Abstract
We examine the the problem of Robust Tensor Principal Component Analysis (TRPCA), a higher-order extension of the well-studied Robust Matrix PCA problem. In the robust setting, a low-rank tensor is perturbed by some sparse error tensor, and we wish to recover the original low-rank tensor. We propose an algorithm that tackles the Robust Tensor PCA problem via Canonical Polyadic-decomposition, and provide a mathematical and empirical justification of improvements in both runtime and robustness. As a byproduct, we also devise a framework for easily building TRPCA algorithms from prior RPCA algorithms. To study the performance of our algorithm, we also perform a comprehensive, standardized empirical analysis with various attacks to examine performance against state-of-the-art TPCA and TRPCA models. Our algorithm demonstrates significant improvements in various contexts (high-dimensions, gross corruption) where previous algorithms fail.

1. Introduction
Extracting low-dimensional structure from high-dimensional data has long been a central goal of machine learning. To this end, many techniques and algorithms ranging from Singular Value Decomposition to Non-negative Matrix Factorization have been devised to optimize for certain performance measures.

In this paper, we focus on an extension of Principal Component Analysis (PCA), known as Tensor Robust PCA (TRPCA) which has a specific emphasis on robustness in performance. As mentioned earlier, many machine learning problems often high-dimensional data; as such, tensor computation is natural to lift conventional methods and capitalize on relationships between dimensions. Notably, tensors are able to learn much higher-order relationships than matrices due to their structure. Some examples of tensor data include recommendation systems (Karatzoglou et al., 2010), image processing (Liu et al., 2012), and anomaly detection (Li et al., 2015). The matrix form of information in these tasks can be represented via a small number of factors – these are known as low-rank matrices, and can be written via principal component analysis.

Similar to conventional PCA, the task of Tensor PCA is to extract the variability and relationships within higher-dimensional data. Tensors can also have such a low-rank structure, but there are many different ways to decompose a tensor. There are two primary methods of tensor decomposition: Tucker decomposition and Canonical-Polyadic (CP) decomposition. Previous research has largely focused on the Tucker decomposition of tensors, as it is a generalized version of the CP decomposition. However, recent research has shown Tucker decomposition can often be heavily overparameterized (Hale & Prater-Bennette, 2021); as a result, there is value in studying CP-based tensor decomposition for the task of TPCA. Just as PCA is weak to data perturbation, such tensor decomposition can have infinitely high errors when a small fraction of the data is arbitrarily modified (Chen et al., 2021). Robust variants of PCA have been developed to combat these simple attacks, and similar variants of Tucker decomposition have also been developed. However, at the time of composition, there exist no algorithms using the CP decomposition that provide robustness guarantees.

In this paper, we propose a framework that allows us to build TRPCA algorithms from pre-existing RPCA algorithms. This framework is based on a variant of CP decomposition. We also design a comprehensive empirical study examining the robustness of existing TPCA and TRPCA methods.

Our algorithm lifts an earlier algorithm designed to solve the task of RPCA (Candès et al., 2009). From our choice of decomposition, our algorithm avoids the issues of overparameterization and also shows improved performance with higher levels of corruption. Notably, as a result of our tensor decomposition procedure, we lift many of the guarantees of the original RPCA method to the tensor counterpart. Our empirical study provides a unified look at the
We begin the paper with an overview of notation and definitions (Section 1). Then, we conduct a survey of existing methods in TRPCA and related fields of research (Section 2). In Section 3, we provide our explicit algorithm, as well as intuition and relevant guarantees. The following section contains various empirical studies studying the performance of our algorithm against existing studies. Finally, we provide a study and rudimentary analysis of runtime bounds of our algorithm. We then proceed to provide an empirical study of current state of the art Tensor Robust PCA (TRPCA) methods and how they compare to our proposed algorithm. Finally, we briefly demonstrate our overall algorithm’s iteration complexity using a more recent robust PCA algorithm.

