LexisNexis® Academic

Print Request: Current Document: 2
Time Of Request: Friday, February 09, 2018 01:56:17 EST
Send To:

MEGADEAL, ACADEMIC UNIVERSE
MASTER'S COLLEGE
LIBRARY
SAN JOSE, CA 00000

Terms: (object)

Source: IET Computer Vision
Project ID:

2 of 389 DOCUMENTS

IET Computer Vision

April 2016

Parallel algorithm implementation for multi-object tracking and surveillance

BYLINE: MohamedElbahrielbahri82_m@yahoo.fr; NasreddineTaleb; KidiyoKpalma; JosephRonsin

SECTION: Pg. 202 - 211 Vol. 10 No. 3 1751-9632

LENGTH: 5448 words

1

Introduction

During the past decades, in computer vision several applications such as human-computer interaction, access control and video surveillance have emerged. At airports, train stations and several other susceptible public places, video surveillance has become an invaluable tool to ensure the safety of these places. The system must help the human to detect the abandoned luggage, recognise suspicious persons or track people. However, the illumination variation, the background changes and the full or partial occlusion, represent some examples of physical phenomena disturbing the **object** tracking systems. In the literature, many algorithms are proposed to handle these issues [1-8]. However, these existing multiperson tracking methods are not effective enough, especially when the moving **objects** leave the scene and reappear again; while the challenge is to ensure the persistence of identity of **objects** all along the video, and to make the system process in real time.

To improve the accuracy results obtained in the literature, the **object** tracking in the present work is addressed based on two approaches: detection and recognition. The **object** recognition problem is expressed as a classification system. With an online dictionary updating, each detected **object** is classified into the most representative sub-class. If the **object** appears for the first time, a new sub-class is created to label this new detected **object**. We based this part of work on sparse representation (SR) computed by [#x2113]1 minimisation, for the **object's** classification [9-13].

In the literature, there are many SR-based **object** tracking methods [14-18]. However, their computational cost increases a lot throughout the process, while the effective **object** tracking systems require a real-time process. The authors in [16] proposed an accelerated proximal gradient approach, but even in this work, the computation time depends on the parameters tuning. As a consequence, the accuracy of the results remains still costly. Wang *et al.* [19] proposed a new multikernel fusion-based SR for a real time visual tracking, but their tracker does not deal with the multi-**object** tracking case.

In the present study, we focus mainly on people tracking and we propose the orthogonal matching pursuit (OMP) [20, 21] as an SR algorithm for classification. Any **object** is classified according to the obtained sparse coefficients. The growth of the dictionary used for classification and the large size of the descriptor slow down the system. The contribution of this work is a novel parallel implementation of OMP on graphics processing unit (GPUs), adapted to the **object** tracking problem. The proposed approach can greatly benefit from GPU implementation in terms of computational efficiency.

In the rest of this paper, we discuss the following issues: Section 2 presents some related work on **object** tracking and the use of GPUs in the field of image and video processing. The principle of **object** detection and **object** representation is presented in Section 3. In Section 4 we give a brief introduction to SR and summarise the sequential OMP algorithm. The GPUs and the parallel OMP are presented in Section 5. Then, Section 6 presents and discusses the obtained experimental results. The last section concludes this paper.

2

Related works

To overcome the tracking issues mentioned above, a variety of SR-based algorithms have been proposed in the literature[14-18]. Mei *et al.* [22] used an SR algorithm for the first time for **object** tracking. The authors of this work give a SR of **objects** by a set of templates

in the initial frame, under the particle filter framework. Other authors [18, 23] employed a discriminative dictionary to encode the appearance information of both the **object** target and the background. In [7], Breitenstein *et al.*, try to handle occlusion in tracking targets; for that they propose to predict the target locations and use a particle filter based on a constant velocity model combined with a class-specific pedestrian detector to detect people. To distinguish between the tracked targets, their algorithm trains an online person-specific classifier. As features, they use a colour histogram (RGB, HS or RGI (red-green-intensity)), LBP texture features (local binary patterns) and Haar wavelets. All these features are extracted from within the bounding box of a detected target. In our work, the features are extracted from the exact region containing the body of the moving target. In [8], the authors represent, as we do in this work, the ground by a discrete grid, and they measure the probability of presence of people in each grid cell. For the **object** classification, they addressed the problem by solving a linear problem on a layered graph. Lu *et al.* [24], used the background subtraction as a technique for detection, and describe the moving **objects** with the appearance feature extracted from the entire bounding box containing this **object**. For the **object** classification Lu *et al.* used the OMP algorithm. As will be seen in the Sections 3 and 4, the kernel of our tracker is similar to the one used in the work presented in [24]; while trying to improve the representation of the **object** for a better recognition and for more accuracy.

Possegger *et al.* [25] rely on geometric cues to identify the moving **objects**; they used motion prediction and spatio-temporal evolution of occluded regions to re-assign the reappearing **objects**. Berclaz *et al.* [26] describe the tracking problem by a graph representing the spatio-temporal locations of the target, and rely to solve the problem on the k-shortest paths algorithm. Wu *et al.* [27] present the **objects** with patches and integrate the used dynamic appearance model into an online tracking system. In other research works [28-30], the authors focused on **object** presentation to handle the problem of the illumination and scale changes using robust features such as SIFT [31] or SURF [32].

