Abstract—Robust high-dimensional data processing has wit-
nessed an exciting development in recent years, as theoretical
results have shown that it is possible using convex programming
to optimize data fit to a low-rank component plus a sparse
outlier component. This problem is also known as Robust
PCA, and it has found application in many areas of computer
vision. In image and video processing and face recognition, an
exciting opportunity for processing of massive image databases
is emerging as people upload photo and video data online in
unprecedented volumes. However, the data quality and consist-
tency is not controlled in any way, and the massiveness of the
data poses a serious computational challenge. In this paper we
present t-GRASATA, or “Transformed GRASATA (Grassmannian
Robust Adaptive Subspace Tracking Algorithm)”. t-GRASATA
performs incremental gradient descent constrained to the
Grassmann manifold of subspaces in order to simultaneously
estimate a decomposition of a collection of images into a low-
rank subspace, a sparse part of occlusions and foreground
objects, and a transformation such as rotation or translation of
the image. We show that t-GRASATA is $4 \times$ faster than state-of-
the-art algorithms, has half the memory requirement, and can
achieve alignment for face images as well as jittered camera
surveillance images.

I. INTRODUCTION

With the explosion of image and video capture, both for
surveillance and personal enjoyment, and the ease of putting
these data online, we are seeing photo databases grow at
unprecedented rates. On record we know that in July 2010
Facebook had 100 million photo uploads per day [24] and
Instagram had a database of 400 million photos as of the
end of 2011, with 60 uploads per second [18]; since then
both of these databases have certainly grown immensely. In
2010, there were an estimated minimum 10,000 surveillance
cameras in the city of Chicago and in 2002 an estimated
500,000 in London [1], [23].

These enormous collections pose both an opportunity and
a challenge for image processing and face recognition: The
opportunity is that with so much data, it should be possible
to assist the users in tagging photos, searching of the image
database, and detecting unusual activity or anomalies. The
challenge is that the data are not controlled in any way so
as to ensure data quality and consistency across photos, and
the massiveness of the data poses a serious computational
challenge.

In video surveillance, many recently proposed algorithms
model the foreground and background separation problem as
one of “Robust PCA”– decomposing the scene as the sum
of a low-rank matrix of background, which represents the
global appearance and illumination of the scene, and a sparse
matrix of moving foreground objects [7] [15] [22] [29] [12].
These popular algorithms and models work very well for a
stationary camera. However, in the case of camera jitter, the
background is no longer low-rank, and this is problematic
for the Robust PCA methods [19] [27] [28]. Robustly and
efficiently detecting the moving objects from an unstable
camera is a challenging problem, since we need to accurately
estimate both the background and the transformation of each
frame. Fig. 1 shows that for a video sequence generated by a
simulated unstable camera, GRASATA [14] [15] (Grassman-
nian Robust Adaptive Subspace Tracking Algorithm) fails
to do the separation, but the approach we propose here, t-
GRASATA, can successfully separate the background and the
moving objects despite camera jitter.

Further recent work has extended the Robust PCA model
to that of the “Transformed Low-Rank + Sparse” model for
face images with occlusions under transformations such as
translations and rotations [25], [26], [30], [32]. Without the
transformations, this can be posed as a convex optimization
problem and therefore convex programming methods can
be used to tackle such a problem. In RASL [26] (Robust
Alignment by Sparse and Low-Rank decomposition), the
authors posed the problem with transformations as well, and
though it is no longer convex it can be linearized in each iteration and proven to reach a local minimum.

Though the convex programming methods used are polynomial in the size of the problem, that complexity can still be too demanding for very large databases of images. We propose Transformed GRASTA, or t-GRASTA for short, to tackle this optimization with an incremental or online optimization technique. The benefit of this approach is threefold: First, it will improve speeds in batch alignment, as we show in Section III. Second, the memory requirement is small which makes the batch alignment for very large databases realistic, since t-GRASTA only needs to maintain the low-rank subspace throughout the alignment process. Finally, if the low-rank background subspace is stable and learned from a representative batch of images, our online algorithm allows for alignment and occlusion removal on images as they are uploaded to the database.

A. Robust Image Alignment

The problem of robust image alignment arises regularly in real data, as large illumination variations and gross pixel corruptions or partial occlusions often occur, such as sunglasses or a scarf for a human subject. The classic batch image alignment approaches, such as congealing [17] [20] or least squares congealing algorithms [10] [11] cannot simultaneously handle such severe conditions, causing the alignment task to fail.

