REAL-TIME INCREMENTAL PRINCIPAL COMPONENT PURSUIT FOR VIDEO BACKGROUND MODELING ON THE TK1

Paul Rodríguez
Pontificia Universidad Católica del Perú, Peru
prodrig@pucp.edu.pe

ABSTRACT

Principal Component Pursuit (PCP) is currently considered to be the state of the art method for video background modeling, an important pre-processing step for automated video analysis systems. Nevertheless, PCP suffers from a number of limitations, two prominent ones being its high computational cost and the fact that it is batch method: a large number of frames have to be observed before starting any processing, resulting in a large memory requirements, usually in the order of $1 \sim 10$ *giga-bytes.*

In this work we present a real-time, GPU-enabled / CUDA-aware (unified memory model) implementation of a novel and fully incremental PCP algorithm for video background modeling: it processes one frame at a time, obtaining similar results to classical (batch) PCP algorithms, while being able to adapt to changes in the background. Our implementation has an extremely low memory footprint, and a computational complexity that allows (on the Jetson TK1 platform) a processing frame rate throughput of 27.8 and 9.4 f.p.s. for grayscale videos of 640×480 *and* 1920×1088 *respectively.*

1. INTRODUCTION

VIDEO background modeling, which consists of segmenting the moving objects or "foreground" from the static ones or "background" is an important task in several applications.

Given the importance of the video background modeling problem, recent publications (see [1] among others) have focused on presenting a systematic evaluation and comparative analysis of several Principal Component Pursuit (PCP) / Robust Principal Component Analysis (RPCA) [2, 3] based algorithms, which are considered to be the state of the art for the video background modeling problem (see [4] for a survey of alternative methods). In this context, the PCP problem is

$$\arg\min_{L,S} \|L\|_* + \lambda\|S\|_1 \;\text{ s.t. } D = L + S \tag{1}$$

where $D \in \mathbb{R}^{m\times n}$ is the observed video of n frames, each of size $m = N_r \times N_c \times N_d$ (rows, columns and depth or channels respectively), $L \in \mathbb{R}^{m\times n}$ is a low rank matrix representing the background, $S \in \mathbb{R}^{m\times n}$ is a sparse matrix representing the foreground, $\|L\|_*$ is the nuclear norm of matrix L (*i.e.* $\sum_k |\sigma_k(L)|$, the sum of the singular values of L), and $\|S\|_1$ is the ℓ^1 norm of S seen as a long vector.

Although [1] shows that PCP provides state of the art performance in the video background modeling problem, [1] also acknowledges several limitations of the PCP method, those of particular relevance to the present context being:

- it has a high computational cost (dominated by a partial SVD computation at each major outer loop);
- it is batch method in that a large number of frames have to be observed before starting any processing; this has also the side effect of very high memory requirements, usually in the order of $1 \sim 10$ giga-bytes (even higher for full HD videos).

2. OBJECTIVE

The objective of this work is to present a real-time, GPU-enabled / CUDA-aware (unified memory model) implementation of a novel incremental PCP algorithm for video background modeling on the Jetson TK1 platform.

3. STATE OF THE ART

To the best of our knowledge, within the video background modeling problem, ReProCS [5] along with GRASTA [6] and pROST [7] are the only PCP-like methods that are considered to be incremental, although some of them have a batch initialization.

ReProCS [5] is not a real-time algorithm, nor can it process real videos where multiple moving objects enter and leave the field of view of the camera; moreover [5] also assumes a known model for the motion of the video's moving objects, and uses a batch PCP method in its initialization step, which can be computationally costly.

GRASTA [6] is presented as an "online" algorithm for low rank subspace tracking: it uses a reduced number of frames, $q \ll n$, compared to the PCP problem (1), to estimate an initial low rank sub-space representation of the background and then processes each frame (which can be spatially sub-sampled) at a time. It must be emphasized that this procedure is not fully incremental, using a time sub-sampled version of all the available frames for initialization. Although GRASTA can estimate and track non-stationary backgrounds, its initialization step can have a relatively high complexity.

pROST [7] is very similar to the GRASTA algorithm, but instead of using an ℓ_1 norm of the singular values to estimate the low rank sub-space representation of the background it uses an ℓ_p norm ($p < 1$); experimental results in [7] show that pROST can outperform GRASTA in the case of dynamic backgrounds.

4. PROPOSED METHOD

Our incremental PCP algorithm, fully described in [8] (Matlab code and other simulations available in [9]), is a modified version of the amFastPCP [10], and it is able to to handle incremental and rank-1 modifications for thin SVD [11, 12, 13].

4.1 amFastPCP algorithm [10]

Instead of solving (1) directly, the amFastPCP algorithm [10] solves the equivalent alternating minimization

$$L^{(j+1)} = \arg\min_{L} \|L + S^{(j)} - D\|_F^2 \text{ s.t. } \operatorname{rank}(L) = r \tag{2}$$

$$S^{(j+1)} = \arg\min_{S} \|L^{(j+1)} + S - D\|_F^2 + \lambda\|S\|_1, \tag{3}$$

where sub-problem (2) is the one that dominates the computational complexity ((3) is solved via shrinkage) and can be solved by computing a partial (with r components) SVD of $D - S^{(j)}$.

