\langle \begin{pmatrix} \nabla_{\boldsymbol{L}} l(\boldsymbol{L}^k,\boldsymbol{S}^k) \\ \nabla_{\boldsymbol{S}} l(\boldsymbol{L}^k,\boldsymbol{S}^k) \end{pmatrix}, \begin{pmatrix} \boldsymbol{L}-\boldsymbol{L}^k \\ \boldsymbol{S}-\boldsymbol{S}^k \end{pmatrix} \right\rangle + \frac{L_f}{2}\left\| \begin{pmatrix} \boldsymbol{L} \\ \boldsymbol{S} \end{pmatrix} - \begin{pmatrix} \boldsymbol{L}^k \\ \boldsymbol{S}^k \end{pmatrix} \right\|_F^2 + \lambda_L \|\boldsymbol{L}\|_* + \lambda_S \|\boldsymbol{S}\|_1 .$$

Here $L_f = 2$ denotes the Lipschitz constant of $\nabla l(\boldsymbol{L},\boldsymbol{S})$ with respect to $(\boldsymbol{L},\boldsymbol{S})$, and $\nabla_{\boldsymbol{L}} l(\boldsymbol{L}^k,\boldsymbol{S}^k) = \nabla_{\boldsymbol{S}} l(\boldsymbol{L}^k,\boldsymbol{S}^k) = \mathcal{P}_\Omega[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}]$. Since $\boldsymbol{L}$ and $\boldsymbol{S}$ are decoupled in (5.1), equivalently we have

$$\boldsymbol{L}^{k+1} = \arg\min_{\boldsymbol{L}} \left\| \boldsymbol{L} - \left(\boldsymbol{L}^k - \frac{1}{2}\mathcal{P}_\Omega[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}] \right) \right\|_F^2 + \lambda_L \|\boldsymbol{L}\|_* , \tag{5.2}$$

$$\boldsymbol{S}^{k+1} = \arg\min_{\boldsymbol{S}} \left\| \boldsymbol{S} - \left(\boldsymbol{S}^k - \frac{1}{2}\mathcal{P}_\Omega[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}] \right) \right\|_F^2 + \lambda_S \|\boldsymbol{S}\|_1 . \tag{5.3}$$

The solution to problem (5.3) can be given explicitly in terms of the proximal mapping of $\|\cdot\|_1$ as introduced in Section 2.2, i.e.,

$$\boldsymbol{S}^{k+1} = \mathcal{T}_{\lambda_S/2}\left[\boldsymbol{S}^k - \frac{1}{2}\mathcal{P}_\Omega[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}] \right].$$

For a matrix $\boldsymbol{X}$ and any $\tau \geq 0$, let $\mathcal{D}_\tau(\boldsymbol{X})$ denote the singular value thresholding operator $\mathcal{D}_\tau(\boldsymbol{X}) = \boldsymbol{U}\mathcal{T}_\tau(\boldsymbol{\Sigma})\boldsymbol{V}^\top$, where $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the singular value decomposition

[§] The ratio $\lambda_L/\lambda_S = \sqrt{\rho\max(m,n)}$ follows the suggestion in [3]. For applications in computer vision at least, our choices in λ_L and λ_S seem to be quite robust, although it is possible to improve the performance by making slight adjustments to our current settings of λ_L and λ_S.

[¶] As discussed in [33, 34], with noisy data, solving optimization problems to high accuracy does not necessarily improve the recovery quality. Consequently, we set ε to a modest value.

[‖] All codes are available at: https://sites.google.com/site/mucun1988/publi

of $\boldsymbol{X}$. It is not difficult to show [35, 36] that the solution to problem (5.2) can be given explicitly by

$$\boldsymbol{L}^{k+1} = \mathcal{D}_{\lambda_L/2}\left[\boldsymbol{L}^k - \frac{1}{2}\mathcal{P}_\Omega[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}]\right].$$

Algorithm 7 summarizes our ISTA implementation for problem (1.5).