With the breakthrough of convex relaxation theory applied to decomposing matrices into a sum of low-rank and sparse matrices [7], [8], the recently proposed algorithm “Robust Alignment by Sparse and Low-rank decomposition,” or RASL [26], poses the robust image alignment problem as a transformed version of Robust PCA. The transformed batch of images can be decomposed as the sum of a low-rank matrix of recovered aligned images and a sparse matrix of errors. RASL seeks the optimal domain transformations while trying to minimize the rank of the matrix of the vectorized and stacked aligned images and while keeping the gross errors sparse. While the rank minimization and $\ell^0$ minimization can be relaxed to their convex surrogates—minimize the corresponding nuclear norm $\|\|_*$, and $\ell^1$ norm $\|\|_1$—the relaxed problem (1) is still highly non-linear due to the complicated domain transformation.

$$\min_{A,E,\tau} \|A\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad D \circ \tau = A + E \quad (1)$$

Here, $D \in \mathbb{R}^{n \times N}$ represents the data (n pixels per each of N images), $A \in \mathbb{R}^{n \times N}$ is the low-rank component, $E \in \mathbb{R}^{n \times N}$ is the sparse additive component, and $\tau$ are the transformations. RASL proposes to tackle this difficult optimization problem by iteratively locally linearizing the non-linear image transformation $D \circ (\tau + \Delta \tau) = D \circ \tau + \sum_{i=1}^{n} J_i \Delta \tau_i e_i^T$, where $J_i$ is the Jacobian of image $i$ with respect to transformation $i$; then in each iteration the linearized problem is convex. The authors have shown that RASL works perfectly well for batch aligning the linearly correlated images despite large illumination variations and occlusions.

In order to improve the scalability of robust image alignment for massive image datasets, [31] proposes an efficient ALM-based (Augmented Lagrange Multiplier-based) iterative convex optimization algorithm ORIA (Online Robust Image Alignment) for online alignment of the input images. Though the proposed approach can scale to large image datasets, it requires the subspace of the aligned images as a prior, and for this it uses RASL to train the initial aligned subspace. Once the input images cannot be well aligned by the current subspace, the authors use an heuristic method to update the basis. In contrast, with t-GRASTA we introduce an update which reduces our cost function with a gradient geodesic step on the Grassmannian, as in [2], [15]. We discuss this in more detail in the next section.

B. Online Robust Subspace Learning

Subspace learning has been an area important to signal processing for a few decades. There are many applications in which one must track signal and noise subspaces, from computer vision to communications and radar, and a survey of the related work can be found in [9], [13].

The GROUSE algorithm, or “Grassmannian Rank-One Update Subspace Estimation,” is an online subspace estimation algorithm that can track changing subspaces in the presence of Gaussian noise and missing entries [2]. GROUSE was developed as an online variant of low-rank matrix completion algorithms. It uses incremental gradient methods that have been receiving extensive attention in the optimization community [3]. However, GROUSE is not robust to gross outliers, and the follow-up algorithm GRASTA [14], [15], can estimate a changing low-rank subspace as well as identify and subtract outliers. Still problematic is that, as we showed in Fig. 1, even GRASTA cannot handle camera jitter. Our algorithm includes the estimation of transformations in order to align frames first before separating foreground and background.

II. ROBUST IMAGE ALIGNMENT VIA ITERATIVE ONLINE SUBSPACE LEARNING

A. Model

In order to robustly align the set of linearly correlated images despite sparse outliers, we consider the following matrix factorization model (2) where the low-rank orthonormal matrix $U$ spans the low-dimensional subspace of the well-aligned images.

$$\min_{U,W,\tau} \|E\|_1 \quad (2)$$

$$\text{s.t.} \quad D \circ \tau = UW + E$$

$$U \in \mathcal{G}(d,n)$$

We have replaced the variable $A$ with the product of two smaller matrices $UW$, and the orthonormal columns of $U \in \mathbb{R}^{n \times d}$ span the low-rank subspace of the images. The set of all subspaces of $\mathbb{R}^n$ of fixed dimension $d$ is called the Grassmannian, which is a compact Riemannian manifold and is denoted by $\mathcal{G}(d,n)$. In this optimization model, $U$ is
constrained to the Grassmannian $\mathcal{G}(d,n)$. Though problem (2) can not be directly solved [26] due to the nonlinearity of image transformation, if the misalignments are not too large, by locally linearly approximating the image transformation $D \circ (\tau + \triangle \tau \approx D \circ \tau + \sum_{i=1}^{N} J_i \triangle \tau \varepsilon_i^T$, the iterative model (3) can work well as a practical approach.