4.2 Incremental/rank-1 modifications for thin SVD [11, 12, 13]

Given $[D\ \mathbf{d}] = U_0\Sigma_0 V_0^T$, with $\Sigma_0 \in \mathbb{R}^{r\times r}$ it is possible to efficiently compute thin SVD($[D\ \hat{\mathbf{d}}]$) $= U_1\Sigma_1 V_1^T$ with r singular values. This is accomplished by noting that $[D\ \hat{\mathbf{d}}] = [D\ \mathbf{d}] + \mathbf{c}\mathbf{e}^T$, where $\mathbf{c} = \hat{\mathbf{d}}$-$\mathbf{d}$, $\mathbf{e}$ is a unitary vector, and using (4) along with Gram-Schmidt orthonormalization of vectors $\mathbf{c}$ and $\mathbf{e}$ w.r.t. to U_0 and V_0 respectively.

$$[D\,\hat{\mathbf{d}}] = [U_0\,\mathbf{c}] \begin{bmatrix} \Sigma_0 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} V_0^T \\ \mathbf{e}^T \end{bmatrix}. \tag{4}$$

4.3 Incremental PCP Algorithm [8]

Assuming that L_{k-1} (low-rank) and S_{k-1} (sparse), where $L_{k-1} + S_{k-1} = D_{k-1}$ is a video with $k-1$ frames, have been computed and that we know the partial (thin) SVD of $L_{k-1} = U_r\Sigma_r V_r^T$, where $\Sigma_r \in \Re^{r\times r}$, then when the next frame is available (video sequence represented by $D_k = [D_{k-1}\ \mathbf{d}_k]$), we note that

$$D_k - S_k^{(0)} = [D_{k-1}\text{-}S_{k-1}\ \mathbf{d}_k] = [L_{k-1}\ \mathbf{d}_k]; \tag{5}$$

then the thin SVD of (5) can be computed in an incremental fashion: e.g. $L_k^{(1)} = \text{incrementalSVD}(D_k - S_k^{(0)})$, and so problems (2) and (3) can also be solved in an incremental fashion; this is summarized in Algorithm 1.

4.4 Implementation on the Jetson TK1

From our algorithm description (see Algorithm 1) it is clear that we need to implement the rank-1 incremental and replace SVD ("incSVD", "repSVD") and shrinkage ("shrink") routines. The latter is trivially parallelizable, whereas the two formers present some challenges: although "incSVD" and "repSVD" can be implemented using mainly level 1 BLAS routines (we used the cuBLAS library for this purpose), some intermediate operations (such SVD computation of 2×2 matrices, operation needed to cast (4) as a regular singular value decomposition) are more efficiently computed on a CPU (rather than on a GPU) and thus careful ordering of the operations must be taken into account.

Finally we mention that using rank equal to one ($r = 1$ in Algorithm 1) suffices to have a good background estimate as well as to have good tracking properties (empirically shown in [8, 14]).

Inputs : observed video $D \in \mathbb{R}^{m\times n}$, regularization parameter λ, number of inner loops *iL*.
Initialization: $L + S = D(:, 1 : k_0)$, initial rank r, $[U_r, \Sigma_r, V_r] =$ partialSVD(L, r)
for $k = k_0 + 1 : n$ **do**
  $[U_k, \Sigma_k, V_k] =$ incSVD($D(:,k), U_{k\text{-}1}, \Sigma_{k\text{-}1}, V_{k\text{-}1}$)
  for $j = 1 : iL$ **do**
    $L(:,k) = U_k(:, 1 : r) * \Sigma_k * (V_k(end,:)')$
    $S(:,k) = shrink(D(:,k) - L(:,k), \lambda)$
    if $j == iL$ **then** break
    $[U_k, \Sigma_k, V_k] =$ repSVD($D(:,k)$, $S(:,k)$, U_k, Σ_k, V_k)
  end
end

Algorithm 1: Incremental PCP algorithm.

5. COMPUTATIONAL RESULTS

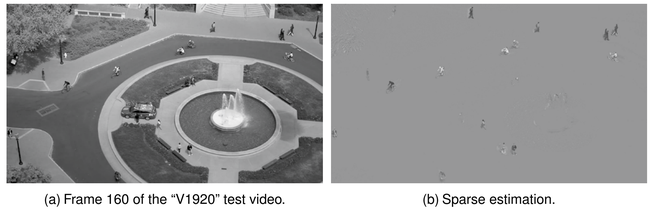
We have used three pre-recorded color video sets, nevertheless these videos are transformed (on the fly) to grayscale before any processing; the videos are labeled (i) "v640": a 640×480 pixel, (ii) "v960": a 960×544 pixel, and (iii) "v1920": a 1920×1088 pixel. Our code uses the ffmpeg library for video reading and simple manipulations (such color space transformations).