1.1. Main Contributions

In this work, we examine the following corruption model. We begin with an unknown low-rank tensor $X_s \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$. Then, we add a $\alpha$-sparse tensor $S \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$. This final tensor $Y$ is what is visible to the algorithm:

$$Y = X_s + S$$

For simplicity, in our analyses, we will have $n_1 = n_2 = \ldots = n_d$, i.e. all tensor dimensions are equal. This will not necessarily hold in empirical experiments.

**Definition 1.1.** A mode-$j$ unfolding is defined by

$$y_{i_1 i_2 \ldots i_d} \rightarrow y_{j,k} \text{ with } k = 1 + \sum_{n=1 \atop n \neq j}^{d} ((i_n - 1) \prod_{m=1 \atop m \neq j}^{n-1} N_m)$$

**Algorithm 1 Unfolding Tensor**

**Input:** tensor $Y$

\[ \forall j \in \{1, \ldots, d\} \quad Y_{(j)} = M_j D(\odot_{k \neq j} M_k) + U_{(j)} \]

Qualitatively, unfolding can be understood as an extension of matrix vectorization lifted to a higher dimension. For a $d$-dimensional tensor, there exist $d$ unfoldings each reshaping a different fiber of the tensor. And in specific, a mode-$j$ unfolding reshapes the mode-$j$ fibers of the tensor into the columns of the resulting matrix.

**Definition 1.2.** The corruption tensor $S \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ is $\alpha$-sparse when it is such that for matrix $M$ built by unfolding $S$ on any dimension,

\[
\|M_{(i,\cdot)}\|_0 \leq \alpha n \forall i \in [n] \\
\|M_{(\cdot,j)}\|_0 \leq \alpha n \forall j \in [n]
\]

It should be noted that further restrictions on the input tensor may be applied depending on what robust PCA algorithm is used. We provide a standardized definition of $\alpha$-sparse error in Definition 1.2. In the runtime analysis provided later in this paper, we further restrict the input low-rank tensor via “incoherence.”

We propose a modification to CP decomposition that is empirically robust. Instead of applying PCA on unfolded matrices, as is performed in CP decomposition, the proposed algorithm uses robust PCA (Candès et al., 2009). As the original tensor is expected to be of low-rank, each unfolded tensor should also be low-rank (Zhang et al., 2021), so robust PCA can accurately separate out a sparse error matrix from the original unfolded tensor. Then, we can build out the CP decomposition via the eigenvalues and eigenvectors of these low-rank unfolded tensors.

Although there are pre-existing robust algorithms for Tucker decomposition (Dong et al., 2023), there are no such variations for CP-decomposition. There is significant value to devising such an algorithm: past empirical research indicates that CP decomposition via Alternating Least-Squares (ALS) is the most computationally efficient in term of runtime (Hale & Stuber-Bennette, 2021). When optimizing for other scenarios, such as low rank factors, CP decomposition methods indeed show comparable performance and superior efficiency. An intuition for such an argument follows naturally: Tucker decomposition-based models tend to be over-parameterized, which means they often require high amounts of data to build a factor model with low variance. CP-decomposition does not have such issues, so it is more suitable for tasks with high-dimensional data (or higher rank tensors). We also provide a detailed empirical study on various tensor robust PCA models in images, video, audio, and synthetic data. In general, prior literature exclusively shows their models’ performance on synthetic data or selective image data, so we hope to provide a more expansive review of these algorithms’ performance in the real world.

1.2. Related Work

1.2.1. Algorithms for Matrix RPCA

Matrix RPCA aims to extract out a low-dimensional matrix from an input matrix with some sparse error. Specifically, given corrupted matrix $Y = M + S$, the goal is to extract $M$ and $S$. Early approaches primarily relied on forms of convex relaxation by minimizing the sum of the nuclear norm of the data and the L1 norm of the error matrices (Cai et al., 2022). For example,

$$\min ||L||_* + \lambda ||S||_1$$

subject to $L + S = M$ (Candès et al., 2009).