The accuracy of the results implies an increase in computation time; the literature approaches mentioned above suffer from this problem. Fortunately, the apparition of GPUs revolutionised the computational efficiency in computer vision. The authors in [33] used an embedded GPU for a real time image segmentation to control a vehicle. Chiang *et al.* [34] proposed a fast implementation on GPUs for real-time content-aware video retargeting. Parallel implementations of OMP were also proposed in the literature. In [35], a parallel implementation of matrix inversion and matrix-vector multiplication is proposed for the OMP algorithm used for compressive sensing. Recently Dai *et al.* [36] proposed a 2D OMP algorithm implemented on GPU, and then applied their parallel implementation to 2D sparse signal recovery. In [35, 36], the authors used the parallel implementation to accelerate the matrix process. For our approach, many signals representing the detected **objects** are processed in parallel. Using GPUs, so that both the memory access time and the processing time will be reduced more than those of the literature.

3

Objects detection and representation

The background subtraction technique is used in the tracking systems, as a preprocessing step, for extracting moving **objects** from a video sequence [37]. In the present work, this technique is used to detect the moving people for their tracking. This process is followed by some image filtering operations as erosion and dilation [38], to reduce noise due to the illumination variation or undesirable movements, in order to get the **object** silhouettes. Fig. 1 represents the principle of constructing the proposed descriptor of the moving **object**. The binary image (Fig. 1*c*) resulting from the background subtraction serves to delineate the silhouette of the body of the moving **object**. After detection, the detected **object** is represented by a descriptor (Fig. 1*f*) defined as a concatenation of two major vectors: (i) the colour histogram representing the appearance features of the **object** and (ii) the position feature. The RGB histogram is built after dividing the body into two parts, the upper part and lower one from the center. Three vectors are obtained for each half, leading to six vectors to describe the entire body: {VRup [#x2208] [#x211d]k×1, VGup [#x2208] [#x211d]k×1, VBup [#x2208] [#x211d]k×1, $V\mathrm{R}_{\mathrm{low}}$ [#x2208] [#x211d]k×1, $V\mathrm{G}_{\mathrm{low}}$ [#x2208] [#x211d]k×1, $V\mathrm{B}_{\mathrm{low}}$ [#x2208] [#x211d]k×1} where k represents the number of grey levels in each colour channel.

Fig. 1 Proposed scheme for the **objects** detection and representation
a Background reference.
b Observed scene.
c Processed result of the background subtraction.
d Moving **objects**' silhouettes.
e Zoom of one of the detected **objects**.
f **Object's** descriptor.
g and h Colour histograms of the upper and lower body.
i **Object** center detected at block (10,16).
j Values assigned to blocks according to the **object** center
k Vector encoding the **object** position

Let *V* positon be the vector encoding the **object** position in the scene. To obtain this vector, the surface of the image is divided into small blocks (Fig. 1*i*). To each block a value *[#x3b1]* [#x2208] [#x211d]+ is assigned according to the distance separating it with the center of the detected **object**. In Fig. 1*j*, the red block corresponding to the position of the **object** center contains a value superior to the one attributed to the blue and green blocks. In this order, the value of any block decreases proportionally to the distance separating it to the **object** center. Finally, *V* positon presented in Fig. 1*k* is obtained after the linearisation of the matrix presented in Fig. 1*j*. This encoding of the **object** position is adopted to improve the **object** classification using the SR algorithm. To obtain the **object** descriptor, the six vectors describing the appearance features are concatenated together with the *V* positon vector in order to improve the **object** recognition.

In [24], Lu *et al.* encode the position in the descriptors of the **objects** as simple coordinates. This position representation performs better when the distances are measured between **objects** for the tracking. However, the first step of the OMP aims to select the most correlated atoms of the dictionary with the input signal representing the detected target. This crucial step relies on the inner product of these vectors without any measurement of distances. In contrast to [24], in our theoretical contribution, a discrete grid (Fig. 1*j*) represents the frame plane and therefore the **object** position is encoded by a vector (Fig. 1*k*) obtained after linearisation of this grid.

In [7, 24], the colour features are extracted from the bounding box containing the **object**; however, there is in this area a significant information representing the background that can falsify the representation of the **object**. To avoid this problem, our tracker uses the silhouette of the detected **object** to extract the colour features. As can be seen in Fig. 2*a*, the higher peak level *La* of the colour histogram represents the pixels of the background. It is noted that in Fig. 2*b* the level L_b is significantly reduced when the colour histogram is computed using the silhouette of the **object**. This approach makes the **object** representation more accurate.

Fig. 2 Example of colour histograms of two images
a Graph representing the colour histogram of (c)
b Graph representing the colour histogram of (d)
c Bounding box containing a detected **object** with background pixels
d Bounding box containing a detected **object** without background pixels

Although we improved the colour feature extraction in this work; the colour features-based **object** representation still have several limitations, especially when there are scale and illumination changes. Thereby, we tried to reduce these limitations by introducing SR for more **object** distinction accuracy; while in other research works [28-30], the authors handle these problems using a set of features invariant to changes in scale, rotation, viewpoint and illumination.

4

Sparse representation-based classification (SRC)

The SR-based classification of moving **object** for tracking can be expressed as follows: Let *y* [#x2208] [#x211d]$M\times1$ be the descriptor of the moving **object**. All **objects** descriptors from the previous frames are collected in a dictionary *A* [#x2208] [#x211d]$M\times N$.

