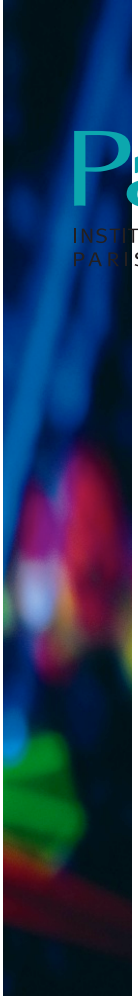


ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

EDITE - ED130

TELECOM ParisTech

2014-ENST-0006

THÈSE

pour obtenir le grade de docteur délivré par

Télécom ParisTech

Spécialité : Signal et Images

Présentée et soutenue publiquement par

Alasdair NEWSON

le 24 Mars 2014

Sur la restauration et l'édition de vidéo : détection de rayures et inpainting de scènes complexes

Jury :

M. Lionel MOISAN, Université Paris Descartes	Président
M. Anil KOKARAM, Google	Rapporteur
M. Simon MASNOU, Université de Lyon	Rapporteur
M. Guillermo SAPIRO, Duke University	Rapporteur
Mme Aurélie BUGEAU, Université de Bordeaux	Examinatrice
M. Andrés ALMANSA, CNRS, Télécom ParisTech	Directeur
M. Yann GOUSSEAU, Télécom ParisTech	Directeur
M. Patrick PEREZ, Technicolor	Directeur

THÈSE

TELECOM ParisTech
École de l'Institut Mines-Télécom - Membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

Table des matières

Table des matières 3

1 Résumé de la thèse 7
- 1.1 Détection de rayures . . . 7
  - 1.1.1 État de l'art . . . 7
  - 1.1.2 Algorithme de détection proposé . . . 8
- 1.2 Filtrage temporel de détections de rayures . . . 11
  - 1.2.1 Travaux précédents . . . 12
  - 1.2.2 Représentation spatio-temporel des rayures . . . 12
  - 1.2.3 Algorithme proposé de filtrage temporel . . . 12
  - 1.2.4 Résultats . . . 13
- 1.3 Remplissage de vidéos - "Inpainting" vidéo . . . 13
  - 1.3.1 Etat de l'art . . . 15
  - 1.3.2 Algorithme d'inpainting proposé . . . 17
  - 1.3.3 Recherche des plus proches voisins . . . 18
  - 1.3.4 Les textures . . . 20
  - 1.3.5 Inpainting vidéo avec caméra mobile . . . 21
  - 1.3.6 Résultats . . . 21
- 1.4 Conclusion . . . 22

2 Introduction 25
- 2.1 Detecting line scratches in films . . . 25
  - 2.1.1 Line scratch detection in single frames . . . 26
  - 2.1.2 Temporal filtering of line scratches . . . 26
- 2.2 Video inpainting . . . 27
- 2.3 Main contributions of the thesis . . . 28
- 2.4 Publications . . . 29
- 2.5 Websites pertaining to the work in this thesis . . . 29

I Detection of Line Scratches in Films 31

3 Spatial Line Scratch Detection 33
- 3.1 Introduction . . . 33
  - 3.1.1 A glossary of film defects . . . 33
- 3.2 Related work . . . 34
  - 3.2.1 Detecting lines in images . . . 37

3.3 Proposed Algorithm for Detecting Line Scratches in Still Frames 41
3.3.1 Pixel-wise detection criteria 41
3.3.2 First approach to scratch point grouping : the Hough transform 43
3.3.3 *A contrario* line segment detection 45
3.3.4 Maximality and exclusion 48
3.3.5 Algorithm speed-up and minimum scratch length 50
3.4 Visual results and discussion 52
3.5 Conclusion on the spatial line scratch detection algorithm 56

4 Temporal Line Scratch Detection **59**
4.1 Introduction 59
4.2 Prior work 59
4.3 Distinguishing between true scratches and false detections 61
4.3.1 Notation 62
4.4 First approach : determining generic scratch detection trajectories 62
4.4.1 Scratch motion model 63
4.4.2 First tracking approach : iterative filters 64
4.4.3 Second tracking approach : a graph-based approach 66
4.5 Proposed approach : determining the trajectories of false alarms 70
4.5.1 Global scene motion estimation 71
4.5.2 Clustering vertical trajectories 72
4.5.3 Rejection of false alarms 73
4.5.4 Further filtering criteria 73
4.6 Visual results of the temporal filtering algorithm, and discussion. 75
4.7 Conclusion and discussion 75

5 Evaluation of line scratch detection **79**
5.1 Quantitative evaluation of line scratches 79
5.1.1 Recall 81
5.1.2 Precision 81
5.1.3 $F1$-score 82
5.1.4 Algorithm parameters 82
5.1.5 Robustness to noise and texture 85
5.2 Future work 85
5.3 Conclusion 86

II Video Inpainting **87**

6 Video inpainting **89**
6.1 Introduction 89
6.2 State-of-the-art 90
6.2.1 Image Inpainting 90
6.2.2 Video inpainting 101
6.2.3 Textures in videos 105
6.3 Choosing the best approach to video inpainting 106
6.4 Inpainting energies 108
6.4.1 Link between the formulation of Arias *et al.* and Shift Maps 108

6.4.2 Optimality questions : a toy binary inpainting setting . . . 110
6.5 Proposed video inpainting algorithm . . . 116
6.5.1 Mimizing a non-local patch-based functional . . . 118
6.5.2 Approximate Nearest Neighbour search . . . 119
6.5.3 Video reconstruction . . . 121
6.5.4 Dealing with textures in videos . . . 123
6.5.5 Inpainting with mobile background . . . 127
6.5.6 Initialisation of the solution . . . 128
6.5.7 Other important algorithmic details . . . 130
6.5.8 Experimental results . . . 131
6.6 Discussion on video inpainting . . . 133
6.6.1 Justification of the use of PatchMatch . . . 135
6.6.2 Why are textures difficult to handle ? . . . 137
6.6.3 The non-local patch-based variational approach *versus* shift maps . . . 140
6.7 Conclusion and further work . . . 141
6.7.1 Further work . . . 142

7 Restoring line scratches **143**
7.1 Introduction . . . 143
7.2 Prior work . . . 143
7.3 Conclusions on prior work . . . 145
7.4 Optical Flow based restoration : a simple approach . . . 145
7.5 Visual comparison of restoration methods . . . 148
7.6 Conclusion on scratch restoration . . . 152

8 Conclusion **153**
8.1 Further work . . . 154

A Motion estimation **157**
A.1 Introduction . . . 157
A.2 Dense differential methods . . . 158
A.3 Multi-resolution estimation . . . 158
A.4 Block matching . . . 159
A.5 Parametric motion models . . . 159

B Proof of the critical occlusion size for binary inpainting convergence **163**
B.1 Critical occlusion size, without fixed border values . . . 164
B.2 Taking into account fixed border values . . . 165
B.3 Critical occlusion size, taking into account fixed border values . . . 167
B.4 Final result . . . 168

Bibliographie **169**

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisors, Andrés Almansa, Yann Gousseau and Patrick Pérez. Anyone who has worked with them, or known them for any other reason, can give you an extensive list of their numerous qualities. I will not try to improve on the other praises which have been written about them, but I will mention some things which have left an impression. Firstly, their rigour in all things professional. PhD students always have a wide variety of different concerns, but during this thesis I was never once worried about the quality of the work that we published together : if it passed my supervisors' inspection then I knew that the rest was out of our hands. For this, I am extremely grateful. I found out that the acquisition of scientific and professional rigour is a long and difficult process, and it is due to my supervisors' efforts that I have at least started on that road. Secondly, even if this has been said many times over, I must at least mention their personal qualities such as great kindness, understanding and intellectual curiosity about life in general, which made working with them a genuine pleasure. Thank you.

My thanks also to the referees and examiners of my thesis who were so enthusiastic and interested in discussing this work.

I would like to take this chance to thank Marc Jäger who first introduced me to image processing while I was at the Technische Universität Berlin during an academic exchange program. His lessons were engaging, clear and gave me (for the first time) a very clear idea of what area I would like to work in.

My thanks go to all my colleagues at Technicolor with whom I spent these past three (and a half) years. In particular, I would like to thank my co-author Matthieu Fradet who supervised me during my internship, and who absolutely always gave me his time if I needed anything. Also, a special thanks to Cedric who basically provided me with all I needed for a social life in Rennes. I would also like to thank Technicolor in general, for allowing me to pursue my research with such freedom. Parts of this work required several months of effort with very little to show for it, and the final results would not have been achieved without the possibility to make mistakes.

Thanks to my friends in Rennes and Paris who were constantly asking how everything was going. In particular, I would like to thank Quang who helped me (on a weekend) with my first published paper. Thanks also to my friends from Acigné rugby club who always gave me a perspective on life which was completely removed from any PhD-based considerations. This is an essential element in the life of any PhD student, and was invaluable for me on numerous occasions. Finally, my thanks to Benoît and Morgane who helped me on innumerable occasions for everything from moving to mending a bike.

Thank you to all my colleagues at Télécom ParisTech, a large number of which have become close friends. There are far too many to name, but I must say a special thanks to my friends at the "Bureau Cool" (Baptiste, Cecilia, Guillaume, Thierry and Yann). Much of the work in this thesis is influenced by discussions I had with them. Work aside, I discovered many wondrous things which I was not familiar with before (or sometimes, never suspected the existence of) such as Les Gladines, chez Michel, le jazz, la "Théorie Variationnelle du kouign-amann" and the Uruguayan Image Processing Mafia. My thanks for making these years full of laughter.

Finally, my thanks to Lanwenn, who I had the luck to meet at the very beginning of this journey. I do not think that I know of anyone who has a greater natural talent for being kind to others, which is a rare trait indeed, and one which helped me throughout my PhD. Without her support and presence, many of the difficult times would have been impossible and the best of times would have been less bright. Trugare braz logodennig.

Chapitre 1

Résumé de la thèse

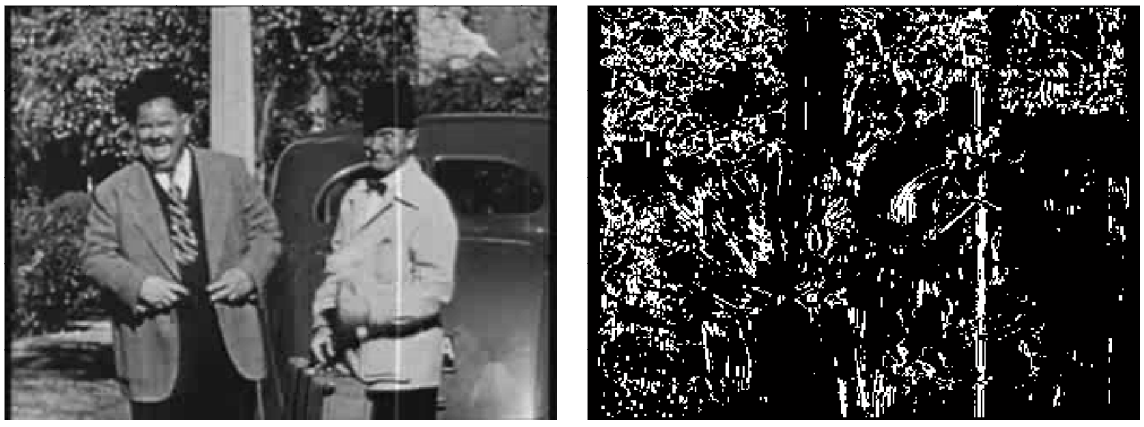
Cette thèse traite deux problèmes de la restauration des images et vidéos. Le premier est la détection de rayures dans les vieux films. Les rayures sont des défauts qui sont dûs à des frottements mécaniques d'objets avec le film physique et qui se manifestent par des lignes verticales sur la vidéo. Le deuxième problème est celui du remplissage d'une région dans une vidéo, ou "inpainting vidéo" en anglais.

1.1 Détection de rayures

Les rayures sont produites lorsqu'un objet se frotte physiquement contre le film. En termes d'apparence, les rayures se présentent comme des lignes fines, approximativement droites et verticales.

1.1.1 État de l'art

Le premier travail concernant la détection et la restauration de rayures a été fait par Kokaram [76]. Ce papier considère un profil horizontal des rayures en forme de sinus cardinal :



(a) Original frame (b) Binary detection image

FIGURE 1.1: Détection pixel-par-pixel de la séquence "Les Choses de la Vie". Les pixels blancs sont détectés, et les pixels noirs ne le sont pas.

$$L^p(x,y) = b_p k_p^{|x-(m_p y+c_p)|} \cos(\frac{3\pi|x-(m_p y+c_p)|}{2\omega_p}). \tag{1.1}$$

Cela est justifié par le fait que la diffraction de la lumière qui passe par la rayure. Kokaram utilise un pré-filtrage des candidats de rayures avec la transformée de Hough, et valide ces candidats avec une estimation Bayesienne des paramètres des rayures. Certains aspects de ce travail ont été repris par Bruni *et al*. [26].

1.1.2 Algorithme de détection proposé

Dans l'algorithme que nous avons développé pour la détection de rayures, nous voulions adresser deux problèmes essentiellement :

– Des hypothèses trop fortes sur la forme des rayuress
– Des fausses détections dues au bruit et au clutter

Nous souhaitions également avoir une détection la plus précise possible.

Critère de détection pixel par pixel

Nous utilisons deux critères de détection pixel par pixel. Le premier est inspiré par le travail de Kokaram [76].

$$c_1(x,y) : |I_g(x,y) - I_m(x,y)| \geq s_{med} \tag{1.2}$$

$$c_2(x,y) : |I_l(x,y) - I_r(x,y)| \leq s_{avg}. \tag{1.3}$$

où, s_{med} et s_{avg} sont des seuils sur le niveau de gris.

Définissons la carte de détection $I_B : \mathbb{N}^2 \rightarrow \{0,1\}$ qui indique si un pixel est détecté ou non :

$$I_B(x,y) = \begin{cases} 1 & \text{si} \quad c_1(x,y) \quad \text{et} \quad c_2(x,y) \\ 0 & \text{sinon} \end{cases} \tag{1.4}$$

On peut voir un exemple de la carte de détection dans la Figure 1.1.

Nous pouvons voir que les rayures apparaissent comme des segments visuellement significatifs dans la carte de détection I_B. On voit qu'il existe également beaucoup de fausses détections dues au bruit et au grain de film. Si l'on restaurait tous les pixels indiqués par I_B, on restaurerait beaucoup d'information non-dégradée. Un être humain pourrait indiquer relativement facilement quels pixels détectés appartiennent à des segments significatifs, en revanche cette tâche est beaucoup plus compliqué à faire de manière automatique. Pour cela, nous avons plusieurs outils possibles, dont la morphologie mathématique et la transformée de Hough. Malheureusement, ces approches contiennent plusieurs paramètres qui doivent être réglés manuellement pour chaque nouvelle séquence. Un autre ensemble de méthodes appelées méthodes "a contrario" ont l'avantage d'être très robustes et surtout contiennent des paramètres qui sont fixes.

Nous nous sommes tournés vers ces méthodes pour le groupement des segments de rayures.

Algorithme de détection proposé

La méthodologie *a contrario* est une approche statistique qui permet de détecter des objets visuels dans les images. Plus précisément, un objet est détecté s'il a une faible chance d'être produit par un *modèle de fond* (ou modèle de bruit) que l'utilisateur de la méthode doit préciser.

Dans ce point de l'*a contrario* est une méthodologie plutôt qu'un algorithme : il faut spécifier le modèle de fond qui convient au problème de détection considéré.

Nous présentons d'abord l'approche dans le cas de la détection de gradients alignés dans les images, ce qui est une situation relativement proche de la détection de segments de rayures.

Groupement de segments Le but ici est de détecter des segments de gradients alignés, qui correspondent de manière générale à des contours ou à des lignes dans les images. L'approche consiste premièrement à discrétiser les directions de gradient dans l'image avec une précision de $p\pi$ radians, avec $p \in [0, 1]$. Le modèle de fond, dans ce cas, sera une image où les directions de gradient sont distribuées de manière uniforme sur $[0, \pi]$. Cela correspond à plusieurs configurations d'image où l'on ne voudrait pas détecter des lignes : un bruit blanc, une zone d'intensité homogène et certain types de texture.

Une fois ce modèle posé, l'approche consiste à tester tous les différents ségments de l'image afin de savoir si elles sont conformes à ce que le modèle de fond pourrait produire. Cette "conformité" est défini à l'aide du *nombre de fausses alarmes* (NFA) du segment. Le nombre de fausses alarmes est le nombre de fois en moyenne qu'un tel segment aurait pu être produit par notre modèle de fond.

Notre objectif est le suivant. Etant donné un segment *observé*, nous voulons calculer la probabilité qu'un segment au moins aussi remarquable aurait pu être produit par le modèle de fond. Une variable aléatoire X_i est associée avec chaque pixel dans le modèle de fond. Ces variables aléatoires sont distribuées de manière indépendante et identique selon une loi de Bernouilli, de paramètre p. Autrement dit, chaque pixel est "aligné" ou non. Définissons également $S_l = X_1 + \cdots + X_l$, le nombre de points alignés dans le modèle de bruit. Considérons maintenant un segment *observé* dans l'image, de taille l et ayant k_0 pixels alignés. La probabilité que le modèle de fond produise un segment au moins aussi remarquable que celui observé suit une loi Binomiale :

$$Pr(S_l \geq k_0) = \sum_{k=k_0}^{l} p^k (1-p)^{l-k} =: B(p; k_0, l). \tag{1.5}$$

Si cette probabilité est petite, alors le segment observé est intéressant à détecter. Sinon, on considère qu'il aurait pu être produit par le modèle de fond est qu'il n'est pas intéressant. Cependant, on ne peut pas valider le segment en considérant $Pr(S_l \geq k_0)$ seulement : il faut le multiplier par le nombre total de segments testés pour avoir le NFA. En effet, même un segment très peu probable peut être produit si l'on teste assez de segments. Nous avons donc :

$$NFA(l, k_0) = N_{tests} B(p, k_0, l), \tag{1.6}$$

où N_{tests} est le nombre de segments testés. Un segment est validé si son NFA est plus petit qu'un seuil ε qui est fixé en fonction du nombre moyen de fausses alarmes qu'on accepte lors de la détection. Un segment dont le NFA est plus petit que ε est appelé *ε-significatif*. Lorsque $\varepsilon = 1$, comme est souvent le cas dans la litérature, on dit simplement que le segment est *significatif*.

Comme dit précédemment, l'algorithme consiste à tester tous les segments dans l'image et les accepter ou non avec l'équation (1.6). Cette approche souffre de deux inconvénients majeures :

- il est assez lent (le nombre de segments à tester est très grand !) ;
- des "lignes" significatifs sont représentés par plusieurs segments de longueurs différentes (la "meilleure" représentation est contenue et contient d'autres segments significatifs).

Ces deux problèmes peuvent être adressés avec les deux propriétés supplémentaires suivantes :

- la maximalité ;

– l'exclusion ;

Un segment qui est *maximal* ne contient, ni est contenu par un autre segment plus significatif que lui. L'exclusion stipule qu'un pixel ne peut appartenir qu'à un seul segment.

Un segment qui est maximal a aussi la propriété suivant : il commence par une transition pixel non-détecté/pixel détecté, et finit de manière symétrique. Cela a pour conséquence de réduire drastiquement le nombre de segments à tester, et donc d'accélérer l'algorithme.

Utilisation de la méthodologie *a contrario* pour la détection de rayures Dans notre cas, nous voudrions utiliser la méthodologie *a contrario* afin de détecter des segments significatifs dans la carte de détection binaire I_B (voir Figure 1.1). Pour cela, nous devons d'abord déterminer un modèle de fond.

Comme pour la détection d'alignements de gradient, ce modèle sera une image où chaque pixel est associé à une variable aléatoire suivant une loi de Bernouilli, qui a une probabilité p d'être détecté. Normalement, p devrait être déterminé pour refléter la probabilité de détecter un pixel de rayure dans du bruit. Malheureusement, cette probabilité varie en fonction de la variance du bruit (ce qui n'est pas le cas pour les alignements de gradient). Ainsi, il n'est pas possbile de réfléter tous les "fonds" possibles avec un p constant.

Nous proposons alors un modèle de fond *spatialement variable*, c'est-à-dire dont la probabilité p est différent pour chaque pixel. L'idée est que cette probabilité change en fonction du fond local, qui peut être constant, bruité ou texturé.

Pour utiliser cette approche, nous devons d'abord nous occuper d'un problème technique. Effectivement, avec ce modèle de bruit spatialement variable, le NFA est modifié. Maintenant il suit une loi binomiale de Poisson :

$$\Pr(S_l \geq k_0) = \sum_{k=k_0}^{l} \sum_{\substack{x_1,\dots,x_l \\ \sum x_i = k}} \prod_{i=1}^{l} p_i^{x_i}(1-p_i)^{x_i}, \tag{1.7}$$

qui est assez lourd à calculer. Afin d'avoir un algorithme plus pratique à utiliser, nous avons recours à l'approximation de Hoeffding, qui est une borne supérieure sur la probabilité qu'une somme de variables aléatoires dévie de son espérance. Cette approximation est valide même quand les variables dans la somme ne sont pas distribuées de manière identique, ce qui est notre cas. Il doivent bien évidemment être distribuées de manière indépendantes. Avec l'approximation de Hoeffding, le NFA peut s'écrire de la manière suivante :

$$\Pr(S_l \geq k_0) = N_{tests} e^{-l\left(r \log \frac{r}{\langle p \rangle} + (1-r) \log \frac{1-r}{1-\langle p \rangle}\right)}, \tag{1.8}$$

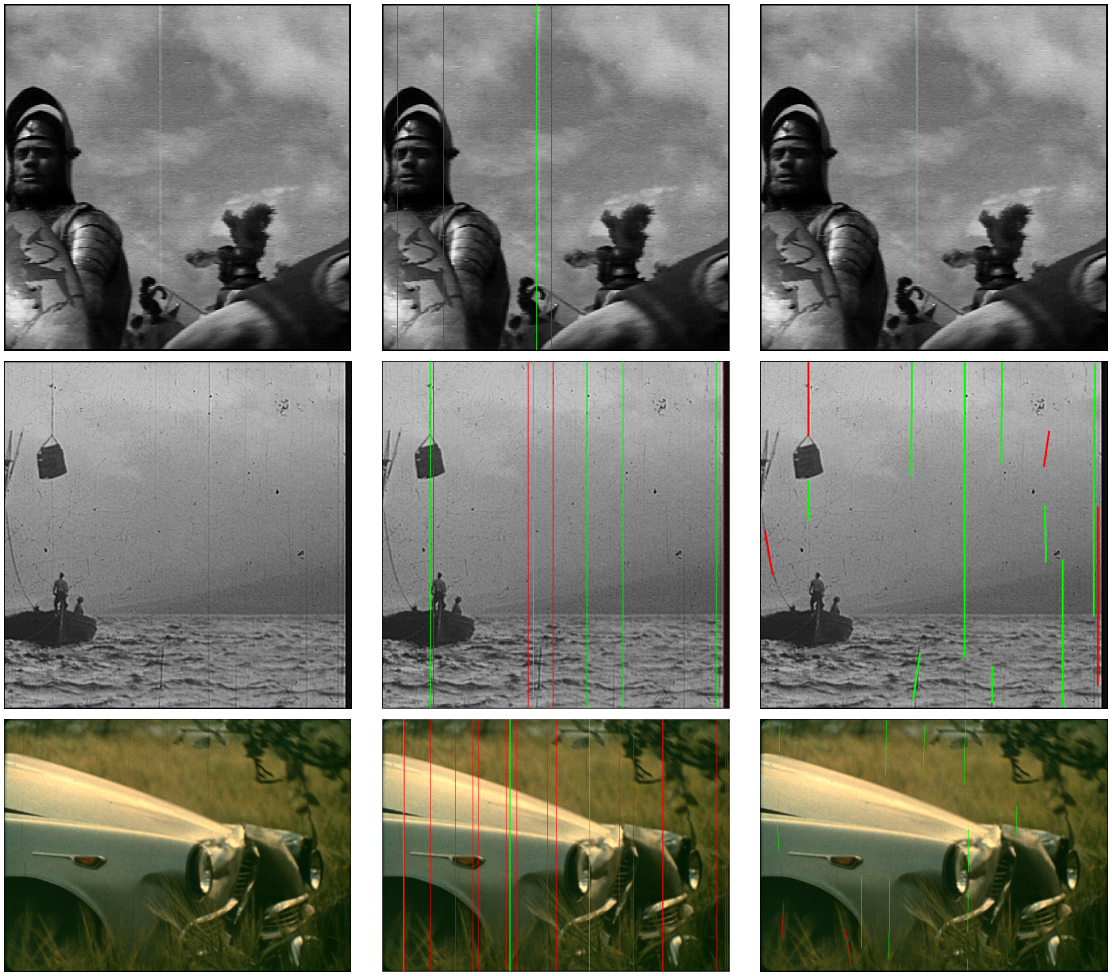
où $r = \frac{k_0}{l}$ et $\langle p \rangle = l^{-1} \sum p_i$ est la probabilité de détection moyenne sur le segment courant.

Une dernière chose que l'on doit vérifier avec cette défintion du NFA est si la maximalité retient les mêmes propriétés qui nous permettent d'accélérer tant l'algorithme. Nous vérifions cela dans la Section 3.3.4.

Résultats visuels

Nous présentons ici quelques uns des résultats de l'algorithme de détection de rayures proposé. Les résultats sont comparés visuellement avec ceux d'une méthode de Bruni *et al*. Ces résultats se trouvent dans la Figure 1.2.

On voit que les détections de la méthode de [25] sont des colonnnes entières. Cela pose un problème dans le cas des rayures qui ne couvrent qu'une partie de la hauteur de l'image, et pour



(a) Image originale (b) Détection de [25] (c) Nos résultats

FIGURE 1.2: De haut en bas : “Knight”, “Sitdown”, et “Les Choses de la Vie”. Les bonnes détections sont indiquées en vert, les fausses détections sont en rouge.

ceux qui ne sont pas verticales. Notre méthode, en revanche, fournit des détections spatiallement précises, et robustes aux bruits et aux textures appartenant à la scène.

1.2 Filtrage temporel de détections de rayures

Malgré les avantages de notre algorithme de détection spatiale, un problème est particulièrement visible en regardant les détections présentées dans la Figure 1.2 : on ne distingue pas les éléments de scène verticaux et fins des véritables rayures.

Ce problème est extrêmement difficile à résoudre dans des images fixes ; même des êtres humains peuvent avoir des difficultés pour distinguer les fausses alarmes de vraies rayures. Cependant, le comportement temporel des éléments de la scène est souvent différent de celui des rayures. En effet, quand la scène bouge, une bonne partie des fausses alarmes détectées devrait bouger dans le même sens. Dans notre travail, nous exploitons cette idée. Avant d’exposer notre approche au

problème, nous faisons un bref rappel des travaux précédents sur le même sujet.

1.2.1 Travaux précédents

Les premiers travaux qui ont analysé le comportement temporel des détections de rayures étaient ceux de Joyeux *et al*. [70, 71, 72]. Les auteurs proposent de suivre des rayures en supposant que les véritables rayures présentent un mouvement sinusoïdal. Lers auteurs supposent que les rayures sont produites lorsqu'une pièce mecanique frotte contre la pélicule, et que ces pièces auront un mouvement sinusoïdal. Ils utilisent le filtre de Kalman afin de suivre les vraies rayures, et considèrent que tout ce qui n'est pas suivi est une fausse detection.

Güllü *et al*. [74] proposent une approche très différente de la précédente. En effet, ils essaient de valider des vraies rayures en se basant sur l'erreur de suivi lors d'une estimation de mouvement par "block matching". En effet, ils considèrent que des vraies rayures n'ont rien à voir avec le fond, et ne bouge pas avec celui-ci (ce qui est globalement vrai). Il y aura donc des erreurs entre l'estimation de mouvement sur la rayure et celle juste à côté. Malheureusement, les auteurs ne traitent pas le problème de comment établir une trajectoire pour chaque objet, ce qui est la tâche la plus délicate, et donc leur méthode ne possède pas une très grande robustesse.

Müller *et al*. [95] proposent d'identifier les fausses alarmes, plutôt que les vraies détections. Comme nous allons le faire, ils supposent qu'une fausse alarme bougera de manière similaire à la scène. Malheureusement, encore une fois, les auteurs ne traitent pas le problème des trajectoires de manière adéquate. En effet, ils utilisent une estimation par block matching qui ne peut pas différencier les rayures de la scène. L'estimation aux alentours de la rayure est alors influencé par la présence de celle-ci. Ainsi, ils ne peuvent pas obtenir un ensemble de trajectoires robustes.

On voit avec cette revue rapide de l'état-de-l'art que la principale difficulté dans ce domaine est de déterminer des trajectoires qui correspondent à *un* objet cohérent (rayure ou élément de scène). Nous présentons maintenant notre approche au problème du filtrage temporel des détections de rayure.

1.2.2 Représentation spatio-temporel des rayures

Nous établissons d'abord une image binaire, que l'on appelera I_T, qui indique la détection d'une rayure. Les lignes de I_T correspondent aux indices temporels et les colonnes correspondent aux colonnes de l'image. Si $I_T(x, y) = 1$, alors il existe au moins une détection de rayure contenue dans la trame y dont l'indice moyen de colonne est égal à x. On dit "au moins un" puisqu'il peut y avoir plusieurs détections dans une image qui peuvent avoir le même indice moyen de colonne, par exemple deux détections verticales parfaitement alignés. Un exemple de cette représentation est visible dans la Figure 1.3. Dans cet exemple, les détections dues à des fausses alarmes bougent de manière cohérente, puisqu'elles suivent bien le mouvement de la scène.

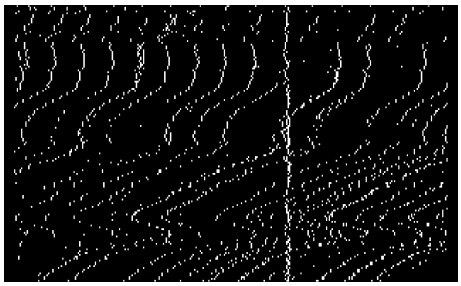
1.2.3 Algorithme proposé de filtrage temporel

Nous présentons maintenant notre approche au filtrage temporel. Celle-ci est constituée de trois étapes :

- recalage des détections par rapport à un mouvement affine ;
- extraction des trajectoires qui correspondent à une rayure ou élément de scène ;
- validation ou rejet de la trajectoire.

Le but de la première étape est de transformer les trajectoires des fausses alarmes en lignes verticales dans I_T. En effet, nous estimons un mouvement affine de manière global qui corres-

(a) Une image de la séquence "Wathenco"



(b) Profil temporel (I_T)

FIGURE 1.3: Première trame (a), et le profil temporel (b) des détections spatiales de la séqeunce "Wathenco". Chaque point blanc représente au moins une détection de rayure. L'axe des x représente l'indice de colonne moyen de la détection de rayure, et l'axe des y représente l'indice de la trame dans laquelle la détection a été faite.

pond au mouvement de la scène. Cette estimation se fait avec l'algorithme présenté dans le travail d'Odobez et Bouthemy [97]. En recalant les détections par rapport au mouvement de la scène, nous sommes en train de *stabiliser* ces fausses détections, qui apparaissent donc comme des lignes verticales.

Une fois cette transformation effectuée, nous devons extraire des trajectoires coherentes qui, nous l'espérons, correspondent à des rayures ou à des structures verticales (fausses alarmes) particulières. Comme indiqué précédemment, cette étape est assez compliquée et est rarement fait de manière très fiable dans d'autres approches [70, 74, 95].

Dans notre cas, cette étape est largement simplifié par le fait que les trajectoires correspondant à des fausses alarmes apparaissent comme des lignes verticales dans I_B. Ainsi, nous pouvons les détecter de manière robuste en utilisant l'algorithme de détection que nous avons proposé pour la détection de lignes verticales.

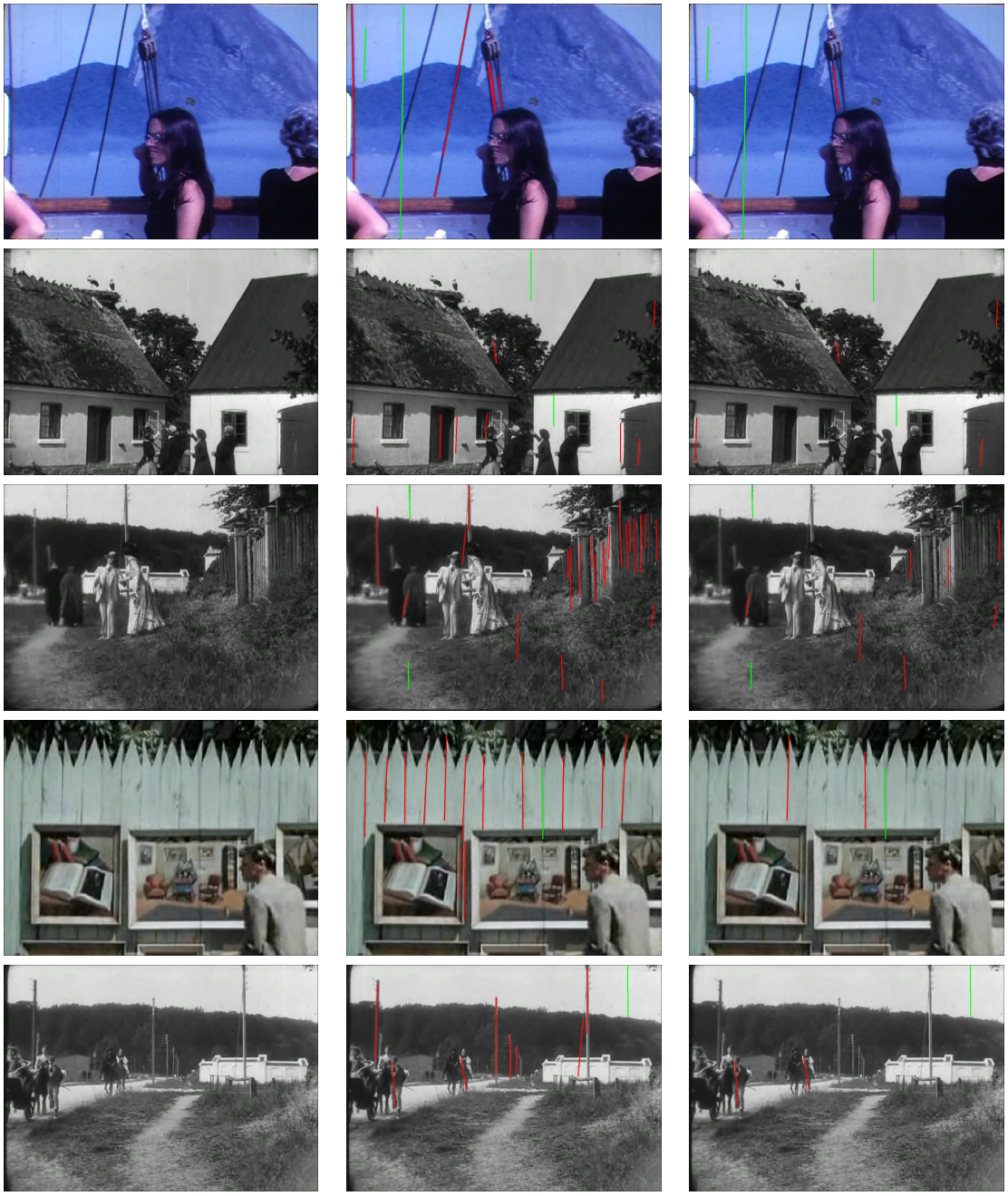
1.2.4 Résultats

Nous présentons maintenant quelques résultats de notre algorithme de filtrage temporel. Ceux-ci sont visibles dans la Figure 1.4. Nous n'utilisons que des séquences qui contiennent du mouvement, car il est impossible de distinguer une fausse alarme d'une vraie rayure sur une vidéo immobile. Nous voyons l'intérêt de notre étape : une bonne partie des fausses alarmes sont filtrées, même dans des situations très difficiles, comme les troisièmes et quatrièmes séquences.

1.3 Remplissage de vidéos - "Inpainting" vidéo

Nous présentons maintenant le travail que nous avons effectué sur le sujet de l'inpainting vidéo. Le but de l'inpainting vidéo est de reconstruire une partie inconnue d'une vidéo, et que ce remplissage paraisse cohérent avec le contenu autour de cette région. Ce problème apparaît naturellement lorsqu'un film est endommagé, ou bien si l'on veut enlever un objet ou une personne d'une vidéo.

Nous définissons d'abord quelques notations. Soit Ω le support d'une image ou vidéo, et une position $p = (x, y, t)$ dans ce support. Soit $u : \Omega \to \mathbb{R}^3$ le contenu (couleur) de cette image ou



(a) Trame originale (b) Détections spatiales (c) Détections après l'étape de filtrage temporel proposé

FIGURE 1.4: Détections de rayures dans plusieurs séquences rayées. Les détections correctes sont en vert, et les fausses alarmes sont en rouge. Nous observons que les situations avec des fausses alarmes dues à des éléments de la scène sont mieux gérées avec notre approche temporel qu'avec une détection spatiale uniquement.

vidéo. Nous notons le support de la zone occultée (à remplir) avec $\mathcal{H}$ (pour "hole" - un trou - en anglais), et $\mathcal{H} = \Omega - \mathcal{H}$ la région connue (pour "données", puisqu'on va l'utiliser pour remplir l'occultation).

1.3.1 Etat de l'art

L'inpainting a été d'abord proposé pour l'image, premièrement par Masnou et Morel [92]. Cette méthode remplit l'occultation avec des lignes de niveau. Ils formulent l'approche avec la minimisation d'une fonctionnelle d'énergie sur l'image. Le nom "inpainting" a été proposé par Bertalmìo *et al*. dans [14], inspiré par l'approche des artistes qui restaurent des œuvres d'art. Celle-ci consiste à remplir le contenu par couches concentriques, de la bordure vers l'intérieur. Les méthodes précédentes ([14, 92, 117]) sont basées sur des équations différentielles partielles (EDP), et plusieurs autres ont été également proposées [9, 36]. L'inconvéniant de toutes ces méthodes est qu'elles ne permettent pas de reconstruire des textures correctement. Ainsi, il est généralement très facile de distinguer la partie remplie du reste de l'image.

Une avancée considérable a été faite avec l'introduction de la notion de *patchs* d'image. Ces patchs sont des petits carrés ou rectangles d'image. Ceci a été fait de manière plus ou moins simultanée par plusieurs auteurs [18, 38, 47].

Définissons le voisinage de patch $\mathcal{N}_p$ comme étant un ensemble de positions dans le support d'une image ou vidéo, qui sont associés à la position p. Le *patch* W_p est défini comme le vecteur suivant :

$$W_p = (u(q_1) \cdots u(q_N))_{q_i \in \mathcal{N}_p} . \tag{1.9}$$

Le plus souvent, le voisinage de patch sera un carré ou rectangle centré sur p, mais en théorie il pourrait être quelconque.

Parmi les première méthodes d'inpainting basées sur la notion de patchs, celle de Criminisi *et al*. [38] est peut-être le plus connu. Celle-ci consiste a remplir, patch par patch, la région manquante (l'occultation). Le remplissage d'un patch W_p se fait en trouvant le patch le plus semblable dans la zone non-occultée. Le plus "semblable" signifie un patch qui minimise une certaine distance de patch $d(W_p, W_{p'})$. Le patch W_q qui minimise $d(W_p, \bullet)$ est appelé le *plus proche voisin* de W_p. Enfin, nous définissons la *carte de déplacements* ("shift map" en anglais) ϕ, qui représente le déplacement spatial (ou spatio-temporel) entre un patch W_p dans l'occultation et son plus proche voisin dans la région non-occultée. Ainsi le plus proche voisin de W_p est $W_{p+\phi(p)}$.

Bien entendu, dans ce genre d'approche qui emploie des patchs, il faut commencer à remplacer les patchs sur la bordure de $\mathcal{H}$ d'abord, où l'on connaît au moins partiellement le contenu et où la comparaison aura du sens. Evidemment, le point crucial de l'algorithme sera l'ordre dans lequel les patchs seront traités. Criminisi *et al*. font cela en associant une priorité à chaque patch, qui tend à favoriser des patchs dont la plupart de l'information est connu et aussi ceux qui contiennent des *isophotes*. Les isophotes sont des lignes sur lesquelles il existe un fort gradient. L'algorithme essaye donc de prolonger ces isophotes dans la direction normale à celle du gradient. Cela a pour but de continuer les structures, comme le font les méthodes basées sur les EDP.

Ce genre de méthode a représenté une grande avancée, puisque pour la première fois il était possible d'obtenir des résultats d'inpainting qui respectaient la texture. Cela est dû à l'idée (simple, mais extrêmement efficace) des patchs. En collant des patchs, on reproduit (partiellement au moins) cette texture. En effet, l'idée de patchs a été tout d'abord proposée dans le cadre de la synthèse des textures [49].

La prochaine innovation dans le domaine de l'inpainting était de formuler le problème en termes d'optimisation, discrete ou continue, en prenant en compte l'idée de patchs. L'idée principale de ces approches est de minimser une fonctionnelle qui est liée à la somme des distances de patchs entre les patchs occultés et ceux qui sont connus. En d'autres termes, le remplissage devrait ressembler d'une certaine manière au contenu qui se trouve autour de l'occultation.

La première méthode employant cette approche est celle de Demanet *et al*. [44]. Ils ont proposé de minimiser l'énergie suivante :

$$E(\phi|u) = \sum_{p\in\mathcal{H}} d^2(W_p, W_{p+\phi(p)}). \tag{1.10}$$

Cependant, les auteurs proposent simplement une méthode heuristique pour la minimisation, et non pas un algorithme d'optimisation à proprement parler.

Les premiers à formuler le problème d'inpainting en termes d'optimisation étaient Komodakis *et al*. [80]. Ils ont employé les champs aléatoires de Markov, et une optimisation discrète.

Pritch *et al*. [105] ont utilisé la méthode d'optimisation appelée "Graph Cuts" [20] pour l'optimisation. Ils ont proposé l'énergie discrète suivante :

$$E(\phi) = \alpha \sum_{p\in\mathcal{H}} E_d(\phi(p)) + \sum_{p\in\mathcal{H}} \sum_{q\in\mathcal{N}_p} E_s(\phi(p), \phi(q)), \tag{1.11}$$

où α est un scalaire. E_d est un terme d'attache aux données, et E_s est un terme de régularisation. On voit que la minimisation de cette énergie est faite par rapport à la carte de déplacements ϕ uniquement (et non pas par rapport à u). En effet, dans cette approche l'inpainting se fait en copiant directement la valeur du plus proche voisin de chaque pixel : $u(p) = u(p + \phi(p))$. Ainsi, la solution est un arrangement des valeurs des pixels dans u.

Le terme d'attache aux données de cette énergie stipule simplement que les pixels utilisés pour l'inpainting doivent provenir de $\mathcal{D}$. Le terme de régularisation spécifie que les déplacements doivent mener à des valeurs qui soient cohérents entre eux. En d'autres termes, l'énergie pénalise des solutions où $u(p + \phi(p))$ est très différent de $u(p + \phi(q))$, pour $q \in \mathcal{N}_p$. Evidemment, cette contrainte pourrait être respecté de manière triviale en copiant et collant un morceau entier de la partie non occultée, mais les conditions de bord empêchent cette solution triviale. Cependant, Pritch *et al*. [105] ne traite pas ce point précis dans leur papier. Enfin, Pritch *et al*. utilisent un schéma multi-échelle afin d'améliorer la qualité des résultats, en évitant les minimas locaux.

Liu et Caselles [88] ont proposé une approche similaire à celle de Pritch *et al*., utilisant les Graph Cuts. Cependant, ils traitent le problème de manière plus rigoureuse, et prennent en compte les conditions de bord de manière correcte. De plus, ils identifient un problème important lié à l'utilisation d'un schéma multi-échelle. En effet, aux échelles grossières beaucoup de détails ne sont pas visibles. En particulier, les textures n'existent pas à ces échelles, ce qui conduit à des ambiguïtés lors des comparaisons de patchs. Comme le résultat final dépend beaucoup de l'initialisation faite à la résolution grossière, il ne faut pas se tromper à ce niveau. Ainsi, les auteurs introduisent des "features" liés au gradient qui reflètent certains contenus (tels que les textures), et qui aident à résoudre les ambiguïtés.

Une autre méthode qui utilise une optimisation *continue* est l'approche variationnelle d'Arias *et al*. [5]. Cette méthode était la première à formuler le problème d'inpainting sous forme d'une fonctionnelle d'énergie qui prend an argument à la fois une fonction de multi-correspondance entre $\mathcal{H}$ et $\mathcal{D}$ (une version plus générale de ϕ) et de l'image elle-même. Ceci est important, car chaque pixel dans le résultat d'inpainting est alors une combinaison linéaire des pixels non-occultés, et non pas simplement des "copié-collés".

Les méthodes présentées précédemment ont pour but l'inpainting *image*. Or, il existe aussi une littérature (plus réduite) dédiée à l'inpainting dans le cas des vidéos.

Le premier travail sur l'inpainting vidéo à proprement parler est celui de Wexler *et al.* [123]. Auparavant, Bertalmìo *et al.* ont proposé la méthode de [13] pour la vidéo, mais il s'agissait simplement d'appliquer une méthode d'inpainting image à une séquence vidéo. Wexler *et al.* ont proposé une méthode basée sur la notion de patchs en trois dimensions (x, y et le temps). Ils essaient de minimiser la fonctionnelle suivante :

$$E_w(u) = \sum_{p \in \mathcal{H}} \min_{q \in \mathcal{D}} d^2(W_p, W_q). \tag{1.12}$$

On peut voir que cette énergie est très fortement liée à celle de Demanet *et al.* [44]. Les auteurs proposent de la minimiser dans un cadre multi-échelle, de manière itérative. Dans notre travail, nous nous appuyerons sur les bases de ce travail.

Une autre approche à l'inpainting vidéo basée sur les patchs est celle de Patwarhdan *et al.* [100]. Ce travail présente un algorithme très similaire à celui de Criminisi *et al.* [38], dans la mesure où il est un algorithme glouton qui remplace les patchs progressivement, avec un ordre de priorité lié au nombre de pixels connus dans le patch et aussi au mouvement. Cette méthode utilise une étape préliminaire de segmentation de la vidéo en avant et arrière plan. Un grand inconvéniant de cette méthode est sa nature gloutonne : effectivement, une telle approche a peu de chance d'assurer une cohérence globale de la solution. Ce manque de cohérence est beaucoup plus visible dans des vidéos que dans les images.

La dernière approche (chronologiquement) basée sur les patchs est celle de Granados *et al.* [58]. Celle-ci utilise une extension de l'approche de Pritch *et al.* [105], c'est-à-dire qu'ils minimise une énergie par rapport à un étiquetage discret. Cet étiquetage définit une association entre chaque pixel occulté et un pixel appartenant à la region non-occultée. Les auteurs montrent des résultats impressionnants de cette méthode, et ce sur des vidéos haute définition pour la première fois. Cependant, mis à part le masque d'occultation, il faut fournir un masque pour *chaque* objet en mouvement à reconstruire, et une segmentation de la video en arrière et avant plan est requis. La recherche de pixels est restreint à la zone indiquée par chaque masque. Même avec cette restriction, l'algorithme prend énormément de temps (jusqu'à 90 heures pour une vidéo de taille 1200×750 durant huit secondes environ) à cause du fait qu'il utilise une optimisation discrète. Ainsi, bien que cette approche puisse fournir des résultats très impressionnants, il souffre d'inconvéniants importants.

Il existe également plusieurs méthodes d'inpainting basées sur la notion d'"objets" plutô t que sur les patchs. Ces méthodes tentent d'identifier des objets en mouvement et les copient au mieux dans l'occultation. Elles se basent dans la vaste majorité des cas sur une segmentation en avant plan et arrière plan. Nous n'entrerons pas en détail dans ces méthodes. Parmi celles-ci, on trouve [37, 68, 112].

Nous présentons maintenant notre algorithme d'inpainting vidéo.

1.3.2 Algorithme d'inpainting proposé

Ayant regardé en détail les différentes approches à l'inpainting, nous avons choisi d'utiliser une méthode basée sur la notion de patchs, dans un cadre variationnel multi-échelle.

Nous utilisons la fonctionnelle d'énergie suivante, inspirée par le travail de Wexler *et al.* [124] :

$$E(u, \phi) = \sum_{p \in \mathcal{H}} d^2(W_p^u, W_{p+\phi(p)}^u), \tag{1.13}$$

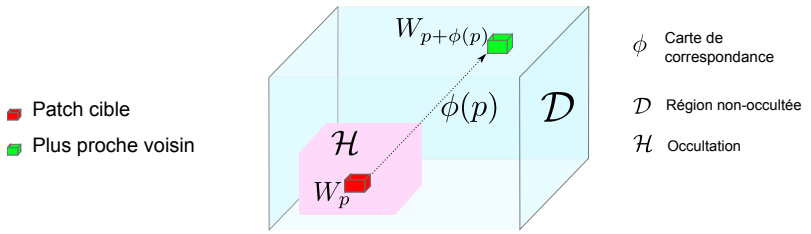


FIGURE 1.5: Illustration des notations utilisées dans la description de notre algorithme d'inpainting vidéo.

avec

$$d^2(W_p^u, W_{p+\phi(p)}^u) = \frac{1}{N} \sum_{q \in \mathcal{N}_p} \|u(q) - u(q + \phi(p))\|_2^2. \tag{1.14}$$

Cette fonctionnelle stipule qu'une bonne solution, u, est une image où chaque patch dans l'occultation ressemble autant que possible à son plus proche voisin dans $\mathcal{D}$, d'une façon analogue à Demanet *et al.* [44]. Une différence majeure dans la formulation que nous utilisons est qu'il s'agit d'une optimisation *continue*, et la vidéo est elle-même un argument à la fonctionnelle, comme dans le travail d'Arias *et al.* [5].

Cette formulation du problème est très puissant, puisqu'il intègre la notion de patchs, mais dans un cadre plus rigoureuse que celle de [38]. Combinée avec l'utilisation des pyramides multi-résolution, cette méthode peut produire des résultats qui conservent à la fois la texture et la structure, même si l'occultation est grande.

Cependant, il y a plusieurs difficultés liées à l'utilisation d'une telle approche :

– le temps d'exécution ;
– des ambiguïtés de comparaison de patchs à des résolutions grossières (notamment par rapport aux textures) ;
– des problèmes pour traiter des vidéos en mouvement.

Nous détaillons ces problèmes et les solutions que nous proposons ici.

1.3.3 Recherche des plus proches voisins

L'étape de loin la plus lourde en termes de temps de calcul est celle de la recherche de plus proches voisins. Effectivement, s'il est effectué de manière exhaustive, il peut prendre plusieurs semaines pour une vidéo de taille très modeste ($300 \times 100 \times 150$), sur un ordinateur de puissance moyenne.

Ce problème doit être adressé non seulement pour avoir un algorithme raisonnablement rapide, mais aussi simplement pour expérimenter. En effet, il est très difficile de savoir ce qui marche ou ne marche pas, où même de diagnostiquer les erreurs d'implémentation avec des temps d'exécution aussi longues. Cela explique pourquoi il y a relativement peu de travaux dans l'inpainting vidéo par rapport à l'inpainting image.

Le seul autre travail qui considère ce problème est celui de Wexler *et al.* [124], qui ont proposé d'utiliser une méthode de recherche basée sur les "kd-Trees" [6]. Malgré l'accélération ainsi obtenue, le temps d'exécution reste relativement élevé. Par rapport à l'exemple précédent d'une vidéo de résolution très basse, cette méthode fait passer le temps de recherche de plus proche

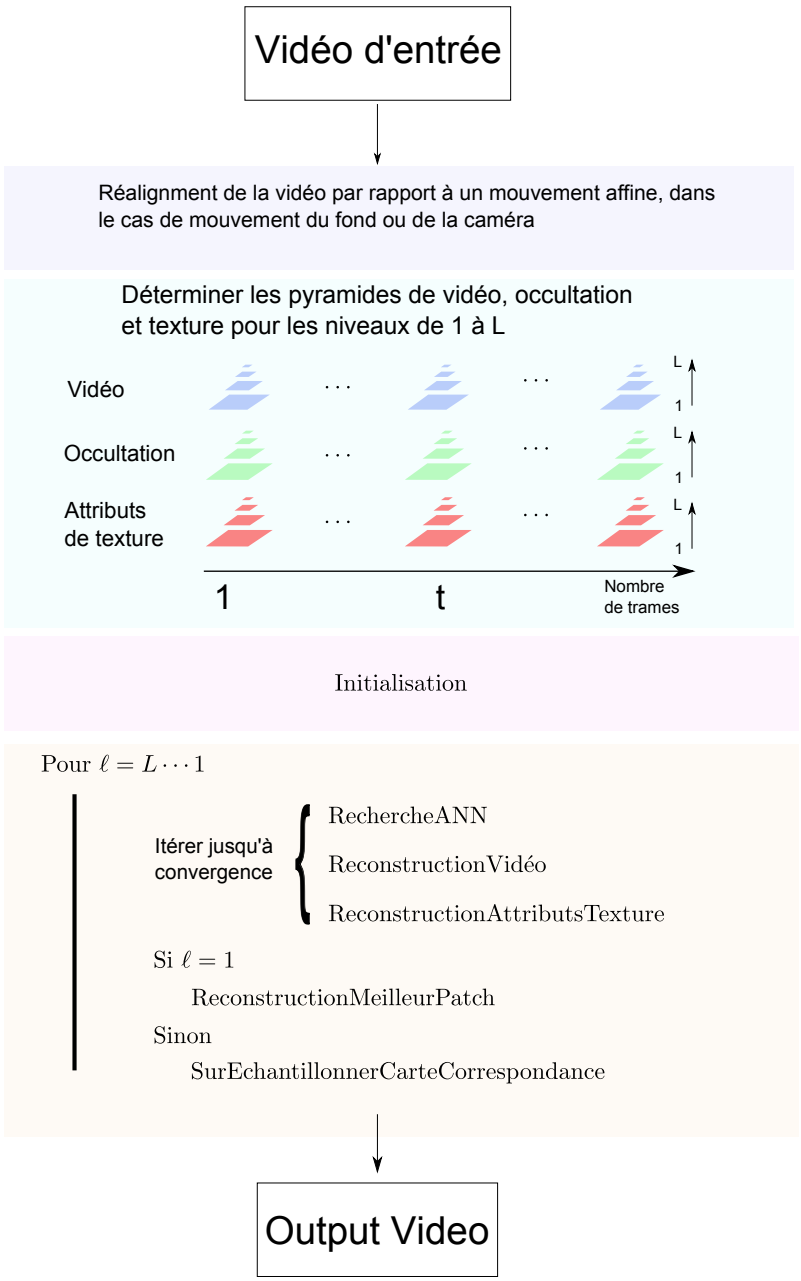


FIGURE 1.6: Illustration des étapes de notre algorithme d'inpainting vidéo.

voisins d'une ou deux semaines à une ou deux heures. Cela représente bien entendu une accélération considérable, mais est quand même trop lent pour être pratique. Sachant qu'il y a plusieurs itérations de la recherche dans l'algorithme, l'obtention d'une solution peut prendre une demie journée.

Pour traiter ce problème, nous proposons une extension de la méthode appelée "PatchMatch" de Barnes *et al*. introduite dans [11]. Cette méthode a été proposée initialement pour la recherche de plus proches voisins dans les images, mais les principes qui la sous-tendent peuvent être étendus au cas des vidéos.

L'algorithme est basé sur l'idée que de bonnes cartes de correspondance ont de fortes chances d'être *constant par morceaux*. En d'autres termes, un décalage relatif $\phi(p)$ qui mène à un bon plus proche voisin aura de fortes chances de mener à de bons plus proche voisins pour les patchs autour de W_p.

L'algorithme comporte les étapes suivantes :

– initialisation aléatoire ;
– propagation ;
– recherche aléatoire ;

L'initialisation consiste à associer un vecteur aléatoire $\phi(p)$ à chaque position p, avec $p + \phi(p) \in \mathcal{D}$. La propagation exploite le caractère constant par morceaux des bonnes cartes de correspondance : on essaie de propager la valeur de $\phi(p)$ aux positions autour de p, c'est-à-dire $(x_p - 1, y_p, t_p)$, $(x_p, y_p - 1, t_p)$ *etc...* . De bonnes valeurs de ϕ seront largement propagées dans la carte de correspondance. Enfin, la recherche aléatoire permet d'éviter de tomber dans des minimas locaux. La propagation et la recherche aléatoire sont itérés plusieurs fois afin de converger vers une bonne solution.

Avec cette méthode de recherche de plus proches voisins, nous observons une accélération d'environ 20-50 fois par rapport à celle de [11]. Les temps d'exécution sont présentés dans Table 1.1.

1.3.4 Les textures

Nous avons indiqué que les méthodes basées sur la notion de patch sont très efficaces pour l'inpainting correcte des textures. Cela est vrai parce que la zone de recherche de patchs est souvent restreint à une région autour de l'occultation. En revanche, lorsque la zone de recherche est toute l'image ou bien la vidéo, des ambiguïtés peuvent apparaître lorsque l'on essaie de comparer des patchs texturés. Cela peut arriver pour deux raisons :

– la résolution est trop grossière pour voir les textures ;
– la distance de patch est mal adaptée aux textures ;

La première raison est facile à comprendre : à une résolution trop grossière, les textures n'existent plus. La deuxième raison nécessite un court raisonnement statistique.

Considérons trois patchs, X, Y et Z. Supposons que les éléments x_i et y_i de X et Y sont des variables aléatoires iid et suivent une distribution Gaussienne : $\mathcal{N}(\mu, \sigma^2)$, où μ représente la moyenne et σ^2 représente la variance. Supposons que les éléments de Z sont égaux à μ partout : $z_i = \mu$. Dans notre raisonnement, les patchs X et Y représentent des patchs "texturés", et le patch Z correspond à un patch "lisse" (ou constant).

Calculons la valeur moyenne (espérance) de la somme des différences au carré (SSD) entre X et Y ($d(X, Y)$), et X et Z ($d(X, Z)$) :

$$E[d^2(X,Y)] = E[\sum_i (x_i - y_i)^2] = N2\sigma^2 \tag{1.15}$$

$$E[d^2(X,Z)] = E[\sum_i (x_i - z_i)^2] = N\sigma^2. \tag{1.16}$$

$$\tag{1.17}$$

On voit qu'en moyenne, la distance entre les patchs texturés est deux fois plus grande qu'entre un patch texturé et un patch constant.

