18660 Project Final Report **10 May 2018**

Comparison of robust PCA techniques in image and video processing

Joel Loo, Wang Xueqiang

1 Introduction

Principal components analysis (PCA) and other dimensionality reduction techniques have gained wide traction in the scientific community because of their utility in extracting salient data from large and complex datasets. However, PCA implicitly assumes that the noise in data is small and variance from the mean is bounded (such as in Gaussian noise), causing it to perform poorly on grossly corrupted data, or data containing outliers of significant magnitude.

While there have been many attempts to robustify PCA, many of these methods are heuristics without any optimality guarantees. However, recent advances in low-rank matrix recovery and completion by Candes et al [1], have led to the formulation of robust PCA (RPCA). In RPCA, the problem is reformulated as the recovery of the matrices L, S such that $M = L + S$, where M is the original matrix, L is a low-rank matrix and S is a sparse matrix. Candes et al made the surprising observation that it is possible to perform this decomposition through a tractable convex optimization, if we assume that the low-rank matrix is not simultaneously sparse [1]. In particular, this can be achieved using the Principal Component Pursuit (PCP) estimate which solves the problem

$$\underset{L,S}{\text{argmin}} \quad \|L\|_* + \lambda\|S\|_1 \quad \text{subject to} \quad L + S = M$$

Since the sparse matrix can have elements of arbitrarily large magnitude, robust PCA thus improves over the PCA technique by allowing the extraction of low-dimensional structure from grossly corrupted data with unbounded outliers. This technique has many important applications in general, and in particular is of great utility in computer vision, where it is used in tackling problems such as background recovery/subtraction (which can be used in video surveillance software), removal of non-Lambertian distortions/specularities/shadows from images (akin to performing inpainting) and even recovery of 3D geometry from low-rank textures in 2D images.

In order to make RPCA practicable on real-world vision systems, a number of reformulations of the problem and improvements have been proposed. Some of these improvements (e.g. Fast PCP [3]) provide significant speed improvements over the method proposed by Candes et al, while others reformulate RPCA as an online algorithm (e.g. online RPCA via stochastic optimization [2], iFrALM).

2 Problem Statement

In this project, we aim to survey several RPCA algorithms - in particular we will implement these algorithms and compare their performance, in terms of both speed and ability to separate low-rank structure from sparse noise. We are primarily interested in applying these algorithms to, and testing them on vision applications. Two applications that we are particularly interested in are the removal of specularities/non-Lambertian distortions from images, as well as background recovery/subtraction in videos. Background

recovery/subtraction in video streams provides an interesting case study for RPCA as well, since it is desirable to have both algorithms that can separate backgrounds in post-processing as well as on the fly. To this end, we will use this particular application to benchmark the performance of both RPCA algorithms that offer significant speed-ups, as well as RPCA algorithms formulated to be online in nature. We will perform the implementations and benchmarking described above in Python, with code available on the private Github repository https://github.com/joelloo/18660_project

3 Methods

In this section, we discuss the formulation of the optimization problems for the various algorithms and our approach to their implementation. As mentioned, we are implementing these algorithms with an eye toward applying them on specularity removal from Lambertian surfaces and realtime background subtraction in video streams. The low-rank/sparse decomposition is relevant in specularity removal from images, since convex-shaped Lambertian objects can generally be approximated by a low dimensional linear subspace [11, 12]. As such, running RPCA on scenes containing objects that can be approximated by convex-shaped Lambertian surfaces cannot allow us to separate distortions such as specularities and occlusions from the scene. With regards to background subtraction in video streams, RPCA is relevant since we can vectorize each frame of the stream and concatenate them to form a matrix. This matrix can then be modelled as the sum of a low-rank matrix, which represents the background (i.e. all that is constant across frames), and the sparse matrix, which represents the movements in the foreground of the image. Hence, the low-rank/sparse decomposition can allow us to separate background and foreground in video streams.

We make the distinction between offline and online RPCA algorithms, i.e. algorithms that require the entire matrix beforehand to perform the low-rank/sparse decomposition as opposed to algorithms that can take in new columns of the matrix as they come in and update the decomposition on the fly. In particular, online RPCA algorithms allow us to perform realtime background subtraction, where we have an online setting with new frames coming in from the video stream.

