Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Digital Movie Line Scratch Restoration by Fusion Techniques

L. Maddalena, A. Petrosino, G. Lancetti

RT-ICAR-NA-06-01 **01-2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Digital Movie Line Scratch Restoration by Fusion Techniques[1]

L. Maddalena[2], A. Petrosino[3], G. Lancetti[4]

Rapporto Tecnico N.:
RT-ICAR-NA-06-01

Data:
01-2006

[1] sottoposto per pubblicazione
[2] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli
[3] Università di Napoli Parthenope, Dipartimento di Scienze Applicate, Via A. De Gasperi 5, 80133 Napoli
[4] Università di Napoli "Federico II", Via Cintia, 80126 Napoli

Digital Movie Line Scratch Restoration by Fusion Techniques

G. Laccetti[1], L. Maddalena[2], and A. Petrosino[3] *Senior Member, IEEE*

[1] University of Naples “Federico II”, Via Cintia, 80126 Naples, Italy
giuliano.laccetti@dma.unina.it

[2] Italian National Research Council, ICAR, Via P. Castellino 111, 80131 Naples, Italy
lucia.maddalena@na.icar.cnr.it

[3] University of Naples “Parthenope”, Via A. De Gasperi 5, 80133 Naples, Italy
alfredo.petrosino@uniparthenope.it

Abstract

Many algorithms have been proposed in literature for digital movie restoration; unfortunately, none of them ensures a perfect restoration whichever is the image sequence to be restored. Here, we propose a new methodology applied to digital scratch restoration, based on image fusion techniques for combining relatively well settled distinct techniques. The resulting algorithm has large memory requirements and is computationally intensive; due to this main reason, we propose a parallel version of the restoration approach, focusing on strategies based on both task and data partitioning to achieve good load balancing and scalability. Such approach well adapts also to be distributed.

Extensive experiments on real image sequences deeply investigate both qualitative results of the presented scratch restoration approach and performance of its parallel implementation.

Index Terms

Digital processing, Scratch restoration, Data fusion techniques, Parallel/distributed computing.

Manuscript created October 18, 2005. This work was supported by the FIRB Project *Enabling platforms for high-performance computational grids (grid.it)*, funded by MIUR, Italy.

I. INTRODUCTION

With the recent advent of digital technologies and the ever increasing need for speed and storage, occluded or missing parts in images and movies is a more and more widespread problem. The problem can occur in several multimedia applications, such as wireless communication and digital movie restoration. The specific application we are concerned with is digital movie restoration, which is the set of image processing methodologies and techniques allowing to reproduce digital copies of damaged movies that are as much as possible similar to the uncorrupted lost original. Several classes of defects can be distinguished that affect movies; in the present paper, we focus on the class of scratch defects, intended as long and thin vertical scratches that affect several subsequent images of a sequence, due to the abrasion of the film by dust particles in the slippage mechanisms used for the development, projection and duplication of the film.

Many scratch restoration methods are reported in literature; as expected, advantages and disadvantages characterize each scratch restoration technique, and any of them could be said to win the competition. A way to deal with this kind of problems is to adopt *fusion techniques* (see for instance [1], [2]): input images which provide alternative and complementary "views" and "characteristics" of a given area are "fused" into a single image in such a way that all the important visual information found in input images is transferred into the fused output image, without the introduction of artifact distortion effects. In this sense machine vision systems can be organized as a set of separated visual modules that act as virtual sensors. In this paper, the term *visual module* indicates an algorithm that extracts information of some specific and descriptive kind from a numerical image.

For what concerns digital scratch restoration, fusion techniques may be applied to both the stages of the automatic restoration process (see §II). Here we propose an approach to line scratch restoration based on fusion techniques which for both the stages takes into account already existing promising algorithms and suitably combines the obtained results in order to provide a restored sequence as similar as possible to the original uncorrupted sequence. As we show, the results produced by the proposed approach upon different damaged movies greatly enhance those produced by each considered approach.

On the other side, automatic batch processing is limited to low degraded video sections and is mainly aimed at the removal of speckle noise and brightness variations. The manual inpainting of high definition digital video is, of course, a very intensive and expensive activity which can be justified in cases of documents of particular historical and artistic meaningfulness. A great improvement in this field should be related to the development of high performance software for the automatic or semiautomatic restoration of degraded image sequences; this could drastically reduce the costs and efforts for the restoration of whole movies, at the same time allowing the use of highly specialized algorithms. Therefore, we present a parallel version of the proposed approach, based on both task and data partitioning strategies, and analyze its performance on real image sequences.

Finally, we show that the proposed algorithm is specifically suited for (and indeed it has been devised for) *grid environments* [3], where several geographically distributed resources can be accessed and exploited for the solution of the problem under investigation.

The contents of this paper are as follows: in §II we describe the scratch restoration problem we are concerned with, outlining the usual restoration process and giving a brief overview of methods reported in literature. In §III we outline the proposed methodology and detail its implementation, giving a brief description of the existing restoration algorithms and of the fusion strategies adopted. In §IV we present the parallelization strategy and a performance model for the parallel algorithm. In §V we describe qualitative results achieved by the proposed approach tested on real image sequences and present experimental results concerning parallel execution times and speedup. In §VI we present ongoing researches concerning a distributed version of the parallel algorithm in the context of grid computational environments. Conclusions are reported in §VII.

II. LINE SCRATCH RESTORATION

A sufficiently general model of degraded video signal is the following for a pixel location $\mathbf{p} = (x, y)$:

$$I(\mathbf{p}, t) = (1 - b(\mathbf{p}, t))I^*(\mathbf{p}, t) + b(\mathbf{p}, t)c(\mathbf{p}, t) + \eta(\mathbf{p}, t) \ , \tag{1}$$

where $I(\mathbf{p}, t)$ is the corrupted signal at spatial position $\mathbf{p}$ in frame t, $b(\mathbf{p}, t) \in \{0, 1\}$ is a binary mask indicating the points belonging to missing parts of the degraded video, I^* is the ideal uncorrupted image. The (more or less constant) intensity values at the corrupted spatial locations are given by $c(\mathbf{p}, t)$. Though noise is not considered to be the dominant degrading factor in the defect domain, it is still included in (1) as the term $\eta(\mathbf{p}, t)$.

In the present paper, we focus on the class of line scratch defects, intended as long and thin vertical scratches that affect several subsequent images of a sequence, due to the abrasion of the film by dust particles in the slippage mechanisms used for the development, projection and duplication of the film.

Commonly, scratch restoration is a two-step procedure. In the first step the scratches need to be detected, i.e., an estimate for the mask $b(\mathbf{p}, t)$ is made (*detection step*). In the second step the values of I^* inside the scratch, possibly starting from information about $c(\mathbf{p}, t)$, are estimated (*removal step*). As usual, we consider scratch reduction as a problem of detection and removal of missing data, i.e. we suppose that any information $c(\mathbf{p}, t)$ has been lost within the scratch.

Several scratch restoration methods are reported in literature.

The detection step is usually accomplished through low/high-pass filters [4], morphological filters [5], [6], [7], [8], [9], [10], [11], adaptive binarization [12], discrete wavelet decomposition [13], statistics and MAP techniques [14], [15], [16], or local gradient measures [17], [18], eventually coupled with techniques such as Hough transform [10], [4], [19], [18] or Kalman filter [6], [7], [8], [9], [10], and possibly followed by Bayesian refinement strategies [4], [19].