In other words, the *N* **objects** descriptors collected from previous frames, are stored in the dictionary *A*. Let *no* denote the number of labeled **objects**, we regroup n_i descriptors corresponding to the *i*th **object** from the previous frames, to create the **object** class

$$A_{ci} = [v_{i1},\ v_{i2},\ v_{i3},\ \ldots,\ v_{in_i}] \in \mathbb{R}^{M\times n_i}$$

, where *v* [#x2208] [#x211d]$M\times1$, $0 < i$ [<=] *no* and

$$\sum_{i=1}^{no} n_i = N$$

. Finally, the dictionary can be expressed as the concatenation of *no* classes corresponding to *no* labeled **objects**: $A = [A_{c1}, A_{c2}, A_{c3}, \ldots, A_{c\ no}]$. (1)

$$y = A\beta + \varepsilon$$

The linear system (1) approximates the test **object** *y* detected from the current frame, where *[#x3b2]* = (*[#x3b2]*$_1$, *[#x3b2]*$_2$,..... *[#x3b2]*$_N$) [#x2208] [#x211d]*N* is the sparse vector and [#x25b] is the tolerated error. *[#x3b2]* is stated (2)

$$k-\text{sparse if } |\text{supp}(\beta)| \le k, \quad \text{where}$$

$$\text{supp}(\beta) = \{ j : \beta_j \neq 0 \}.$$

4.1

Objects classification

The **object** classification is done according to the value of the coefficients of *[#x3b2]*. The detected **object** belongs to the *i*th class, if all or the majority of the obtained non-zero coefficients of *[#x3b2]* are regrouped in the *i*th class. If the coefficients are scattered among all classes, the detected **object** is considered as not labeled before.

Fig. 3*a*, shows input signal *y* to be linearly approximated. Figs. 3*b*-*e*, represent classes of labeled **objects**. These classes constitute dictionary *A*. They contain descriptors of **objects** detected at the previous frames. As can be seen in Fig. 3*b*, the class contains three vectors representing the same **object** at different detection steps.

Fig. 3 Graphic representation of linear system y = A[#x3b2]
a Detected test **object**
b-e Represent classes of labeled **objects**

The sparse solution can be solved as a linear regression problem, and can be formulated by the following stable [#x2113]1-minimisation problem derived from the sparse solution of (1) (3)

$$(\ell^1): \ \hat{\beta} = \mathbf{argmin}\{\|y - A\beta\|_2 + \tau\|\beta\|_1\}$$

where *[#x3c4]* is a cost parameter. To classify each new test **object** represented by *y*, we compute its SR by minimising the number of non-zero entries of sparse vector *[#x3b2]* and the residual. (4)

$$\hat{\beta} = \mathbf{argmin}\,\|\beta\|_1 \quad \mathbf{subject\ to} \quad \|y - A\beta\|_2 \leq \varepsilon$$

Now, for associating the **object** represented by *y* to the *ith* class in the training set, *y* can be approximated by (5)

$$\hat{y}_i = A\beta_i$$

where

$$\beta_i = \left[0, \ .., \ 0, \ \beta_{i1}, \ \beta_{i2}, \ \ldots, \ \beta_{in_i}, \ 0, \ \ldots, \ 0\right]$$

is a vector obtained by setting to zero all entries not associated to the *i*th class.

Using this approximation, any detected **object** can be assigned to the *ith* class that satisfies (6)

$$\mathbf{class}\,(y) = \underset{i}{\mathbf{argmin}}\,\|y - \hat{y}_i\|_2 = \underset{i}{\mathbf{argmin}}\,\|y - A\beta_i\|_2$$

where 0 < *i* [<=] *no*. When non-zero coefficients are scattered over all classes, and (7)

$$\mathbf{max}\left(\sum_{l=1}^{n_i}\beta_{il}\right) < \theta\sum_{j=1}^{N}\beta_j$$

then the **object** is considered as a new class, where 0 < *[#x3b8]* < 1 is a threshold constant.

4.2

OMP algorithm

Among several algorithms proposed in the literature for SRC [39], OMP has been chosen in this work for its speed of convergence. In [24], OMP shows a great performance for **object** classification. Cai *et al.* [40] used OMP in the bounded noise cases to recover exactly the support of *[#x3b2]*. According to (6), OMP is extended to Group OMP [41] for the **object** classification. To get the sparse coefficients of *[#x3b2]*, the OMP algorithm can be stated as follows: As inputs, we have a signal *y* [#x2208] [#x211d]*M*×1 and a matrix *A* [#x2208] [#x211d]*M*×*N* containing *N* columns. *A*(*c*) is a submatrix of *A* with indices in the subset *c* [#x2286] {1, 2, ..., *N*}. We denote by *t* the number of iterations and by c_t the subset corresponding to the *t*th iteration. Once the residual r_t or the number of iterations reaches some fixed threshold, vector *[#x3b2]t* [#x2208] [#x211d]*t*×1 represents the non-zero coefficients of the sparse vector *[#x3b2]*, where *t* is the last iteration.