3.1 Offline RPCA Algorithms

The original PCP formulation of the RPCA algorithm has several immediately apparent solutions. We can consider the relaxation $\underset{L,S}{\text{argmin}} \|L\|_* + \|S\|_1 + \frac{1}{2}\|M - L - S\|_F^2$. Since $\|\cdot\|_*$ and $\|\cdot\|_1$ have well-known proximal operators in the singular value thresholding operator [9] and the soft thresholding operator, while $\|\cdot\|_F$ is differentiable, it is clear that we can solve this formulation via proximal gradient descent. Indeed, Lin et al. [5] provide an accelerated proximal gradient descent (APG) based algorithm that solves this formulation by alternately minimizing L (with singular value thresholding) then E (with soft thresholding) at each iteration.

Another possible means of approaching the PCP formulation is via the augmented Lagrangian multiplier method (ALM), as suggested by Candes et al. in [1] and elaborated on by Lin et al. in [8]. We can write the Lagrangian function of the PCP formulation as $\mathcal{L}(L, S, Y, \mu) = \|L\|_* + \lambda\|S\|_1 + \langle Y, M - L - S\rangle + \frac{\mu}{2}\|M - L - S\|_F^2$.

The augmented Lagrangian multiplier algorithm takes the approach described in algorithm 1. In order to perform the minimization of $\mathcal{L}$ wrt L, S, we can perform an alternating minimization where we first update L_{k+1} (using singular value thresholding) and then S_{k+1} (using soft thresholding), similar to APG. We can perform this until convergence, to obtain L_{k+1}^*, S_{k+1}^*. In contrast to this exact ALM approach, Lin et al. also proposed an inexact ALM approach, wherein the minimization of $\mathcal{L}$ is simply replaced by one iteration of the alternating minimization approach (i.e. a single singular value thresholding and soft thresholding step). This can be proved to converge to the optimal value, while doing less work by removing the nested loop [8].

Algorithm 1 Augmented Lagrangian multiplier method

Initialize $Y_0 = sgn(D)/J(sgn(D))$; $\mu_0 > 0$; $\rho > 1$; $k = 0$
while not converged **do**
  $(L^*_{k+1}, S^*_{k+1}) = \underset{L,S}{\operatorname{argmin}}\ \mathcal{L}(L, S, Y^*_k, \mu_k)$
  $Y^*_{k+1} = Y^*_k + \mu_k(M - L^*_{k+1} - S^*_{k+1})$
  Update μ_k to μ_{k+1}
end while
return L_k, S_k

This is referred to as the inexact ALM approach, since we are not solving exactly for L^*_{k+1}, S^*_{k+1}.

A major difficulty in implementing inexact ALM as described by Candes et al. lay in the initialization of Y_0. Our algorithm with a randomly initialized Y_0 did not perform as well as the results from studies implementing inexact ALM. However, Lin et al. suggest the usage of the initializer $Y_0 = sgn(D)/J(sgn(D))$ where $J = \max\{\|\cdot\|_2, \|\cdot\|_\infty\}$, which ensures that $\langle D, Y_0\rangle$ is a reasonably large value to begin with [8], thereby speeding convergence.

We also considered the Fast PCP algorithm [3]. In contrast to the previous approaches, Fast PCP relaxes the constraint $M = L + S$ to a penalty in the objective function while also transforming the nuclear norm penalty into the constraint $\|L\|_* \leq t$. Rodriguez et al. argue that in practice we are mainly interested in solutions for which this constraint is active, i.e. $\|L\|_* = t$. Considering only this case, we can express the nuclear norm equality constraint directly as a rank constraint, i.e. $\text{rank}(L) \leq t$. This yields the problem

$$\underset{L,S}{\operatorname{argmin}} \quad \|L + S - M\|_F + \lambda\|S\|_1 \quad \text{subject to} \quad \text{rank}(L) = t$$

We recognize that this problem is amenable to alternating minimization, since it comprises the sub-problems

$$\begin{cases} L_{k+1} = \underset{L}{\operatorname{argmin}} \quad \|L_k + S_k - M\|_F \quad \text{subject to} \quad \text{rank}(L_k) = t \\ S_{k+1} = \underset{S}{\operatorname{argmin}} \quad \|L_{k+1} + S_k - M\|_F + \lambda\|S\|_1 \end{cases}$$

We recognize the first sub-problem as a rank-constrained least squares problem, for which the canonical solution is to compute the partial SVD of $M - S$ up to rank t. The second sub-problem is similar in form to LASSO and can be solved by direct application of the soft thresholding operator.