These methods are very slow with high-dimensional prob-
Very early methods for tensor RPCA used a similar algorithm to what we present – unfolding the tensor and passing it directly into a matrix RPCA algorithm (Goldfarb & Qin, 2014). HoRPCA, proposed by Goldfarb, et. al. simply runs the convex optimization for input corrupted tensor $Y$:

$$\min_{X,E} \sum_{i=1}^{N} ||X_{(i)}|| + \lambda ||E||_1$$

subject to $X + E = Y$.

As one might expect, though, this has sub-optimal performance as many higher-order complex interactions get ignored. We get around this by running matrix RPCA on many different unfolded tensor modes.

More recent TRPCA models operate directly in the tensor space. However, this is significantly more complicated than in the matrix case. Firstly, tensor nuclear norm (TNN) is NP-hard to compute – this makes convex relaxation strategies less helpful (Friedland & Lim, 2018). We include an empirical analysis of one such algorithm in our paper.

Some recent work also uses scaled gradient descent. We include one such example with provable guarantees in our empirical study.

## 2. Analyzed TRPCA Models

### 2.1. Nuclear Norm (TRPCA-TNN, TRPCA-SNN)

The original RPCA paper devised a method involving the nuclear norm of matrices to invoke superior guarantees and bounds on performance and robustness (Candès et al., 2009). A natural direction of research in TRPCA has been to lift the nuclear norm to the tensor level, thereby lifting similar improvement in robustness. Initial attempts to lift the nuclear norm involved using the sum of the nuclear norms of the unfolded matrices as a regularization factor. However, this factor is actually extremely suboptimal, a fact that has been well-explored (Mu et al., 2014).

More recently, research has lifted the idea of tensor Singular Value Decomposition (t-SVD) to define a new tensor nuclear norm (not to be confused with matrix nuclear norm). The formulation of t-SVD requires a redefinition of tensor operations in a novel Fourier domain that is unrelated to other algorithms, so we leave a reference to the exact derivations. A qualitative representation of t-SVD is depicted in Figure 1.

A first paper by (Lu et al., 2018) approaches the problem of

---

**Figure 1. A Visualization of t-SVD (Lu et al., 2019)**

---
robust Tucker decomposition by establishing a rudimentary nuclear norm for TRPCA. Both our and the paper’s original experiments show insignificant improvement from this definition of nuclear norm. A follow-up paper (Lu et al., 2019) by the same group redefines the nuclear norm to simply be the sum of the entries of the diagonal tensor, which achieves improved guarantees as well as significant improvements in empirical results.

However, both papers suffer from several shortcomings. First, the guarantees on optimality are only provided for tensors of rank 3, which undermines the high-dimensional purpose of tensor operations. Furthermore, empirical experiments fail to examine performance degradation for higher levels of corruption. In subsequent sections we examine, in an empirical setting, how performance degrades with regards to rank, corruption level, and type of adversarial attack.

2.2. Gradient Descent (TRPCA-GD)

Other studies seek to lift advances in gradient descent to solve TRPCA. Specifically, Dong et al. examine the Tucker decomposition again via pruning and Scaled Gradient Descent (Scaled GD) (Dong et al., 2023) (Tong et al., 2021). The authors lift Scaled GD, which updates factor matrices independently and with different learning rates, to achieve guarantees on local linear convergence. Furthermore, they use corruption pruning to ablate certain error terms. Specifically, when calculating loss values for propagation:

\[ y = \left( U_{t}^{(1)}, U_{t}^{(2)}, U_{t}^{(3)} \right) \cdot g_{t} \]

the authors use a geometrically decaying operator \( T_{t+1} \) such that subsequent loss calculations are as follows:

\[ S_{t+1} = T_{t+1} \left( y - \left( U_{t}^{(1)}, U_{t}^{(2)}, U_{t}^{(3)} \right) \cdot g_{t} \right), \quad t = 0, 1, \ldots \]

Intuitively, as the model trains, more of the error term becomes influenced by noise, and thus error pruning provides robustness against noise. Moreover, this use of Scaled GD allows for parallelization which theoretically improves runtime of the algorithm.