Bien sûr, ce raisonnement est seulement une partie de l'histoire : on doit savoir si *parmi un ensemble* de patchs texturés, il existe *au moins un* qui est meilleur en termes de SSD que le patch

lisse. La probabilité de cela est plus compliqué à déterminer. En effet la SSD entre X et Y suit une loi de χ^2 de paramètre $2\sigma^2$. Après des simulations numériques, nous avons déterminé qu'à partir d'une taile de patch de 9×9, il est relativement peu probable que, dans une région texturée de taille raisonnable, on trouve un meilleur patch que le patch constant.

Ce problème est exacerbé par l'utilisation de PatchMatch. En effet, cet algorithme repose sur la propagation des bonnes valeurs de ϕ. Or, cette propagation n'est possible que si la carte de correspondance est constant par morceaux, ce qui n'est pas le cas des textures.

Afin de palier ce problème, nous proposons une modification à la distance de patch. La distance devient alors la SSD entre chaque élément de patch qui est représenté par le vecteur suivant :

$d^2 = [R, G, B, \beta T_x, \beta T_y]$.

Les composants T_x et T_y sont des composants qui reflètent la nature texturée d'une région. Après avoir testé plusieurs choix, nous avons décidé d'utiliser des attributs liés à la valeur absolue du gradient. Pour une région constitué d'un bruit Gaussien, ces attributs tendent vers la variance, multiplié par un constant. Ils représentent alors, d'une certaine manière, la texture locale.

Un point important est que ces attributs doivent être calculés à l'échelle la plus fine. En effet, à d'autres échelles, les textures disparaissent.

Bien entendu, les attributs sont inconnus au début du traitement. Ainsi, il faut remplir la zone inconnue par rapport aux couleurs de l'image, et par rapport aux attributs. Ce remplissage est fait de la même manière pour les attributs que pour les couleurs, et en parallèle.

1.3.5 Inpainting vidéo avec caméra mobile

Très souvent, il est nécessaire de remplir une vidéo dont le fond est en mouvement. Cela arrive lorsque le caméra qui a pris la vidéo est mobile. Cela pose de grandes difficultés pour des méthodes d'inpainting basées sur la notion de patchs pour la raison suivante.

En effet, l'utilisation de patchs repose sur la *redondance* de ceux-ci dans l'image ou la vidéo. Lorsqu'une caméra est en mouvement (surtout lorsqu'il y a des mouvements aléatoires), il y a peu de chance qu'un patch entier soit répété à l'identique ailleurs dans la vidéo. Afin de résoudre ce problème, nous effectuons une étape préliminaire de stabilisation de la vidéo (similaire à l'opération éffectué dans le filtrage temporel des rayures) en utilisant le travail d'Odobez et Bouthemy [97]. Cela a pour effet d'encourager fortement la redondance des patchs, et donc d'améliorer la qualité finale du résultat. Un exemple d'un tel résultat peut être vu dans la Figure 1.7.

1.3.6 Résultats

Dans cette section, nous présentons quelques uns des résultats de notre algorithme d'inpainting vidéo. Nous voudrions souligner en particulier la capacité de notre algorithme à :

- traiter des scènes avec des objets en mouvement (qui peuvent s'occulter) ;
- gérer correctement les textures vidéos ;
- gérer les scènes avec caméra mobile ;
- traiter des vidéos en haute définition.

On voit en particulier dans l'exemple "Les Loulous" (voir Figure 1.9) que l'algorithme peut gérer des situations avec caméra mobile, des textures vidéos et des objets en mouvements en même temps. De plus, nous pouvons traiter des vidéos de haute définition, de manière automatique.


Trame originale : “Girl” [57]


Résultat de [57]


Notre résultat

FIGURE 1.7: Une comparaison de notre résultat d’inpainting avec celui de l’algorithme d’inpainting d’arrière plan de Granados *et al.* [57]. Dans ces cas avec un fond mobile, nous obtenons des résultats de haute qualité (comme le font Granados *et al.*), mais nous le faisons dans un algorithme unifié. Cela illustre la capacité de notre algorithme à produire de bons résultats dans des situations difficiles et variées.

Algorithme	Temps d’exécution de la recherche de plus proches voisins.				
	Beach Umbrella 264x68x200	Crossing Ladies 170x80x87	Jumping Girl 300x100x239	Duo 960x704x154	Museum 1120x754x200
Wexler (kdTrees)	985 s	942 s	7877 s	-	-
Proposé (PatchMatch 3D)	50 s	28 s	155 s	29 min s	44 min
Algorithme	Temps d’exécution totale				
Granados	11 heures	-	-	-	90 heures
Proposé	24 mins	15 mins	40 mins	5.35 heures	6.23 heures

TABLE 1.1: Temps d’exécution totaux et partiels d’inpainting vidéo. Les temps partiels représentent le temps nécessaire pour une itération de la recherche de plus proches voisins.

1.4 Conclusion

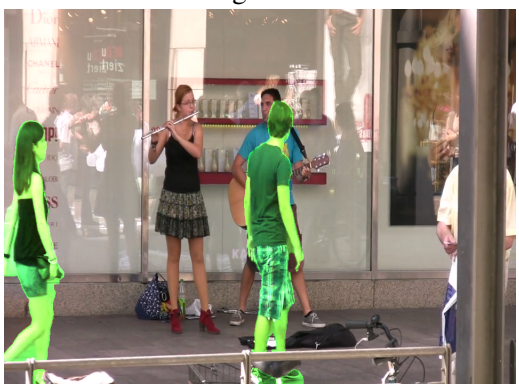
Dans cette thèse, nous nous sommes intéressés à deux sujets appartenant au domaine de la restauration d’images et de vidéos. La première était la détection de rayures dans le films anciens, et la seconde était l’inpainting vidéo.

Concernant la détection de rayures, nous avons proposé un algorithme robuste au bruit et au “clutter” dans les images, en se basant sur la méthodologie *a contrario*. Cet algorithme produit des détections avec une très grande précision, et peut gérer des rayures qui ne sont pas verticales, et qui ne sont pas complètement droites.

Dans un second temps, nous avons proposé un algorithme de filtrage de ces détections de rayures. En effet, l’algorithme spatiale laisse des fausses détections dues à des éléments verticaux de la scène. Notre algorithme identifie les trajectoires des objets qui conforment au mouvement de la scène, et les rejette comme étant des fausses alarmes. Ceci augmente la précision finale de la détection.

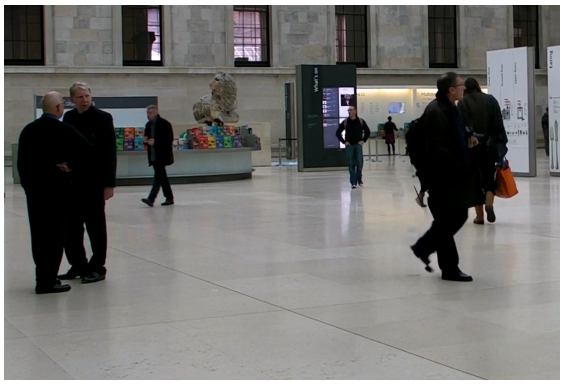
Enfin, nous nous sommes intéressés au problème de l’inpainting vidéo. Nous avons proposé un algorithme basé sur la notion de patchs, dans un cadre de minimisation multi-échelle. Le premier problème auquel nous nous sommes confrontés était l’accélération de l’étape de recherche de plus proches voisins des patchs. Nous avons étendu l’algorithme appélé PatchMatch au cas spatio-temporel afin de réduire les temps d’exécution de l’algorithme. Ensuite, nous avons modifié la distance de patch en introduisant des attributs de texture pour mieux identifier les zones texturées et éviter des reconstructions incohérentes au niveau de la texture. Enfin, nous avons adressé le

Trame originale : “Duo”



Trame originale : “Museum”

Résultat de [58]



Notre résultat d’inpainting

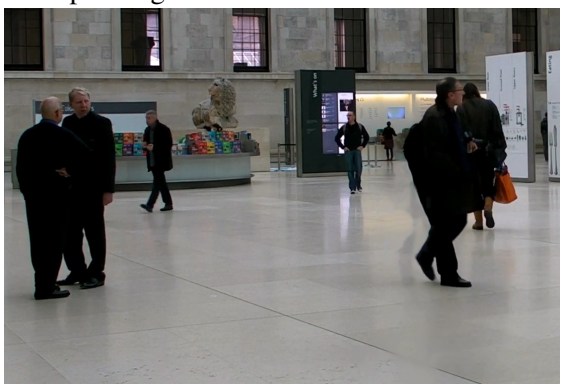


FIGURE 1.8: Nous obtenons des résultats similaires à ceux de [58] avec une accélération d’une ordre de grandeur, sans intervention manuel. Les masques d’occultation sont surlignées en vert. Les vidéos sont visibles à l’adresse suivante : http://www.enst.fr/~gousseau/video_inpainting.

FIGURE 1.9: Résultats d’inpainting vidéo.

problème de l’inpainting vidéo dans le cas des caméras mobiles.

Chapitre 2

Introduction

The restoration of images and videos is an essential task due to the inevitable degradations and modifications which these images and videos undergo in either the acquisition or storage processes. The general goal is to retrieve the original image or video, or at least to produce a plausible version. Restoration encompasses a wide range of tasks (denoising, inverse filtering, *etc*.), depending on the nature of the degradation which is being considered. In particular, one situation might require that we restore only a specific region of an image or video, as opposed to considering that the whole content is affected. This obviously requires a *detection* stage, which locates the problematic area. We might also consider that the information in this region is so greatly affected that it cannot be used for the restoration process, which leads to tasks such as "inpainting".

This thesis considers two such restoration problems : that of detecting (and locating) the presence of *line scratches* in films, and secondly the problem of filling in a region in a video, also known as "video inpainting". While the first problem is exclusively limited to the task of film restoration, the second can be said to go beyond such goals : we may also wish to modify the information for aesthetic purposes.

2.1 Detecting line scratches in films

Until the development of digital photography and cinematography, images and videos were captured using photographic film containing some photosensitive material (in particular, silver halide crystals). Unlike digital storage devices, film is necessarily subject to various physical degradations, such as tearing, scratching and various chemical and biological degradations. In particular, a very common defect is the *line scratch*, which occurs when a sharp object or part of a machine is scratched along the film itself, leading to black or white vertical lines on the projected image. Coloured scratches can also exist when the photosensitive material of the film is partially removed. Given the inevitable tendency of film to deteriorate, vast quantities of film need to be digitalised and restored in order to preserve their content. Manual restoration projects are extremely labour-intensive and therefore reserved for only the most high profile and important films. This leaves a large amount of film which will only be restored if automatic or semi-automatic processes are available.

A variety of film defects have been studied in image processing since the early 90's, such as dirt and sparkle, blotches, line scratches, flickering and vinegar syndrome. The line scratch's specific characteristics, in particular its *temporal persistence* (the fact that it appears in a similar spatial position in several subsequent frames), mean that it cannot be detected or restored in the

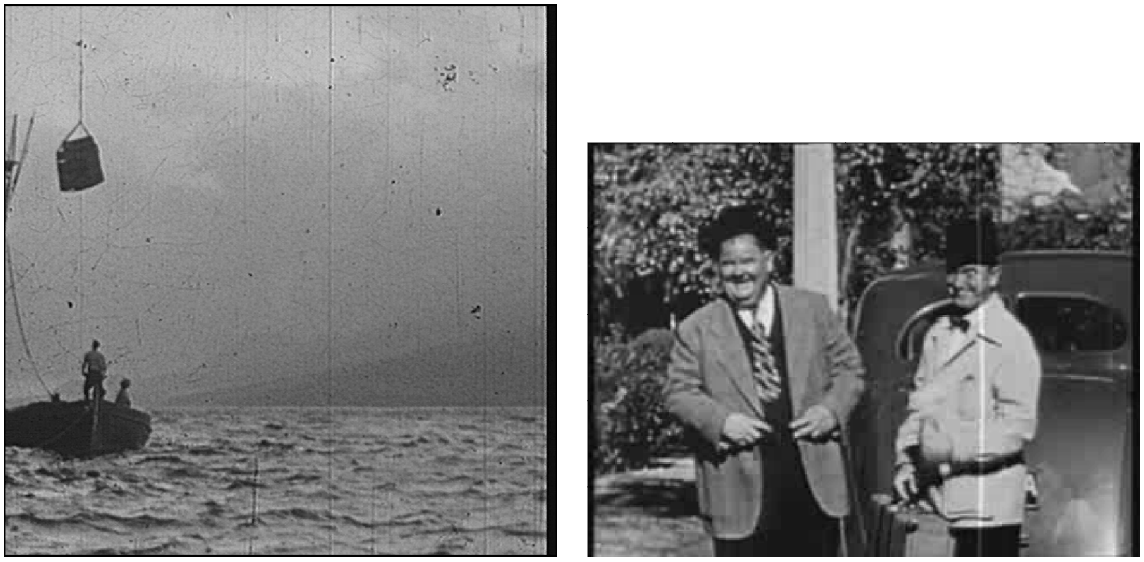


FIGURE 2.1: Two examples of line scratches in old films.

same manner as impulsive defects such as dirt and blotches. Line scratches may, in some cases, be restored with physical means such as chemical baths. However, such methods are necessarily more expensive and slower than digital restoration. Therefore, line scratch detection and restoration algorithms have been of interest to researchers. Two examples of line scratches can be seen in Figure 2.1.

2.1.1 Line scratch detection in single frames

The detection of line scratches was first considered by Kokaram in 1996 in [76]. The method presented in this work noted a useful tool for scratch detection : a 1D signal obtained by the summation of the image information over the columns. The presence of line scratches is amplified in this representation. Subsequently, this tool was used by many other detection algorithms [21, 25, 26, 72, 74, 119]. A common problem of such methods is that they do not locate line scratches precisely ; in general the output of these algorithms is a detected pixel column. This supposes that line scratches are straight, vertical and cover the whole of the image. These hypotheses are not always verified. Furthermore, experiments show that these approaches can be sensitive to noise and clutter.

In Chapter 3, we address the previous concerns. We propose a new algorithm for line scratch detection in still images and provide visual comparisons with the state-of-the-art. Quantitative evaluations of our algorithm may be found in Chapter 5.

2.1.2 Temporal filtering of line scratches

The main weakness of all line scratch detection algorithms in still images is their tendency to detect thin, vertical structures as line scratches. These structures can often be impossible to distinguish from true scratches, even for a human being. However, the difference becomes clear when the video is viewed, and not just the still image. This is because line scratches have very different temporal characteristics than those of objects which really belong to the scene content. In particular, the *motion* of the scratches and that of the scene content often differ vastly.

Input image Our inpainting result

FIGURE 2.2: An example of results of video inpainting (using the algorithm proposed in Chapter 6). The person in the foreground has been removed from the video.

The first to use such temporal information were Joyeux *et al*. [72] who hypothesised that line scratches display *sinusoidal* motion, which should distinguish them from scene elements. Subsequently, several other methods also tried to exploit temporal aspects for scratch filtering [16, 74, 71, 95]. In such approaches, the major challenge is to determine the *trajectories* of the scratch detection.

A common weakness of previous work is that they determine the aforementioned trajectories in a very *local* manner, resulting in a lack of robustness in the filtering algorithms. In Chapter 4, we formulate quite general hypotheses which help to distinguish true scratches from false detections, and explore how reliable scratch detection trajectories may be obtained.

Chapter 5 provides quantitative evaluations of both our detection and filtering algorithms in comparison to previous approaches, in terms of recall, precision and $F1$ score. We annotated several image sequences, and made the annotations publicly available. Systematic evaluations of line scratch detection algorithms was clearly missing from the literature, in part due to a standard database with both sequences and annotations.

2.2 Video inpainting

The second Part of the thesis deals with the problem of automatically filling in regions in videos, a task which is also known as *video inpainting*. Inpainting was first introduced by Masnou and Morel in 1998 [92] in the case of images, and the term "inpainting" was coined by Bertalmío *et al*. in [14]. These algorithms and other approaches in the same short time period [9, 34, 36, 52] were mainly concerned with the goal of correctly reconstructing *structure* or geometry in images.

After this, it became clear very quickly that an essential requirement for the successful inpainting of medium and large regions is the ability to correctly reconstruct *textures*. Texture synthesis approaches [49] provided the notion of image *patches* or *exemplars*. These patches were shown to be efficient at reflecting both structure and texture in images. Consequently, a series of patch-based inpainting approaches [18, 38, 47] exploited the notion of patches and multi-resolution techniques to produce impressive results on large occluded regions.

The next major step for image inpainting methods was to formulate the previous patch-based approach in terms of a global optimisation, in a discrete [44, 80, 88, 105] or continuous [5, 8, 41] setting. These methods represent some of the most sophisticated approaches to image inpainting available.

When applied frame-by-frame to videos, image inpainting provides inadequate results, largely because *temporal coherence* is not ensured. An approach specifically designed for video inpainting was first proposed by Wexler *et al*. in 2004 [123]. The goals of correctly inpainting structure and texture obviously apply for the video case, but the difficult task of reconstructing moving objects is added. The same notions as used by image inpainting, in particular patches, are reused for the video case.

Our goal in Chapter 6 is to produce an automatic, generic video inpainting algorithm which is able to deal with a wide range of situations, in reasonable execution times. To do this, we take an iterative patch-based approach, embedded within a multi-resolution framework. With such an approach many challenges appear, and we address these in the Chapter. Firstly, we look at the search for of similar spatio-temporal patches, which represents a major computational bottleneck. Next, we consider the problem of correctly inpainting textures in videos, something which has not been previously dealt with in video inpainting, and is particularly a problem in an iterative patch-based multi-resolution approach. We also consider the case of inpainting with moving background and/or cameras. Some more theoretical points are also addressed, such as the link between two popular inpainting approaches based on the optimisation of a global, patch-based functional, and certain convergence properties of one approach in very simple inpainting situations. An example of an inpainting result of the algorithm which we propose in Chapter 6 can be seen in Figure 2.2.

As a particular case study, Chapter 7 looks at the restoration of line scratches and in particular whether video inpainting is a good choice for the restoration of this defect. We also briefly explore the possibility of determining an optical flow in videos containing line scratches, and restoring the scratches using the trajectories of the pixels contained in the scratched area. We compare several restoration approaches on real and synthetic scratched image sequences.

2.3 Main contributions of the thesis

The main contributions of this thesis are as follows :

1. In Chapter 3, we propose an automatic, pixel precision algorithm which detects line scratches in single frames. This algorithm uses the *a contrario* detection methodology [46] to group and validate scratched pixels, with respect to a *background noise model*. We extend the classical *a contrario* approach by introducing a noise model which varies spatially. The resulting algorithm is robust to noise and clutter of varying characteristics. We compare the algorithm with previous work both visually and quantitatively. The proposed algorithm gives good recall on all the sequences which we used, and significantly improved precision with respect to the state-of-the-art, mainly due to the robustness of the algorithm with respect to noise and clutter.
2. In Chapter 4, we present an algorithm which temporally filters the scratch detections provided by the algorithm in Chapter 3. Our filtering algorithm is based on certain hypotheses concerning the motion of false alarms, and employs a robust, affine motion estimation [97] to estimate the global motion of the image sequence. Interestingly, the frame-by-frame detection algorithm of Chapter 3 is used to provide a robust estimation of the trajectories of the false alarms. Again, we evaluate our algorithm both visually and quantitatively. We observe that the filtering step improves precision, with almost no loss in recall (the filtering step *removes* detections, therefore it can only *decrease* recall).
3. In Chapter 6, we develop an automatic, generic video inpainting algorithm which is able to deal with a wide variety of challenging situations. At the core of the algorithm is a multi-

resolution *patch*-based approach which optimises a functional to produce coherent results. Firstly, we tackle the problem of looking for spatio-temporal patches which are in some sense similar to each other. This step is fundamental for our approach, but has previously presented such a computational bottleneck that such an approach has been infeasible for videos of reasonable resolutions. For this, we propose an extension of the *PatchMatch* algorithm [11] to the spatio-temporal case. Secondly, we consider the problem of correctly inpainting video textures. The distance used for comparing patches is modified to reflect certain simple gradient-based texture attributes. These attributes are used within a multi-resolution pyramid. Next, we look at the problem of inpainting with moving background or cameras. This problem is addressed with a pre-processing step which stabilises the video with respect to a global, affine motion. Finally, the question of the initialisation of the inpainting solution is considered in detail, and is shown to have a large influence on the quality of the final inpainting result. The resulting algorithm is automatic, and able to deal with many different situations. In particular, it is able to correctly inpaint dynamic textures, something which has not previously been achieved. Furthermore, we are able to deal with different situations (such as moving background) in one single framework, whereas previously separate algorithms [57, 58] were necessary. The execution time of our algorithm is an order of magnitude less than the state-of-the-art [58].

2.4 Publications

The work contained in this thesis have led to the following publications or submissions :

- **Adaptive line scratch detection in degraded films.** Alasdair Newson, Patrick Pérez, Andrés Almansa, Yann Gousseau. *European Conference on Visual Media Production (CVMP), 2012*.
- **Temporal filtering of line scratch detections in degraded films.** Alasdair Newson, Andrés Almansa, Yann Gousseau, Patrick Pérez. *International Conference on Image Processing (ICIP), 2013*.
- **Vers un inpainting vidéo automatique, rapide et générique.** Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, Patrick Pérez. *Gretsi, 2013*.
- **Towards fast, generic video inpainting.** Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, Patrick Pérez. *European Conference on Visual Media Production (CVMP), 2013*. **BEST STUDENT PAPER AWARD**.
- **Robust automatic line scratch detection in films.** Alasdair Newson, Andrés Almansa, Yann Gousseau, Patrick Pérez. *IEEE Transactions on Image Processing, to appear, 2014*.
- **Video inpainting of Complex Scenes.** Alasdair Newson, Andrés Almansa, Matthieu Fradet, Yann Gousseau, Patrick Pérez. *Journal submission.*

2.5 Websites pertaining to the work in this thesis

During the course of this thesis, several websites were used to display results, especially video results, and some explanatory videos. These websites may be found with the following links :

Section	url
Part I	http://perso.telecom-paristech.fr/~gousseau/scratches/
Part II	http://perso.telecom-paristech.fr/~gousseau/video_inpainting/

Première partie

Detection of Line Scratches in Films

Chapitre 3

Spatial Line Scratch Detection

3.1 Introduction

In this Chapter, we consider the problem of detecting line scratches in images, that is in still frames (as opposed to detection based on a sequence of images from a film). We shall first of all present a list of common film defects and identify the specific characteristics of line scratches. These characteristics will obviously have a considerable impact on the methods used for their detection. We then look at the literature dedicated to line scratch detection in still frames, and identify some of the advantages and drawbacks of the approaches of previous authors. After this, we present the proposed approach to line scratch detection which relies heavily on the *a contrario* detection methodology, after which we discuss the strengths and weaknesses of the algorithm and show some visual results. A quantitative evaluation of the proposed algorithm is carried out in Chapter 5.

3.1.1 A glossary of film defects

To understand the nature of the different film defects which we shall mention here, it is necessary to consider the physical film itself, that is the photo-sensitive material which is used to record a scene. A film is composed of several layers, the main ones being the surface coating, the emulsion and the base. The emulsion contains the photo-sensitive chemicals which are necessary for representing the light of a scene. The surface coating protects the emulsion to a certain extent, and the base acts as a support for the emulsion and surface coating, which would not be able to stay together otherwise. For further details, a good explanation of the composition of a film may be found at [1].

Given the physical characteristics of the different film components, different degradations may happen to the film. Some of the most common include :

- Dirt, dust ;
- Scratches ;
- Blotches ;
- Mould ;

Further effects which are analysed in old films include flickering and the removal/restoration of film grain, however these are not “defects” as such, since they do not concern degradations of the physical film.

In this section, we shall look at the *line scratch* in further detail. In particular, we shall deal with the problem of the *detection* of line scratches. The restoration of this defect will be considered

in Chapter 7.

Line scratches are produced in films when the film is rubbed against an object, often a film projector part, and scored so that either the base of the film, or the photosensitive material (the "emulsion") is removed, leaving a visible scratch on the resulting image. The scratch can either be on the base of the film, or the emulsion. In the case of the base, the image content is not destroyed. However, when light passes through the base for either viewing or scanning, it is diffracted and the subsequent image contains a visible semi-transparent vertical line. On the other hand, if the emulsion is scratched, then the actual image content is lost. These two types of defect production have consequences on methods of detection and restoration of the image content. For example, in the case of base scratches, one physical restoration possibility is the use of "wet gates". These are chemical baths in which the film is laid, which have the same refractive index as the base layer material, so as to reduce the diffraction effects. While this works with base scratches, it cannot restore completely unknown image content. This example illustrates the variable nature of line scratches, something which must be taken into account when designing detection and restoration algorithms.

The physical origin of the line scratch means that it has some very particular characteristics, and distinguishes it from defects such as noise, dirt and dust. These characteristics can be listed as :

- Scratches in old films are usually either black or white ;
- Scratches are roughly vertical ;
- Scratches are roughly straight, or at least piecewise straight ;
- Scratches are *temporally persistent*. This means that if a scratch is present at some position in the frame, then it will most likely continue onto the next frame in a similar position.

The first characteristic is obviously due to the fact that old films are often in black or white. However, even in colour film, black and white scratches exist, in the case of base scratching. In our work, it will be sufficient to consider that scratches are white or black. In the case of colour scratches, the film can either be converted to grey-scale or we can consider one colour channel only. The three other characteristics can be explained by the physical origins of line scratches. Line scratches are vertical due to the direction of their motion in projectors : the mechanical part which scratches them runs along the film parallel to this motion. Similarly, the scratches are straight because the horizontal motion of the mechanical part is slow compared to the motion of the video projector. Finally, again due to the mechanical part's motion, the scratch is temporally persistent : the motion of the mechanical part is not great enough to make the scratch move a great deal from image to image.

A good physical explanation of line scratches may be found at [1].

The previous characteristics mean the detection and restoration of scratches are a difficult problem. In the case of defects such as dust and blotches, for example, the temporal impulsiveness of the defects is a recognisable characteristic, and therefore is a great help to both detection and restoration (see [29]). Scratches, on the other hand, present less easily identifiable characteristics and may often be confused with thin vertical structures in images. Therefore, both the detection and restoration of line scratches must be carried out with methods which are specially adapted to the characteristic features of scratches.

3.2 Related work

The first work on defect detection in films which was specifically geared towards line scratches was done by Kokaram in [76]. Kokaram suggested an additive grey-level profile in the spatial

vicinity resulting from the line scratch. Let x and y represent, respectively, horizontal and vertical coordinates in the images. If we consider that the degraded image is be affected by P scratches, and the centres of the scratches occur at the horizontal indices x_p the degradation model of Kokaram can be written :

$$G(x,y) = I(x,y) + \sum_{p=0}^{P-1} L^p(x - x_p) + e(x), \tag{3.1}$$

where I is the original (undegraded) image, G is the observed (degraded) image, L^p is the profile due to the scratch centred at x_p and $e(x)$ is an additive Gaussian noise. The scratch profile is defined as :

$$L^p(x,y) = b_p k_p^{|x-(m_p y + c_p|} \cos(\frac{3\pi|x - (m_p y + c_p)|}{2\omega_p}), \tag{3.2}$$

where b_p is the brightness of the line scratch, k_p is the decay coefficient of the scratch profile, m_p is the slope of the line and c_p is the point at which the line scratch intercepts the horizontal axis. This scratch profile model is based on an empirical observation concerning the shape of scratches, notably the presence of "side-lobes" either side of a scratch. These side-lobes are the consequence of the diffraction of light through the scratches, as explained by Bruni *et al*. in [26].

For a given line in the image, the parameters are estimated in a Bayesian manner, and a threshold on the value of b_p is used to decide whether the considered line corresponds to a line scratch. In order to speed up the execution time, a pre-processing step is used to determine likely candidates for line scratches. This is done by computing the Hough transform on a binary image I_B. This binary image is obtained by thresholding the absolute difference between the image I and a horizontally median filtered version of the same image (which we call I_m) :

$$I_B(x,y) = \begin{cases} 1 & \text{if} \quad |I(x,y) - I_m(x,y)| \geq s_{med}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

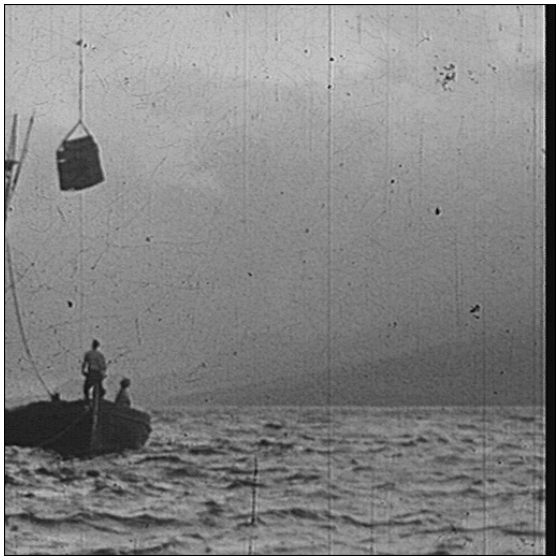
Additionally, Kokaram vertically subsamples the image I in order to accentuate the presence of scratches. An example of such a binary image may be seen in Figure 3.1.

A series of papers by Bruni *et al*. ([25], [26], [119]) extend the previous work of Kokaram. In these papers, the authors use the same hypothesis concerning the presence of side-lobes around the line scratches. However, rather than estimating the line profile (which is a damped sinusoid) in a Bayesian manner, they approximate it with a triangle, for each candidate scratch. The area of this triangle is calculated, and corresponds to the "energy" of the candidate scratch. This energy has to be above a certain threshold in order to be visible and therefore accepted. The threshold is set to :

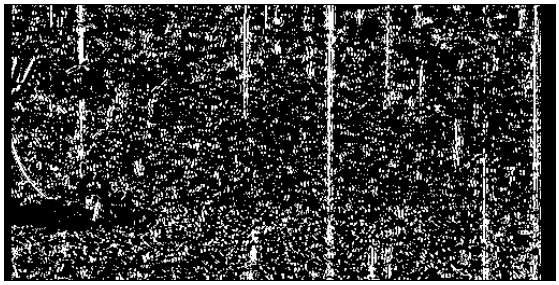
$$E_{min} = \frac{E(\tilde{x})}{0.98}, \tag{3.4}$$

where $E(\tilde{x})$ is the average energy of all the potential line scratches. The value 0.98 is based on *Weber's law*, which states that an object of brightness f_0 is visible amongst other objects of brightness f_s if $\frac{|f_s - f_0|}{f_0} \geq 0.02$.

This approach suffers from several drawbacks. Firstly, the line scratch is represented as a completely vertical line covering the whole of the image. This is clearly not the case for many scratch examples. Secondly, the visibility threshold is calculated in a global manner, meaning that the brightness of a scratch which is on the far left hand side of the image may affect the detection



Original image



Subsampling and detection of potential scratch points with a median filter

FIGURE 3.1: Illustration of the binary detection map produced by the approach of Kokaram [76].

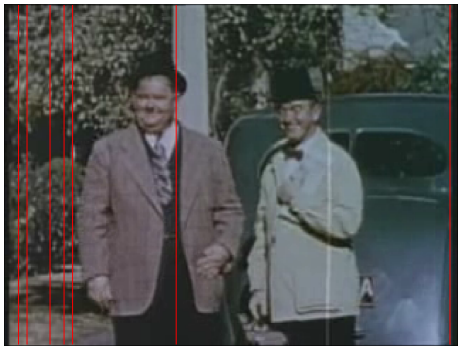
of a scratch on the far right hand side, which is not very robust, and does not adapt locally to the varying contrast of the image.

In Figure 3.2, two examples of Bruni's detection results may be seen. On the first example ("Laurel and Hardy"), we observe over-detection in the top left hand side of the image, which corresponds to tree leaves. The second example is a random image, with a Gaussian distribution. This example illustrates the lack of robustness of the threshold based on Weber's law. Obviously, the case of pure noise may be extreme, but it is a good illustration of the algorithm's lack of robustness in the presence of noise and texture.

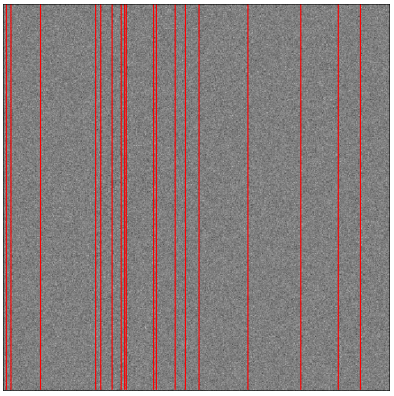
In [21], Bretschneider *et al.* use a simple thresholding of the vertical detail coefficients of a wavelet decomposition to identify scratches.

In [75], Kim *et al.* propose a binary pixel-level scratch detection method, after which the detections must be grouped into scratch segments, for example with morphological operations.

There are several unresolved problems in the literature concerning spatial scratch detection. Firstly, the scratch is most often represented as a straight, vertical line. While this may be the case in the examples which are commonly shown in most of the papers, it is not true in general. In some examples, the slant of the scratch may be quite significant, which renders the vertical representation almost useless for restoration purposes. Also, line scratches may be constituted of



Scratch detection from "Laurel and Hardy"



Scratch detection in a white noise image

FIGURE 3.2: Detection with the algorithm of Bruni *et al*. [25]. A noticeable problem is the tendency of the algorithm to mistake textured areas for line scratches. This is exacerbated when we try to detect scratches in a white noise image containing no scratches.

several segments of differing slopes, making the common vertical representation inefficient.

Additionally, we also note that that existing algorithms cope badly in noisy or textured regions. An example which illustrates this may be seen in Figure 3.2. We shall address this problem directly in our work.

3.2.1 Detecting lines in images

The problem of detecting "lines" in images is quite an old problem in the image processing community. The approach generally taken is to calculate the image gradient and detect lines as alignments of strong gradients in the same direction. The challenging part of this processing is the robust detection of an alignment of strong gradients. Undoubtedly the most famous method for doing this is the use of the Hough transform.

The Hough transform

The Hough Transform was first proposed by Paul Hough in [67]. The form which is commonly used now was proposed by Duda and Hart in 1972, in [48], for the purpose of automatically detecting lines and curves in images. An interesting introduction to the Hough Transform may be found in [61]. One of the advantages of this method is that it considers the prominence of the lines or curves in a global manner, which makes it robust to errors and missing parts of the objects. The Hough transform takes a binary input image and transforms it into a *parameter* domain. This parameter domain basically shows how prominent a line or a curve with a given set of parameters is in the original binary image. The pixels in the binary image represent points of the initial image which have a strong gradient norm, which is often the result of the application an edge detector.

In the case of lines, a parametrisation must be chosen. The first method that one would naturally think of is the use of the two following parameters : the slope and intercept of the line. This is represented by the equation $y = ax + b$. Unfortunately, as mentioned by [48], the parameters are not bounded, so they propose a parametrisation using the polar coordinate system : $\rho = x\cos(\theta) + y\sin(\theta)$. If we consider the vector going from the origin of the coordinate system to the closest point to the line being considered, then ρ is the norm of the vector, and θ is its angle.

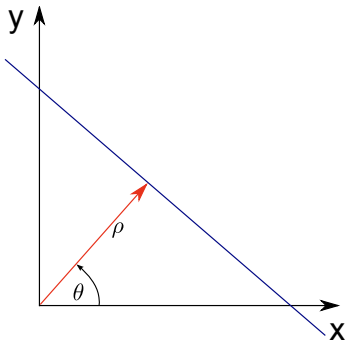


FIGURE 3.3: Illustration of the parametrisation of a line as used by the Hough transform.

In this situation, ρ and θ are bounded : $\rho \in [0, \sqrt{M^2 + N^2}]$ (M and N are the image dimensions) and $\theta \in [0, \pi]$. Furthermore, perturbations with respect to ρ and θ are directly related to geometric transformations (translations and rotations). The Hough transform consists of counting the number of pixels which "vote" for each line in the image. This voting process is also called accumulation, and the transformed image is sometimes known as the accumulator. Finally, to detect the prominent lines, the local maxima in the accumulator are found (often above a certain threshold), the final result are the lines described by the parameters given at these local maxima. An illustration of the parametrisation of a line can be seen in Figure 3.3

Mathematical morphology

Another set of methods which is reasonable to use for line detection in images is mathematical morphology. As before, this sort of approach takes a binary image which is produced using the image gradient. We will not use mathematical morphology here, so we shall not go into any further detail. We note, however, that these methods have been extensively used for film defect restoration by Décencière in his thesis [42], and are mentioned by Joyeux *et al*. in [72]. One disadvantage of such approaches is that they detect lines in a local manner, without considering whether the line is globally very present in the image. This contrasts with the Hough transform.

The two previous methodologies (the Hough transform and mathematical morphology) each present different advantages and drawbacks. One clear contrast is that the Hough transform detects complete lines in an image, whereas mathematical morphology-based approaches necessarily detect *segments*, that is to say collections of pixels. The former is an exclusively *global* approach and the second can only be a local approach.

A contrario detection

The *a contrario* methodology is a generic way to detect visual "objects" in digital images. The general approach is to detect objects in images which are unlikely to have been produced by some *background model*. This model usually relies on an independence assumption on the basic elements to be grouped for the detection.

We shall first present the *a contrario* approach as it is used to detect line segments in [45], although it may also be used for the automatic detection of various other "objects" (histograms, vanishing points, *etc*.). Given a line segment made of l pixels, a variable x_i is associated to each pixel. The variable x_i is equal to 1 if the pixel is aligned with the segment and 0 otherwise. "Aligned" pixels are those whose gradient orientation is orthogonal to the segment orientation (up

to some angular precision $p\pi$ radians, with $p \in [0, 1]$). Let $s = x_1 + \cdots + x_l$ be the number of aligned pixels. Larger values of s are associated to more meaningful line segments. Now, the detection of segments require thresholds that depend on l and p and are therefore non-trivially set. The aim of the *a contrario* approach is precisely to set these thresholds. The detection relies on the probability distribution of s under some *background model*.

In the case of line segments the background model specifies that all gradient orientations are independent and follow a uniform distribution in $[0, 2\pi]$. Let X_i be a random variable associated to the deterministic observation x_i. Under the background model, each X_i is independently and identically distributed, following a Bernoulli distribution of parameter p (equal to the angular precision), so that the random number of aligned points $S_l = X_1 + \cdots + X_l$ follows a binomial law :

$$\Pr(S_l \geq k_0) = \sum_{k=k_0}^{l} \binom{l}{k} p^k (1-p)^{l-k} =: B(p; k_0, l). \tag{3.5}$$

Segments of length l having k_0 aligned pixels are meaningful when $B(p; k_0, l)$ is small enough. Intuitively, this probability is small when the observed segment has a number of aligned points k_0 which is too large to occur by chance (as specified by the background model). In order to threshold this probability, the total number of tested line segments has to be taken into account, that is the number of possible segments that are being considered in the image. For this, one considers the *number of false alarms* (NFA), defined in [45] as :

$$\mathrm{NFA}(l, k_0) = N_{tests} B(p; k_0, l), \tag{3.6}$$

where N_{tests} is the total number of segments to be tested. It is easily seen that this number may be approximated as $N_{tests} = M^2 N^2$, with M and N the linear dimensions of the image. Segments are then detected using the following definition :

Definition 1. *A segment is ε-meaningful if* $\mathrm{NFA}(l, k_0) \leq \varepsilon$ *for some parameter ε.*

In other words, a segment is only meaningful if the number of false alarms under the background model is less than ε. It is shown in [46] that such a definition of meaningful segments implies that the expected number of detected segments under the background model is bounded by ε.

Maximality With the previous detection procedure, many redundant segments are detected. This is because a very meaningful segment often contains, and is contained by other segments which are also ε-meaningful. In order to keep only the best detection for such cases, Desolneux *et al*. introduced in [45] the notion of *maximality*.

Definition 2. *A segment A is* maximal *if :*
- $\forall B \subset A, NFA(A) \leq NFA(B)$
- $\forall B \supset A, NFA(A) \leq NFA(B)$.

In other words, a group of objects is maximal if it neither contains nor is contained by another group which is more meaningful (that is, a segment with a smaller NFA) than itself. A segment which is both maximal and meaningful is referred to as *maximal meaningful*.

This property is extremely useful for two reasons. First of all, it provides the best spatial localisation of the objects being processed, which is important for detection purposes. Secondly, the NFA has the following properties (see [45]) :

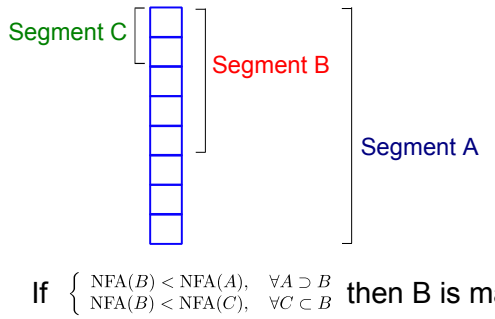


FIGURE 3.4: Illustration of the maximality principle.

- $NFA(k_0 + 1, l_0) < NFA(k_0, l_0)$
- $NFA(k_0, l_0) < NFA(k_0, l_0 + 1)$
- $NFA(k_0 + 1, l_0 + 1) < NFA(k_0, l_0)$.

These properties imply that maximal meaningful segments start with an aligned pixel preceded by an unaligned pixel (a 0-1 transition), and end with an aligned pixel followed by an unaligned pixel (a 1-0 transition). So, for example, we know that a segment which has several undetected pixels at its extremities is not maximal meaningful. An important consequence of this is that the number of segments which need to be tested in the binary image is greatly reduced, which reduces the computational cost of the algorithm.

Exclusion principle Since line segments are often have a thickness of several pixels, segments with different orientations may correspond to the same line. Again, in the interest of as precise a representation of the segments as possible, Desolneux *et al*. introduced the *exclusion principle*.

Definition 3. *If A and B are two groups obtained by using the* a contrario *principle, then the* exclusion *principle states that a pixel may belong to one group only.*

If a pixel p is contained by several segments, then the most meaningful segment retains p. All other segments which contain p have this pixel removed. The NFAs of the modified segments are then recalculated, and those that are no longer ε-meaningful are thrown away. This principle can be extended to include pixels which are at a distance of τ_x from more than one segment. In other words, pixels which are at a distance of τ_x from two or more segments must be associated with one of the segments only.

We note here that the reason why the maximality and exclusion principles can be safely applied is that the *a contrario* methodology provides us with a criterion (the NFA) with which we can compare segments of any length. This would not necessarily be the case if we based the "meaningfulness" of a segment on a more naïve criterion, such as simply the number of aligned pixels in the segment.

In comparison with the other previously mentioned line detection methods (the Hough transform and mathematical morphology), the *a contrario* methodology has the advantage of incorporating local and global detection aspects. It may be seen as "local" because the output is a collection of segments. It may be viewed as "global" because the segments are validated based on a criterion (the NFA) which is dependant upon the total number of tested segments, and also because the maximality principle chooses the best segments out of all those which belong to a certain line.

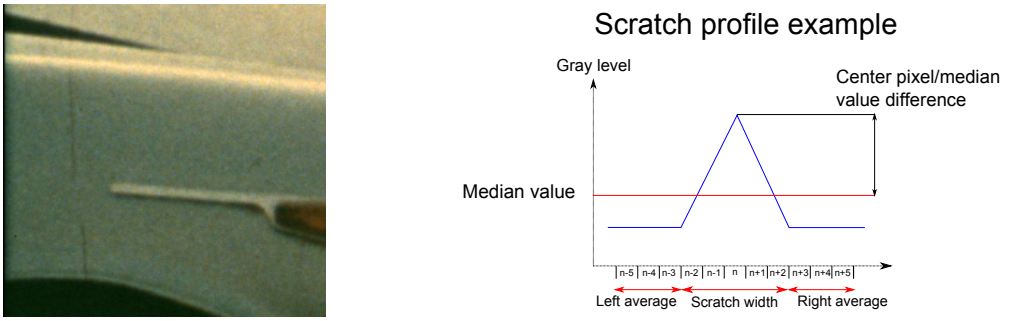


FIGURE 3.5: Closeup of a line scratch and an illustration of the 1D profile of a line scratch.

3.3 Proposed Algorithm for Detecting Line Scratches in Still Frames

As previously mentioned, most line detection algorithms start by producing a binary image based on the strong image gradients, and follow this by some sort of grouping step. We shall also take this approach.

We note that all the processing done in the context of scratch detection is carried out on grey-scale images, since the scratches are black or white in the majority of cases. When the image is a colour image, we transform it into a grey-scale image. It is true that some scratches may be coloured in more recent content, but we have tested our algorithm on such images and found that it applies also to such examples. In actual fact, we could probably enhance the detection by only processing the channel which corresponds to the colour of the scratch, however this may be considered a parameter which may be tuned easily for each sequence, if so desired.

3.3.1 Pixel-wise detection criteria

The first task is to identify the potential scratch points by relying on a pixel-wise detection criterion. While the detection of line segments in images entails only the detection of strong gradients, for the case of line scratches the case is slightly more complicated. We want to detect "peaks" or "troughs" in the grey level of the images. An illustration of this may be seen in Figure 3.5. This figure also shows the scratch model which we have chosen. This is simply a peak (in the case of a white scratch) surrounded by two areas of similar average intensity. We now present the method of detecting pixels which verify this scratch model.

Our pixel-wise detection criterion is a close variant of the classical test introduced by Kokaram [76]. It consists of a threshold on the difference between the grey-scale image, and a horizontally median filtered version of this image. This test basically detects outliers with respect to a horizontal neighbourhood. Contrary to the work of Kokaram, we do not take the central pixel into account when determining the median value, since it is supposed to represent the peak or trough, and we therefore wish to minimise its influence on the median value of the "background". In [76], the image is vertically subsampled before the thresholding, to highlight the scratches. Instead of this pre-processing step, we use a 3x3 Gaussian filter with a standard deviation of one pixel to reduce the noise in the image. This pre-processing step is crucial for our pixel-wise detection criterion to show the scratches correctly, as may be seen in Figure 3.7.

This criterion has the advantage of being quite generic : there are relatively few missed scratch pixels. On the other hand, a drawback of this criterion is its tendency to detect steep intensity

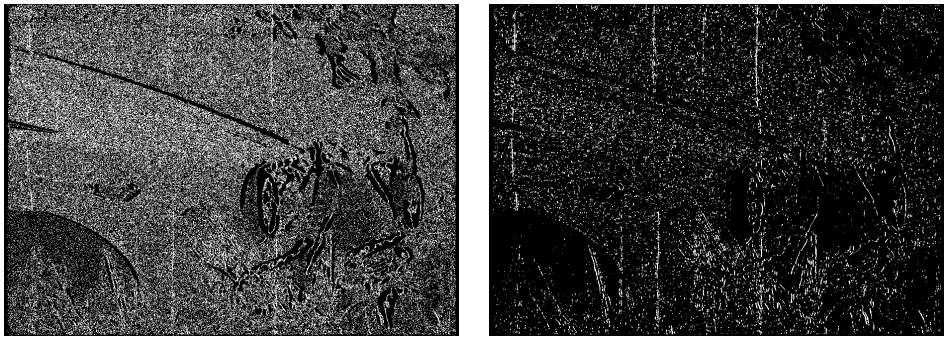
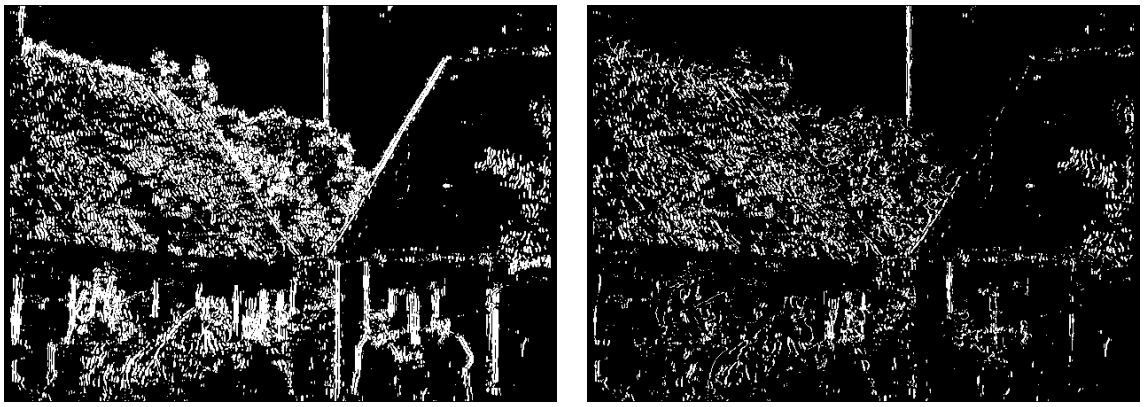


FIGURE 3.6: Binary detection of scratches, without and with a Gaussian filtering pre-processing step.



Binary scratch point detection with Equation (3.7)

Binary scratch point detection with (3.7) and (3.8)

FIGURE 3.7: Binary detection with first pixel-wise detection criterion c_1 (left) and with both criteria c_1 and c_2. The first criterion (difference with a median filtered version of the image) picks up strong unwanted intensity ramps (object edges). These detections can be filtered by a further criterion concerning the average grey levels of the left and right neighbourhoods.

fronts, rather than just "peaks". We avoid this by stipulating that the average grey-level values should be similar on either side of the scratch, as is coherent with our scratch model in Figure 3.5.

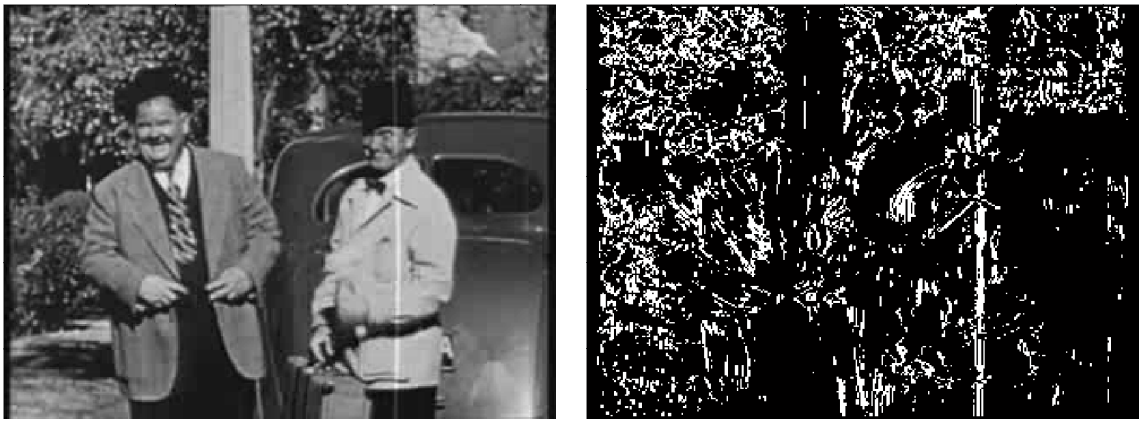
Our final pixel-wise detection criteria may be written in the following manner. Let $I_g(x, y)$ be the Gaussian filtered grey level image. Let $I_m(x, y)$ denote the median value over a local horizontal neighbourhood of pixel (x, y), and $I_l(x, y)$ and $I_r(x, y)$ be the left and right horizontal averages, as defined below. The two Boolean criteria are :

$$c_1(x, y) : |I_g(x, y) - I_m(x, y)| \geq s_{med} \tag{3.7}$$

$$c_2(x, y) : |I_l(x, y) - I_r(x, y)| \leq s_{avg}. \tag{3.8}$$

where, s_{med} and s_{avg} are grey-level thresholds. We can therefore define a binary image indicating detections as

$$I_B(x, y) = \begin{cases} 1 & \text{if} \quad c_1(x, y) \quad \text{and} \quad c_2(x, y) \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$



(a) Original frame (b) Binary detection image

FIGURE 3.8: Binary detection image from "Laurel and Hardy". White pixels are detected and black pixels are not.

Figure 3.5 shows an illustration of our two detection criteria.

For the proposed criteria, we set the width of the median filter to 5 pixels, and the value of s_{med} to 3 grey levels. These values are the same as in [76] and also appeared to us to be good empirical choices. The left and right averages are each taken over 3 pixels on either side of the 5 central pixels, and s_{avg} has been experimentally set to 20 grey levels. These parameters were used for all the examples in the experimental section.

As may be seen in Figure 3.8, such a pixel-wise detection is bound to produce many false alarms, and also misses some scratch pixels. Therefore, a further *grouping* step is needed in order to determine the *significant scratch segments* present in the pixel-wise scratch detections.

3.3.2 First approach to scratch point grouping : the Hough transform

As mentioned previously, the Hough transform is probably the most well known method of detecting lines in images. Therefore, this was the first method we tested for the grouping step.

The most common way to exploit the Hough transform is to identify local maxima in the accumulator and consider them as prominent lines. Figure 3.9 shows the results obtained if this approach is taken. The 50 strongest peaks were taken, and the corresponding lines were segmented using a simple heuristic : each segment has a minimum length, and a maximum gap size. This sort of approach is clearly very similar to using mathematical morphology. The parameters were tuned for the first sequence, "Les Choses de la Vie". For this sequence, the detection works relatively well. However, several problems can be seen in this and the other sequences. Firstly, in the cases where the detections correspond to true scratches, the detection is not particularly precise, since spurious pixel-wise detections can make the segments longer than we would wish. Secondly, it may be seen that the "prominent" lines are strongly influenced by the spurious pixels or "noise". This is very clear with the second two examples shown in Figure 3.9. Noisy areas induce strong maxima in the Hough accumulator, and are therefore detected, and since the threshold is a global one there is no way to take the local noise level into account.

Consequently, a robust approach is needed to group the pixels into segments. In order to do this we turn to a more sophisticated set of methods known as *a contrario* methods, used for alignment

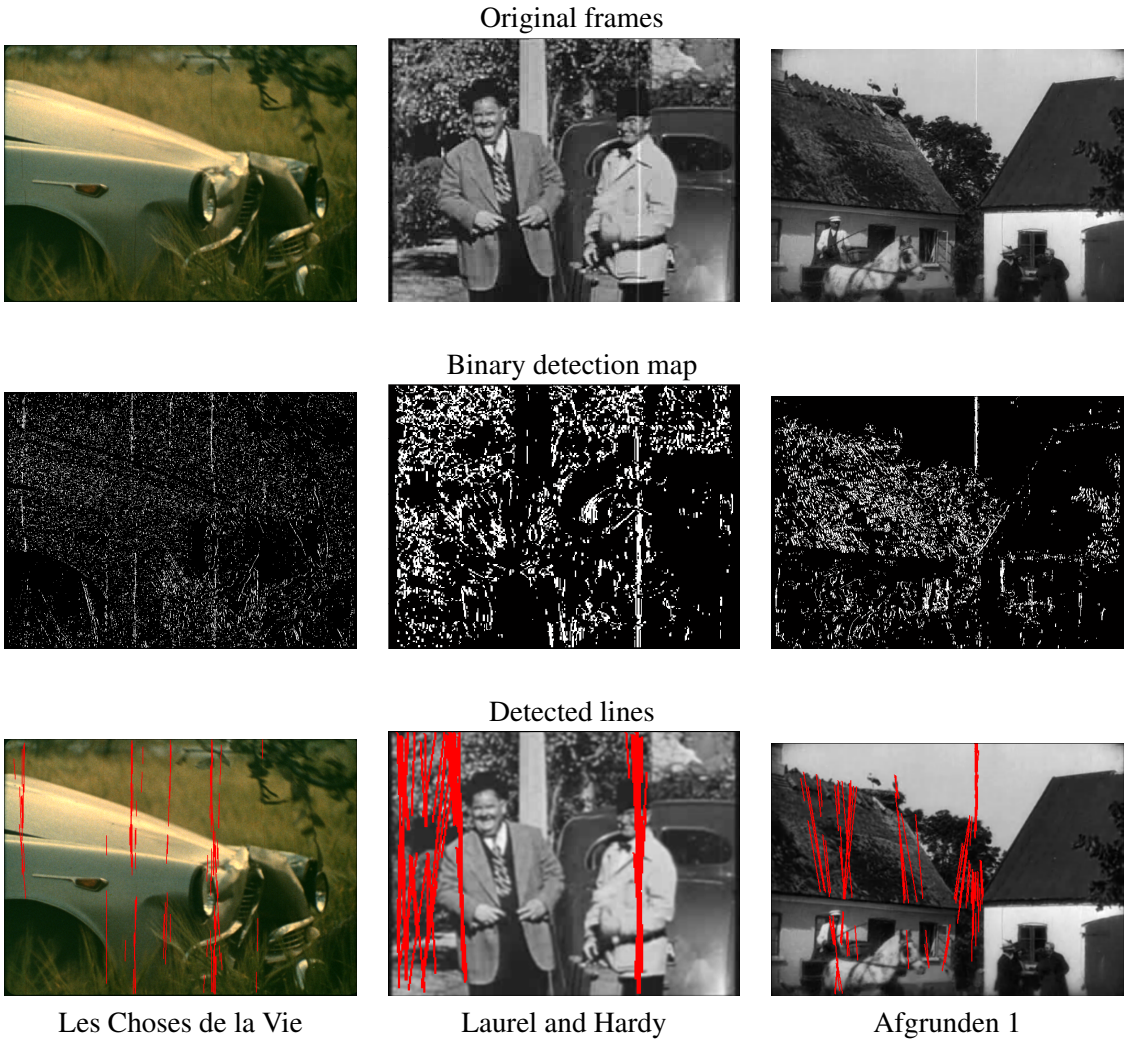


FIGURE 3.9: Detection using the Hough transform and a simple heuristic to group the pixels belonging to prominent lines. The parameters were tuned for the first sequence ("Les Choses de la Vie"). It is quite clear that using the same set of parameters is not satisfactory when different sequences are considered. Furthermore, it may be seen that using a global threshold on the prominent lines found in the Hough transform can induce incorrect detections in noisy and cluttered areas.

detection by Desolneux *et al.* in [46] as previously mentioned. This methodology posses certain desirable properties such as controlling the number of false alarms under a background model and producing a precise and unique detection for each scratch by using the maximality and exclusion principles. We now describe our use of the *a contrario* methodology.