The removal step is obtained through interpolation or approximation [13], [10], [18], [20], [15], [21], the adoption of autoregressive models [4], [19], morphological filters [5], or mean vector filters [12], eventually followed by the reconstruction of the image grain (high-frequency components) via Fourier series [9], MAP techniques [10] or least squares-based grain estimation [20]. Scratch removal techniques can also be borrowed by *image inpainting*, that is the set of techniques for making undetectable modifications to images [22]. Such techniques are generally used to fill-in missing data or to substitute information contained in small image regions [23]. Inpainting has been pursued in literature also under different names, such as *image interpolation* (e.g. [24]) and *fill-in* (e.g. [25], [26]); the problem has been afforded also as *disocclusion*, since missing data can be considered as occlusions hiding the image region to be reconstructed (e.g. [28], [29]). Finally inpainting is also related to *texture synthesis*, where the problem consists in generating, given a sample texture, an unlimited amount of image data which will be perceived by humans as having the same texture [30]; specifically, inpainting can be considered as a specific constrained texture synthesis problem [31], [26], [27]. Following also [32], inpainting methods reported in literature can be classified as devoted to reconstructing only the image structure or geometry (structure inpainting; some results are reported in [25], [33], [34], [35], [36], [37], [38], [29], [39], [40]), only the image texture (texture synthesis; some results are reported in [41], [31], [42], [43], [44], [30], [45], [46], [47], [48], [49], [50], [51], [26], [27], [52], [53], [54], [55]), or both (e.g. [32], [56]).

It should also be noted that some literature [57], [58] approaches the more general missing data problem with a joint detection and removal methodology including also motion reconstruction.

Even though many of the cited algorithms achieve very accurate results, none of them ensures a perfect restoration whichever is the image sequence to be restored and whichever is the kind of scratch by which it is affected; therefore scratch restoration is still considered an open problem [59].

III. LSR: Line Scratch Restoration algorithm

The proposed algorithm for scratch restoration in image sequences is based on a new methodology for the solution of classes of problems to be dealt with in digital movie restoration, which takes into account already existing promising algorithms and suitably combines the obtained results in order to provide a restored sequence as similar as possible to the original uncorrupted sequence [60]. In the case of line scratch restoration, the basic idea of the *compound algorithm* consists, for each sequence frame, in:

1) applying a set of d existing scratch detection algorithms;
2) combining obtained scratch masks $B^j, j = 1, \ldots, d$, to produce the final scratch mask B^C;
3) applying a set of r existing scratch removal algorithms using scratch mask B^C;
4) combining obtained restored images $R^j, j = 1, \ldots, r$, to produce the final restored image R^C.

For the implementation of the LSR algorithm we have considered as underlying restoration modules $d = 3$ detection algorithms and $r = 2$ removal algorithms, briefly described in §III-A and §III-B. Moreover, for the combination of results we adopted suitable fusion techniques, briefly described in §III-C.

A scheme for our implementation of LSR algorithm is given in Fig. 1.

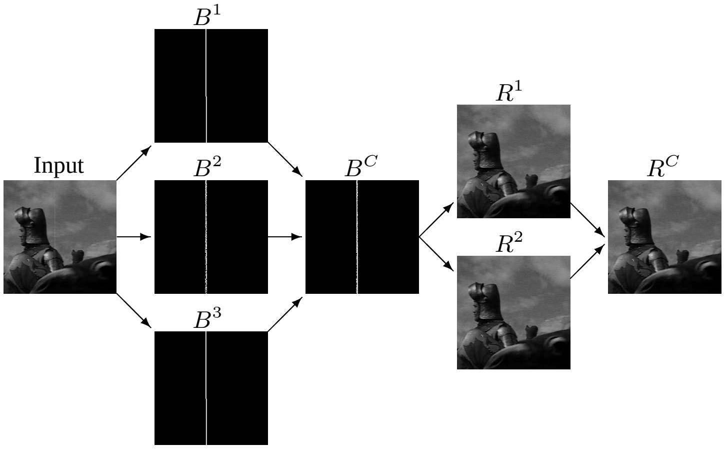


Fig. 1. Scheme of the LSR algorithm.

A. Detection algorithms

In this section we consider the problem of scratch detection, that is, given the corrupted image sequence I, construct an estimate B for the mask b in (1). In the following some hints will be provided about the visual modules performing detection that have been chosen for the implementation of the LSR algorithm.

1) Detection algorithm 1: Following [9], the first detection algorithm taken into account can be described as follows:

- pre-processing of each sequence frame, obtained by using the Radon projection in the vertical direction, with height S depending on the maximum inclination of the scratch to the vertical direction and on the image spatial resolution:
$$I_S(x,y) = \sum_{j=0}^{S-1} I(x, j + S * y)$$
producing an image I_S with height N/S, N being the height of the original image;
- vertical linear structures detection in subsampled images I_S by a local maximum detection procedure based on a top-hat morphological transformation, with structuring element of dimension $w+1$, followed by a thresholding process and by the line Hough transform;
- creation of the scratch mask B^1 for each sequence frame, using the results obtained in the previous phase on subsampled images I_S.

The produced results could be enhanced, by retaining only the scratches that appear fixed in time; as suggested in [9] this can be made by tracking the scratches. Anyway to maintain comparable the computational complexities of the considered detection algorithms, we did not implement this part of the procedure.

2) Detection algorithm 2: In [12] the detection scheme consists of:

- adaptive binarization of each sequence frame, producing a binary image C, by adopting a threshold σ computed over a horizontal neighborhood of any pixel at position (x,y) as:
$$\sigma = 1/n \sum_{i=-n/2}^{n/2} I(x+i, y),$$
where $n = w+1$ for even w and $n = w+2$ for odd w;
- construction of the scratch mask B^2 by labelling as belonging to the scratch all the pixels of columns x of C that contain at least 70% of pixels with value $C(x,\cdot) = 1$.

3) Detection algorithm 3: Following [4], the third detection algorithm taken into account consists of:

- low-pass filtering with a Gaussian vertical filter, followed by a vertical subsampling of width S, producing an image I_S with height N/S;
- location of the horizontally impulsive features in I_S by:
  - horizontal median filtering I_S, in order to eliminate straight lines with width lower than the filter window size $2w+1$, producing I_M;

  - thresholding the difference $I_S - I_M$, in order to obtain the binary mask C;
  - line Hough transform in order to characterize the straight lines in C (the candidate scratches);
  - creation of the scratch mask B^3 for each sequence frame, using the results obtained in the previous phase on subsampled images I_S.

In [4] this process is followed by a Bayesian refinement strategy which allows to select, among candidate scratches, those that can be best modelled by a horizontal scratch section model. As for Detection algorithm 1, we avoided to implement this part of the procedure.

B. Removal algorithms

This section explores the scratch removal problem, that is, given the corrupted image sequence I and the estimated scratch mask B, construct an estimate R of the ideal uncorrupted image I^* in (1). Two methods are adopted and compared: one belongs to the class of interpolating methods [20] and the other one to the class of inpainting algorithms [31]. Both algorithms attempt to reconstruct not only the structure of the image in the scratch domain (pixels where mask B gets value 1), but also its texture.

1) Removal algorithm 1: The first scratch removal algorithm taken into account is based on a non-parametric Markovian model adopted in [31] for texture synthesis, and adapted to our purposes. The probabilistic model is based on the assumption of spatial locality: the probability distribution for one pixel given the values of its neighborhood is independent of the rest of the image. The model is non-parametric in the sense that the probability function is not imposed or constructed explicitly, but it is approximated by a reference sample image, which must be large enough to capture the stationarity of the texture.

For each pixel p to be reconstructed, the algorithm proceeds as follows:

- determination of the sample image C, chosen as the square window of dimension A centered in p;
- construction of the set Q of pixels in C having a neighborhood *similar* to that of p. The similarity of two neighborhoods is measured according to the normalized sum of squared differences and it is weighted by a two-dimensional Gaussian of dimension G, in order to give more importance to the pixels that are near the center of the window than to those at the edge;
- reconstruction of pixel p, obtained assigning it the intensity value of a pixel randomly drawn from the set Q.

In order to establish the reconstruction order, for each scratch pixel detected in the binary mask B the number of its valid neighbors is enumerated; pixels are then replaced starting from the ones having the most valid neighbors. Scratches are thus simultaneously and progressively filled from the edges to the center of the scratch.

2) Removal algorithm 2: In [20] a simple interpolating method is adopted and the interpolation result is corrected by adding to it the estimated displacement between the adopted model and the real model. Specifically, provided the scratch mask B, the procedure consists of:

- interpolation of the pixels pertaining to the scratch domain;
- estimate of the image texture in the scratch neighborhood, obtained by computing the displacement between the least square fitting over an uncorrupted neighborhood of the scratch and the same neighborhood pixels;
- addition of the estimated texture to the pixels belonging to the scratch domain.