Algorithm 7 ISTA for problem (1.5)

Initialization: $\boldsymbol{L}^0 = \boldsymbol{0}$, $\boldsymbol{S}^0 = \boldsymbol{0}$;
for $k = 0,\ 1,\ 2,\ \cdots$ **do**
  $\boldsymbol{L}^{k+1} = \mathcal{D}_{\lambda_L/2}\left[\boldsymbol{L}^k - \frac{1}{2}\mathcal{P}_\Omega[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}]\right]$;
  $\boldsymbol{S}^{k+1} = \mathcal{T}_{\lambda_S/2}\left[\boldsymbol{S}^k - \frac{1}{2}\mathcal{P}_\Omega[\boldsymbol{L}^k + \boldsymbol{S}^k - \boldsymbol{M}]\right]$;
end for

Regarding ISTA's speed of convergence, it can be proved that $f(\boldsymbol{L}^k, \boldsymbol{S}^k) - f^\star = O(1/k)$, where $f^\star$ denotes the optimal value of problem (1.5).

Fast iterative soft-thresholding algorithm (FISTA) introduced in [20], is an accelerated version of ISTA, which incorporate a momentum step borrowed from Nesterov's optimal gradient scheme [37]. For FISTA, a better convergence result, $f(\boldsymbol{L}^k, \boldsymbol{S}^k) - f^\star = O(1/k^2)$, can be achieved with a cost per iteration that is comparable to ISTA. Algorithm 8 summarizes our FISTA implementation for problem (1.5).

Algorithm 8 FISTA for problem (1.5)

Initialization: $\hat{\boldsymbol{L}}^0 = \boldsymbol{L}^0 = \boldsymbol{0}$, $\hat{\boldsymbol{S}}^0 = \boldsymbol{S}^0 = \boldsymbol{0}$, $t_0 = 1$;
for $k = 0,\ 1,\ 2,\ \cdots$ **do**
  $\boldsymbol{L}^{k+1} = \mathcal{D}_{\lambda_L/2}\left[\hat{\boldsymbol{L}}^k - \frac{1}{2}\mathcal{P}_\Omega[\hat{\boldsymbol{L}}^k + \hat{\boldsymbol{S}}^k - \boldsymbol{M}]\right]$;
  $\boldsymbol{S}^{k+1} = \mathcal{T}_{\lambda_S/2}\left[\hat{\boldsymbol{S}}^k - \frac{1}{2}\mathcal{P}_\Omega[\hat{\boldsymbol{L}}^k + \hat{\boldsymbol{S}}^k - \boldsymbol{M}]\right]$;
  $t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2}$;
  $\hat{\boldsymbol{L}}^{k+1} = \boldsymbol{L}^{k+1} + \frac{t^k-1}{t^{k+1}}(\boldsymbol{L}^{k+1} - \boldsymbol{L}^k)$;
  $\hat{\boldsymbol{S}}^{k+1} = \boldsymbol{S}^{k+1} + \frac{t^k-1}{t^{k+1}}(\boldsymbol{S}^{k+1} - \boldsymbol{S}^k)$;
end for

Partial SVD. In each iteration of either ISTA or FISTA, we only need those singular values that are larger than $\lambda_S/2$ and their corresponding singular vectors. Therefore, a partial SVD can be utilized to reduce the computational burden of a full SVD. Since most partial SVD software packages (e.g. PROPACK [38]) require specifying in advance the number of top singular values and singular vectors to compute, we heuristically determine this number (denoted as sv^k at iteration k). Specifically, let $d = \min\{m, n\}$, and svp^k denote the number of computed singular values that were larger than $\lambda_S/2$ in the k-th iteration. Similar to [17], in our implementation, we start with $sv^0 = d/10$, and adjust sv^k dynamically as follows:

$$sv^{k+1} = \begin{cases} \min\{svp^k + 1, d\} & \text{if } svp^k < sv^k \\ \min\{svp^k + \text{round}(0.05d), d\} & \text{otherwise.} \end{cases}$$

