


title

Jialei Wang  
University of Chicago  
jialei@uchicago.edu

Peder A. Olsen  
Andrew R. Conn  
Aurélie C. Lozano  
IBM T.J. Watson Research Center  
{pederao,arconn,aclozano}@us.ibm.com

Abstract

We consider the problem of removing and replacing clouds in satellite image sequences, which has a wide range of applications in remote sensing. Our approach first detects and removes the cloud-contaminated part of the image sequences. It then recovers the missing scenes from the clean parts using the proposed “TECROMAC” (TEmporally Contiguous ROBust MAtrix Completion) objective. The objective function balances temporal smoothness with a low rank solution while staying close to the original observations. The matrix whose the rows are pixels and columns are days corresponding to the image, has low-rank because the pixels reflect land-types such as vegetation, roads and lakes and there are relatively few variations as a result. We provide efficient optimization algorithms for TECROMAC, so we can exploit images containing millions of pixels. Empirical results on real satellite image sequences, as well as simulated data, demonstrate that our approach is able to recover underlying images from heavily cloud-contaminated observations.

1. Introduction

Optical satellite images are important tools for remote sensing, and suitable analytics applied to satellite images can often benefit applications such as land monitoring, mineral exploration, crop identification, etc. However, the usefulness of satellite images is largely limited by cloud contamination [18], thus a cloud removal and reconstruction system is highly desirable.

In this paper, we propose a novel approach for cloud removal in temporal satellite image sequences. Our method improves upon existing cloud removal approaches [17, 20, 21, 23, 24, 28, 32] in the following ways: 1) the approach does not require additional information such as a cloud mask, or measurements from non-optical sensors; 2) our model is robust even under heavy cloud contaminated situations, where several of the images could be entirely covered by clouds; 3) efficient optimization algorithms makes our approaches scalable to large high-resolution images.

2. Related work

Cloud detection, removal and replacement is an essential prerequisite for downstream applications of remote sensing. This paper belongs to the important line of research of sparse representation, which has received considerable attention in a variety of areas, including noise removal, inverse problems, and pattern recognition [13, 33, 38]. The key idea of sparse representation is to approximate signals as a sparse decomposition in a dictionary that is learnt from the data (see [25] for an extensive survey). For the task of cloud removal and image reconstruction, [23] developed compressive sensing approaches to find sparse signal representations using images from two different time points. Subsequently [20] extended dictionary learning to perform multitemporal recovery from longer time series. A group-sparse representation approach was recently proposed by [19] to leverage both temporal sparsity and non-

1For example in 8-bit representations (0,0,0) is typically black while (255,255,255) is white, so clouds can be assumed to have high pixel intensity.
local similarity in the multi-temporal images. A method was proposed in [20] to co-train two dictionary pairs, one pair generated from the high resolution image (HRI) and low resolution image (LRI) gradient patches, and the other generated from the HRI and synthetic-aperture radar (SAR) gradient patches. It is demonstrated that such a combination of multiple data types improves reconstruction results as it is able to provide both low- and high-frequency information.

Our method has its origin in robust principal component analysis, [5, 12] and matrix completion [6, 7]. However, these models require uniform or weighted sampled observations to ensure the recovery of low-rank matrix [14, 29, 34], and thus cannot handle images with extensive cloud cover.

3. Problem description and approach

Let \( Y \in \mathbb{R}^{m \times n \times c \times t} \) be a 4-th order tensor whose entry values include the intensity and represent the observed cloudy satellite image sequences. The dimensions correspond to latitude, longitude, spectral wavelength (color) and time. The images have size \( m \times n \) with \( c \) colors\(^2\) and there are \( t \) time points corresponding to each image. Our goal is to remove the clouds, and recover the background scene, to facilitate applications such as land monitoring, mineral exploration and agricultural analytics.

As already mentioned, our cloud removal procedure is divided into two stages: namely **cloud detection** followed by the **underlying scene recovery**. The cloud detection stage aims to provide an indication as to what pixels are contaminated by clouds at a particular time. Given the cloud mask, the recovery stage attempts to recover the obfuscated scene by leveraging the partially observed, non-cloudy image sequences. Figure 1 provide an illustration of the proposed two-stage framework. We describe the details of these two stages in the following sections.