Different versions of the above described method can be obtained by adopting different interpolation methods and neighborhood sizes.

C. Fusion strategies

With *fusion techniques* input images which provide alternative and complementary "views" and "characteristics" of a given area are "fused" into a single image (see for instance [1], [2]). Fusion techniques should ensure that all the important visual information found in input images is transferred into the fused output image, without the introduction of artifact distortion effects.

Concerning digital scratch restoration, fusion techniques may be applied both to the detection stage and the removal stage of the automatic restoration process. Strategies used for the implementation of LSR algorithm are briefly described in the following.

1) Detection fusion strategies: The main goal of using more than one detection visual module is to make up for deficiencies in the individual modules, while retaining their features, thus achieving a better overall detection result than each single module could provide. In this case the combination should be made among scratch masks B^j produced by the detection modules, $j = 1, \ldots, d$. Here, two different combining methods or aggregation operators are adopted; supposing, for simplicity, that damaged images are affected by just one scratch, their result B^C is given by:

- *Union aggregation operator*: $B^C = \cup\{B^j : j = 1, \ldots, d\}$ such that

$$B^C(x, y) = \max_j\{B^j(x, y)\}, \ \forall(x, y);$$

- *Maximum Covering (MC) aggregation operator*: $B^C = MC\{B^j : j = 1, \dots, d\}$ such that, for all $y = 0, \dots, N-1$ (N = number of image rows),

$$B^C(x,y) = \begin{cases} 1 & \text{if} x \in [x^{mean} - W, x^{mean} + W] \\ 0 & \text{otherwise} \end{cases},$$

where

$$W = \max\{|x^{mean} - x^{min}|, |x^{mean} - x^{max}|\},$$
$$x^{mean} = \text{mean}(X), x^{min} = \min(X), x^{max} = \max(X),$$
$$X = \{x : \cap_j B^j(x,y) = \min_j\{B^j(x,y)\} = 1, \forall y\}.$$

2) Removal fusion strategies: The problem here can be stated as follows: given r images representing heterogeneous data on the observed phenomenon, take a decision D_i on an element (x,y) where D_i belongs to a decision space D. In image fusion the information relating (x,y) to each possible decision D_i is represented as a number M_i^j, where j indexes the decision making module having different properties and different meanings depending on the mathematical framework.

Given images R^j obtained by removal module j, $j = 1, \dots, r$, if we assume that $M_i^j(x,y) = R^j(x,y)$, with (x,y) in the scratch domain, represents the probability degree to which the pixel (x,y) could be seen as "restored" (i indexes the values of this appearance), we can claim all the advantages of the Bayesian framework relying in the variety of combination operators. Here we adopt the averaging aggregation operator, known in the Bayesian framework as the Basic Ensemble Method [61] (see also [62], [63]) for combining different classification modules, which has been demonstrated to significantly improve the classification performance of each single module: $R^C = BEM\left\{R^j : j = 1, \dots, r\right\}$, such that

$$R^C(x,y) = \frac{1}{r}\sum_{j=1}^{r} R^j(x,y) \ \forall (x,y).$$

We are currently investigating the possibility of using more elaborate operators, such as the Ordered Weighted Aggregation (OWA) operators due to Yager [64] or the Non-Liner Generalized Ensemble Method as introduced in [65].

IV. P-LSR: Parallel Line Scratch Restoration algorithm

The dependency graph of the various modules that constitute the LSR algorithm can be described through the Image processing Application Task Graph (IATG) shown in Fig. 2, since detection (removal) modules are completely independent of each other, while the combining module for detection (removal) can be used only after the detection (removal) modules have been executed.

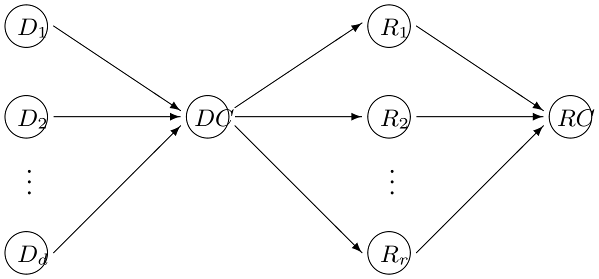


Fig. 2. IATG for the LSR algorithm.

We recall that an IATG can be modelled by a Macro Dataflow communication Graph (MDG) [66], which is a directed acyclic graph where each node stands for a visual module and each edge stands for a precedence constraint between two adjacent operators. The given description of the LSR algorithm through the IATG of Fig. 2 easily allows to individuate a parallelization strategy based on task partitioning.

Concerning the individuation of tasks to be partitioned, we could think to consider single detection/removal/fusion modules; however, they can have very different computational complexities (for example, in our implementation Removal algorithm 1 takes more than 50% of total restoration time), and this task partitioning would result in load unbalance of the compound algorithm. We therefore propose to consider as tasks the two detection and removal tasks, intended as the union of the first and of the second couple of graph columns, respectively. Each process is assigned either one of the two tasks. This simple scheme results in a better balanced scheme and, as it will be remarked in the following, well adapts to *grid* environments.

Due to the small number of tasks and the huge amount of input data (image sequences), in order to enhance the scalability of the parallel algorithm, we coupled the described task partitioning strategy with a data partitioning strategy. Such strategy consists in subdividing the input image sequence into disjoint subsets of images, and assigning each subset to a couple of processes devoted to the detection/removal tasks. Setting:

- $I_1, \ldots, I_K$ the sequence of K input images, each of size $M \times N$ with a scratch of width w;
- P the number of available processes, with $P = 2 * Q$ (P even);

the input image sequence is partitioned into Q disjoint subsets, each one consisting of K/Q images. Specifically, in the considered implementation, for each $j = 1, \ldots, Q$, the j-th subset A_j consists of the images:

$$A_j = \left\{ I_j, I_{j+Q}, \ldots, I_{J+\left(\frac{K}{Q}-1\right)Q} \right\} = \{I_{j+iQ}\}_{i=0,\ldots,\frac{K}{Q}-1} \, .$$

Each subset is assigned to a processes pair devoted to detection/removal tasks; specifically, in the considered implementation, subset A_j is assigned to the two adjacent processes P_{2j-2} and $P_{2j-1}, j = 1, \ldots, Q$, and for each image belonging to subset A_j:

- process P_{2j-2} performs the detection task and sends the result to process P_{2j-1};
- process P_{2j-1} receives the scratch mask from process P_{2j-2} and performs the removal task.

Communication operations among processes, which are all one-to-one, have been implemented through blocking sends and non-blocking receives, in order to overlap sends and receives and to hide latency times. In particular, since the scratch width w is usually small (at most 20 pixels) and much less than the image width M, in order to reduce communication times *ad hoc* high level communication routines for sending/receiving arrays of length $w \times N$ containing exclusively the rectangular region of the scratch mask including interesting pixels have been devised.

A. Performance model

We have devised a model for the execution time of P-LSR based on the assumption that the computational environment is homogeneous in terms of computation nodes and interconnection network. Setting:

- P number of available processors, with P even;
- $Q = P/2$;
- K number of images in the sequence;
- N number of rows of each image;
- M number of columns of each image;
- w scratch width in each image;
- t_{comp} execution time of one operation on one processor;
- t_{comm} time for communication of one item between adjacent processors;
- t_{lat} latency time related to a communication between adjacent processors;

the execution time of a processor can be modelled as a function of the computational complexity of its task and of the time for the communication needed. The computational complexities T_D and T_R of the detection and removal tasks are given by the sum of the computational complexities of their underlying algorithms; here we do not give explicit expressions for T_D and T_R for our specific choices of underlying restoration modules in order to avoid too many useless implementation details. An estimate of communication times can be obtained counting the number of sent/received data for each processor and adding latency times related to one communication (latency times of subsequent sends are hidden by non-blocking receives). Therefore, execution time $T_j(K,P)$ of the j-th processor for the solution of a problem of size K using $P = 2Q$ processors is given by:

$$T_j(K,P) = \left[mod(j+1,2) \times \frac{K}{Q} \times T_D + \ mod(j,2) \times \frac{K}{Q} \times T_R \right] t_{comp} + \left[\frac{K}{Q} \times w \times N \right] t_{comm} + t_{lat}, \tag{2}$$

and total execution time $T(K,P)$ for the solution of a problem of size K using P processors is given by:

$$T(K,P) = \max_{j=0,\ldots,P-1} T_j(K,P). \tag{3}$$

We explicitly observe that the performance model (2)-(3) is sufficiently general, being related to the proposed methodology and parallelization strategy and not only to the specific implementation presented here. Therefore it can be applied whichever is the number and the type of restoration algorithms taken into account for the compound algorithm.