In practice, however, issues with initialisation and convergence guarantees result in considerably slow performance for even rudimentary tasks. In fact, this algorithm provides a guarantee for constant-rate convergence at a higher-expense (Dong et al., 2023). Empirically, this per-iteration expense creates a 10-100 time slowdown compared to our algorithm, rendering it ineffective for time-sensitive tasks like image/video compression and audio processing. Additionally, the decaying operator must be carefully orchestrated for every scenario, and in our own empirical experiments required great deals of fine-tuning for effective performance.

3. Proposed Algorithm

CP decomposition is a special case of the Tucker decomposition with a diagonal core tensor. Specifically, it has much fewer parameters, so it is much faster to learn. As such, it is desirable to robustly learn the CP decomposition as opposed to the Tucker decomposition, as prior work has done.

Robust CP decomposition, described in Algorithm 2 involves 3 main steps. First, the input (perturbed) tensor is unfolded in each dimension. This unfolding process is described in Algorithm 1. Broadly, this is a generalization of vectorization, where the tensor is unfolded along each of \( d \) modes.

In standard CP decomposition, PCA is used to estimate \( M_j \) and \( \otimes_{k \neq j} M_k \) for each \( j \). In our algorithm we use Robust PCA instead, which lets us estimate the relevant low-rank unfolded tensor, which summed together with a sparse error matrix, gives \( M_j \). For completeness, this is detailed in Algorithm 3. From this low-rank unfolded matrix, it is simple to calculate the eigenvalues and eigenvectors.

4. Numerical Experiments

We conduct various experiments to compare the performance of our novel algorithm with standard algorithms, with a specific emphasis on robustness. In the context of robustness, the evolving nature of adversarial attacks often necessitates comprehensive experiments using modern attacks. However, given the large variety of optimization approaches used for TPCA, many attacks can only specific target certain algorithms. For instance, a gradient ascent attack may affect performance of TRPCA via Scaled GD, but will be completely ineffective against TRPCA via ALS. Moreover, simple attacks still expose differences in robustness especially at higher corruption levels.

Our experiments cover various implementations on standardized datasets the specific algorithms are described
• **TPCA** - A non-robust version of TPCA using CP-decomposition. (Babii et al., 2023)

• **TPCA-GD** - A robust Tucker-decomposition version of TPCA using error pruning and Scaled GD. (Dong et al., 2023)

• **TRPCA-TNN** - A robust Tucker-decomposition version of TPCA using the tensor nuclear norm. (Lu et al., 2019)

• **TRPCA-SNN** - A robust Tucker-decomposition version of TPCA using the matrix nuclear norm. (Lu et al., 2019)

• **CP-TRPCA** - Our algorithm leveraging RPCA methods to build a robust estimation of CP-decomposition.

To this end, we perform simple Gaussian noise injection over three datasets: synthetic numeric data, grayscale images, and video images. We also present a rudimentary adversarial attack in the context of facial recognition to provide a qualitative measure of performance.

Note that there are many tables where TRPCA-GD has empty data entries. These indicate cases where the gradient descent algorithm was unable to consistently converge in approximately 9 hours.

### 4.1. Synthetic Data

We begin with a synthetic data evaluation with randomly generated decompositions of multilinear rank \((5,10,\ldots,50)\) with corruption fraction \((0.1,0.2,0.3)\). First, a tensor of low multi-linear rank is generated, with all elements between \(-1000\) and \(1000\). The corrupted elements are selected uniformly at random, and all corrupted elements are random numbers between 900 and 1000.

Results are presented in Table 2 and 3, where the rank and corruption fractions are varied respectively. It should be noticed that our algorithm performs relatively better to at higher ranks, but worse at lower ranks. The main reason for this is that Tucker decomposition has a high number of parameters compared to CP-decomposition. Thus with such high rank, this leads to significant over-parametrization, and the Tucker decomposition models simply learn noise from the data.