This discriminative approach was proposed for a more precise classification and to assign the correct identity of the **object**, however, in the OMP algorithm steps there are many matrix operations that increase the computational cost. This was a good motivation for us to parallelise the first step of the OMP algorithm based selection (Algorithm 1 see Fig. 4)

Fig. 4 OMP algorithm

5

Parallel design of the OMP on GPU

The first step of the OMP consists to select, at each iteration, the column of the dictionary that generates the largest inner product with descriptor of the test **object**. The inner product is considered as one instruction executed N times for N columns of the dictionary. In other words, there is one instruction for multiple data. This approach can be parallelised using GPUs.

5.1

GPUs and compute-unified device architecture (CUDA)

In this section, we present the GPU implementation of the proposed parallel OMP. Initially, GPUs have been introduced for the acceleration of 2-D and 3-D graphics. These cards which consist of hundreds of processor cores, are now designed to be a single instruction multiple data (SIMD) multithreaded manycore processor. Hence, GPUs are used in personal computers, workstations and other equipments as coprocessing units. In 2007, NVIDIA proposed the parallel programming model of CUDA [42]. Under the CUDA architecture, the implementation of parallel programs is quite transparent, which permits to any programmer without strong knowledge of graphics concepts to use these cards for a SIMD parallelisation. Unlike the other parallel platforms, CUDA makes easy the visual result of the processed data directly from the microprocessor. The function that executes the parallel computation on the device (GPU) is called a kernel. When the kernel is launched, many threads are regrouped in blocks and executed in parallel. Moreover, a collection of blocks is called a grid. To launch the kernel, the dimension of the grid must be specified. CUDA offers to the programmer the choice of a one-dimensional or a two-dimensional grid to regroup the threads, and once the kernel is launched, its dimensions cannot change. The block is identified using the term (blockId.x) for the one-dimension grid, and by (blockId.x, blockId.y) for the two-dimension grid. The threads indices (threadId) are one dimensional, two dimensional, or three dimensional depending on whether blocks are one, two, or three dimensional. Fig. 5 [42] shows a two-dimensional grid. The block (1,1) has its blockId.x = 1 and blockId.y = 1. The thread labeled by (2,1,0) has its threadId.x = 2, threadId.y = 1 and threadId.z = 0.

Fig. 5 Example of a two dimensional grid

The maximum number of threads in each block depends on the characteristics of the hardware. For example, the GPU used in this work is GeForce GT 630M, the maximum of thread per block is 1024. The global flexibility of distributing these threads into the three dimensions cannot exceed 1024; thus the product of the indices of a thread must be lower than 1024. For example threads (15,20,3) and (500,2,0) are allowable, while thread (15,20,4) is not allowable. As for the memory systems, the GPUs cards have a global memory and four types of on-chip memories: shared memory, texture cache, constant cache, and register files [42].

5.2

Parallel implementation of OMP using GPU

Many parallel implementations of the OMP are proposed in the literature [35, 36]. All of them focus on the matrix multiplication and the matrix inverse to reduce the processing time, the first step of the OMP that consists of computing the projection onto the linear space is parallelisable. According to the obtained results in [35], the efficiency of these implementations depends on the level of sparsity. For the signal recovery and the compressive sensing field, the OMP algorithm requires a high level of sparsity for the convergence of the residual expressed by (4). If the level is high, the size of the matrices needed to compute the projection P_t is high too. In this case, the GPUs computational efficiency is great. Contrarily, for **object** tracking, the high level of sparsity is not necessarily required to classify an input test **object**. This means that the matrices sizes are not very large. However, the dictionary size may be very large since it increases at each updating. We propose here a parallelisation of the first step of the OMP, which consists of selecting the most correlated column of the dictionary with the descriptor y of the input test **object**. This approach is chosen to reduce the processing time when the dictionary size is very large. Moreover, in order to reduce the memory access time, we propose to launch the parallel kernel for all detected **objects** at the same time in parallel.

The parallel OMP algorithm can be presented as follows: As input, all **objects** descriptors y_j where j [#x2208] {1, 2,...p} are regrouped in matrix yp [#x2208] [#x211d]$M\times p$ where p is the number of the detected **objects**. Matrix A [#x2208] [#x211d]$M\times N$ represents the dictionary and contains N columns. $A(cj)$ is a submatrix of A with indices in the subset cj [#x2286] {1, 2, ..., N}. We denote by t_j the number of iterations and by
$c_{t_j}^{j}$
the subset corresponding to the
$t_j{}^{th}$
iteration. X represents a subset of **objects** indices (Algorithm 2 see Fig. 6).

Fig. 6 Parallel OMP algorithm

As output, vectors:

$$\{\beta_1^{t_1} \in \mathbb{R}^{t_1 \times 1},\ \beta_2^{t_2} \in \mathbb{R}^{t_2 \times 1},\ \ldots,\ \beta_p^{t_p} \in \mathbb{R}^{t_p \times 1}\}$$

contain the sparse coefficients to classify the **objects** y_j, where j [#x2208] {1, 2, ...p}, and t_j is the number of iterations needed for the **objects** residuals

$$r_{t_j}^{j}$$

convergence.

For a clear expression, subset X contains all **object's** indices. When the **object's** residual converge respecting (4), the sparse solution satisfies (6), and the **object** is classified. Then, this **object** will no longer be taken into consideration by the algorithm, hence it is eliminated from subset X.