V. Experimental results

A. Test data

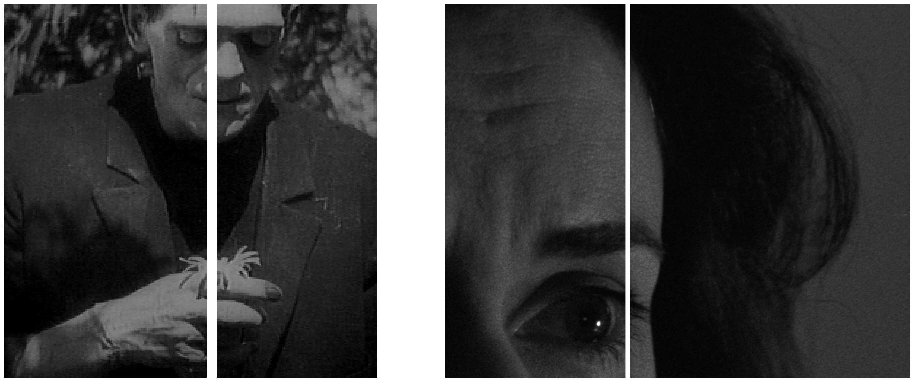
Detection and removal algorithms here presented have been tested on several real images; one example is given in Fig. 4(a), obtained from an image belonging to the sequence *Knight* available in [67], affected by a scratch of width $w = 5$.

Moreover, the considered algorithms have been tested also on artificially corrupted images. Specifically we considered $J = 20$ sequences of $K = 120$ uncorrupted original B/W images $I_k^j, j = 1,\ldots,J; k = 1,\ldots,K$, each of size $M \times N$, and the corresponding images with an artificial scratch of odd width w, denoted as $I_k^{j,w}, j = 1,\ldots,J; k = 1,\ldots,K; w = 3,5,\ldots,19$, obtained as:

$$I_k^{j,w}(x,y) = \begin{cases} 255 & \text{if}(x,y) \in \Omega_w - \partial\Omega_w \\ 200 & \text{if}(x,y) \in \partial\Omega_w \\ I_k^j(x,y) & \text{otherwise} \end{cases}, \tag{4}$$

where Ω_w denotes the scratch domain, that is the rectangular subset of the image domain of size $w \times N$ having as first column the center column $M/2$ of the image: $\Omega_w = \Big\{(x,y) : x = \frac{M}{2},\ldots,\frac{M}{2} + w - 1;$ $y = 0,\ldots,N-1\Big\}$, and $\partial\Omega_w$ denotes its border.

Examples of such images with artificial scratch of width $w = 7$ are reported in Fig. 3(a) ($M \times N = 256 \times 256$), obtained from an image belonging to the sequence *Frank* available in [67] and in Fig. 3(b) ($M \times N = 720 \times 576$), obtained from an image belonging to the movie *Animali che attraversano la strada* (2000).



(a) (b)

Fig. 3. Examples of images described in (4) with artificial scratch of width $w = 7$: (a) $I_1^{1,7}$ of size $M \times N = 256 \times 256$; (b) $I_1^{2,7}$ of size $M \times N = 720 \times 576$.

B. Computational environment

For the performance evaluation of P-LSR we used a Beowulf system consisting of 20 PC's connected through a Fast Ethernet switch; each node is a Pentium IV, 1.5GHz, 512MB RAM memory and 40GB hard disk. The algorithm has been implemented in C programming language, using MPI (version 1.2.5) communication routines [68].

C. LSR qualitative results

1) Detection results: The accuracy of the result of the detection algorithms taken into account is quite high, as it is shown by the scratch masks reported in Figs. 4(b-d) for the sequence frame of Fig. 4(a). Nonetheless, the aggregated masks reported in Figs. 4(e-f) seem more appropriate for the successive removal phase. Comparing the two aggregated masks, the union aggregation operator, whose results are reported in Fig. 4(e), seems more appropriate for the successive removal phase, compared with the MC aggregation operator (see Fig. 4(f)). Specifically, the union aggregation operator for scratch masks allows to consider all details captured by the different detection algorithms, while retaining the minimum support

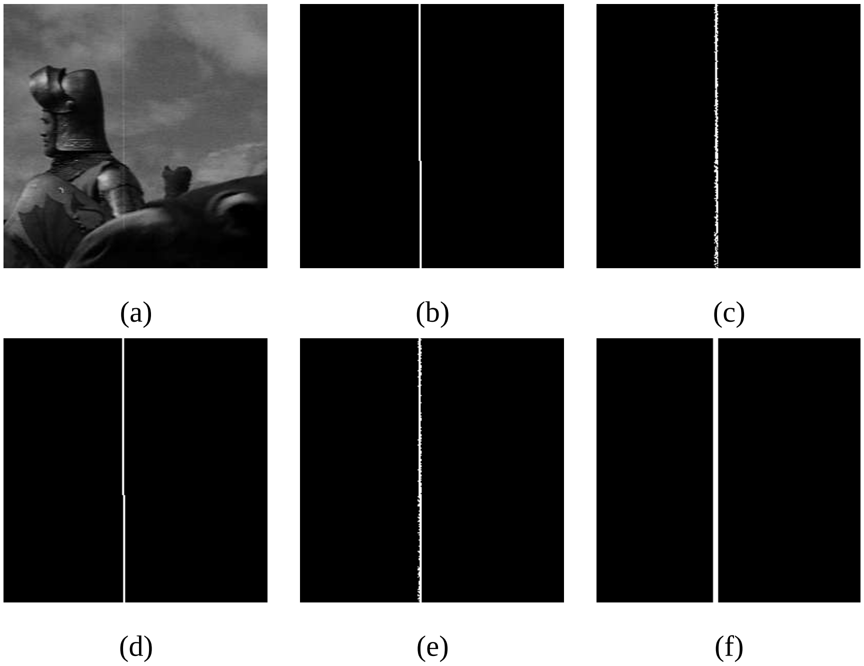


Fig. 4. Detection results over the original frame (a) by adopting: (b) algorithm 1, (c) algorithm 2, (d) algorithm 3, (e) the union fused image and (f) the MC fused image.

of the mask; the MC aggregation operator, instead, leads to a mask that appears unnatural for real images (being perfectly rectangular) but that allows to avoid several false positives detected by the algorithms taken into account.

In order to obtain an objective estimate of detection algorithms, for each mask $B_{j,k}$ computed with anyone of the described detection algorithms for the artificially scratched image $I_k^{j,w}$ described in (4) we count:

- $C_k^{j,w} = card\{(x,y) : (x,y) \in \Omega_w, B_{j,k}(x,y) = 1\}$, number of *correct detections* (pixels of the scratch that are included in the computed scratch mask);
- $F_k^{j,w} = card\{(x,y) : \ (x,y) \notin \Omega_w, B_{j,k}(x,y) = 1\}$, number of *false alarms* (pixels not belonging to the scratch that are included in the computed scratch mask),

and their mean values C^w and F^w over the $J * K$ images of (4):

$$C^w = \frac{1}{J * K} \sum_{j=1}^{J} \sum_{k=1}^{K} C_k^{j,w}; \qquad F^w = \frac{1}{J * K} \sum_{j=1}^{J} \sum_{k=1}^{K} F_k^{j,w}.$$

Given the scratch width w, the measures adopted for the objective estimation of the detection algorithms are:

- *correct detection rate* $r_C = \frac{C^w}{w \times N}$, $w \times N$ being the number of corrupted pixels (i.e. the dimension of the set Ω_w). Such measure gives values in [0,1]; the higher the value of r_C, the better the detection result;
- *false alarm rate* $r_F = \frac{F^w}{M \times N - w \times N}$. Such measure gives values in [0,1]; the lower the value of r_F, the better the detection result.