$$\min_{U^k, W, E, \triangle \tau} \|E\|_1 \quad \text{subject to} \quad D \circ \tau + \sum_{i=1}^{N} J_i \triangle \tau \varepsilon_i^T = U^k W + E \quad \text{and} \quad U^k \in G(d^k, n).$$

At algorithm iteration $k$, $\tau^k = [\tau_1^k, \ldots, \tau_N^k]$ are the current estimated transformations at iteration $k$, $J_i^k$ is the Jacobian of the $i$-th image with respect to the transformation $\tau_i^k$, and $\{\varepsilon_i\}$ denotes the standard basis for $\mathbb{R}^n$. Note, at different iterations the subspace may have different dimensions, i.e. $U^k$ is constrained on different Grassmannian $\mathcal{G}(d^k, n)$.

At each iteration of the iterative model (3), we consider this optimization problem as the subspace learning problem. That is, our goal is to robustly estimate the low-dimensional subspace $U^k$ which best represents the locally transformed images $D \circ \tau^k + \sum_{i=1}^{N} J_i^k \triangle \tau$ despite sparse outliers $E$. In order to solve this subspace learning problem both efficiently with regards to both computation and memory, we propose to learn $U^k$ at each iteration $k$ in model (3) via the online robust subspace learning approach [14] [15].

At iteration $k$, given the $i$-th image $I_i$, its estimate of transformation $\tau_i^k$, the Jacobian $J_i^k$, and the current estimate of $U_i^k$, to quantify the subspace error robustly, we use the $\ell^1$ norm as follows:

$$F(S; t, k) = \min_{m, \triangle \tau} \|U_i^k w - (I_i \circ \tau_i^k + J_i^k \triangle \tau)\|_1$$

With $U_i^k$ known (or estimated, but fixed), this $\ell^1$ minimization problem is a variation of the least absolute deviations problem, which can be solved efficiently by the technique of ADMM (Alternating Direction Method of Multipliers) [5]. We rewrite the right hand of (4) as the equivalent constrained problem by introducing a sparse outlier vector $e$:

$$\min_{w, E, \triangle \tau} \|e\|_1 \quad \text{subject to} \quad I_i \circ \tau_i^k + J_i^k \triangle \tau = U_i^k w + E.$$  

**B. ADMM Solver for the Locally Linearized Problem**

The augmented Lagrangian of problem (5) is

$$\mathcal{L}(U_i^k, w, e, \triangle \tau, \lambda) = \|e\|_1 + \lambda^T h(w, e, \triangle \tau) + \frac{\mu}{2} \|h(w, e, \triangle \tau)\|^2$$

where $h(w, e, \triangle \tau) = U_i^k w + e - I_i \circ \tau_i^k - J_i^k \triangle \tau$, $\lambda \in \mathbb{R}^n$ is the Lagrange multiplier or dual vector.

Given the current estimated subspace $U_i^k$, transformation parameter $\tau_i^k$, and the Jacobian matrix $J_i^k$ with respect to the $i$-th image $I_i$, the optimal $(w^*, e^*, \triangle \tau^*, \lambda^*)$ can be computed by the ADMM approach as follows:

$$\begin{align*}
\triangle \tau^{p+1} &= (J_i^k J_i^k^T)^{-1} J_i^k^T (U_i^k w^p + e^p - I_i \circ \tau_i^k + \frac{1}{\rho} \lambda^p) \\
w^{p+1} &= (U_i^k U_i^k^T)^{-1} U_i^k^T (I_i \circ \tau_i^k + J_i^k \triangle \tau^{p+1} - e^p - \frac{1}{\rho} \lambda^p) \\
\epsilon^{p+1} &= S_1 (h \circ \tau_i^k + J_i^k \triangle \tau^{p+1} - U_i^k w^{p+1} - \frac{1}{\rho} \lambda^p) \\
\lambda^{p+1} &= \lambda^p + \mu h(w^{p+1}, e^{p+1}, \triangle \tau^{p+1}) \\
\mu^{p+1} &= \rho \mu^p
\end{align*}$$

where $S_1$ is the elementwise soft thresholding operator [6], and $\rho > 1$ is the ADMM penalty constant which makes $\{\mu^p\}$ as a monotonically increasing positive sequence, then the iteration (7) indeed converges to the optimal solution of the problem (5) [4]. We summarize this ADMM solver as Algorithm 2 in Section II-D.