3.3.3 *A contrario* line segment detection

The first step when using the *a contrario* methodology is to determine a background model, and therefore the detection probability p. To avoid too many different notations, we keep same letter (p) for the detection probability as in the case of the detection of aligned gradients. However, we underline that p now corresponds to the probability of a pixel in the background model being detected. This background model should represent as closely as possible everything which we do not wish to detect. In the case of the meaningful alignments detection of Desolneux *et al.* [45], this is an image where the gradient directions are uniformly and independently distributed in $[0, 2\pi]$. This accurately describes the areas of an image which contain no strong edges (textured areas, flat areas, noisy areas *etc.*). We shall see that our case is somewhat different.

Let us consider first of all a Gaussian white noise image, with variance σ^2, as a background model. Let $f_G(x; \mu, \sigma^2)$ represent the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. The variable x represents a grey-level value. In our case, the probability of the background model producing a "detected pixel" needs to be revisited. The probability of the second criterion (Equation (3.8)) being verified in our noise model is easy to calculate. This is simply :

$$\Pr[c_2] = \int_{-s_{avg}}^{+s_{avg}} f_G(x; 0, \frac{2}{w_{avg}}\sigma^2)dx, \tag{3.10}$$

where w_{avg} is the number of pixels taken into account in the left and right neighbourhoods. The probability of the first criterion (Equation (3.7)) being true in our noise model is more complicated to calculate. Let w_s be the scratch width, and w_m the index of the median over $w_s - 1$ pixels, so that $w_m = \frac{w_s}{2}$ if w_s is even. The distribution f_m of the median value can be calculated using order statistics [122] in the following manner :

$$f_m^{\mu,\sigma^2}(x) = \frac{(w_s-1)!}{(w_m-1)!(w_s-1-w_m)}[F_G(x;\mu,\sigma^2)]^{w_m-1}[1-F_G(x;\mu,\sigma^2)]^{w_s-1-w_m} f_G(x;\mu,\sigma^2). \tag{3.11}$$

To determine if the absolute difference between this median value and the central is greater than s_{med} one would need to perform the correlation between the distributions $f_m(x)$ and $f_G(x; \mu, \sigma^2)$, and integrate the tails of the resulting distribution. The final distribution would be the product of the distributions corresponding to our two criteria.

Figure 3.10 illustrates the detection probability of detecting a scratch pixel in various Gaussian background noise models. It is obvious that the detection probability depends on the value of σ. This is in stark contrast to the criterion used for line segment detection, for which the detection probability is constant. Given that the theoretical calculation of the scratch detection probability shown in Equation (3.10) and Equation (3.11) are relatively complicated to calculate, and moreover that they depend on a parameter σ which may vary from image to image, it is preferable to estimate the scratch detection probability numerically and directly on the image being considered.

Global estimation of the background noise model One very obvious approach to estimating the detection probability p in the background model is simply to calculate the proportion of pixels

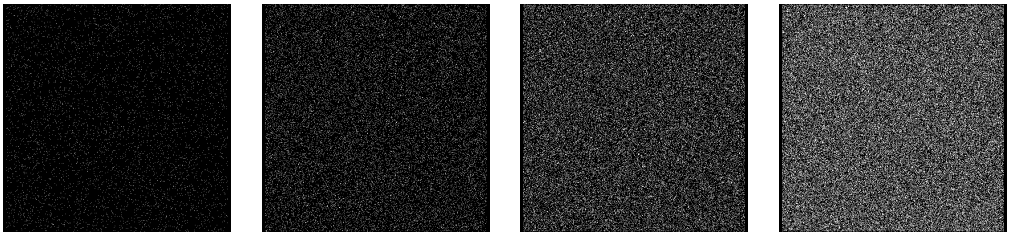


FIGURE 3.10: Pixel-wise scratch detection on different Gaussian background models with increasing standard deviation σ. The detection probability is clearly dependent on the value of σ.

in the image which have been detected by our pixel-wise criteria. Clearly, this makes the hypothesis that the number of detected pixels which correspond to true scratches is small in comparison to those which correspond to detections in the background model. The result of such an approach may be seen in Figure 3.15. Evidently, the global estimation of the background noise is not satisfactory. This is because the pixel-wise detection step produces an amount of detections which varies greatly across the image. In particular, the value of p is greatly underestimated in textured or cluttered areas, yielding many more detections than in smooth regions. In fact, this is exactly the same problem experienced in Section 3.3.2, where the areas with many detections due to noise and clutter were extremely salient in the Hough transform. Clearly we need to deal with the estimation of p in a more local manner.

Locally adaptive grouping We wish to estimate the parameter p locally. This would yield a background model in which each X_i is associated with its own detection probability p_i. In order to use such a method we need to address a technical issue which arises with such a locally varying background model.

Under this background model, the probability for a given segment to have at least k_0 pixels with a label value of 1 is no longer given by a binomial distribution. It is now given by a *Poisson binomial distribution* and is equal to

$$\Pr(S_l \geq k_0) = \sum_{k=k_0}^{l} \sum_{\substack{x_1,\dots,x_l \\ \sum x_i = k}} \prod_{i=1}^{l} p_i^{x_i} (1-p_i)^{x_i}, \tag{3.12}$$

where $p_1, \dots, p_l$ are the local detection probability at each pixel, the definition of which will be given below. This expression is quite costly to estimate and an approximation is therefore needed. In [46], Desolneux *et al.* suggest the use of Hoeffding's inequality [65], which provides an upper bound on the probability that the sum of some independently distributed random variables exceeds a certain value. In the present case, the interest of this approximation is that it still holds when the variables are *independent but not identically distributed* [65]. Therefore, it provides us with an approximation of $\Pr(S_l \geq k_0)$, where again S_l is the number of pixels having a label value of 1 along a segment of length l. The approximation is the following :

$$\Pr(S_l \geq k_0) \leq H(l, k_0) := e^{-l\left(r \log \frac{r}{\langle p \rangle} + (1-r) \log \frac{1-r}{1-\langle p \rangle}\right)}, \tag{3.13}$$

where $\langle p \rangle = l^{-1} \sum p_i$ is the average detection probability along the segment, $r = \frac{k_0}{l}$, and $\langle p \rangle l < k_0 < l$. We therefore define the number of false alarms of a segment as

$$\mathrm{NFA}(l, k_0) = N_{tests} H(l, k_0). \tag{3.14}$$

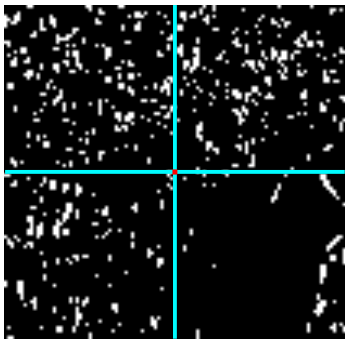


FIGURE 3.11: Method of background noise estimation. The noise density considered at the central pixel (in red) is that of the sub-square with the highest noise density.

We recall (Definition 1) that a segment is ***detected*** if its NFA is ***smaller*** than ε, and such a segment is said to be "ε-meaningful".

Now that we have laid down the theoretical basis for a locally varying background model, we must estimate the local background noise. As in the case of the global background estimation, we shall do the estimation empirically. The label probability for a given pixel is estimated as the maximum detection density on four squares of equal size surrounding the pixel. Four squares are used in order to deal with situations where the pixel is on the border of areas with different noise densities. An illustration of this method may be seen in Figure 3.11. The size of the sides of these squares is set to the width of the image, divided by a constant, which we set to 30.

Let us make a few remarks on this methodology choice. First of all, it is quite clear that the estimation is biased due to the presence of true scratches. Unfortunately, since we do not know ahead of time which pixels belong to true scratches and which correspond to background noise, it is difficult to avoid this bias. One solution would be to detect the meaningful segments, remove the pixels belonging to them and reiterate the algorithm. Another alternative would be to estimate the noise level using estimators which are robust to the outliers (true scratch pixels). However we do not use such sophisticated methods since we have good results with the proposed algorithm. Another criticism that could be made about this approach is that we perform no explicit segmentation of the noise image. This could be done by using tools such as texture segmentation [108]. Again, we did not have to resort to such measures due to the satisfactory performance of the presented approach. Furthermore, it is not certain that using a strict segmentation would perform better than the proposed method, which allows a smoother spatial variation in the value of p.

This local noise estimation approach is certainly the most delicate step of our algorithm, the rest being either extensively experimentally validated (for the pixel-wise detection criterion) or based on solid theoretical foundations (the case of the *a contrario* framework). Although this step could be pushed further, we shall keep the methodology as presented. We can give an intuition as to why this approach should work so well. We recall that the presence of true scratches will bias the noise estimation towards *over-estimation*. This over-estimation is exacerbated by the fact that we take the *maximum* of four estimations of the background noise density. In spite of this, we are clearly able to pick up the true scratches and avoid false detections in noisy areas. Therefore, it would appear that in practice the scratches are sufficiently present in the image that the *a contrario* method is able to identify them even when over-estimating the parameter p. If this is indeed the case, it contributes greatly method's success, and is therefore an important point to keep in mind when considering possible extensions of this work.

We now continue the presentation of our algorithm. In all experiments, we use the parameter $\varepsilon = 1$, as in [45]. This choice is reasonable, since ε is a bound on the expected number of false detections under the background model. However, it may be further tuned to fit the user's needs, depending on whether the importance should be put on recall or precision. Furthermore, as explained in [45], detection results vary slowly with respect to ε. Since scratches are roughly vertical, we test all segments with a maximum deviation from the vertical direction of ± 10 degrees. We discretise these angles by 0.5 degrees, which corresponds approximately to a deviation of 1 horizontal pixel for every 115 vertical pixels, yielding a precise representation of the line scratches. The N_{tests} parameter is set to $M^2 N \Theta$, where Θ is the number of angles tested. With the aforementioned parameters, we have $\Theta = 40$. This formula for N_{tests} changes from the initial formulation (which was $M^2 N^2$). However since we have a very fine angle discretisation, it seems logical to include it in the number of tests.

3.3.4 Maximality and exclusion

We obviously wish take advantage of the very useful maximality and exclusion principles found in [46] and include them in our algorithm.

It was shown in [46] that maximal meaningful segments begin with a 0-1 transition, that is to say a detected pixel preceded by a non-detected pixel, and end with a 1-0 transition. This is of great practical use, since it implies that many segments in the image need not be tested. This result relies on the original definition of the NFA (3.6) and the properties of the binomial law. With the proposed definition of the NFA (3.14), this property also remains true, as we shall now prove.

Proof that two properties of the maximality principle hold true when Hoeffding's approximation of the Binomial law is used. We wish to prove two properties of the meaningful segments defined using the NFA relying on Hoeffding's approximation, Equation (3.14). We recall these properties :

- If one appends a 0 (non-detected pixel) to the segment, its meaningfulness decreases (its NFA increases)
- If one appends a 1 (detected pixel) to the segment, its meaningfulness increases (its NFA decreases)

Using Equation (3.14), these properties reduce to :

$$H[l,k] \leq H[l+1,k], \tag{3.15}$$

and

$$H[l,k] \geq H[l+1,k+1], \tag{3.16}$$

where H is defined as

$$H(l,k) := \exp(-k \log \frac{k}{lp} - (l-k) \log \frac{l-k}{l(1-p)}), \tag{3.17}$$

where $lp < k < l$. Since the exponential function is strictly increasing, we need to study the following function :

$$f(k,l) = -k \log \frac{k}{lp} - (l-k) \log \frac{l-k}{l(1-p)}. \tag{3.18}$$

Now, let us prove Equations (3.15) and (3.16).

For Equation (3.15) to be true, we need the partial derivative of f with respect to l to be positive :

$$\frac{\partial f(k,l)}{\partial l} = k\frac{1}{l} - [\log\frac{l-k}{l(1-p)} + (l-k)\frac{\partial}{\partial l}\log\frac{l-k}{l(1-p)}]. \tag{3.19}$$

We have the partial result :

$$\frac{\partial}{\partial l}\log\frac{l-k}{l(1-p)} = \frac{k}{l(l-k)}. \tag{3.20}$$

Therefore,

$$\begin{aligned}\frac{\partial f(k,l)}{\partial l} &= \frac{k}{l} - \log\frac{l-k}{l(1-p)} - (l-k)\frac{k}{l(l-k)} \\ &= \log\frac{l-lp}{l-k}.\end{aligned} \tag{3.21}$$

We know that $l - lp > l - k$, because $lp < k$ (the condition for the Hoeffding approximation to hold true). Therefore, the right hand term of Equation (3.19) is strictly positive, so that $f(k,l)$ increases strictly with l. This means that when a 0 is appended to a segment, its meaningfulness decreases (since its probability increases). This shows the first inequality (Equation (3.15)) in the case of Hoeffding's approximation.

Now we prove Equation (3.16). This case is slightly more complicated, since two variables (k and l) vary at the same time. However, since we add the same quantity to both these variables, it is enough to study the partial derivative of $f(k+t, l+t)$ with respect to t.

We have

$$f(k+t,l+t) = -(k+t)\log\frac{k+t}{(l+t)p} - (l-k)\log\frac{l-k}{(l+t)(1-p)}, \tag{3.22}$$

so that :

$$\begin{aligned}\frac{\partial f(k+t,l+t)}{\partial t} &= -\log\frac{k+t}{(l+t)p} - (k+t)\frac{\partial}{\partial t}\log\frac{k+t}{(l+t)p} \\ &- (l-k)\frac{\partial}{\partial t}\log\frac{l-k}{(l+t)(1-p)}.\end{aligned} \tag{3.23}$$

Now

$$\frac{\partial}{\partial t}\log\frac{k+t}{(l+t)p} = \frac{l-k}{(k+t)(l+t)},$$

and

$$\frac{\partial}{\partial t}\log\frac{l-k}{(l+t)(1-p)} = -\frac{1}{l+t},$$

so that

$$\begin{aligned}\frac{\partial f(k+t,l+t)}{\partial t} &= -\log\frac{k+t}{(l+t)p} - (k+t)\frac{l-k}{(k+t)(l+t)} \\ &+ (l-k)\frac{1}{l+t} \\ &= \log\frac{lp+tp}{k+t}.\end{aligned}$$

This quantity is strictly negative, since $lp < k$ and $tp < t$. Therefore, $f(k, l)$ decreases when k and l increase from the same quantity. We have in particular, that $H(l + 1, k + 1) < H(l, k)$, meaning that meaningfulness increases if a 1 is appended to a segment. Thus, the second inequality (Equation 3.16) holds true.

We have proven the two necessary properties for a segment to be maximal meaningful in the case of Hoeffding's approximation, thus we can safely prune the search for maximal meaningful segments. This yields the same crucial acceleration of the method as in the original *a contrario* approach.

One special case concerning maximality worth noting was brought up in our work. This is the case where the maximality principle "over-eliminates" segments. Consider three segments, A, B and C, with $B \subset A$ and $C \subset A$, and assume that $NFA(B) < NFA(A) < NFA(C)$. In this situation, the segment A eliminates C, but is itself eliminated by segment B. A similar fact was noted in [46] for the segmentation of histograms. An illustration of the situation can be seen in Figure 3.12. The two potential scratches are quite far apart, and should be detected separately. However, the scratch situated below the square object on the left hand side of the image is covered by a long detection grouping both the detection above and below the object. This long scratch is then itself eliminated, and therefore so is the lower detection.

To try and address this problem, we reformulated the idea of maximality in the form of a *reduced* maximality. This principle would simply state that a segment is maximal meaningful if *it does not contain any more meaningful segments*. This would allow the algorithm to detect such segments as presented in Figure 3.12 (b).

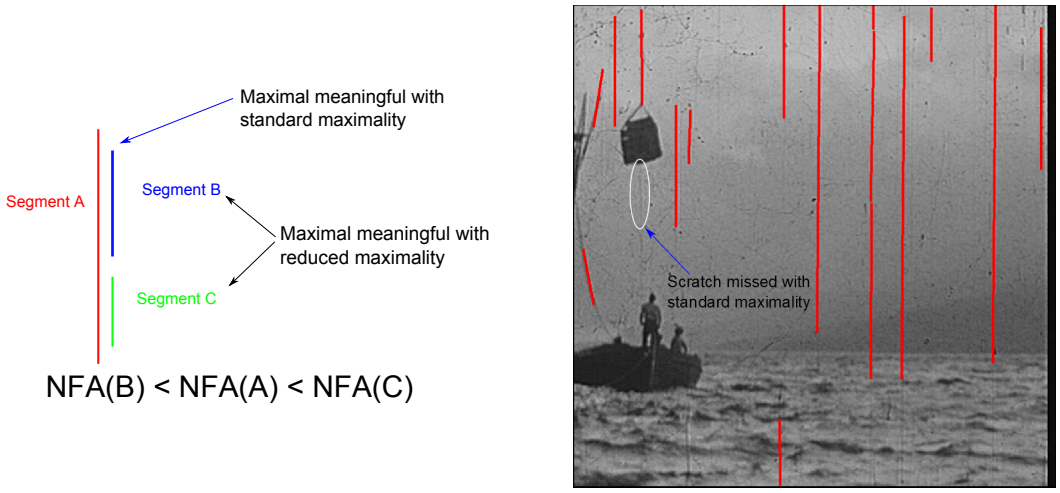
In spite of the over-elimination observed with the regular principle we have not used the modified, reduced maximality principle, because this principle does not in fact achieve our goal in a rigorous manner. Consider for example the segment $A = [1 \cdots 1 \cdots 1]$. With the reduced definition, *each and every* segment in A is maximal meaningful, since none of them contain a more significant segment. Therefore, we would need to modify our reduced maximality principle for it to be applicable. However, this is obviously not visible in the result presented in Figure 3.12, since the sub-segments do not appear separately from the "containing" segment. Also, in practice, this phenomenon did not appear enough to merit re-developing this part of the *a contraio* theory.

Let us briefly recall the exclusion principle as it is expressed in Definition 3 : a pixel may belong to one and only one segment (or *grouping*). We apply the exclusion principle directly here. The only parameter linked with the exclusion principle is the dilation size, which we refer to as τ_x. We have set this to three pixels on both sides of the scratch, guaranteeing a precise and unique description of the line scratches. This removes repeated detections along structures which are several pixels thick.

An illustration of the improvement that may be obtained with the maximality and exclusion principles may be seen in Figure 3.13. We note that these examples appear coherent with the illustrations which may be found in [46]. It is clear that without the maximality principle in particular, the scratch detection would be quite unsatisfactory. As it is, this principle is extremely useful and powerful especially in cases such as line scratch detection, where a precise description of the defect is required.

3.3.5 Algorithm speed-up and minimum scratch length

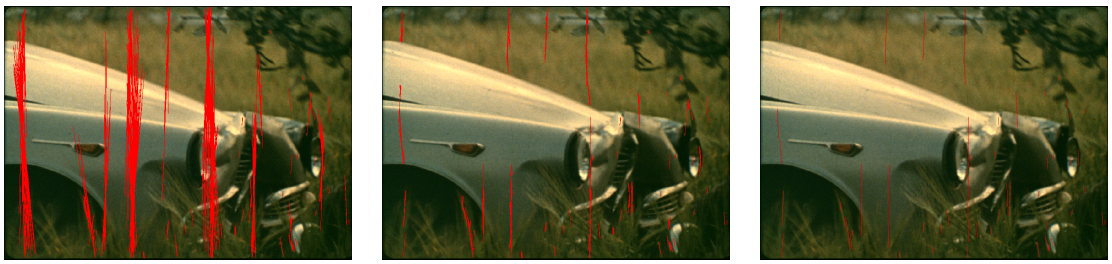
In order to speed up the procedure, we apply a pre-selection of scratch candidates. For this, we apply a very permissive Hough transform to I_B, and only analyse the lines which correspond to local maxima. Tests have shown that no or very few real scratches are lost by this pre-processing.



(a) Schematic illustration of a case where the maximality principle "over-eliminates" scratch detections

(b) Example of the "over-elimination" effect in a real image

FIGURE 3.12: An illustration of a drawback of the maximality principle. The theoretical situation shown in (a) is produced in (b).



(a) ε-meaningful segments

(b) Maximum meaningful segments

(c) Maximum meaningful segments with the exclusion principle applied

FIGURE 3.13: An illustration of the importance of the maximality and exclusion principles. Without the maximality principle, in particular, the line scratch detection is much worse.

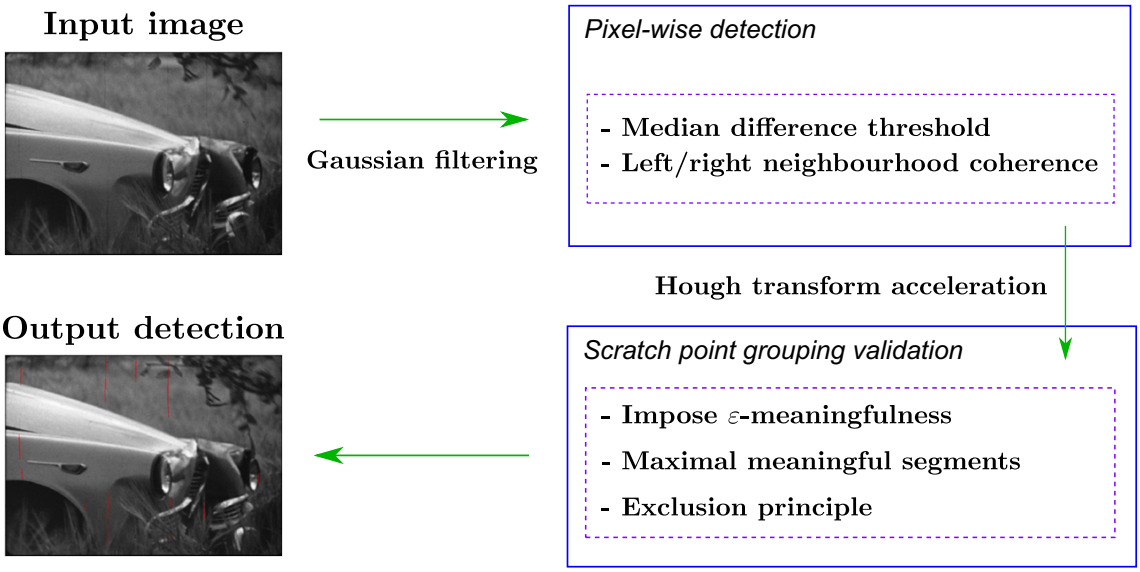


FIGURE 3.14: Summary diagram of the proposed spatial detection algorithm.

In practice, we consider all the local minima whose presence in the transformed domain is greater than the average number of votes of all the accumulators.

Finally, we impose a minimum scratch length of one tenth of the height of the image. This criterion is relatively pragmatic and reasonable, and can be imposed with very little risk of missing persistent scratches. It should be noted, however, that this criterion should be imposed *before* carrying out the maximality step. Otherwise we could be in danger of rejecting meaningful scratches which are larger than the minimum size in preference for smaller scratches which will consequently be removed.

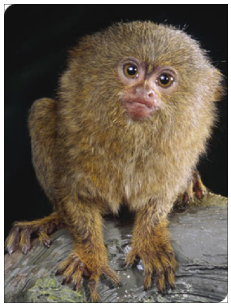
The entire spatial line scratch detection algorithm is represented in Figure 3.14.

3.4 Visual results and discussion

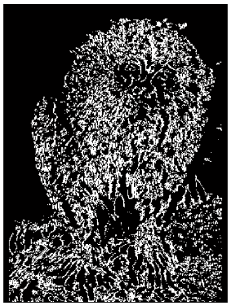
We shall now present some visual results of our line scratch detector. These results can be seen in Figure 3.18. We observe that the proposed algorithm is able to detect the scratches with a high degree of accuracy. This is largely due to the maximality principle of the *a contrario* methodology. In particular, the algorithm is able to identify weakly contrasted scratches on backgrounds with high degrees of texture and film grain (example of "Les Choses de la Vie").

One of the main goals of our our algorithm is to avoid detection in noisy/textured/cluttered areas. Some illustrations of the aspect are shown in Figure 3.15.

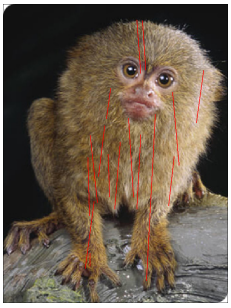
We also show an example of scratch detection in high definition images, in Figure 3.16 (1920×1080 pixels). High definition images represent an interesting case, since the restoration of old films will most likely be with such resolutions. It can be seen that the algorithm from [25] is unable to detect the faint, white scratches present on the right hand side of the image, whereas the proposed method locates them with a high degree of spatial precision. In the example in Figure 3.16, it may be seen that certain false alarms are present which are not due to vertical scene structures. This is likely due to the preliminary Gaussian filtering stage of our spatial detection algorithm. This filtering may not be sufficient for dealing with noise, since the standard deviation is fixed. Setting this parameter adaptively could help performance.



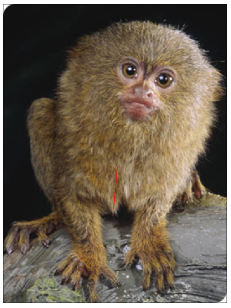
(a) Original frame



(b) Binary detection

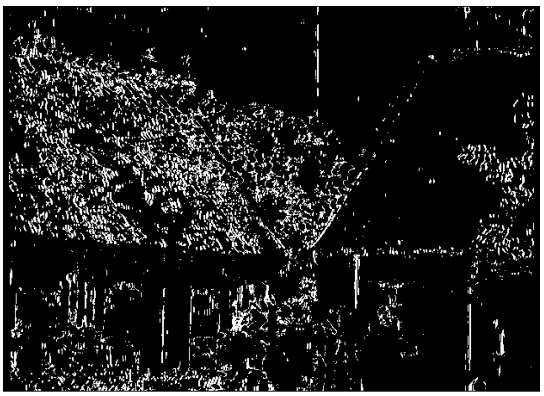


(c) *A contrario* detection with global noise estimation

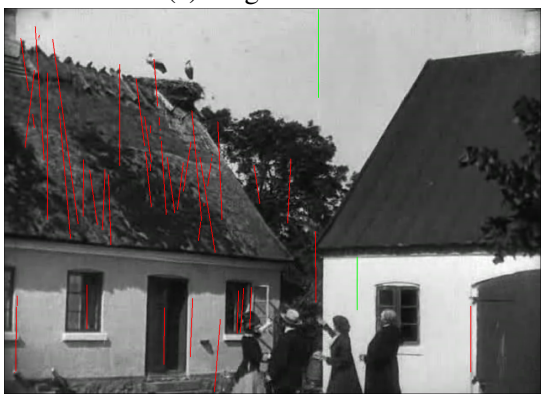


(d) *A contrario* detection with local noise estimation

(e) Original frame



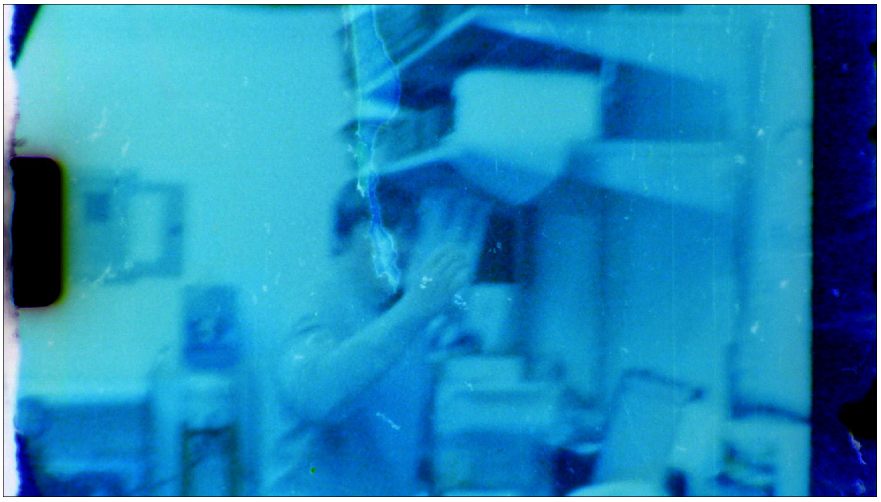
(f) Binary detection image



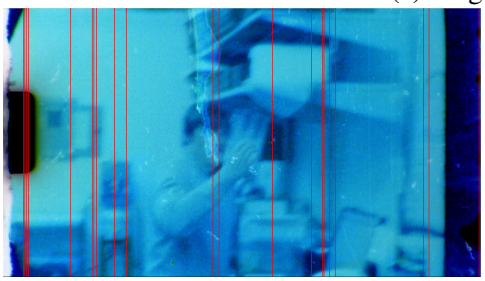
(g) *A contrario* detection with global noise estimation

(h) *A contrario* detection with local noise estimation

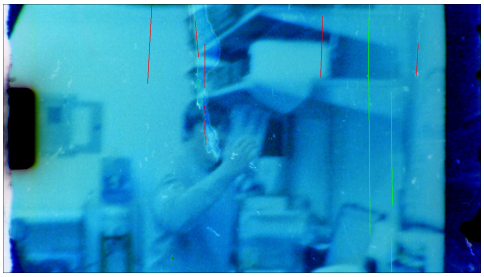
FIGURE 3.15: An illustration of our spatial method's robustness to texture and noise on two highly textured test images. In the first example (which contains no line scratches), our locally adaptive method produces only one false detection, whereas with a global noise estimation, the algorithm fails. The second example illustrates this principle on an image with real scratches. True scratches are in green and false detections are in red.



(a) Original frame



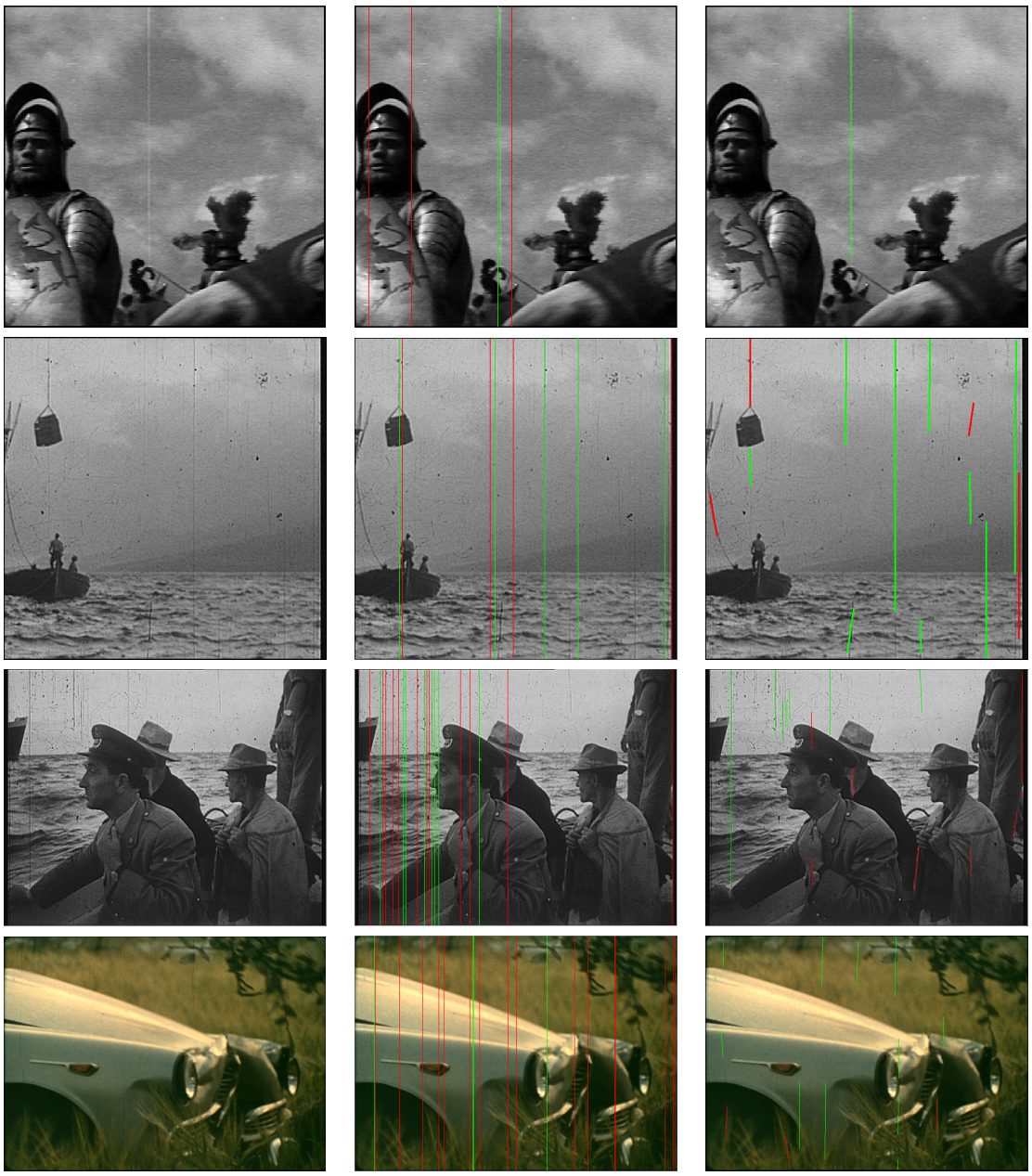
(b) Detection from [25]



(c) Our spatial detection

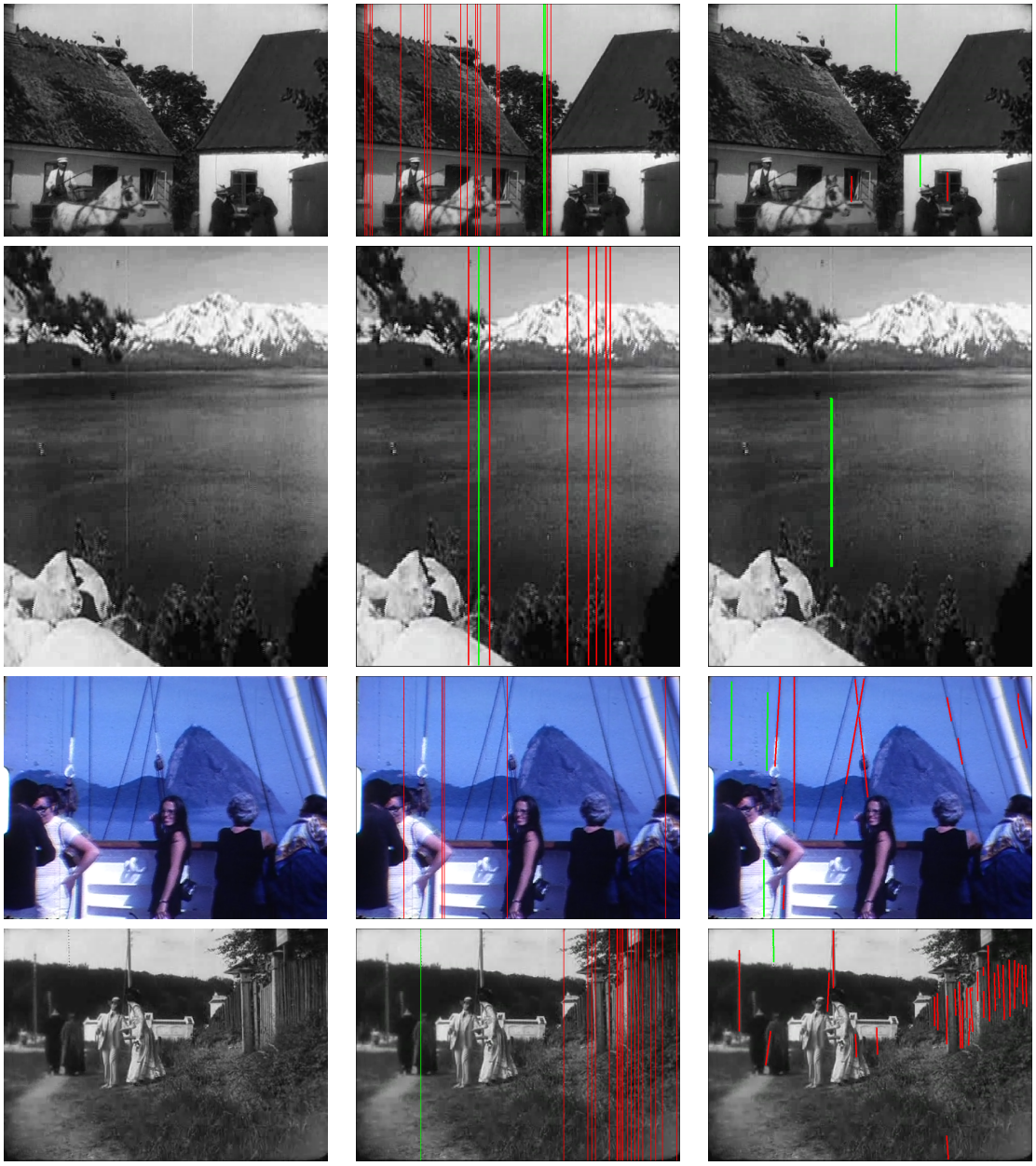
FIGURE 3.16: A high definition scratched film (1074x1920). Correct detections are shown in green, false alarms in red.

One drawback of our approach with respect to the algorithm of Bruni *et al*. is increased computation times. If we evaluate the complexity in a naïve manner, the complexity of our algorithm is $\mathcal{O}(M^2 N\Theta)$ (the number of tested segments). However, as we have mentioned, this is greatly reduced by the properties of the maximality principle. The speedup given by this is difficult to evaluate precisely, since it depends on the amount of "holes" in the binary image. This may be verified experimentally : the sequences such as "Sitdown", "Star" and "Les Choses de la Vie" contain much background noise and film grain, leading to many detections. As such, these sequences take much longer than the others to process (up to 106 seconds for one image of 512×512 pixels). It is quite possible that our discretisation of the angles determining the lines which are tested is too fine for our purposes and could be modified. As it is set in our experiments (0.5 degrees) it represents a deviation of 1 horizontal pixel for every 115 vertical pixels. However, no attempt was made to accelerate the algorithm beyond writing specific mex functions for the *a contrario* steps. The rest of the code was written in Matlab. In a finished product, each frame would most likely be processed in parallel, greatly accelerating the processing. Alternatively, the processing of each line could be done in parallel. In fact, the majority of the algorithm can be parallelised. Furthermore, the acceleration techniques used by von Gioi *et al*. for the Line Segment Detector presented in [120] could be adapted to our approach. Indeed, the aligned gradients could be simply replaced with binary scratch pixel detections.



(a) Original frame (b) Detection from [25] (c) Our spatial detection

FIGURE 3.17: From top to bottom : “Knight”, “Sitdown”, “Star” and “Les Choses de la Vie”. Correct detections are shown in green, false alarms in red.



(a) Original frame (b) Detection from [25] (c) Our spatial detection

FIGURE 3.18: From top to bottom : "Afgrunden horse", "California", "Keldjian" and "Gate". Correct detections are shown in green, false alarms in red.

3.5 Conclusion on the spatial line scratch detection algorithm

In this Chapter, we have considered the problem of line scratch detection in still frames. We have proposed a new algorithm which takes advantage of the various strong points of the visual object grouping methodology known as the *a contrario* method. The proposed algorithm consists of a pixel-by-pixel scratch detection step, followed by the grouping and validation of these de-

tections into visually significant scratch segments. The pixel-wise detection is based on a model whose parameters have been tuned to fit the scratches commonly found in videos of modest resolution (roughly 700×500 pixels). It is possible that to deal with films scanned at higher resolutions, this step could be adapted, in particular with respect to the line scratch width. The second part of the algorithm, employs the *a contrario* methodology, and therefore offers a generic and automatic approach to setting detection parameters. Furthermore, we have proposed a modification of the classical approach, and introduced a *local estimation* of noise models, which allows for grouping under spatially varying conditions of noise and clutter. In the present case, this local estimation is vital for the success of the detection algorithm. An interesting question is whether this local estimation could be extended to other problems.

Chapitre 4

Temporal Line Scratch Detection

4.1 Introduction

In the previous chapter we described an algorithm which is able to detect most line scratches. However, along with these line scratches, many false detections are produced which correspond to thin vertical structures which are part of the scene. In still images it is extremely difficult to separate these false detections from true scratches. A common solution to this problem is to observe the behaviour of line scratch detections over the whole image sequence. The temporal behaviour of false detections and true scratches is often quite different, and so it becomes possible to tell them apart. We refer to this process as *temporal filtering*.

In this Chapter, we present our work concerning the temporal filtering of line scratches. To do this, we formulate some very general temporal hypotheses which we shall use for the purpose of distinguishing false alarms from true scratches. After this, we present two temporal filtering approaches which we have explored. The first approach attempts to track both false alarms and true scratches. The second considers the more constrained problem of tracking false alarms which are due to background or camera motion. In the final line scratch detection algorithm which we propose, we retained the second approach due to its superior robustness. We discuss the shortcomings of the first approach, and present visual results of the second. First of all, let us present the literature concerning the temporal filtering of line scratch detections.

4.2 Prior work

Relatively little research has been done concerning the temporal aspects of line scratch detection. Several papers by Joyeux *et al*. [70, 71, 72] used temporal information available concerning scratches, to separate false detections from true ones. They hypothesise that true line scratches have *sinusoidal* motion. This hypothesis stems from the mechanical origin of the line scratch, that is to say the rubbing of a mechanical part against the film. Their method then uses the Kalman filter to *track* scratches, which are then rejected or accepted depending on whether they conform to the scratch motion model. The first major drawback of the method is that this hypothesis is not always verified. Also, even if the hypothesis is verified, the scratch may not be present on the film for long enough for the sinusoidal motion to become clear. The method also suffers from several other practical problems. For example, the input to the filter is the closest detection to the predicted state, something which may be dangerous in the presence of several trajectories and noise. Also, it is not clear in the authors' work how the trajectories are accepted or rejected after the tracking stage. In the thesis of Joyeux [69], it is briefly mentioned that the trajectories are filtered based on

their length and horizontal speed, but a specific methodology is not given. Even considering that the specifics are given, the method suffers from the fact that in many situations, a trajectory may only be partially rejected. For instance, if a false detection moves up to a certain point in time and is then still, only the first part will be eliminated. This can happen very often in certain sequences.

In [16], Besserer and Thiré proposed a method based on Multiple Hypothesis Tracking. They create a trellis on which the path of a scratch may be defined. The initial frame-by-frame scratch detections are produced by a projection onto the x-axis, followed by a grey-scale morphological filtering step. This method is interesting in that it provides a "quality" measure $Q(x,t)$ during the detection step, which is zero for non-detected columns and positive otherwise. This quality is dependant on the amplitude of the projection of the image onto the x-axis. A scratch is then tracked by generating several potential trajectories over a limited time span, and keeping only a certain number. The retained trajectories are those with the smallest "cost". The cost is dependant on the quality measure of each detected scratch and also the difference between the predicted position of the scratch and an observed position. The prediction is based on a polynomial motion model. This work is interesting in that it allows for several possible trajectories to be held in memory simultaneously, and to choose among them. However, this method has several limitations. Firstly, as in the case of the work of Joyeux *et al.* , the motion model seems to be quite restrictive. In fact, it contradicts the sinusoidal hypothesis of the latter authors. Another questionable choice is that the authors suppose that the scratch paths have a maximum horizontal motion of one pixel per frame. Also, both the creation and termination of scratch trajectories include parameters which are not specified. For instance, a track is initialised if an "unmatched, isolated and relevant" scratch detection is present. The term relevant is left unspecified, although it could simply refer to a scratch detection. A trajectory is no longer followed if the quality $Q(x,t)$ at the current state of the filter falls below a threshold. This threshold is left unspecified. Finally, as in the work of Joyeux *et al.*, it is unclear exactly how the algorithm distinguishes between true scratches and false detections. It is possible that the algorithm is intended to be semi-automatic, and the only goal is to track the scratches so that a human can determine whether each track corresponds to a true scratch or a false alarm. It is also possible that *all* the detected trajectories are considered to be true scratches. However, if this is the case then the algorithm will fail in the same cases as that of Joyeux *et al.* , that is to say when only a portion of a trajectory moves in a manner which is not coherent with their model.

Another method proposed by Güllü *et al.* which exploits temporal information is [74], where the authors use the local block matching error to determine if a detection is a true scratch or a vertical structure. Each scratch is followed until the matching error exceeds a certain value, at which point all the previous detections are validated as true scratches. This is an interesting idea, but unfortunately the task of tracking the scratches is not dealt with. This is in actual fact the most difficult task of temporal scratch filtering algorithms, and so one can readily imagine that the algorithm may not be very robust. Also, the block matching error is quite sensitive to contrast changes induced by phenomena such as flickering. Finally, many of the parameters are not given, which makes accurate comparisons with the method difficult, even if the algorithm itself is easy enough to re-implement.

Another method which seeks to separate true scratches from false alarms is that of Müller *et al.* [95]. This method tries to eliminate false alarms by analysing the scratch detections' motion and comparing it to the local motion of the underlying scene. The hypothesis is that any scratch detection which moves with the scene must be a thin vertical structure belonging to this scene. This is a very interesting hypothesis, and seems somewhat more usable and robust than the previous sinusoidal one. We shall also be working with similar hypotheses in our work. Unfortunately,

the technique used by Müller *et al*. to estimate the scratches motion is too naïve for the method to perform well. They use block matching on the centre of the detected scratch to estimate its motion. This is clearly not very robust, since scratches are not visually characteristic enough to be tracked in this manner ; there is nothing for the block matching to "recognise". Therefore, it is probable that the motion estimated at the centre of the scratch will be the motion of the underlying scene itself, in which case *all* scratch detections are likely to be rejected.

Having looked at the previous work, it is relatively clear that the core challenge of temporal line scratch filtering remains the *tracking* of the scratches and determining reliable trajectories. Generally speaking, the previous methods imply that two algorithmic steps are used, a first step to determine scratch detection trajectories, and a second to decide whether the trajectories conform or not to the hypotheses set out for true and/or false scratches. It is quite interesting to note that no previous algorithm has looked into both steps in sufficient detail. The work of Joyeux *et al*. and Besserer and Thiré concentrate on tracking the scratch detections, but give very little detail on how they distinguish between a true and a false detection, and the work of Güllü *et al*. and Müller *et al*. provide reasonable hypotheses to distinguish true and false detections, but give almost no indications concerning how the trajectories are determined.

Before exploring the tracking step in greater depth ourselves, we first discuss and set out the hypotheses which we will use to distinguish true scratches from false alarms.

4.3 Distinguishing between true scratches and false detections : motion coherency hypotheses

In the previous methods, the hypotheses used for scratch filtering are always temporal in nature, and are usually coupled with some type of *motion analysis*. Throughout this work, two such temporal hypotheses will structure our approaches. They are as follows :

Hypothesis 1. *We consider that a scratch detection which moves* with *the underlying scene corresponds to a* false alarm.

Obviously the case of a scratch moving with the scene is possible, however it is extremely unlikely that a scratch should move in accordance with the underlying scene for the whole time it is present on the film.

Hypothesis 2. *We consider that if a scratch moves significantly* contrary *to the scene motion, then it is considered to be a true scratch.*

Again, exceptions are possible, since we may have scene elements such as ropes whose motion can be completely independent of the scene. However, it is difficult to find more generally applicable hypotheses, and it must be said that the aforementioned cases are very difficult, requiring almost certainly human intervention.0 Therefore, we shall keep these hypotheses throughout our work. We shall refer to these hypotheses as *motion coherency* hypotheses. Given these suppositions we would like, ideally, to be able to track both false alarms and true scratches so as to be able to actively confirm certain detections as true scratches and reject others as detections of scene elements.

In order to do this, two important elements are needed. Firstly, we require a method for identifying the trajectories of scratch detections, in order to compare their motion to that of the scene. Secondly, we obviously need some way of estimating the scene motion. The estimation of the

scene motion can be done by well-known and robust methods, such as the parametric motion estimation algorithm of Odobez and Bouthemy [97]. We will go into further detail concerning this step further on. We underline that the estimation of scene motion will not be the subject of any specific research in this work, however we briefly present some of the main motion estimation methods in Appendix A.

Before continuing, let us mention another type of approach which could possibly be taken, which is to use *semantic* information to distinguish true scratches from false alarms. A human may do this by deciding that a line which goes across several objects is probably a scratch. Unfortunately, this is difficult to do, and is most likely prone to many exceptions. For example, in the case of a picture taken through a window with a fissure in the glass, this hypothesis would not hold. So although scene analysis would have the advantage of dealing with scratches frame by frame, realistically it is not possible to deal with the problem in this manner. However, we make a note of the approach since it has not been mentioned in the literature.

4.3.1 Notation

Throughout the following presentation of the different methods of determining scratch trajectories which were explored, we have used the spatio-temporal scratch detection representation found in the work of Joyeux *et al* [71]. In this representation, a scratch detection is characterised by its temporal position (the frame it is detected in), and its column coordinate. If a detection is described by more than one column coordinate (the case of a slanted scratch), then we take the average column of the detection as its spatial position. This representation leads naturally to a binary "image", which we denote I_T, where each non-zero pixel represents at least one scratch detection. The rows of I_T represent frame indices, and the columns represent the horizontal axis of each image. It is possible for a pixel in I_T to correspond to more than one scratch detection, since the scratches are projected onto the x-axis. For instance, if a detected element (scratch or scene element) in a given frame is split up into several segments by our *a contrario* scratch detector, then it is possible that the segments will correspond to the same pixel in I_T.

The resulting representation I_T, which we refer to as the "temporal profile", can be seen in figure 4.1. We will be tracking scratch detections using this representation. It is possible to add the y coordinates of the scratch detections, and include them as information for tracking, but we do not choose to do that here.

We shall refer to a scratch detection provided by a spatial detection algorithm as a scratch *segment*, denoted by S. Let $\tilde{x}(S)$ and $\tilde{y}(S)$ be, respectively, the average column and row indices of the scratch. Let $t(S)$ represent the frame in which the segment was detected.

4.4 First approach : determining generic scratch detection trajectories

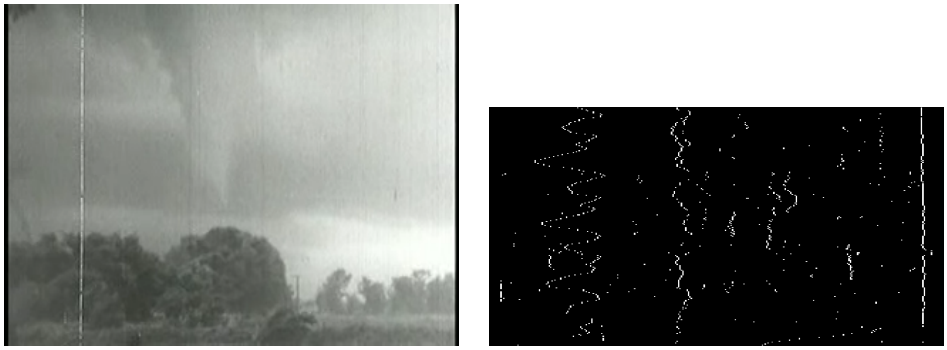
In this Section, we present the first approaches to scratch detection tracking which we have explored. It should be underlined that the methods presented in this Section were not used in our final temporal filtering algorithm. For readers who wish to view the algorithm which we finally retained, please see Section 4.5.

The goal in this Section is to track *all* scratch detections, not just true scratches (as is done in the work of Joyeux *et al*.).

In order to track the scratch detections, we require some hypotheses concerning the motion of any given scratch detection. We decided not to work with the same hypothesis as Joyeux *et al*. (the

(a) Original frame ("Wathenco") (b) Temporal profile (I_T)

FIGURE 4.1: First frame (a), and temporal profile (b) of the spatial scratch detections from the "Wathenco" sequence. Each white point of the temporal profile represents at least one scratch detection. The x-axis of this profile represents the average column index of the line scratch, and the y-axis represents the index of the frame in which the scratch detection was made.



(a) Original frame ("Tornado") (b) Temporal profile I_T

FIGURE 4.2: First frame, and temporal profile of the scratches from the "Tornado" sequence. All of the detections are true scratches in this sequence. Notice that while two scratches seem to have sinusoidal motion (one is motionless, and one has a sinusoidal motion with a non-zero amplitude) the others can not be said to display such motion.

sinusoidal motion hypothesis), since it seemed to be somewhat too restrictive. Indeed, even if examples of such motion are definitely present in old films, it is less clear whether they represent a large enough percentage of the cases to lead to a robust algorithm. Therefore, we shall try to find some other hypotheses for the task of tracking scratch detections. For an interesting example of various scratch detection motions, see Figure 4.2.

4.4.1 Scratch motion model

Having looked at many different examples of scratch detection trajectories (both true scratches and false alarms), and tried to formulate motion hypotheses which are as generic and reasonable as possible. We retained two very generic criteria :

- Scratch detections have *continuous* motion.
- Scratch detections have *smooth* motion.

Intuitively speaking, these hypotheses mean two things. Firstly, that a scratch detection may not jump very far (spatially) from one frame to the next. Otherwise, we shall just consider them to be in the same category as impulsive defects (*e.g.* blotches and dirt). Secondly, if we have to choose between several paths for a scratch detection (again true or false), we shall choose the smoothest one. These hypotheses are justified by the fact that true scratches are caused by the rotation of the film in the projector, whose motion may be considered smooth. Also, a vertical scene object will also move smoothly, if the sampling rate of the video is sufficient. These seem to be the most generic hypotheses available, especially since the motion of line scratches, in particular, can be so variable.

To recap, these two hypotheses will be crucial for us in the process of *tracking* the detections. We underline here that the continuity and smoothness hypotheses are not the same as the hypotheses which distinguish a true scratch from a false alarm (the motion coherency hypotheses). The continuity and smoothness criteria simply allow us to *track* the scratch detections. Once we have tracked the scratch detections, we will use the motion coherence hypotheses to reject or accept different trajectories.

We must now decide on a formal method in which to incorporate our tracking hypotheses.

4.4.2 First tracking approach : iterative filters

Arguably the most famous tracking method is the *Kalman filter*. This is an iterative filter which estimates the *state* of a system given the knowledge of the system's dynamics, a measurement of the state, a user input and a noise model for both the system state and the measurement. The state may be any property of the system, so for example for a car it would likely be the position, speed and acceleration of the vehicle. The noise model represents the fact that we consider our measurement of the state of the system to be imperfect. More formally, if X represents the state vector, U is the input, Z is the measurement vector, W is the system noise and V is the measurement noise, then we have :

$$\begin{aligned} X_{k+1} &= A_k X_k + B_k U_k + G_k W_k, \qquad W_k \sim \mathcal{N}(0, Q_k) \\ Z_k &= C_k X_k + V_k. \qquad V_k \sim \mathcal{N}(0, R_k) \end{aligned} \tag{4.1}$$

(4.2)

The matrices A_k, B_k, G_k, C_k, Q_k and R_k are considered to be known. This is obviously a very generic and flexible formulation of the state estimation problem, but we may make many simplifications in our case. For the moment, we shall consider the state vector to contain just the average x position of a scratch. We may dispense with the control input U_k. Furthermore, our measurement is only noisy, and has not been put through some sort of filter. Therefore, we may rewrite Equation 4.2 as :

$$\begin{aligned} X_{k+1} &= A_k X_k + W_k, \qquad W_k \sim \mathcal{N}(0, Q_k) \\ Z_k &= C_k X_k + V_k. \qquad V_k \sim \mathcal{N}(0, R_k). \end{aligned} \tag{4.3}$$

(4.4)

The state and measurement are then estimated using two phases : prediction and update. For the Kalman filter to work, we need to have some sort of knowledge concerning the system's dynamics.

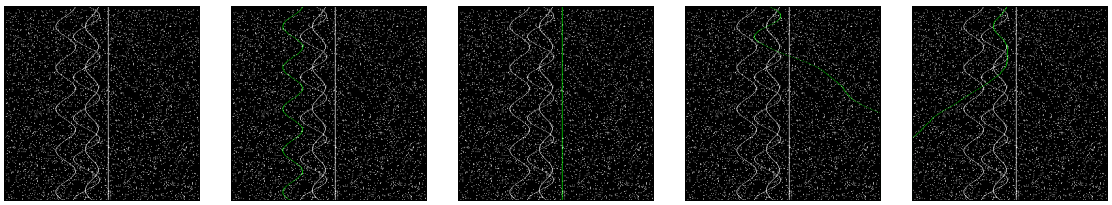


FIGURE 4.3: Illustration of tracking synthetic trajectories using the PDAF tracker. The image on the left represents the initial detections. These detections consist of true trajectories, and randomly detected spatio-temporal positions. The other images show different tracked trajectories in green. We see that once a track has been lost, it is not found again.

In the case of Joyeux *et al*. this is a sinusoidal motion hypothesis. We have set out other hypotheses, in particular the *continuity* and *smoothness* criteria.

An extension of the Kalman filter is the Probabilistic Data Assocation Filter (PDAF) of Kirubarajan and Bar-Shalom [10]. This accounts for the presence of *several* measurements (of which at most *one* may correspond to the measurement of the true object being tracked), where the Kalman filter only has one. This is likely to be useful for scratch tracking, since as mentioned above, several detections may be present in the same spatio-temporal vicinity.

We first experimented with the PDAF filter for scratch detection tracking. To do this, we use a state vector of two components $X = \begin{bmatrix} x \\ v \end{bmatrix}$, where x is the position and v is the horizontal speed of the scratch detection trajectory. Since we want to ensure the *smoothness* criterion, we expect the speed to remain as constant as possible.

Given these hypotheses, and supposing that a frame step represents a time step of one unit, the system dynamics and measurement matrices can be written simply as :

$$\begin{aligned} F &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \\ H &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

Obviously, we consider that there are no control inputs, and we consider that both the system dynamics and measurement errors are very small.

Some visual representations of the results of PDAF filtering may be seen in Figure 4.3.

In this synthetic example, four "trajectories" are present, with (approximately) a 5 percent of noisy detection density. We suppose that the tracks are correctly initialised : we know that they are not noisy detections, and we have a good estimation of their initial speed. We see that two trajectories are correctly followed, while two are not. This is obviously a very unrealistic example, since hopefully the noisy detection density will not be so great in real examples. Also, these sort of failures do not happen all the time, and parameter tuning or perhaps other iterative filters may certainly improve results.

However, we do note that in the worst case scenario the track is lost "for good", and is unlikely to be found again. This is a highly undesirable property for our tracking algorithm. Iterative filters are by their nature local, and in no way guarantee any global optimality of the trajectory. In our situation, we have access to *all* of the detections, which is very different from the settings in which Kalman and Kalman-like filters are used. Furthermore, our detections have discrete spatio-

temporal positions, which makes graph theory a possible candidate for tracking the scratches. Intuitively, we imagine that if we are able to determine the beginning and end points of a trajectory, such problems as the shortest path between two nodes could be very well-adapted to the tracking problem.