In this experiment, we include results from non-robust TPCA (Babii et al., 2023) as proof that our contribution indeed has significant effect.

![Figure 2. Synthetic Data – Varying Rank](image1)

![Figure 3. Synthetic Data – Varying Error Sparsity](image2)

### 4.2. Image Denoising

We first consider a simple extension to practical application: denoising of images with Gaussian noise. We evaluate the performance of various TPCA methods in this context. Our data is prepared from MIRFLICKR25k (Huiskes & Lew, 2008), a dataset of natural images. We further set the images to be grayscale, and Gaussian noise is then randomly added. Qualitative examples are shown in Table 4.2.

For nuclear norm and TNN-based methods, we use a simple implementation of Alternating Direction Method of Multipliers (ADMM) for the optimization task, and training/error graphs are provided below. We run experiments on various levels of noise (see Tables 1 and 2), and compare the results of TRPCA methods over two metrics: Peak Signal-to-Noise ratio (PSNR), as well as Relative Squared Error (RSE). We choose these two metrics because they are natural to image denoising, and also because the intrinsic error/loss-functions vary in implementation and are hard to interpret.
Table 1. Varying % Gaussian Error’s effect on Mean RSE for various TRPCA methods – Image Recovery

<table>
<thead>
<tr>
<th></th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRPCA-GD</td>
<td>–</td>
<td>–</td>
<td>0.0073</td>
<td>0.00928</td>
</tr>
<tr>
<td>TRPCA-TNN</td>
<td>0.0032</td>
<td>0.0052</td>
<td>0.0072</td>
<td>0.0123</td>
</tr>
<tr>
<td>TRPCA-SNN</td>
<td>0.0030</td>
<td>0.0052</td>
<td>0.0071</td>
<td>0.0121</td>
</tr>
<tr>
<td>CP-TRPCA (OURS)</td>
<td>0.0029</td>
<td>0.0057</td>
<td>0.0091</td>
<td>0.0181</td>
</tr>
</tbody>
</table>

Table 2. Varying % Gaussian Error’s effect on Average PSNR for various TRPCA methods – Image Recovery

<table>
<thead>
<tr>
<th></th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRPCA-GD</td>
<td>–</td>
<td>–</td>
<td>38.893</td>
<td>30.002</td>
</tr>
<tr>
<td>TRPCA-TNN</td>
<td>47.8916</td>
<td>40.9723</td>
<td>37.2662</td>
<td>31.8766</td>
</tr>
<tr>
<td>TRPCA-SNN</td>
<td>45.4785</td>
<td>38.5519</td>
<td>30.8129</td>
<td>28.1065</td>
</tr>
<tr>
<td>CP-TRPCA (OURS)</td>
<td>47.032</td>
<td>43.232</td>
<td>33.120</td>
<td>25.129</td>
</tr>
</tbody>
</table>

For the task of image denoising, our algorithm indeed performs significantly worse than all state-of-the-art methods, including TRPCA-SNN, which has shown to have suboptimal guarantees in the tensor setting. However, note that the tensor representation of grayscale images are 2D, and therefore resembles the matrix RPCA task. Since both TNN and SNN hold strong convergence guarantees in the matrix RPCA task, their strong performance is natural. We hypothesize several factors may also explain suboptimal performance in low-dimensional tasks:

- Compared to the original implementation of TRPCA-GD, we add an unnecessary regularizing factor.
- The Tucker decomposition used in the three methods demonstrated may be more well-suited to this task.

4.3. Video Denoising

In this experiment, we examine performance on a 3D tensor. We use a grayscale format of the Videezy4K dataset (Guo et al., 2022). The grayscale operation is applied since performance guarantees for TRPCA-TNN and TRPCA-GD methods only hold for tensors of dimension three. Specifically, given $N$ grayscale images of dimension $(H, W)$, we concatenate them into a tensor of shape

$$(H \times W \times N)$$

The concatenation operation preserves the low-rank nature of the dataset since differences between images are miniscule. In other words, for our models, we assume a low-rank tensor structure along the video sequence, but not within each individual image. For these videos, we used multiple methods of adversarial perturbation.