\(^2\)A standard digital image will have a red, green and blue channel. A grayscale image has just one channel

---

2755

---

3.1. Detecting clouds

Given the observed image sequences \( Y \), the goal of a cloud detector is to produce a set of indices \( \Omega \), called a cloud mask, such that a latitude-longitude-color-time position \((i, j, k, l) \in \mathbb{Z}_m \times \mathbb{Z}_n \times \mathbb{Z}_c \times \mathbb{Z}_t\) is covered by a cloud if \((i, j, k, l) \notin \Omega\) and is not cloud contaminated if \((i, j, k, l) \in \Omega\). Note that typically the entire color channel will be contaminated by clouds.

We design a simple, yet effective cloud detector by thresholding on the dark channel (the minimum pixel intensity values across the RGB channels), [16] followed by a post-processing stage to distinguish between a stationary white background and white clouds. This works because clouds are predominantly white so that the minimum intensity value is large and there are typically no other white objects on the ground (except for snow in winter, please refer to section 3.1.1 below for a related discussion). Our estima-
tor for $\Omega$ is therefore
\[ \hat{\Omega} = \{(i, j, k, l)\mid \min_{c=(r,g,b)} Y(i, j, c, l) < \gamma \} , \] (1)
where $\gamma$ is the thresholding value that controls the trade-offs between false positives/negatives. Figure 2 illustrates the above described method.

There are several other approaches for cloud detection, for example, one could use support vector machines [11] to build linear classifiers based on the RGB values. However, this approach requires additional labeled training data, which is usually not easy to obtain. Moreover, in our experience we observed that the classification approach is typically no better than the simple thresholding method.

### 3.1.1 Post processing after thresholding

The thresholding approach described above cannot distinguish between a stationary “white background” and “white clouds”. White objects in the landscape, such as houses, should remain in the reconstructed images. Fortunately, we can exploit the transient nature of clouds to label pixels always absent from $\Omega$ as having a white background. Since the stationary pixels are also contaminated by clouds we maintain the background by treating the median pixel value as the background color and locate some cloud free instances by a k-nearest neighbors search [1]. This approach is primitive, but since the stationary white objects do not dominate the scene, the method is sufficient (though it will not be able to handle extreme cases such as heavy snow as background).

The overall procedure for cloud detection is summarized in Algorithm 1 below.

#### Algorithm 1: Cloud detection procedure.

**Input:** temporal sequence of cloudy images $Y$; parameter $K$ in k-nearest-neighbors search.

**Output:** $\hat{\Omega}$: indication set of non-cloudy pixels.

1. Obtain the initial guess via thresholding as in (1), $\Omega = \hat{\Omega};$
2. Identify the “always white” pixel sequences: $\mathcal{W} = \{(i, j)\mid \forall k, l \quad (i, j, k, l) \not\in \Omega\};$
3. for $(i, j) \in \mathcal{W}$ do
   1. Compute the median pixel vector: $m = \text{Median}(\{Y(i, j, :, l), l \in [t]\});$
   2. Find the indices of the k-nearest-neighbors:
      \[ \hat{\Omega} = \text{Knn-Search}(\{Y(i, j, :, l), l \in [t]\}, m, K); \]
   end

### 3.2. Image sequences recovery

Given the cloud detection result $\Omega$, the image sequences recovery model reconstructs the background from the partially observed noisy images. For pixels in $\Omega$ the recovered values should stay close to the observations. Also, the reconstruction should take the following key assumptions/intuitions into consideration:

- **Low-rank:** The ground observed will consist of a few slowly changing land-types such as forest, agricultural land, lakes and a few stationary objects (roads, buildings, etc.). We can interpret the land-types as basis elements of the presumed low-rank reconstruction. More complex scenes would thus require a higher rank, but we expect in general the number of truly independent frames (pixels evolving over time) to be relatively small.

- **Robustness:** Since the cloud detection results are not perfect, and there is typically a large deviation between the cloudy pixels and the background scene pixels, the recovery model should be robust with respect to the (relatively) sparsely corrupted observations.

- **Temporal continuity:** The ground contains time-varying objects with occasional dramatic changes (e.g. harvest). Most of the time the deviation between two consecutive frames should not be large given appropriate time-steps (days or weeks).