For these reasons, we decided to move away from iterative filters, and towards graph-based methods. It is acknowledged that continued research concerning the former methods would most likely improve results, however we decided that the discrete graph setting was more adapted to the scratch detection tracking problem.

4.4.3 Second tracking approach : a graph-based approach

Observing Figure 4.1 (b), it is relatively easy to imagine how I_T representation may be transformed into a graph, since it is already a discrete representation of the scratch detections. Therefore, we will try to formulate the tracking problem in the domain of graph theory. Our basic goal is to create a graph which will reflect the notions of *continuity* and *smoothness* allowing us to track each trajectory.

Before continuing, let us define certain notions concerning graphs.

Definition 4. *We define a* directed graph *to be a set of* vertices $\mathcal{V}$ *which are linked together by a set of* edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. *We shall denote such a directed graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

As all the graphs we will be considering are directed, we shall refer to them simply as graphs. First of all, we explain how such a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is constructed from the set of scratch detections in the video sequence. Each scratch detection corresponds to a vertex in the graph. A scratch detection S is defined by the frame in which it is found $t(S)$ and its average x coordinate $\tilde{x}(S)$. Each vertex is connected to all other vertices which are in the next τ_t frames and whose spatial distance is less or equal to τ_s.

We will now see how the two hypotheses, continuity and smoothness, may be formulated in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

Implementation of the continuity and smoothness criteria

To enforce the continuity hypothesis, we simply find the connected components in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Each connected component corresponds to one or more trajectories, but we suppose that no trajectory can be present in more than one connected component. From here on we shall deal with each sub-graph separately and in the same manner. To avoid introducing any unnecessary notation, we shall denote each sub-graph as $G = \mathcal{G}(\mathcal{V}, \mathcal{E})$, as in the case of the complete graph.

At this point, having transformed the binary map I_T into a set of graphs, we may be tempted to simply find the trajectories by using a shortest path algorithm, and this did in fact constitute one of our first experiments using graphs. Unfortunately, we come up against two problems if we wish to proceed in such a manner. Firstly, we need to determine the beginning and ending points of the trajectories, which is certainly not trivial. We shall look at this particular problem further on. However, even if we manage to do this, there still remains a second obstacle. For the shortest path algorithm to have any meaning, we need a weight for each edge. Now, this weight should reflect to what extent the edge, which represents a very localised part of a trajectory, respects our smoothness criterion. We may consider the "smoothness" of a trajectory, in physical terms, to be the minimisation of acceleration. The notion of acceleration, or equivalently a variation of speed, must therefore be incorporated into the weights of the edges, which cannot be done with the graph G in its current state (each edge represents a speed, but not an acceleration). For example, we

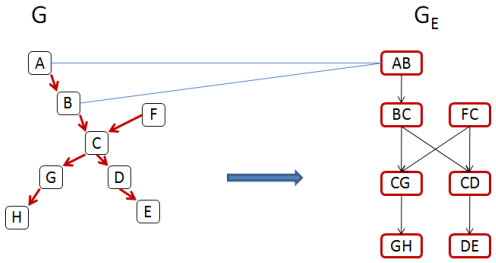


FIGURE 4.4: Example of the creation of an edge graph. Note that nodes A and B are merged to form a new node AB.

would consider a straight, vertical line and a slanted line in I_T to be equally smooth, but they would necessarily have very different costs if the total path cost was the sum of the trajectory *speeds*. In order to do this, G will be transformed into its *edge graph*, (also known as a *line graph*) which we will call G_E.

Definition 5. *Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a (directed) graph. The corresponding edge graph G_E is a graph such that each vertex of G_E corresponds to an edge in $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and two vertices of G_E are connected if and only if their corresponding edges in $\mathcal{G}(\mathcal{V}, \mathcal{E})$ are connected through a common vertex.*

To give an example, each edge (a, b) will become a vertex $v_{a,b}$ in G_E. Two vertices, $v_{a,b}$ and $v_{c,d}$ of the edge graph will be connected by an edge if and only if $b = c$ or $a = d$. An example of such a transformation may be seen in figure 4.4.

Edge weight Function

With the graph G_E it is possible to establish an edge weight function which includes an approximation of the acceleration. Since we may skip certain frames (vertices are temporally connected up to τ_t), but we would like to avoid it if possible, we also include a term which penalises frame skipping.

Let us briefly recall the notation introduced in Subsection 4.3.1. We recall that $\tilde{x}(a)$ and $t(a)$ are, respectively, the column and frame indices of the vertex a in the graph G. If we consider an edge $(v_{a,b}, v_{b,c})$, belonging to G_E, its weight is defined as follows :

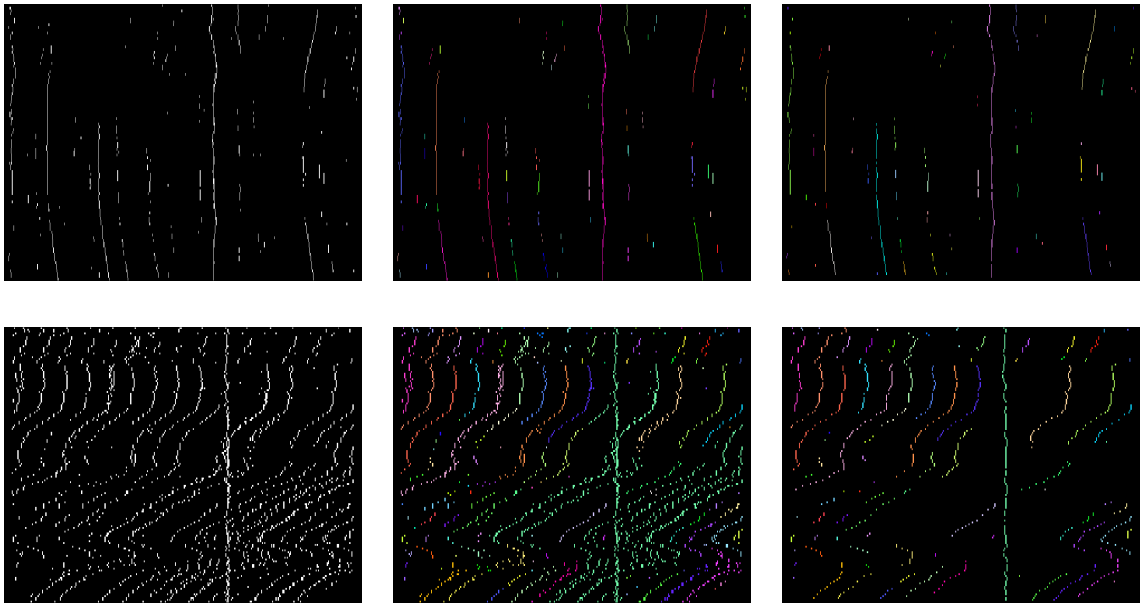
$$W(v_{a,b}, v_{b,c}) = \left| \frac{\tilde{x}(c) - \tilde{x}(b)}{t(c) - t(b)} - \frac{\tilde{x}(b) - \tilde{x}(a)}{t(b) - t(a)} \right| + \alpha \left(t(c) - t(a) \right), \tag{4.5}$$

where α is a coefficient which penalises frame skipping.

At this point, we need to extract the different trajectories from G_E. An obvious idea is to find the shortest path from the beginning and end points of each trajectory. This will not work, however, since we do not know *a priori* where these points are, nor how many real trajectories exist in each G_E. An illustration of this problem may be seen in Figure 4.5. It is clear that separating trajectories which are spatially connected is difficult. For this task we must try to *group* the detections into separate trajectories.

Trajectory extraction

Perceptual grouping problems may be solved using clustering methods. Various clustering algorithms are available, such as k-means, hierarchical clustering or density-based clustering. With



Original detections | Sub-graphs resulting from the continuity criterion | Shortest paths in each sub-graph

FIGURE 4.5: Continuity criterion and shortest path extraction for each connected sub-graph. In the second column, each colour represents a separate sub-graph resulting from imposing the continuity criterion. The third column represents the final trajectories which are extracted. In the first example (first row), it is relatively easy to follow each separate path, since they are clearly distinct. However, the second example (second row) shows that more complicated situations are difficult to handle with this approach : one of the sub-graphs contains multiple trajectories.

respect to the current problem, hierarchical clustering seems to be the most adapted. K-means needs a predefined number of clusters, and density-based clustering does not seem convenient, since the notion of density does not appear in an obvious manner.

Hierarchical methods (or connectivity-based methods) try to group the objects being clustered into different groups which exhibit a certain similarity inside each group. In the following case, we have chosen to implement an algorithm which is similar to *single-linkage agglomerative clustering*. The goal of the algorithm is to produce clusters which correspond to complete scratch detection trajectories.

Generally, the hierarchical methods try and produce several levels of clustering, represented with a *dendrogram*. In the present case we do not create such a dendrogram, since we are only interested in the clusters of the highest level, that is to say the level at which no clusters can be merged further.

The clustering algorithm starts by creating a list of all the edges in G_E. Let us refer to this list as F. Initially, each cluster contains one node only. We consider the edge of lowest cost, and remove it from the list of edges. There are two different possibilities : either the edge connects two nodes which belong to the same cluster, or it connects two nodes belonging to different clusters. In the first case, we ignore it and continue to the next edge. In the second, we try and merge the two clusters which the edge connects. We use the following *cannot-link* constraint : if the considered clusters contain any two detections in the same frame which are further than a certain

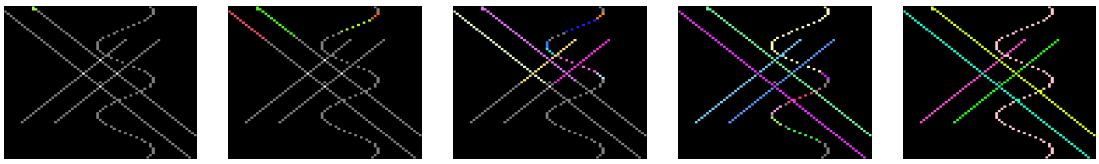


FIGURE 4.6: Evolution of clustering algorithm for a synthetic example. The coloured pixels belong to clusters, whereas the grey pixels have not been processed yet. Each colour represents a single cluster. In this simple situation, each final cluster represents a trajectory.

tolerance, they shall not be connected. This process is repeated until the list F is empty. At the end of the algorithm, the different groups correspond to the different trajectory groups. The algorithm is summarised in Algorithm 1.

The reason for the chosen cannot-link constraint is relatively obvious : we consider that a line scratch can be in one and only one position in each frame.

Data: Edge graph G_E
Result: Set of clustered nodes
$F \leftarrow$ list of all the edges in G_E;
Associate a cluster $C_{a,b}$ to each node $v_{a,b}$ in G_E;
while $E \neq \emptyset$ **do**
  Remove the edge $(v_{i,j}, v_{j,k})$ of minimum weight from F;
  if $C_{i,j} \neq C_{j,k}$ **then**
    if $C_{i,j}$ *and* $C_{j,k}$ *respect the cannot-link constraint* **then**
      Create a merged cluster with the nodes of $C_{i,j}$ and $C_{j,k}$;
    end
  end
end

Algorithm 1: Single-linkage trajectory clustering algorithm

Now that the trajectories have been separated, we would like to have as precise a representation as possible of each trajectory, in order to compare the trajectories with the scene motion. Unfortunately the trajectory grouping does not provide a unique path for each trajectory. This may be obtained with a shortest path algorithm between two points which represent the temporal extremities of the cluster (the "first" and "last" points). An example of the complete tracking may be seen in Figure 4.7.

Drawbacks of the graph-based tracking method Although the proposed method works well for simple and synthetic examples such as Figure 4.6, it fails on complicated situations. Parts of the trajectories are correctly extracted, but the algorithm has trouble identifying the entire trajectory. This is due to the fact that the algorithm is greedy and in no way guarantees the optimality of the results in a global context. An example of such problems may be seen in Figure 4.7. A legitimate question is whether it is a problem that only partial trajectories are extracted. In actual fact, the algorithm is quite accurate when it comes to the extraction of the partial trajectories, and makes very few mistakes apart from in very complicated situations. However, it is problematic that the total trajectories are not identified for the following reason. It is quite common that a trajectory

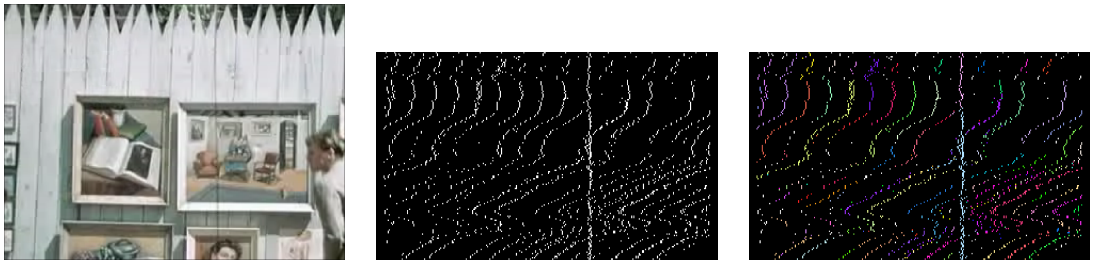


FIGURE 4.7: Trajectory clustering in a complex situation. Although the algorithm does a reasonable job of identifying and grouping parts of different trajectories, it has trouble establishing complete trajectories. Each colour represents a final trajectory cluster.

may be identified as a true scratch or a false alarm only during *part* of its total lifespan. For example, imagine a scene with both thin vertical scene elements and true scratches. If the scene does not move for a time, then it is impossible to distinguish true from false detections. However, once the scene moves, then it becomes possible to reject the vertical scene structures. It is also at this moment that tracking becomes more difficult, and trajectories are likely to be split into sub-trajectories. Thus in this situation, it becomes difficult to reject the *first* part of the trajectory, resulting in a severe weakness of the algorithm.

What is clearly needed is a global clustering of the trajectories. This is quite a difficult task : we want to extract globally optimal trajectories using very permissive *a priori* information (continuity and smoothness), and we do not know where the trajectories begin and end. Previously, work has been carried out on similar problems, such as that of Tepper *et al*. [116] and that found in the thesis of Maël Primet [104]. Interestingly, both of these approaches use the *a contrario* methodology to make decisions concerning the identification of likely trajectories. However, the work of Primet presents more flexibility with respect to gaps in the trajectories, which is obviously a great advantage in the present case.

Such methods are necessarily quite complicated. They are complicated because there is so little prior information on either the appearance of that being tracked (as in face tracking) or the nature of the motion. Instead of using these approaches, we shall now propose a different method. Up until this point, we have been concerned with tracking both true scratches and false detections, the hypothesis being that the trajectories of both these elements are *continuous* and *smooth*, and if the tracking is successful, we may either reject false alarms, accept true scratches, or both. In the following method, we concentrate on the *rejection* of false trajectories. We shall see it may be formulated in a more constrained manner than the previous problem, and is therefore easier to solve.

4.5 Proposed approach : determining the trajectories of false alarms

In this section, we present the approach which was finally retained for temporal line scratch filtering. Let us recall that the difficult part of temporal filtering is establishing reliable trajectories of scratch detections. In the previous Section, we saw that tracking both true scratches and false alarms is a very difficult problem, because the scratch detections have such a broad range of motions. Since this is a tricky problem, we now concentrate on identifying the trajectories of *false alarms*, since we shall see that they have less generic motion than true scratches. We still employ a motion coherence hypothesis, but only for false alarms, that is to say we suppose that any trajectory

which moves *with* the scene is a false alarm (see Hypothesis 1).

Let us recall certain notations. S represents a detected scratch segment, $\tilde{x}(S)$ and $\tilde{y}(S)$ represent, respectively, the average column and row indices of the scratch. We denote the frame in which the segment was detected with $t(S)$. Finally, I_T is an x-t binary image which represents all scratch detections in the sequence (for an example, refer to Figure 4.1).

If we look at Figure 4.7, we may notice that the false alarms display very characteristic motion : they all move in a similar manner. This is simply because they belong, by and large, to the background scene. Consequently, we have *a priori* information concerning the motion of false alarms which is much more precise than the continuity and smoothness constraints, and is therefore easier to use. In order to exploit this knowledge, we propose first of all to *realign* the scratch detections with respect to a series of *globally* estimated motions in each frame. The idea behind this operation is that if we subtract the scene motion from the scratch detections, the false alarms will appear in the realigned version of I_T as straight, vertical lines. Their detection should therefore be much easier, since we are looking to find vertical lines in a binary image which is a much more constrained problem than generic tracking. It is interesting that the problem of detecting vertical lines has been addressed in Chapter 3. We note that this general approach does *not* require any hypotheses such as continuity and smoothness, as was necessary in the previous Section, because we consider that the trajectories of false alarms appear as straight, vertical lines in I'_T, the realigned version of I_T. We note that this approach supposes that a dominant motion does indeed exist, and that false detections are highly likely to stem from the scene which induces the dominant motion. This is less the case with complex dynamic scenes. For an example of I'_T, see Figure 4.8 (b)).

4.5.1 Global scene motion estimation

In order to realign the segments, we need to obtain an estimation of the scene's global motion. To do this, we use the algorithm from Odobez and Bouthemy [97] to estimate an affine approximation of the dominant motion in a robust manner. At a pixel position $q = (x_q, y_q)$, the motion vector $(u(q), v(q))$ in such representations is expressed as :

$$\left\{ \begin{array}{rcl} u(q) & = & c_1 + a_1 x_q + a_2 y_q \\ v(q) & = & c_2 + a_3 x_q + a_4 y_q \end{array} \right. , \tag{4.6}$$

where c_1 and c_2 are the parameters describing the constant motion components, and $a_1 ... a_4$ are the parameters associated with the spatially varying components of the motion.

Let (x, y) be a pixel in a frame t and (x', y') be the corresponding position in frame $t + 1$. We have the following relationship :

$$\left[\begin{array}{c} x' \\ y' \\ 1 \end{array} \right] = \left[\begin{array}{ccc} a_1 + 1 & a_2 & c_1 \\ a_3 & a_4 + 1 & c_2 \\ 0 & 0 & 1 \end{array} \right] \left[\begin{array}{c} x \\ y \\ 1 \end{array} \right] := \tilde{A}_{t,t+1} \left[\begin{array}{c} x \\ y \\ 1 \end{array} \right] . \tag{4.7}$$

The motion estimation is carried out between each pair of consecutive frames throughout the image sequence.

Let x and y represent the spatial coordinates of a pixel in frame t. It is possible to find its corresponding coordinates, (x^r, y^r), in a frame r with the following relationship :

$$\left[\begin{array}{c} x^r \\ y^r \\ 1 \end{array} \right] = \tilde{A}_{r,t}^{-1} \left[\begin{array}{c} x \\ y \\ 1 \end{array} \right] , \tag{4.8}$$

with

$$\tilde{A}_{r,t} := \tilde{A}_{t-1,t}\tilde{A}_{t-2,t-1} \ldots \tilde{A}_{r,r+1}. \tag{4.9}$$

This provides us with the necessary tools with which to realign the segments. The binary detection map I'_T resulting from this realignment is defined as :

$$I'_T(x,t) = \begin{cases} 1 & \text{if} \quad \exists S \mid \tilde{x}^r(S) = x, \;\; t(S) = t \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

where $\tilde{x}^r(S)$ is the average column index of the segment warped to the reference frame. The reference frame is defined as the first frame of the sequence. It is important to note that it is the *original detection segments* which are realigned with respect to the global motion, and not the detection map I_T itself. This is due to the fact that several segments may correspond to the same spatio-temporal point in I_T. Since the dominant affine motion is not constant with respect to the y coordinate, two scratch segments may have the same average y coordinate, but be separated in I'_T. An example of a realigned detection map may be seen in Figure 4.8(b).

4.5.2 Clustering vertical trajectories

Once the detections have been realigned, we need to detect vertical line segments in I'_T, and thereby determine the trajectories of the false detections. Interestingly, the *a contrario* step of the algorithm presented in Chapter 3.2.1 may be used for this task, since we wish to detect line segments in a new binary image (the detection map). Before detecting the trajectories, we perform a horizontal morphological dilation of one pixel on $I'_T(x,t)$. This is necessary, since the spatio-temporal trajectories shown in Figure 4.8(c) are discretised, and therefore the trajectories may not be precisely detected without dilation.

In the previous approach presented in Section 4.4.3 (the graph-based approach) we noted that the key failing of the proposed algorithm was the greediness of the method and its subsequent difficulty in detecting complete trajectories. We also noted that to detect "complete" trajectories, we would need to have a framework which favoured smoother and longer trajectories. By using our modification of the *a contrario* methodology for detecting vertical lines in I'_T, we have single-handedly dealt with the problem of detecting complete trajectories and favouring the longest possible trajectories. Indeed, in the *a contrario* framework, longer trajectories are inherently more unlikely and therefore more meaningful. The maximality principle guarantees that the best trajectory possible is returned. We may recall that one of the difficulties which lead to the trajectory clustering algorithm was the fact that we could not be sure at what spatio-temporal point a trajectory began and ended. If this were not a problem, then we could simply have used a shortest path algorithm. As it is, the maximality principle elegantly solves the problem of finding the beginning and ending points of the trajectories.

The exclusion principle also plays an important role in that it clusters the detections into separate trajectories which have no detections in common. In this sense, it is quite similar to the *cannot-link* constraint used in Section 4.4.3. This property is of great practical importance, since it avoids having to make tedious decisions about what to do with a detection which is close to several trajectories. In terms of implementation details of the trajectory detection, we allow a maximum slope of 5 degrees in comparison to the vertical direction. This corresponds, roughly, to allowing a trajectory to deviate one pixel every eleven frames with respect to the underlying scene motion. The exclusion width parameter τ_x is set to 3 pixels, as in Chapter 3.2.1.

We now define a *trajectory* T as a list of x-t positions. Let $\mathcal{T}$ denote the set of all the trajectories detected with the *a contrario* detector. We define $\mathbf{S}(T)$ to be the set of all the scratch segments

S whose positions $(\tilde{x}(S), t(S))$ are within a horizontal distance less or equal to τ_x from any x-t position of T.

Returning to the temporal scratch filtering algorithm, we now need a *rejection criterion* for the scratch detections which belong to trajectories which are coherent with the scene motion.

4.5.3 Rejection of false alarms

Up to this point in the presentation of our work, it has been implied that all the trajectories detected as vertical lines in I'_T correspond to scene elements, and are therefore false alarms. Unfortunately, true scratches may be coherent with the global scene motion when the scene is static. Therefore, scratches may be totally or partially represented as straight vertical lines in I'_T. Such a situation may be seen in Figure 4.8 (c). Two short trajectories are present (in green) which correspond to the partial trajectories of true scratches. Since these scratches happen to be static, their trajectories are detected as vertical segments in I'_T when the scene does not move, but naturally are lost when the scene moves (they become significantly slanted after that point). The scratches should not be rejected as there is no significant motion of the scene over the time interval during which they are detected. Therefore, we need a robust criterion to determine whether a trajectory corresponds to false alarms.

This criterion is based on how much the *scene* has moved during the trajectory's lifespan. We determine this by inspecting the maximum *horizontal* motion of the scene. For this purpose, we could inspect the accumulated motion of the scene. However, we can also analyse the original positions of the segments in the trajectory set. Since we know that they display similar motion to the scene, these two processes are equivalent. Since these have been identified as conforming to the underlying scene motion, their original positions reflect this motion. This obviously holds true in the case of a static scratch and no scene motion. Therefore, we reject all the scratch segments in the set $\mathbf{S}(T)$ if there exist any two segments Q and R belonging to $\mathbf{S}(T)$ such that :

$$|\tilde{x}(Q) - \tilde{x}(R)| \geq \tau_m, \tag{4.11}$$

where τ_m is a motion threshold. This corresponds to the *maximum absolute distance* that the scene has moved in the frames of the trajectory set. It is important to note that this corresponds to the maximum scene motion *locally* in the area of the scratches. This can be crucial for situations such as zooming, in which case the scene presents different motions at different positions in the image. In all of our evaluations, we set the parameter τ_m to 10 pixels. The complete temporal filtering algorithm is represented in Algorithm 2.

4.5.4 Further filtering criteria

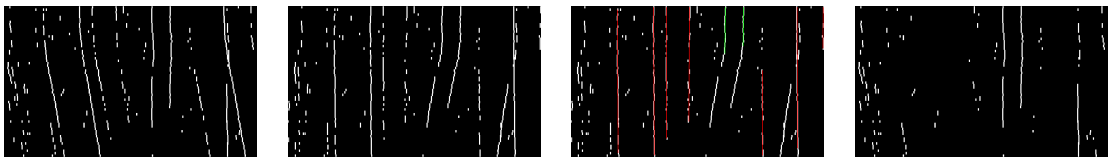
In the algorithm based on global scene motion estimation, we have concentrated on removing detections which follow the scene motion. This motion generally corresponds to elements of the background. However, it is clear that such situations do not cover all false detection possibilities. For example, we also produce impulsive detections with our spatial detection algorithm. Also, if a scene does not move, then there is no way of distinguishing true scratches from false detections.

Apart from motion coherence, there are other criteria which are reasonable to use for scratch filtering, and which are much more easily implemented than the motion coherence. One example, which we have used in our experiments, is a *scene cut* hypothesis. This simply stipulates that an entire trajectory set is rejected if the beginning and ending frame indices of the trajectory are within a temporal distance τ_c from a scene cut. In our experiments, we set τ_i to 5 pixels and τ_c

Data: Spatial scratch detections I_T, input video
Result: Filtered detections

Affine motion estimation with the method of Odobez and Bouthemy [97];
Realign I_T using the estimated motion, with respect to a reference frame;
$\mathcal{T} \leftarrow$ detect vertical trajectories using the *a contrario* detection algorithm of Section 3.3.3;
forall the *trajectories* $T \in \mathcal{T}$ **do**
  if $\exists (Q, R) \in \boldsymbol{S}(T)^2$, *such that* $|\tilde{x}(Q) - \tilde{x}(R)| \geq \tau_m$ *(see Equation* (4.11)*)* **then**
    Reject all the scratch segments $S \in \mathbf{S}(T)$ as false alarms;
  else
    Accept the scratch segments belonging to $\mathbf{S}(T)$;
  end
end

Algorithm 2: Pseudo-code for the temporal scratch detection filtering algorithm.



(a) Original detections (I_T)

(b) Realigned detections (I'_T)

(c) Detected trajectories. In red false alarms, in green true scratches.

(d) Final filtered detections

FIGURE 4.8: Different x-t binary maps, for the "Afgrunden 2" sequence. The horizontal axis represents the average horizontal position of a scratch, and the vertical axis represents the frame number (t) which the scratch belongs to. Each white point corresponds to detected scratch segment(s) in a given frame.

to 4 frames. We used a simple Edge Change Ratio based scene cut detection algorithm (see [87]) for the detection of the scene cuts, in which the edge detection was done using the Sobel operator. This can obviously be replaced by other scene cut detection algorithms, if necessary. This is a minor step of the proposed procedure. Indeed, in all of our experiments, it was used only for the "Laurel and Hardy" sequence.

Another criterion which is reasonable to implement is a non-impulsiveness hypothesis. This just corresponds to the idea that true scratches should have trajectories which last for a certain time. This criterion was very briefly mentioned in the thesis of Joyeux [69]. In Section 4.5, the global motion based algorithm will not process such scratch detections, and therefore treat them by default as true scratches.

We should underline that these two criteria are reasonable to implement, but do not constitute the core innovation of the proposed algorithm.

4.6 Visual results of the temporal filtering algorithm, and discussion.

In this Section, some visual results of our temporal scratch filtering algorithm will be presented. A quantitative evaluation of the algorithm will be done in the next Chapter, at the same time as the evaluation of the spatial detection algorithm. We will also compare our work to previous methods in the next Chapter.

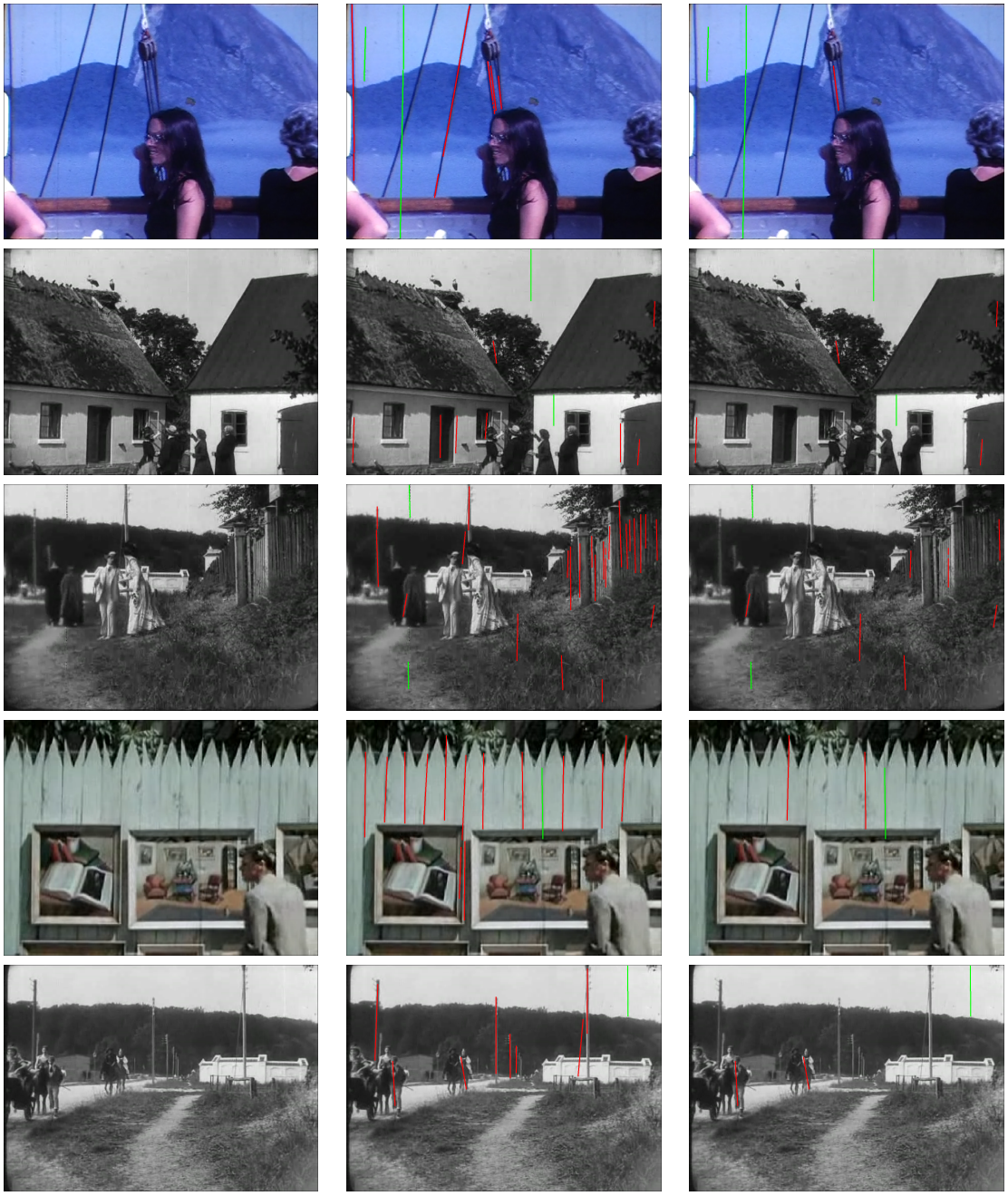
Figure 4.9 shows some of these examples. These are all sequences in which the temporal filtering "makes sense" in that enough motion is present to evaluate the performance of the algorithm. Therefore some sequences which we showed in Chapter 3 are not used here, such as "Sitdown" and "Star" which are only around 10 frames long. Visually, it is clear that the filtering removes many unwanted detections, especially those due to thin vertical objects which belong to the scene, such as posts and gates. In two of the sequences, gates produce a large amount of false alarms. Since there is a camera pan at some point in the sequence, the false alarms appear in a very clear fashion. This sort of example is especially difficult because the detections can be especially close together (see the third example in Figure 4.9) making tracking very tricky. The second to last example in this Figure presents a very interesting situation. At certain points in the sequence, the scratch (shown in green) overlaps with the gate. In this situation, our algorithm has the "correct" behaviour, that is to say it does not detect scratches : we do not want detection since we do not want to restore a structure ! This is a situation which is difficult to handle with previous tracking approaches, since the assumption is always made that a scratch detection corresponds to one object.

4.7 Conclusion and discussion

In this Chapter, we have looked in detail at the problem of filtering false detections remaining from spatial line scratch detection. Having looked at the state-of-the-art, carried out various experiments, and approached the problem from several different angles, it became clear that the most challenging part of any temporal filtering algorithm would be robustly tracking scratch detections. First of all, we tried to achieve this goal using classical tracking approaches, but soon identified limitations of such methods. In particular, these methods determine trajectories "on-line", and therefore cannot take the global quality of a trajectory into account. An example of this problem may be seen when the filter diverges : it has very little chance of picking up the correct track again. To remedy this, we turned to graph theory, which seemed particularly well-adapted to our problem. After re-defining the problem on a graph, we tried to use a clustering-based method to determine scratch detection trajectories. While this yielded good results in synthetic and simple examples, the algorithm turned out to suffer from the same inability to choose globally optimal trajectories.

Finally, we opted for an algorithm which aimed specifically at identifying the false detections. Using a global motion estimation to realign the detections, the tracking problem is reduced to a problem of detecting visually significant vertical trajectories in a binary image, which was treated in the first Chapter. This approach displays vastly increased robustness ; when we reject false detections, we are quite sure of the decision. This robustness comes from the combination of global affine motion estimation [97] and the maximality principle of the *a contrario* methodology.

On the other hand, the method may be said to have certain limitations. For instance, we have made the hypothesis that the false detections all belong to a scene which follows a single motion, as in the case of camera motion. This does clearly not cover the whole spectrum of false detections. For example, if there is a moving camera and a moving object with thin vertical structures



(a) Original frame (b) Spatial detections (c) Detections after temporal filtering by the proposed method

FIGURE 4.9: Detections from different scratched sequences. Correct detections are shown in green, false alarms in red. Sequences of variable difficulty are shown. In particular, we can see that situations with scene elements such as gates are better handled with our temporal filtering step.

(think of the door frames of a vehicle), estimating one global motion will clearly not be sufficient. Obviously, one would be tempted to extend the method presented to the case of several motions. However this poses several problems, such as automatically determining the number of motions (since manually setting this parameter would be undesirable). Also, it is not clear how to process detections which span several segmented motions. Finally, it is worth noting that as we consider increasingly local motions, we approach the initial tracking challenge of this Chapter, that is to say following any and all detections, which was shown to be quite difficult. Therefore, a possible research goal would be to find a compromise between tracking objects which respect a single global motion, and those which have several local motions. Alternatively, we could also continue using the continuity and smoothness hypotheses and use much more sophisticated trajectory clustering methods such as in the work of Primet [104].

Chapitre 5

Evaluation of line scratch detection

5.1 Quantitative evaluation of line scratches

In this Chapter, we present quantitative results of our line scratch detection algorithms. Its performance is compared with other approaches with respect to three criteria : recall, precision and the $F1$-score. Recall is defined as the number of true detections divided by the total number of true scratches present in an image. Precision is defined as the number of true detections divided by the total number of detections. Basically, recall determines what percentage of the line scratches are detected, and precision shows what percentage of our detections were correct detections. The $F1$-score is a reflection of both criteria, and defined as their harmonic.

We evaluate both the spatial detection step and the temporal filtering algorithm. These two contributions are compared with three other algorithms : the spatial method of Bruni *et al*. [25] and the temporal algorithms of Güllü *et al*. [74], and Müller *et al*. [95].

While other approaches exist, we have decided not to compare our work with theirs for several reasons. The initial work of Kokaram [76] is considered to have been improved upon by that of Bruni *et al*. , so we have not gone back so far chronologically in the literature. Another more recent spatial method [75] exists, it is a supervised algorithm (contrary to Bruni's and ours, which are automatic) and contains several parameters which are not given in the paper, such as the number of nodes in the input and hidden layers of the neural network, making implementation impossible without testing a series of architectures.

In terms of temporal scratch filtering methods, their are a very limited number of possibilities. We have chosen the two most recent approaches [74, 95] because while many of their parameters are not given, it is feasible to tune them to obtain a relatively fair comparison. While the work of Joyeux *et al*. [71] and Besserer *et al*. [16] present many interesting ideas, the lack of precision in their methodology make such fair comparisons difficult. Joyeux *et al*. do not specify the exact method of distinguishing true scratches from false alarms. In the case of Besserer *et al*. , it is not specified whether the tracking is supposed to lead to an automatic or semi-automatic rejection of the false alarms.

In our experiments, all the parameters are set to the values given throughout the document. In particular, the detection threshold ε is set to 1, the exclusion parameter τ_x is set to 3 pixels, and the motion threshold τ_m to 10 pixels. Note that we impose a minimum scratch length of one tenth of the image height for all of our spatial detections. The parameters for Bruni's algorithm are those given in [25] and [24], apart for the scratch colour parameter (black or white), which was set manually for each sequence. Three parameters are required by Güllü's algorithm. As it relies on Bruni's algorithm for the initial spatial detection step, the scratch colour must be specified (black

or white). The second parameter is a maximum search distance for the block matching algorithm. For this, we chose a maximum distance of 7 pixels either side, which corresponds to the maximum motion we expect in the sequence. Finally, we need a maximum mean absolute difference (MAD) threshold which identifies the presence of a scratch. We chose 15 grey levels for this threshold. These last two thresholds are not specified in [74]. Finally, Müller *et al*. [95], propose both a spatial approach and a temporal filtering step. Unfortunately, their spatial detection algorithm is not fully detailed, and cannot be reimplemented. However, we can compare the temporal step of the present approach with the temporal step from [95], which has some similarities with ours. Therefore, we use our spatial detections as inputs and filter these detections with the temporal part of the approach from [95]. This algorithm requires the setting of a neighbourhood size on the left and right hand sides of the scratch. This parameter is not specified in [95], and we chose a horizontal neighbourhood of 5 pixels. The other necessary thresholds are given in [95].

We are presented with a difficulty when comparing our algorithm, which produces a precise description of the line scratches, with methods that suppose that line scratches cover the entire height of the image. Our approach has an advantage in terms of precision, whereas the second type has an advantage with respect to recall. In terms of recall, we shall evaluate all the algorithms on a pixel-wise basis, in other words the number of annotated scratch pixels detected divided by the total number of scratch pixels. For the algorithms of Bruni *et al*. and Güllü *et al*. we shall consider that all the pixels in a detected column are detected. On the other hand, a fair and meaningful comparison of precision is more difficult to achieve. Naturally, we should evaluate our algorithm and that of Müller *et al*. on a pixel-wise basis, that is to say the number of annotated pixels detected divided by the total number of pixels detected. However, if we do this for the algorithms in [25] and [74], we shall bias the precision of their algorithms, especially when short scratches are present. Therefore, for these algorithms, we shall consider that a detection is "correct" if it touches at least one annotated pixel. This obviously confers a considerable advantage on Bruni's and Güllü's algorithms, but it would be unfair to evaluate them otherwise. For comparison, we have also included the pixel-wise precision evaluation for Bruni's and Güllü's algorithms in Table 5.1, which are written in smaller font below the main evaluation. For all of these evaluations, we allow a spatial detection error of 2 pixels, in other words a detection is considered correct if it is within a distance of 2 pixels from an annotated pixel.

Tests were carried out on ten film sequences of varying characteristics. The first three ("Knight", "Sitdown" and "Star") are commonly found in the line scratch literature, and are found in Kokaram's book [77]. "California" and "Laurel and Hardy", contain straight, vertical scratches, similar to the first three examples. "Les Choses de la Vie" displays scratches which are more difficult to detect (not completely straight, slanted and/or faint). While the first six sequences are useful for the evaluation of our spatial line scratch detection algorithm, the temporal filtering step is of little use in these cases, since the sequences are either very short, or contain no false alarms which may be rejected using temporal aspects. The last four sequences are longer and illustrate the improvement on precision we are able to obtain by using temporal filtering.

The annotation of the sequences was done by manually noting the beginning and end points of each scratch segment. In the case of scratches which were not completely straight, several consecutive segments were annotated. We performed this annotation task because, to the best of our knowledge, no standard database exists for scratch detection. The complete annotated sequences, as well as the detection results can be downloaded from the following address : `http://www.enst.fr/~gousseau/scratches`.

5.1.1 Recall

In the first four sequences, Bruni's and Güllü's algorithms produce better recall than ours. This is due to the fact that our evaluation gives the benefit of the doubt to these algorithms by considering that all of the pixels in a detected column are detected. Our spatial algorithm, on the other hand, must determine the beginning and end points of the scratch with high spatial precision. In the "Knight" sequence, for example, we are also able detect the correct column indices of the scratches 100 percent of the time, but sometimes miss certain parts of a scratch.

In the remaining sequences, we see our spatial algorithm's strong points : it is able to detect scratches with varying characteristics. This may be explained by our algorithm's ability to detect and represent slanted and disjointed scratches as a collection of segments with varying length and angle. It is also able to detect faintly contrasted scratches, even in highly textured areas (as in the extract from "Les Choses de la Vie" example). Contrary to the other tested methods, our recall is high for all sequences.

It can be seen that the algorithm of Müller *et al*. produces relatively poor recall on all of the sequences. This is because their corresponding rejection criterion is often verified by true scratches. Indeed, due to the lack of an efficient tracking scheme, a large number or correct detections are rejected as long as the local motion is large enough (more than 0.2 pixels per frame, in absolute vale). Furthermore, in practice many "temporal holes" are observed in the resulting scratch detections. This means that a restoration process using these detections are likely to produce flickering scratches, which may be a very undesirable result.

It should be noted that algorithms which filter the detections according to a temporal criterion (ours as well as those of Güllü *et al*. and Müller *et al*.) can only decrease recall (since no new detections are produced). Therefore, an important property of the temporal filtering stage is that it should not deteriorate recall. It may be observed in Table 5.1 that our algorithm induces very little loss of recall, with a maximum loss of 4.77 percent in the "Les Choses de la Vie" sequence. Müller's algorithm, on the other hand, decreases recall by a maximum of 69.77 percent ("Knight").

5.1.2 Precision

As stated earlier, our evaluation procedure confers a strong advantage on Bruni's and Güllü's algorithms in terms of precision. In spite of this advantage, our spatial algorithm is able to outperform these algorithms in nine out of ten of the sequences. This performance is due to the *a contrario* grouping and validation process, which limits the number of false detections in noisy situations as well as in textured areas.

As previously mentioned, the first six sequences do not present any interesting situations in terms of temporal filtering, which explains why the precision is practically the same for our spatial and temporal algorithms for these sequences. In the last four sequences, however, our temporal algorithm significantly improves the precision of our spatial algorithm, with a maximum increase of 22.91 percent ("Afgrunden 2"). The temporal filtering step increases precision in all of the sequences apart from "Sitdown", which decreases by 0.22 percent only.

Müller's algorithm presents good precision on most of the videos. In three cases, this algorithm outperforms our temporal approach. Unfortunately, this precision comes with very low recall, which is of little use for restoration purposes. This is reflected in the $F1$-scores, which are generally quite low.

Güllü's algorithm fails to significantly increase precision because the MAD threshold introduced in [74] is not robust enough. It is sufficient that one MAD value be quite high for an entire trajectory to be validated as a true scratch. In practice, this happens often even though we allow a

very tolerant MAD threshold. Conversely, our algorithm robustly determines a complete trajectory, so that a better-informed decision can be made.

5.1.3 $F1$-score

The $F1$-score is defined as the harmonic mean of the recall and precision :

$$F = 2\frac{recall \times precision}{recall + precision} \tag{5.1}$$

This score illustrates the performances of the methods more clearly than either the recall or precision alone. The results show that our spatial algorithm retains a good $F1$-score for all of the sequences, and outperforms Bruni's, Güllü's and Müller's algorithms in nine out of ten sequences. Furthermore, our temporal filtering step improves the $F1$-score of our spatial algorithm in all of the sequences apart from "Sitdown" where it decreases by 0.64 percent, and "Les Choses de la Vie" where it decreases by two percent. This is an important point, since it implies that this temporal filtering step may be used on any type of sequence (short, long, with or without motion) with practically no deterioration in the resulting detection.

To resume, both the spatial and temporal algorithms introduced in this work provide a significant improvement on previous methods. The same set of parameters was used for all the sequences. We do not have to specify to our algorithm whether black or white scratches are being detected, which is a significant advantage over other methods. Furthermore, our algorithm produces pixel-precision detections, which can be crucial for avoiding restoring non-degraded parts of images. Our experiments and evaluations were carried out without any sequence-dependant tuning, which illustrates the robustness of the algorithms.

5.1.4 Algorithm parameters

In this section, we briefly discuss several of the more important parameters in our algorithm. Although the same parameters were used for all of our experiments, we would like to illustrate the evolution of the algorithm's performance with respect to certain key parameters. In particular, we inspect the influence of the box size used for local noise estimation and the parameter ε. These two parameters are of great importance, since they represent determine our background model and line scratch detection threshold. The results of this analysis may be seen in Figure 5.1.

We can see that the $F1$ score is relatively stable around $\varepsilon = 1$, which is to be expected given the log-dependence of the NFA on ε (see [45]). It may be seen that in some sequences, the maximum value of the $F1$ score is not centred on $\varepsilon = 1$. However, this does not mean that we have chosen the incorrect value of ε. Since the spatial algorithm is incapable of distinguishing between true scratches and thin vertical structures, lowering ε does not imply an increase of the $F1$ score ; we need the temporal filtering step for this. Therefore, these results should be interpreted only as meaning that the $F1$ score varies slowly with ε. We may also see that the $F1$ score is stable with respect to the box size used for empirically estimating the local noise density. We have shown a range of values from $\frac{1}{10}$ to $\frac{1}{50}$ of the image dimension. This means that it is a reasonable choice to make this parameter dependant on the image size.

The thresholds on our scratch model (a median filter over five pixels, and left and right average values) were empirically determined for the scratches which were found in our sequences. However, they may need to be changed for higher resolution images, in which the scratch may cover more pixels.

Evaluation	Algorithm	Film				
		Knight	Sitdown	Star	California	Laurel-Hardy
Recall	Bruni	**100.00**	**80.93**	**95.00**	**82.07**	41.87
	Güllü	**100.00**	**80.93**	**95.00**	35.56	38.74
	Müller	09.57	17.26	41.20	54.07	38.65
	Spatial	79.34	73.72	82.21	81.10	**59.35**
	Temporal	79.34	72.63	79.68	80.82	59.26
Precision	Bruni	29.54	56.47	56.87	10.79	08.84
	Pixel-wise evaluation	23.93	28.54	13.67	06.62	07.88
	Güllü	29.54	56.47	**57.32**	11.91	12.01
	Pixel-wise evaluation	23.93	28.54	13.78	06.67	10.37
	Müller	**79.86**	51.62	46.57	**81.22**	31.89
	Spatial	71.70	**72.07**	53.15	79.60	38.31
	Temporal	73.97	71.85	56.25	79.89	**45.29**
$F1$-score	Bruni	45.61	66.52	71.15	19.08	14.60
	Güllü	45.61	66.52	**71.50**	17.84	18.33
	Müller	17.09	25.87	43.72	64.92	34.95
	Spatial	75.33	**72.88**	64.56	80.34	46.56
	Temporal	**76.56**	72.24	65.95	**80.35**	**51.34**
Execution time (s)	Bruni	0.03	0.05	0.08	0.03	0.03
	Spatial	0.49	106.22	98.67	17.13	1.85

Evaluation	Algorithm	Film				
		Choses Vie	Afgrunden 1	Afgrunden 2	Keldjian	Gate
Recall	Bruni	43.43	75.11	59.43	12.63	49.81
	Güllü	29.99	68.75	52.98	11.48	31.29
	Müller	16.31	50.65	42.34	14.06	47.12
	Spatial	**64.88**	**86.66**	**94.01**	**77.35**	**89.03**
	Temporal	60.11	86.39	93.71	77.16	**89.03**
Precision	Bruni	25.06	09.35	07.85	07.11	02.45
	Pixel-wise evaluation	05.67	03.20	01.85	01.75	00.48
	Güllü	25.45	10.27	07.43	05.98	02.27
	Pixel-wise evaluation	03.47	03.56	01.74	01.47	00.44
	Müller	**75.48**	48.17	35.28	29.08	03.06
	Spatial	67.80	45.85	28.01	17.43	03.32
	Temporal	69.19	**67.76**	**50.92**	**38.06**	**11.93**
$F1$-score	Bruni	31.78	16.62	13.86	09.10	04.68
	Güllü	27.53	17.86	13.03	07.86	04.23
	Müller	26.82	49.38	38.49	18.96	05.74
	Spatial	**66.31**	59.97	43.16	28.45	06.40
	Temporal	64.33	**75.94**	**65.99**	**50.98**	**21.04**
Execution time (s)	Bruni	0.12	0.08	0.08	0.06	0.06
	Spatial	2.52	5.33	4.65	2.24	12.63

TABLE 5.1: Recall, precision and F1 values comparison, in percentage. We compare our spatial and temporal results with those of Bruni *et al*., Güllü *et al*. and Müller *et al*. In smaller font are the results of the pixel-wise evaluation of precision for Bruni's and Güllü's algorithms.

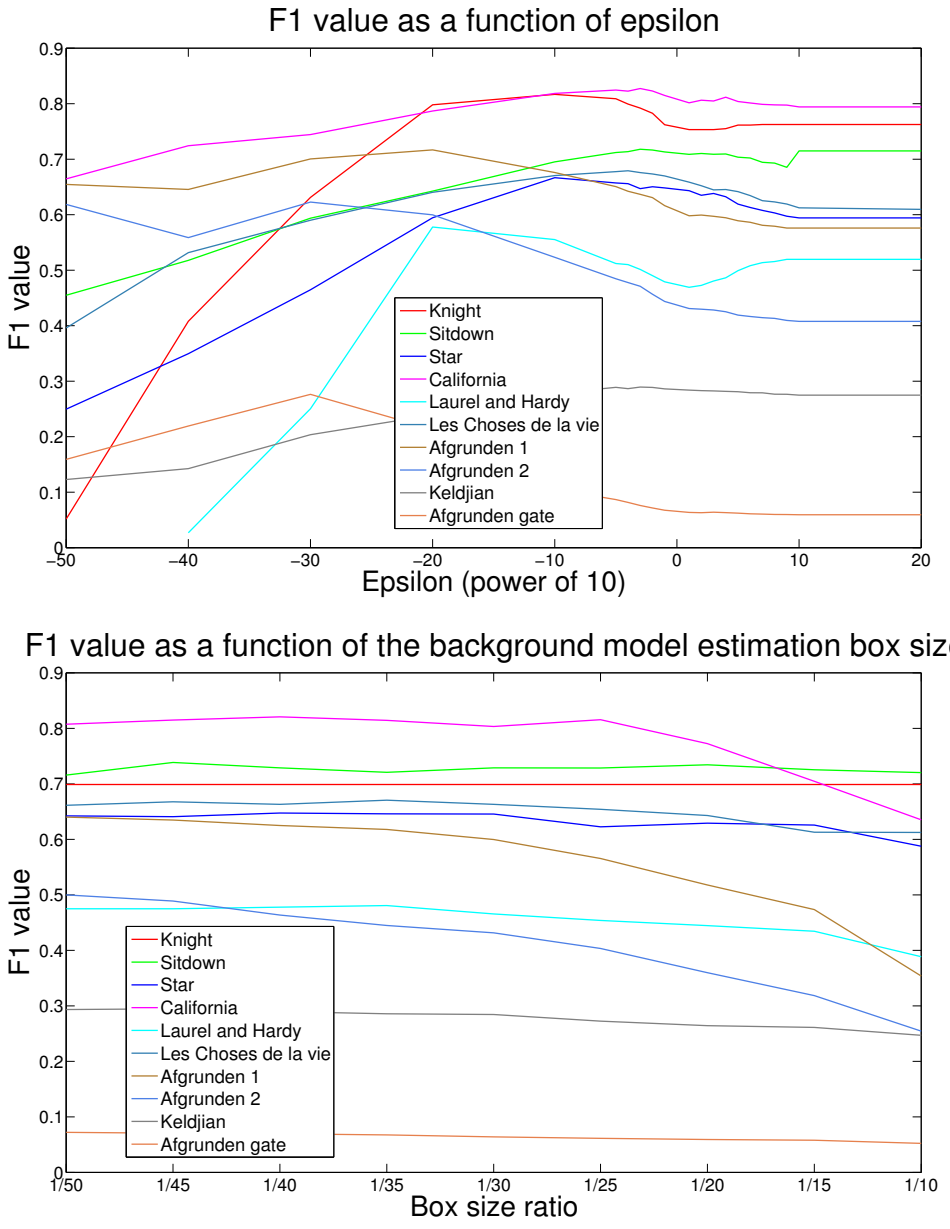


FIGURE 5.1: Variation of the performance ($F1$-score) of the spatial detection algorithm with respect to the values of ε and the box size used for noise estimation. The box size is a fraction of the image dimension for each sequence.

Another parameter which may be discussed is τ_m, the motion threshold which flags a trajectory as being a series of false detections. The most important aspect of this threshold is that it should not be set too low otherwise all the true scratches in still frames will become flagged as false alarms ; we need to be very sure that the scene has moved significantly before taking any action. Ideally, we would like to determine the maximum error of the global motion estimator of [97] and set τ_m to a value greater than this. However, this may be too close an analysis for such a task. The parameter was set to a conservative value and was sufficient to deal with all the sequences in our experiments.

5.1.5 Robustness to noise and texture

One of the major assets of *a contrario* detection procedures is their ability to control the number of false detections in noisy or cluttered images. Often, such methods rely on statistics that are learned globally from the considered image. In our case, we found it necessary to estimate such statistics *locally*, in order to be resistant to textured regions.

In this section, we illustrate the ability of the approach to control the number of false detections and the importance of performing local statistical evaluation.

This can be observed in the images in Figure 3.15. Due to the highly textured nature of the images, false detections are produced when we try to use our algorithm with no local noise estimation. When we introduce local noise estimation, detections are limited in areas with high noise density, which illustrates one of the strong points of our algorithm. In the image of the monkey, we produce one false alarm with local noise estimation, which is coherent with the chosen value for ε.

5.2 Future work

There are several aspects of the current work which could be developed further. Firstly, the spatially varying *a contrario* model presented here could potentially be applied to other detection problems, such as the detection of parametric shapes in preprocessed images. One advantage of the approach is that it does not require a closed formula to determine the probability of the considered object being produced by a noise model, which decreases the amount of work necessary to extend the approach.

Secondly, the global motion model estimates one dominant, affine motion only. This is obviously a relatively simple model, and several motions could potentially be estimated to allow for more complex situations. However, the estimation of more complex motions would decrease the robustness of the temporal filtering step. This decrease in robustness must be compared with the robustness of actually tracking the true scratches (as in [74]), and a compromise or a mixture of these methods could therefore be of great interest. Ideally, we would like to be able to track both false detections and true scratches, in order to make a more informed decision on whether to accept or reject the detections.

Finally, although we have carried out quantitative testing of our detection procedure, the true evaluation of defect detection lies in the final restored sequence. In future, therefore, we could also evaluate the detection qualitatively by inspecting the resulting restoration.

5.3 Conclusion

In this work, we have presented a precise spatial line scratch detection algorithm and a temporal filtering step which eliminates false alarms. The spatial algorithm uses an *a contrario* validation step to determine if the detected segments are visually significant or not. Our algorithm provides a precise description of the detected scratches, which is not given by any other fully automatic algorithm. Furthermore, it has similar performance to the state-of-the-art in simple cases, and outperforms the latter considerably in more difficult situations.

The temporal filtering step of the line scratch detection eliminates false alarms which are caused by thin vertical structures belonging to the scene. This is done by identifying scratch detections which are coherent with the scene's motion or which stop at a scene cut. This allowed us to reject false alarms in a robust manner, which improves the precision of the algorithm, with almost no decrease in recall.

We have evaluated these contributions with respect to previous work. With respect to these algorithms, we found our methods have the following advantages :

- Our spatial detection step provides a pixel-precision description of line scratches
- Scratches which are not completely straight are approximated by a series of line segments
- Our spatial detection step is designed to be robust to noise and clutter, which is a common problem in old films
- Our temporal filtering step relies on more robust hypotheses than previous methods
- The temporal filtering problem is reformulated into a problem which uses the advantages of our *a contrario* based line segment detector

The spatial precision and robustness of the approach makes it a good candidate for automated line scratch detection in real world restoration applications.

Deuxième partie

Video Inpainting

Chapitre 6

Video inpainting

6.1 Introduction

Inpainting is the task of filling in an unknown region with some content which is coherent with the rest of the image or video. The problem of inpainting differs from other common restoration techniques in that the section to inpaint is *completely* unknown, and generally speaking the information which determines the result is that contained in the rest of the image.

The inpainting problem is an inverse problem, and one of its most difficult aspects is that the evaluation criteria are inherently subjective. The result must be "coherent" with the rest of the content, but this coherency may be interpreted in different ways. For instance, one could specify that the texture should maintain the same properties inside and outside of the occlusion, or perhaps that the structures should look coherent, or a mixture of both. A very high-level approach would be to specify that the result should make sense with respect to scene content, for instance adding a beach ball to a seaside scene. All of these approaches may lead to "good" solutions, and there is not necessarily any clear way to rank them. In this sense, it is an ill-posed problem : there is not necessarily a unique solution. It is also ill-posed because, even if we assume that a unique solution exists, there is certainly no guarantee that this solution would change continuously with small changes in the border (initial) conditions.

It is possible, however, to identify some important goals that should be obtained while resolving the problem. For images, they are :

– Correctly reconstruct structure and achieve geometric coherence
– Maintain similar textures

For videos, these goals must be maintained, but the following are added :

– Reconstruct background and foreground, in a manner which preserves their separate properties
– Convincingly reconstruct moving objects which are occluded
– Reconstruct video textures, such as waves, flames and moving tree leaves
– In a general manner, impose temporal coherence

It is clear, therefore, that while the inpainting problem applies to many domains in general, the characteristics of a satisfactory solution will depend entirely upon which domain is being considered.

Our aim in this Chapter is to produce an automatic and generic video inpainting algorithm which can achieve the aforementioned goals. We will address several problems associated with video inpainting, and describe the algorithm which we propose in Section 6.5. We will provide visual results of our algorithm and compare it to previous work.