**C. Subspace Update**

Identifying the best $U_i^k$ in the model (3) is actually the Grassmannian optimization problem. That is to say, we seek the sequence $\{U_i^k\} \in \mathcal{G}(d^k, n)$ such that $U_i^k \rightarrow U_i^k^*$ (as $t \rightarrow \infty$). The critical problem of this optimization is to chose a proper subspace loss function. Since, regarding $U$ as the variable, the loss function (4) is not differentiable everywhere, we choose instead to use the augmented Lagrangian (6) as the subspace loss function once we have estimated $(w^*, e^*, \triangle \tau^*, \lambda^*)$ by ADMM (7) from previous $U_i^k$ [14] [15].

In order to take a gradient step along the geodesic of the Grassmannian, according to [13], we first need to derive the gradient formula of the real-valued loss function (6) $\mathcal{L} : \mathcal{G}(d, n) \rightarrow \mathbb{R}$. The gradient $\nabla \mathcal{L}$ can be determined from the derivative of $\mathcal{L}$ with respect to the components of $U_i^k$:

$$\frac{d\mathcal{L}}{dU_i^k} = (\lambda^* + \mu h(w^*, e^*, \triangle \tau^*)) w^{*T}$$

Then the gradient is $\nabla \mathcal{L} = (I - U_i^k U_i^k)^T \frac{d\mathcal{L}}{dU_i^k}$ [13]. From Step 6 of Algorithm 1, we have that $\nabla \mathcal{L} = \Gamma w^{*T}$ (see the definition of $\Gamma$ in Alg. 1). It is easy to verify that $\nabla \mathcal{L}$ is rank one since $\Gamma$ is a $n \times 1$ vector and $w^*$ is a $d \times 1$ weight vector. The following derivation of geodesic gradient step is similar to GROUSE [2] and GRASTA [14] [15]. We rewrite the important derivation steps here for completeness.

The sole non-zero singular value is $\sigma = \|\Gamma\|\|w^*\|$, and the corresponding left and right singular vectors are $\frac{\Gamma}{\|\Gamma\|}$ and $\frac{w^*}{\|w^*\|}$ respectively. Then we can write the SVD of the gradient explicitly by adding the orthonormal set $x_2, \ldots, x_d$ orthogonal to $\Gamma$ as left singular vectors and the orthonormal set $y_2, \ldots, y_d$ orthogonal to $w^*$ as right singular vectors as follows:

$$\nabla \mathcal{L} = \begin{bmatrix} \Gamma^T \\ \Gamma x_2 \quad \ldots \quad x_d \end{bmatrix} \times \text{diag}(\sigma, 0, \ldots, 0) \times \begin{bmatrix} w^* \\ \|w^*\| y_2 \quad \ldots \quad y_d \end{bmatrix}^T.$$

Finally, following Equation (2.65) in [13], a geodesic gradient step of length $\eta$ in the direction $-\nabla \mathcal{L}$ is given by
\[ U(\eta) = U + (\cos(\eta \sigma) - 1) \frac{U w_i^T w_i^T}{\|w_i^T\| \|w_i^T\|} - \sin(\eta \sigma) \Gamma \frac{w_i^T}{\|\Gamma\| \|w_i^T\|}. \quad (9) \]

D. Algorithms

From the discussion of Sections II-B and II-C, given the batch of unaligned images \( D \), their estimate of transformation \( \tau^k \) and their Jacobian \( J^k \) at iteration \( k \), we can robustly identify the subspace \( U^k \) by incrementally updating \( U^k \) along the geodesic of Grassmannian \( \mathcal{G}(d^2, n) \) (9). When \( U_i^j \rightarrow U^k \) (as \( t \rightarrow \infty \)), the estimate of \( \Delta \tau_i \) for each initially aligned image \( I_i \circ \tau_i^k \) also approaches its optimal value \( \Delta \tau_i^* \).