**Notation:** Let the observation tensor be reshaped into a matrix $Y \in \mathbb{R}^{mn \times ct}$ defined by $Y_{u v} = Y_{i j k l}$ when $u = i + jm$ and $v = k + lc$. Similarly, let $X$ be the reconstruction we are seeking and $X \in \mathbb{R}^{mn \times ct}$ be the corresponding reshaped matrix. Further we let $I$ denote the indicator function. Based on above discussed principles we propose the following reconstruction formulation:
\[
\min_X \text{rank}(X) \\
\text{s.t.} \quad \sum_{(i, j, k, l) \in \Omega} I\left(X(i, j, k, l) \neq Y(i, j, k, l)\right) \leq \kappa_1, \\
\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{c} \sum_{l=2}^{t} \left(\left\|X(i, j, k, l) - X(i, j, k, l - 1)\right\|^2\right) \leq \kappa_2.
\]

The above formulation finds a low-rank reconstruction $X$ such that:

- $X$ disagrees with $Y$ at most $\kappa_1$ times in the predicted non-cloudy set $\Omega$, but can disagree an arbitrary amount on the $\kappa_1$ pixels (robustness);
- The sum of squared $\ell_2$ distances between two consecutive frames is bounded by $\kappa_2$ (continuity).
Unfortunately, both the rank and the summation of the indicator function values are non-convex, making the above optimization computationally intractable in practice, thus we introduce our temporally contiguous robust matrix completion (TECROMAC) objective which is computationally efficient, first introducing the forward temporal shift matrix, \( S_+ \in \mathbb{R}^{t \times t} \) defined as

\[
[S_+]_{i,j} = \begin{cases} 1 & \text{if } i = j + 1, i < t \\ 1 & \text{if } i = j = t \\ 0 & \text{otherwise} \end{cases}
\]

and the discrete derivative matrix \( R \in \mathbb{R}^{t \times t} \) as

\[
R = \begin{bmatrix} I_1 - S_+ \\ I_1 - S_+ \\ \vdots \\ I_1 - S_+ \end{bmatrix}
\]

Let \( \| \cdot \|_F \) denote the Frobenius norm, the TECROMAC objective can be written as:

\[
\min_{X} \| P_{\Omega}(Y - X) \|_1 + \lambda_1 \| X \|_+ + \frac{\lambda_2}{2} \| XR \|_F^2, \tag{2}
\]

where \( \lambda_1 \) controls the rank of the solution and \( \lambda_2 \) penalizes large temporal changes.

In the objective (2), the first term \( \| P_{\Omega}(Y - X) \|_1 \) controls the reconstruction error on the predicted non-cloudy set \( \Omega \). The second term encourages low rank solutions, as the nuclear norm is a convex surrogate for direct rank penalization [36]. Noting that

\[
\| XR \|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{c} \sum_{l=2}^{t} (X(i, j, l, k) - X(i, j, l, k - 1))^2,
\]

is the finite difference approximation of the temporal derivatives one can see that the last term encourages smoothness between consecutive frames.

### 3.3. Existing reconstruction model paradigms

Following the cloud detection stage, there are several existing models reconstruction paradigms that could be applied. For example, one could use

- Interpolation [27]; although this violates the low-rank and robust assumptions
- Matrix completion (MC) [7]; which violates robustness and continuity.
- Robust matrix completion (RMC) [8]; which uses \( \| \cdot \|_1 \) loss instead of \( \| \cdot \|_F^2 \) to ensure robustness. However, it still violates the continuity assumption. RMC is an extension to MC and is inspired by robust principal component analysis (RPCA) [5], and has been analyzed theoretically [8, 9].

MC and RMC both require each image to have some pixels not corrupted by clouds to ensure successful low-rank matrix recovery [14, 29, 34]. Unfortunately, this is often not the case, as seen in Figure 3. Tropical regions for example can be completely covered by clouds as often as 90% of the time. Consequently, as we also show in the experiments below, directly applying MC or RMC will lead to significant information loss for the heavily contaminated frames. Table 1 summarizes the properties of the different approaches.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Interpolation</th>
<th>MC</th>
<th>RMC</th>
<th>TECROMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-rank</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Robustness</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Continuity</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1. Comparison of properties of various approaches.

One could also use RPCA [5] or stable principle component pursuit (SPCP) [39] directly on cloudy image sequences [2], without the cloud detection stage. However, these approaches only work for images with sparse cloud, and usually fail under more challenging dense cloudy cases, as we have observed in practice.

### 4. Optimization algorithms

In this section, we present computationally efficient algorithms to solve the optimization problem (2). The optimization is challenging since the objective (2) contains two non-smooth terms (the \( \ell_1 \) term and the nuclear norm term). Our overall algorithms are based on an augmented Lagrangian methods (ALM) [4, 22, 30], with special designs for the subproblem solvers.