The first step of the OMP algorithm is parallelised as follows: using the CUDA architecture, a parallel kernel is launched with a two-dimensional grid (Fig. 7*b*). Each thread in the block has also two-dimensional indices (threadId.x, threadId.y). Indice threadId.x aims index j of **object** y_j in matrix yp. Indice threadId.y aims index i of column A_i of matrix A. Each thread (threaded.x, threadId.y), computes the inner product of the $j$threadId.x **object** descriptor with column $A_{\text{threadId.}y}$. Fig. 7*a*. shows thread (0, 2) designed to compute the inner product of **object** number 0 with column number 3 of the dictionary A.

Fig. 7 Employed threads
a Thread designed to execute the inner product between y0 and A2
b block of threads
c **Objects** detected in the current frame
d Dictionary containing all descriptors of the **objects** detected in the previous frames

The total number of threads executed is $p \times N$. As mentioned in Section 5.1, the maximum number of threads per block is limited to 1024 for the used card. We set in this work the maximum indices for each thread at (5, 200). Thus, if $(p \times N) > (5 \times 200)$, one block will not be enough to satisfy all **objects** and columns. For that, the grid dimension depends on the number of **objects** and the size of the matrix A. To have enough threads, the kernel is launched with a two dimensional grid with a size of (gridDim.x) × (gridDim.y). In this case, gridDim.$x = p/5$ and gridDim.$y = N/200$. For example, if the tracker detects 15 persons and the dictionary contains 1000 columns, a grid of (3 × 5) blocks is needed to execute (15 × 1000) threads.

6

Experimental assessment

The proposed approach is evaluated on the challenging benchmark video PETS '09 dataset [43, 44], in order to validate the proposed SR-based tracking algorithm accuracy, and compare its results with those of the methods of the literature: MTUSR [24], ETHZ [7] and EPFL [8]. The execution time of the parallel OMP is also compared with that of the sequential OMP. The proposed **object** descriptor size is 1936 floats, necessary to represent both the appearance feature and the position feature.

6.1

Experimental environment

The proposed system is implemented in bouth C++ & CUDA, and runs on a laptop with an Intel Core (TM) i7 CPU and 8 GB RAM. The laptop is equipped with an NVIDIA GeForce GT 630M GPU card. This card has two multiprocessors and 96 CUDA cores with 1 GB of global memory. The GPU card used in this work has a limited memory size compared with other professional cards dedicated to accelerate the processing time.

6.2

Database PETS'09

The PETS'09 video contains 795 frames recorded from a high viewpoint in a campus. Ten persons with similar colour are moving and walking in and out of the frames. The second row of Table 1 describes the number of entries in the scene of each person. Most persons intersect each others making the recover process a hard task, especially when the intersecting persons wear clothes of similar colours. In our result, the first person who appears in the scene is labeled with the number 0 and the last person with the number 9. In MTUSR and EPFL the authors stared with the number 1, and for ETHZ, they discriminated the persons with different colours.
Table 1 **Objects** entries and reappearances in the scene, the F value means false **object** initialisation. The last column (NE) contains the

number of errors of each method

Objects	0	1	2	3	4	5	6	7	8	9
Number of entries	2	4	2	2	2	2	1	1	2	1
Method	**Detected entries**									**NE**
EPFL	1	2	1	1	1	1	1	1	2	1	7
MTUSR	2	4	2	1	1	2	1	1	2	1	2
ETHZ	2	4	2	1	1	1	1	1	F	F	6
proposed approach	2	4	2	2	2	2	1	1	2	1	0

6.3

Object tracking evaluation

As a result, all persons are correctly initialised and recognised [45]. They are indexed starting from 0 according to the order of their appearance in the video. The proposed algorithm deals with all the issues. Fig. 8 shows a visual result of our approach for tracking and recognition in frame_263 and frame_717. As can be seen, each person is labeled with a colour and a number placed on top of its bounding box. The left side and the right side of Figs. 8*a* and *b* contain a zoom of the detected persons. It is noted that, even after 500 frames, the **objects**' identity is the same. There is a confusion between intersecting people when their clothes are similar. As can be seen in Fig. 9, the EPFL method is not very robust to handle these situations, and the overlapping reverses the indices of **objects** 6 and 8 between frame_0667 and frame_0675. The first three approaches dealt successfully with these cases. In Fig. 10, the proposed method recovers the reentering **objects** indexed 3 and 4 in frame_0031, frame_0535 and frame_0626. The MTUSR method considers the **objects** indexed 4 and 5 as a new **object** indexed 10 in frame_0535. In frame_0626, only **object** 4 is recovered by MTUSR. ETHZ does not detect the **object** in frame_0535 and switches two **object** identity in frame_0626. In contrast to ETHZ, MTUSR and our approach, The EPFL method does not rely on any online trained dictionary; thereby, this approach cannot re-assign correctly the **object** which leaves the scene and reappears again and considers it as a new one. As can be seen, the identities of the detected **objects** in frame_0031 change in the next frames. Furthermore, in frame_0535 and frame_0626, the detected **objects** are indexed as 15 and 17, while there are only 10 **objects** that appear in the video.

Fig. 8 Screen shot of the obtained result video
a Detected and indexed **objects** at frame_263
b Detected and indexed **objects** at frame_ 717

Fig. 9 Illustration of the problem of switching identities caused by overlapping: visual comparison of our method with MTUSR, ETHZ and EPFL

Fig. 10 Initialisation and the recovery process: comparison of our method with MTUSR, ETHZ and EPFL

The number of the **objects** entries and reappearances in the scene is summarised in Table 1. According to Table 1, MTUSR, ETHZ and EPFL have respectively 2, 6 and 7 initialisation and recovery errors of the deteted **objects**. It is to note that Table 1. does not show the **object** misdetection errors. The proposed approach is really robust in re-assigning the correct indices of the **objects** once detected; this is due to the use of the SR algorithm for the **object** classification and the dictionary learning.