The first attack involved random Gaussian noise added to the entire frame. An example frame is shown in Table 6. Results for these experiments are shown in Tables 4 and 5.

Our second simple attack took blocks of pixels out of the original video frames and replaced them with pure salt-and-pepper noise (i.e. either 0 or 1 values) – the same blocks were chosen in each frame of a given video. Some qualitative examples are shown in Table 7.

As mentioned earlier, our video procedure generates a 3D noisy tensor. Each tensor has $N = 12$, therefore this task is significantly more higher-dimension than the image denoising task. TRPCA-SNN, which uses the matrix nuclear norm, indeed performs significantly worse on this task. On the other hand, our algorithm now shows improvement over the RPCA-TNN method. Notably, performance is comparable for smaller corruption levels ([0.01, 0.05]) but a non-trivial discrepancy appears at higher ones ([0.1, 0.25]).

Furthermore, the second targeted attack (see (7)) indicates significant improvement over all previous TRPCA methods.
Table 4. Varying % Gaussian Error’s effect on Mean RSE for various TRPCA methods – Video Recovery

<table>
<thead>
<tr>
<th></th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRPCA-GD</td>
<td>–</td>
<td>–</td>
<td>0.0103</td>
<td>0.0118</td>
</tr>
<tr>
<td>TRPCA-TNN</td>
<td>0.0040</td>
<td>0.0094</td>
<td>0.0144</td>
<td>0.0264</td>
</tr>
<tr>
<td>TRPCA-SNN</td>
<td>0.0420</td>
<td>0.0936</td>
<td>0.1344</td>
<td>0.2244</td>
</tr>
<tr>
<td>CP-TRPCA (Ours)</td>
<td>0.0039</td>
<td>0.0093</td>
<td>0.0101</td>
<td>0.0172</td>
</tr>
</tbody>
</table>

Table 5. Varying % Gaussian Error’s effect on Average PSNR for various TRPCA methods – Video Recovery

<table>
<thead>
<tr>
<th></th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRPCA-GD</td>
<td>–</td>
<td>–</td>
<td>31.823</td>
<td>31.823</td>
</tr>
<tr>
<td>TRPCA-TNN</td>
<td>33.9924</td>
<td>29.1577</td>
<td>26.7644</td>
<td>23.0984</td>
</tr>
<tr>
<td>TRPCA-SNN</td>
<td>20.2631</td>
<td>15.7690</td>
<td>13.7657</td>
<td>11.1865</td>
</tr>
<tr>
<td>CP-TRPCA (Ours)</td>
<td>42.498</td>
<td>31.293</td>
<td>30.111</td>
<td>28.982</td>
</tr>
</tbody>
</table>

The block of concentrated noise corrupts the convergence of TRPCA-GD, leading to a blurry reconstruction. The block of noise remains for TRPCA-TNN, suggesting an inability to tackle such gross corruptions. Meanwhile, our algorithm faithfully reconstructs the noisy patch. Notably, the background info remains with decent fidelity, indicating that our low-rank representation still captures fine details.

4.4. Facial Interpolation

In this experiment, we present another rudimentary vision-based adversarial attack. Using the ORL database of facial data (Samaria & Harter, 1994), we perform simple interpolation between two faces with weights ranging from [0.5-1.0]. We then recover both the low-rank and sparse components from the result of our TRPCA algorithms.

4.5. Convergence

In addition to qualitative and quantitative measures, which measure the final performance of our presented algorithms, it is also insightful to study the training/iteration graphs which provide a crude reconstruction of the optimization landscapes of the algorithms. Sample plots are provided in the appendix. As expected, the optimization with sum of matrix nuclear norms indeed suggested a non-smooth landscape, as training follows a linear slope, pointing to errors such as algorithm stagnation or poor gradient information. In contrast, both experiments with the novel tensor nuclear norm as well as gradient descent suggested convergence in training.