Values for r_C and r_F obtained with all the described detection algorithms are reported in Tables I and II, varying the scratch width w =3, 5, 7, and 9. Here we can observe that r_C values are generally very close to 1 for all detection algorithms and that only few false alarms are generated. Specifically, we observe that the union fusion strategy reaches the best r_C and the worst r_F values achieved by the underlying detection algorithms; the MC fusion strategy reaches the best r_C values attainable ($r_C = 1$), but r_F values worse than any other algorithm.

TABLE I

CORRECT DETECTION RATES r_C FOR THE DETECTION ALGORITHMS APPLIED TO IMAGES OF (4) VARYING THE SCRATCH WIDTH w.

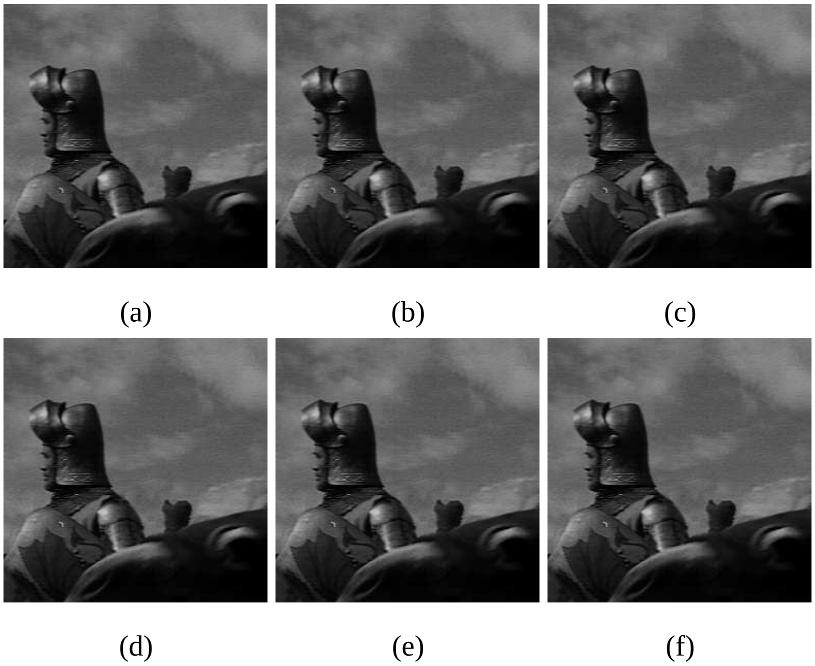
	$w=3$	$w=5$	$w=7$	$w=9$
Detection alg. 1	0.9984	0.9991	0.9991	0.9991
Detection alg. 2	0.9684	0.9565	0.9697	0.9765
Detection alg. 3	0.9984	0.9991	0.9996	0.9995
Union fusion aggr. op.	0.9984	0.9991	0.9996	0.9995
MC fusion aggr. op.	1.0000	1.0000	1.0000	1.0000

TABLE II

FALSE ALARM RATES r_F FOR THE DETECTION ALGORITHMS APPLIED TO IMAGES OF (4) VARYING THE SCRATCH WIDTH w.

	$w=3$	$w=5$	$w=7$	$w=9$
Detection alg. 1	0.0000	0.0000	0.0000	0.0000
Detection alg. 2	0.0006	0.0015	0.0017	0.0017
Detection alg. 3	0.0000	0.0000	0.0000	0.0000
Union fusion aggr. op.	0.0006	0.0015	0.0017	0.0017
MC fusion aggr. op.	0.0012	0.0024	0.0024	0.0028

2) Removal results: The results of the removal algorithm for the sequence frame of Fig. 4(a) are reported in Figs. 5(a) and 5(b) for the inpainting and interpolative algorithms respectively based on the mask produced by the union aggregation operator; Figs. 5(c) and 5(d) show the results over the mask produced by the MC aggregation operator. Fig. 5(e) and 5(f) report the results obtained by adopting the BEM method over the images depicted in Fig. 5(a)-(b) and 5(c)-(d) respectively. From these results we can observe that, even though the removal algorithms taken into account perform quite well, their reconstruction accuracy can be enhanced; the aggregated results, instead, tend to smooth the inaccuracies, still retaining the good performance of the considered algorithms.


(a) (b) (c)

(d) (e) (f)

Fig. 5. Removal results for the image of Fig. 4(a) by adopting: (a) algorithm 1 and (b) algorithm 2 over the union mask, (c) algorithm 1 and (d) algorithm 2 over the MC mask. In (e) and (f) results obtained by adopting the BEM method over the images (a)-(b) and (c)-(d) respectively.

In order to obtain objective measures of the removal algorithms accuracy, we tested them on the artificially scratched images $I_k^{j,w}$ described in (4). Given the scratch width w, let be, for $j = 1, \ldots, J$ and $k = 1, \ldots, K$:

- $\mathbf{o}_k^j$ the subimage of original image I_k^j containing only pixels in Ω_w,
- $\mathbf{r}_k^j$ the subimage of the restored image R_k^j obtained with anyone of the considered removal algorithms, containing only pixels in Ω_w.

We consider the following objective measures:

- $MeanMSE = \dfrac{1}{J * K} \sum_{j=1}^{J} \sum_{k=1}^{K} \dfrac{1}{w * N} \left\| \mathbf{o}_k^j - \mathbf{r}_k^j \right\|^2$, mean, over the $J * K$ restored images, of the

Mean Square Error (MSE) between the original and the restored images ($\|\cdot\|$ is intended as vector norm). Such measure gives a nonnegative value; the smaller the value of $MeanMSE$, the better the restoration result;

- $MeanPSNR = \frac{1}{J*K}\sum_{j=1}^{J}\sum_{k=1}^{K}\left(10*log_{10}\left(\frac{255^2}{\frac{1}{w*N}\left\|\mathbf{o}_k^j - \mathbf{r}_k^j\right\|^2}\right)\right)$, mean, over the $J*K$ restored images, of the Peak-to-Noise-Ratio between the original and the restored images obtained considering the MSE. Such measure gives a nonnegative value; the higher the value of $MeanPSNR$, the better the restoration result;
- $MeanSSIM = \frac{1}{J*K}\sum_{j=1}^{J}\sum_{k=1}^{K}\frac{\left(2*\mu_{\mathbf{o}_k^j}*\mu_{\mathbf{r}_k^j}+C_1\right)\left(2*\sigma_{\mathbf{o}_k^j\mathbf{r}_k^j}+C_2\right)}{\left(\mu^2_{\mathbf{o}_k^j}+\mu^2_{\mathbf{r}_k^j}+C_1\right)\left(\sigma^2_{\mathbf{o}_k^j}+\sigma^2_{\mathbf{r}_k^j}+C_2\right)}$,
mean, over the $J*K$ restored images, of the Structural Similarity Index [69] applied to the original and the restored images, where $C_1 = (K_1 * L)^2, C_2 = (K_2 * L)^2, K_1 = 0.01, K_2 = 0.03$, and $L = 255$. Such measure gives values in [0,1]; the higher the value of $MeanSSIM$, the better the restoration result.