Once the subspace \( U^k \) is accurately learned, we will have the updated estimate of transformation \( \tau_{i+1}^k = \tau_i^k + \Delta \tau_i^k \) for each image. Then in the next iteration, the new subspace \( U^{k+1} \) can also be learned from \( D \circ \tau_{i+1}^k \), and the accuracy of the estimated transformation \( \tau \) can be further improved, until we reach the stopping criterion, e.g., \( \|\Delta \tau_i^k\|_2 < \varepsilon \) or reach the maximum iteration \( K \).

We summarize our algorithms as follows. Algorithm 1 is the batch image alignment approach via iterative online robust subspace learning. For Step 7, there are many ways to pick the step-size, for example, diminishing and constant step-sizes adopted in GROUSE [2], and multi-level adaptive step-size for fast convergence in GRASTA [14].

Algorithm 2 is the ADMM solver for the locally linearized problem (5). From our extensive experiments, if we set the ADMM penalty parameter \( \rho = 2 \) and the tolerance \( \varepsilon_{tol} = 10^{-7} \), Algorithm 2 has always converged in less than 20 iterations.

E. Discussion of Memory Usage

We compare the memory usage of our algorithm t-GRASA to that of RASL. RASL requires storage of \( A, E \), a lagrange multiplier matrix \( Y \), the data \( D \), and \( D \circ \tau \), each of which require storage of the size \( n \). To do the singular value decomposition, to compare fairly to t-GRASA which assumes a \( d \)-dimensional model, we suppose RASL uses a thin SVD of size \( d \), which requires \( nd + Nd + d^2 \) memory elements. Finally for the Jacobian per image, RASL needs \( nNp \), and for \( \tau \) RASL needs \( Np \), but we will assume \( p \) is a small constant regardless of dimension and ignore it. Therefore RASL’s total memory usage is \( 60nN + nd + Nd + d^2 + N \).

t-GRASA also must store the Jacobian, \( \tau \), and the data as well as the data with transformation, using memory size \( 3nN + N \). Otherwise, t-GRASA only needs to store a single \( U \) matrix of size \( nd \), and the vectors \( e, \lambda, \Gamma \), and \( w \) for \( 3n + d \) memory elements. Thus t-GRASA’s memory total is \( 3nN + nd + 3n + d + N \).

For a problem size of 100 images, each with \( 100 \times 100 \) pixels, and assuming \( d = 10 \), t-GRASA uses 51% of the memory of RASL. For 10000 mega-pixel images, t-GRASA uses 50% of the memory of RASL. The scaling remains about half throughout mid-range to large problem sizes.

Algorithm 1 Transformed GRASTA

**Require:** An initial \( n \times d^2 \) orthogonal matrices \( U^0 \). A sequence of unaligned images \( I_i \) and the corresponding initial transformation parameters \( \tau_i^0 \). \( (i = 1 \ldots N) \). The maximum iteration \( K \).

**Return:** The estimated well-aligned subspace \( U_i^{k+1} \) for the well-aligned images. The transformation parameters \( \tau_i^k \) for each well-aligned image.

1: while not converged and \( k < K \) do
2: Update the Jacobian matrix of each image:
   \( J_i^k = \frac{\partial (I_i \circ \tau_i^k)}{\partial \tau_i^k} \) \( (i = 1 \ldots N) \)
3: Update the wrapped and normalized images:
   \( I_i \circ \tau_i^k = \frac{\vec{vec}(I_i \circ \tau_i^k)}{\|\vec{vec}(I_i \circ \tau_i^k)\|_2} \)
4: for \( j = 1 \rightarrow N, \ldots, \) until converged do
5: Estimate the weight vector \( w_i^k \), the sparse outliers \( e_i^k \), the locally linearized transformation parameters \( \Delta \tau_i^k \), and the dual vector \( \lambda_i^k \) via the ADMM algorithm 2 from \( I_i \circ \tau_i^k, J_i^k \), and the current estimated subspace \( U_i^k \)
   \( (w_i^k, e_i^k, \Delta \tau_i^k, \lambda_i^k) = \arg \min_{w_i, e_i, \Delta \tau_i, \lambda_i} \mathcal{L}(U_i^k, w, e, \lambda) \)
6: Compute the gradient \( \nabla \mathcal{L} \) as follows:
   \( \Gamma_i = \lambda_i^k + \mu \frac{\nabla \mathcal{L}(w_i^k, e_i^k, \Delta \tau_i^k)}{\|\nabla \mathcal{L}(w_i^k, e_i^k, \Delta \tau_i^k)\|} \),
   \( \nabla \mathcal{L} = \Gamma_i w_i^k \)
7: Compute step-size \( \eta_i \).
8: Update subspace:
   \( U_i^{k+1} = U_i^k + \left( \cos(\eta_i \sigma) - 1 \right) U_i \frac{w_i^k}{\|w_i^k\|} - \sin(\eta_i \sigma) \Gamma_i \frac{w_i^k}{\|\Gamma_i\| \|w_i^k\|} \) \( \|w_i^k\| \)
9: end for
10: Update the transformation parameters:
    \( \tau_i^{k+1} = \tau_i^k + \Delta \tau_i^k \), \( (i = 1 \ldots N) \)
11: end while