All simulations presented here have been run on a Intel i7-2670QM quad-core (2.2 GHz, 6MB Cache) based laptop and on the Jetson TK1 board (which has a quad-core ARM Cortex-A15 processor and a Keppler GPU with 192 CUDA cores). We differentiate two versions of our code: (i) it does not use any GPU acceleration (labeled "ANSI-C"), and (ii) it is CUDA aware and uses the unified memory model (labeled "GPU enabled"). We also mention that Jetson TK1 board was setup for maximum performance on the CPU and GPU.

The results listed in Table 1 show that our GPU enabled (unified memory model) implementation of our incremental PCP algorithm [8] can solve, on the Jetson TK1 platform, the PCP problem with frame rate throughput of 27.8, 22.6 and 9.4 f.p.s. for grayscale videos of 640×480, 960×544 and 1920×1088 respectively. Finally we highlight that these frame rates would not be possible using any other existing PCP algorithms.

Test video	Sparse video is displayed			Sparse video not displayed		
	ANSI-C i7-2670QM	ANSI-C Q-core A15	GPU enabled TK1	ANSI-C i7-2670QM	ANSI-C Q-core A15	GPU enabled TK1
V640	49.6	11.5	18.3	47.6	15.8	27.8
V960	23.8	8.7	15.9	29.1	11.4	22.6
V1920	7.2	2.2	4.9	7.9	2.9	9.4

Table 1: We present the processing frame rate throughput (average over at least 900 frames) for our implementation. "ANSI-C" indicates that the code does not use any GPU acceleration, whereas "GPU enabled" refers to our CUDA aware (which uses the unified memory model) code.



(a) Frame 160 of the "V1920" test video. (b) Sparse estimation.

Figure 1: Original frame (160) of the full HD "v1920" test video and corresponding sparse approximation computed on the Jetson TK1 board.

6. CONCLUSIONS

- We have presented a PCP algorithm / implementation that has an extremely low memory footprint, and a computational complexity that allows, on the Jetson TK1 platform, a processing frame rate throughput which can be considered to be real-time for most applications.
- The core operations of the proposed algorithm are mainly Level 1 BLAS operations, and can take advantage of the cuBLAS library and the unified memory model.
- Future work will target color video sequences, test on other CUDA platforms and a thoroughly assessment of the proposed algorithm / implementation.

REFERENCES

[1] T. Bouwmans and E. Zahzah, "Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance," *Computer Vision and Image Understanding*, vol. 122, pp. 22–34, 2014.

[2] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Adv. in Neural Inf. Proc. Sys. (NIPS) 22*, pp. 2080–2088, 2009.

[3] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM*, vol. 58, May 2011.

[4] M. Shah, J. Deng, and B. Woodford, "Video background modeling: recent approaches, issues and our proposed techniques," *Machine Vision and Apps.*, pp. 1–15, 2013.

[5] C. Qiu and N. Vaswani, "Automated recursive projected cs (ReProCS) for real-time video layering," in *Int'l Recognition (CVPR)*, 2012.

[6] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1568–1575, June 2012.

[7] F. Seidel, C. Hage, and M. Kleinsteuber, "pROST: a smoothed lp-norm robust online subspace tracking method for background subtraction in video," *Machine Vision and Apps.*, vol. 25, no. 5, pp. 1227–1240, 2014.

[8] P. Rodríguez and B. Wohlberg, "Incremental principal component pursuit for video background modeling," submitted, Springer Journal of Mathematical Imaging and Vision, 2015.

[9] P. Rodríguez and B. Wohlberg, "incremental PCP simulations." http://sites.google.com/a/istec.net/prodrig/Home/en/pubs/incpcp.

[10] P. Rodríguez and B. Wohlberg, "Fast principal component pursuit via alternating minimization," in *Proc. of IEEE Int'l. Conf. on Image Proc. (ICIP)*, (Australia), pp. 69–73, 2013.

[11] Y. Chahlaoui, K. Gallivan, and P. Van Dooren, "Computational information retrieval," in *In Computational Information Retrieval*, ch. An Incremental Method for Computing Dominant Singular Spaces, pp. 53–62, SIAM, 2001.

[12] C. Baker, K. Gallivan, and P. V. Dooren, "Low-rank incremental methods for computing dominant singular subspaces," *Linear Algebra and Apps.*, vol. 436, pp. 2866 – 2888, 2012.

[13] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra and its Apps.*, vol. 415, no. 1, pp. 20 – 30, 2006.

[14] P. Rodríguez and B. Wohlberg, "A matlab implementation of a fast incremental principal component pursuit algorithm for video background modeling," accepted, 2014 IEEE Int'l. Conf. on Image Proc. (ICIP), 2014.