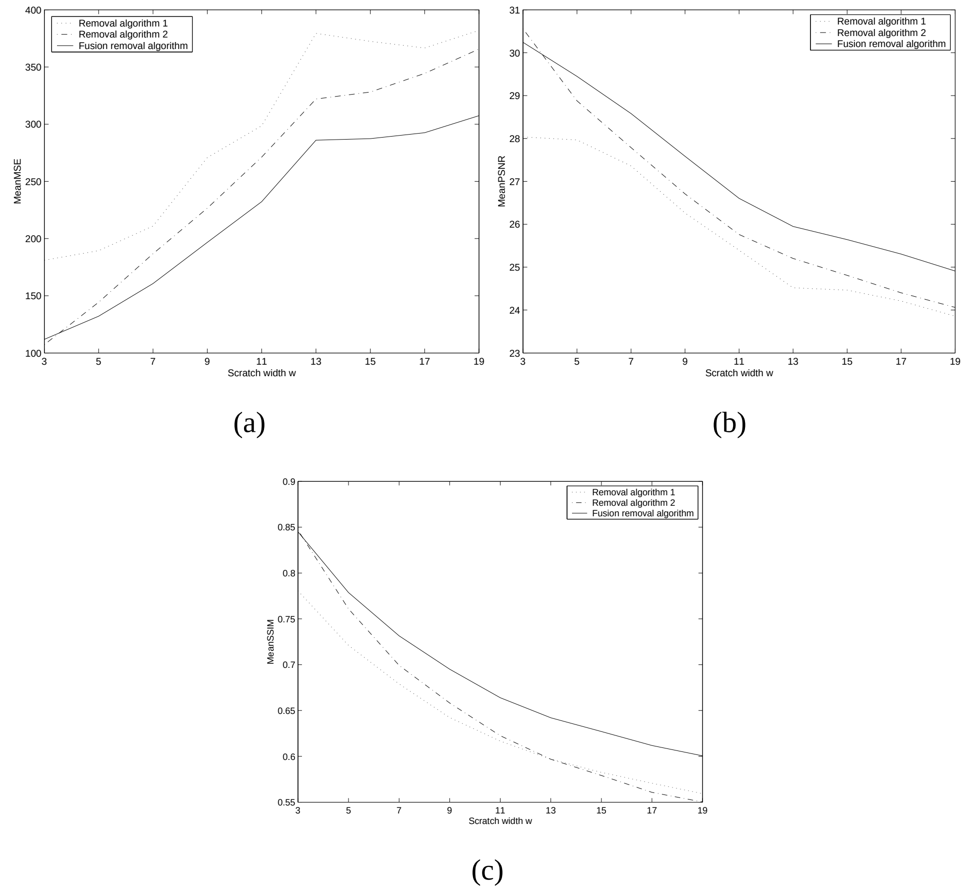
Results in terms of the described measures obtained with the removal algorithms using the union aggregated masks varying the scratch width w are reported in Fig. 6. It can be observed that $MeanMSE$ values obtained with the fusion removal algorithm are always lower than those obtained with the two removal algorithms and that $MeanPSNR$ and $MeanSSIM$ values obtained with the fusion removal algorithm are always higher than those obtained with the two removal algorithms. Moreover, for each removal method, results obtained with all the considered measures show lower accuracy increasing the scratch width, in accordance with the increasing reconstruction difficulty as the reconstruction area widens.

In summary, we can state that the considered measures indicate the fusion method as the most accurate among the considered removal methods.

D. P-LSR performance results

For validating the performance model described in §IV-A we compared experimental execution times with corresponding estimates, where:

- experimental execution times have been computed as the maximum of execution times on each processor of P-LSR executed on P processors in the considered computational environment; they do not include input and output of image sequences;
- estimated execution times have been obtained from the performance model (2)-(3) using the values:
  - $t_{comp} = 1.4 \times 10^{-9}$ secs (time for the execution of one integer operation in the considered computational environment);



(a)

(b)

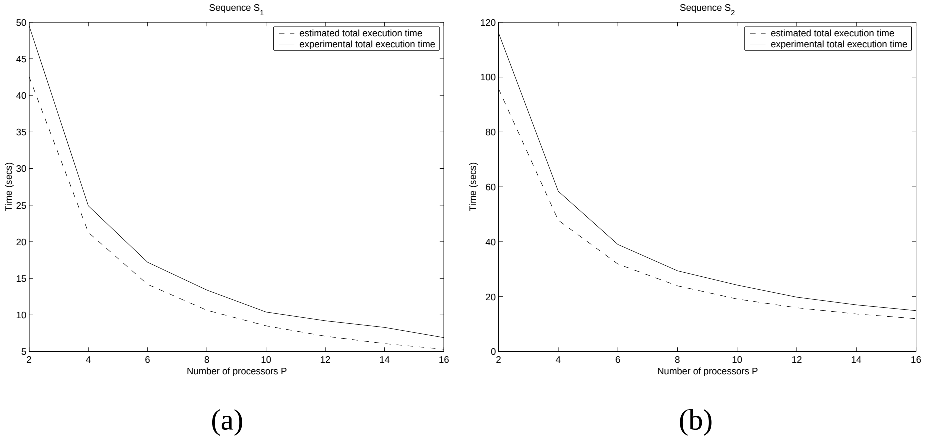
(c)

Fig. 6. Accuracy measures of the removal algorithms applied to images described in (4): (a) *MeanMSE*, (b) *MeanPSNR*, (c) *MeanSSIM*.

- $w \times N \times t_{comm} = 5.78 \times 10^{-4}$ secs for $w = 7$ and $N = 576$, or $w \times N \times t_{comm} = 3.98 \times 10^{-4}$ secs for $w = 7$ and $N = 256$ ($w \times N \times t_{comm}$ is the time for sending a message of length $w \times N$);
- $t_{lat} = 8.0 \times 10^{-5}$ secs (communication latency time, computed as the time for sending a message of zero length).

Communication time t_{comm} and latency time t_{lat} related to one communication between two adjacent processors have been computed using `mpptest` program, which measures the performance of some basic MPI routines [70].

In Fig. 7 we report estimated and experimental execution times of P-LSR, varying the number of processors P, for two of the image sequences described in (4) and reported in Fig. 3: $S_1 = \{I_k^{1,7}\}_{k=1,\ldots,120}$



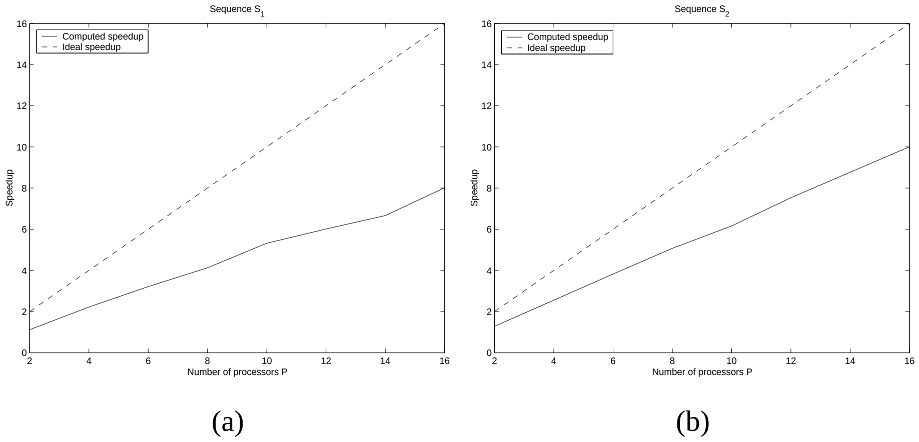
(a)

(b)

Fig. 7. Comparison of estimated and experimental execution times (secs) of P-LSR in image sequence: (a) S_1; (b) S_2.

(images of size $M \times N = 256 \times 256$) and $S_2 = \{I_k^{2,7}\}_{k=1,\ldots,120}$ (images of size $M \times N = 720 \times 576$). Here we can observe that the proposed performance model is quite accurate.

In Fig. 8 we report experimental and ideal speedups of P-LSR for image sequences S_1 and S_2, varying the number of processors P. The figure highlights that acceptable speedup is achieved by P-LSR, and it gets better when the dimension of the problem grows.



(a)

(b)

Fig. 8. Comparison of experimental and ideal speedups of P-LSR for image sequences: (a) S_1; (b) S_2.

VI. Ongoing researches

The new methodology and the parallelization strategy adopted for designing P-LSR are well suited to grid environments, where several geographically distributed resources can be accessed and exploited for the solution of the problem under investigation. The main advantages in using grids are in an extremely high computing power, in a better use of idle resources, in a shared remote access of special purpose resources or data sources, in the support of collaborative work via a virtual shared space [3].