In addition to the visual comparisons, we use some performance metrics [46]: multiple **object** tracking precision (MOTP) and multiple **object** tracking accuracy (MOTA). MOTP aims at evaluating the positive trajectories of the targets, while MOTA, composed of false negative rate, false positive rate and number of identity switches, measures the accuracy. We measured both MOTP and MOTA on the used benchmark. We obtained 70% of precision (MOTP) and 85% of accuracy (MOTA). Using the background subtraction as a technique of detection has some limitations especially when the **object** is occluded by another one or when two **objects** are very close to each other. These misdetection cases reduce our tracker MOTP to 70%. The MOTA metric of our approach is also affected by these limitations at the detection step, but our tracker based on SR for the **object** recognition performs very well in comparison with other approaches. Table 2 shows the MOTP and MOTA metrics of our approach and compares them with other works using the same benchmark. Based on this table, one can conclude that the proposed approach gives satisfactory results despite the limitation of background subtraction.
Table 2 MOTP and MOTA evaluation results

Method	MOTP, %	MOTA, %
ETHZ [7]	52	50
EPFL [8]	<50	<50
Possegger *et al.* [25]	64,8	66
Berclaz *et al.* [26]	50	58
Wu *et al.* [27]	73,3	73,2
proposed	70	85

6.4

Execution time

The processing times of the parallel implementation of the first step of the OMP algorithm, running on GPU is compared with the respective runtime on CPU. The complexity of the inner product between the vector containing the **object** descriptor and the columns of dictionary A is dominated by the matrix ($M \times N$) and vector ($M \times 1$) multiplication. Hence, the complexity of this step is $O(MN)$. In [36], the authors demonstrate that the complexity of this step is very important. As can be seen in Fig. 12, the complexity is approximately reduced to $O(\log MN)$.

At any frame, there are p **objects** and N columns. The results of some experiments presented in Table 3 show the process time of the first step according to the number of **objects** detected and the sizes of the dictionary. From Table 3 and Fig. 11, we can see that the GPU computation efficiency is very high since the processing time is significantly reduced when using GPUs. We can note also that our approach performs better in terms of reducing the computation time when the number of **objects** increases more. The proposed parallel implementation has led to some better runtime performances of 10-22 frames per second (fps) excluding the detection step. In [7, 27], the runtime performances are respectively 1-2 fps and 0.4-2 fps. The execution time, in [7, 27] and our implementation depends on the number of the detected **objects**. In [8, 25], an average of 0.25 fps and 2.1 fps has been obtained, respectively. These approaches are implemented in sequential style on a standard PC with a 3.4 GHz Intel CPU [25] and also excluding the detection step. In [26], the authors claim that their tracker performs in real time exceeding 25 fps.

Fig. 11 Processing times for the parallel approach and the sequential one.
The horizontal axes represent the size of the dictionary, and the vertical axes the ratio of the processing time between GPU and CPU

Fig. 12 Processing times for the parallel approach and the sequential one. The horizontal axes represent the size of the dictionary, and the vertical axes the processing time
a Graph of the parallel processing time of the inner product for one person
b Graph of the sequentiel processing time of the inner product for one person
c Graph of the parallel processing time of the inner product for 5 people
d Graph of the sequentiel processing time of the inner product for 5 people
e Graph of the parallel processing time of the inner product for 10 people
f Graph of the sequentiel processing time of the inner product for 10 people

Table 3 Obtained processing times of the inner product per the number of detected **objects** and the dictionary sizes

	1 **object**			5 **objects**			10 **objects**		
Dictionary size	GPU (ms)	CPU (ms)	Ratio (CPU/GPU)	GPU (ms)	CPU (ms)	Ratio (CPU/GPU)	GPU (ms)	CPU (ms)	Ratio (CPU/GPU)
100	0,9	3	3,33	1	14	14	1,08	26	24,07
200	0,93	5	5,37	1,41	24	17,02	1,56	115	73,71
300	0,98	9	9,18	1,54	48	31,16	2,62	197	75,19
400	1,06	12	11,32	1,58	68	43,03	3,04	240	78,94
500	1,92	17	8,85	2,63	92	34,98	4,21	366	86,93
600	1,94	20	10,30	2,99	100	33,44	4,59	400	87,14
700	2,06	23	11,16	3,12	129	41,34	5,83	480	82,33
800	2,14	29	13,55	3,14	146	46,49	6,03	591	98,00
900	2,89	32	11,07	4,25	161	37,88	7,27	630	86,65
1000	2,92	34	11,64	4,62	165	35,71	7,75	720	92,90

7

Conclusion

We have presented in this paper, a multi-**object** tracking system for video surveillance based on SR. Both, the proposed **object** description and the SR-based classification show a promising solution to ensure the identity of the moving **objects**. We have demonstrated that some steps of the OMP algorithm having a high level of complexity can be highly parallelisable and can be upgraded with a GPU implementation. The contribution of the used approach seems very advantageous since the performances in terms of reducing the computation time achieve up to 100× speedup, especially in crowded scenes. Since the SR and the use of GPUs for a parallel implementation have their own advantages, we will explore in future works the multiview **object** detection and recognition for more robustness of the video surveillance.