In the next section, we will review the image and video inpainting literature. In particular, we shall see which of the inpainting goals have been considered and the manner in which they are formalised. We will try to accentuate some of the key works and go into greater depth when a reference has introduced some new or useful idea to the domain. In this manner, some of the common tools and notations should appear in as natural a fashion as possible. We also note that the emphasis will be put upon *patch-based* inpainting methods, as they will be most useful to us in our work further on. We shall maintain a coherent notation throughout our exploration, and modify that of the original papers where necessary.

6.2 State-of-the-art

Before entering into the inpainting literature in a strict sense, we shall briefly discuss some aspects of *interpolation* which, after all, has the same goal as inpainting, but with different *a priori* knowledge on the nature of the signal to reconstruct. First, let us set out some notation which we will maintain in the rest of the Chapter.

Let $\Omega \subset \mathbb{R}^2$ be the support of an image, over which the image is defined. Let u be the image itself. Also, we denote with $\mathcal{H}$ the support of an *occluded* region in the inpainting problem, and $\mathcal{D} = \Omega - \mathcal{H}$ be the unoccluded region. In practical terms, $\mathcal{H}$ is a binary mask which indicates the occluded pixels. This is mostly defined manually. Where necessary, we shall use u^* to denote the known (fixed) information of the input image.

As mentioned, one method of inpainting is simply an application of mathematical interpolation. This could be as simple as a linear interpolation in a certain direction, if the occlusion has a specific shape. More generally though, one of the most well-known ways of smoothly interpolating the border conditions of an occlusion is to minimise the following functional :

$$\min_u \int_{\mathcal{H}} |\nabla u|^2 \tag{6.1}$$

$$\text{with } u|_{\partial\mathcal{H}} = u^*|_{\partial\mathcal{H}}, \tag{6.2}$$

where $\partial\mathcal{H}$ is the occlusion boundary. The Euler-Lagrange equation associated with the previous minimisation is Laplace's equation with Dirichlet border conditions :

$$\Delta u = 0, \tag{6.3}$$

$$\text{with } u|_{\partial\mathcal{H}} = u^*|_{\partial\mathcal{H}}, \tag{6.4}$$

where $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ is the Laplacian operator. The resolution of this equation interpolates the image smoothly, using the boundary conditions. However, it is sufficient to try this method once on real images to see its limitations. It fails to correctly reconstruct either structure or texture. This is obviously problematic, since these are two of the most important goals. For an example, see Figure 6.1.

Solutions to these problems have been proposed in the literature specifically designed for the inpainting problem, to which we shall now turn.

6.2.1 Image Inpainting

Geometric and PDE formulations The first work on the specific problem of inpainting was that of Masnou and Morel [92, 91], although the problem is referred to as "disocclusion". They propose

Original frame : “Lena”, occlusion in green

Result using Laplace’s equation (smooth interpolation)

FIGURE 6.1: An example of a very “naïve” approach to inpainting. A region of the “Lena” image is inpainted by resolving Laplace’s equation with Dirichlet border conditions around the occluded region.

a level-lines based algorithm which has the appreciable quality of respecting strong discontinuities. One of the main motivations for this work was the work of Gaetano Kanizsa [73] on the perceptual continuation of occluded parts of objects by the human brain, also named “amodal completion”. This is implemented by minimising the the following functional :

$$\min_u \int_\Omega |\nabla u|(1 + |\text{curv}(u)|)dx, \quad \text{with } u|\mathcal{D} = u^*|\mathcal{D}, \tag{6.5}$$

where $\text{curv}(u) = \text{div}\frac{\nabla u}{|\nabla u|}$, and we recall that u^* is the known (fixed) content of the image in the region $\mathcal{D}$. This functional specifies three things. Firstly, that the length of the level lines should be minimal. Secondly, that the total angle variation should be minimal along the lines. Finally, it specifies the boundary conditions of the solution (which are given, naturally, by the input image u). Masnou and Morel use the coarea formula (see [55]) to link the total variation of u (given by $\int_\Omega |\nabla u|$) to the perimeter of the level lines, which corresponds to the first condition. This yields a geometrical solution to Equation (6.5), obtained by finding an optimal set of curves interpolating level lines in the occlusion. In the case of Equation (6.5), these curves are polygonal lines, although more general curves may be required when considering different powers of the curvature.

As mentioned above, one advantage of this approach is that it respects strong boundaries, in this case represented by the level lines. Therefore, this approach highlighted a difference between interpolation in a general sense and inpainting, which is simply that images do not behave in a smooth manner, as would some mathematical functions.

The term “inpainting” was first introduced by Bertalmío *et al*. in [14], and was inspired by the artistic restoration of paintings. In this process, the artists paint from the border towards the centre of the occlusion, filling in structure and then fine details. The goal of their algorithm is to correctly continue “isophotes” into the occluded area. Isophotes are simply lines of constant brightness (in fact a very similar notion to level lines). This is done by using an iterative algorithm

which progressively propagates image information *along* the isophotes into the occlusion, and this process alternates with the resolution of an anisotropic diffusion equation. While this method allows a more flexible treatment of structures, it does not consider any notion of textures, and as such may not be used for most large zones, as is also the case for the approach of [92].

Ballester *et al*. [9] proposed a similar approach with an energy whose goal is also the correct continuation of isophotes in images. This paper poses some of the important notions of [14] in a more mathematically rigorous context.

Another PDE based approach is that of Chan and Shen [36]. This work is interesting in that it studies some simple, stable solutions of inpainting functionals based on total variation which fail to connect structures.

Elder and Goldberg presented a method [52] which inpaints images using their *edges maps*. The reasoning behind this idea is that edge maps reflect the image content much better than pixels. For example, edges can represent object boundaries, shadow boundaries and phenomena resulting from the acquisition of a 3D scene (such as folds and creases). Contours are automatically extracted from the image, and the user is allowed to manipulate them, with operations such as deletion, cropping and copy-and-paste. The idea that the pixel should be replaced by another basic image element is interesting, but is perhaps overtaken by the subsequently introduced idea of image *patches* which are able to encode both structure and texture.

In [15], Bertalmío *et al*. decompose the image into structure and texture components, and these are inpainted separately. The separation is achieved by a minimisation of a total-variation based energy functional proposed by Vese and Osher in [118]. The texture part is inpainted using the work of Efros and Leung [49], whereas the structure part is dealt with in a similar manner to their previous work [14].

Other methods based on partial differential equations (PDEs) have been proposed such as [110, 117]. The improvements proposed, with respect to the work of Bertalmío *et al*. [14] include contour preservation via an anisotropic diffusion equation [117] and the work of Bornemann and März [19].

To resume, this family of methods is almost exclusively concerned with the correct inpainting of structures, and generally speaking they fail at respecting texture, which results in algorithms that are better adapted to small occlusions rather than large ones. The next set of methods are also only applicable to such small occlusions.

Inpainting using image transforms This category of inpainting methods tries to reconstruct missing image information in various transform domains. One example of such an approach is that of Elad *et al*. [51]. This algorithm reconstructs two different parts of an image : the “cartoon” part where the image is piecewise smooth, and the texture part which represents the oscillating nature of the image. They suppose that these two parts can be sparsely represented in different domains, and this guides their subsequent optimisation process. They develop their approach for a general set of domain generating functions (tight frames), but implement it using piecewise linear polynomial functions for the cartoon part, and the DCT basis for the texture part. However approaches which try to achieve this using optimisation in various domains are usually only applicable to occlusions which are very small or thin.

In [35] Chan *et al*. used total variation minimisation in the wavelet domain to inpaint damaged wavelet coefficients. The original aspect of this method is that the *final* goal is to reconstruct the coefficients. An example for the use of such methods include reconstructing wavelet information in compressed images, as in the case of the compression standard JPEG2000.

An approach which inpaints in the *framelet* domain may be found in [32]. Again, this approach presents results on small occlusions only. One example shows the results of inpainting where the occlusion is defined using a regular grid. Although such examples are quite specific and do not really correspond to the cases which we wish to deal with, it is nevertheless important to keep in mind that if they arise, a class of methods is available which performs very well in such situations.

Inpainting using probabilistic methods Levin *et al*. [85] exploit global image statistics, and learn an exponential family distribution conditioned on certain features of the image such as gradient direction. This distribution provides an inpainting solution, when the border conditions are specified.

Wang [121] modified the Piecewise Linear Estimators method of Yu and Sapiro [126] to inpaint small occluded regions. Each occluded patch is associated with of a series of patch classes, which themselves follow a Gaussian distribution. At the same time, the parameters of each class's distribution are estimated so that the variability of each class is adjusted to the observed image. Obvioulsy, this algorithm can only deal with small occlusions, since the image content reconstruction requires that each patch be associated with a patch class, and this is impossible if a patch's content is *completely* unknown.

The common feature of the previous algorithms is that they do not deal in a satisfactory manner with large occlusions. This is in many cases due to the incapacity of the algorithms to recreate *textures* which are coherent with the rest of the image.

Texture synthesis We now take a short detour into the subject of *texture synthesis*, because certain ideas in this domain played a very important role in the development of inpainting itself, in particular the notion of image *patches*. This notion is so important to inpainting that a brief look at its origins is in order.

Traditionally, textures are modelled in a stochastic manner, that is to say that each pixel of a texture may be considered as a random sample of a certain distribution, which must be estimated. Such an approach may be found in [40]. In this paper, textures are modelled using Markov random fields, and an algorithm to generate the textures is proposed.

In [49], Efros and Leung introduced the idea of the non-parametric estimation of distributions for texture synthesis. This led to the notion of image patches, also called *exemplars*. Before going into this method in greater detail, we shall set out some important definitions concerning patches.

Definition 6. *We define a* patch neighbourhood*, $\mathcal{N}_p$, to be a spatial neighbourhood of the position p.*

This spatial neighbourhood can be any shape or size, and does not even have to be connected in any spatial sense. However, for our purposes, $\mathcal{N}_p$ will most often correspond to a square or a rectangle centred on the pixel p.

Definition 7. *We define a* patch W_p^u *to be a vector of the image values found at the positions belonging to $\mathcal{N}_p$.*

$$W_p = \left(u(q_1) \cdots u(q_N)\right)_{q_i \in \mathcal{N}_p}, \tag{6.6}$$

where $N = card(\mathcal{N}_p)$.

In the case where we have a colour image, W_p^u is simply the concatenation of each colour vector for all the positions in $\mathcal{N}_p$. We denote the set of all the patches in an image as Ψ. Two patches may be compared with the use of a distance (or metric). Generally, for any two patches

W_p^u and W_q^u, we shall refer to the distance between them as $d(W_p^u, W_q^u)$. Practically speaking, we usually want this distance to indicate whether W_p^u and W_q^u are perceptually similar or not.

We shall often drop the superscript u in W_p^u for simplicity's sake, as it is clear that the patches come from the image u.

Definition 8. *We define a* nearest neighbour *of a patch $W_p \in \Psi$ in a set of patches Ψ' as* $\arg\min_{W_q \in \Psi'} d(W_p, W_q)$. *The nearest neighbour will be abbreviated with NN.*

The set Ψ' may be any set of patches in general, however in the present work it will mostly be patches W_p such that for all $q \in \mathcal{N}_p$, $q \in \mathcal{D}$, in other words none of the patches contain any occluded pixels.

Definition 9. *We define the* ε-approximate nearest neighbour *of W_p as a patch $W_{q'}$, such that there exists $\varepsilon \in \mathbb{R}^+$ so that $d(W_p, W_{q'}) \leq (1+\varepsilon) d(W_p, W_q)$, where W_q is the exact nearest neighbour of W_p. This shall be abbreviated as an ε-ANN.*

We shall also refer to *approximate nearest neighbours*. This is the case when we have found a patch which is a "good" match with W_p, but is not the exact NN. In this situation we have found an ε-ANN, but we do not know necessarily what the value of ε is, but we know that it is small.

Now that we have clearly defined the notion of an image patch, we return to the inpainting and texture synthesis literature. Efros and Leung [49] use patches to obtain a non-parametric estimation of the distribution of the value of the pixel p, which is to be synthesised. The estimation is found by searching for a set of patches which have a small distance to W_p, and inspecting the image values at the central patch positions. Once this estimation is determined, p is randomly sampled from it. This is repeated as many times as wished, and the texture is "grown" outwards from a beginning "seed". An interesting and important question is how to define patch "similarity". In [49], it is defined as a spatially weighted sum of squared differences of each patch element, $d = d_{ssd} * G$, where d_{ssd} is the vector of squared differences and G is a 2D Gaussian kernel. The method of Efros and Leung presented some ideas which turn out to be extremely useful for image inpainting, and in fact they showed some applications of their method to inpainting-like situations. An implementation of this algorithm is available in [3].

Exemplar/patch-based methods We will see shortly that the introduction of patches into the domain led to a plethora of new algorithms. There are two main reasons why this idea is so successful for inpainting. Firstly, patches are an extremely simple and efficient way of representing the local structure and texture of an image in a non-parametric (and therefore more flexible) manner. Secondly, it so happens that images present quite a lot of redundancy and repetition (indeed, this is inherent to the very nature of textures and structures), which means that it makes sense to fill in occlusions with copies of the information contained elsewhere in the same image, and in particular with patches.

The first inpainting method to explicitly use the notion of patches (inspired by the non-parametric sampling ideas of Efros and Leung) was that of Bornard *et al.* [18]. This method showed that patch-based methods can be very useful for dealing with the smoothing problem. Three methods were published in a very short space of time proposing patch-based methods : that of Bornard *et al.* [18], that of Drori *et al.* [47] and that of Criminisi *et al.* [38] (and each were not necessarily aware of the others' contributions). However, the method of Criminisi *et al.* is arguably the most well-known and widely-used, so we shall introduce patch-based inpainting methods with a brief tour of their method.

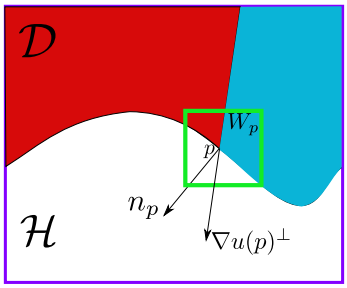


FIGURE 6.2: An illustration of the exemplar-based approach of Criminisi *et al*. [38]. Please note that this diagram is heavily inspired by the original paper of Criminsi *et al*.

Criminisi *et al*. [38] referred to patches as "exemplars". Their work presented convincing results on large occlusions, something which had not been achieved by PDE and geometry based methods. The two basic goals of image inpainting (structure and texture reconstruction) are achieved by a greedy algorithm which replaces the occluded region patch by patch. The ordering of the replacing scheme is obviously crucial. In fact, apart from the patch distance itself, it is the only component of the method which can be modified in an algorithmic sense. Therefore, Criminisi *et al*. propose the following *priority function* which determines the priority of a patch W_p centred at p :

$$P(p) = C(p)D(p), \tag{6.7}$$

where $C(p)$ and $D(p)$ are the *confidence* term and *data* term, respectively :

$$C(p) = \frac{\sum_{q \in \mathcal{N}_q \cap \mathcal{D}} C(q)}{|\mathcal{N}_p|}, \tag{6.8}$$

$$D(W_p) = \frac{|\nabla u(p)^\perp \cdot n_p|}{\alpha}, \tag{6.9}$$

where $|\mathcal{N}_p|$ is the area of $\mathcal{N}_p$, or, more precisely, the cardinal of the set of positions belonging to $\mathcal{N}_p$, α is a normalisation factor, and n_p is a unit vector which is orthogonal to the border $\partial\mathcal{H}$. Note that in this approach, $\partial\mathcal{H}$ changes throughout the algorithm, as more and more patches are replaced. Initially, the confidence term is $C(p) = 1$ for all $p \in \mathcal{D}$, and $C(p) = 0$ for all $p \in \mathcal{H}$. A visual illustration of the algorithm may be seen in Figure 6.2. The confidence term essentially stipulates that if a patch contains a high number of known pixels, then it is a patch with a high priority, although the formula is slightly more subtle than that, since the contribution of a known pixel is itself weighted by the amount of known pixels in the patch surrounding this latter pixel. The data term attempts to incorporate the structural information of the area surrounding the occlusion. This is done by using the notion of isophotes, in a similar manner as [14], which are represented in $D(W_p)$ as the orthogonal of the image gradient at p, which is projected onto n_p. Conceptually, the data term quantifies the following criterion : to what extent is an isophote arriving orthogonally to the occlusion boundary, and how strong is this isophote ? Therefore, obvious structures which seem to penetrate the occlusion head-on should be reconstructed first, as this gives the highest probability that they will be correctly joined with themselves on the other side of the occlusion.

The third "seminal" patch-based method which used the notion of image patches is that of Drori *et al*. [47]. This was published almost exactly at the same time as the work of Criminisi *et al*. This work takes a very similar approach to that of Criminisi *et al*., but does not explicitly

encourage the continuation of strong isophotes. However, their method uses a multi-resolution pyramid and considers the problem of smoothly compositing patches into the target region. As in [38], the importance of the order of filling in the occluded region is highlighted.

Perhaps the most sophisticated approach to simultaneous structure and texture reconstruction using a patch based approach is that of Cao *et al*. [33]. In this work, the authors compute a rough structure sketch based on segment detection and reconstruct this sketch with Euler spirals, extending the work of [92]. One of the goals of this work is to continue very long structures and keeping their boundaries crisp and unblurred.

The introduction of the notion of patches to inpainting led to a series of methods based on similar principles. However, most of these methods inpaint the image in a greedy manner, which means that no notion of a global coherence of the solution is considered. This can perhaps be likened to putting together small, coherent regions of a jigsaw puzzle, without ever considering that the final, whole picture is of a house !

While the previous analogy makes it look like global coherence would be very difficult to obtain, subsequent authors have produced algorithms whose goal is precisely that. We now look at those algorithms which tried to formulate the inpainting problem in terms of discrete or continuous, global optimisation.

Inpainting using shift maps and/or discrete optimisation All of the global optimisation based approaches which we look at hereafter are based on the same premise : a "good" solution is one that in some sense resembles as much as possible the content outside of the occlusion, and is coherent. This can be formulated mathematically in several ways. Let us note that one could raise the objection that the previous criterion could be respected by simply copying and pasting a large piece of image information into the occlusion. In practice, this is not possible because the use of patches implies coherency at the occlusion borders. This is a good thing, since such a solution would clearly not be visually pleasing.

In the literature, the first to formulate inpainting in terms of a *labelling* problem were Demanet *et al*. in 2003 [44]. This work introduced a notion which would later be very widely used by many other authors [8, 80, 105, 88], called *shift maps*. In actual fact, in the original paper [44], these shift maps were called *correspondence* maps but we shall use the former name, since it is more widely known. Before describing the algorithm of [44], let us define the notion of the *shift map*.

Definition 10. *We define a shift map $\phi : \Omega \rightarrow \mathbb{N}^2$ as a vector field over the support of an image. This vector field represents the spatial offsets between the central pixel of a patch and the central pixel of another patch in the image, with which it is associated.*

In practice, the association represented by ϕ often corresponds to the concept of nearest neighbours. So, for a patch W_p, its NN (or its ANN) will often be $W_{p+\phi(p)}$. As we mentioned, an alternative notion used by Demanet *et al*. is the *correspondance map*, which directly indicates the spatial location of the NN of W_p.

In [44], the image is reconstructed by copying and pasting pixels directly from the area indicated by the shift map : $u(q) = u^*(p + \phi(p))$ for all $p \in \mathcal{H}$. Therefore, the unknown is no longer u (the image), but ϕ. Demanet *et al*. introduce an energy which is optimised over the shift map. There are two important points in such a formulation. Firstly, the solution is a function ϕ in $\mathbb{N}^2$ which can have important implications concerning the possible optimisation methods. Secondly, the solution is copied and pasted directly from the known area, therefore no combinations of image information are possible. The functional of Demanet *et al*. is defined as :

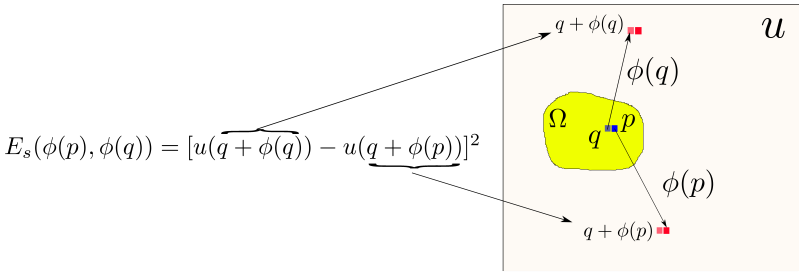


FIGURE 6.3: Visual illustration of the smooth energy of Pritch *et al.*

$$E(\phi|u) = \sum_{p \in \mathcal{H}} d^2(W_p^u, W_{p+\phi(p)}^u), \tag{6.10}$$

where $d^2(\cdot, \cdot)$ is defined as the l^2 distance. The optimisation of this energy is done with a simple algorithm which is basically a repeated "onion peel", which means that each layer of the occlusion is successively processed. When a pixel is visited, and the closest patch is found and ϕ is updated in consequence. This is done for every pixel in a one-pixel thick layer, and the next layer is then processed. This whole procedure is repeated until convergence of the energy. One important detail which is not clarified in this paper is the exact order in which the pixels are visited *inside* a layer. They are not processed in parallel, but the order is unspecified.

In [114], Sun *et al.* propose a semi-automatic method which completes occluded structures in an image using discrete optimisation techniques, and then proceeds to inpaint the occlusion using a patch-based technique similar to that of Criminisi *et al.* Furthermore, Poisson Image Editing [102] is used to hide the borders of the inpainted area. This is a technique which is widely used by other inpainting algorithms.

Komodakis *et al.* propose an approach [80] which is similar, although it does not directly introduce the notion of shift maps. In actual fact, it would seem that Komodakis *et al.* were not aware of the work of Demanet *et al.*, since they did not comment on the similarities of these approaches. This paper formulates the problem using Markov Random Fields (MRF) : each node of the MRF corresponds to a patch in $\mathcal{H}$, and these nodes are linked with edges defined by a 4-neighbourhood. Their energy over the set of labels of the nodes is optimised using *belief propagation*. As in [44], the solution is obtained by copying and pasting image information into the occlusion.

A very different approach to inpainting was taken by Hays and Efros in [62]. They propose to use information *other* than that contained in the image to inpaint. The interesting idea is to use photos of the same scene found on the internet for inpainting. This is useful in situations in which the information needed for inpainting is unavailable in the image itself, typically when occlusions are large. This kind of approach raises the question of what type of information is "legitimate" to use for the purposes of inpainting. Indeed, if the occlusion is very large, then it is often possible to put unrelated content, as long as the seams between information from different origins are not visible. In this situation, the goal of inpainting becomes even more difficult (than usual) to define. Likewise, the systematic evaluation of such methods is very complicated.

Whyte *et al.* [125] propose a similar approach. However, they extract only images of the same scene in order to avoid inserting clearly incongruous information into the occlusion. The images are registered both geometrically and photometrically with the target image, which leads to a list of possible inpainting candidates for each pixel in the occluded region and, again, to a labelling problem.

Pritch *et al*. [105] proposed a very successful method, called "Shift Map Image Editing" which used *Graph Cuts* to optimise an energy over the shift map, and in fact introduced the term "shift map". They were the first to use Graph Cuts for the inpainting problem, although it had been used as early as 2003 by Kwatra *et al*. for texture synthesis [83]. Since the Graph Cuts technique is used several times in the image and video inpainting literature [58, 64, 88, 105], we shall briefly present it before going into the details of the algorithm of Pritch *et al*.

The discrete optimisation scheme named "Graph cuts" was introduced by Boykov *et al*. in 1999 in [20]. This paper deals with approximate energy minimisation on a discrete, finite label space $\mathcal{L}$. Boykov *et al*. propose two approaches to minimising energies of the following form :

$$E(f) = E_s(f) + E_d(f), \tag{6.11}$$

with

$$E_s(f) = \sum_{(p,q)\in N} V_{p,q}(f_p, f_q), \tag{6.12}$$

where f is the solution, p and q are *sites* with which the labels are associated, and N is the set of all the pairs of connected sites. V is called an *interaction potential*. This kind of energy is quite common to the image processing and computer vision domains. For example, it closely resembles the formulation of the optical flow problem (which is, on the other hand, generally formulated in a continuous framework). The smooth term usually corresponds to some kind of regularisation, and the data term means that the solution should be close, in some sense, to a separate set of input data.

Boykov *et al*. propose two minimisation algorithms : the "α-β-swap" algorithm and the "α-expansion" algorithm. These work on *semi-metric* and *metric* interaction potentials. A potential V is metric if, for any three labels α, β and γ, it verifies the following properties :

1. $V(\alpha, \beta) = V(\beta, \alpha) \geq 0$
2. $V(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta$
3. $V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta)$

A semi-metric need only verify the first two conditions. Both algorithms solve successive binary sub-problems in order to solve the total multi-label problem. These sub-problems are formulated as the task of finding the minimum cut of a graph.

We now come back to the inpainting problem. Pritch *et al*. formulated the problem in the following manner in [105]. The sites of the problem are the pixels, and the labels correspond to the shift map ϕ. The goal is to find the shift map which minimises the following energy :

$$E = \alpha \sum_{p\in\mathcal{H}} E_d(\phi(p)) + \sum_{p\in\mathcal{H}} \sum_{q\in\mathcal{N}_p} E_s(\phi(p), \phi(q)), \tag{6.13}$$

where α is a scalar. The data term E_d is set to be very high if $p+\phi(p) \in \Omega$, and very low otherwise.

The smoothness energy term is defined as :

$$\begin{aligned} E_s(\phi(p), \phi(q)) \quad = \quad & [u(q + \phi(q)) - u(q + \phi(p))]^2 + \\ & \beta[\nabla u(q + \phi(q)) - \nabla u(q + \phi(p))]^2 \end{aligned} \tag{6.14}$$

Let us make some remarks on this formulation. The data term states simply that the inpainting solution should come from pixels belonging to the unoccluded region, which is obviously a

necessary condition. The smooth energy is rather more complex. Rather than stipulating that the shift map should be regularised in some fashion, it requires only that the *image values* be coherent. This is a subtle, but extremely important distinction. Put differently, it means that for any pixel p, the values of $u(p + \phi(q))$ for all $q \in \mathcal{N}_p$ should be as similar to each other as possible. In an even more intuitive fashion : if we start from p, we should arrive at the same value of u even if we take the paths indicated by the pixels around p. So for instance, if we want to replace a homogeneous region, we may do it with a completely irregular shift map (apart from along the borders). This is quite different to the regularisations found in optical flow, for instance.

Another formulation of the inpainting problem using Graph Cuts is given in the paper of He and Sun [64]. They propose an interesting, although potentially limited approach. They hypothesise that the shifts needed for successful inpainting only belong to a small subset of the all the possible offsets, in other words that there is a *sparsity* of offsets in the optimised inpainting solutions. Furthermore, they suggest that this subset may be estimated by using the properties of the same image. This is motivated by situations such as regular, spatially periodic structures or textures. Briefly, they see which offsets are the most common when they try to match unoccluded patches with other known patches in the same image. This gives a histogram of shifts, and they keep a certain number of the most common. This approach is interesting in that many inpainting situations are well resolved by using piecewise constant shift maps, and therefore a few shifts only. However, it is clear that this approach presents certain limits ; it vastly diminishes the flexibility of the inpainting, since the algorithm is less able to aggregate small zones together, which is often very necessary. Nevertheless, the idea of the *sparseness* of a good shift map is an interesting idea which is worth further thought.

The last approach in the image inpainting literature which uses Graph Cuts for optimisation purposes is that of Liu and Caselles [88]. Their work presents some of the most impressive inpainting results to date. This approach is quite similar to the original shift map formulation of [105], but addresses two very important points which were not discussed in the work of Pritch *et al.* :

- Dealing with border conditions for the problem ;
- The inclusion of additional descriptors at the coarsest pyramid level, to compensate for certain high frequency information which is lost at such levels.

The importance of the first point certainly needs no explanation : without border conditions, the algorithm could simply copy and paste a region of the image into the occlusion. While it is probable that the authors of Shift Map Image Editing did, in fact, use some kind of border conditions, the problem is not mentioned in [105], and requires clarification.

Finally, Aujol *et al.* [8] extended the work of Demanet *et al.*, in a more theoretical approach to the problem. The idea is to formulate a patch-based energy whose argument is the shift map, as in [44], but in a well-posed manner and in the continuous domain. In particular, they consider the solution domain defined by the set of piecewise rotations and translations of patches : as they put it nicely "the model we have in mind actually picks pieces of patches in $\mathcal{D}$ [Ω with their notation], then rotates and translates them onto $\mathcal{H}$ [A with their notation]". The authors propose several patch-based inpainting functionals and prove the existence of minimisers of these energies, although they do not give any schemes to optimise the functionals. Several very interesting points were brought up in this paper such as certain cases were global minimisers are clearly not unique.

Non-local patch-based variational formulation If we summarise the trends in image inpainting up until this point in the literature, we can see that formulating the problem in terms of PDEs was the first, natural approach. This was subsequently outperformed to a certain extent by patch-based methods which could maintain texture, while doing a reasonably good job of continuing

structures. This approach was then reformulated in terms of discrete optimisation, using notions such as correspondence maps and shift maps. However, the resolution with methods such as Graph Cuts is limited, since a proper optimisation is only carried out at a coarse resolution. It is clear that a reformulation of patch-based inpainting methods is desirable.

Let us firstly recall some notation here. The image content is denoted with u. The occluded region is noted $\mathcal{H}$ and the unoccluded region is noted $\mathcal{D}$. A patch neighbourhood centred on a pixel p is denoted with $\mathcal{N}_p$ and the corresponding patch is noted W_p.

Arias *et al*. proposed in [5] a very interesting formulation of the problem. Instead of proposing a functional over the *shift map*, the function is optimised both over the image information itself and over a *weighting function* $\omega(x, y)$ which indicates the similarity between an occluded and an unoccluded pixel. This similarity is based on a patch distance of some description. Since this is such a significant reformulation of the problem, we shall stop awhile to explain it. In [4], Arias *et al*. present a more complete analysis of their framework, in particular with respect to the existence of optimal solutions.

Their functional takes the following form :

$$E_a(u, w) = \frac{1}{h}\tilde{F}_w(u) - \int_{\tilde{\mathcal{D}}} H_w(p)dp, \tag{6.15}$$

with,

$$\begin{aligned} \tilde{F}_w(u) &= \int_{\tilde{\mathcal{H}}}\int_{\tilde{\mathcal{D}}} w(p,q)d(W_p, W_q)dqdp \\ H_w(p) &= -\int_{\tilde{\mathcal{D}}} w(p,q)\log w(p,q)dq \end{aligned}$$

and

$$w(p,q) = \exp(-\frac{d(W_p - W_q)}{h}).$$

The parameter h controls the selectivity of the weighting function w. The sets $\tilde{\mathcal{H}}$ and $\tilde{\mathcal{D}}$ correspond to dilated and eroded versions of $\mathcal{H}$ and $\mathcal{D}$, respectively.

This functional is optimised using variational analysis, alternately with respect to the image information u and the weighting function ω. The formulation is interesting for the following reasons :

- It is more general and rigorous than most other formulations :
  - Arbitrary patch sizes are included in the model
  - A general class of patch distances are considered
- The link between non-local methods (such as non-local denoising) and inpainting is quite significant.
- It is the first to propose a functional and a corresponding variational analysis based optimisation algorithm

Let us note that the first to propose such a formulation were Wexler *et al*. [123], although their goal was video inpainting, so we shall look at their work in more detail in Section 6.2.2. Another method which uses a similar approach to that of Arias *et al*. is that of Darabi *et al*. [41]. This is in fact an extension of a video inpainting algorithm [124], which we shall see further on. The algorithm is based on two repeated steps : an ANN search, and a reconstruction step. There are two major differences proposed by Darabi *et al*. Firstly, they search for ANNs across translations, rotations and scales. This is done with the Generalised PatchMatch algorithm. In

particular, this extension makes it possible to inpaint objects for which the correct patches do not exist. An example is the inpainting of the side of a round object (a lemon, to be precise) : the correct patch exists if we allow rotations. The second new element proposed by the authors is to reconstruct both the colours and the gradients of the image. In this paper, other applications than just inpainting are proposed : texture interpolation and image melding are also possible. It should be noted that considering scales and rotations slows the inpainting algorithm down. Therefore, while it is nice to have these features, it is important to consider whether they are necessary for the desired application.

6.2.2 Video inpainting

The review of the inpainting literature up until this point has purely been concerned with images. There is also a (somewhat smaller) literature devoted to *video* inpainting. Our original goal, removing defects and objects, is also geared towards videos and so it is this domain which we shall be mostly exploring. However, our tour of the image inpainting literature was necessary, and introduced some key concepts in a natural manner.

Before looking at this literature, we need to make some remarks about notation. We shall try and maintain, as far as possible, a notation which is coherent with that introduced for image inpainting. We will not create a new set of notations for the concepts, as most of them are exactly same as in the case of images. In particular, $\Omega \subset \mathbb{R}^3$ will correspond to the support of a video, and $u : \Omega \rightarrow \mathbb{R}^3$ will be the video information. A spatio-temporal position is represented by $p = (x_p, y_p, t_p)$, where t_p corresponds to the position in the temporal dimension. The patch neighbourhood is defined in the spatial and temporal directions, and thus definition of a patch does not change, since it is simply a vector defined with a neighbourhood and a set of values u. The shift map is now $\phi : \Omega \rightarrow \mathbb{N}^3$. The distance between to spatio-temporal patches W_p and W_q is still denoted as $d(W_p, W_q)$. Finally, the notion of nearest neighbours and approximate nearest neighbours remains unchanged.

As mentioned in the introduction of this Chapter, the requirements of a "good" inpainting solution will vary from domain to domain. In video inpainting, the same constraints are maintained (*i.e.* respecting structure and texture), and the following are added :

- Reconstruction of background and foreground in a coherent manner ;
- Reconstruction of objects moving behind and occluded area ;
- Reconstruction of video textures ;

Several different approaches to this problem have been proposed. These can be roughly grouped into two categories : patch-based, and object-based. In our work, we have concentrated essentially on patch-based approaches. This is because they are more generic and require fewer hypotheses concerning the content to inpaint. We shall present the patch-based literature first of all and then review the object-based methods.

Patch-based video inpainting The first paper to discuss the problem of video inpainting was that of Bertalmío *et al*. in [13]. However, the method was simply applied frame-by-frame to videos, and not tailored to the specific goals which are listed above, in particular temporal coherence.

The first true video inpainting algorithm was that of Wexler *et al*. in [123]. This is a patch-based algorithm which uses an heuristic to optimise a global *coherency* function. A fundamental element of this algorithm is the use of *spatio-temporal* (3D) patches. While this extension obviously seems natural, it is the key to reconstructing such elements as moving objects. The coherency function is :

$$E_w(u) = \sum_{p \in \mathcal{H}} \min_{q \in \mathcal{D}} d(W_p, W_q). \tag{6.16}$$

This energy specifies that each patch in the occluded area should resemble as much as possible its *most similar* patch in the unoccluded area. It is interesting to note that this energy was first proposed for image inpainting by Demanet *et al*. [44] as explained in the previous section, and later used for texture synthesis by Kwatra *et al*. [82]. The minimisation of the energy in Equation (6.16) consists of a repeated use of two steps : a nearest neighbour search, and a reconstruction of the video content. These two steps are carried out alternatively on the current solution, u, until it converges.

The nearest neighbour search in the video setting is not a trivial problem. An exhaustive search would be far too costly to carry out. Therefore an algorithm for the retrieval of ε-ANNs is used [6]. This greatly accelerates the search, although it is still quite slow. For example, Wexler *et al*. report an ANN search of one hour for a video of $340 \times 120 \times 100$, with an occlusion of 422,000 pixels. This is obviously still very long, since the search and reconstruction steps are iterated several times. The reconstruction step consists of a weighted mean. Given a shift map ϕ, each pixel is reconstructed with the following formula :

$$u(p) = \frac{\sum_{q \in \mathcal{N}_p} \alpha_p^i s_p^q u(p + \phi(q))}{\sum_{q \in \mathcal{N}_p} \alpha_p^q s_p^q} \tag{6.17}$$

with

$$s_p^q = \exp - \frac{d^2(W_q, W_{q+\phi(i)})}{2\sigma^2},$$

where s_p^i is the weight given to $u(p + \phi(q))$. This is done independently for each colour channel. The value α_p^i is a weight given to the pixel i which reflects the distance of i to the occlusion boundary. More precisely, $\alpha_p^i = \gamma^{dist(i)}$, where $dist(p)$ is the distance from p to the occlusion boundary, and γ is set to 1.3. Finally, σ is defined as the 75th percentile of all the distances $d(W_i, W_{i+\phi(i)}), i \in \mathcal{N}_p$.

Chronologically, the next approach to video inpainting is that of Patwardhan *et al*. [99]. The goal of this work is the correct inpainting of moving objects (mostly people) in static backgrounds. It may be seen as a natural extension of the method of Criminisi *et al*. [38]. This method inpaints the occlusion sequentially, in a greedy fashion. The video is segmented into moving foreground and static background, and the two are inpainted using different algorithms. The background is inpainted using temporal information where possible, that is to say if the background becomes visible at some point. The areas corresponding to positions which are never revealed are inpainted using an identical technique to that of [38]. During the inpainting of the moving foreground, the data term (Equation (6.9)) is modified : the isophote term is replaced with a simple indicator function which shows whether a pixel belongs to the moving foreground or not. The confidence term (Equation (6.8)) is not modified. The advantage of this method is that it is quite simple conceptually. Also, it is derived from a method which has been extensively tried and tested in the image inpainting setting. However, in the spatio-temporal case global consistency is paramount, and humans are very sensitive to temporal inconsistencies especially with respect to moving objects. Therefore, the greedy nature of the algorithm likely makes it unreliable. In particular it is clear that such a method will not function without restriction of the search space, which is done in this case using foreground/background separation.

Patwardhan *et al*. extended the previous approach in [100]. In this paper, moving cameras are dealt with by realigning each frame with respect to a camera motion estimated by block matching.

The camera motion between a frame t and a frame $t+1$ is defined as the median value of the motions estimated by block matching. The realigned background is directly copied and pasted from the corresponding positions. This supposes that the realignment is perfectly executed, which is a drawback. Another significant change in this version concerns the data term. This is modified to reflect the direction of the moving objects : the objects moving perpendicularly to the occlusion boundary have a large data term (increasing the priority), whereas those moving tangentially to the boundary have a low data term. The data term is written as :

$$D(W_p) = \frac{|(\nabla M_c^{\perp})_p \cdot n_p|}{\alpha},$$

where ∇M_c is the *motion confidence mask*, which says whether a pixel is part of the moving foreground or the background.

Another approach to the task of video inpainting was proposed by Kokaram *et al*. in [79]. This method reconstructs motion vectors in occluded areas so that known video information may be propagated into the unknown region. The reconstruction of the motion is done in a Bayesian fashion. Kokaram *et al*. enforce a pixel brightness constancy criterion and a temporal smoothness criterion to produce a motion estimation in the occluded area. Once this motion is determined, the occluded pixels are reconstructed using a weighted interpolation of the available pixels at the positions indicated by the motion estimation. The goal of this approach is to remove thin structures from videos, rather than generic objects. Indeed, it is clearly a difficult problem to reconstruct motion vectors in large areas or in occlusions which do not move greatly. In [113], Shiratori *et al*. proposed a video inpainting algorithm based on inpainting patches of optical flow. This is an interesting idea, which can be linked to the general concept of inpainting by constructing reliable pixel trajectories inside an occluded area. The authors construct an optical flow outside the occluded area using the Lucas-Kanade algorithm. For two optical flow vectors $\vec{v}_1$ and $\vec{v}_2$, distance is defined as $d(\vec{v}_1, \vec{v}_2) = 1 - \cos(\theta)$, where θ is the angle between $\vec{v}_1$ and $\vec{v}_2$. The patch distance is simply the sum of these distances over the patch. The motion vectors are copied patch by patch in a greedy fashion into the occlusion. Once this is done, a graph is established where the edges are given by the motion vectors. The colour of the pixels is found by solving a system of equations where each pixel's colour is defined by the colours of the connected pixels in the graph. This approach is interesting for several reasons, and in particular because it does not *per se* require periodicity of motion. For example it is possible to imagine that an object slowing down could be correctly reconstructed. The method is also contrast independent (as long as the optical flow is robust to this), which can represent a significant advantage. However, a clear drawback is an inability to process temporally long occlusions. Spatially speaking, it can deal with large occlusions, which is useful for such tasks as film restoration, but inpainting motion vectors is a tricky task for motions with long periods.

Shen *et al*. proposed in [111] a motion manifold-based approach to video inpainting. This assumes strict periodicity of object motion, and segments the video into foreground and background. The segmentation is done by manually specifying the foreground in the first frame, and then tracking this with a mean-shift tracker. The idea is to inpaint pixels along the manifolds of each foreground object. The manifolds are obtained by curve fitting, although the precise method is not specified. Then, each path indicated by a manifold is inpainted using a set of patches from the unoccluded region. The patches are directly copied and pasted onto the curve defined by the manifold, using a shortest path algorithm. In effect, several "anchor points" are specified, and the resulting series of patches is solved by finding the minimal path through these anchor points with a patch distance as a cost function. Each anchor point is thus designated to a single patch outside the

occlusion. The background is inpainted either by copying and pasting pixel values from a frame in which the point is unoccluded or by using the work of Criminisi *et al*. (a technique which is very often used for inpainting the background).

Raimbault *et al*. propose a motion-based approach in [107]. The inpainting is carried out in a greedy fashion, pixel-by-pixel, with an ordering based on a priority term which is updated dynamically throughout the algorithm. The video is inpainted using patches which are constrained to belong to a certain trajectory which defines the NN search space for each patch. As in the case of [111], an advantage of this approach is the acceleration of the search for NNs. However, creating a reliable trajectory for generic moving objects to be reconstructed during inpainting remains a difficult task.

The final video inpainting approach which we will look at was proposed by Granados *et al*. and extended the idea of working with shift maps to videos. This algorithm is the first to show results on high-resolution videos. The formulation is very similar to that of Pritch *et al*. They add a multiplicative term in the smoothness term :

$$E_s(\phi(p),\phi(q)) = \begin{cases} 0 \quad \text{if} \quad \phi(p)=\phi(q), \\ h(p,q)\gamma(p,q) \quad \text{otherwise}, \end{cases} \tag{6.18}$$

where $\gamma(p,q)$ is a weight which is maximal on the occlusion border and decreases towards the centre of the occlusion. The term $h(p,q)$ is a coherency term similar to that of Pritch *et al*. . [105] :

$$\begin{aligned} h(\phi(p),\phi(q)) = \quad & \big(|| \quad u(p+\phi(p)) - \quad u(p+\phi(q))||_2^2 \quad + \\ & || \quad u(q+\phi(p)) - \quad u(q+\phi(q))||_2^2\big)^{\Psi} + \\ \beta \quad & \big(||\nabla u(p+\phi(p)) - \nabla u(p+\phi(q))||_2^2 \quad + \\ & || \ \nabla u(q+\phi(p)) - \nabla u(q+\phi(q))||_2^2\big)^{\Psi} + \lambda \end{aligned} \tag{6.19}$$

Furthermore, Granados *et al*. deal explicitly with border conditions, which is a very big omission in the work of Pritch *et al*. In particular, they specify that pixels on the *outer* border of the occlusion (these pixels are *not* occluded) should have shift maps which are not null. Finally, Granados *et al*. give precise details concerning their implementation parameters, something which is of vital importance in video inpainting. The results of this algorithm are very impressive, and are shown on high definition images (around 1200×900 pixels) of around seven seconds. The disadvantage of the algorithm is the extremely long execution times (around 90 hours for the previously mentioned example). Furthermore, the algorithm requires manual segmentation of the video into background and moving objects. Each moving object which needs to be reconstructed has to be manually indicated with a bounding box in each frame. While the algorithm of Wexler *et al*. can be accelerated with new ANN search techniques, it is difficult to see how the execution time of the Graph Cuts based formulation may be reduced, since the search space is already reduced as far as possible. In the end, the complexity of the graph cuts algorithm presents a significant barrier to practical use of the algorithm. Also, if it were to be used in real circumstances, the manual indication of bounding boxes for moving objects would have to be replaced by some sort of automatic segmentation, which is likely to be quite difficult.

"Object-based" video inpainting The term "object-based" refers to the idea that moving objects can be inpainted using prior knowledge of the nature of the objects, in particular if they are human beings, as is often the case.

The first work to be done in this area was that of Jia *et al*. [68]. In this algorithm, the background and moving foreground are segmented, as in most of such algorithms. The moving foreground is separated into "movels" that is to say segments of periodic moving pixels. The idea is to identify the periodicity of moving objects and in particular the correct beginning and end points of the period which allow the movel to be best fitted into the occlusion. The actual fitting is done by warping the movel using a homographic transformation. The parameters of the homography are estimated using a minimisation between the warped movel and the actual video. A considerable disadvantage of this algorithm is the strict periodicity of motion supposed. Furthermore, the user is required to roughly segment the video into layers of coherent motion, and it is also mentioned that the identification of the movels can require manual intervention. This indicates that the method is quite fragile, and requires the success of many different steps for a good result to be produced.

In [37], Cheung *et al*. propose an object-based approach similar to that of Jia *et al*., except that the movels are not modified in any way. The best unoccluded spatio-temporal segment of the moving object is chosen and realigned and pasted into the occlusion. As is common, the method of Criminisi *et al*. is used to inpaint the background. This method may be seen as a simpler version of the work of Jia *et al*.

Shih *et al*. proposed in [112] a method to modify the motion of objects so that the resulting video is "falsified". Although this is not video inpainting in the strict sense, it has some common points with object-based video inpainting. As always in these methods, the video is segmented into moving foreground and background. The moving foreground is separated into different moving objects. Stick representations of the objects are obtained by a morphological thinning operation. This representation is used to guide the reconstruction of the objects. The authors consider several types of "falsification" such as speeding up motion, slowing down and changing direction. In the case of slowing down, motion interpolation is necessary and done with a non-parametric patch-based method.

6.2.3 Textures in videos

We have looked at textures in the case of images, but *video textures* (or *dynamic* textures) have also been studied. We shall see further on that such elements are quite common in video inpainting. Therefore, we take a short tour of the literature devoted to this particular domain.

The work of Szummer and Picard [115] was the first to investigate temporal textures, and provided the following definition : "temporal textures are textures with motion. Examples include wavy water, rising steam." Szummer and Picard use a linear spatio-temporal autoregressive model to predict the value of a signal at a point in space and time (a pixel, here). Both the choice of the model and the estimation of its parameters is done adaptively with respect to the current video texture under consideration. With this, they can either synthesise or recognise video textures.

In [109], Schödl *et al*. were the first to coin the specific term "video textures". They proposed a method which would create an infinite video loop based on an input video. They summarised their method well by stating that "the general approach ... is to find places in the original video where a transition can be made to some other place in the video clip without introducing noticeable discontinuities." An important point concerning this algorithm is that they do not synthesise any new content *per se*. Rather, they rearrange the identified parts of the image sequence in a manner which has some element of randomness. To produce the final result, they sew the image elements together with various techniques (cross-fading, "morphing", *etc*.).

Kwatra *et al*. presented a graph cuts based image and video texture synthesis method in [83] ("Graph-Cut Textures). This approach stitches together pieces of images by finding an optimal

seam for the image or video pieces to be put together. This is one of the first papers to introduce the graph cuts technique for image and video synthesis and inpainting.

Agarwala *et al*. extend the idea of stitching pieces of images together to create video texture panoramas [2]. They first register the image sequence and determine a coordinate system onto which all the video content is mapped. We should note that their method differs from that of Schödl *et al*. in that they determine a *mapping* (analogously to the shift map) from the pixels in the input video to an output video, rather than an *ordering* of pre-identified image sections, although one could argue that the approach of Agarwala *et al*. simply redefines the regions to be individual pixels. Each pixel may only be shifted in time, and not space. Finally, they determine the mapping with discrete optimisation techniques.

Eisemann *et al*. [50] proposed a free viewpoint video texture mapping technique which takes the input of several cameras, and allows the user to view them from any viewpoint. While this is not explicitly texture synthesis, the process of projecting textures and rendering the video for a certain viewpoint makes it worth mentioning.

Levieux *et al*. [84] proposed an approach closely linked to that of Schödl *et al*. to generate video textures which can be interactively inspected from any viewpoint along a curve. An additional component of this algorithm with respect to the work of Eisemann *et al*. is the capacity to synthesise parts of the video texture using a sequence obtained from a single camera (which makes it closer to the task of video inpainting).

From this quick review of some of the video texture literature, it is clear that the most successful approach is to identify certain elements in the video and stitch them together seamlessly. It is quite clear that such methods are extremely similar to discrete labelling approaches used by shift maps based inpainting techniques [44, 58, 64, 105].

6.3 Choosing the best approach to video inpainting

Having now done an extensive review of both the image and video inpainting literatures, we would like to recap the approaches to inpainting which seem most promising and viable. We recall that our goal is to produce an inpainting algorithm which can deal successfully and robustly with many different situations, and with reasonable execution times (something which traditionally has been an obstacle in video inpainting). Let us first note note that we set aside object-based approaches, since they seem too specifically adapted to certain situations, and not generic enough. Furthermore, the flexible patch-based formulations of the video inpainting problem are much more attractive and have a more solid theoretical basis. In terms of patch-based approaches, then, there are basically four options available :

- Develop a greedy approach inspired by that of Patwardhan *et al*. [100] ;
- Use the idea of iterative nearest neighbour search and video reconstruction of Wexler *et al*. [124] ;
- Try to recreate the *motion* inside the occluded areas, as in [79, 111, 113] ;
- Use a shift map based approach as done by Granados *et al*. [58] ;

The first option has the advantage of being directly modified from the seminal image inpainting approach of Criminisi *et al*. [38]. This is an advantage because this algorithm has been extensively "tried and tested" and, on balance, having experimented with many different image inpainting approaches, the approach of Criminisi *et al*. seems to be one of the best in terms of results, execution time and adaptability. Therefore, it would be reasonable to assume that it could work in the case of videos. However, we set aside this option since such greedy approaches are not conducive to the global coherence needed for such tasks as reconstructing moving objects. This lack of robustness is

confirmed by the fact that this method requires a foreground/background separation. Furthermore, if we are resigned to using segmentation it is preferable to use a Graph Cuts based approach, such as that of [58], since the performance of their method has been proven on difficult, high-resolution videos.

The major challenge of the second type of approach (that of Wexler *et al.*) is long execution times, due to the high computational requirements of the nearest neighbour search. Indeed, this is such a problem that very few people have implemented the method, or been able to experiment with it. Quite a few implementation details are not specified in [123] or [124], making it difficult to reproduce the results. To the best of our knowledge, only the following authors have implemented and shown results of the work of Wexler *et al*. on their own videos : Bugeau *et al*. [31] and Granados *et al*. [58]. An implementation is available in the case of images only (see [93]). Therefore if we choose this option, we must make sure before anything else that we have access to a good NN or an ANN search algorithm.

The third type of approach, first explored by Kokaram *et al*. in [79] uses motion estimation to guide video inpainting. The problem with this is determining motion inside the occlusion is very difficult. Also, in the case of the work of Shen *et al*. a high quality segmentation is required so that several trajectories can be estimated. The clear advantage of such approaches is the increase in quality and the acceleration a patch-based method will have if the NN search space is correctly restricted. In the case of Raimbault *et al*. [107], good results are shown on examples where the background must be reconstructed. However, such trajectories are likely to be much more difficult to determine if moving objects are considered.

Finally, if we are to pursue the last possibility (discrete optimisation), the biggest problem is clearly that of execution time. Indeed, this is such a big problem that even in the case of images [105], a true optimisation is only carried out at the coarsest pyramid level, with only very small corrections being tested at finer levels. This constraint is obviously even more important in the case of videos [58]. In this case, not only is the search space drastically reduced at finer levels, but the video must be correctly and reliably segmented into several layers of background and moving foreground objects. For the moving objects, Granados *et al*. do this manually with a loose bounding box placed around the each object by the user. Even with this considerable restriction of the search space, the method of [58] can take up to 90 hours for the reconstruction of *each* separate moving object, in a high definition video ($1120\times720\times200$). The essential point here is that it is difficult to see how the shift maps based approach could be speeded up, given that the Graph Cuts algorithm cannot really be accelerated.

We summarise this information concerning the video inpainting state-of-the-art in Table 6.1.

In light of this information, we choose to use a *non-local patch-based variational* approach, as taken by Arias *et al*. [5] for images and Wexler *et al*. [124] for videos. This approach is generic, flexible and has also been used with success in other domains such as texture synthesis [82]. The core of our algorithm shall be based on the optimisation of the global patch-based functional in Equation (6.10) which was first proposed in the framework of discrete optimisation by Demanet *et al*. [44] and then proposed by Wexler *et al*. in [124]. This functional is simply the sum of the distances between each occluded patch and its nearest neighbour in the unoccluded area.

Before going into the details of the algorithm which we propose, we would like to make a short detour to discuss some more theoretical points of certain inpainting approaches in the literature. These comments concern two of the general approaches to inpainting : *shift map* based inpainting and non-local patch-based variational inpainting. For readers who wish to skip directly to the proposed algorithm, see Section 6.5.

Method	Advantages	Disadvantages
Iterative patch-based Wexler *et al*. 2004, 2007	- Generic - Automatic	- Very slow - Several details not given
Greedy patch-based Patwardhan *et al*. 2005, 2007	- Quite simple to implement - Relatively fast	- Segmentation required - No global optimisation
Motion estimation based Shen *et al*. [111], Shiratori *et al*. [113], Raimbault *et al*. [106, 107]	- Search space is reduced	- Creating coherent pixel trajectories in generic situations is difficult
Graph Cuts patch-based optimisation Granados *et al*. 2012	- Generic - High definition results	- Manual object segmentation - Very slow

TABLE 6.1: Table summarising the different patch-based video inpainting approaches in the literature.

6.4 Inpainting energies

6.4.1 Link between the formulation of Arias *et al*. and Shift Maps

In the inpainting literature, two of the main approaches which optimise a global objective functional include non-local patch-based methods [5, 124], which are closely linked to Non-Local Means denoising [28], and shift map-based formulations such as [58, 88, 105]. We will show here that the two formulations are closely linked, and in particular that the shift map formulation is a specific case of the very general formulation of Arias *et al*. [5].

As far as possible we keep the notation introduced previously. In particular p is a position in $\mathcal{H}$, q is a position in $\mathcal{N}_p$ and r is a position in $\tilde{\mathcal{D}}$. In its most general version, the variational formulation of Arias *et al*. is not based on a one-to-one shift map, but rather on a positive weight function $w : \Omega \times \Omega \to \mathbb{R}^+$ that is constrained by $\sum_r w(p,r) = 1$ to be a probability distribution for each point p. Inpainting is cast as the minimisation of the following energy functional (that we rewrite here in its discretised version) :

$$E_{\text{arias}}(u,w) = \sum_{p\in\mathcal{H}} \left[\sum_{r\in\tilde{\mathcal{D}}} w(p,r) d^2(W_p, W_r) - \gamma \sum_{r\in\tilde{\mathcal{D}}} w(p,r) \log w(p,r) \right], \tag{6.20}$$

with γ a positive parameter. We have used r to denote the positions in $\mathcal{D}$ instead of q because we will be using q later in the discussion. The first term is a "soft assignment" version of (6.16) while the second term is a regulariser that favors large entropy weight maps.

Arias *et al*. propose the following patch distance :

$$d^2(W_p, W_r) = \sum_{q\in\mathcal{N}_p} g_a(p-q)\varphi[u(q) - u(r+(p-q))], \tag{6.21}$$

where g_a is the centred Gaussian with standard deviation a (also called the intra-patch weight function), and φ is a squared norm. This is a very general and flexible formulation.

Arias *et al*. optimise this function using an alternate minimisation over u and w, and derive solutions for various patch distances. Let us choose the ℓ^2 distance for $d(W_p, W_r)$, as is the case in many inpainting formulations (and in particular the one which we use in our work). In this case, the minimisation scheme leads to the following expressions :

$$w(p,r) = \frac{1}{Z_p} \exp\left(-\frac{d^2(W_p, W_r)}{\gamma}\right), \tag{6.22}$$

$$u(p) = \sum_{q \in \mathcal{N}_p} g_a(p-q)\Big(\sum_{r \in \tilde{\mathcal{D}}} w(q,r)u(r+(p-q))\Big). \tag{6.23}$$

The parameter γ controls the selectivity of the weighting function w. In the limit case where $\gamma \to 0$, each weight function $w(p,.)$ must collapse to a single Dirac centered at a single match $p + \phi(p)$:

$$w(p,q) = \begin{cases} 1 & \text{if } q = p + \phi(p) \\ 0 & \text{otherwise.} \end{cases} \tag{6.24}$$

If in addition, we consider that the intra-patch weighting is uniform, in other words $a = \infty$, the cost function E_{arias} reduces to :

$$E_{\text{arias}}(u, \phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}_p} \|u(q) - u(q + \phi(p))\|_2^2, \tag{6.25}$$

which is the formulation of Wexler *et al*. [124].

Rewriting (6.23) in the particular case just described, yields the optimal inpainted image

$$u(p) = \frac{1}{|\mathcal{N}_p|} \sum_{q \in \mathcal{N}_p} u(p + \phi(q)), \ \forall p \in \mathcal{H}, \tag{6.26}$$

as the (aggregated) average of the examples indicated by the NNs of the patches which contain p. Suppose that this aggregation can be reduced to a single shift-map :

$$u(p) = u(p + \phi(p)), \ \forall p \in \mathcal{H}, \tag{6.27}$$

as is the case in the shift map-based formulations.

Then the functional becomes :

$$E_{\text{arias}}(u, \phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}_p} \|u(q + \phi(q)) - u(q + \phi(p))\|_2^2, \tag{6.28}$$

which effectively depends only upon ϕ. We note that using the reconstruction of Equation 6.27 poses problems on the occlusion border, but we ignore this here for the sake of simplicity and clarity. This problem can be avoided by allowing the pixels belonging to $\tilde{\mathcal{H}}$ to be inpainted, as is done by Granados *et al*. in [58].