In the case of our image sequence restoration application, the geographically distributed resources we refer to are not only computational resources (such as high-performance architectures), but also data resources (data repositories where the huge amount of data necessary to store digitized image sequences can be accessed), and restoration algorithms (to be taken into account by the approach). Moreover, for digital image sequence restoration two considerations are in order:

- typically each grid node cannot execute all the involved procedures, but we can suppose it can execute either detection or removal of scratches;
- for copyright restrictions, data (image sequences to be restored) should not be distributed on the grid nodes; the data distribution could be made according to our data parallelization strategy or according to secret sharing schemes [71], since in both cases we divide data D into n pieces such that D could be easily reconstructed from any k pieces, but even complete knowledge of $k-1$ pieces reveals no information about D.

From the above considerations it is clear that the task parallelism combined with the data parallelism approach as described, could result in an appropriate approach to grid computational environments. The simplest architectural scheme can be seen as a two layered scheme, where a *grid* node, that holds the data, works as a master node, belonging to the first layer and the other nodes (the slaves) to the second layer. During the scratch detection phase, the master distributes the data to various slaves such that node i works on frames numbered by $i, i+P, \ldots$, where P is the number of *grid* nodes now. Each slave node operates the appropriate detection and maintains the data results in the secondary memory. As soon as a slave has completed the detection phase over a frame t, it sends the same to another *grid* node involving removal, while it signals its availability to the master that then sends another data set. This greedy approach that randomly partitions the sequence frames and assigns them to the first available slaves achieves in practice a good balance, although it should be noted that this scheme in no way guarantees load balancing in the scratch restoration, since each *grid* node could be loaded in different manner, as for instance in our case those deputed to the detection and those to the removal.

The implementation of the adopted strategy and the assessment of the performance model in a grid environment is under development.

VII. Conclusions

This paper described a new methodology for the solution of classes of problems to be dealt with in digital movie restoration, which takes into account already existing promising algorithms and combines the obtained results through data fusion techniques in order to provide a restored sequence as similar as possible to the original uncorrupted sequence.

For the specific case of line scratch restoration, we presented the LSR sequential algorithm, based on such methodology, and described its implementation using suitable restoration modules.

Moreover, we described P-LSR, a parallel version of LSR algorithm based on a combination of task and data partitioning strategies, and defined a theoretical performance model of the algorithm.

The implementations of LSR and P-LSR algorithms have been tested on several corrupted and artificially corrupted real image sequences, in order to analyze the accuracy of restoration results, the performance model and the attainable speedup. The analysis lead to the conclusion that the LSR algorithm outperforms the underlying restoration methods in terms of accuracy, the performance model for P-LSR algorithm is quite faithful, and acceptable speedups can be achieved using P-LSR.

Finally, we presented some considerations concerning a distributed version of P-LSR algorithm in the context of *grid* computational environments, which is the object of further researches under development.

References

[1] I. Bloch, “Information combination operators for data fusion: a comparative review with classification,” *IEEE Trans. Systems, Man, Cybernetics A,* vol. 26, no. 1, pp.52-67, 1996.

[2] T.K. Ho, J.J. Hull, and S.N. Srihari, “Decision combination in multiple classifier systems,” *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 18, pp. 66-75, 1994.

[3] I. Foster and C. Kesselman (Eds.), *The Grid: blueprint for a new computing infrastracture*. Los Altos, CA: Morgan Kauffmann Publisher, 1998.

[4] A.C. Kokaram, R. Morris, W. Fitzgerald, and P. Rayner, “Detection of missing data in image sequences,” *IEEE Trans. on Image Processing* Part I-II, pp. 1496-1519, 1995.

[5] E. Decenciere Ferrandiere, “Restauration automatique de films anciens,” *PhD Thesis*, Ecole Nationale Superieure des Mines de Paris, Nov. 1997.

[6] L. Joyeux, O. Buisson, B. Besserer, and S. Boukir, “Detection and removal of line scratches in motion picture films,” in *Proc. IEEE Int. Conf. On Computer Vision and Pattern Recognition (CVPR’99)*, 1999, pp. 548-553.

[7] L. Joyeux, “Reconstruction de sequences d’images haute resolution. Application a la restauration de films cinematographiques,” *PhD Thesis*, Universit̀de La Rochelle, Jan. 2000.

[8] L. Joyeux, S. Boukir, and B. Besserer, “Film line scratch removal using Kalman filtering and Bayesian restoration,” presented at the IEEE Workshop on the Application of Computer Vision (WACV’00), Palm Springs, California, Dec. 2000.

[9] L. Joyeux, S. Boukir, B. Besserer, and O. Buisson, “Reconstruction of degraded image sequences. Application to film restoration,” *Image and Vision Computing*, vol. 19, pp. 503-516, 2001.

[10] L. Joyeux, S. Boukir, and B. Besserer, “Tracking and MAP reconstruction of line scratches in degraded motion pictures”, *Machine Vision and Applications*, vol. 13, n. 3, pp. 119-128, Springer-Verlag New York, Inc., 2002.

[11] T. Saito, T. Komatsu, T. Ohuchi, and T. Seto, “Image processing for restoration of heavily-corrupted old film sequences,” in *Proc. IEEE-ICPR’00*, 2000, pp. 3017-3020.

[12] O. Kao and J. Engehausen, “Scratch removal in digitised film sequences,” in *Proc. Int. Conf. on Imaging Science, Systems, and Technology (CISST’00)*, 2000, pp. 171-179.

[13] T. Bretschneider, O. Kao, and P.J. Bones, “Removal of vertical scratches in digitised historical film sequences using wavelet decomposition”, in *Proc. Image and Vision Computing*, New Zealand, pp. 38-43, 2000.

[14] F. Isgró and D. Tegolo, “A distributed genetic algorithm for restoration of vertical line scratches”, *Parallel Computing, to be published*.

[15] R.D. Morris, “Image sequence restoration using Gibbs distributions,” *PhD Thesis*, University of Cambridge, May 1995.

[16] R.D. Morris, W.J. Fitzgerald, and A.C. Kokaram, “A sampling based approach to line scratch removal from motion picture frames,” in *Proc. IEEE Conf on Image Processing (ICIP’96)*, 1996, vol. I, pp. 801-804.

[17] A. Anzalone and A. Machì, “A method for accurate detection of linearly scratched areas in motion pictures,” in *Proc. IASTED-VIIP01*, 2001, pp. 565-570.

[18] A. Machì, F. Collura, and F. Nicotra, “Detection of irregular linear scratches in aged motion picture frames and restoration using adaptive masks,” in *Proc. IASTED Int. Conf. SIP’02*, 2002, pp. 254-259.

[19] A.C. Kokaram, “Detection and removal of line scratches in degraded motion picture sequences”, in *Proc. Signal Processing VIII*, 1996, vol. 1, pp. 5-8.

[20] L. Maddalena, “Efficient methods for scratch removal in image sequences,” in *Proc. IEEE Intern. Conf. Image Analysis and Processing*, 2001, pp. 547-552.

[21] L. Rosenthaler, A. Wittmann, A. Gunzl, and R. Gschwind, “Restoration of old movie films by digital image processing,” in *Proc. IMAGE’COM 96*, 1996, pp. 1-6.

[22] G. Sapiro, “Image inpainting,” *SIAM News*, vol. 35, no. 4, 2002.

[23] J. Verdera, V. Caselles, M. Bertalmio, and G. Sapiro, “Inpainting surface holes,” in *Proc. ICIP’03*, 2003, pp. 14-17.

[24] J. Shen, “Inpainting and the fundamental problem of image processing,” *SIAM News*, Vol. 36, n. 5, June 2003.

[25] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, “Filling-in by joint interpolation of vector fields and grey levels,” *IEEE Trans. on Image Processing*, vol. 10, pp. 1200-1211, 2001.

[26] A.C. Kokaram, “A statistical framework for picture reconstruction using AR models,” in *Proc. European Conference of Computer Vision*, Workshop on Statistical Methods for Time Varying Image Sequences, 2002, pp. 73-78.

[27] A.C. Kokaram, “Parametric texture synthesis for filling holes in pictures,” in *Proc. IEEE International Conference in Image Processing (ICIP’02)*, 2002, pp. 325-328.

[28] C. Ballester, V. Caselles, and J. Verdera, “Disocclusion by joint interpolation of vector fields and grey levels,” *SIAM Journal Multiscale Modelling and Simulation*, vol. 2, N. 1, pp. 80-123, 2003.