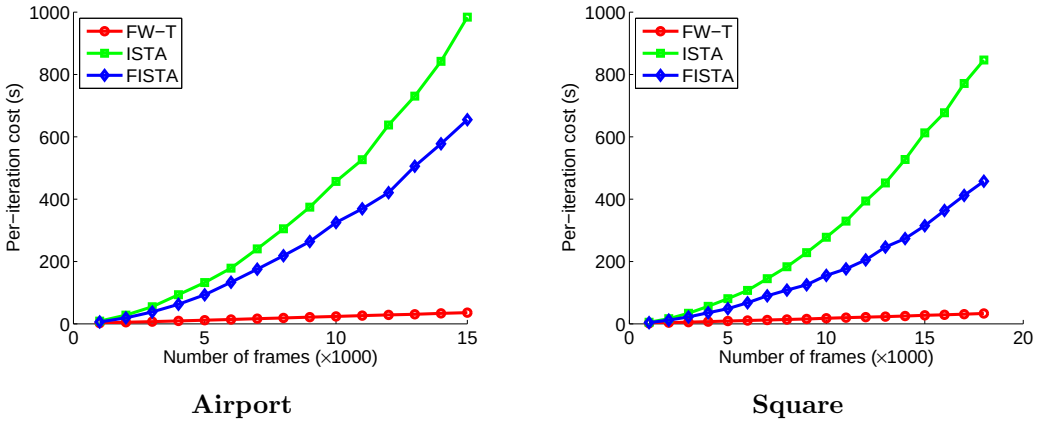

FIG. 2. ***Per-iteration cost vs. the number of frames in Airport and Square videos with full observation.*** *The per-iteration cost of our FW-T method grows linearly with the size of data, in contrast with the superlinear per-iteration cost of ISTA and FISTA. That makes the FW-T method more advantageous or may even be the only feasible choice for large problems.*

5.2. Foreground-background separation in surveillance video. In surveillance videos, due to the strong correlation between frames, it is natural to model the background as low rank; while foreground objects, such as cars or pedestrians, that normally occupy only a fraction of the video, can be treated as sparse. So, if we stack each frame as a column in the data matrix $\boldsymbol{M}_0$, it is reasonable to assume that $\boldsymbol{M}_0 \approx \boldsymbol{L}_0 + \boldsymbol{S}_0$, where $\boldsymbol{L}_0$ captures the background and $\boldsymbol{S}_0$ represents the foreground movements. Here, we solved problem (1.5) for videos introduced in [39] and [40]. The observed entries were sampled uniformly with ratio ρ chosen respectively as 1, 0.8 and 0.6.

Table 1 summarizes the numerical performances of FW-T, ISTA and FISTA in terms of the iteration number and running time (in seconds). As can be observed, our FW-T method is more efficient than ISTA and FISTA, and the advantage becomes more prominent as the size of the data grows and the observations are more compressed (with smaller sampling ratio ρ). Even though the FW-T method took more iterations than FISTA and in many cases than ISTA, it took less time in many cases but one due to its low per-iteration cost. To illustrate this more clearly, in Figure 2, we plot the per-iteration cost of these three methods on the Airport and Square videos as a function of the number of frames. The computational cost of FW-T scales linearly with the size of the data, whereas the cost of the other methods increases superlinearly. Another observation is that as the number of measurements decreases, the iteration numbers of both ISTA and FISTA methods grow substantially, while those of the FW-T method remain quite stable. This explains the more favorable behavior of the FW-T method when ρ is small. In Figure 3, frames of the original videos, the backgrounds and the foregrounds produced by the FW-T method are presented, and the separation achieved is quite satisfactory.