If we look at the shift map formulation proposed by Pritch *et al*., we find the following energy functional over the shift map only :

$$E_{\text{pritch}}(\phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}_p} \Big(\|u(q) - u(q + \phi(p))\|_2^2 + \|\nabla u(q) - \nabla u(q + \phi(p))\|_2^2\Big). \tag{6.29}$$

Let us consider the first part, concerning the image colour values. Since we have $u(p) = u(p + \phi(p))$, we obtain again :

$$E_{\text{pritch}}(\phi) = \sum_{p \in \mathcal{H}} \sum_{q \in \mathcal{N}(p)} \|u(q + \phi(q)) - u(q + \phi(p))\|_2^2. \tag{6.30}$$

Thus, the shift map functional is linked to the variational functional under the following conditions :

- $d^2(W_p, W_p) = \sum_{q \in \mathcal{N}_p} ||u(p) - u(p + \phi(q))||_2^2$
- $\gamma = 0$
- The intra-patch weighting function g_a is uniform ($a = \infty$)
- $u(p) = u^*(p + \phi(p))$.

This short note on the link between the two formulations shows that the same functional is present in both, therefore it may be logical to expect similar results. However, one should keep in mind that they are certainly not equivalent, for reasons such as the difference in optimisation methodology, the presence of a gradient term in the formulation of Pritch *et al*, and the fact that the variational reconstruction (6.26) may be smoother than the shift-map reconstruction (6.27).

6.4.2 Optimality questions : a toy binary inpainting setting

Questions concerning optimality in inpainting are, generally speaking, very difficult to answer. However, we can at least say that the inpainting problem is an ill-posed problem, implying that if a solution exists then it may not be unique. Some intuitive and interesting examples demonstrating the non-uniqueness of the solutions may be found in [8].

For our purposes, some important optimality questions are the following :

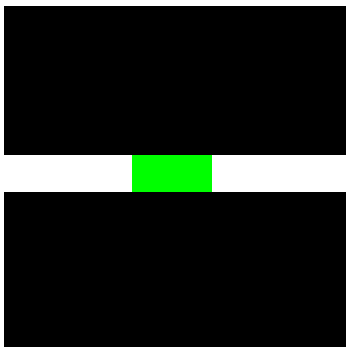
- Does a global minimiser of the inpainting energy exist, and is it unique ?
- If so, is the global minimum of the problem actually a solution which we consider to be "good" ?
- Does the inpainting scheme converge to a global or local minimum ?

As mentioned, these questions are difficult to answer. In particular, the second question seems almost impossible to address, without some reliable and objective method of evaluating inpainting solutions. With respect to the last point, the implication is that a global optimum should be "better" than a local optimum, however we note that even this is not certain : a locally optimal solution could in fact be more visually plausible than the global minimum !

In order to look at some of these questions in greater detail, we carried out some inpainting experiments in a very specific setup. We recall that we intend to use a non-local patch-based variational approach for our inpainting algorithm. Let us also recall the energy functional which we intend to use :

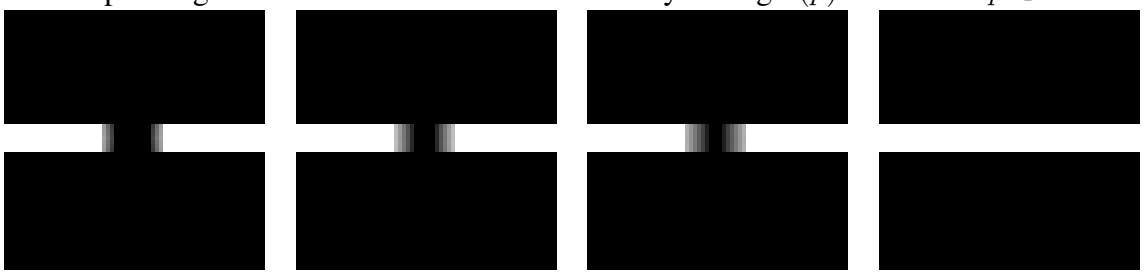
$$E_w(v) = \sum_{p \in \mathcal{H}} \min_{q \in \mathcal{D}} d(W_p, W_q). \tag{6.31}$$

The image to inpaint consists of a black background with a white, horizontal band which is occluded. Figure 6.4 illustrates this experiment. The advantage of this simple situation is the fact that we know what the global minimum is : an image where the white band is perfectly joined up from left to right. This is because the patches in the occlusion of such an image would be perfect copies of their NNs outside the occlusion. In particular, it is also clear that the solution with respect to the functional of Equation (6.16) is unique and the energy of this solution is 0. Another interesting point about this case is that the shift map ϕ which leads to this solution is



Original image (occlusion in green)

Inpainting results. The solution is initialised by setting $u(p) = 0$ for all $p \in \Omega$



Patch size : 5×5 Patch size : 7×7 Patch size : 9×9 Patch size : 11×11

FIGURE 6.4: Example of inpainting a simple structure using a non-local patch-based variational approach. We initialise the occluded region to a "bad" solution, that is black everywhere.

certainly not unique ! This means that the "shift map" formulation of the inpainting problem does not lead to a unique solution in this case, whereas the non-local patch-based variational approach does.

We used a simplified version of Wexler's algorithm to inpaint the image. This algorithm is shown in Algorithm 3.

Data: Input signal u, occlusion $\mathcal{H}$
Result: Inpainted signal

$u \leftarrow$ Initialisation(u);
while u *not converged* **do**
  $\phi \leftarrow$ NearestNeighbourSearch($u, \mathcal{H}$) with partial patch comparison;
  $u \leftarrow$ Reconstruction(u, ϕ) with unweighted mean;
end

Algorithm 3: A simplified version of Wexler's inpainting algorithm used for 1D binary inpainting. In particular, we use a full search for exact NNs and an unweighted mean for reconstruction. Also, no multi-resolution scheme is used here.

We used the exact NNs, to avoid an ANN search algorithm having any influence on the results. Also, we use an unweighted mean based reconstruction scheme (this is coherent with our discussion in Section 6.4.1). We have initialised the occlusion with what we consider to be a "bad" solution, by setting all the occluded pixels to 0.

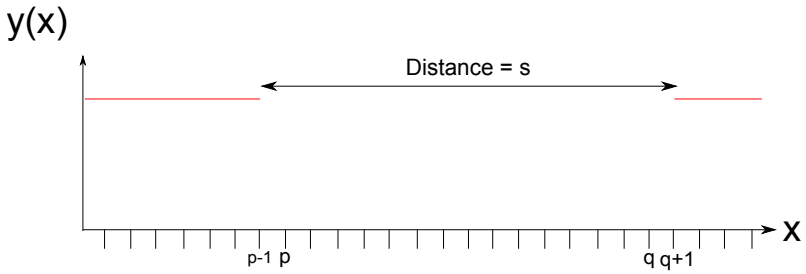


FIGURE 6.5: A simple 1D discrete signal to be inpainted. The signal is equal to 1 everywhere apart from in the occlusion, where it is unknown.

The main result of these experiments is that we observe, empirically, that the algorithm can get stuck in local minima depending on the occlusion size. In light of such results, two important questions appear. Firstly, is there a theoretical reason for the apparent local minimum of the algorithm ? In other words, can we find a stable state of the algorithm which corresponds to the situations in Figure 6.4 ? Secondly, if this is indeed the case, what is the critical size of the occlusion with a given patch size which guarantees that the algorithm will converge in this situation to the global minimum ? This question is of significant practical interest, since it would give a theoretical basis for choosing the number of pyramid levels used for inpainting.

Binary, 1D inpainting To explore this question in a more rigorous manner, we shall consider inpainting in an even more simplified situation. We shall consider a 1D signal which contains a simple structure, and we and try to join up the two edges of this structure. The structure will be represented by a signal which is equal to 1 everywhere, apart from inside the occlusion, where it is unknown (similarly to Figure 6.4). In this situation, it is obvious that the global maximum is a function which is equal to 1 everywhere. The general approach will be to initialise the signal to a "bad" solution in the occlusion, then to apply the inpainting algorithm repeatedly and see if a local or global minimum is reached. For this we need one additional hypothesis : only two patches are available for inpainting, a patch which is equal to 0 everywhere and a patch which is equal to 1 everywhere. For this reason, we refer to this setting as "binary" inpainting.

More formally, let us consider a one-dimensional signal, $y : \mathbb{N} \rightarrow \mathbb{R}$. That is, while x is discrete, $y(x)$ is a real number. We have chosen x to be discrete since this is the closest situation to real inpainting. We use the following hypotheses :

- We have only two patches, of length N, to use for inpainting, Ψ_0 and Ψ_1, with :
  - $\Psi_0(i) = 0, \quad \forall i = 1 \cdots N$
  - $\Psi_1(i) = 1, \quad \forall i = 1 \cdots N$
- The patch size N is odd, so that there is an integer h such that $N = 2h + 1$.
- We use an *unweighted mean* for reconstruction, instead of the weighted mean of Equation (6.17).

As previously mentioned, we wish to inpaint an occluded segment of a signal y which is equal to 1 everywhere apart from inside the occlusion where the signal is unknown. In this very simple situation, the globally optimal inpainting solution is clear : y should be equal to 1 everywhere, and the energy of this solution is zero : each patch in the occlusion has an exact correspondent, Ψ_1. Otherwise, we will have some patches which will have no exact correspondent among Ψ_0 or Ψ_1, and therefore the inpainting energy will be non-zero. An illustration of this situation can be seen in Figure 6.5.

The main goal of our analysis is to predict the evolution of y when the inpainting scheme is repeatedly applied, and in particular to see if the algorithm can be stuck in local minima or converges to a global minimum. The main idea to achieve this goal is to reformulate the inpainting algorithm in a series of convolution and thresholding operations. To do this, we look at the two steps of our inpainting algorithm, nearest neighbour search and reconstruction.

Let us define the function :

$$g(x) = \begin{cases} 0 & \text{if the nearest neighbour of the patch centred at } x \text{ is } \Psi_0 \\ 1 & \text{if the nearest neighbour of the patch centred at } x \text{ is } \Psi_1 \end{cases}$$

This is similar to the shift map ϕ in the real inpainting algorithm, but we give it a different name in this case, since it is not a *shift*. It will be of great use to consider the evolution of this function rather than y to see the behaviour of the inpainting algorithm.

Let us define an averaging kernel of size N equal to $\omega = \frac{1}{N}$ everywhere. We can reformulate the nearest neighbour search in the following manner :

$$g(x) = S[(y * \omega)(x), \frac{1}{2}], \tag{6.32}$$

where

$$S[b, a] = \begin{cases} 0 & \text{if } b < a \\ 1 & \text{otherwise} \end{cases} \tag{6.33}$$

We explain this in the following manner. Let us consider a patch W_p centred at the position p. We wish to know whether the nearest neighbour of W_p is Ψ_0 or Ψ_1, or equivalently, $g(p) = 0$ or $g(p) = 1$. We know that if $g(p) = 0$, we need :

$$\begin{aligned} \sum_{i=-h}^{h} [W_p(p+i) - 0]^2 &< \sum_{i=-h}^{h} [W_p(p+i) - 1]^2 \\ \iff \sum_{i=-h}^{h} [W_p(p+i)]^2 &< \sum_{i=-h}^{h} [(W_p(p+i))^2 - 2W_p(p+i) + 1] \\ \iff \sum_{i=-h}^{h} W_p(p+i) &< \frac{N}{2} \\ \iff \frac{1}{N} \sum_{i=-h}^{h} W_p(p+i) &< \frac{1}{2}. \end{aligned} \tag{6.34}$$

This shows us that the nearest neighbour is Ψ_0 if the average value of W_p is less than $\frac{1}{2}$, otherwise the nearest neighbour is Ψ_1. In terms of the function g, this corresponds to a convolution of W_p with ω, followed by the thresholding operation.

Next, concerning the reconstruction scheme, since we are using an unweighted mean ($y(x) = \frac{1}{N} \sum_{i=-h}^{h} g(p+i)$ it is very easy to see that $y(x) = (g * \omega)(x)$. To resume, we have :

$$\begin{cases} g(x) &= S[(y * \omega)(x), \frac{1}{2}] \\ y(x) &= (g * \omega)(x). \end{cases} \tag{6.35}$$

Thus, we can say, that at a stable inpainting solution, we have :

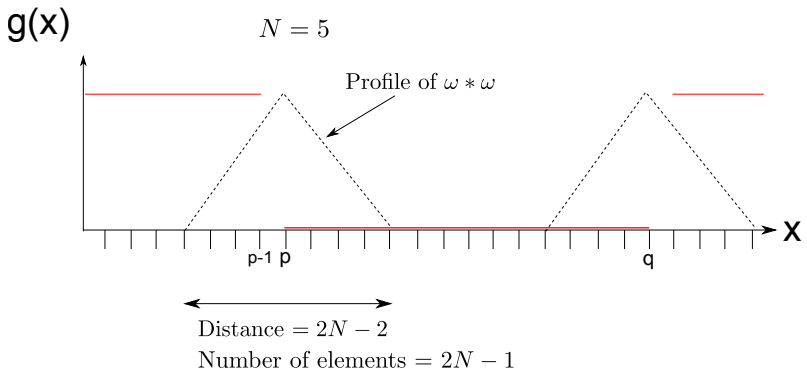


FIGURE 6.6: Illustration of the convolution of $g_{p,q}$ with $\omega * \omega$

$$g(x) = S[(g * \omega * \omega)(x), \frac{1}{2}], \forall x \in \mathbb{N}. \tag{6.36}$$

This equation will allow us to study the behaviour of the algorithm much more easily.

We now present some of the results in this situation. First of all, we shall consider the (further !) simplified case where *all* of the points are inpainted, since the situation is slightly more complicated when we consider fixed conditions outside of the occlusion. In such a situation, we can say the following :

Proposition 1. *Define a discrete, binary signal $g_{p,q}(x)$, such that $g_{p,q}(x) = \begin{cases} 0 & \text{if } p \leq x \leq q \\ 1 & \text{otherwise} \end{cases}$. If $p - q \geq n$, this function is stable under the inpainting scheme, and therefore the function $g_{p,q} * \omega$ is also stable.*

Note again that this in the case where *all* of the pixels in y are inpainted, not just those in the occlusion.

Proof This is relatively easily to prove. Look at Figure 6.6. We wish to know whether at each point, the function $g_{p,q}$ will change after applying the inpainting scheme. In particular, we wish to calculate $g_{p,q}(p)$ and $g_{p,q}(p-1)$, and likewise for $g_{p,q}(q)$ and $g_{p,q}(q+1)$.

We use the hypothesis that $p - q \geq n$, therefore we can safely calculate the value of $g_{p,q}$ at p and $p-1$ without having to take into account the value of $g_{p,q}$ at $q+1$. Based on the symmetrical nature of the kernel ω, we see that, for the point p the convolution of g_p with $\omega * \omega$ will be less than $\frac{1}{2}$, and similarly at $p-1$, the value will be greater than $\frac{1}{2}$. A similar argument can be made for the positions q and $q+1$. This proves that the function $g_{p,q}$ does not change with the inpainting scheme, and therefore neither does $y = g_{p,q} * \omega$.

This information is quite significant, since it shows that $g_{p,q}$ is a true local minimum, and that allowing the algorithm to iterate will not get the solution out of this minimum. Since a large number of the inpainting literature justifies the use of techniques such as multi-resolution pyramids with the existence of local minima, it is quite significant to be able to rigorously identify some of them, even if it is in a very simplified situation.

Conditions for convergence Now that we have shown that the function $g_{p,q} * \omega$ is stable under the inpainting scheme, we would very much like to know what are the critical conditions for convergence to a function y' which is the global optimum, that is to say equal to 1 everywhere.

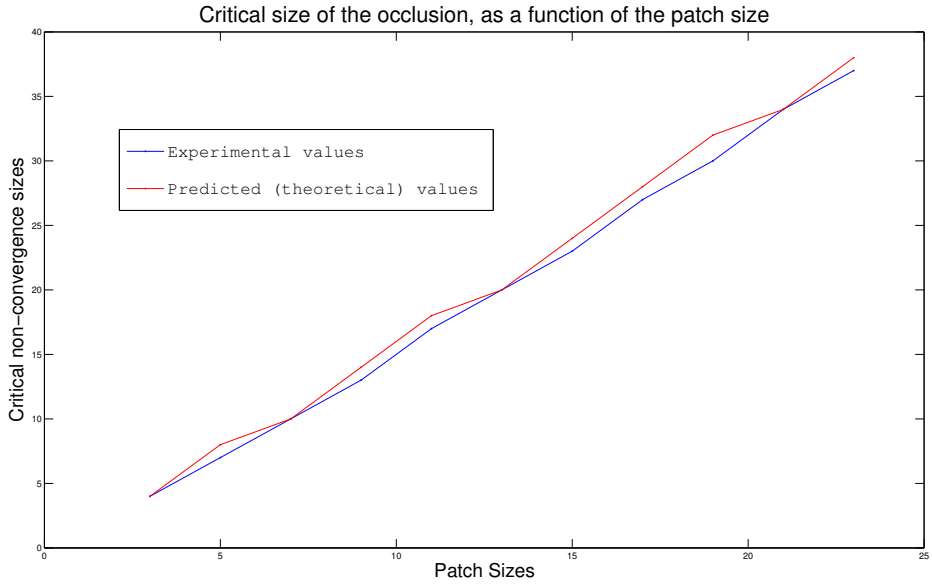


FIGURE 6.7: Critical patch size for inpainting, with a "bad" initial solution. Below this critical size, the solution converges to the "good" solution, above or equal to this size, the inpainting becomes stuck in a local minimum.

Indeed, we can say, trivially, that if $p - q \leq h$, then the algorithm will converge. This is simply because in this situation, all the patches inside the occluded area are matched with Ψ_1, and therefore $y = y'$. Thus, we readily imagine that for a given patch size, there is some sort of critical occlusion size which determines whether the algorithm converges or not. One can determine the condition on the occlusion size, but the calculations are longer than needed for Proposition 1, so they will be shown in Appendix B. We will simply show the final result here. Note that this result takes into account the fact that the algorithm only inpaints positions *inside* the occlusion.

Proposition 2. *Given a patch size* $N = 2h + 1$*, the function* $g_{p,q}$ *will converge if* $p - q < 2N - 2\sqrt{N} + 1$.

Proof : See Appendix B.

To resume, we have shown that in the simple case of 1D binary inpainting, the non-local patch-based variational algorithm shown in Algorithm 3 can get stuck in local minima, and we have calculated the critical occlusion size which determines whether the global optimum is attained or not. While real inpainting situations are certainly much more complicated, we can refer to these results when choosing the number of pyramid levels appropriate for inpainting. As we see in the next paragraph, the theoretical results given here are largely verified in real inpainting situations.

Experimental verification We present some experimental results to see if our analysis is correct. These experiments were carried out in the 2D case with a horizontal structure (see Figure 6.4). The results may be seen in Figure 6.7.

We can see that our analysis has an error of 1 pixel at most. This is due to the fact that we made a hypothesis in the final analysis ; we suppose that s is odd, even though it can be even. However, we have a general idea of how the critical patch size evolves.

This presentation of a special inpainting case is useful for simplified situations, but since we wish to address much more complicated situations, we cannot use this as a general rule as such. In Section 6.5.7, we look at some real examples of video inpainting, and we compare the patch and occlusion sizes at the coarsest resolution level with those predicted in our simple, theoretical inpainting case. We see that the parameters in the real video inpainting case are roughly the same as in the theoretical case. However, we note that this is purely observation and we do not propose any solid, theoretical proposition for such complicated cases as video inpainting with moving objects.

6.5 Proposed video inpainting algorithm

As we have stated, our video inpainting algorithm will be based on a non-local patch-based variational approach. We optimise a global patch-based functional with an iterative algorithm, as was first formulated for video inpainting by Wexler *et al.* [124], and used for image inpainting in [5, 41]. The central machinery of the algorithm is based on the alternation of two core steps : a search for the nearest neighbours of patches which contain occluded pixels, and a reconstruction step based on the aggregation of the information indicated by the nearest neighbours.

This iterative algorithm is embedded in a multi-resolution pyramid scheme, similarly to [47, 124]. The multi-resolution scheme is vital for the correct reconstruction of structures and moving objects in large occlusions.

Various challenges arise naturally within this framework. In this Section we look at these problems and we shall propose solutions. First of all, we must address the problem of finding ANNs of spatio-temporal patches with reasonable execution times. This problem is vital to resolve, otherwise it is practically impossible to carry out any experiments or gain any knowledge about what works and what does not. We propose an extension of the PatchMatch algorithm [11] to the spatio-temporal case to deal with the ANN search. Secondly, the successful inpainting of dynamic textures in videos appears as a particularly important problem. We shall see that this fails in the classic framework of patch-based variational approaches, and we propose a modification of the patch distance to account for textural and high frequency components to counter this. Next, we deal with the case of moving background and cameras by estimating an affine, dominant motion in the video which is used to realign this video. This step is crucial in order to obtain repetitive spatio-temporal patches, even in the case of motion with small amplitude. We also deal with the issue of initialisation, which is also very important and often overlooked in previous work.

All algorithmic choices are made explicit (such as those concerning the use of multi-resolution pyramids), so that the algorithm is readily usable. A summary of the different steps in our algorithm can be Figure 6.8.

Notation Before describing our algorithm, let us first of all recall the notation introduced in Section 6.2. A diagram which summarises this notation can be seen in Figure 6.9. Let $u : \Omega \to \mathbb{R}^3$ represent the colour video content, defined over a spatio-temporal volume Ω. In order to simplify the notation, u will correspond both to the information being reconstructed inside the occlusion, and the unoccluded information which will be used for inpainting. We denote a spatio-temporal position in the video as $p = (x, y, t) \in \Omega$ and by $u(p) \in \mathbb{R}^3$ the vector containing the colour values of the video at this position.

Let $\mathcal{H}$ be the spatio-temporal occlusion (the "hole" to inpaint) and $\mathcal{D}$ the data set (the unoccluded area). Note that $\mathcal{H}$ and $\mathcal{D}$ correspond to spatio-temporal *positions* rather than actual video content and that they form a partition of Ω, that is $\Omega = \mathcal{H} \cup \mathcal{D}$ and $\mathcal{H} \cap \mathcal{D} = \emptyset$.

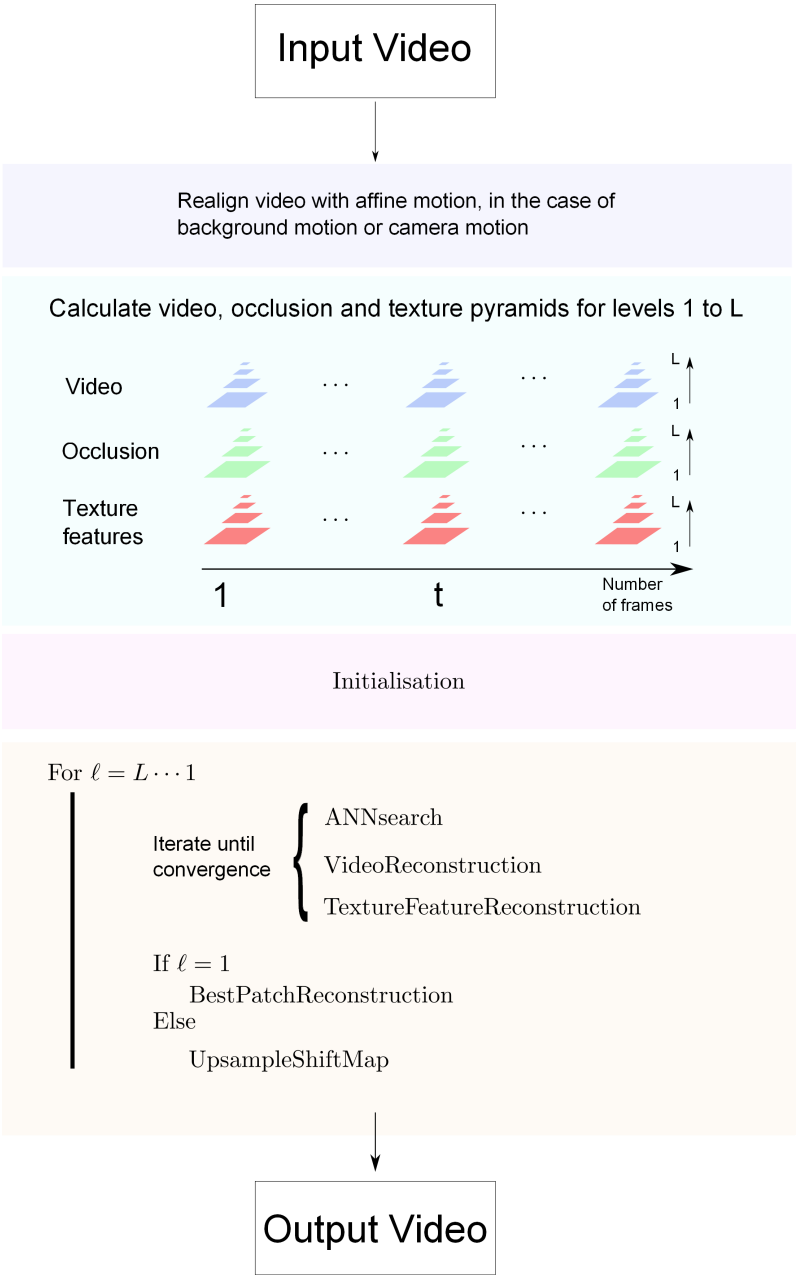


FIGURE 6.8: Diagram of the proposed video inpainting algorithm.

Let $\mathcal{N}_p$ be a spatio-temporal neighbourhood of p. This neighbourhood is defined as a rectangular cuboid centred on p. The video *patch* centered at p is defined as vector $W_p^u = (u(q_1) \cdots u(q_N))$ of size $3 \times N$, where the N pixels in $\mathcal{N}_p$, $q_1 \cdots q_N$, are ordered in a predefined way.

Let us note $\tilde{\mathcal{D}} = \{p \in \mathcal{D} : \mathcal{N}_p \subset \mathcal{D}\}$ the set of unoccluded pixels whose neighborhood is also unoccluded (video patch W_p^u is only composed of known colour values). We shall only use patches stemming from $\tilde{\mathcal{D}}$ to inpaint the occlusion. Also, let $\tilde{\mathcal{H}} = \cap_{p \in \mathcal{H}} \mathcal{N}_p$ represent a dilated version of $\mathcal{H}$.

Given a distance $d(\cdot, \cdot)$ between video patches, a key tool for patch-based inpainting is to define a correspondence map that associates to each pixel $p \in \Omega$ (notably those in occlusion) another position $q \in \tilde{\mathcal{D}}$, such that patches W_p^u and W_q^u are as similar as possible. This can be formalised using *shift map* $\phi : \Omega \to \mathbb{N}^3$ that captures the shift between a position and its correspondent, that is $q = p + \phi(p)$ is the "correspondent" of p. This map must verify that $p + \phi(p) \in \tilde{\mathcal{D}}$, $\forall p$ (see Figure 6.9 for an illustration).

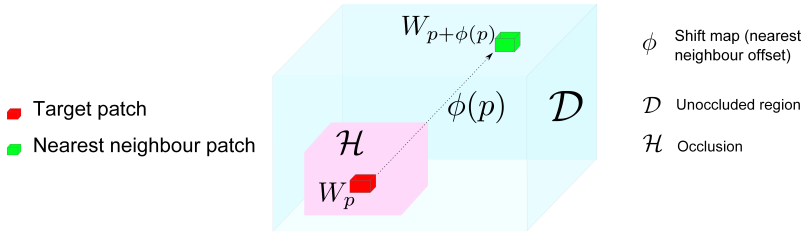


FIGURE 6.9: Illustration of the notation used for our video inpainting algorithm

6.5.1 Mimizing a non-local patch-based functional

The cost function which we use, following the work of Wexler *et al.* [124], has both u and ϕ as arguments :

$$E(u, \phi) = \sum_{p \in \mathcal{H}} d^2(W_p^u, W_{p+\phi(p)}^u), \tag{6.37}$$

with

$$d^2(W_p^u, W_{p+\phi(p)}^u) = \frac{1}{N} \sum_{q \in \mathcal{N}_p} \|u(q) - u(q + \phi(p))\|_2^2. \tag{6.38}$$

We also note that this energy has been used by Kwatra *et al.* [82] for the purposes of texture synthesis.

In all that follows, in order to avoid cumbersome notation we shall drop the u from W_p^u and simply denote patches as W_p.

We have shown in Section 6.4.1, that this functional is in fact a special case of the formulation of Arias *et al.* [5]. As mentioned at the beginning of this Section, this functional is optimised using the following two steps :

– A search for nearest neighbours of all patches contained in the occlusion ;
– A reconstruction step which uses these nearest neighbours to inpaint the video.

These steps are iterated so as to converge to a satisfactory solution. The process may be seen as an alternated minimisation over the shift map and the video content. As in many image processing

and computer vision problems, this approach is implemented in a multi-resolution framework in order to improve results and avoid local minima.

Now that we have given a general outline of our algorithm, we proceed to address some of the key challenges in video inpainting. The first of these concerns the search for the nearest neighbours of patches centred on pixels which need to be inpainted.

6.5.2 Approximate Nearest Neighbour search

When considering the high complexity of the NN search step, it quickly becomes apparent that searching for exact nearest neighbours would take far too long. Therefore, an *approximate nearest neighbour* (ANN) search is carried out. Wexler *et al*. proposed the k-d tree based approach of Arya and Mount [6] for this step, but this approach remains quite slow. For example, one ANN search step takes about an hour for a video containing $120 \times 340 \times 100$ pixels, with about $422,000$ missing pixels, which represents a relatively small occlusion (the equivalent of a 65×65 pixel box in each frame). We shall address this problem here, in particular by using an extension of the PatchMatch algorithm [11] to the spatio-temporal case. We note that the PatchMatch algorithm has also been used in conjunction with a 2D version of Wexler's algorithm for *image* inpainting, in the Content-Aware Fill tool of Photoshop [56], and by Darabi *et al*. [41]. We discuss our decision to use PatchMatch in greater detail in Section 6.6.1, in comparison to other possible algorithms.

Spatio-temporal extension of PatchMatch [11] Barnes *et al*. 's PatchMatch is a conceptually simple algorithm based on the hypothesis that, in the case of image patches, the shift map defined by the spatial offsets between ANNs is piece-wise constant. This is essentially because the image elements which the ANNs connect are often on rigid objects of a certain size. In essence, the algorithm looks randomly for ANNs and tries to "spread" those which are good. We extend this principle to the spatio-temporal setting. Our spatio-temporal extension of the PatchMatch algorithm consists of three steps : (i) initialisation,(ii) propagation and (iii) random search.

- Initialisation
- Propagation
- Random search

Let us recall that $\tilde{\mathcal{H}}$ is a dilated version of $\mathcal{H}$. Initialisation consists of randomly associating an ANN to each patch W_p, $p \in \tilde{\mathcal{H}}$, which gives an initial ANN shift map, ϕ. In fact, apart from the first iteration, we already have a good initialisation : the shift map ϕ from the previous iteration. Therefore, except during the initialisation step (see Section 6.5.6), we use this previous shift map in our algorithm instead of initialising randomly.

The propagation step encourages shifts in ϕ which lead to good ANNs to be spread throughout ϕ. In this step, all positions in the video volume are scanned lexicographically. For a given patch W_p at location $p = (x, y, t)$, the algorithm considers the following three candidates : $W_{p+\phi(x-1,y,t)}$, $W_{p+\phi(x,y-1,t)}$ and $W_{p+\phi(x,y,t-1)}$. If one of these three patches has a smaller patch distance with respect to W_p than $W_{p+\phi(p)}$, then $\phi(p)$ is replaced with the new, better shift. The scanning order is reversed for the next iteration of the propagation, and the algorithm tests $W_{p+\phi(x+1,y,t)}$, $W_{(p+\phi(x,y+1,t)}$ and $W_{p+\phi(x,y,t+1)}$. For a visual illustration of the neighbours used during the propagation step, see Figure 6.10.

In the two different scanning orderings, the important point is obviously to use the patches which have already been processed in the current propagation step. For this step, it is important to know whether the scanning order of the video dimensions has any influence on the quality of the resulting ANNs. In particular, the regularity properties (upon which the PatchMatch algorithm is

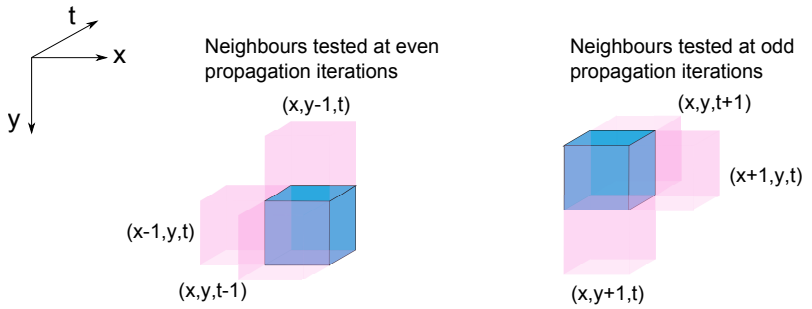


FIGURE 6.10: A visual illustration of the neighbour whose shift maps are tested during the propagation step of the spatio-temporal PatchMatch. PatchMatch changes the neighbours used for testing between odd and even iterations.

based) may be different for different dimensions. For example, if we suppose that there is greater information redundancy along the t axis, then it is preferable to analyse the ANNs along this axis before analysing the others. This sort of consideration was not taken into account in [11], since there is no reason *a priori* to suppose that the x and y axes present different degrees of coherence.

	Mean ANN patch error, per component						
Propagation order	t,y,x	y,t,x	t,x,y	x,t,y	y,x,t	x,y,t	Full search
Beach Umbrella	9.22	9.61	9.49	9.54	9.57	9.11	6.83
Crossing Ladies	7.53	7.44	7.42	7.51	7.50	7.35	6.14
Jumping Girl	6.48	6.52	6.40	6.49	6.50	6.45	4.80

TABLE 6.2: Comparison of propagation scanning ordering, in terms of average component error between a patch and its approximate nearest neighbour. On the last row is the average error of the true nearest neighbour.

Table 6.2 compares the average ANN error (in terms of the sum of square differences) per patch component, for all the scanning orderings (nine in total). The three standard sequences of Wexler *et al.* [124] are analysed in this manner. We observe that the patch error is very stable with respect to the scanning order. Therefore, we have chosen (arbitrarily) to maintain the original scanning order of PatchMatch, and add the t dimension afterwards.

The third PatchMatch step, the random search, consists in looking randomly for better ANNs of each W_p in an increasingly small area around $p+\phi(p)$, starting with a maximum search distance. At iteration k, the random candidates are centred at the following positions :

$$q = p + \phi(p) + \lfloor r_{\max} \rho^k \delta_k \rfloor, \tag{6.39}$$

where $r_{\max}$ is the maximum search radius around $p + \phi(p)$, δ_k is a 3-dimensional vector drawn from the uniform distribution over the unit cube $[-1, 1] \times [-1, 1] \times [-1, 1]$ and $\rho \in (0, 1)$ is the reduction factor of the search window size. In the original PatchMatch, ρ is set to 0.5. This random search avoids the algorithm getting stuck in local minima. The maximum search parameter $r_{\max}$ is set to the maximum dimension of the video, at the current resolution level.

The propagation and random search steps are iterated several times to converge to a good solution. In our work, we set this number of iterations to 10. We also note here that we store the distance $d(W_p, W_q)$ to the current ANN W_q in memory. During any patch comparison, if the distance surpasses this value in memory, then we stop the comparison and indicate that the

new ANN is not better than the old one. This algorithmic detail leads to a speedup of about 3 times. Also, before comparing W_p and W_q, obviously we check whether W_q is already the current ANN. This is particularly important for the propagation step. For further details concerning the PatchMatch algorithm in the 2D case, see [11]. Our spatio-temporal extension is summarized in Algorithm 4.

Data: Current inpainting configuration u, ϕ, $\tilde{\mathcal{H}}$
Result: ANN shift map ϕ

for $k = 1$ **to** 10 **do**
  if k *is even* **then** `/* Propagation on even iteration */`
    for $p = p_1$ **to** $p_{|\tilde{\mathcal{H}}|}$ *(pixels in* $\tilde{\mathcal{H}}$ *lexicographically ordered)* **do**
      $a = p - (1,0,0)$, $b = p - (0,1,0)$, $c = p - (0,0,1)$;
      $q = \arg\min_{r \in \{p,a,b,c\}} d(W_p^u, W_{p+\phi(r)}^u)$;
      if $p + \phi(q) \in \tilde{\mathcal{D}}$ **then** $\phi(p) \leftarrow \phi(q)$
    end
  else `/* Propagation on odd iteration */`
    for $p = p_{|\tilde{\mathcal{H}}|}$ **to** p_1 **do**
      $a = p + (1,0,0)$, $b = p + (0,1,0)$, $c = p + (0,0,1)$;
      $q = \arg\min_{r \in \{p,a,b,c\}} d(W_p^u, W_{p+\phi(r)}^u)$;
      if $p + \phi(q) \in \tilde{\mathcal{D}}$ **then** $\phi(p) \leftarrow \phi(q)$
    end
  end
  for $p = p_1$ **to** $p_{|\tilde{\mathcal{H}}|}$ **do** `/* Random search */`
    $q = p + \phi(p) + \lfloor r_{\max} \rho^k \text{RandUniform}([-1,1]^3) \rfloor$;
    if $d(W_p^u, W_{p+\phi(q)}^u) < d(W_p^u, W_{p+\phi(p)}^u)$ ***and*** $p + \phi(q) \in \tilde{\mathcal{D}}$ **then** $\phi(p) \leftarrow \phi(q)$
  end
end

Algorithm 4: ANN search with 3D PatchMatch

6.5.3 Video reconstruction

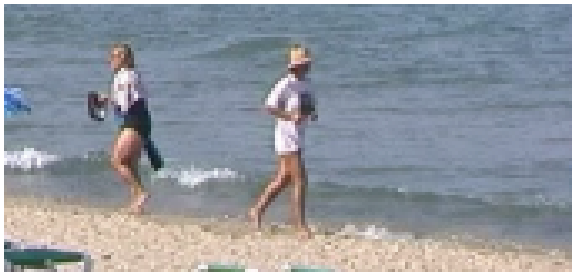
Concerning the reconstruction step, we use a a weighted mean based approach, inspired by the work of Wexler *et al.*, in which each pixel is reconstructed in the following manner :

$$u(p) = \frac{\sum_{q \in \mathcal{N}_p} s_p^q \, u(p + \phi(q))}{\sum_{q \in \mathcal{N}_p} s_p^q}, \quad \forall p \in \mathcal{H}, \tag{6.40}$$

with

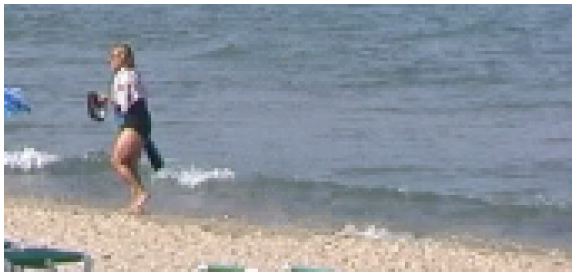
$$s_p^q = \exp\left(-\frac{d^2(W_q, W_{q+\phi(q)})}{2\sigma_p^2} \right). \tag{6.41}$$

Wexler *et al.* proposed the use of an additional weighting term to give more weight to the information near the occlusion border. We dispense with this term, since in our scheme it is somewhat replaced by our method of initialising the solution which will be detailed in Section 6.5.6. The parameter σ_p is defined as the 75th percentile of all distances $\{d(W_q, W_{q+\phi(q)}),\ q \in \mathcal{N}_p\}$ as in [124].

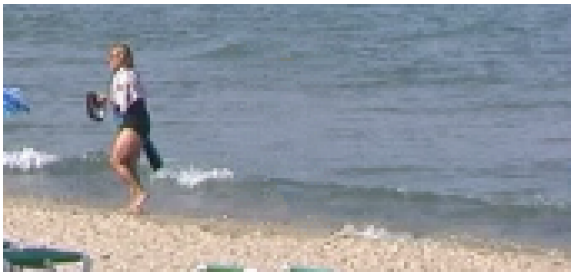


Original frame : “Crossing ladies”

Weighted average



Mean shift



Best patch

FIGURE 6.11: Comparison of different reconstruction methods. We observe that the reconstruction using the best patch at the end of the algorithm produces similar results to the use of the mean shift algorithm. Please note that the blurring effect is best viewed in the pdf version of the manuscript.

We mentioned in Section 6.4.1 that in order to minimise (6.37), the natural approach would be to do the reconstruction with the non-weighted scheme ($s_p^q = 1$ in Equation (6.40)) that stems from $\frac{\partial E}{\partial u(p)} = 0$. However, the weighted scheme above tends to accelerate the convergence of the algorithm, meaning that we produce good results faster. Thus we keep it in our work.

An important observation is that, in the case of regions with high frequency details, the use of this mean reconstruction (weighted or unweighted) often leads to blurry results, even if the correct patches have been identified. This phenomenon was also noted in [124]. Although we shall propose in Section 6.5.4 a method to correctly identify textured patches in the matching steps, this does not deal with the *reconstruction* of the video. We thus need to address this problem, at least in the final stage of the approach : throughout the algorithm, we use the unweighted mean given in Equation (6.40) and, at the end of the algorithm, when the solution has converged at the finest pyramid level, we simply inpaint the occlusion using the best patch among the contributing ones. This corresponds to setting σ_p to 0 in (6.40)-(6.41) or, seen in another light, may be viewed as a very crude annealing procedure. The final reconstruction at position $p \in \mathcal{H}$ reads :

$$u^{(\text{final})}(p) = u(p + \phi^{(\text{final})}(q^*)), \quad \text{with } q^* = \arg\min_{q \in \mathcal{N}_p} d(W_q^{(\text{final}-1)}, W_{q+\phi^{(\text{final})}(q)}). \tag{6.42}$$

Wexler *et al.* proposed another solution to this problem based on the mean shift algorithm. The idea is to find the dominant modes of the colours of the different $u(p+\phi(q)$ which contribute to the reconstruction of the pixel p, and only use those which have been categorised as belonging to the dominant mode. Progressively, throughout the algorithm, the band-width used in the mean shift algorithm is reduced resulting in a very few colours being chosen for reconstruction at the end. This has a similar effect to our approach, and avoids blurring in the output. However, it also

Original image | Image melding [41] | Wexler *et al.* [124] | Proposed approach

FIGURE 6.12: Illustration of the necessity of spatial features for image inpainting. Without the proposed texture features, the correct texture may not be found. The occlusion border is highlighted in green.

clearly increases the complexity and execution time of the algorithm. Figure 6.11 shows that very similar results to those in [124] may be obtained with our much simpler approach.

Arias *et al.* [5] proposed an image inpainting method which gradually reduces the number of patches used for reconstruction. Such a technique could perhaps be used for video inpainting.

6.5.4 Dealing with textures in videos

In order for any patch-based inpainting algorithm to work, it is necessary that the patch distance identify "correct" patches. In the video inpainting framework which we use in our work, this is problematic for several reasons. Firstly, as noticed by Liu and Caselles in [88], the use of multiresolution pyramids can make patch comparisons ambiguous, especially in the case of textures, in images and videos. Secondly, the commonly used ℓ^2 patch distance is ill-adapted to comparing textured patches. Similar ambiguities were also identified by Bugeau *et al.* in [30], but their interpretation of the phenomenon was somewhat different. Thirdly, the PatchMatch algorithm itself can contribute to the identification of incorrect patches. These reasons are explored in much greater detail in Section 6.6.2. However, for the sake of the clarity of the algorithm, we shall directly explain our solution to the problem in this Section.

A good example showing the kind of ambiguities which may arise during the multiresolution inpainting process may be seen in Figure 6.12. In this example, part of the beard is incorrectly inpainted with the texture of the person's hat, since their low-frequency components are similar. The ANNs leading to this incorrect inpainting may be seen in Figure 6.13.

We propose a set of texture features which will allow the inpainting algorithm to correctly match textured patches. Several possible attributes were considered for this purpose :

- An estimation of the local variance of the underlying noise
- *Scattering* operators (Bruna and Mallat [23])
- An average absolute value of the x/y image derivatives

Figure 6.14 shows the different texture attributes which we considered.

In the end, we decided to use a simple, gradient-based texture attribute. This attribute was proposed for image inpainting for Liu and Caselles in [88], and consists of absolute value of the image derivatives, averaged over a certain spatial neighbourhood ν. This choice of attributes was partly motivated by the execution time of the algorithm. In the case of video inpainting, we cannot afford to extend the patch distance excessively by adding many elements, which would have been necessary for the use of the scattering operator, for example.

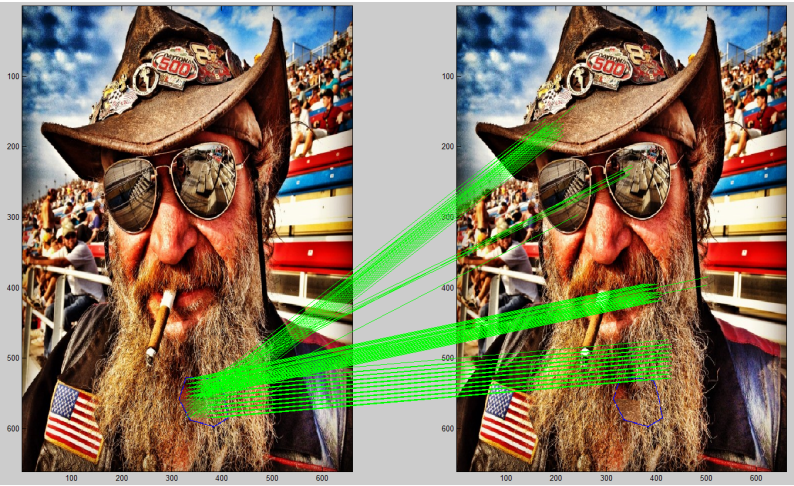


FIGURE 6.13: Result of the PatchMatch ANN search algorithm in a textured zone. It is clear that for inpainting purposes, the ANNs are incorrect.

Local noise variance | The scattering operator [23] | Average absolute value of the x-derivative

FIGURE 6.14: Comparison of several possible texture features. In each case, the highly textured areas, such as the beard, are highlighted.

More formally, we introduce the two-dimensional texture feature $T = (T_x, T_y)$, computed at each pixel $p \in \Omega$:

$$T(p) = \frac{1}{\text{card}(\nu)} \sum_{q \in \nu} (|I_x(q)|, |I_y(q)|), \tag{6.43}$$

where $I_x(q)$ (resp. $I_y(q)$) is the derivative of the image intensity (grey-level) in the x (resp. y) direction at the pixel q. The squared patch distance is now defined as

$$d^2(W_p, W_q) = \frac{1}{N} \sum_{r \in \mathcal{N}_p} \Big(\|u(r) - u(q + r - p))\|_2^2 + \lambda \|T(r) - T(q + r - p)\|_2^2 \Big), \tag{6.44}$$

where λ is a weighting scalar. Figure 6.15 shows the impact of including these texture features in the patch distance. We observe that similar textures are correctly matched, even when using the PatchMatch algorithm.

ANN search with regular patch distance ANN search with modified patch distance

FIGURE 6.15: Comparison of the ANN search with PatchMatch, with the regular patch distance and the patch distance with texture features (see Equation (6.43)).

Now we have chosen which texture attributes to use, we need to decide how to use them. Indeed, it is not sufficient to just insert the attributes into the patch distance and carry on the inpainting as usual. This is simply because the texture information does not exist at coarser levels : most regions seem smooth. To use the features, we use two separate multi-resolution pyramids for the texture features.

The feature pyramid is then set up by subsampling the texture features of the full-resolution input video u. We note here that each level is obtained by subsampling the information contained at the *finest pyramid resolution*, and *not* by calculating T^ℓ based on the subsampled video at the level ℓ :

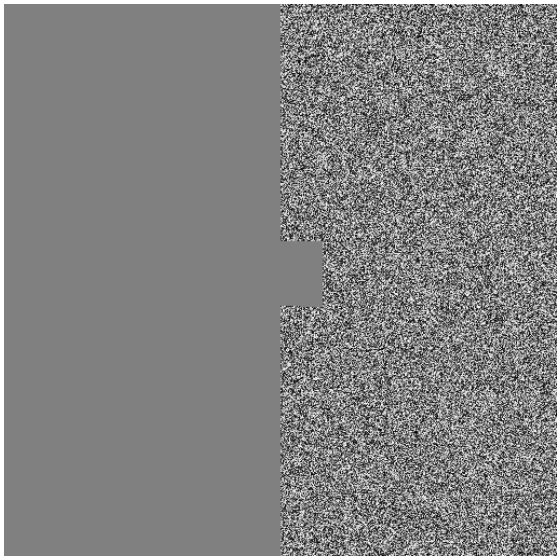
$$\forall (x,y,t) \in \Omega^\ell,\ T^\ell(x,y,t) = T(2^\ell x, 2^\ell y, t), \quad \ell = 1 \cdots L. \tag{6.45}$$

This is an important point, since the required textural information *does not exist* at coarser levels. Features are not filtered before subsampling, since they have already been averaged over the neighbourhood ν. In the experiments done in this paper, this neighbourhood is set, by default, to the area to which a coarsest-level pixel corresponds, which is a square of size 2^{L-1}, as is done in [88]. However, in a more general setting, the size of this area should be independent of the number of levels, so care should be taken in the case where few pyramid levels are used.

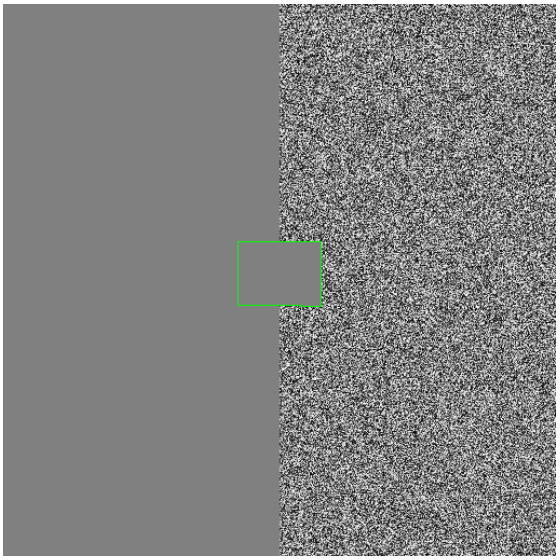
A notable difference with respect to the work of Liu and Caselles [88] is the fact that we use the texture features at *all* pyramid levels. Liu and Caselles do not do this, since they perform graph cut based optimisation at the coarsest level, and at the finer levels only consider small relative shifts with respect to the coarse solution.

To summarise, these features may be seen as simple texture descriptors which help the algorithm avoid making mistakes when choosing the area to use for inpainting, for the various reasons explained above. While these added features may seem like extra work, they have important consequences on the inpainting result, in particular in the case of *video textures*.

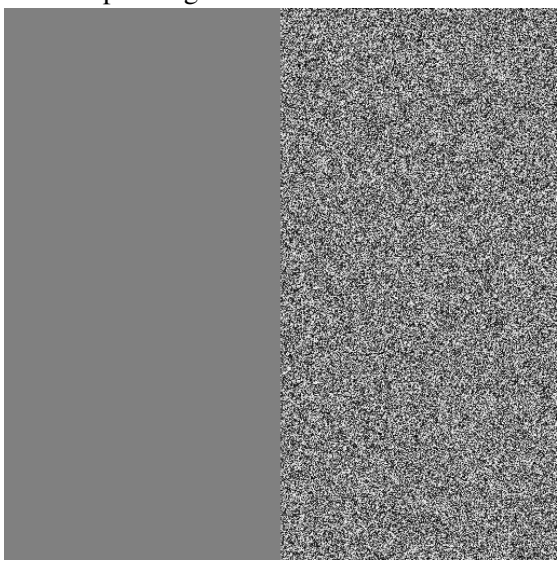
The methodology which we have proposed for dealing with dynamic video textures is important for the following reasons. Firstly, to the best of our knowledge, this is the first inpainting approach which proposes a global optimisation and which can deal correctly with textures in images and videos, without restricting the search space (contrary to [38, 41, 58, 88, 105]). Secondly, while the problem of recreating video textures is a research subject in its own right and algorithms have been developed for their synthesis [83, 109], ours is the first algorithm to achieve this within an inpainting framework. Finally we note that algorithms such as that of Granados *et al.* [57], which are specifically dedicated to background reconstruction, cannot deal with background video tex-



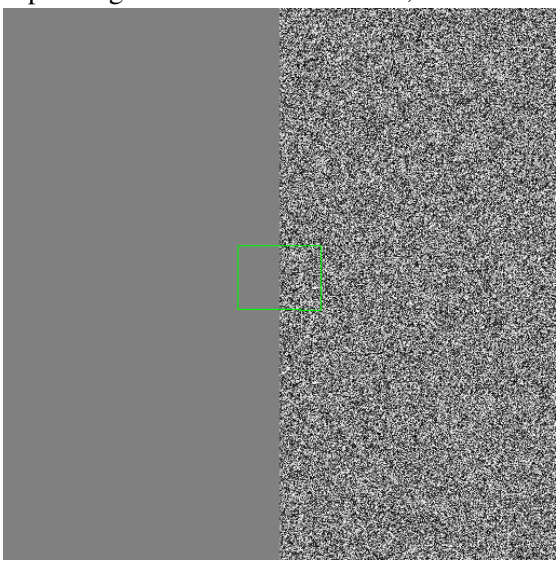
Inpainting without texture features



Inpainting without texture features, with border



Inpainting with texture features



Inpainting with texture features, with border

FIGURE 6.16: An interesting example of the utility of the spatial features. With them, we are able to distinguish between pure noise (right) and the constant area (left), and thus recreate the noise. The grey levels of the noisy part are distributed between 0 and 100, and the constant area's grey level is 50.

tures (such as waves), since they suppose that the background is rigid. This hypothesis is clearly not true for video textures. An example of the impact of the texture features in video inpainting may be seen in Figure 6.19. Another interesting situation is shown in Figure 6.16. Here, we really wish to inpaint an area which contains half noise and half constant grey-level values. This example is a "sanity check" for our algorithm : we are indeed able to "inpaint noise".

Original frame : "Jumping man" [57] | Inpainting without realignement | Inpainting with realignement

FIGURE 6.17: A comparison of inpainting results with and without affine realignment. Notice the incoherent reconstruction on the steps. This is due to random camera motion which makes spatio-temporal patches difficult to compare. These random motions are corrected with affine motion estimation and inpainting is performed in the realigned domain.

6.5.5 Inpainting with mobile background

We now turn to another common case of video inpainting, that of mobile backgrounds. This is the case, for example, when hand-held cameras are used to capture the input video.

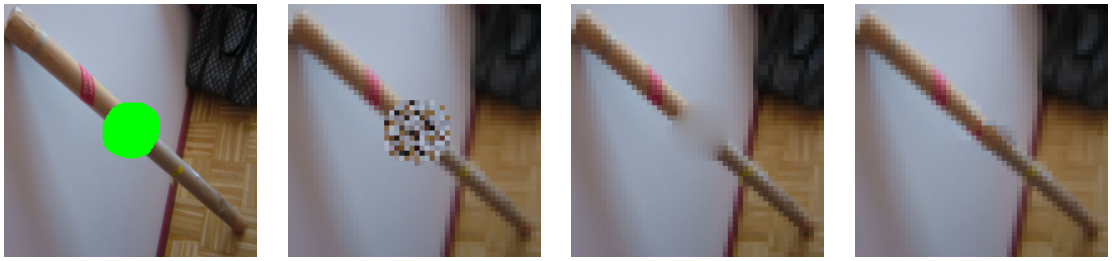
There are several possible solutions to this problem. Patwardhan *et al*. [100] segment the video into moving foreground and background (which may also display motion) using motion estimation with block matching. Once the moving foreground is inpainted, the background is realigned with respect to a motion estimated with block matching, and the background is filled by copying and pasting background pixels.

Granados *et al*. [57] propose a homography-based algorithm for this task. They estimate a set of homographies between each frame and choose which homography should be used for each occluded pixel belonging to the background.

Both of these algorithms require that the background and foreground be segmented, which we wish to avoid here. Furthermore, they have the quite strict restriction that pixels are simply copied from their realigned positions, meaning that the realignment must be extremely accurate. Here, we propose a solution which allows us to use our non-local patch-based variational framework for both tasks (foreground and background inpainting) simultaneously, without any segmentation of the video into foreground and background.

The fundamental hypothesis behind patch-based methods in images or videos is that content is redundant and repetitive. This is easy to see in images, and may appear to be the case in videos. However, the temporal dimension is added in video patches, meaning that a *sequence* of image patches should be repeated throughout the video. This is not the case when a video displays random motion (as with a mobile camera) : even if the required content appears at some point in the sequence, there is no guarantee that the required spatio-temporal patches will repeat themselves with the same motion. Empirically, we have observed that this is a *significant problem even in the case of motions with small amplitude*.

To counter this problem, we estimate a dominant, affine motion between each pair of successive frames, and use this to realign each frame with respect to one reference frame. In our work, we chose the reference frame to be the middle frame of the sequence (this should be adapted for larger sequences). We use the work of Odobez *et al*. [97] to realign the frames. The occlusion $\mathcal{H}$ is



(a) Occluded input image

(b) Random initialisation

(c) Smooth zero-Laplacian interpolation

(d) Onion peel initialisation

(a) Original image

(b) Inpainting after random initialisation

(c) Inpainting after zero Laplacian

(d) Inpainting after onion peel initialisation

FIGURE 6.18: Comparison of initialisation schemes and their consequences on the image inpainting result. Note that the only scheme which successfully reconstructs the tube is the "onion peel" approach.

obviously also realigned. Once the frames are realigned with the reference frame, we inpaint the video as usual. Finally, when inpainting is finished, we perform the inverse affine transformation on the images and paste the solution into the original occluded area. Figure 6.17 compares the results with and without this pre-processing step. Without it, it is not possible to find coherent patches which respect the border conditions.

6.5.6 Initialisation of the solution

The iterative procedure at the heart of our algorithm relies on an initial inpainting solution. The initialisation step is very often left unspecified in work on video inpainting. As we shall see in this Section, it plays a vital role, and we therefore explain our chosen initialisation method in detail.

We inpaint at the coarsest level using an "onion peel" approach, that is to say we inpaint one layer of the occlusion at a time, each layer being one pixel thick.

More formally, let $\mathcal{H}'$ be the current occlusion, and $\partial\mathcal{H}' \subset \mathcal{H}'$ the current layer to inpaint. We define the *unoccluded* neighbourhood $\mathcal{N}'_p$ of a pixel p, with respect to the current occlusion $\mathcal{H}'$.