The algorithm derived from this reformulation of the PCP objective is remarkably simple. To compute the partial SVD, we used scikit-learn's *randomized_svd* function to efficiently compute the SVD up to the estimated rank. As suggested in [3], we also implemented a heuristic check to enforce low rank in L, wherein we compute the contribution of each successive singular value, and stop increasing the rank when the singular value becomes too small. Since the soft thresholding step is not computationally expensive, and the cost of computing the SVD is heavily mitigated since we are only computing the partial SVD up to a low rank, this algorithm is computationally cheap and thus fast.

3.2 Online RPCA Algorithms

An online algorithm adapted to process video streams on the fly has to be able to take in new frames, and has to have a mechanism for adding new frames to the matrix of vectorized frames and updating the decomposition. These are not features the previous algorithms possess, precluding them from use as true online algorithms. We first consider incremental stochastic RPCA (STOC-RPCA) [2]. This algorithm adapts the original PCP formulation, relaxing the $M = L+S$ constraint as a penalty in the objective and expressing the

nuclear norm as $\inf_{L,R}\{\frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 : X = LR^T\}$, where L can be thought of as a fixed basis. Substituting this back into objective expression and converting into a form that takes each column vector of R and S (corresponding to each video frame) as arguments, we have $\frac{1}{t}\sum_{i=1}^{t}(\frac{1}{2}\|\mathbf{m}_i - L\mathbf{r}_i - \mathbf{s}_i\|_2^2 + \frac{\lambda_1}{2}\|\mathbf{r}_i\|_2^2 + \lambda_2\|\mathbf{s}_i\|_1) + \frac{\lambda_1}{2t}\|L\|_F^2$. This can be solved through a alternating minimization, where we first minimize the objective with respect to $\mathbf{r}_i$ and $\mathbf{s}_i$, then update the basis L using these vectors.

The paper recommended solving the first sub-problem to find $\mathbf{r}_i, \mathbf{s}_i$ using standard solvers. Since the first sub-problem is a convex problem, we attempted to use CVXPY to minimize the first sub-problem. This approach yielded poor results for reasons which we do not fully understand. However, a closer inspection showed that CVXPY was applying the ECOS solver for SOCPs to our problem. This sub-problem resembles a QP more than an SOCP, and we surmise that applying another solver more suited for QPs might resolve this issue. Our final solution to the first sub-problem was to take an alternating minimization approach, first updating $\mathbf{r}_i$ and then updating $\mathbf{s}_i$. The objective is fully differentiable with respect to $\mathbf{r}_i$ and we can simply use the closed-form partial derivative to minimize for given $\mathbf{s}_i$. The objective with respect to $\mathbf{s}_i$ has the form of LASSO, and the minimizer of $\mathbf{s}_i$ given $\mathbf{r}_i$ for this iteration is given by the soft thresholding operator. Once we have computed $\mathbf{r}_i, \mathbf{s}_i$, we can update the basis L using coordinate descent as suggested in [2]. Experiments indicate that this approach has good convergence properties, and yields acceptable results in a fairly fast.

Aside from this, we also implemented Incremental Fast PCP [10], a modification of the Fast PCP algorithm to turn it into an online algorithm. While the formulation of the optimization problem remains the same, the algorithm is modified to accommodate a constant incoming stream of video frames. Since the soft thresholding operator is an elementwise operator, it does not need to be modified for the online algorithm. For the partial SVD step however, the matrix has to be updated along with the SVD. Rodriguez et al. propose a method making use of the following SVD update functions,