First, we re-write (2) in the equality constrained form:

\[
\min_{E,X} \| P_{\Omega}(E) \|_1 + \lambda_1 \| X \|_+ + \frac{\lambda_2}{2} \| XR \|_F^2, \tag{3}
\]

\[
es.t. Y = X + E. \tag{4}
\]

The augmented Lagrangian method (ALM) tackles the inequality constraints indirectly by introducing Lagrange multipliers and a quadratic penalty to control deviations...
from the equality constraint. The unconstrained objective in our case is

\[
L(X, E, Z, \mu) = \|P_1(E)\|_1 + \lambda_1 \|X\|_* + \frac{\lambda_2}{2} \|XR\|_F^2 + \langle Z, Y - X - E\rangle + \frac{\mu}{2} \|Y - X - E\|_F^2,
\]

where \(Z\) are the Lagrange multipliers corresponding to the equalities and \(\mu\) is the quadratic penalty weight. A quadratic penalty function alone requires that the penalty parameter \(\mu\) tends to infinity whereas the Lagrangian term allows one to update the Lagrangian multipliers, thereby enabling the penalty parameter to remain finite. The general framework to update the Lagrangian multipliers, thereby enabling the penalty parameter to remain finite. The general framework of ALM works as follows: at each iteration \(r\) we solve the following problem:

\[
\{X^{r+1}, E^{r+1}\} = \arg\min_{X, E} L(X, E, Z^r, \mu^r). \tag{5}
\]

Then we update the dual variables \(Z\) as

\[
Z^{r+1} = Z^r + \mu^r(Y - X^{r+1} - E^{r+1}), \tag{6}
\]

and also increase the penalty parameter \(\mu\) via

\[
\mu^{r+1} = \rho \mu^r. \tag{7}
\]

The update for the multipliers, (6), can be determined by differentiating the augmented Lagrangian with respect to \(Z\) and comparing coefficients with a Lagrangian form.

The augmented Lagrangian method is often efficient and robust, however, each iteration involves solving the subproblem (5). We describe two inexact algorithms to accelerate the optimization process: the inexact proximal gradient (IPG), and the alternating minimization (ALT) algorithms, both of which use shrinkage operators [3, 31].

4.1. Shrinkage Operators

We write \(S_\lambda(X)\) for the elementwise soft-thresholding operators, \([S_\lambda(X)]_{ij} = \text{sign}(x_{ij})(|x_{ij}| - \lambda)_+\) and the singular value thresholding \(\text{SVT}_\eta(X) = U \Sigma \eta \Sigma V^T\), where \(X = U \Sigma V^T\) is the singular value decomposition of \(X\).

4.2. Inexact Proximal Gradient

In the inexact proximal gradient algorithm (which is summarized in Algorithm 2), we only update \(X\) and \(E\) once to avoid many expensive SVD computations. Given the current \(E^r, Z^r, \mu^r\), minimizing the following objective with respect to \(X^r\):

\[
L(X) = \lambda_1 \|X\|_* + f(X), \quad \text{where}
\]

\[
f(X) = \frac{\lambda_2}{2} \|XR\|_F^2 + \langle Z^r, Y - X - E^r\rangle + \frac{\mu^r}{2} \|Y - X - E^r\|_F^2,
\]

\[
\text{Algorithm 2: ALM-IPG: Inexact Proximal gradient method for optimizing (4).}
\]

**Input:** Observation set \(\Omega\) and data \(Y\); regularization parameters \(\lambda_1, \lambda_2\).

**Output:** Recovered matrix \(X\).

**while not converged do**

| Updating \(X\) using (8) ; |
| Updating \(E\) using (9) ; |
| Updating \(Z\) using (6) ; |
| Updating \(\mu\) using (7). |

**end**

Since \(f(X)\) is quadratic we can also write it in the following forms

\[
f(X) = \frac{1}{2} \text{trace}(XHX^T) + \text{trace}(G^T X) + f(0)
\]

\[
= \frac{1}{2} \|XA - B\|_F^2 + f(0) - \|B\|_F^2,
\]

where the Hessian at 0 is represented by \(H = \lambda_2 RR^T + \mu I\) and the gradient at 0 is \(G = -Z - \mu(Y - E), AA^T = H\) and \(B = -GA^T\).