5. Runtime Analysis

Unfortunately, the original Principal Component Pursuit algorithm (Candès et al., 2009) does not provide any concrete runtime/iteration bounds. As this specific robust PCA algorithm is what we use in our implementation, we cannot provide concrete runtime bounds for our proposed algorithm. However, a direction for future work would be to implement and integrate a robust PCA algorithm with such guarantees (Yi et al., 2016). As such, we provide a brief proof sketch of how such run-time bounds could be calculated.

Notice that our algorithm simply requires $d$ repeated calls to a robust PCA algorithm. We provide a very brief analysis using one such robust PCA algorithm. Relevant notation and definitions are included in the appendix.

For simplicity, we assume that $n_1 = n_2 = \ldots = n_d$.

**Lemma 5.1.** Let matrix $Y \in \mathbb{R}^{d \times d}$ be with decomposition $Y = M + S$, where $M$ is a rank $r$ matrix and $S$ is a sparse corruption matrix. Furthermore, $M$ is $\mu$-incoherent (defined in 5.2) and $S$ is $\alpha$-sparse matrix, defined in A.1. Then, Algorithm 4 runs in $O(rd^3 \log(1/\epsilon))$

**Definition 5.2.** Matrix $M$ is $\mu$-incoherent when: For any singular value decomposition $M = L \Sigma R^T$, $||L||_2,\infty \leq \sqrt{(\mu r)/d}$ $||R||_2,\infty \leq \sqrt{(\mu r)/d}$

**Definition 5.3.** Tensor $X \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ is $\mu$-incoherent when the unfolded tensor $X_{(i)}$ is always $\mu$-incoherent as a matrix: $\forall i \in [d]$, singular value decomposition for unfolded
Table 7. Qualitative results of video recovery after block of salt-and-pepper noise is inserted (1 frame visualized)

<table>
<thead>
<tr>
<th>Original</th>
<th>Perturbed</th>
<th>Recovery (TRPCA-GD)</th>
<th>Recovery (TRPCA-TNN)</th>
<th>Recovery (CP-TRPCA, Ours)</th>
</tr>
</thead>
</table>

Tensor along mode $i$ satisfies

$$||L||_{2,\infty} \leq \sqrt{\frac{(\mu r_i)}{n}}$$

$$||R||_{2,\infty} \leq \frac{\sqrt{\mu r_i}}{n}$$

**Theorem 5.4.** Suppose we have tensor $Y\in\mathbb{R}^{n_1\times n_2\times \ldots \times n_d}$ which is $\mu$-incoherent such that there exist $X\in\mathbb{R}^{n_1\times n_2\times \ldots \times n_d}$ where $X$ has multi-linear rank $r = (r_1, r_2, \ldots, r_d)$ and $S$ is $\alpha$-sparse tensor as defined in Definition 1.2. Then, Algorithm 2 can restore $X$, $S$ in time $O(\prod_{i=1}^{d} n_i^2 \log(1/\epsilon))$.

Theorem 5.4 follows directly from repeated usage of Lemma 5.1 as in Algorithm 2.

**6. Conclusion**

In this work, we have both proposed and analyzed a novel framework for CP-based TRPCA, as well as provided a reproducible framework for lifting and evaluating RPCA algorithms to TRPCA. We have also provided a variety of empirical experiments, including numerical experiments as well as image and video denoising, to measure the utility of existing algorithms in a variety of contexts.

Our results suggest that our algorithm shows marked improvements in robustness through a variety of adversarial attacks. Additionally, for higher-dimensional tasks, our algorithm shows improvement compared to Tucker-based TRPCA algorithms at higher levels of corruption, both in performance as well as runtime.