5.3. Shadow and specularity removal from face images. Images taken under varying illumination can also be modeled as the superposition of low-rank and sparse components. Here, the data matrix $\boldsymbol{M}_0$ is again formed by stacking each image as a column. The low-rank term $\boldsymbol{L}_0$ captures the smooth variations [41], while the sparse term $\boldsymbol{S}_0$ represents cast shadows and specularities [42, 8]. CPCP can

TABLE 1

Comparisons of FW-T, ISTA and FISTA on surveillance video data. *The advantage of our FW-T method becomes prominent when the data are at large scale and compressed (i.e. the small ρ scenarios).*

		FW-T		ISTA		FISTA	
Data	ρ	iter.	time	iter.	time	iter.	time
Lobby	1.0	96	1.94e+02	144	3.64e+02	41	**1.60e+02**
(20480 × 1000)	0.8	104	**2.33e+02**	216	1.03e+03	52	3.55e+02
	0.6	133	**3.12e+02**	380	1.67e+03	74	5.10e+02
Campus	1.0	45	**1.56e+02**	78	1.49e+03	23	4.63e+02
(20480 × 1439)	0.8	44	**1.57e+02**	122	2.34e+03	30	6.45e+02
	0.6	41	**1.39e+02**	218	4.27e+03	43	1.08e+03
Escalator	1.0	81	**7.40e+02**	58	4.19e+03	25	2.18e+03
(20800 × 3417)	0.8	80	**7.35e+02**	90	8.18e+03	32	3.46e+03
	0.6	82	**7.68e+02**	162	1.83e+04	43	5.73e+03
Mall	1.0	38	**4.70e+02**	110	5.03e+03	35	1.73e+03
(81920 × 1286)	0.8	35	**4.58e+02**	171	7.32e+03	44	2.34e+03
	0.6	44	**5.09e+02**	308	1.31e+04	62	3.42e+03
Restaurant	1.0	70	**5.44e+02**	52	3.01e+03	20	1.63e+03
(19200 × 3055)	0.8	74	**5.51e+02**	81	4.84e+03	26	1.82e+03
	0.6	76	**5.73e+02**	144	9.93e+03	38	3.31e+03
Hall	1.0	60	**6.33e+02**	52	2.98e+03	21	1.39e+03
(25344 × 3584)	0.8	62	**6.52e+02**	81	6.45e+03	28	2.90e+03
	0.6	70	**7.43e+02**	144	1.42e+04	39	4.94e+03
Airport	1.0	130	**6.42e+03**	29	2.37e+04	14	1.37e+04
(25344 × 15730)	0.8	136	**6.65e+03**	45	6.92e+04	18	4.27e+04
	0.6	154	**7.72e+03**	77	1.78e+05	24	7.32e+04
Square	1.0	179	**1.24e+04**	29	3.15e+04	13	1.51e+04
(19200 × 28181)	0.8	181	**1.26e+04**	44	1.04e+05	17	6.03e+04
	0.6	191	**1.31e+04**	78	2.63e+05	22	9.88e+05

FIG. 3. ***Surveillance videos.*** *The videos from top to bottom are respectively Lobby, Campus, Escalator, Mall, Restaurant, Hall, Airport and Square. The left panel presents videos with full observation ($\rho = 1$) and the right one presents videos with partial observation ($\rho = 0.6$). Visually, the low-rank component successfully recovers the background and the sparse one captures the moving objects (e.g. vehicles, pedestrians) in the foreground.*

be used to remove the shadows and specularities [3, 8]. Here, we solved problem (1.4) for YaleB face images [43]. Table 2 summarizes the numerical performances of FW-T, ISTA and FISTA. Similar to the observation made regarding the above surveillance video experiment, the number of iterations required by ISTA and FISTA grows much faster than it does for the FW-T method when ρ decreases. However, unlike in those tests, where the number of frames in each dataset was at least several thousand, the number of frames here is just 65. This prevents the FW-T method from significantly benefiting from its linear per-iteration cost and consequently, while FW-T still outperforms ISTA for values of $\rho \leq 0.7$, the FISTA method is always the