LOAD-DATE: March 17, 2016

LANGUAGE: ENGLISH

BIBLIOGRAPHY:

REFERENCES

1 Huo F., Hendriks E.A.: 'Multiple people tracking and pose estimation with occlusion estimation', Comput. Vis. Image Underst., 2012, 116, pp. 634-647 (doi: 10.1016/j.cviu.2011.12.006)

2 Andriluka M., Roth S., Schiele B.: 'People-tracking-by-detection and people-detection-by-tracking'. IEEE Conf. on CVPR, 2008, pp. 1-8

3 Bercla J., Fleuret F., Fua P.: 'Robust people tracking with global trajectory optimization'. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 2006, vol. 1, pp. 744-750

4 Huang C., Wu B., Nevatia R.: 'Robust **object** tracking by hierarchical association of detection responses', Proc. European Conf. 'Computer Vision-ECCV', 2008, (Springer, Berlin Heidelberg), pp. 788-801

5 Leibe B., Schindler K., Cornelis N., : 'Coupled **object** detection and tracking from static cameras and moving vehicles', IEEE Trans. Pattern Anal. Mach. Intell., 2008, 30, (10), pp. 1683-1698 (doi: 10.1109/TPAMI.2008.170)

6 Chen X., Qin Z., An L., : 'An online learned elementary grouping model for multi-target tracking'. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2014, pp. 1242-1249

7 Breitenstein M.D., Reichlin F., Leibe B., : 'Online multiperson tracking-by-detection from a single, uncalibrated camera', IEEE Trans. Pattern Anal. Mach. Intell., 2011, 33, (9), pp. 1820-1833 (doi: 10.1109/TPAMI.2010.232)

8 Ben Shitrit H., Berclaz J., Fleuret F., : 'Tracking multiple people under global appearance constraints'. IEE Int. Conf. on Computer Vision (ICCV), November 2011, pp. 137-144

9 Wright J., Yang A.Y., Ganesh A., : 'Robust face recognition via sparse representation', IEEE Trans. Pattern Anal. Mach. Intell., 2009, 31, (2), pp. 210-227 (doi: 10.1109/TPAMI.2008.79)

10 Chen S.S., Donoho D.L., Saunders M.A.: 'Atomic decomposition by basis pursuit', SIAM J. Sci. Comput., 1998, 20, (1), pp. 33-61 (doi: 10.1137/S1064827596304010)

11 Yan Q., Li L.: 'Kernel sparse tracking with compressive sensing', IET Comput. Vis., 2014, 8, (4), pp. 305-315 (doi: 10.1049/iet-cvi.2013.0095)

12 Aharon M., Elad M., Bruckstein A.: 'The K-SVD: an algorithm for designing of over complete dictionaries for sparse representation', IEEE Trans. Signal Process., 2006, 54, (11), pp. 4311-4322 (doi: 10.1109/TSP.2006.881199)

13 Xie C., Tan J., Chen P., : 'Multiple instance learning tracking method with local sparse representation', IET Comput. Vis., 2013, 7, (5), pp. 320-334 (doi: 10.1049/iet-cvi.2012.0228)

14 Shao J., Dong N., Tong M.: 'Multi-part sparse representation in random crowded scenes tracking', Pattern Recogn. Lett., 2013, 34, (7), pp. 780-788 (doi: 10.1016/j.patrec.2012.07.008)

15 Shen Y., Miao Z.: 'Multi-human tracking from sparse detection responses', IET Comput. Vis., 2012, 6, (6), pp. 590-602 (doi: 10.1049/iet-cvi.2011.0198)

16 Bao C., Wu Y., Ling H., : 'Real time robust l1 tracker using accelerated proximal gradient approach'. , IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2012, pp. 1830-1837

17 Han Z., Jiao J., Zhang B., : 'Visual **object** tracking via sample-based Adaptive Sparse Representation (AdaSR)', Pattern Recogn., 2011, 44, (9), pp. 2170-2183 (doi: 10.1016/j.patcog.2011.03.002)

18 Wang Q., Chen F., Xu W., : 'Online discriminative **object** tracking with local sparse representation'. IEEE Workshop on Applications of Computer Vision (WACV), January 2012, pp. 425-432

19 Wang L., Yan H., Lv K., : 'Visual tracking via kernel sparse representation with multikernel fusion', IEEE Trans. Circuits Syst. Video Technol., 2014, 24, (7), pp. 1132-1141 (doi: 10.1109/TCSVT.2014.2302496)

20 Pati Y.C., Rezaiifar R., Krishnaprasad P.S.: 'Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition', Signals Syst. Comput., Conf. Record of The Twenty-Seventh Asilomar Conf. on IEEE, November 1993, pp. 40-44 (doi: 10.1109/ACSSC.1993.342465)

21 Mallat S.G., Zhang Z.: 'Matching pursuits with time-frequency dictionaries', IEEE Trans. Signal Process., 1993, 41, (12), pp. 3397-3415 (doi: 10.1109/78.258082)

22 Mei X., Ling H.: 'Robust visual tracking and vehicle classification via sparse representation', IEEE Trans. Pattern Anal. Mach. Intell., 2011, 33, (11), pp. 2259-2272 (doi: 10.1109/TPAMI.2011.66)