F. Discussion of Online Image Alignment

If the subspace \( U_i^k \) of the well-aligned images is known as a prior, for example \( U_i^k \) is trained by Algorithm 1 from a "well selected" dataset of one category, we can simply use \( U_i^k \) to align the rest of the unaligned images of the same category. Here "well selected" means the training dataset should cover enough of the global appearance of the object, such as different illuminations, which can be represented by the low-dimensional subspace structure. By category, we mean a particular object of interest or a particular background scene in the video surveillance data.

For massive image processing tasks, it is easy to collect such good training datasets by simply randomly sampling a
Algorithm 2 ADMM Solver for the Locally Linearized Problem (5)

Require: An \( n \times d \) orthogonal matrix \( U \), a wrapped and normalized image \( I \), \( \tau \in \mathbb{R}^n \), the corresponding Jacobian matrix \( J \), and a structure OPTS which holds four parameters for ADMM: ADMM penalty constant \( \rho \), the tolerance \( \epsilon_{tol} \), and ADMM maximum iteration \( K \).

Return: weight vector \( w^* \in \mathbb{R}^d \); sparse outliers \( e^* \in \mathbb{R}^n \); locally linearized transformation parameters \( \tau^* \); and dual vector \( \lambda^* \in \mathbb{R}^n \).

1: Initialize \( w, e, \Delta \tau, \Delta \lambda \), and \( \mu \): \( e^1 = 0, w^1 = 0, \Delta \tau^1 = 0, \Delta \lambda^1 = 0, \mu = 1 \)
2: Cache \( P = (U^T U)^{-1} U^T \) and \( F = (J^T J)^{-1} J^T \)
3: for \( k = 1 \to K \) do
4: Update \( \Delta \tau \): \( \Delta \tau^{k+1} = F(U w^k + e^k - I \circ \tau + \frac{1}{\mu} \Delta \lambda^k) \)
5: Update weights: \( w^{k+1} = P(I \circ \tau + J \Delta \tau^{k+1} - e^k - \frac{1}{\mu} \Delta \lambda^k) \)
6: Update sparse outliers:
7: Update dual:
8: if \( \| h(w^{k+1}, e^{k+1}, \Delta \tau^{k+1}) \|_2 \leq \epsilon_{tol} \) then
9: Converge and break the loop.
10: end if
11: end for
12: end while
13: \( w^* = w^{k+1} \), \( e^* = e^{k+1} \), \( \Delta \tau^* = \Delta \tau^{k+1} \), \( \lambda^* = \lambda^{k+1} \)

small fraction of the whole image set. Once \( U^k \) is learned from the training set, we can use a variation of Algorithm 1 to align each unaligned image \( I \) without updating the subspace, since we have the assumption that the remaining images also lie in the trained subspace. The simple online approach is summarized as Algorithm 3.

III. PERFORMANCE EVALUATION

In this section, we conduct comprehensive experiments on a variety of alignment tasks to verify the efficiency and superiority of our algorithm. We first demonstrate the ability of the proposed approach to cope with occlusion and illumination variation during the alignment process. After that, we further demonstrate the robustness of our approach by testing it on a more challenging database, the Labeled Faces in the Wild database [16]. Finally, we apply our approach to solving the interesting background foreground separation problem in the case of camera jitter.