TABLE 2

Comparisons of FW-T, ISTA and FISTA on YaleB face data. *The number of frames, 65, is relatively small for this application. This disables the FW-T method to significantly benefit from its linear per-iteration cost and consequently the FISTA method consistently has a better performance.*

		FW-T		ISTA		FISTA	
Data	ρ	iter.	time	iter.	time	iter.	time
YaleB01	1.0	65	34.0	49	21.4	17	**8.69**
(32256×65)	0.9	68	35.6	59	23.9	19	**8.62**
	0.8	79	42.2	76	35.3	22	**10.9**
	0.7	76	39.9	97	44.0	25	**11.1**
	0.6	71	37.5	127	50.2	29	**12.9**
	0.5	80	40.5	182	77.9	35	**15.2**
YaleB02	1.0	64	34.6	51	19.2	18	**7.31**
(32256×65)	0.9	64	26.8	61	22.6	20	**7.32**
	0.8	71	33.9	78	27.7	22	**8.61**
	0.7	71	31.3	99	36.6	26	**11.0**
	0.6	73	36.6	132	53.7	30	**12.4**
	0.5	63	28.0	177	64.6	35	**13.4**
YaleB03	1.0	62	26.0	49	16.6	18	**6.00**
(32256×65)	0.9	71	27.5	62	20.3	20	**6.43**
	0.8	69	30.0	78	26.0	22	**8.32**
	0.7	78	31.5	101	32.9	26	**9.00**
	0.6	73	28.7	132	40.4	30	**10.6**
	0.5	70	28.0	181	60.3	36	**12.8**
YaleB04	1.0	63	28.5	47	16.6	17	**6.35**
(32256×65)	0.9	67	28.7	58	23.1	19	**7.98**
	0.8	68	31.7	72	26.3	23	**9.39**
	0.7	69	30.7	92	35.9	26	**9.84**
	0.6	71	29.4	124	40.0	29	**10.1**
	0.5	74	29.4	174	67.3	36	**14.3**

fastest. In Figure 4, the original images, the low-rank and the sparse parts produced by the FW-T method are presented. Visually, the recovered low-rank component is smoother and better conditioned for face recognition than the original image, while the sparse component corresponds to shadows and specularities.

6. Discussion. In this paper, we have proposed scalable algorithms called Frank-Wolfe-Projection (FW-P) and Frank-Wolfe-Thresholding (FW-T) for norm constrained and penalized versions of CPCP. Essentially, these methods combine classical ideas in Frank-Wolfe and Proximal methods to achieve linear per-iteration cost, $O(1/k)$ convergence in function value and practical efficiency in updating the sparse component. Extensive numerical experiments were conducted on computer vision related