These are all the pixels in the neighbourhood of p which are not currently occluded with respect to $\mathcal{H}'$:

$$\mathcal{N}'_p = \{q \in \mathcal{N}_p, q \notin \mathcal{H}'\}. \tag{6.46}$$

Some choices are needed to implement this initialisation method. First of all, we only compare the unoccluded pixels during a patch comparison. The distance between two patches W_p and $W_{p+\phi(p)}$ is therefore redefined as :

$$d^2(W_p, W_{p+\phi(p)}) = \frac{\sum_{q \in \mathcal{N}'_p} ||u(q) - u(q+\phi(p))||_2^2 + \lambda ||T(q) - T(q+\phi(p))||_2^2)}{|\mathcal{N}'_p|}. \tag{6.47}$$

We also need to choose which neighbouring patches to use for reconstruction. Some will be quite unreliable, as only a small part of the patches are compared. In our implementation, we only use the ANNs of patches whose centres are located outside the current occlusion layer. Formally, we reconstruct the pixels in the current layer by using the following formula, modified from Equation (6.17) :

$$u_p = \frac{\sum_{q \in \mathcal{N}'_p} s_p^q u(p + \phi(q))}{\sum_{q \in \mathcal{N}'_p} s_p^q}. \tag{6.48}$$

The same reconstruction is applied to the texture features. A pseudo-code for the initialisation procedure may be seen in Algorithm 5.

Data: Coarse level inputs u^L, T^L, $\mathcal{H}^L$
Result: Initialised solution

$B \leftarrow 3 \times 3 \times 3$ structuring element;
$\mathcal{H}' \leftarrow \mathcal{H}^L$;
while $\mathcal{H}' \neq \emptyset$ **do**
  $\partial\mathcal{H}' \leftarrow$ one-pixel border layer of $\mathcal{H}'$;
  $\phi^L \leftarrow$ ANNsearch($u^L, \mathcal{T}^L, \partial\mathcal{H}'$) with partial patch comparison;
  $u^L \leftarrow$ Reconstruction($u^L, \phi^L, \partial\mathcal{H}'$) with Equation 6.48;
  $T^L \leftarrow$ Reconstruction($T^L, \phi^L, \partial\mathcal{H}'$) with Equation 6.48;
  $\mathcal{H}' \leftarrow$ Erosion($\mathcal{H}', B$);
end

Algorithm 5: Inpainting initialisation algorithm

Figure 6.18 shows some evidence to support a careful initialisation scheme. Three different initialisations have been tested : random initialisation, zero-Laplacian interpolation and onion peel. Random initialisation is achieved by initialising the occlusion with pixels chosen randomly from the image. Zero Laplacian interpolation involves solving the following equation : $\Delta u = 0$, with Dirichlet border conditions. It may be seen (Figure 6.18) that the first two initialisations are unable to join the two parts of the cardboard tube together, and that the subsequent iterations do not improve the situation. In contrast, the proposed initialisation produces a satisfactory result.

6.5.7 Other important algorithmic details

We now present some other algorithmic details which are in fact very important for achieving good results. These details are often not discussed in other work, in spite of their importance.

Our first remarks concern the implementation of the multi-resolution pyramids. Wexler *et al*. and Granados *et al*. have both noted that *temporal* subsampling can be detrimental to inpainting results. This is due to the difficulty of representing motion at coarser levels. For this reason we choose not to subsample in the temporal direction, as in [58]. The only case where we need to temporally subsample is when the objects spend a long time behind the occlusion, (this was only done in the "Jumping girl" sequence). This is quite a hard problem to solve since it becomes increasingly difficult to decide what motion an occluded object should have when the occlusion time grows longer, unless there is strictly periodic motion. If we are in the situation where the object spends a long period of time, and the motion of the object is not very finely sampled, then it is extremely difficult, if not impossible, for the algorithm to produce good results. In the end, this question has been left open. Indeed, an algorithm which could deal successfully with such situations would require a more sophisticated implementation of the multi-resolution scheme than is currently used. This could be investigated in further work.

A crucial choice when using multi-resolution schemes is the number of pyramid levels to use. Most methods leave this parameter unspecified, or fix the size of the image/video at the coarsest scale, and determine the resulting number of levels [58, 88, 105]. As we have seen in Subsection6.4.2 in the image case, the number of pyramid levels should be chosen to respect a certain relationship between the occlusion size and the patch size. In our experiments, we follow this general rule of thumb. It should be noted that one may always have to adjust the number of levels. Table 6.3 shows a rough estimation of the occlusion size at the coarsest pyramid level. We observe that the image is subsampled until the occlusion is between one and two times the patch size, apart from the "Jumping Girl" example. This is relatively coherent with the results from Subsection 6.4.2, although the theoretically predicted size in the case of binary inpainting is somewhat smaller (4.5 in the case of a patch size of 5).

Sequence	Beach umbrella	Crossing ladies	Jumping girl	Duo	Museum
Number of levels	3	3	4	5	5
Occlusion size at coarsest level	5.75	6.25	3.12	6.25	9.06

TABLE 6.3: A rough estimation of the sizes of the occlusions at the coarsest pyramid level in each sequence. To be in agreement with the theoretical arguments of Section 6.4.2, the occlusion size should be roughly 5 pixels, since we are using $5 \times 5 \times 5$ patches. Note that these results represent the occlusion sizes in the direction in which the occlusion is thinnest, so the width of the torso of a person instead of his or her height, for example.

Another question which is of interest is how to pass from one pyramid level to another. Wexler *et al*. presented quite an intricate scheme in [124] to do this, whereas Granados *et al*. propose a simple upsampling the shift map. This is conceptually much simpler than the approach of Wexler *et al*. and, after experimentation, we chose this option as well. Therefore, the shift map ϕ is upsampled using nearest neighbours interpolation, and both the higher resolution video and the higher resolution texture features are reconstructed using Equation (6.17). One final note on this point is that we use the upsampled version of ϕ as an initialisation for the PatchMatch algorithm at each level apart from the coarsest.

Original frame : "Waves" Inpainting without features Inpainting with features

FIGURE 6.19: Comparison with and without texture features in the case of video textures. Without the features, the algorithm fails to recreate the waves correctly. Further such examples can be seen at http://www.enst.fr/~gousseau/video_inpainting.

We also require a threshold which will stop the iterations of the ANN search and reconstruction steps. In our work, we use the average colour difference in each channel per pixel between iterations as a stopping criterion. If this falls below a certain threshold, we stop the iteration at the current level. We set this threshold to 0.1. In order to avoid iterating for too long, so we also impose a maximum number of 20 iterations at any given pyramid level.

The patch size parameters were set to $5 \times 5 \times 5$ in all of our experiments. We set the texture feature parameter λ to 50. Concerning the spatio-temporal PatchMatch, we use ten iterations of propagation/random search during the ANN search algorithm and set the window size reduction factor β to 0.5 (as in the paper of Barnes *et al.* [11]).

6.5.8 Experimental results

Our goal is to achieve high quality inpainting results in varied, complex video inpainting situations, with reduced execution time. Therefore, we evaluate our results in terms of visual quality and execution time. We compare our work to that of Wexler *et al.* [124] and to the most recent video inpainting method of Granados *et al.* [58]. All of the videos in this section can be viewed and downloaded at http://www.enst.fr/~gousseau/video_inpainting.

Visual evaluations First of all, we have tested our algorithm on the videos proposed by Wexler *et al.* [124] and Granados *et al* [58]. The visual results of our algorithm may be seen in Figure 6.21 and Figure 6.22. We note that the inpainting results of the previous authors on these examples are visually almost perfect, so very little qualitative improvement can be made. It may be seen that our results are of similarly high quality to the those of the previous algorithms. In particular we are able to deal with situations where several moving objects must be correctly recreated, without requiring manual segmentation as in [58]. We also achieve these results in at least an order of magnitude less time than the previous algorithms. We note that it is not feasible to apply the method of Wexler *et al.* to the examples of [58], of which the resolution is too large (up to $1120 \times 754 \times 200$ pixels).

Next, we provide experimental evidence to show the ability of our algorithm to deal with various situations which appear frequently in real videos, some of which are not dealt with by previous methods. Figure 6.19 shows an example of the utility of using texture features in the inpainting process : without them, the inpainting result is quite clearly unsatisfactory. We have not directly compared these results with previous work. However it is quite clear that the method of [58] cannot deal with such situations. This method supposes that the background is static, and


Original frame : “Girl” [57]


Result from [57]


Our result

FIGURE 6.20: A comparison of our inpainting result with the that of the background inpainting algorithm of Granados *et al.* [57]. In such cases with moving background we are able to achieve high quality results (as do Granados *et al.*), but we do this in one, unified algorithm. This illustrates the capacity of our algorithm to perform well in a wide range of inpainting situations.

in the case of dynamic textures, it is not possible to restrict the search space as proposed in the same method for moving objects. Furthermore, the background inpainting algorithm of Granados *et al.* [57] supposes that moving background undergoes a homographic transformation, which is clearly not the case for video textures. By relying on a plain colour distance between patches, the algorithm of Wexler *et al.* is likely to produce results similar to the one which may be seen in Figure 6.19 (middle image). Finally, to take another algorithm of the literature, the method of Patwardhan *et al.* [100] would encounter the same problems as that of [57], since they copy-and-paste pixels directly after compensating for a locally estimated motion. More examples of videos containing dynamic textures can be seen at : `http://www.enst.fr/~gousseau/video_inpainting`.

Our algorithm’s capacity to deal with moving background is illustrated by Figure 6.20. We do this in the same unified framework used for all the video inpainting examples in our work, whereas a specific algorithm is needed by Granados *et al.* [57] to achieve this. Thus, we see that the same core algorithm (iterative ANN search and reconstruction) can be used in order to deal with a series of inpainting tasks and situations. Furthermore, we note that no foreground/background segmentation was needed for our algorithm to produce satisfactory results. Finally, we note that such situations are not managed using the algorithm of [124]. Again, examples containing moving backgrounds can be viewed at the referenced website.

In Figure 6.23, two further examples show the flexibility of our algorithm. We shot these videos in real situations, with a handheld camera or mobile phone, which resulted in videos with moving background. In both of these examples, the affine realignment of the video proved crucial. Furthermore, we have dynamic textures (the sea, for example) which we must correctly reconstruct. These examples give further evidence that our algorithm responds well in many different situations.

The generic nature of the proposed approach represents a significant advantage over previous methods, and allows us to deal with many different situations without having to resort to manual intervention or the creation of specific algorithms.

Execution times One of the goals of our work was to accelerate the video inpainting task, since this was previously the greatest barrier to development in this area. Therefore, we compare our execution times to those of [124] and [58] in Table 6.4.

Comparisons with Wexler’s algorithm should be obtained carefully, since several crucial pa-

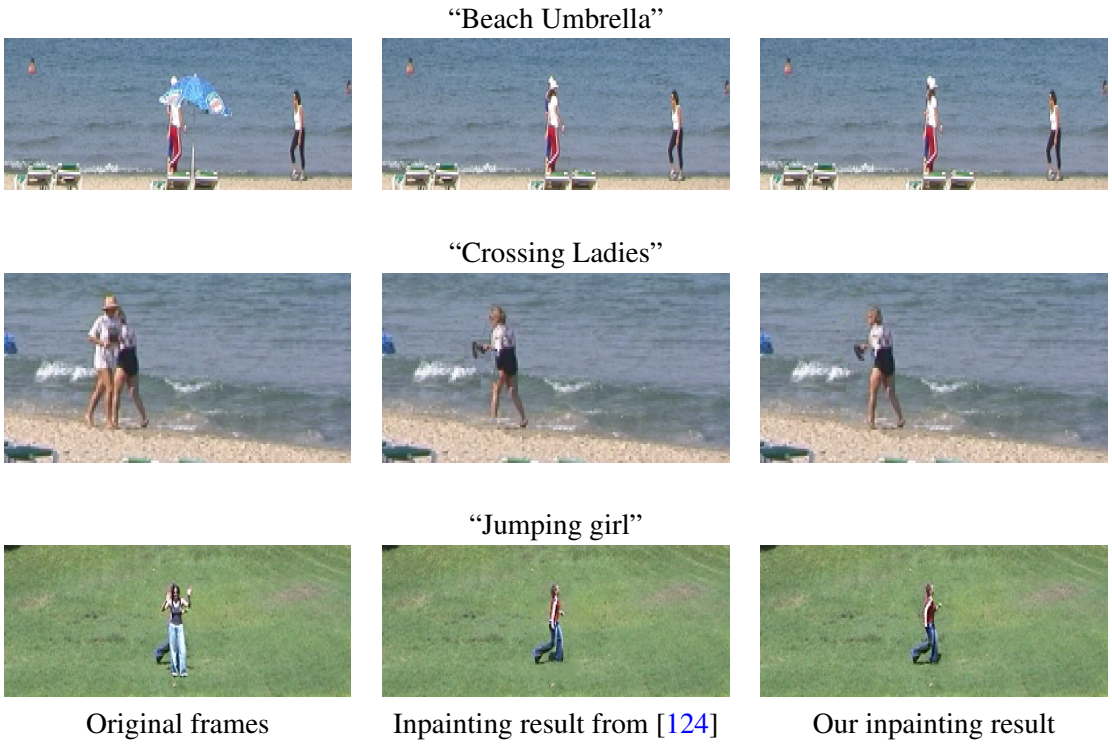


FIGURE 6.21: Comparison of our results with the those of [124]. Visually, the obtained inpainted is very similar, but we are able to reduce the ANN search time by a factor of up to 50 times.

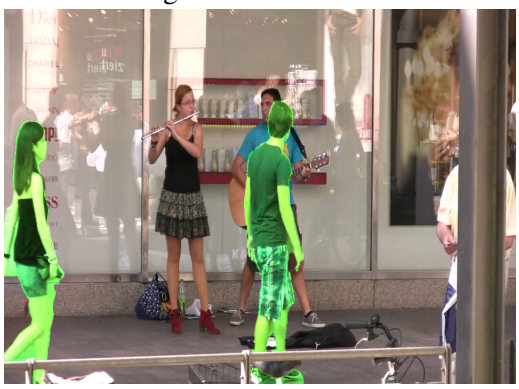
rameters are not specified. In particular, the ANN search scheme used by Wexler *et al*. requires a parameter, ε, which determines the accuracy of the ANNs, and has an extremely important influence on the computational times. For our comparisons, we set this parameter to 10, which produced ANNs with a similar average error per patch component as our spatio-temporal PatchMatch. Another parameter which is left unspecified by Wexler *et al*. is the number of iterations of ANN search/reconstruction steps per pyramid level. This has a very large influence on the total execution time. Therefore, instead of comparing total execution times we simply compare the ANN search times, as this step represents the majority of the computational load. We obtain a speedup of 20-50 times over the method of [6]. We also include our total execution times, to give a general idea of the time taken with respect to the video size. These results show a speedup of around an order of magnitude with respect to the semi-automatic methods of Granados *et al* [58]. In Table 6.4, we have also added our execution times without the use of texture features to illustrate the additional computational load which this adds.

These computation times show that our algorithm is clearly faster than the approaches of [124] and [58]. This advantage is significant because not only is the algorithm more practical to use, but it is also much easier to experiment and therefore make progress in the domain of video inpainting.

6.6 Discussion on video inpainting

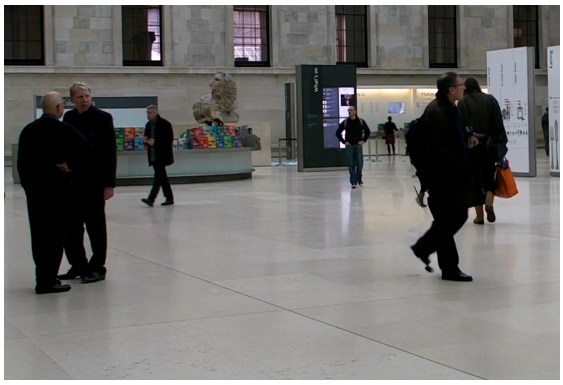
Several points of the video inpainting algorithm presented in Section 6.5 merit further explanations. We did not do this in the previous Section, since we wished to concentrate on algorithmic

Original frame : “Duo”



Original frame : “Museum”

Inpainting result from [58]



Our inpainting result

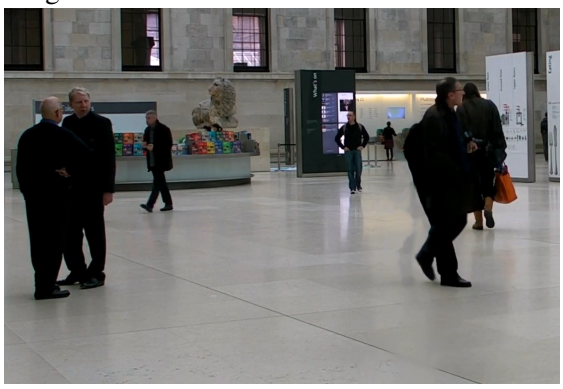


FIGURE 6.22: We achieve similar results to those of [58] in an order of magnitude less time, without user intervention. The occlusion masks are highlighted in green. Result videos are viewable at http://www.enst.fr/~gousseau/video_inpainting.

Algorithm	Approximate nearest neighbour execution times, for all occluded pixels at full resolution.				
	Beach Umbrella 264x68x200	Crossing Ladies 170x80x87	Jumping Girl 300x100x239	Duo 960x704x154	Museum 1120x754x200
Wexler (kdTrees)	985 s	942 s	7877 s	-	-
Ours (3D PatchMatch)	50 s	28 s	155 s	29 min s	44 min

Algorithm	Total execution time				
Granados	11 hours	-	-	-	90 hours
Ours (with texture features)	24 mins	15 mins	40 mins	5.35 hours	6.23 hours
Ours (without texture features)	14 mins	12 mins	35 mins	4.07 hours	4 hours

TABLE 6.4: Partial and total inpainting execution times on different examples. The partial inpainting times represent the time taken for the ANN search for all occluded patches at the full resolution. Note that for the "museum" example, Granados's algorithm is parallelised over the different occluded objects and the background, whereas ours is not.

FIGURE 6.23: Further video inpainting examples. Result videos are viewable at http://www.enst.fr/~gousseau/video_inpainting.

details rather than including too many theoretical discussions. We shall address these points here.

6.6.1 Justification of the use of PatchMatch

In this section, we provide some justification for our choice of ANN search algorithm. We are required to defend this choice because alternative options to PatchMatch such as Coherency Sensitive Hashing [81] and [63, 98] report improved performance with respect to PatchMatch

for ANN searches in images, in terms of time and accuracy. We shall refer to the second two algorithms as Propagation-Assisted KdTrees [63] and TreeCANN [98].

Coherency Sensitive Hashing (CSH) replaces the random search in PatchMatch by a hashing step, which makes the ANNs likely to by better than with random searching. The TreeCANN and the Propagation-Assisted KdTrees both replace this random search by the results of a kdTree search. All of these three algorithms transform the patches onto another set of basis functions. In the case of CSH, this is done instead of projecting patches onto a random line, and in the other two algorithms this is done for dimensionality reduction, to avoid the higher-dimension problems associated with kdTrees.

We tested an adaptation of the algorithm TreeCANN to the spatio-temporal setting. Propagation-Assisted KdTrees has a similar approach to TreeCANN, since it uses kdTrees coupled with dimensionality reduction and propagation, so we use the performance of TreeCANN as an indication of its possible performance.

The authors of TreeCANN report an average of five times speedup with respect to PatchMatch. This is due to fact that PatchMatch needs several iterations of the propagation and random search steps to produce good matches. In order to understand these differences in performance, we need to inspect the different complexities of the algorithm components.

Let n be the number of pixels in the data set, or the reference image/video, and n_o be the number of occluded pixels. Let d be the patch dimension, and d_r be the reduced dimensionality used in TreeCANN.

It is a delicate task to evaluate the different algorithms' complexities accurately. The important complexity dependences for the PatchMatch algorithm are the patch dimension and the number of PatchMatch iterations. The time complexity of the propagation step is $O(6n_o d)$ and that of the random search is $O(n_o d \log(n))$. The time-consuming part of the algorithm is obviously the full patch comparison. However, there are certain practical rules which reduce this complexity in practice. Firstly, during propagation, we test the patches to see if the offset being propagated is already used, in which case the comparison is not made. Secondly, we stop a patch comparison if the SSD exceeds the current ANN of the patch (whose SSD is stored in memory). This has quite an important effect on true complexity. Although this obviously depends on many factors, it is possible to give a general idea of the importance of this technique. After a few iterations, in the propagation step, the comparison stops at about 75 percent of the total patch, and in the random search step it stops at about 20 percent. Early terminations correspond to around 99 percent of the patch comparisons. In practice, this leads to a speedup of about 3 times.

The TreeCANN algorithm requires the creation of the kdTree, but this is only carried out once for each pyramid level. On the other hand, the projection of patches onto the principle components costs $O(n_o d d_r)$. In practice, this takes as much time as the ANN search itself, whose complexity is $O(n_o[(C + d_r)\log(n)])$, where C is a constant (see [7] for details). It is clear that the crucial point in evaluating the gain of TreeCANN with respect to PatchMatch is the factor d_r. This is to be compared with the number of operations done in the patch distance before stopping the patch comparison. A crucial point here concerning the ANN search is that we initialise PatchMatch with the current shift map ϕ. Given that both TreeCANN and Propagation-Assisted KdTrees use propagation preceded by an "improved" initialisation/random search scheme, it is likely that the effort required for this improved step is unnecessary. In our case, the PatchMatch algorithm carries out very few operations after several iterations at the coarsest level. Preliminary results showed only a modest speed-up when a modified TreeCANN was used (about 2 times).

In practice, PatchMatch appears to be a good option because of its simplicity and good performance. Also, the potential complexity gain of methods such as TreeCANN depend on many pa-

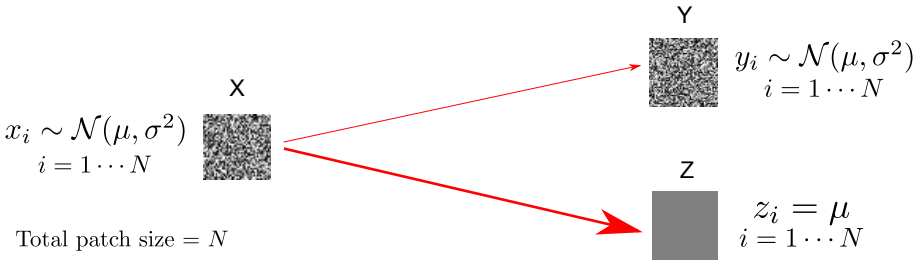


FIGURE 6.24: Illustration of the fact that, on average, the constant patch is preferable to the random patch in terms of the classic ℓ^2 distance.

rameters and specialised "tricks" (such as working on a sparse grid), the impact of which seemed difficult to judge in the case of multi-resolution video inpainting. However, further acceleration is certainly something which could be developed in the future.

6.6.2 Why are textures difficult to handle ?

In this section, we explore certain problems encountered when inpainting with *textures*, in particular in a multi-resolution patch-based framework. In Section 6.5, we stated that the identification and matching of textured areas can be problematic, and briefly listed several reasons for this. We now look at these reasons in greater detail.

Multi-resolution ambiguities One of the reasons for this problem is the multi-resolution approach used in our video inpainting approach. This phenomenon was noted by Liu and Caselles in [88]. The explanation is simply that the texture information does not exist at coarser levels, leading to ambiguities when comparing textured and smooth patches. One could make the argument that this should not change the results, since the whole point of multi-resolution schemes is to first reconstruct low frequency elements and then high frequency ones. However, in the case of both our algorithm and that of Liu and Caselles the initialisation at each pyramid level ℓ (apart from the coarsest one) is done using the *shift map* ϕ resulting from the optimisation at the level $\ell + 1$. This means that the initialised version of u^ℓ will contain information from incorrect patches.

SSD Patch distance The ambiguities introduced by the use of a multi-resolution pyramid are not the only reason for matching incorrect patches. The patch distance itself also contributes to this phenomenon. We observe that when using the classical patch distance (SSD of colour components), textured patches have a tendency to match with smooth patches which have the same low-frequency component. This was also noted by Bugeau *et al*. in [30].

To explain and illustrate this problem, let us consider a white noise patch, W, which is a vector of i.i.d. random variables $W_1 \cdots W_N$, where N is the number of components in the patch (number of pixels for grey-level patches), and the distribution of all W_i's is f_W. Let μ and σ^2 be, respectively, the average and variance of f_W. Let us consider another random patch V following same distribution, and the constant patch Z, composed of $Z_i = \mu$, $i = 1 \cdots N$. To simply the demonstration, we consider only grey-level patches.

In this simple situation, we see that $\mathbb{E}[\|W - V\|_2^2] = 2\mathbb{E}[\|W - Z\|_2^2]$. Therefore, on average, the sum-of-squared-differences (SSD) between two patches of the same distribution is *twice* as great as the SSD between a randomly distributed patch and a constant patch of value μ. Figure 6.24 gives a visual illustration of this phenomenon.

The previous remark is only valid on average between three patches W, V and Z. In reality, we have many random patches V to choose from, and it is sufficient that one of these be better than Z for the patch distance to identify a "textured" patch. Therefore, a more interesting question is the following. Given a white noise patch W, what is the probability that the patch V will be better than the constant patch Z. This is slightly more involved, and we shall limit ourselves to the case where W and V consist of i.i.d. pixels with a normal distribution.

The SSD patch distance between W and V follows a chi-square distribution $\chi^2(0, 2\sigma^2)$, and that between W and Z follows $\chi^2(0, \sigma^2)$. With this, we may numerically compute the probability of a random patch being better than a constant one.

Patch size	3×3	5×5	$3 \times 3 \times 3$	7×7	9×9	11×11	$5 \times 5 \times 5$
Probability	8×10^{-2}	10^{-2}	6×10^{-3}	4.1×10^{-4}	5.5×10^{-6}	3×10^{-7}	2×10^{-7}

TABLE 6.5: **Probability of producing a random 2D or 3D patch that is closer to a random reference patch than to a constant one with same mean value**. Values are obtained through numerical simulations averaged over ten run for each experiment. Components of random patches are i.i.d. according to the centred normal law with a grey level variance of 25.

In Table 6.5, we show the corresponding numerical values for both 2D (image) and 3D (video) patches. It may be seen that for a patch of size 9×9, there is very little chance of finding a better patch than the constant patch. In the video case, we see that in the case of $5 \times 5 \times 5$ patches, there is a 2×10^{-7} probability of creating a better patch randomly. This corresponds to needing an area of $170 \times 170 \times 170$ pixels in a video in order to produce on average one better random patch. While this is possible, especially in higher-definition videos, it remains unlikely for many situations.

PatchMatch : ill-adapted to matching textures We have shown that the l^2 patch distance may be problematic for inpainting in the presence of textures. However, we also identify another problem : PatchMatch itself. PatchMatch relies on the piecewise constancy of ϕ. This is less and less the case as textures become more and more stochastic and their structural nature lessens. Because of this lessening of pieceweise constancy, PatchMatch obviously has difficulty propagating good offsets. Furthermore, as we have indicated, on average, the patch distance much prefers matching textured areas with smooth areas containing the same low-frequency component, rather than textured areas with other similarly textured areas. Offsets leading to textured areas are therefore a *good compromise* and much easier to propagate for PatchMatch. This has been previously remarked upon by the authors of the CSH ANN search algorithm [81].

Naturally, it is of vital importance to know whether PatchMatch is the main culprit in our problems with textures, in which case it would be preferable to simply change the search method rather than introducing extra features into the patch distance. To answer this question, we inpainted textured areas using an exhaustive search (on images only) and found that the same problem occurred, which ruled out PatchMatch as the sole culprit. The reason is that, in practice, the patch averaging of the inpainting method induces a smoothing of the texture during the first few iterations and once the texture is lost, the ANNs are always found in smooth areas. Therefore, instead of changing the ANN search algorithm, we add textural features to the patch distance (Section 6.5.3).

Restricting the ANN search space in inpainting We have shown that inpainting textures correctly can be problematic in the case of patch-based algorithms. An important question that arises is, "why has no one else reported such problems in other work patch-based inpainting ?". It is

Restricted search space

Extended search space

Criminisi *et al*. [38] | Liu and Caselles [88] | Wexler [124] using PatchMatch

FIGURE 6.25: Comparison of several well-known inpainting algorithms with and without a restriction on the search space allowed when looking for nearest neighbour patches. The difference in results is quite remarkable. This means that many inpainting algorithms may find incorrect patches, due to the patch distance used. The occlusion border is outlined in green. The result of our algorithm in this example may be seen in Figure 6.12.

clearly a great obstacle, and yet almost no previous methods mention it. The answer to this question is very straightforward : in most other patch-based inpainting methods, the search space used when looking for patches is either manually or automatically restricted. For instance, in the work of Criminisi *et al*. [38], Granados *et al*. [58] and Liu and Caselles [88] this is the case. In the cases of images, this makes a lot of sense since the most useful information is situated around the occlusion. Therefore, the algorithms *cannot* choose incorrect patches. While this is ostensibly done to accelerate the algorithms, it is acknowledged in [88] that this restriction also improves the quality of the inpainting results. However, this means that very few people consider whether the patch distance is adequate for video inpainting purposes.

An example of what happens in a few previous approaches if the restriction is not used may be seen in Figure 6.25. It is quite clear that the problem of incorrectly matching patches can be observed with various algorithms (which is rather reassuring for our reasoning !).

Unfortunately, this restriction principle does not hold in video inpainting, since the information

can be found anywhere in the video. For example, we might have to search quite far for the correct period of a motion to be visible. One could say that this is only the case for objects moving in the foreground, and that the principle applies to the background, but to apply the principle in this manner one would need to segment the video into foreground and background, which we do not wish to do in our work.

Justification of the choice of texture features In order to justify our choice of texture features (see Equation (6.43)), let us consider a textured image which follows a stochastic model, as we done throughout this section, that is to say $u(p) \sim \mathcal{N}(\mu, \sigma^2)$. Let us consider $T_x(p)$ in this situation. We know that $|u_x(q)|$ follows a *half-normal* distribution. Therefore :

$$\lim_{card(\nu)\to\infty} \frac{1}{card(\nu)} \sum_{q\in\nu_p} |u_\bullet(q)| = \frac{2|\nu|}{\sqrt{\pi}}\sigma.$$

In particular, this means that, if we take our previous notation, the SSD of the texture features between the two random patches W and V will tend to zero as as the size of ν increases, and the SSD of the texture features between W and Z (the constant patch) will tend towards $\frac{2|\nu|}{\sqrt{\pi}}\sigma$ and therefore achieves our goal. We note that this is purely theoretical, since it supposes that W, V and Z are situated in infinitely large regions of coherent texture/constancy. However, we find that these features fulfil our immediate purpose.

6.6.3 The non-local patch-based variational approach *versus* shift maps

In our work, we have seen that patch based approaches are very powerful in the case of video inpainting. This potential was first illustrated by the approach of Wexler *et al*. [124], but was not visible since computational cost of the ANN search step was so great. This was the first problem we addressed for video inpainting. This is quite a significant difficulty, since it renders experimentation extremely tedious, thereby making good decisions on what research directions to take extremely difficult. In the case of this work, we decided to develop a non-local patch-based variational approach, but nothing could guarantee that this was a good idea, since the experimental evidence was so sparse.

It turns out that the non-local patch-based variational formulation is flexible and powerful. However the use of shift-maps remains very popular in image inpainting [44, 64, 105], and should remain an option for video inpainting. There are various advantages and disadvantages of the two approaches, which we may recap with Table 6.6.

At the heart of this discussion is whether it is better to inpaint by optimising over the shift map, or over the colour values of the inpainting result (as in [124]). This is an important question, and should be considered in the future. Generally speaking, it would seem that the shift maps approach is the most popular one, especially in the case of images. This is largely due to the capacity of this approach to avoid blurred inpainting results. One fact does appear from various sources : that it is preferable to copy and paste as large amounts of coherent image/video content as possible.

Therefore an important question is whether it is possible to have an inpainting method which copies large regions (as the shift maps approach tends to do), but can use arbitrary patch sizes and performs a more rigorous optimisation than is currently proposed in the Graph Cuts formulation of the problem. Aujol *et al*. dealt with a very similar question in [8], and while they proved the existence of optima for their functional, they did not provide an algorithm to reach these optima.

This question remains open.

TABLE 6.6: A summary of the advantages and disadvantages of the non-local patch based inpainting method versus the shift map based algorithm.

Method	Advantages	Disadvantages
Non-local patch based variational	- Capable of using arbitrary patch sizes - Possible to *synthesise* new pixels with linear combinations of known pixels - Presumably good for synthesising missing parts of structures	- Subject to blurring on the borders of coherent inpainted regions - Do not explicitly take complete blocks of image/video information
Shift map based	- Tends to copy large blocks of image/video information - Directly copies pixels : reduced blurring - Good for maintaining textures	- Combinatorial labelling optimisation (shift map *etc.*) computationally expensive - Cannot combine patches : presumably a problem for interpolating structures

6.7 Conclusion and further work

In this Chapter, we have shown that video inpainting in highly complex and varied scenes is possible using a non-local patch-based variational approach. At the centre of this approach is a patch-based functional (6.37) which is optimised using an iterative multi-resolution scheme. While this framework is very powerful and flexible, several problems appear for video inpainting. Firstly, the obstacle of the ANN search for video patches is so great that the approach is, for all practical purposes, unusable without addressing this problem. We propose an extension of the PatchMatch algorithm [11] to do this. We also observed that the correct inpainting of dynamic textures in videos poses a significant problem due to the incorrect identification of source patches. To address this, we construct a gradient-based texture feature pyramid which is incorporated into the patch comparison distance. To deal with mobile background and camera, we have proposed a pre-processing step consisting of an affine motion estimation and realignment. This is integrated into the general inpainting framework, rather than having to create a separate algorithm for this case, as in the work of [57]. Also we do not require a moving foreground mask. We have shown that the initialisation of the solution has an important influence on the result, and therefore care must be taken during this initialisation.

The result is an algorithm which is automatic and generic. It can deal with a much wider range of video inpainting situations than other approaches. In particular no other method, to the best

of our knowledge, has explicitly considered the problem of video inpainting in the presence of dynamic textures. Furthermore, we obtain our results with no segmentation of the video, manual or otherwise, which is present in almost every other approach.

During the course of this Chapter on video inpainting, we have also made some remarks on the links between two common formulations of the inpainting problem, and shown in particular that the energy of one formulation may be seen as a special case of the other (see Section 6.4.1). Finally, we arrived at some theoretical results concerning the convergence of a non-local patch-based variational approach in very simple inpainting situations.

6.7.1 Further work

There are several points which could be improved upon in the present work. Firstly, the case where a moving object is occluded for long periods remains very difficult to handle and is not dealt with in a unified manner here. The one solution to this problem (temporal subsampling) does not perform well when complex motion is present. Given the success of the texture attributes in the form of separate multi-resolution pyramids, this problem could perhaps be addressed by including further, temporal, attributes into the patch distance. Another possibility which we considered was to use patches which are spread over a greater temporal span, without increasing the patch size. This could be done by redefining the patch neighbourhood $\mathcal{N}_p$. In this way, a patch could contain information from frames which are quite far apart, allowing it to "join up" the motion of objects which spend a long time in the occlusion. However, this would likely require strictly repetitive motion.

Finally, it is acknowledged that videos of high resolutions still take quite a long time to process (up to several hours). Further acceleration could be achieved by dimensionality-reducing transformations of the patch space. As mentioned in Section 6.6.1, we have tried to do this using Principle Component Analysis, as in the TreeCANN algorithm [98], but this yielded a relatively modest speed-up.

Chapitre 7

Restoring line scratches

7.1 Introduction

The restoration of line scratches has been studied in previous work. Generally speaking, there are two types of approaches : those which suppose that the content in the occlusion is completely unknown, and those which consider that this content may be used. The latter approaches usually include some sort of semi-transparency hypothesis.

Following the development of a video inpainting method in Chapter 6, it is natural to see if it can be applied successfully to the problem of line scratch restoration. Therefore, in this Chapter we compare several relatively simple approaches to scratch restoration in order to determine whether such sophisticated methods as video inpainting are necessary. We also briefly explore a new approach which is based on an optical flow estimation. The advantages and limits of the different approaches are discussed, and in particular we consider which algorithms are appropriate for which situations.

7.2 Prior work

The first well-known work on restoring line scratches is that of Kokaram [76]. This work uses an autoregressive image model in the vicinity of the scratches, and estimates the image parameters (values) supposing a normally distributed noise model. The variance of the noise model is estimated in a Bayesian manner during the detection stage. The interpolated image values are sampled and used to restore the image. This approach is continued and described in greater detail in [78].

Morris describes in his thesis [94] an approach based on Markov Random Fields which uses a spatio-temporal neighbourhood to restore scratched image pixels. However, the temporal neighbourhood was restricted to the previous frame and following frame of a pixel to restore, which is obviously insufficient in the case of scratches, as they are temporally persistent and therefore remain in roughly the same position over a few frames. However, it should be noted that this was the first approach to propose the spatio-temporal neighbourhood. Haindl *et al*. used an autoregressive model in [60] for restoring scratch pixels situated in a local spatio-temporal neighbourhood. This method is perhaps better adapted to tasks such as blotch and dirt detection.

Joyeux *et al*. approached the problem by interpolating first the low-frequency components of the image, and then the high-frequency ones. In [72], the low-frequency components were reconstructed using polynomial interpolation of the unscratched pixels, and the high-frequency ones were processed in the Fourier domain. Joyeux *et al*. continued this sort of approach (separate low-frequency and high-frequency reconstruction), this time using the Maximum A Posteriori (MAP)

technique [43]. The authors describe the use (as in [94]) of a spatio-temporal neighbourhood, however important precisions are lacking in the description. Also, this algorithm differs from the previous work of Joyeux *et al*. in that it takes the degraded pixels into account during the MAP algorithm ; they are considered to be the noisy observations of the underlying process to be estimated (in this case the unscratched grey-levels of the pixels in the scratch area).

Another set of methods [21, 27] perform restoration in the wavelet domain. Intuitively, this approach is attractive if we suppose that scratches are predominant in certain sub-bands of the Wavelet decomposition : if this is the case, then the modification of these sub-bands will have a reduced influence on the rest of the image content. This is particularly important for keeping high-frequency components of the image, which are often removed by more conventional interpolation. The first method to use Wavelets was that of Bretschneider *et al*. [21]. This approach transformed the image using Haar Wavelets and restored the approximation and vertical detail sub-bands. Three interpolation methods were proposed : a median filter (over the coefficients untouched by the scratch), a cubic spline interpolation and third-degree polynomials. The median filter is indicated as the best choice for the reconstruction of the vertical details. Bruni *et al*. continued and developed this sort of approach in [27]. In this paper, Bruni *et al*. use a more sophisticated scheme to remove the influence of scratches. Rather than interpolating the "known" (unscratched) values, they suggest a dampening scheme which should reduce the presence of the scratch until it drops below a visual perception threshold.

Güllü *et al*. use a spatio-temporal search window to look for pixels to use for restoration [74]. The search for a "best" pixel to use is carried out using the mean squared error (MSE) in a spatio-temporal window around the pixel to be restored. In this algorithm, the "best" pixel directly replaces the scratched pixel, instead of using statistical estimation techniques as in [76, 94]. The method may be seen as a relatively direct extension of the patch-based inpainting method of Criminisi *et al*. [38]. The major difference here is that a *spatio-temporal* search window is used (purely spatial patches are used). One clear disadvantage of this method is that the search for the best pixel is influenced by the presence of the scratch itself. Also, the authors specify neither the spatial neighbourhood used, nor the spatio-temporal search window. One way of countering this could possibly be to mask scratched pixels in the MSE calculation.

Grossauer describes in [59] a method of inpainting movies using optical flow. This work takes into account blotches as well as scratches. The blotches are detected using the optical flow itself, by comparing forward and backward vectors. They hypothesise that in the presence of blotches, the forward and backward vectors will be clearly different. The scratches are detected using a simple summation of grey-level values over the columns to produce a 1D signal in which the scratches are presumably easy to detect (see Section 3.2). The method then copies and pastes uncorrupted pixels from the directly preceding and following frames (at times $t-1$ and $t+1$), with a preference for the pixels from frame $t-1$. If no uncorrupted pixels are available, an image inpainting method is used for restoration (the exact method is not specified). Furthermore, the optical flow is used with no modification, although it is clearly influenced by the presence of the film defects, indeed this is the basis of their blotch detection. Finally, as in the case of the work of Güllü *et al*., only the directly adjacent frames are used, which as already mentioned is not well adapted to persistent defects such as scratches.

Elgharib *et al*. [53] recently proposed a blotch and scratch removal algorithm based on a Bayesian framework which also takes into account temporal coherence. For scratches, the algorithm considers that at some point in the image sequence, the required information will be available. The model also considers that the corrupted information is semi-transparent, and establish a degradation model accordingly. Block matching based motion compensation is used to look for areas

in which uncorrupted pixels are available for restoration. Again, as in the case of Grossauer, the motion estimation does not take the presence of the scratch into account, although Elgharib *et al.* use a large enough box size (containing at least 100 uncorrupted pixels) to attenuate the problem.

7.3 Conclusions on prior work

Having reviewed the scratch detection literature, we may make some preliminary remarks. Firstly, a great deal of the work done has targeted the reconstruction of high frequencies in scratched images (noise, film grain, *etc.*). The reason for this is that, while interpolation of image values in a horizontal neighbourhood is quite efficient for continuing structures, it deals quite poorly with textures and noise. When these elements are ignored, the restoration result can be as visibly marked as the initial degraded video. This task is relatively well carried out by methods such as restoration in the Wavelet domain.

Secondly, the use of a spatio-temporal neighbourhood is used by several authors [53, 59, 74, 70, 94] for the purpose of restoration. However, several points are not addressed in the literature :

- Only the directly previous and following frames are taken into account, even though scratches are temporally persistent defects ;
- Motion estimation (block matching, optical flow) is carried out without taking into account the presence of line scratches.

7.4 Optical Flow based restoration : a simple approach

In the video inpainting and restoration literatures, a common approach is to establish a spatio-temporal zone or trajectory which can be used for completing the video. Therefore, in this Section, we briefly explore a simple optical-flow based line scratch restoration algorithm for purposes of comparison with our video inpainting algorithm. In the case of restoring line scratches, we can imagine that then the scratches will be thin enough to establish a reliable motion estimation, after which restoration may be carried out by directly copying and pasting pixels into the scratched area. However, as mentioned above, all of the previous approaches estimate the motion of the video directly in the occlusion, where the content is supposed to be degraded.

One possibility for obtaining a reliable optical flow inside scratched areas is to perform a post-processing step on the optical flow to correct it. This could be done by using a smooth interpolation (see Equation (6.3)) in the scratched area. Another, more sophisticated approach would be to reconstruct the optical flow in an inpainting-like manner, as done in [12, 79] to reconstruct damaged optical flow fields. However, we make the hypothesis that better results can be obtained by taking the presence of the scratches into account when calculating the optical flow.

We briefly review the basics of motion estimation in Appendix A. Let us recall that an optical flow is a dense correspondence between two images, which supposedly respects the intensity constancy assumption. Most often, a smoothness prior is used to alleviate the ill-posed nature of the problem. Let (u, v) represent the optical flow vector, with u corresponding to the horizontal component and v corresponding to the vertical component. Optical flow is often formulated using the following generic minimisation problem.

$$(u, v) = \min_{u,v} \int_{\Omega} E_{data}(I, u, v) + E_{smooth}(u, v), \tag{7.1}$$

where E_{data} corresponds to the intensity consistency assumption, and E_{smooth} to the smoothness prior.


Two consecutive frames from the “Afgrunden 2” sequences

(a) TV-L1 optical flow [127]

(b) TV-L1 optical flow [127], using the energy in Equation 7.4

FIGURE 7.1: An example of the effect of line scratches on optical flow. The optical flow on the left is clearly influenced by the scratch in the upper middle part of the image. The modified optical flow allows the smoothness term to avoid the discontinuities induced by scratches.

In our case, it is necessary to overcome the problem of the presence of the line scratches in order to obtain a reliable optical flow. An example of this problem may be seen in Figure 7.1. One quite natural way of filling in the optical flow is to only use the energy E_{smooth} in the scratched areas, and set E_{data} to 0. Therefore, we have the following energy :

$$E(I,u,v) = \int_{\Omega} \begin{cases} E_{smooth}(u,v) & \text{if } (x,y) \text{ occluded,} \\ E_{data}(I,u,v) + E_{smooth}(u,v) & \text{otherwise.} \end{cases}$$

For the choice of the optical flow scheme, we used an implementation of a TV-L1 optical flow [127] by Pérez *et al.* [101]. This implementation provides a high quality optical flow, and also has the practical advantage of a mask which disables the data term at certain points. Originally, this mask was used to avoid trying to determine the data term when the optical flow led to positions outside the image domain, but it happens that this is very well adapted for our needs as well.

Figure 7.1 shows the effect of line scratches on optical flow. We observe that in the case of the scratch in the top middle side of the image, the optical flow is considerably affected by the presence of the scratch. The modified version on the right side of the figure shows that the smoothness term regularises the optical flow.

Once we have determined an optical flow, we may use it to restore the image. As in the case of Grossauer’s algorithm, this is done by creating a path for each pixel in a certain temporal window. The path is created by concatenating the optical flows from frame to frame. If the pixel is not covered by a scratch at some point in its path, then it is possible to use its colour value at that

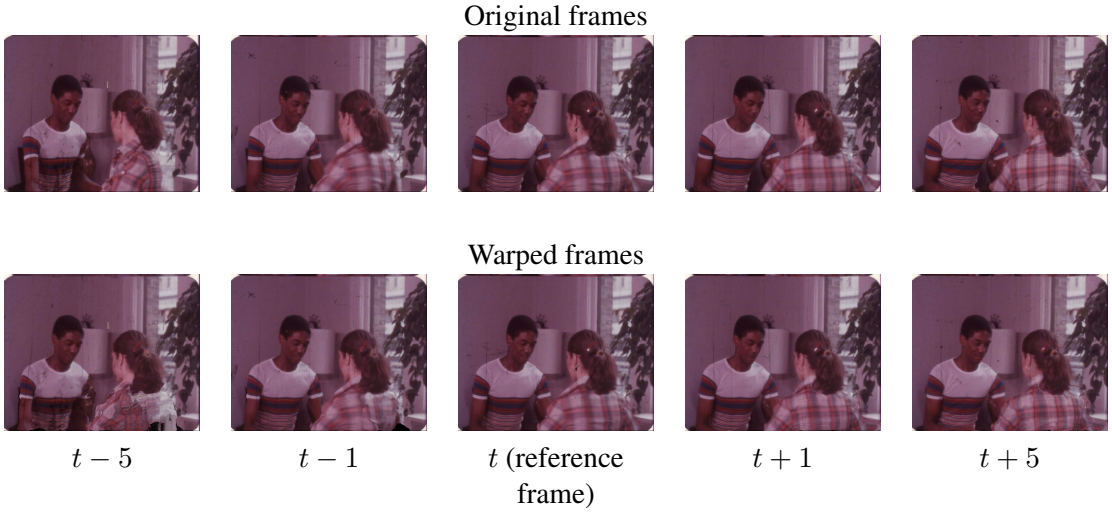


FIGURE 7.2: Image warping over several frames using concatenated optical flow vectors. Each pixel is warped to its corresponding position in frame t. It can be seen that in the case of reasonably fast motion, the warping breaks down after a few frames (for an example, see the image on the bottom left).

spatio-temporal position for restoration. The concatenation of the optical flow vectors is done with bilinear interpolation.

We retain the same notation as found throughout the document. Let p be a spatio-temporal position, let $\mathcal{P}_p$ be the set of pixels belonging to the path of p, and let $\mathcal{H}$ be the occluded (scratched) region. Let $\mathcal{N}_p$ be a patch neighbourhood around p. Note that we shall use a *spatial* patch neighbourhood, and not a spatio-temporal one, since the temporal dimension should be redundant if the optical flow estimation was carried out correctly. Given the success of the reconstruction scheme presented in Section 6.5.3, we take a similar approach here, and replace $u(p)$ with the following weighted average of the unoccluded pixels in $\mathcal{P}_p$:

$$u(p) = \sum_{q \in \mathcal{P}_p \setminus \mathcal{H}} \frac{1}{Z} u(q) \exp\left(-\frac{d^2(\mathcal{N}_p, \mathcal{N}_q)}{2\sigma^2}\right), \tag{7.2}$$

where σ is a parameter, Z is a normalising factor and $d^2(W_p, W_q)$ is the l^2 patch distance between W_p and W_q. While in Section 6.5.4 we needed to modify the patch distance, we do not do this here as we consider that a pixel's path is sufficiently constrained so as not to introduce any ambiguous patch decisions.

For the pixels which have no unoccluded pixels available, we use a horizontal linear interpolation between the nearest restored pixels on either side.

The major problem of this approach is the reliability of the optical flow. For the method to succeed, the optical flow needs to be very precise. Additionally, the greater the number of optical flow vectors which are concatenated, the more errors are accumulated, which leads to very poor quality restoration results. This problem is illustrated in Figure 7.2.

7.5 Visual comparison of restoration methods

We now show some visual comparisons of the restoration results produced by several different algorithms (including our video inpainting algorithm). We chose the algorithms of Bruni *et al.* [27], Güllü *et al.* [74], the optical flow-based algorithm which we explored in Section 7.4 and the video inpainting algorithm developed in Section 6.5. The algorithms of Bruni *et al.* and Güllü *et al.* are our implementations of their work.

The examples shown in Figure 7.3, Figure 7.3 and Figure 7.4 represent various situations which arise in the restoration of scratches. We have used four sequences to compare the restoration algorithms. The first two, "Sitdown" and "Star" (Figure 7.3), are quite marked by noise and film grain. The third sequence ("Dance", Figure 7.3) is interesting in that the back of the girl is moving very quickly, meaning that the patch-based and optical flow-based approaches cannot succeed, because neither the patches nor the optical flow are coherent. In such a case, even a simple linear interpolation is a better choice. The fourth sequence ("Face") contains information which is completely irretrievable (the eye of the person in the frame). Finally, the last example shows synthetic scratches on a high definition video (1280×720). The positions of the scratches were extracted from a real scratched film, and the spatial scratch model used to simulate the scratches is that of Kokaram [76]. The scratch has a width of 15 pixels. This width is justified by the fact that the resolution is roughly twice that of the other sequences, whose scratch width is 7 pixels.

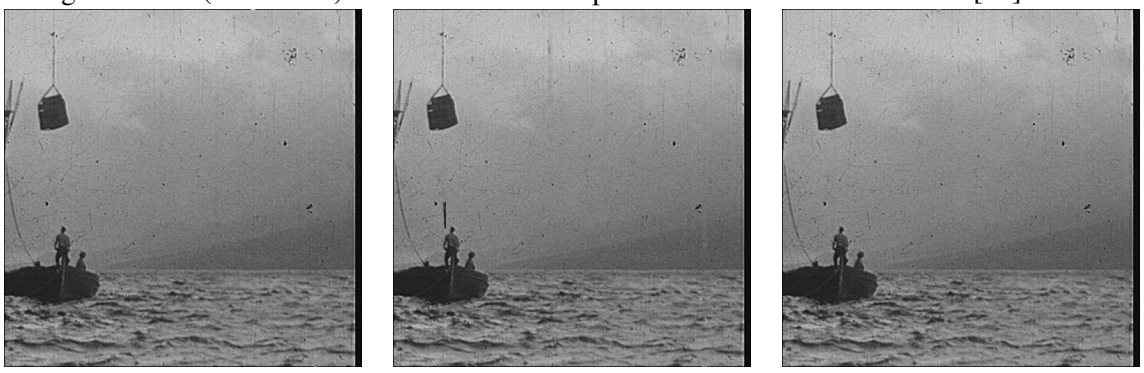
It is clear that the quality of the algorithms' results depends greatly on the situation. In particular, we observe that the algorithm of Bruni *et al.* (our implementation) is particularly well-adapted to situations with high levels of noise and film grain, (see the "Sitdown" frame, for example). However, in situations where the image content is completely occluded ("Face"), such an approach is clearly limited. We observe that our video inpainting algorithm works well in such situations.

The approach of Güllü *et al.* gives quite good results on most sequences. It is able to reconstruct both high and low frequency components, which leads to a coherent restoration. Another interesting point is the fact that the "search space" restriction, which is necessary in the approach of Güllü *et al.*, is extremely useful. Indeed, in the case of small, thin occlusions, there is very little point in allowing the algorithm to search throughout the entire image sequence. The "face" example shows a case where this approach fails. This is due to the fact that the scratch stays in the same position (over the eye) for a number of frames which is greater than the temporal search parameter. While the parameter could obviously be tuned, this shows that such a method is potentially limited. We note that generally speaking, it produces good results when viewed on a frame-by-frame basis. However, the lack of temporal coherence is clearly visible when the video is seen at a normal speed.

Next, it is quite clear that the optical flow-based restoration approach developed in 7.4 is inadequate in many situations. In fact, the only sequence in which it produced good results was the "face" sequence. This is because the face of the person does not move very much, meaning that a good trajectory for each pixel is relatively easy to establish. In the "Sitdown", "Dance" and "Les loulous" sequences, where the trajectories are not at all obvious, the method does not produce good results.

Finally, we see that the video inpainting method described in Section 6.5 produces good results on all examples except for "Dance". This is because the patches corresponding to the back of the dancing girl are not repeated throughout the video. While this is clearly a disadvantage, we also see that the method is successful in most situations. In particular, we see that we are able to correctly identify textured areas ("Sitdown"), and recreate details of various objects ("Face" and "Les loulous").

Original frame ("Sitdown") | Linear interpolation | Bruni [27]

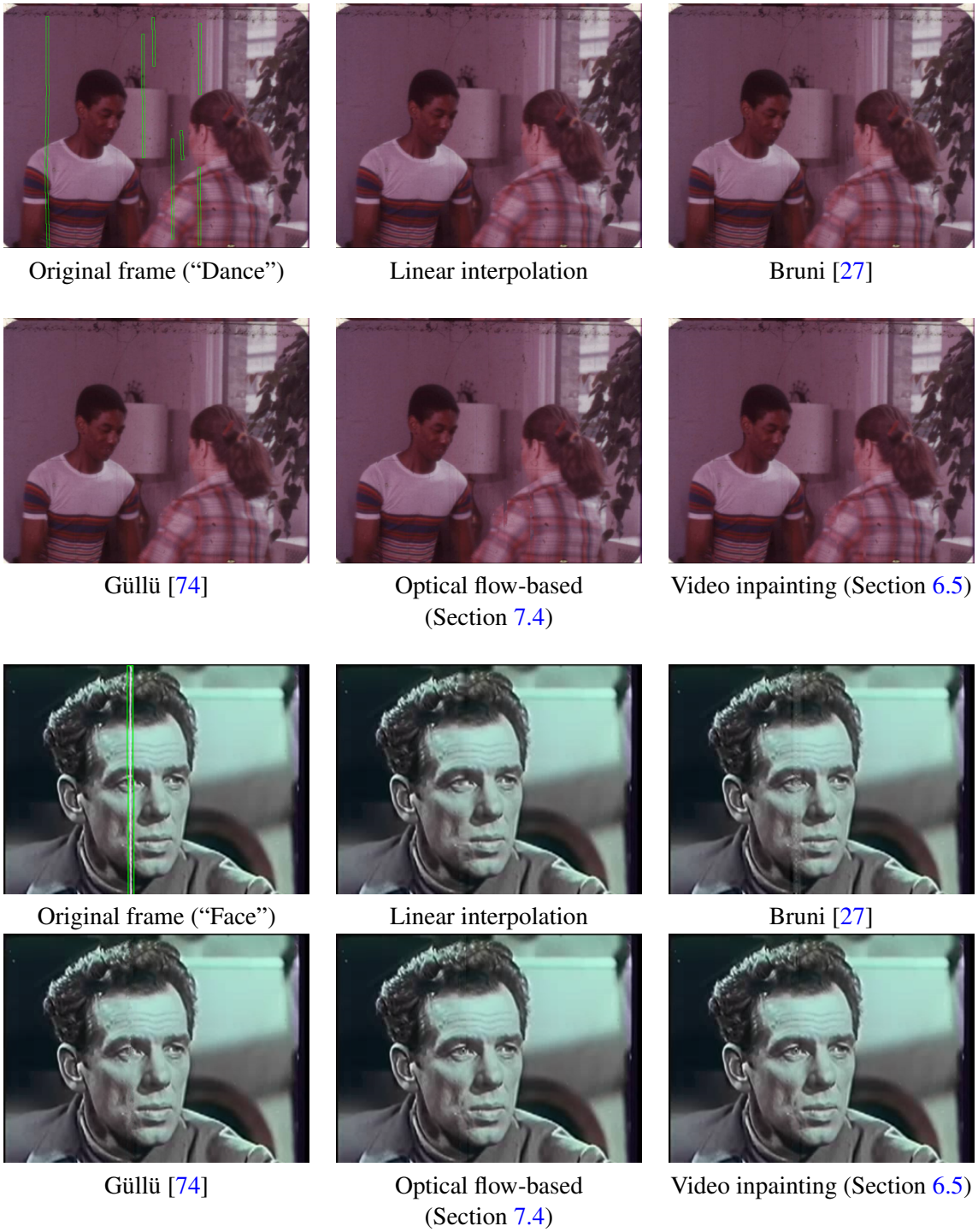


Güllü [74] | Optical flow-based (Section 7.4) | Video inpainting (Section 6.5)

Original frame ("Star") | Linear interpolation | Bruni [27]

Güllü [74] | Optical flow-based (Section 7.4) | Video inpainting (Section 6.5)

FIGURE 7.3: Restoration of some image sequences commonly found in the scratch restoration literature, using several different restoration approaches. The scratches are outlined in green.



Original frame (“Dance”) | Linear interpolation | Bruni [27]

Güllü [74] | Optical flow-based (Section 7.4) | Video inpainting (Section 6.5)

Original frame (“Face”) | Linear interpolation | Bruni [27]

Güllü [74] | Optical flow-based (Section 7.4) | Video inpainting (Section 6.5)

FIGURE 7.3: Scratch restoration on a sequence with fast motion (“Dance”) and another with details which are impossible to retrieve in a frame-by-frame manner (the eye of the man in the “face” example).

Original frame (synthetic scratches)

Linear interpolation

Bruni [27]

Güllü [74]

Optical flow-based (Section 7.4)

Video inpainting (Section 6.5)

FIGURE 7.4: Restoration of synthetic scratches on a high definition video (1280×720 pixels). The width of the scratches was extrapolated from the width of lower resolution scratches.