- Update: compute $U_1\Sigma_1V_1^T = [D\ \mathbf{d}]$ given $U_0\Sigma_0V_0^T = D$ and new frame $\mathbf{d}$
- Replace: compute $U_1\Sigma_1V_1^T = [D\ \hat{\mathbf{d}}\ E]$ given $U_0\Sigma_0V_0^T = [D\ \mathbf{d}\ E]$ and modified frame $\hat{\mathbf{d}}$
- Downdate: compute $U_1\Sigma_1V_1^T = D$ given $U_0\Sigma_0V_0^T = [D\ \mathbf{d}]$

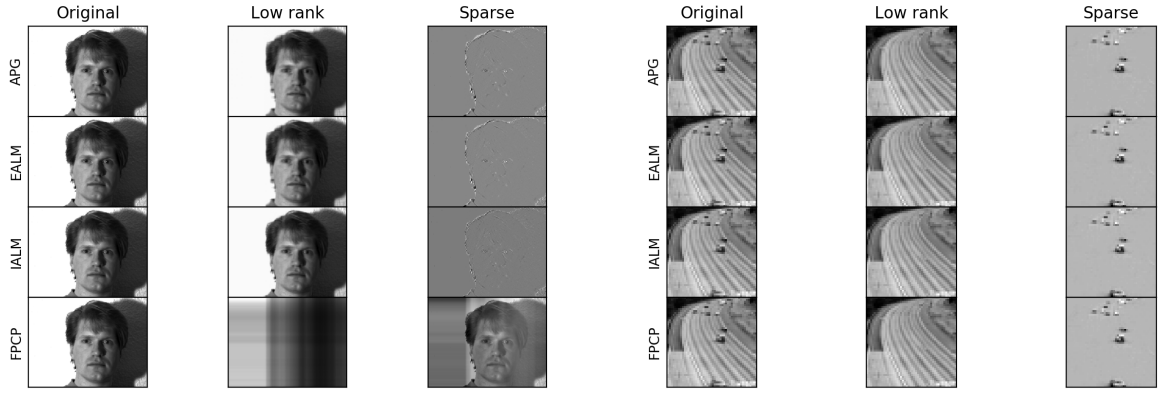
Recall that the partial SVD step takes the SVD of $M - S$ up to rank t. Given a new frame $\mathbf{m}$, we can use the update function to find the SVD of the matrix $[M - S\ \mathbf{m}]$. We want to obtain the partial SVD of $[M - S\ \mathbf{m\text{-}s}]$. To do this, we can perform several iterations of alternating minimization on L_k, S_k using the decomposition of the initial updated matrix $[M - S\ \mathbf{m\text{-}s}]$, using the partial SVD and soft thresholding operator as described in Fast PCP. Once we have converged on a better estimate of the sparse component S, we can use the replace operation to find the decomposition of $[M - S\ \mathbf{m\text{-}s}]$ instead, where $\mathbf{s}$ is the updated sparse component for the new frame $\mathbf{m}$. In addition, Rodriguez et al. suggest the use of the downdating operation to remove earlier frames and maintain a sliding window of frames in the matrix M.

The workhorses of this algorithm are the SVD update operations, and we spent a significant amount of time understanding, implementing and testing for these operations for correctness. Brand [13] gives a concise description of how to implement rank-1 modifications for the SVD that allow us to achieve all 3 of the required SVD update operations listed above. In particular, these update algorithms allow us to compute a rank r partial SVD in linear time, i.e. $O(pqr)$ for a matrix of size $p \times q$ and rank r, where columns of data are being streamed in to the matrix. Aside from the modifications needed to make the algorithm suitable for streaming data, and to make use of the SVD update operations, the implementation of Incremental Fast PCP remains similar in spirit to the implementation of Fast PCP.

4 Results and comparisons

We analyze and compare the performance of offline and online RPCA algorithms separately. We explore the performance in terms of the goodness of separation of low-rank and sparse components of an image or video stream, as well as the runtime performance of the various algorithms.