A. Occlusion and illumination variation

We first test our approach on the dataset ‘dummy’ described in [26]. Here, we want to verify the ability of our approach to effectively align the images despite occlusion and illumination variation. The dataset contains 100 images of a dummy head taken under varying illumination and with artificially generated occlusions created by adding a square patch at a random location of the image. Fig. 2 shows 10 misaligned images of the dummy. We align these images by Algorithm 1. The canonical frame is chosen to be \( 49 \times 49 \) pixels and the subspace dimension is set to 5. Here and in the rest of our experiments, for simplicity we set \( d^k \) of Algorithm 1 to a fixed \( d \) in every iteration. The last three rows of Fig. 2 show the results of alignment, from which we can see that our approach is successful at aligning the misaligned images while removing the occlusion at the same time.

B. Robustness

In order to further demonstrate the robustness of our approach, we apply it on more challenging and realistic images taken from the Labeled Faces in the Wild database [16]. The LFW contains more severely misaligned images, for it also includes remarkable variations in pose and expression as well as illumination and occlusion, which can be seen in Fig. 3(c). We chose 16 subjects from LFW, each of them with 35 images. Each image is aligned to an \( 80 \times 60 \) grid.
canonical frame using $\tau$ which are from the group of affine transformations $G = Aff(2)$, as in [26]; these are translations, rotations, and scale transformations. For each subject, we set the subspace dimension $= 15$ and use Algorithm 1 to align each image. In this example, we demonstrate the robustness of our approach by comparing the average face of each subject before and after alignment, which are shown in Fig. 3(a)-(b). We can see that the average faces after alignment are much clearer than those before alignment. Fig. 3(c)-(d) provides more detailed information, showing the unaligned and aligned images of John Ashcroft (marked by red boxes in Fig. 3(a)-(b)).

**C. Video Jitter**

Here we apply t-GRASTA to the task of separating moving objects from static background in the video footage recorded by an unstable camera. We note that in [15], the authors simulate a virtual panning camera to show that GRASTA can quickly track sudden changes in the background subspace caused by a moving camera. Their low-rank subspace tracking model is well-defined, as the camera after panning is still stationary, and thus the recorded video frames are accurately pixelwise aligned. However, for an unstable camera, the recorded frames are no longer aligned; the background can not be well represented by a low-rank subspace unless the jittered frames are first aligned. In order to show that t-GRASTA can tackle this challenging separation task, we consider a highly jittered video sequence generated by a simulated unstable camera. To simulate the unstable camera, we randomly translate the original well-aligned video frames in x- / y- axis and rotate them in the plane.

In this experiment, we compare t-GRASTA with RASL and GRASTA $^1$. We use the first 200 frames of the “Hall” dataset$^2$, each $144 \times 176$ pixels. We first perturb each frame artificially to simulate camera jitter. The rotation of each frame is random, uniformly distributed within the range of $[-\theta_0/2, \theta_0/2]$, and the ranges of x- and y-translations are limited to $[-x_0/2, x_0/2]$ and $[-y_0/2, y_0/2]$. In this example, we set the perturbation size parameters $[x_0, y_0, \theta_0]$ with the values of $[20, 20, 10^\circ]$.

For comparing with RASL, unlike [31], we just let RASL run its original batch model without forcing it into an online algorithm framework. The task we give to RASL and t-GRASTA is to align each frame to a $62 \times 75$ canonical frame, again using $G = Aff(2)$. The dimension of the subspace in t-GRASTA is set to be 10. We first randomly select 30 frames of the total 200 frames to train the subspace by Algorithm 1 and then align the rest by the simple online approach of Algorithm 3. The visual comparison between RASL and t-GRASTA are shown in Fig. 4. Table I illustrates the numerical comparison of RASL and t-GRASTA, for which we ran each algorithm 10 times to get the statistics. From Table I and Fig. 4 we can see that the two algorithms achieve a very similar effect, but t-GRASTA runs much faster than RASL: On a PC with Intel P9300 2.27GHz CPU and 2 GB of RAM, the average time for aligning a newly arrived frame is 1.1 second, while RASL needs more than 800 seconds to align the total batch of images, or 4 seconds per frame. Moreover, our approach is also superior to RASL regarding memory efficiency. These superiorities become more dramatic as one increases the size of the image database.

---

$^1$We note that [31] proposes the online algorithm ORIA, but the code hasn’t been released yet at the time of preparation of the paper. We intend to make the comparison once the authors release their code.