Directions for future work primarily include developing concrete correctness and runtime bounds for more robust PCA algorithms for various distributions. In addition, although our algorithm shows heuristic promise, further research can be conducted to examine performance with other tensor datasets. Although our paper has largely focused on performance numerically and in natural scenes, there exist other naturally noisy panel datasets where our algorithm for TRPCA would be applicable and beneficial.

Finally, we only present runtime bounds for one such RPCA algorithm. There are likely other algorithms which could lead to reduced computation (Yi et al., 2016), many of which may require further research into lifting RPCA techniques into higher dimensions.

**References**


Dong, H., Tong, T., Ma, C., and Chi, Y. Fast and provable tensor robust principal component analysis via scaled gradient descent, 2023.


Guo, S., Yang, X., Ma, J., Ren, G., and Zhang, L. A differentiable two-stage alignment scheme for burst image reconstruction with large shift. 2022.


A. Runtime Analysis Omitted Details

For completeness, we describe the Robust PCA algorithm used in Section 5 (Yi et al., 2016).

We denote the given corrupted matrix as \( Y = M^* + S^* \) where \( M^* \) is rank \( r \) and \( S^* \) is \( \alpha \)-sparse as defined below for matrices.

**Definition A.1.** Matrix \( M \) is \( \alpha \)-sparse when

\[
\| M(i, \cdot) \|_0 \leq \alpha n \quad \forall i \in [n]
\]

\[
\| M(\cdot, j) \|_0 \leq \alpha n \quad \forall j \in [n]
\]

It proceeds in two phases: initialization, then projected gradient descent. This initialization step is key in giving a final concrete iteration bound.

A sparse estimation operator is defined as follows. For any matrix \( A \in \mathbb{R}^{d_1 \times d_2}, \forall (i, j) \in [d_1] \times [d_2] \),

\[
\mathcal{T}_\alpha[A] := \begin{cases} A(i,j) & \text{if } |A(i,j)| \geq |A(i,\cdot)| \text{ and } |A(i,j)| \geq |A(\cdot,j)| \\ 0 & \text{otherwise} \end{cases}
\]

Finally, we provide the complete algorithm.

**Algorithm 4 Fast RPCA**

**Input:** Observed matrix \( Y \) with rank \( r \) and corruption fraction \( \alpha \). Parameters \( \gamma, \eta \). Number of iterations \( T \).

\( S_{\text{init}} \leftarrow \mathcal{T}_\alpha[Y] \)

\( [L, \Sigma, R] \leftarrow \text{SVD}(Y - S_{\text{init}}) \)

\( U_0 \leftarrow \Pi_U(L\Sigma^{1/2}), V_0 \leftarrow \Pi_V(R\Sigma^{1/2}) \)

for \( t = 0 \) to \( T - 1 \) do

\( S_t \leftarrow \mathcal{T}_\alpha[Y - U_tV_t^T] \)

\( U_{t+1} \leftarrow \Pi_U(U_t - \eta \nabla_U \mathcal{L}(U_t, V_t; S_t) - \frac{1}{2} \eta U_t(U_t^TU_t - V_t^TV_t)) \)

\( V_{t+1} \leftarrow \Pi_V(V_t - \eta \nabla_V \mathcal{L}(U_t, V_t; S_t) + \frac{1}{2} \eta V_t(U_t^TU_t - V_t^TV_t)) \)

end for

**Output:** \((U_T, V_T)\)

This algorithm notably runs in \( O(\kappa r d^2 \log(1/\epsilon)) \), but incoherence bounds allow for \( \kappa = O(1) \).

B. Brief Discussion on Correctness

The correctness bounds are entirely dependent on the RPCA algorithm that is used. In general, this will come down to just applying some sort of Union and Chernoff bound. In this work, we do not provide such an example – an interesting direction for future work would be clearly defining such bounds for various Robust PCA methods.
D. Facial Interpolation Experimental Results

<table>
<thead>
<tr>
<th>Interpolation Weight</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
<td>![Image]</td>
</tr>
</tbody>
</table>

*Table 9. Low-Rank Reconstruction Results*