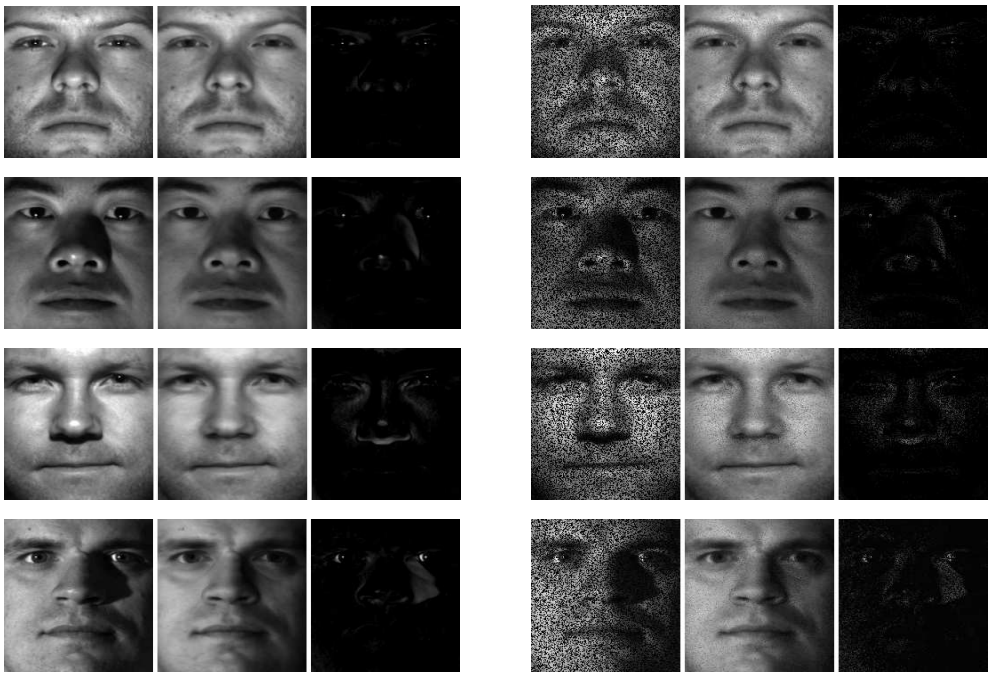

FIG. 4. ***Face images.*** *The pictures from top to bottom are respectively YaleB01, YaleB02, YaleB03 and YaleB04 face images. The left panel presents the case with full observation ($\rho = 1$), while the right panel presents the case with partial observation ($\rho = 0.6$). Visually, the recovered low-rank component is smoother and better conditioned for face recognition than the original image, while the sparse component corresponds to shadows and specularities.*

applications of CPCP, which demonstrated the great potential of our methods for dealing with problems of very large scale. Moreover, the general idea of leveraging different methods to deal with different functions may be valuable for other demixing problems.

We are also aware that though our algorithms are extremely efficient in the beginning iterations and quickly arrive at an approximate solution of practical significance, they become less competitive in solutions of very high accuracy, due to the nature of Frank-Wolfe. This suggests further hybridization under our framework (e.g. using nonconvex approaches to handle the nuclear norm) might be utilized in certain applications (see [44] for research in that direction).

Acknowledgements. We are grateful to the associate editor Chen Greif and three anonymous reviewers for their helpful suggestions and comments that substantially improve the paper.

REFERENCES

[1] J. Wright, A. Ganesh, K. Min, and Y. Ma, "Compressive principal component pursuit," *Information and Inference*, vol. 2, no. 1, pp. 32–68, 2013.

[2] V. Chandrasekaran, S. Sanghavi, P. Parrilo, and A. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, no. 2, pp. 572–596, 2011.