4.1 Offline RPCA Algorithms



(a) Specularity removal

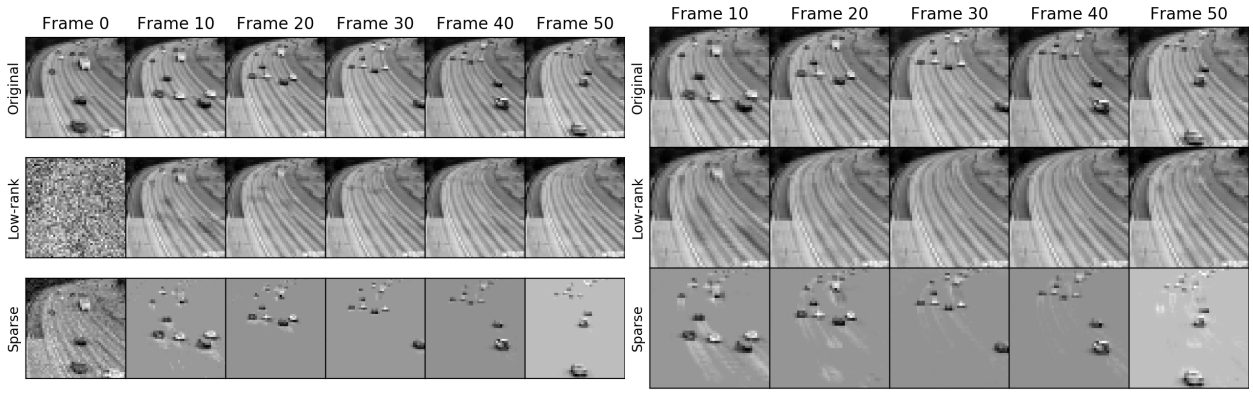
(b) Video background subtraction

Figure 1.1: Performance of offline RPCA algorithms on image/video processing

We benchmarked the performance of our offline RPCA algorithms by running them on specularity removal and background subtraction for videos. The human face under varying illumination can be approximated as a Lambertian surface, i.e. it lies close to a low-dimensional linear subspace, hence allowing us to directly apply RPCA to separate specularities from images of faces. Hence, we chose to run specularity removal tests on images from the YaleFaces database. For the video background subtraction tasks, we ran our algorithms on datasets from [14].

Fig. 1.1(a) shows an image from YaleFaces (with subject under left-aligned lighting) separated into low-rank and sparse components using the various offline RPCA algorithms implemented. We observe that APG, exact ALM and inexact ALM yield good results, with specularly reflected light in the subject's hair and eyes separated into the sparse component. Fast PCP yields visually undesirable results because it makes use of a heuristic that tends to prefer extremely low ranks for the low-rank matrix. The low-rank component separated is therefore of too low a rank to capture sufficient information about the subject's face. This heuristic however, works well on the video background subtraction task, where the rank of the vectorized collection of frames is indeed extremely low. This is evident through the excellent performance of Fast PCP in separating the moving cars from the background. We also observe similarly good performance from APG, exact and inexact ALM on the highway dataset.

4.2 Online RPCA Algorithms

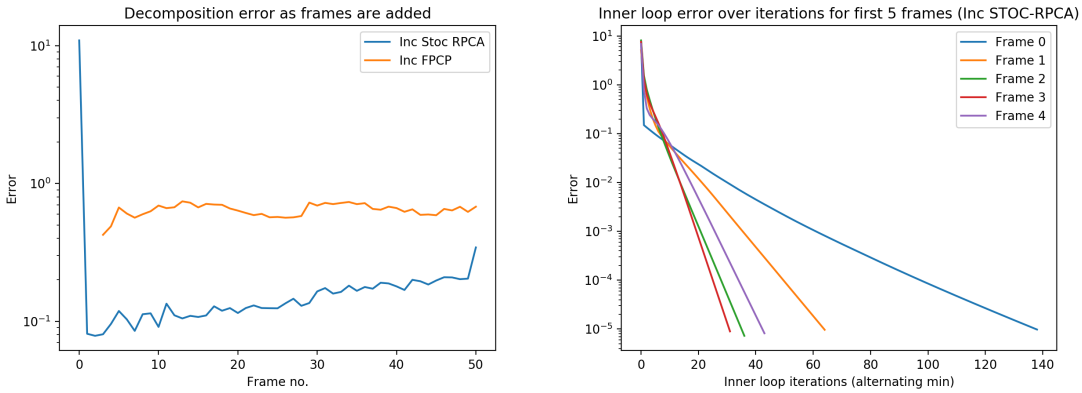


(a) STOC-RPCA

(b) Incremental Fast PCP

Figure 1.2: Selected frames from online background subtraction using STOC-RPCA, Incremental FPCP