Introducing the auxiliary function

\[
Q(X, \hat{X}) = F(X) + \frac{c}{2} \|X - \hat{X}\|_F^2 - \frac{1}{2} \|X - \hat{X}\|_F^2
\]

we see that \(Q(X, X) = F(X)\) and \(Q(X, \hat{X}) \geq F(X)\) if \(cI \succ AA^T = H\). We can rewrite \(Q(X, \hat{X})\) to clarify how we can efficiently optimize the function with respect to \(X\). We have

\[
Q(X, \hat{X}) = \lambda_1 \|X\|_* + \frac{c}{2} \|X - \hat{X}\|_F^2 - \text{trace} \left( X(AB^T - AA^T \hat{X} + c) \right) + \text{const}
\]

Completing the square gives

\[
Q(X, \hat{X}) = \lambda_1 \|X\|_* + \frac{c}{2} \|X - V\|_F^2 + \text{const}
\]

where

\[
V = \hat{X} + \frac{1}{c} (AB^T - AA^T \hat{X}) = \hat{X} - \frac{1}{c} \nabla f(\hat{X}).
\]

It follows that the optimum is achieved for

\[
X = \text{SVT}_{\lambda_1/c} \left( \hat{X} - \frac{1}{c} \nabla f(\hat{X}) \right). \tag{8}
\]

Since the optimum with respect to \(\hat{X}\) is given by \(\hat{X} = X\) this gives the iteration scheme

\[
X^{r+1} = \text{SVT}_{\lambda_1/c} \left( X^r - \frac{1}{c} \nabla f(X^r) \right).
\]

Input: Observation set Ω and data Y; regularization parameter \( \lambda_1, \lambda_2 \), stepsize \( \eta \).
Output: Recovered matrix \( X \).

\[ \text{while not converged} \]
\[ \quad \text{while not converged do} \]
\[ \quad \quad \text{Updating } U \text{ using (11)} ; \]
\[ \quad \quad \text{Updating } V \text{ using (12)} ; \]
\[ \quad \quad \text{Updating } E \text{ using (9)} ; \]
\[ \quad \text{end} \]
\[ \quad \text{Updating } Z \text{ as (6)} ; \]
\[ \quad \text{Updating } \mu \text{ as (7)} . \]
end

\[ \nabla f(X^r) = \lambda_2 X^r R R^\top - Z^r - \mu^r (Y - X^r - E^r) . \]

The objective with respect to \( E \) is:

\[ \min_E \| P_{\Omega}(E) \|_1 + \langle Z^r, Y - X^{r+1} - E \rangle + \frac{\mu^r}{2} \| Y - X^{r+1} - E \|_F^2 , \]

with the closed form solution

\[ E_{ij} = \begin{cases} S_{1/\mu}(Y_{ij} - X_{ij}^{r+1} + \frac{1}{\mu^r} Z_{ij}^r) & \text{if } (i, j) \in \Omega \\ Y_{ij} - X_{ij}^{r+1} + \frac{1}{\mu^r} Z_{ij}^r & \text{if } (i, j) \notin \Omega. \end{cases} \]

4.3. Alternating Minimization

In this section we describe the alternating least squares approach (summarized in Algorithm 3) that is SVD free. This idea is based on the following key observation about the nuclear norm [15, 35] (see also a simple proof at [37]):

\[ \|X\|_* = \min_{X=U V^\top} \frac{1}{2} (\| U \|_F^2 + \| V \|_F^2) . \]

The paper [15] suggested combining the matrix factorization equality (10) with the SOFT-IMPUTE approach in [26] for matrix completion.

For our TECROMAC objective, instead of directly optimizing the nuclear norm, we can use the matrix factorization identity to solve the following ALM subproblem:

\[ \{ U^{r+1}, V^{r+1}, E^{r+1} \} = \arg \min_{U, V, E} L(U, V, E, Z^r, \mu^r) , \]

where

\[ L(U, V, E, Z, \mu) = \| P_{\Omega}(E) \|_1 + \frac{\lambda_1}{2} \| U \|_F^2 + \frac{\lambda_1}{2} \| V \|_F^2 + \frac{\lambda_2}{2} \| U V^\top R \|_F^2 + \langle Z, Y - U V^\top - E \rangle + \frac{\mu}{2} \| Y - U V^\top - E \|_F^2 . \]