$^2$Find these along with the videos at [http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html](http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html).
TABLE I
Statistics of errors in two pixels \( P_1 \) and \( P_2 \), selected from the original video frames and traced through the jitter simulation process to the RASL and t-GRASTA output frames. Max error and mean error are calculated as the distances from the estimated \( P_1 \) and \( P_2 \) to their statistical center \( E(P_1) \) and \( E(P_2) \). Std are calculated as the standard deviation of four coordinate values \( (X_1,Y_1) \) for \( P_1 \) and \( (X_2,Y_2) \) for \( P_2 \) across all frames.

<table>
<thead>
<tr>
<th></th>
<th>Max error</th>
<th>Mean error</th>
<th>( X_1 ) std</th>
<th>( Y_1 ) std</th>
<th>( X_2 ) std</th>
<th>( Y_2 ) std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial misalignment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASL</td>
<td>2.96</td>
<td>1.73</td>
<td>0.56</td>
<td>0.31</td>
<td>0.34</td>
<td>1.47</td>
</tr>
<tr>
<td>t-GRASTA</td>
<td>6.62</td>
<td>0.84</td>
<td>0.48</td>
<td>1.11</td>
<td>0.57</td>
<td>0.74</td>
</tr>
</tbody>
</table>

In order to compare with GRASTA, we use 200 perturbed images to recover the background and separate the moving objects for both algorithms; Fig. 5 illustrates the comparison. For both GRASTA and t-GRASTA, we set the subspace rank = 10 and randomly selected 30 images to train the subspace first. For t-GRASTA, we again use the affine transformation \( G = Aff(2) \). From Fig. 5, we can see that our approach successfully separates the foreground and the background and simultaneously align the perturbed images. But GRASTA fails to learn a proper subspace, thus, the separation of background and foreground is poor. Although GRASTA has been demonstrated to successfully track a dynamic subspace, e.g., the panning camera, the dynamics of an unstable camera are too fast and unpredictable for the GRASTA subspace tracking model to succeed in this context without pre-alignment of the video frames.

IV. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this paper we have presented an iterative Grassmannian optimization approach to simultaneously identify an optimal set of image domain transformations for image alignment and the low-rank subspace matching the aligned images. These are such that the vector of each transformed image can be decomposed as the sum of a low-rank part of the recovered aligned image and a sparse part of errors. This approach can be regarded as an extension of GRASTA and RASL: We extend GRASTA to transformations, and extend RASL to the incremental gradient optimization framework. Our approach is faster than RASL and more robust to alignment than GRASTA. We can effectively and computationally efficiently learn the low-rank subspace from misaligned images which is very practical for computer vision applications.

While preparing the final version of this paper, we noticed an interesting alignment approach proposed in [21]. Though the two approaches of ours and [21] are both obtained via optimization over the special manifold, they perform alignment for very different scenarios. For example, the approach in [21] focuses on semantically meaningful videos or signals, and then it can successfully align the videos of the same object from different views; t-GRASTA manipulates the set of misaligned images or the video of unstable camera to robustly identify the low-rank subspace, and then it can align these images according to the subspace.

B. Future Work

Though this work presents an approach for robust image alignment more computationally efficient than state-of-the-art, a foremost remaining problem is how to scale the proposed approach to a very large streaming dataset such as real-time video processing. We have shown that given the accurate estimated aligned subspace, the simple online method Algorithm 3 can efficiently align a very large scale of misaligned images, but what if the subspace of the streaming video data is changing over time? The GRASTA algorithm can successfully track the changing subspace, but this problem becomes much more difficult and less well-defined when images are misaligned. We are very interested to seek a truly online subspace learning algorithm for this very difficult problem. Another question of interest is regarding the estimation of \( d^k \) for the subspace update. Though we fix the rank \( d \) in this paper, estimating \( d^k \) and switching between Grassmannians is a very interesting future direction.

V. ACKNOWLEDGEMENTS

This work of Jun He is supported by NSFC (61203273) and by Collegiate Natural Science Fund of Jiangsu Province (11KJB510009). Laura Balzano would like to acknowledge 3M for generously supporting her Ph.D. studies.

REFERENCES

Fig. 5. Video background and foreground separation with jittered video. 1st row: 10 misaligned video frames randomly selected from artificially perturbed images; 2nd row: images aligned by t-GRASTA; 3rd row: background recovered by t-GRASTA; 4th row: foreground separated by t-GRASTA; 5th row: background recovered by GRASTA; 6th row: foreground separated by GRASTA.