We benchmarked the performance of the online RPCA algorithms by running them on the dataset used in [14] and visually inspecting the results. We present some of the decompositions using both STOC-RPCA and Incremental Fast PCP on the highway dataset, at intervals of 10 frames. As expected, we observe in both cases that the low-rank/sparse decomposition improves as more frames are fed into the matrix. Based solely on the visuals, it appears that STOC-RPCA is able to converge on a better decomposition than Incremental Fast PCP can. This is borne out by the presence of more artifacts at the places where cars should be, in the low-rank component of Incremental Fast PCP as compared to that of STOC-RPCA. We also notice trails behind the moving vehicles that are extracted into the sparse component in Incremental Fast PCP that are not always present in STOC-RPCA. This suggests that STOC-RPCA converges onto a better decomposition.

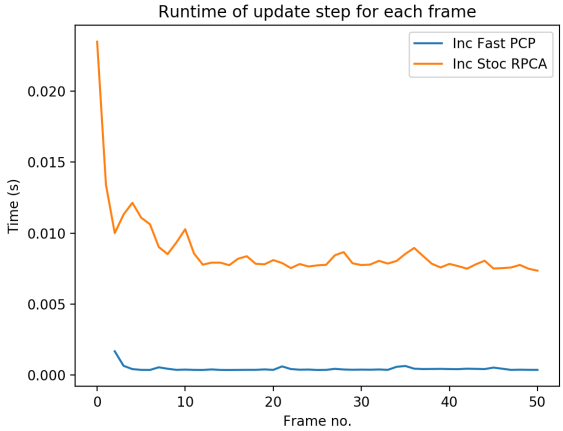


(a) Decomposition error over number of frames

(b) STOC-RPCA error over iterations

Figure 1.3: Performance of offline RPCA algorithms on image/video processing

The results in Fig. 1.3 appear to support the conjecture that STOC-RPCA yields a better decomposition. Fig 1.3(a) represents the decomposition error $\|M - L_k - S_k\|_F$ after the kth frame has been added to the matrix. We observe that STOC-RPCA has consistently lower error than Incremental Fast PCP in general, suggesting the $L + S$ is closer to the original M with STOC-RPCA than with Incremental Fast PCP. In this sense, STOC-RPCA produces a better decomposition. The large drop in error for STOC-RPCA at the beginning occurs because L, S are randomly initialized in STOC-RPCA while Incremental Fast PCP is initialized using the decomposition of the first few frames. Hence, STOC-RPCA starts off with a much larger error than Incremental Fast PCP does. This is also borne out in Fig 1.3(b), which demonstrates that far more iterations of the alternating minimization in STOC-RPCA are needed to reach convergence for the very first frame with randomly initialized L, S, than for the later frames which have relatively close estimates of L, S from earlier frames.



(a) Runtime for each update

(b) Realtime background subtraction with Incremental Fast PCP

Figure 1.4: Performance of offline RPCA algorithms on image/video processing

While STOC-RPCA may yield a better decomposition that is closer to the original image than Incremental Fast PCP, Incremental Fast PCP is clearly the faster algorithm. STOC-RPCA involves an alternating minimization procedure and coordinate descent at each update (i.e. each time a new frame appears). However, Incremental Fast PCP only requires a low-rank partial SVD with a few iterations of alternating minimization to improve the estimate of L_k, S_k. Experimentally, we have determined that we can obtain good results with Incremental Fast PCP when we use ranks as low as 1-3 and as few as 3-4 iterations of alternating minimization. As a result Incremental Fast PCP is exceedingly fast, and as shown in Fig 1.4(a), each update step for Incremental Fast PCP runs much faster than an update step for STOC-RPCA. The efficiency of Incremental Fast PCP allowed us to implement a realtime background subtraction application, with some results demonstrated in Fig 1.4(b). There are some ghosting effects, which are particularly evident in the sparse matrix, most likely due to the fact that the camera was not perfectly stable and was shaking slightly. Stabilization of the camera and perhaps more aggressive downdating of matrix might be able to help mitigate such ghosting.

5 Conclusions