[29] S. Masnou and J.-M. Morel, “Level-lines based disocclusion,” in *Proc. 5th IEEE Int. Conf. on Image Process (ICIP’98)*, 1998, vol. 3, pp. 259-263.

[30] A.A. Efros and W.T. Freeman, “Image Quilting for texture synthesis and transfer,” in *Proc. 28th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH ’01)*, 2001, ACM Press, pp. 341-346.

[31] R. Bornard, E. Lecan, L. Laborelli, and J.-H. Chenot, “Missing data correction in still images and image sequences,” in *Proc. ACM Multimedia 2002*, 2002, ACM, pp. 355-361.

[32] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, “Simultaneous structure and texture image inpainting,” *Proc. CVPR 2003*, 2003, IEEE Computer Society, pp. 707-712.

[33] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” *Computer Graphics*, pp. 417-424, 2000.

[34] T.F. Chan and J. Shen, “Mathematical models for local non-texture inpaintings,” UCLA CAM Report no. 00-11, 2000.

[35] T.F. Chan and J. Shen, “Euler’s elastica and curvature based inpaintings,” UCLA CAM Report no. 01-12, 2001.

[36] T.F. Chan and J. Shen, “Non-texture inpainting by Curvature-Driven Diffusions (CDD),” *J. Visual Communication and Image Representation*, vol. 12, no. 4, pp. 436-449, 2001.

[37] S.H. Kang, T.F. Chan, and S. Soatto, “Landmark based inpainting from multiple views,” UCLA CAM Report no. 02-11, 2002.

[38] T.F. Chan and J. Shen, “On the role of the BV image model in image restoration,” UCLA CAM Report no. 02-14, 2002.

[39] M. Nitzberg, D. Mumford, and T. Shiota, *Filtering, segmentation and depth*. Lecture Notes in Computer Science, Vol. 662, Berlin: Springer-Verlag, 1993.

[40] M.M. Oliveira, B. Bowen, R. McKenna, and Y.-S. Chang, “Fast digital image inpainting,” in *Proc. Int. Conf. on Visualization, Imaging and Image Processing (VIIP 2001)*, 2001, pp. 261-266.

[41] S.T. Acton, D.P. Mukherjee, J.P. Havlicek, and A.C. Bovik, “Oriented texture completion by AM-FM reaction diffution,” *IEEE Trans. on Image Processing*, vol. 10, no. 6, pp. 885-896, 2001.

[42] J.S. De Bonet, “Multiresolution sampling procedure for analysis and synthesis of texture images,” in *Proc. 24th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH ’97)*, 1997, ACM Press/Addison-Wesley Publishing Co., pp. 361-368.

[43] J.S. De Bonet and P. Viola, “A non-parametric multi-scale statistical model for natural images,” *Adv. in Neural Info. Processing*, vol. 9, MIT Press, Dec. 1997.

[44] A.A. Efros and T.K. Leung, “Texture synthesis by non-parametric sampling,” in *Proc. IEEE Conference on Computer Vision (ICCV’99)*, 1999, pp. 1033-1038.

[45] G. Gorla, V. Interrante, and G. Sapiro, “Texture synthesis for 3D shape representation,” *IEEE Trans. on Visualization and Computer Graphics*, Vol. 9, no. 4, pp. 512-524, 2003.

[46] D.J. Heeger and J.R. Bergen, “Pyramid-based texture analysis/synthesis,” *Computer Graphics Proc.*, pp. 229-238, 1995.

[47] A.N. Hirani and T. Totsuka, “A projection based algorithm for scratch and wire removal in images,” in *Proc. 5th Sony Research Forum (SRF ’95)*, 1995, pp. 20-25.

[48] A.N. Hirani and T. Totsuka, “Combining frequency and spatial domain information for fast interactive noise removal,” in *Proc. SIGGRAPH’96*, 1996, pp. 269-276.

[49] A.N. Hirani and T. Totsuka, “Dual domain image noise removal,” in *Proc. IEEE Int. Conf. on Image Processing*, 1996, vol. I, pp. 797800.

[50] A.N. Hirani and T. Totsuka, “Dual domain noise removal: shaded, color and moving images,” in *Proc. 6th Sony Research Forum (SRF ’96)*, 1996.

[51] H. Igehy and J. Pereira, “Image replacement through texture synthesis,” in *Proc. ICIP’97*, 1997, pp. 186-189.

[52] J. Portilla and E.P. Simoncelli, “A parametric texture model based on joint statistics of complex wavelet coefficients,” *Int. J. of Computer Vision*, vol. 40, no. 1, pp. 49-71, 2000.

[53] E.P. Simoncelli and J. Portilla, “Texture characterization via joint statistics of wavelet coefficient magnitudes,” in *Proc. V Intern. Conf. on Image Processing*, 1998, IEEE, vol. 1, pp. 62-66.

[54] T. Strohmer, “Computationally attractive reconstruction of bandlimited images from irregular samples,” *IEEE Image Processing*, pp. 540-548, 1997.

[55] L.-Y. Wei and M. Levoy, “Fast texture synthesis using tree-structured vector Quantization,” in *Proc. SIGGRAPH’00*, 2000, ACM Press, pp. 479-488.

[56] S. Rane, G. Sapiro, and M. Bertalmio, “Structure and texture filling-in of missing image blocks in wireless transmission and compression applications,” *IEEE Trans. on Image Processing*, vol 12, no. 3, March 2003.

[57] A.C. Kokaram and S. Godsill, “Joint detection, interpolation, motion and parameter estimation for image sequences with missing data,” in *Proc. IEEE Int. Conf. on Image Processing*, 1997, IEEE, pp. 191-194.

[58] A.C. Kokaram, “Practical, unified, motion and missing data treatment in degraded video,” *J. of Mathematical Imaging and Vision*, vol. 20, pp. 163-177, 2004.

[59] B. Delaney, “User requirements final report,” *PrestoSpace Project Deliverable 2.1*, 2004.

[60] G. Laccetti, L. Maddalena, and A. Petrosino, “Parallel/distributed film line scratch restoration by fusion techniques,” in *Computational Science and Its Applications - ICCSA 2004*, Lagan, A. et al. (eds.), Lecture Notes in Computer Science, Vol. 3044, Springer, 2004, pp. 524-534.

[61] M.P. Perrone and L.N. Cooper, “When networks disagree: ensemble method for neural networks,” in *Artificial Neural Networks for Speech and Vision*, R.J Mammone (Ed.), New York: Chapman & Hall, 1993, pp. 126-142.

[62] G. Fumera and F. Roli, “A theoretical and experimental analysis of linear combiners for multiple classifier systems,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 942-956, 2005.

[63] F. Roli, “Linear combiners for fusion of pattern classifiers,” presented at Int. School on Neural Nets, E.R. Caianiello, 7th Course on Ensemble Methods for Learning Machines, Vietri sul Mare, Italy, 2002.

[64] R.R. Yager and J. Kacprzyk, *The Ordered Weighted Averaging operation: theory, methodology and applications*. Norwell, MA: Kluwer, 1997.

[65] M. Ceccarelli and A. Petrosino, “Multifeature adaptive classifiers for SAR image segmentation,” *Neurocomputing*, vol. 14, pp. 345-363, 1997.

[66] S.S.S. Ramaswamy and P. Banerjee, “A framework for exploiting task and data parallelism on distributed memory multicomputers,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 8, no. 11, pp. 1098-1115, 1997.

[67] A.C. Kokaram, *Motion picture restoration: digital algorithms for artefacts suppression in archived film and video*. Springer-Verlag, 1998.

[68] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI-The complete Rreference. Volume 1*, 2nd ed. The MPI Core, 1998.

[69] Z. Wang, L. Lu, A.C. Bovik, “Video quality assessment based on structural distortion measurement,” *Signal Processing: Image Communication*, vol. 19, pp. 121-132, 2004.

[70] W. Gropp and E. Lusk, “Reproducible measurements of MPI performance characteristics,” in *Proc. 6th European PVM/MPI Users’ Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, 1999, Lecture Notes In Computer Science. Springer-Verlag, vol. 1697, pp. 11–18.

[71] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, pp. 612-613, 1979.