Table 2. Relative reconstruction error and times of various approaches

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Escalator</th>
<th>Lobby</th>
<th>Watersurface</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>0.2760</td>
<td>0.2677</td>
<td>0.9880</td>
</tr>
<tr>
<td>RMC</td>
<td>0.2155</td>
<td>0.2465</td>
<td>0.9127</td>
</tr>
<tr>
<td>Interpolation</td>
<td>0.0489</td>
<td>0.1151</td>
<td>0.0307</td>
</tr>
<tr>
<td>TECMAC</td>
<td>0.0481</td>
<td>0.0646</td>
<td>0.0461</td>
</tr>
<tr>
<td>TECROMAC</td>
<td>0.0153</td>
<td>0.0246</td>
<td>0.0045</td>
</tr>
</tbody>
</table>

Table 3. Comparison of accuracy and efficiency of two optimization algorithms to solve TECROMAC

We alternate minimizing over \( U, V \) and \( E \) to optimize the objective \( L(U, V, E, Z, \mu) \). To update \( U \) and \( V \), we perform a gradient step:

\[ U^{r+1} = U^r - \eta \nabla_U L(U^r, V^r, E^r, Z^r, \mu^r) \]
\[ V^{r+1} = V^r - \eta \nabla_V L(U^{r+1}, V^r, E^r, Z^r, \mu^r) , \]

where

\[ \nabla_U L(U, V, E, Z, \mu) = \lambda_1 U + \lambda_2 U V^\top R R^\top V - Z V - \mu (Y - U V^\top - E) \]
\[ \nabla_V L(U, V, E, Z, \mu) = \lambda_1 V + \lambda_2 R R^\top V U^\top U - Z^\top U - \mu (Y - U V^\top - E)^\top U . \]

After obtaining \( U^{r+1}, V^{r+1} \), we update \( E \) by substituting \( X^{r+1} = U^{r+1}(V^{r+1})^\top \) into (9).

5. Simulation and quantitative assessment

In this section we present extensive simulation results for our proposed algorithm. We manually added clouds to some widely used background modeling videos\footnote{http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html}. Since for the simulation we have access to the ground truth \( Y^* \) (i.e. the cloud-free video) we can report the relative reconstruction error (RRE) for our estimation \( \hat{Y} \) (shown in Table 2):

\[ \text{RRE}(\hat{Y}, Y^*) = \frac{\| \hat{Y} - Y^* \|_2}{\| Y^* \|_2} . \]

The thresholding parameter \( \gamma \) was set to 0.6, the regularization parameters \( \lambda_1 \) and \( \lambda_2 \) were set to be 20 and 0.5,
Figure 4. Comparisons of various approaches for cloud removal. 1. Original video. 2. Cloudy video. 3. Cloud detector. 4. Matrix Completion. 5. Robust Matrix Completion. 6. Interpolation. 7. TECMAC. 8. TECROMAC.
respectively. These parameter values were used for the experiments in the next section also. The cloud removal and reconstruction results were shown in Figure 4 (best viewed in Adobe Reader). We make the following observations:

- The cloud detection algorithm performs well and can handle the “white background” cases (e.g. some places in the escalator video) reasonably well;

- On some heavily cloud-contaminated frames, MC and RMC just output black frames. The temporally contiguous matrix completion approaches handles this issue very well;

- **TECROMAC** slightly outperforms **TECMAC** (matrix completion with temporal continuity constraint added) in all cases, which verifies that using the robust $\ell_1$ loss is a preferred choice.

We also compare the performance and time efficiency of the proposed computational algorithms, as summarized in Table 3, where for ALM-ALT algorithm we use rank 20. As we can see, ALM-ALT generally is more efficient than ALM-IPG, while slightly sacrificing the recovery accuracy by adopting a non-convex formulation.

6. Experiments on satellite image sequences

We also tested our algorithm on the real world MODIS satellite data, where we chose a subset consists of 181 sequential images of size $400 \times 400$. The results are shown in Figure 5 (best viewed in Adobe Reader). We observe that our algorithm recovers the background scenes from the cloudy satellite images very well, and visually produces much better recovery than existing models.

7. Conclusion

We have presented effective algorithms for cloud removal from satellite image sequences. In particular, we proposed the “TECROMAC” (TEmporally Contiguous RO-bust MAtrix Completion) approach to recover scenes of interest from partially observed, and possibly corrupted observations. We also suggested efficient optimization algorithms for our model. The experiments demonstrated superior performance of the proposed methods on both simulated and real world data, thus indicating our framework potentially very useful for downstream applications of remote sensing with clean satellite imagery.

5Publicly available http://modis.gsfc.nasa.gov/data/

6For more experimental results, please refer to the supplemental file.
References