23 Collins R.T., Liu Y., Leordeanu M.: 'Online selection of discriminative tracking features', IEEE Trans. Pattern Anal. Mach. Intell., 2005, 27, (10), pp. 1631-1643 (doi: 10.1109/TPAMI.2005.205)

24 Lu W., Bai C., Kpalma K., : 'Multi-**object** tracking using sparse representation'. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), May 2013, pp. 2312-2316

25 Possegger H., Mauthner T., Roth P.M., : 'Occlusion geodesics for online multi-**object** tracking'. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), June 2014, pp. 1306-1313

26 Berclaz J., Fleuret F., Türetken E., : 'Multiple **object** tracking using k- shortest paths optimization', IEEE Trans. Pattern Anal. Mach. Intell., 2011, 33, (9), pp. 1806-1819 (doi: 10.1109/TPAMI.2011.21)

27 Wu Z., Zhang J., Betke M.: 'Online motion agreement tracking'. Proc. BMVC, 2013

28 Guo Y., Chen Y., Tang F., : '**Object** tracking using learned feature manifolds', Comput. Vis. Image Underst., 2014, 118, pp. 128-139 (doi: 10.1016/j.cviu.2013.09.007)

29 Zhou H., Yuan Y., Shi C.: '**Object** tracking using sift features and mean shift', Comput. Vis. Image Underst., 2009, 113, pp. 345-352 (doi: 10.1016/j.cviu.2008.08.006)

30 He W., Yamashita T., Lu H., : 'Surf tracking'. IEEE 12th Int. Conf. on Computer Vision, September 2009, pp. 1586-1592

31 Lowe D.G.: 'Distinctive image features from scale-invariant keypoints', Int. J. Comput. Vis., 2004, 60, (2), pp. 91-110 (doi: 10.1023/B:VISI.0000029664.99615.94)

32 Bay H., Ess A., Tuytelaars T., : 'Speeded-up robust features (SURF)', Comput. Vis. Image Underst., 2008, 110, (3), pp. 346-359 (doi: 10.1016/j.cviu.2007.09.014)

33 Abramov A., Pauwels K., Papon J., : 'Real-time segmentation of stereo videos on a portable system with a mobile gpu', IEEE Trans. Circuits Syst. Video Technol., 2012, 22, (9), pp. 1292-1305 (doi: 10.1109/TCSVT.2012.2199389)

34 Chiang C.K., Wang S.F., Chen Y.L., : 'Fast JND-based video carving with GPU acceleration for real-time video retargeting', IEEE Trans. Circuits Syst. Video Technol., 2009, 19, (11), pp. 1588-1597 (doi: 10.1109/TCSVT.2009.2031462)

35 Fang Y., Chen L., Wu J., : 'Gpu implementation of orthogonal matching pursuit for compressive sensing'. Parallel and Distributed Systems (ICPADS), December 2011, pp. 1044-1047

36 Dai Y., He D., Fang Y., : 'Accelerating 2D orthogonal matching pursuit algorithm on GPU', J. Supercomput., 2014, 69, (3), pp. 1363-1381 (doi: 10.1007/s11227-014-1188-8)

37 Amato A., Mozerov M.G., Roca F.X., : 'Robust real-time background subtraction based on local neighborhood patterns', EURASIP Adv. Signal Proc, 2010, 34, pp. 1-7

38 Babu J.J.J., Sudha G.F.: 'Non-subsampled contourlet transform based image denoising in ultrasound thyroid images using adaptive binary morphological operations', IET Comput. Vis., 2014, 8, (6), pp. 718-728 (doi: 10.1049/iet-cvi.2014.0008)

39 Efron B., Hastie T., Johnstone I., : 'Least angle regression', Annals Stat., 2004, 32, (2), pp. 407-499 (doi: 10.1214/009053604000000067)

40 Cai T.T., Wang L.: 'Orthogonal matching pursuit for sparse signal recovery with noise', IEEE Trans. Inf. Theory, 2011, 57, (7), pp. 4680-4688 (doi: 10.1109/TIT.2011.2146090)

41 Swirszcz G., Abe N., Lozano A.C.: 'Grouped orthogonal matching pursuit for variable selection and prediction'. Advances in Neural Information Processing Systems, 2009, pp. 1150-1158

42 Lindholm E., Nickolls J., Oberman S., : 'NVIDIA Tesla: a unified graphics and computing architecture', IEEE Micro, 2008, (2), pp. 39-55 (doi: 10.1109/MM.2008.31)

43 PETS 2009 Benchmark Data 'http://cs.binghamton.edu/~mrldata/pets2009'

44 Ferryman J., Ellis A.L.: 'Performance evaluation of crowd image analysis using the PETS2009 dataset', Pattern Recogn. Lett., 2014, 44, pp. 3-15 (doi: 10.1016/j.patrec.2014.01.005)

45 Our results: https://www.youtube.com/watch?v=8WX2Rvn36mQ

46 Bernardin K., Stiefelhagen R.: 'Evaluating multiple **object** tracking performance: the CLEAR MOT metrics', J. Image Video Process., 2008, 1, pp. 1-10 (doi: 10.1155/2008/246309)

PUBLICATION-TYPE: Magazine

---- End of Request ----
Print Request: Current Document: 2
Time Of Request: Friday, February 09, 2018 01:56:17 EST