[3] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 11:1–11:37, 2011.
[4] Z. Zhou, X. Li, J. Wright, E. Candes, and Y. Ma, "Stable principal component pursuit," in *ISIT*, 2010.
[5] D. Hsu, S. Kakade, and T. Zhang, "Robust matrix decomposition with sparse corruptions," *IEEE Transactions on Information Theory*, vol. 57, no. 11, pp. 7221–7234, 2011.
[6] A. Agarwal, S. Negahban, and M. Wainwright, "Noisy matrix decomposition via convex relaxation: Optimal rates in high dimensions," *The Annals of Statistics*, vol. 40, no. 2, pp. 1171–1197, 2012.
[7] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2233–2246, 2012.
[8] Y. Zhang, C. Mu, H. Kuo, and J. Wright, "Towards guaranteed illumination models for non-convex objects," in *ICCV*, 2013.
[9] L. Wu, A. Ganesh, B. Shi, Y. Matsushita, Y. Wang, and Y. Ma, "Robust photometric stereo via low-rank matrix completion and recovery," in *ACCV*, 2011.
[10] R. Otazo, E. Candès, and D. K. Sodickson, "Low-rank plus sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic components," *Magnetic Resonance in Medicine*, 2014.
[11] K. Min, Z. Zhang, J. Wright, and Y. Ma, "Decomposing background topics from keywords by principal component pursuit," in *CIKM*, 2010.
[12] V. Chandrasekaran, P. Parrilo, and A. Willsky, "Latent variable graphical model selection via convex optimization," *Annals of Statistics*, vol. 40, no. 4, pp. 1935–1967, 2012.
[13] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," in *CAMSAP*, 2009.
[14] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *arXiv preprint arXiv:1009.5055*, 2010.
[15] X. Yuan and J. Yang, "Sparse and low-rank matrix decomposition via alternating direction methods," *preprint*, 2009.
[16] N. S. Aybat, D. Goldfarb, and G. Iyengar, "Fast first-order methods for stable principal component pursuit," *arXiv preprint arXiv:1105.2126*, 2011.
[17] M. Tao and X. Yuan, "Recovering low-rank and sparse components of matrices from incomplete and noisy observations," *SIAM Journal on Optimization*, vol. 21, no. 1, pp. 57–81, 2011.
[18] N. S. Aybat, D. Goldfarb, and S. Ma, "Efficient algorithms for robust and stable principal component pursuit problems," *Computational Optimization and Applications*, pp. 1–29, 2012.
[19] R. Tütüncü, K. Toh, and M. Todd, "Solving semidefinite-quadratic-linear programs using sdpt3," *Mathematical Programming*, vol. 95, no. 2, pp. 189–217, 2003.
[20] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.
[21] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.
[22] E. Levitin and B. Polyak, "Constrained minimization methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 6, no. 5, pp. 1–50, 1966.
[23] M. Jaggi and M. Sulovsk, "A simple algorithm for nuclear norm regularized problems," in *ICML*, 2010.
[24] Z. Harchaoui, A. Juditsky, and A. Nemirovski, "Conditional gradient algorithms for norm-regularized smooth convex optimization," *Mathematical Programming*, pp. 1–38, 2014.
[25] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *ICML*, 2013.
[26] V. F. Demianov and A. M. Rubinov, *Approximate methods in optimization problems*. Modern analytic and computational methods in science and mathematics, American Elsevier Pub. Co., 1970.
[27] J. C. Dunn and S. Harshbarger, "Conditional gradient algorithms with open loop step size rules," *Journal of Mathematical Analysis and Applications*, vol. 62, no. 2, pp. 432 – 444, 1978.
[28] M. Patriksson, "Partial linearization methods in nonlinear programming," *Journal of Optimization Theory and Applications*, vol. 78, no. 2, pp. 227–246, 1993.
[29] T. Zhang, "Sequential greedy approximation for certain convex optimization problems," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 682–691, 2003.
[30] K. Clarkson, "Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm," *ACM Trans. Algorithms*, vol. 6, no. 4, pp. 63:1–63:30, 2010.

[31] R. M. Freund and P. Grigas, "New analysis and results for the frank–wolfe method," *Mathematical Programming*, vol. 155, no. 1-2, pp. 199–230, 2016.
[32] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l_1-ball for learning in high dimensions," in *ICML*, 2008.
[33] J. Yang and Y. Zhang, "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM Journal on Scientific Computing*, vol. 33, no. 1, pp. 250–278, 2011.
[34] J. Yang and X. Yuan, "Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization," *Mathematics of Computation*, vol. 82, no. 281, pp. 301–329, 2013.
[35] J. Cai, E. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.
[36] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," *Mathematical Programming*, vol. 128, no. 1-2, pp. 321–353, 2011.
[37] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," in *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.
[38] R. M. Larsen, "Propack-software for large and sparse svd calculations," *Available online. URL http://sun. stanford. edu/rmunk/PROPACK*, pp. 2008–2009, 2004.
[39] L. Li, W. Huang, I. Y. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1459–1472, 2004.
[40] N. Jacobs, N. Roman, and R. Pless, "Consistent temporal variations in many outdoor scenes," in *CVPR*, 2007.
[41] R. Basri and D. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, 2003.
[42] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.
[43] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.
[44] S. Laue, "A hybrid algorithm for convex semidefinite optimization," in *ICML*, 2012.