An interesting note is that in many situations (*e.g.* “Sitdown” and “Dance”), a simple linear interpolation produces relatively good results. This is not surprising since scratches are relatively thin. However, this approach is clearly insufficient for high resolution videos.

7.6 Conclusion on scratch restoration

Restoring line scratches is a very specific task, and having seen several examples and compared different restoration algorithms, we can draw certain conclusions. Firstly, in many cases it is important to maintain textures, as old films often contain high levels of noise and film grain. Furthermore we have seen that, even for small, thin occlusions it is a very difficult task to create pixel trajectories which are reliable enough to use for restoring scratched zones. However, the problem of establishing such trajectories is a subject of increasing interest in the image processing domain, and has been studied in works such as [39]. If such trajectories are available, algorithms such as [54] are able to fill in the occluded (scratched) regions successfully. However, it is clear that obtaining the trajectories themselves is difficult.

According to our experiments, our video inpainting approach appears successful at restoration, in particular in high definition sequences. The other patch-based restoration method which we tried (that of Güllü*et al*.) is also generally very successful. The clear drawback to this method is that the lack of temporal coherence is visible when the video is played at normal speed, especially in the high definition example. This shows that for high quality and resolution restoration, video inpainting methods may be preferable, even if they are at the moment quite slow.

One drawback of using video inpainting for scratch restoration is that we consider that all of the information is unknown in the scratched area. This is clearly not always the case, and represents a worst case scenario. In reality, some information is often available, and several restoration approaches use this information [27, 76]. Therefore, it is probable that the most successful sort of approach to line scratch restoration would be a combination of a video inpainting technique with a semi-transparent scratch model.

Chapitre 8

Conclusion

This thesis has explored two separate goals of the image and video restoration and editing processes. The first of these goals is the precise and reliable detection of line scratches in films, and the second is the task of inpainting in videos.

Chapter 3 dealt with the detection of line scratches in still frames. The proposed algorithm produces a spatially precise description of the line scratches, and is robust to the presence of noise and clutter in the image. This robustness is due to the *a contrario* detection methodology which is put to use in our algorithm. In particular, we have extended the classical *a contrario* method by introducing a background noise model which is spatially varying. We have provided visual and quantitative evaluation of the algorithm and shown its strengths with respect to previous work.

In Chapter 4, the subject of temporal filtering of line scratch detections was considered. The goal of such filtering is the removal of false alarms which are caused by thin vertical structures in the scene. The filtering is based on temporal information contained in the scratched image sequences, and in particular the motion of scratch detections in comparison with the scene motion. Using a hypothesis concerning this motion, we propose an algorithm which is able to remove false detections which are due to structures which belong to the scene. The approach employs an estimation of an affine, global motion which reflects the scene motion, and groups scratch detections into coherent trajectories by reusing the algorithm of Chapter 3. Again, we evaluate our algorithm visually and quantitatively in comparison with previous work, and find that the algorithm improves the precision of the purely spatial detection method.

In Chapter 6 the problem of video inpainting was dealt with. An approach based on the multi-resolution minimisation of a global patch-based functional was used. We looked at some of the problems which arise naturally in such a framework, such as the computationally expensive search for ANNs of spatio-temporal patches. This problem was addressed by proposing an extension of the PatchMatch algorithm [11] to the spatio-temporal case. The resulting search scheme leads to a speedup of 20-50 times with respect to the previously used ANN search algorithm [6]. Next, we have tackled the problem of successfully inpainting dynamic textures. We identified a drawback of using the standard ℓ^2 patch distance for this purpose and proposed a new distance which includes simple gradient-based texture features that reflect the texture content of the patch. These features are implemented in the form of a multi-resolution pyramid, which is used to "guide" the inpainting towards using patches of similar texture. Next, we have considered the problem of inpainting with moving cameras or background. This poses a problem in a patch-based inpainting framework, as the essential property necessary for such methods to work (the repetition of spatio-temporal patches) no longer holds. We address this by including a realignment (or stabilisation) pre-processing step which ensures that patches are self-similar. We have also discussed some other

very important algorithmic details which have an important influence on the inpainting result. The result of this work is an algorithm is able to deal with a wide variety of inpainting situations in an automatic manner, with a speedup of an order of magnitude with the most recent approach [58] (which is a semi-automatic approach). Furthermore, our algorithm is able to deal with this range of situations in a single, unified framework, rather than having to create separate algorithms for each task as in [57, 58].

In Chapter 7 we looked at some results of scratch restoration with our video inpainting algorithm. We saw that in some situations, restoration can be carried out relatively well with very simple approaches. However, in more complex situations, especially when image content has been completely removed, we see that the use of video inpainting is justified. In particular such methods may be necessary for the restoration of high resolution videos.

8.1 Further work

1. One of the most interesting aspects of the spatial line scratch detection algorithm presented in Chapter 3 is the introduction of a spatially varying background noise model to the *a contrario* methodology. This extended framework is possible thanks to Hoeffding's approximation of the Binomial distribution tail, which holds valid for sums of *independantly* distributed variables, and not just those which are also follow the same distribution. Furthermore, in our algorithm, we do not calculate the detection probability under the noise model theoretically, rather we estimate it from the observed pixel-wise scratch detections. While this approach obviously makes the assumption that false detections reflecting the noise model are far more present in the detection image than true scratch detections, it has a clear algorithmic appeal. The estimation of the background model is both simple and fast. It would be very interesting to see if such an approach could be used for other detection applications.
2. The temporal filtering algorithm in Chapter 4 makes certain simplifying hypotheses. In particular, it is assumed that the false detections belong to a scene which moves in a coherent fashion. This clearly disregards the possibility of several independently moving objects which cause false scratch detections. One possibility to addressing this issue is to estimate several dominant motions in the scene, and realign the detections based on the motion present in the local area around each detection. Unfortunately, this sort of approach necessarily leads to other complicated questions such as the number of motions present in the scene, the fineness of the motion estimation and what to do if a scratch detection belongs to several different motion regions. In reality, the true goal we wish to achieve is the tracking of *all* scratch detections which have definite trajectories. As seen in Section 4.4, this problem is quite difficult, and the algorithm which we developed (but did not finally retain) for the purpose lacked robustness due to its local and greedy approach. Future work in this direction would need to consider the problem in a more global manner. We have mentioned the work of Mael Primet [104] which detects trajectories based on an *a contrario* approach. The definition of the NFA is formulated in the case of accelerating trajectories. One of the drawbacks of such an approach is its combinatorial nature, which increases computational complexity. One option would be to use the clustering approach presented in Section 4.4, and to extract trajectories based on the NFA criterion as formulated by Primet [104].
3. Section 6.5 explored a wide range of challenges which arise in video inpainting. The first challenge we faced was the acceleration of the ANN search, a step which was absolutely necessary to produce a usable algorithm with which we could experiment. While this lead to

a significant acceleration, high definition videos still take a long time to process, for example around 5 hours for a video of $1120 \times 754 \times 200$ pixels. A further improvement which could be taken into consideration is a transformation of the patch representation in order to calculate the patch distance faster. We have considered using a PCA, however a drawback of this method is the time taken to establish the principal components of the unoccluded patches (at each pyramid level), and the projection of the patches being reconstructed onto this basis. Other transformations with smaller time complexities could instead be considered.

4. The proposed video inpainting algorithm showed that simple gradient-based texture features greatly improve the inpainting result. These are purely spatial, which naturally raises the question of whether such features could be calculated in a spatio-temporal context. This could potentially make patch comparisons more discriminative, leading to results of higher quality.
5. In Section 6.4.2, we made some remarks about the convergence properties of a non-local patch-based variational inpainting approach which is very similar to the approach which we used for video inpainting. We studied a very simple 1D structure inpainting case, and showed that the algorithm could indeed get stuck in local minima, depending on the relationship between the patch size and the occlusion size. It would be of great interest to see if some of these results could be extended to a more general setting, for example with a wider variety of patches to choose from in the inpainting process. Given that we have seen that the non-local patch-based variational inpainting approach may be applied to the 1D, 2D and 3D cases, it would be interesting to see if we can guarantee any properties by using certain hypotheses about the signal to be reconstructed.

Annexe A

Motion estimation

A.1 Introduction

The estimation of apparent motion in image sequences is an old problem in image and video processing, and it arises frequently in the context of many other domains such as computer vision, compression, object tracking and others. Since we use the tools of motion estimation at various points in our work (Section 4.5, Section 6.5.5 and Section 7.4), we recall certain important notions of this domain. We note that this is just a brief tour, as the motion estimation literature is vast.

Let us consider the content of a pixel $p = (x, y, t)$. The estimation of the motion of this pixel is most often based on the *brightness constancy hypothesis*. This states that the content of the pixel p should be found at another position $(x + \Delta x, y + \Delta y)$ at time $t + \Delta t$. This is formalised with the *brightness change constraint equation* :

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t). \tag{A.1}$$

A Taylor series expansion of this equation gives :

$$I(x, y, t) = I(x, y, t) + \Delta x \frac{\partial I(x, y, t)}{\partial x} + \Delta y \frac{\partial I(x, y, t)}{\partial y} + \Delta t \frac{\partial I(x, y, t)}{\partial t} + ... \tag{A.2}$$

If we consider small motions, we may ignore the higher order terms in the Taylor expansion. This yields :

$$\begin{aligned} \frac{\Delta x}{\Delta t} \frac{\partial I(x, y, t)}{\partial x} + \frac{\Delta y}{\Delta t} \frac{\partial I(x, y, t)}{\partial y} + \frac{\partial I(x, y, t)}{\partial t} &= 0 \\ u I_x(x, y, t) + v I_y(x, y, t) + I_t(x, y, t) &= 0, \end{aligned}$$

where u and v are the components of the optical flow vectors, which we would like to estimate. The subscripts represent partial derivatives. This is the equation upon which many motion estimation techniques are based.

Since several types of motion estimation exist, it is necessary to explain certain terms found in the literature. Motion estimation can be done in a *dense* or a *sparse* manner. Dense estimation means that the motion is estimated at every point (or pixel) in the image. Conversely, sparse estimation is only carried out at certain points in the image. In particular, this is done when *interest points* or *features* are used.

Motion estimation can also be *parametric* or *non-parametric*. This corresponds to the choice of motion model. In the case of parametric motion, the motion vector at a certain position is considered to be a function of this spatial position, and a certain number of parameters which have to be

estimated. The number of parameters is usually quite small (around six for an affine motion model) and thus storing the motion requires very little memory. This type of motion usually supposes some sort of smooth model which are known to approximate the apparent 2D motion of rigid 3D objects well, at least over some spatial extent. It is therefore quite useful for estimating motions due to a camera such as panning, zooming and rotation. Non-parametric motion estimation, on the other hand, estimates a motion vector for each considered position. This obviously requires much more memory and computation in general than in the parametric case. On the other hand, each vector may have an arbitrary value, which makes the estimation more flexible and able to incorporate discontinuities into the vector field. This is particularly important when the motions of several objects, or that of objects with articulated moving parts (*e.g.* the limbs of a person) need to be estimated.

A.2 Dense differential methods

Let us first analyse in depth one of the most common forms of motion estimation : differential methods.

The first work on the subject was carried out by Horn and Schunck [66], and is based on the constant brightness equation (A.3). Unfortunately, this equation is under-constrained, as there are two unknowns, and only one equation. Therefore, further assumptions are introduced in order to make the problem tractable. The most common hypothesis is spatial *smoothness*, which stipulates that the optical flow vector field should be as smooth as possible. This is reasonable in the sense that the motion of an object should not vary abruptly.

Including this prior is done by setting a functional to minimise :

$$\int (I_x u + I_y v + I_t)^2 + \lambda(|\nabla u|^2 + |\nabla v|^2), \tag{A.3}$$

where $\Delta u = \frac{\partial^2 u}{\partial x^2}$ and $\Delta v = \frac{\partial^2 u}{\partial x^2}$ represent the Laplacians of u and v. The corresponding Euler-Lagrange equation reads at each pixel :

$$\begin{cases} I_x^2 u + I_x I_y v - \lambda \Delta u &= -I_x I_t \\ I_y I_x u + I_y^2 v - \lambda \Delta v &= -I_y I_t \end{cases} \tag{A.4}$$

These equations are solved numerically with a discretisation scheme, such as finite differences.

A.3 Multi-resolution estimation

We recall that the previous formulation of the optical flow problem required the motion vectors to be small, since the Taylor series development of $I(x+\Delta x, y+\Delta y, t+\Delta t)$ was used. Therefore, in order to deal with large motions, a multi-resolution scheme is often used in order to reduce the scale of the motion, and Equation (A.3) is minimised at a coarse pyramid level. The resulting solution is then upsampled, and the first of the two images at the finer level is *warped* to the second one. The same procedure is applied to the warped image, and the resulting motion vectors are simply the sum of the initial (coarse) vectors and the fine vectors.

Many variants on the previous method are available. The well-known Lucas-Kanade method [90] assumes that the optical flow is locally constant, which yields several equations for the brightness consistency hypothesis and constrains the problem. In fact, the number of equations means that the problem is over-constrained, and is therefore resolved using a least squares approach.

Other well-known differential methods deal with such issues as respecting discontinuities in the motion vector field [17], including additional constraints in the optical flow equation [22], and improving the optimisation scheme itself [127].

Other methods exist which belong to the optical flow category. They include frequency-domain methods and cross-correlation-based methods. Frequency-domain methods set the hypothesis that the phase of the image signal is preserved between images, therefore they solve the following equation : $\frac{d\phi}{dt} = 0$, where ϕ is the phase. In the case of cross-correlation, the motion is defined as the displacement which maximises the cross-correlation.

A.4 Block matching

Block matching is perhaps the simplest method of estimating local motion. The method consists in searching locally in the frame $t+1$ for the information found at a certain position p in the frame number t. Obviously, if this is done on a pixel-resolution level it will not work : since there is no regularisation (each "search" is done independently), there is no way to deal with the motion of smooth objects or areas. Therefore, the method searches for the information contained in a "block". The position p' of the matching block in the next frame is found with :

$$p' = \underset{q \in \mathcal{N}_s(p)}{\arg\min} \sum_{r \in \mathcal{N}(q)} d(I(r, t+1), I(r, t)), \tag{A.5}$$

where $\mathcal{N}_s$ is the spatial search area, and $\mathcal{N}$ is the spatial block neighbourhood, and $d(I(q, t+1), I(q, t))$ is a comparison distance. The spatial search area of p $\mathcal{N}_s$ is generally an area around p in the frame $t+1$, and should reflect the maximum motion which we expect in the sequence. A large part of the work on these methods [86, 89, 96, 103] deals with the best way of searching for the correct block in the frame $t+1$.

A.5 Parametric motion models

The previous set of methods estimate an motion vector for each pixel, and is therefore quite flexible. On the other hand, it has high memory requirements. Furthermore, in some situations, additional information is available concerning the nature of the motion. For example, if we are sure that the apparent motion is a simple translation for the entire image (*e.g.*, a camera pan with an approximately fronto-parallel scene), then the entire motion can be described with one 2D vector. Another common model is the *affine* motion model, which includes translations. This model is appreciated since it has relatively few parameters to estimate and can represent motions such as translation, zooming and rotation. In this model, the motion vector at a pixel $p = (x, y)$ is defined as follows :

$$\left\{ \begin{array}{rcl} u(p) & = & c_1 + a_1 x + a_2 y \\ v(p) & = & c_2 + a_3 x + a_4 y \end{array} \right. , \tag{A.6}$$

where c_1 and c_2 are the parameters describing the constant motion components, and $a_1 ... a_4$ are the parameters associated with the spatially varying components of the motion. In homogeneous form this can be written as :

$$\left[\begin{array}{c} u(p) \\ v(p) \\ 1 \end{array} \right] = \left[\begin{array}{ccc} a_1 & a_2 & c_1 \\ a_3 & a_4 & c_2 \\ 0 & 0 & 1 \end{array} \right] \left[\begin{array}{c} x \\ y \\ 1 \end{array} \right] \tag{A.7}$$

The parameter matrix must now be estimated. This amounts to finding the transformation that minimises the compensation error. One such algorithm is that of Odobez and Bouthemy [97], who add robust estimators and a multi-resolution scheme to the parametric motion estimation.

A visual example of such an affine parametric motion vector field may be found in Figure A.1. In this case, the motion is a camera zoom, centred around the head of the woman on the left. This example illustrates the capacity of an affine motion model to describe various motion types.

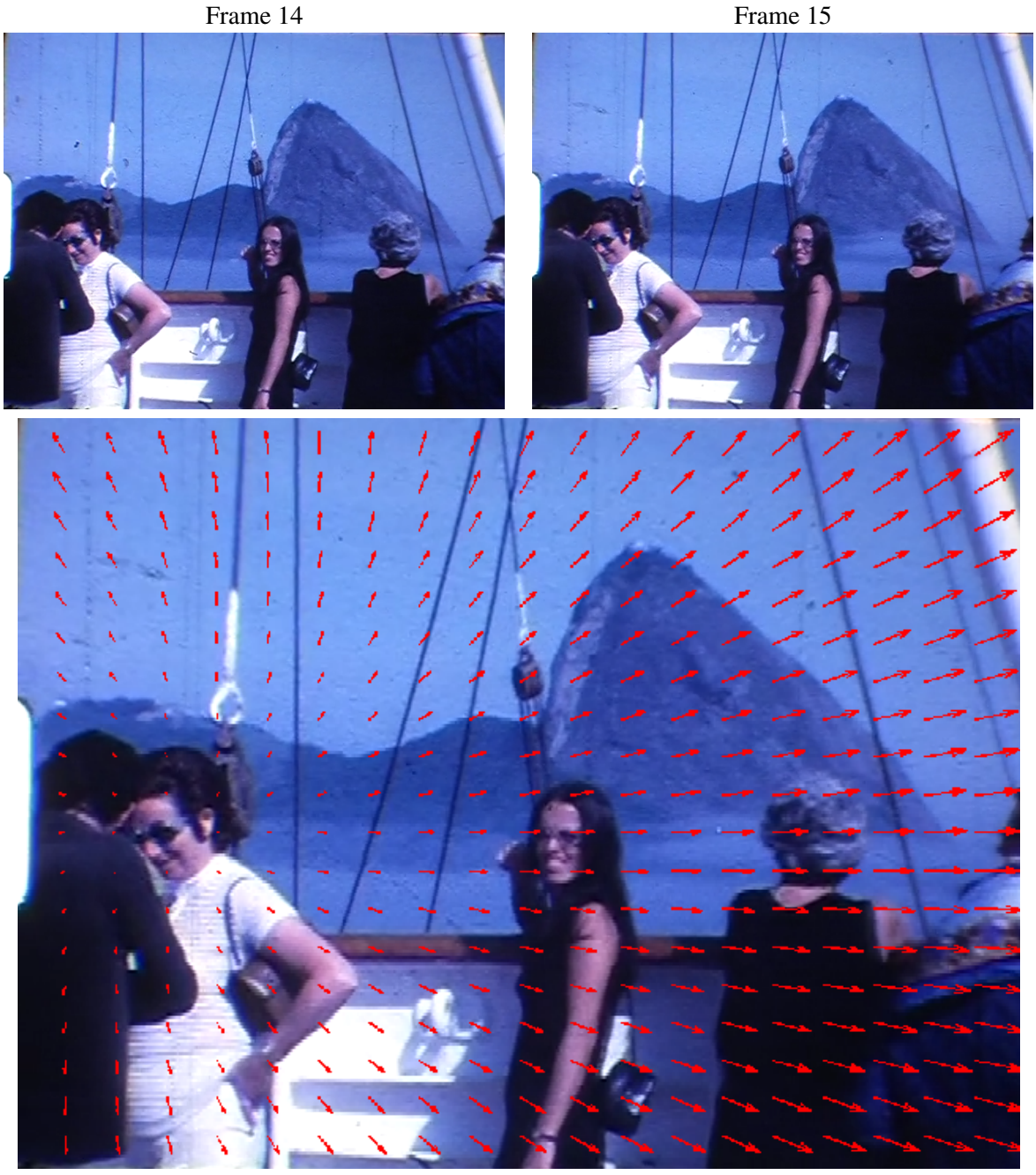


FIGURE A.1: Parametric motion vector, represented as arrows. In this case, the motion is a zoom towards a point in the centre left part of the image. Such motions may be correctly described using affine motion.

Annexe B

Proof of the critical occlusion size for binary inpainting convergence

In this Appendix, we establish certain theoretical convergence properties of a simple non-local patch-based variational inpainting approach (see Algorithm 3), in a very simple toy example. We recall that we are in a 1D situation, that is we are trying to inpaint a function $y : \mathbb{N} \to \mathbb{R}$ which is equal to 1 everywhere, apart from inside the occlusion. We recall that we have restricted our patch dictionary to two elements Ψ_0 and Ψ_1 such that : $\Psi_0(i) = 0, \forall i = 1 \cdots N$ and $\Psi_1(i) = 0, \forall i = 1 \cdots N$, where N is the patch size. in this situation, we know that the global minimum is a function where y is equal to 1 everywhere. When we refer to the *critical occlusion size*, we mean the size of the occlusion below which the function y will converge to the global minimum, and above which y will become stuck in a local minimum.

We assume the patch size to be odd, so that $\exists h, N = 2h + 1$. In Section 6.4.2, we showed that the two steps of Algorithm 3, nearest neighbour search and reconstruction, could be re-written as a series of convolution and thesholding operations. In particular, if we define the binary function $g : \mathbb{N} \to \{0, 1\}$, then we can say that a function is stable under the inpainting scheme if :

$$g(x) = S[(g * \omega * \omega)(x), \frac{1}{2}], \forall x \in \mathbb{N}, \tag{B.1}$$

where $S[x, a]$ is a thresholding operation such that :

$$S[x, a] = \begin{cases} 0 & \text{if } x < a \\ 1 & \text{otherwise} \end{cases} \tag{B.2}$$

The function $g(x)$ is somewhat analogous to the shift map function ϕ in the general inpainting setting. That is, if $g(x) = 0$ then the NN of the patch W_x is Ψ_0, and if $g(x) = 1$ then the NN is Ψ_1. The function g is much easier to study than y, therefore we shall look at its evolution rather than that of y. If we wish to find y, it is simply given by $g * \omega$.

Our goal here is to consider a special family of functions $g_{p,q}$ such that :

$$g_{p,q}(x) = \begin{cases} 0 & \text{if } p \leq x \leq q \\ 1 & \text{otherwise} \end{cases}, \tag{B.3}$$

and to see if applying the inpainting scheme will result in convergence to the global minimum or not. To do this, we shall study the evolution of the point p after the inpainting is applied. We consider this family of functions to represent a "bad" inpainting initialisation. We refer to the

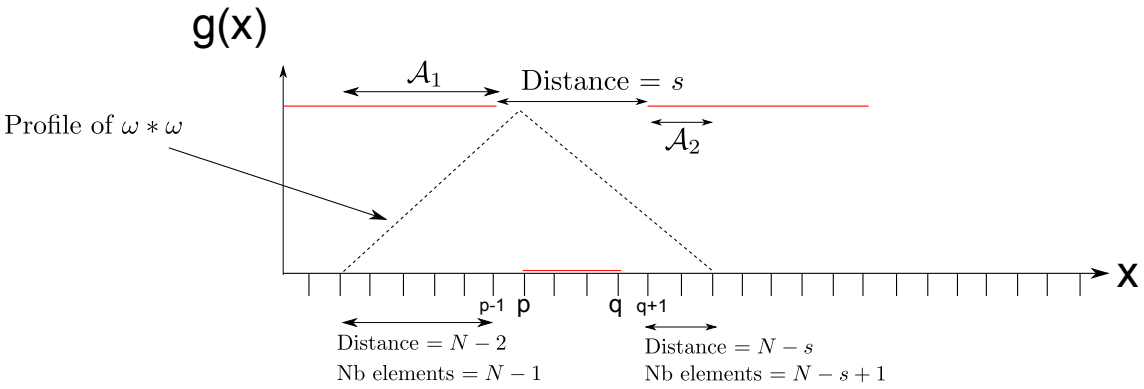


FIGURE B.1: Example of $g_{p,q}$, with no fixed border values. Whether the solution converges or not will depend on the relation between the patch size N and the occlusion size s.

distance between $p-1$ and $q+1$ as s. This situation is described in Figure B.1. It is the 1D equivalent of the situation we had in Section 6.4.2 (see Figure 6.4).

B.1 Critical occlusion size, without fixed border values

In our demonstration, we shall first consider the critical occlsion size in the case where we are allowed to modify $g_{p,q}$ *everywhere* ($\forall x \in \mathbb{N}$). This is not how inpainting is truly done, since in reality we are not allowed to modify the pixels outside of the occlusion. We shall refer to these fixed values of $g_{p,q}$ as *fixed border values*.

Proposition 3. *The non-local inpainting algorithm will converge to the global minimum for* $g_{p,q}$ *when* $s \leq N + \frac{3}{2} \pm \sqrt{N + \frac{1}{4}}$*, when fixed border values are not taken into account.*

Proof : We wish to see if $g_{p,q}(p)$ evolves with the inpainting algorithm, and if so, under what conditions. For this, we need to calculate the value of $g_{p,q} * \omega * \omega$. This is given by :

$$\begin{aligned}(g_{p,q} * \omega * \omega)(p) &= \frac{1}{N^2}\left[\mathcal{A}_1 + \mathcal{A}_2\right] \\ &= \frac{1}{N^2}\left[\frac{(1+(N-1))(N-1)}{2} + \frac{(1+(N-s+1))(N-s+1)}{2}\right] \\ &= \frac{1}{N^2}\left[\frac{(N^2-N)+(N-s+2)(N-s+1)}{2}\right]. \end{aligned} \tag{B.4}$$

Therefore we have :

$$\begin{aligned}(g_{p,q} * \omega * \omega)(p) &< \frac{1}{2} \\ (N^2-N)+(N-s+2)(N-s+1) &< N^2 \\ s^2 + s(-2N-3) + 2N + 2 &< 0 \end{aligned} \tag{B.5}$$

The solution(s) of this quadratic equation are the following :

$$s = N + \frac{3}{2} \pm \sqrt{N + \frac{1}{4}} \tag{B.6}$$

B.2 Taking into account fixed border values

The previous proposition gives us a condition under which the non-local patch-based variational algorithm described in Algorithm 3 will converge in a very simple, 1D situation. However, as we have mentioned, this is not the complete picture : in the previous calculation, we have not taken into account the fixed border values, which are not allowed to be modified during the inpainting process. To study this in more detail, we shall consider a function g_p such that :

$$g_p(x) = \begin{cases} 0 & \text{if } p \geq x \\ 1 & \text{otherwise} \end{cases} \tag{B.7}$$

Furthermore, we consider that this function is not modifiable for all positions on smaller or equal to a position b, such that $b < p$. The positions $x \leq b$ represent the fixed border values. We want to know what the minimum distance between b and p is, so that the function g_p is stable. We define r to be the distance between b and $p - 1$. This situation is represented in Figure B.2.

Proposition 4. *The function g_p is stable under the inpainting algorithm (3), with the following condition : $|p - 1 - b| \geq h + \frac{1}{2} - \sqrt{2h + \frac{5}{4}}$, where b is the position of the fixed border values such that $g_p(x)$ is not modifiable for all $x \leq b$.*

Proof : As previously, we wish to calculate the value of $(g_p * \omega * \omega)(p)$. Figure B.2 shows the function g_p and its convolution with ω.

We have :

$$(g_p * \omega * \omega)(p) = \frac{1}{(2h+1)} \left[h - r + \left(\frac{(g_p * \omega)(p_1) + (g_p * \omega)(p_2)}{2} \right) (h + r) \right] \tag{B.8}$$

The points p_1 and p_2 are shown on the diagram in Figure B.2. We know that $(g_p * \omega)(p_2) = \frac{1}{2h+1}$. Therefore :

$$(g_p * \omega)(p_1) = \frac{1}{2h+1} + \frac{h + r - 1}{2h+1} = \frac{h + r}{2h+1}, \tag{B.9}$$

since the slope between p_1 and p_2 is $-\frac{1}{2h+1}$. Putting this together, we have :

$$(g_p * \omega * \omega)(p) = \frac{1}{(2h+1)} \left[h - r + \frac{(h + r + 1)(h + r)}{2(2h+1)} \right] \tag{B.10}$$

Now, we want to find whether $(g_p * \omega * \omega)(p)$ is less than $\frac{1}{2}$, in which case the algorithm converges. Therefore :

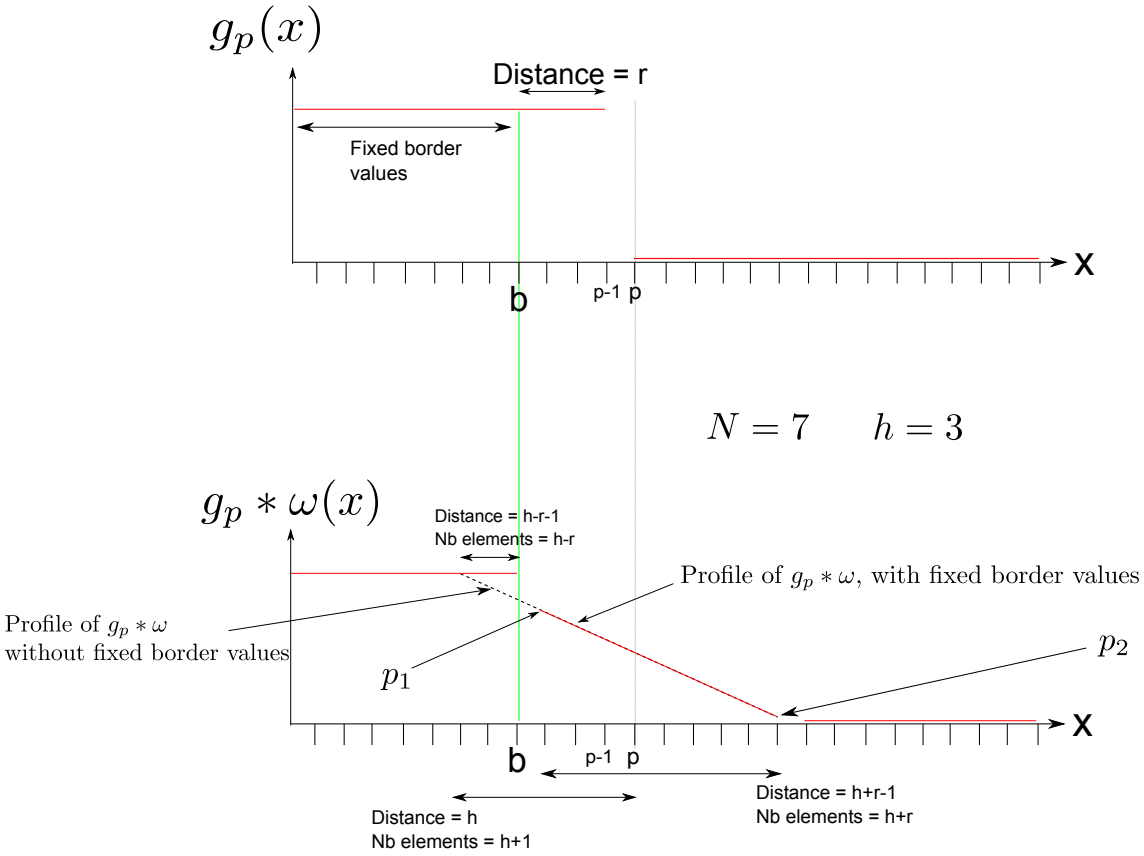
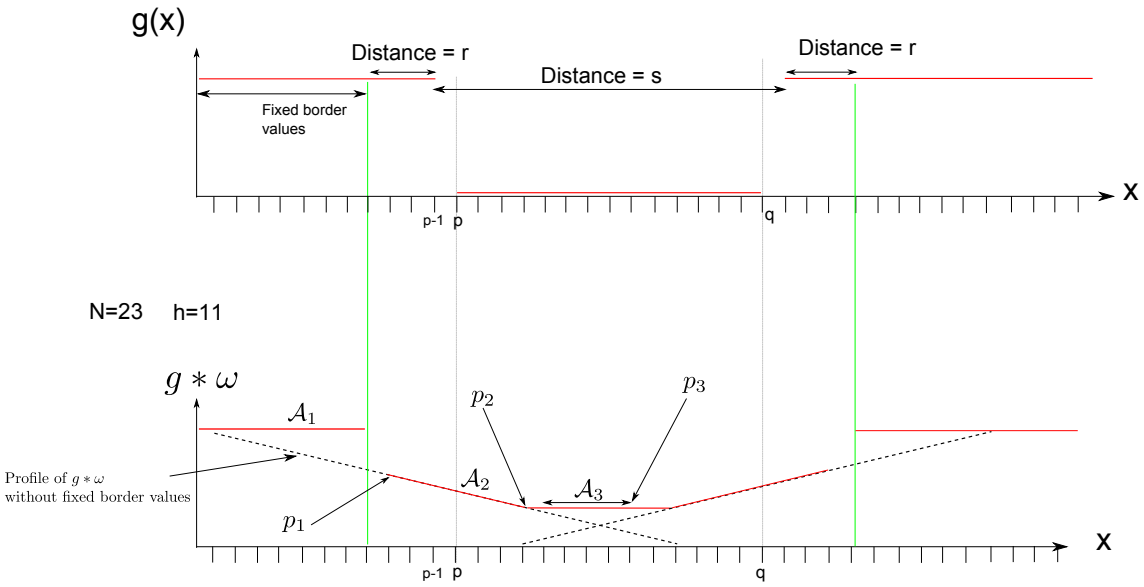


FIGURE B.2: Example illustrating the effects of the fixed border values on a stable solution g_p. The values of the function g are not modifiable for all $x \leq b$. Intuitively, the fixed border values "push" the position of p towards the right.

$$
\begin{aligned}
(g_p * \omega * \omega)(p) &\leq \frac{1}{2} \\
2h - 2r + \frac{h + r + 1}{2h + 1}(h + r) &\leq 2h + 1 \\
-2r - 1 + \frac{h^2 + hr + hr + r^2 + h + r}{2h + 1} &\leq 0 \\
-(2r + 1)(2h + 1) + h^2 + 2hr + r^2 + h + r &\leq 0 \\
-4hr - 2r - 2h - 1 + h^2 + 2hr + r^2 + h + r &\leq 0 \\
r^2 + r(-2h - 1) + h^2 - h - 1 &\leq 0 \qquad \text{(B.11)}
\end{aligned}
$$

The two critical points of this equation are $r = h + \frac{1}{2} \pm \sqrt{2h + \frac{5}{4}}$. Now, it is reasonable to assume that $r \leq h + 1$, because we suppose the fixed border values to be within a distance of h from the point p. Otherwise, they would have no effect on the evolution of $g_p(p)$.

Therefore, if we have $r \geq h + \frac{1}{2} - \sqrt{2h + \frac{5}{4}}$, then we have a stable inpainting solution with the function g_p. We shall see that this calculation will help us find the final result concerning the critical distance for convergence of the inpainting algorithm.



$$\mathcal{A}_1 = h - r$$
$$\mathcal{A}_2 = \frac{((g*\omega)(p_1)+(g*\omega)(p_2))(s+r-h-1)}{2}$$
$$\mathcal{A}_3 = \frac{N-s+1}{N}(N-s+1)$$

FIGURE B.3: Critical solution, with fixed border values. This will converge, or not, to a "good" solution depending on the total occlusion size $2r + s$.

B.3 Critical occlusion size, taking into account fixed border values

We now look at the critical occlusion size in the true inpainting situation, that is to say taking into account the fixed border values. We will consider a function $g_{p,q}$ as in Section B.1, but to simplify notation we will refer to it simply as g. Figure B.3 represents this situation. Ultimately, we wish to know the total distance separating the fixed border values which will guarantee convergence. In Figure B.3, this total distance is given by $2r + s$. We have labelled three areas : $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$. They represent :

- $\mathcal{A}_1$: the fixed border values which must be reset to 1 after the operation $g * \omega$;
- $\mathcal{A}_2$: the descending slope corresponding to the "normal" values of $g * \omega$;
- $\mathcal{A}_3$: those values $g*\omega(x)$ which are constant because $|x-(p-1)| \leq h$ and $|(q+1)-x| \leq h$.

These three areas influence the value of $g(p)$. More formally, we have :

$$(g * \omega * \omega)(p) = \frac{\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3}{2h+1}. \tag{B.12}$$

We have $\mathcal{A}_1 = h-r$. We also know that $p_1 = p-r$ and $p_2 = p-1+s-h-1 = p+s-h-2$. Therefore, the number of points belonging to $\mathcal{A}_2$ is $p_2 - p_1 + 1 = (p+s-h-2) - (p-r) + 1 = s+r-h-1$. For simplicity, we suppose that s is odd, which means that the value of $(g*\omega)(p_2)$ is $\frac{N-s+1}{N}$. The number of points in $\mathcal{A}_3$ is $p_3 - p2 + 1 = p+h-(p+s-h-2)+1 = N-s+1$. Therefore we have :

$$(g * \omega * \omega)(p) = \frac{1}{2h+1}\left[h - r + \frac{((g*\omega)(p_1) + (g*\omega)(p_2))(s+r-h-1)}{2} + \frac{(2h+1-s+1)^2}{2h+1}\right]. \quad \text{(B.13)}$$

We know that $(g*\omega)(p_2) = \frac{2h+1-s+1}{2h+1}$, and $(g*\omega)(p_1) = \frac{2h+1-s+1}{2h+1} + \frac{(s+r-h-2)}{2h+1}$. This gives :

$$(g * \omega * \omega)(p) = \frac{1}{2h+1}\left[h - r + \frac{(2\frac{2h+1-s+1}{2h+1} + \frac{s+r-h-2}{2h+1})(s+r-h-1)}{2} + \frac{(2h+1-s+1)^2}{2h+1}\right]. \quad \text{(B.14)}$$

After some developing, we obtain :

$$(g * \omega * \omega)(p) = \frac{1}{2h+1}\frac{9h^2 + s^2 + r^2 + 13h - 2hr - r - 4hs - 5s + 6}{4h+2}. \quad \text{(B.15)}$$

Setting $(g * \omega * \omega)(p) < \frac{1}{2}$ leads to :

$$5h^2 + 9h - 2hr - r - 4hs - 5s + s^2 + r^2 + 5 < 0. \quad \text{(B.16)}$$

This is an equation with two unknowns, however we also have Equation (B.11). If we subtract the Equation (B.11) from Equation (B.16), we obtain :

$$s^2 + s(-4h - 5) + 4h^2 + 10h + 6 < 0. \quad \text{(B.17)}$$

The critical point of this equation is $s = 2h + \frac{5}{2} - \frac{1}{2} = 2h + 2 = N + 1$.

B.4 Final result

The conclusion of this analysis is that the critical occlusion size is $Q = 2 * \lceil r \rceil + \lceil s \rceil = 2 * \lceil h + \frac{1}{2} - \sqrt{2h + \frac{5}{4}} \rceil + 2h + 2$. Qualitatively speaking, this is more or less equal to $2N - 2\sqrt{N} + 1$. Therefore, the conclusion of this analysis is that the critical occlusion size for the convergence of the inpainting algorithm is approximately equal to $2N - 2\sqrt{N} + 1$.

Bibliographie

[1] Physical Damage, National Film and Sournd Archive, www.nfsa.gov.au/preservation/handbook/damage-films/physical-damage/. Accessed on 27th September 2013.

[2] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic Video Textures. In *ACM Transactions on Graphics (TOG)*, pages 821–827, New York, NY, USA, 2005. ACM.

[3] C. Aguerrebere, Y. Gousseau, and G. Tartavel. Exemplar-based Texture Synthesis : the Efros-Leung Algorithm. *Image Processing On Line*, 2013 :213–231, Oct. 2013.

[4] P. Arias, V. Caselles, and G. Facciolo. Analysis of a Variational Framework for Exemplar-Based Image Inpainting. *Multiscale Modeling & Simulation*, 10(2) :473–514, Jan. 2012.

[5] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro. A Variational Framework for Exemplar-Based Image Inpainting. *International Journal of Computer Vision*, 93(3) :319–347, July 2011.

[6] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms.*, pages 271–280, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[7] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6) :891–923, Nov. 1998.

[8] J.-F. Aujol, S. Ladjal, and S. Masnou. Exemplar-Based Inpainting from a Variational Point of View. *SIAM Journal on Mathematical Analysis*, 42(3) :1246–1285, Jan. 2010.

[9] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Transactions on Image Processing*, 10(8) :1200–1211, 2001.

[10] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5) :451–460, Sept. 1975.

[11] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch : a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3), July 2009.

[12] B. Berkels, C. Kondermann, C. Garbe, and M. Rumpf. Reconstructing Optical Flow Fields by Motion Inpainting. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 388–400, Berlin, Heidelberg, 2009. Springer-Verlag.

[13] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 355–362, 2001.

[14] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, /./, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[15] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8) :882–889, Aug. 2003.

[16] B. Besserer and C. Thiré. Detection and Tracking Scheme for Line Scratch Removal in an Image Sequence. In T. Pajdla and J. Matas, editors, *Computer Vision - ECCV*, Lecture Notes in Computer Science, pages 264–275. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[17] M. J. Black and P. Anandan. The robust estimation of multiple motions : parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1) :75–104, Jan. 1996.

[18] R. Bornard, E. Lecan, L. Laborelli, and J. H. Chenot. Missing data correction in still images and image sequences. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 355–361. ACM, 2002.

[19] F. Bornemann and T. März. Fast Image Inpainting Based on Coherence Transport. *J. Math. Imaging Vis.*, 28(3) :259–278, July 2007.

[20] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *International Conference on Computer Vision*, volume 1, pages 377–384, 1999.

[21] T. Bretschneider, O. Kao, and P. J. Bones. Removal of vertical scratches in digitised historical film sequences using wavelet decomposition. In *Proceedings of the Image and Vision Computing New Zealand*, pages 38–43, 2000.

[22] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High Accuracy Optical Flow Estimation Based on a Theory for Warping. In *Computer Vision-ECCV*, volume 4, pages 25–36, 2004.

[23] J. Bruna and S. Mallat. Classification with scattering operators. In *Computer Vision and Pattern Recognition*, pages 1561–1566, June 2011.

[24] V. Bruni, P. Ferrara, and D. Vitulano. Color Scratches Removal Using Human Perception Image Analysis and Recognition. In A. Campilho and M. Kamel, editors, *Image Analysis and Recognition*, volume 5112 of *Lecture Notes in Computer Science*, chapter 4, pages 33–42. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.

[25] V. Bruni and D. Vitulano. A generalized model for scratch detection. *IEEE Transactions on Image Processing*, 13(1) :44–50, Jan. 2004.

[26] V. Bruni, D. Vitulano, and A. Kokaram. Line scratches detection and restoration via light diffraction. In *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, volume 1, pages 5–10 Vol.1, 2003.

[27] V. Bruni, D. Vitulano, and A. Kokaram. Fast removal of line scratches in old movies. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pages 827–830, Aug. 2004.

[28] A. Buades, B. Coll, and J. M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2 of *CVPR '05*, pages 60–65, Washington, DC, USA, June 2005. IEEE.

[29] A. Buadès, J. Delon, Y. Gousseau, and S. Masnou. Adaptive blotches detection for film restoration. In *International Conference on Image Processing*, pages 3317–3320. IEEE, 2010.

[30] A. Bugeau, M. Bertalmio, V. Caselles, and G. Sapiro. A Comprehensive Framework for Image Inpainting. *IEEE Transactions on Image Processing*, 19(10) :2634–2645, Oct. 2010.

[31] A. Bugeau, P. Gargallo, O. D'Hondt, A. Hervieu, N. Papadakis, and V. Caselles. Coherent Background Video Inpainting through Kalman Smoothing along Trajectories. In *Modeling, and Visualization Workshop*, 2010.

[32] J.-F. Cai, R. H. Chan, and Z. Shen. A framelet-based image inpainting algorithm. *Applied and Computational Harmonic Analysis*, 24(2) :131–149, Mar. 2008.

[33] F. Cao, Y. Gousseau, S. Masnou, and P. Pérez. Geometrically Guided Exemplar-Based Inpainting. *SIAM Journal on Imaging Sciences*, 4(4) :1143–1179, Jan. 2011.

[34] V. Caselles, J. M. Morel, and C. Sbert. An axiomatic approach to image interpolation. *IEEE Transactions on Image Processing*, 7(3) :376–386, 1998.

[35] T. Chan, J. Shen, and H.-M. Zhou. Total Variation Wavelet Inpainting. *Journal of Mathematical Imaging and Vision*, 25(1) :107–125, July 2006.

[36] T. F. Chan and J. Shen. Nontexture Inpainting by Curvature-Driven Diffusions. *Journal of Visual Communication and Image Representation*, 12(4) :436–449, Dec. 2001.

[37] S. S. Cheung, J. Zhao, and M. Vijay Venkatesh. Efficient Object-Based Video Inpainting. In *IEEE International Conference on Image Processing*, pages 705–708. IEEE, Oct. 2006.

[38] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 721–728, June 2003.

[39] T. Crivelli, P. H. Conze, P. Robert, and P. Perez. From optical flow to dense long term correspondences. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 61–64. IEEE, 2012.

[40] G. R. Cross and A. K. Jain. Markov Random Field Texture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1) :25–39, Jan. 1983.

[41] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding : combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.*, 31(4), July 2012.

[42] E. Décencière. *Restauration automatique de films anciens*. PhD thesis, Dec. 1997.

[43] M. H. DeGroot. *Optimal Statistical Decisions*. John Wiley & Sons, Inc., Hoboken, NJ, USA, Apr. 2004.

[44] L. Demanet, B. Song, and T. Chan. Image Inpainting by Correspondence Maps : a Deterministic Approach. 2003.

[45] A. Desolneux, L. Moisan, and J. M. Morel. Meaningful Alignments. *International Journal of Computer Vision*, 40(1) :7–23, Oct. 2000.

[46] A. Desolneux, L. Moisan, and J. M. Morel. *From Gestalt Theory to Image Analysis : A Probabilistic Approach*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[47] I. Drori, D. C. Or, and H. Yeshurun. Fragment-based image completion. *ACM Transactions on Graphics (TOG)*, 22(3) :303–312, July 2003.

[48] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1) :11–15, Jan. 1972.

[49] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038, 1999.

[50] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating Textures. *Computer Graphics Forum*, 27(2) :409–418, Apr. 2008.

[51] M. Elad, J. L. Starck, P. Querre, and D. L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Applied and Computational Harmonic Analysis*, 19(3) :340–358, Nov. 2005.

[52] J. H. Elder and R. M. Goldberg. Image Editing in the Contour Domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 :291–296, Mar. 2001.

[53] M. A. Elgharib, F. Pitié, and A. Kokaram. Blotch and scratch removal in archived film using a semi-transparent corruption model and a ground-truth generation technique. *EURASIP Journal on Image and Video Processing*, 2013(1) :33+, 2013.

[54] G. Facciolo, R. Sadek, A. Bugeau, and V. Caselles. Temporally Consistent Gradient Domain Video Editing. In Y. Boykov, F. Kahl, V. Lempitsky, and F. Schmidt, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 6819 of *Lecture Notes in Computer Science*, chapter 5, pages 59–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[55] H. Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93(3) :418–491, 1959.

[56] D. Goldman, E. Schechtman, C. Barnes, I. Belaunde, and J. Chien. Content-Aware Fill. Accessed on 25 January, 2014.

[57] M. Granados, K. Kim, J. Tompkin, J. Kautz, and C. Theobalt. Background Inpainting for Videos with Dynamic Objects and a Free-Moving Camera. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision âĂŞ ECCV*, pages 682–695. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[58] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, and C. Theobalt. How Not to Be Seen - Object Removal from Videos of Crowded Scenes. *Computer Graphics Forum*, 31(2pt1) :219–228, May 2012.

[59] H. Grossauer. Inpainting of Movies Using Optical Flow. In O. Scherzer, editor, *Mathematical Models for Registration and Applications to Medical Imaging*, volume 10, pages 151–162. Springer Berlin Heidelberg, 2006.

[60] M. Haindl and J. Filip. Fast restoration of colour movie scratches. In *Proceedings. 16th International Conference on Pattern Recognition*, volume 3, pages 269–272, 2002.

[61] P. E. Hart. How the Hough transform was invented [DSP History]. *Signal Processing Magazine, IEEE*, 26(6) :18–22, Oct. 2009.

[62] J. Hays and A. A. Efros. Scene Completion Using Millions of Photographs. In *ACM Transactions on Graphics (TOG)*, pages 4+. ACM, 2007.

[63] K. He and J. Sun. Computing nearest-neighbor fields via Propagation-Assisted KD-Trees. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 111–118, June 2012.

[64] K. He and J. Sun. Statistics of Patch Offsets for Image Completion. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, C. Schmid, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer VisionâĂŞECCV*, volume 7573, pages 16–29. Springer, 2012.

[65] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301) :13–30, Mar. 1963.

[66] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. In *Artificial intelligence*, volume 17, pages 185–203, 1981.

[67] P. Hough. Machine Analysis Of Bubble Chamber Pictures. In *2nd International Conference On High-Energy Accelerators*, Sept. 1959.

[68] J. Jia, Y.-W. Tai, T.-P. Wu, and C.-K. Tang. Video repairing under variable illumination using cyclic motions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5) :832–839, May 2006.

[69] L. Joyeux. *Reconstruction de séquences d'images haute résolution. Application à la réstauration de films cinématographiques.* PhD thesis, Jan. 2000.

[70] L. Joyeux, S. Boukir, and B. Besserer. Film line scratch removal using Kalman filtering and Bayesian restoration. In *Fifth IEEE Workshop on. Applications of Computer Vision*, pages 8–13. //IEEE, 2000.

[71] L. Joyeux, S. Boukir, and B. Besserer. Tracking and MAP reconstruction of line scratches in degraded motion pictures. *Machine Vision and Applications*, 13(3) :119–128, 2002.

[72] L. Joyeux, O. Buisson, B. Besserer, and S. Boukir. Detection and Removal of Line Scratches in Motion Picture Films. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 548–553, 1999.

[73] G. Kanizsa. *Organization in Vision : Essays on Gestalt Perception*. Praeger, 1979.

[74] M. Kemal Gullu, O. Urhan, and S. Erturk. Scratch detection via temporal coherency analysis and removal using edge priority based interpolation. In *IEEE International Symposium on Circuits and Systems*, pages 4591–4594, 2006.

[75] K.-T. Kim and E. Y. Kim. Film line scratch detection using neural network and morphological filter. In *Conference on Cybernetics and Intelligent Systems*, pages 1007–1011, Sept. 2008.

[76] A. Kokaram. Detection and Removal of Line Scratches in Degraded Motion Picture Sequences. *Signal Processing VIII*, I :5–8, Sept. 1996.

[77] A. C. Kokaram. *Motion Picture Restoration : Digital Algorithms for Artefact Suppression in Degraded Motion Picture Film and Video*. Springer-Verlag, London, UK, UK, 1st edition, 1998.

[78] A. C. Kokaram. Removal of line artefacts for digital dissemination of archived film and video. In *IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 245–249. //IEEE, July 1999.

[79] A. C. Kokaram, B. Collis, and S. Robinson. Automated rig removal with Bayesian motion interpolation. *Vision, Image and Signal Processing, IEE Proceedings* -, 152(4) :407–414, Aug. 2005.

[80] N. Komodakis and G. Tziritas. Image Completion Using Efficient Belief Propagation Via Priority Scheduling and Dynamic Pruning. *IEEE Transactions on Image Processing*, 16(11) :2649–2661, Nov. 2007.

[81] S. Korman and S. Avidan. Coherency Sensitive Hashing. In *Proceedings of the 2011 International Conference on Computer Vision*, pages 1607–1614, 2011.

[82] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture Optimization for Example-based Synthesis. *ACM Transactions on Graphics (TOG)*, 24(3) :795–802, July 2005.

[83] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures : image and video synthesis using graph cuts. *ACM Transactions on Graphics (TOG)*, 22(3) :277–286, July 2003.

[84] P. Levieux, J. Tompkin, and J. Kautz. Interactive Viewpoint Video Textures. In *Proceedings of the 9th European Conference on Visual Media Production*, pages 11–17, 2012.

[85] A. Levin, A. Zomet, and Y. Weiss. Learning how to inpaint from global image statistics. In *IEEE International Conference on Computer Vision*, volume 1, pages 305–312, Oct. 2003.

[86] R. Li, B. Zeng, and M. L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4) :438–442, Aug. 1994.

[87] R. Lienhart. Reliable Transition Detection in Videos : A Survey and Practitioners Guide. *International Journal of Image and Graphics*, pages 469–486, 2001.

[88] Y. Liu and V. Caselles. Exemplar-Based Image Inpainting Using Multiscale Graph Cuts. *IEEE Transactions on Image Processing*, 22(5) :1699–1711, May 2013.

[89] J. Lu and M. L. Liou. A simple and efficient search algorithm for block-matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(2) :429–433, Apr. 1997.

[90] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. pages 674–679, 1981.

[91] S. Masnou. Disocclusion : a variational approach using level lines. *Image Processing, IEEE Transactions on*, 11(2) :68–76, Feb. 2002.

[92] S. Masnou and J. M. Morel. Level lines based disocclusion. In *International Conference on Image Processing*, volume 3, pages 259–263, Oct. 1998.

[93] Z. Y. Michal, M. Ofry, A. Bruker, and T. Hassner. ImageCompletion Software, http ://www2.mta.ac.il/˜tal/ImageCompletion/. Accessed on 10th September 2013.

[94] R. D. Morris. *Image Sequence Restoration Using Gibbs Distributions*. PhD thesis, 1995.

[95] S. Müller, J. Bühler, S. Weitbruch, C. Thebault, I. Doser, and O. Neisse. Scratch detection supported by coherency analysis of motion vector fields. In *Proceedings of the 16th IEEE international conference on Image processing*, pages 89–92, 2009.

[96] Y. Nie and K.-K. Ma. Adaptive rood pattern search for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 11(12) :1442–1449, Dec. 2002.

[97] J. M. Odobez and P. Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communications and Image Representation*, 1995.

[98] I. Olonetsky and S. Avidan. TreeCANN - k-d tree coherence approximate nearest neighbor algorithm. In *Proceedings of the 12th European conference on Computer Vision*, pages 602–615, Berlin, Heidelberg, 2012. Springer-Verlag.

[99] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting of occluding and occluded objects. In *IEEE International Conference on Image Processing*, volume 2, pages II–69–72, 2005.

[100] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video Inpainting Under Constrained Camera Motion. *IEEE Transactions on Image Processing*, 16(2) :545–553, Feb. 2007.

[101] J. S. Pérez, E. Meinhardt-Llopis, and G. Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, pages 137–150, 2013.

[102] P. Pérez, M. Gangnet, and A. Blake. Poisson Image Editing. *ACM Transactions on Graphics (TOG)*, 22(3) :313–318, July 2003.

[103] L.-M. Po and W.-C. Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3) :313–317, June 1996.

[104] M. Primet. *Probabilistic methods for point tracking and biological image analysis*. PhD thesis, Nov. 2011.

[105] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *IEEE 12th International Conference on Computer Vision*, pages 151–158, Sept. 2009.

[106] F. Raimbault and A. Kokaram. Stereo video inpainting. *IS&T/SPIE Electronic Imaging*, 21(1), 2011.

[107] F. Raimbault, F. Pitie, and A. Kokaram. Stereo video completion for rig and artefact removal. In *13th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4, May 2012.

[108] M. Rousson, T. Brox, and R. Deriche. Active unsupervised texture segmentation on a diffusion based feature space. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–699–704, June 2003.

[109] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video Textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 489–498, 2000.

[110] J. Shen and T. F. Chan. Mathematical Models for Local Nontexture Inpaintings. *SIAM Journal on Applied Mathematics*, 62(3) :1019–1043, Jan. 2002.

[111] Y. Shen, F. Lu, X. Cao, and H. Foroosh. Video Completion for Perspective Camera Under Constrained Motion. In *Proceedings of the 18th International Conference on Pattern Recognition*, pages 63–66, 2006.

[112] T. K. Shih, N. C. Tan, J. C. Tsai, and H.-Y. Zhong. Video falsifying by motion interpolation and inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[113] T. Shiratori, Y. Matsushita, X. Tang, and S. B. Kang. Video Completion by Motion Field Transfer. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 411–418, 2006.

[114] J. Sun, L. Yuan, J. Jia, and H. Y. Shum. Image Completion with Structure Propagation. *ACM Transactions on Graphics (TOG)*, 24(3) :861–868, July 2005.

[115] M. Szummer and R. W. Picard. Temporal texture modeling. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 823–826, Sept. 1996.

[116] M. Tepper, P. Musé, and A. Almansa. Meaningful Clustered Forest : an Automatic and Robust Clustering Algorithm. 2011.

[117] D. Tschumperle and L. Brun. Non-local image smoothing by applying anisotropic diffusion PDE's in the space of patches. In *16th IEEE International Conference on Image Processing*, pages 2957–2960, Nov. 2009.

[118] L. Vese and S. Osher. Modeling Textures with Total Variation Minimization and Oscillating Patterns in Image Processing. *Journal of Scientific Computing*, 19(1-3) :553–572, 2003.

[119] Vitulano. Line Scratch Detection on Digital Images : An Energy Based Model. *Special Issue of Journal of WSG*, 10(2) :477–484, 2002.

[120] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. LSD : A Fast Line Segment Detector with a False Detection Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4) :722–732, Apr. 2010.

[121] Y. Q. Wang. E-PLE : an Algorithm for Image Inpainting. *Image Processing On Line*, pages 261–275, 2013.

[122] E. W. Weisstein. Order statistic, http ://mathworld.wolfram.com/OrderStatistic.html. Accessed on 23 January, 2014.

[123] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 120–127, June 2004.

[124] Y. Wexler, E. Shechtman, and M. Irani. Space-Time Completion of Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3) :463–476, Mar. 2007.

[125] O. Whyte, J. Sivic, and A. Zisserman. Get Out of my Picture ! Internet-based Inpainting. In *BMVC*, pages 1–11, 2009.

[126] G. Yu, G. Sapiro, and S. Mallat. Solving Inverse Problems With Piecewise Linear Estimators : From Gaussian Mixture Models to Structured Sparsity. *IEEE Transactions on Image Processing*, 21(5) :2481–2499, May 2012.

[127] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM conference on Pattern recognition*, pages 214–223, 2007.