*Fares Haddad.* *fhaddad@etu.info.unicaen.fr*
***M**aster **P**rofessionnel **R**éseaux, **A**pplication **D**ocumentaire,*
***I**ngénierie et **S**écurité (RADIS)*
*Mars 2009*


# *Vidéo Inpainting*

**Encadrer par** : Youssef Chahir. Youssef.chahir@info.unicaen.fr

Table de Matière :

# *1. Introduction*

Aujourd'hui le concept d'inpainting attire de plus en plus l'attention des chercheurs notamment dans le domaine du traitement d'image. De manière générique, ce terme désigne le fait de déterminer, de la manière la plus automatique possible, la couleur de pixels considérés comme *manquants* dans une image, c'est-à-dire, dont on ne connait pas les valeurs à priori. Ce type d'algorithme de reconstruction de données est intéressant à plus d'un titre, de par les nombreuses applications concrètes qu'il peut traiter. Un algorithme *d'inpainting* peut être utilisé par exemple pour restaurer de manière numérique des images dégradées par des artefacts ayant détruit de manière complète certaines parties des images (rayures sur des films anciens ou taches sur des photographies). Une application dérivée intéressante consiste en la *suppression cohérente* d'objets réels dans des images, en signifiant à l'algorithme que ces objets sont considérés comme des artefacts que l'on veut corriger. Pour fonctionner, un algorithme d'inpainting nécessite donc, d'une part la donnée de l'image à traiter, et d'autre part, un masque binaire M définissant les zones que l'on cherche à reconstruire.

Plusieurs algorithmes dans la littérature ont été proposés pour la retouche vidéo se basant sur les EDPs : on cite parmi les premiers travaux ceux qui traitent la vidéo à retoucher frame par frame [1] suivie de [2]. Dans [1] les EDPs est appliqué spatialement, et complète la vidéo frame par frame. Cette méthode ne prend pas en considération l'aspect temporel de la vidéo, et son application de ce fait est limitée.

D'autres méthodes combinent entre plusieurs techniques l'inpainting pour aboutir à un résultat acceptable. Dans [3] les auteurs combinent entre deux étapes Une première étape, de reconstruction de la géométrie globale, à l'intérieur des régions de données manquantes : cette étape se base naturellement sur une méthode de type EDP de diffusion. Une deuxième étape, de synthèse des textures à l'intérieur des régions de données manquantes : cette étape se base sur les techniques récentes de synthèse de textures utilisant des correspondances de bloc d'images grâce au résultat de la première étape.

Un travail intéressant pour réparer la vidéo endommagée a été proposé dans [4]. Leur méthode consiste à utiliser une gamme de différentes techniques ce qui rend le processus de l'inpainting complexe. L'utilisateur doit manuellement dessiner les frontières des différentes régions à retoucher de toutes les frames de la vidéo. En outre, c'est un algorithme d'apprentissage du fond statique de la vidéo. Le mouvement des objets à l'arrière-plan est limité pour être périodique, ce qui implique que les objets également ne changent pas de taille pendant qu'ils se déplacent, ainsi le mouvement est approximativement parallèle au plan de

projection de l'appareil photo. Des objets mobiles endommagés sont reconstruits en synthétisant un nouvel objet intact, en le recouvrant dans la séquence, et en le déplaçant le long d'une nouvelle trajectoire interpolée. Cette approche produit un flou très apparent quand les objets se déplacent d'une manière peu réaliste.

Dans [5] les auteurs proposent de remplir les régions détériorées en utilisant un algorithme de synthèse de texture. Mais le succès de cette méthode dépend de l'ordre du choix des pixels se trouvant sur le bord de la région détériorée. L'algorithme consiste à associer à chaque pixel du bord de la région détériorée un patch centré par ce pixel. Calculer sa priorité avec deux cœfficients :

Terme de confidence : correspond au nombre de pixels non détérioré ce trouvant dans le patch ou bien des pixels précédemment corrigés.

Terme de donnée : décrit si il y a une structure linaire forte dans se patch. Si c'est le cas on lui attribue une priorité élevée. Remplir ce patch en priorité aide à maintenir la structure linéaire de l'image.

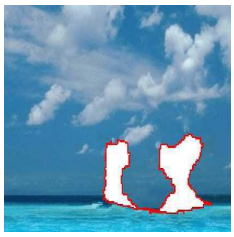
La priorité est donc égale au produit du terme de confidence par le terme de donnée. Il y a un équilibre sensible entre la confiance et le terme de données. Le terme donné tend à pousser des isophotes rapidement vers l'intérieur, alors que le terme de confiance tend à supprimer avec précision cette sorte d'irruption dans la région à traiter.

Dans la première partie de ce travail je vais vous présenter la reconstruction globale des régions manquantes par EDP de diffusion en utilisant l'équation de poisson ; ainsi que les différents domaines d'applications, ensuite je vais vous présenter un algorithme de synthèse de texture afin de remédier aux problèmes des EDPs .

## *2. Pourquoi l'équation de poisson*

L'équation de Poisson est une équation différentielle partielle utilisée dans différents domaines tels que la physique, la chimie…l'imagerie où elle est utilisée pour diverses applications telles que la représentation et classification d'une action, la segmentation… Dans notre travail on l'a utilisé pour trois objectifs différents :

- La retouche sur image et vidéo




- L'élimination d'un objet dans une scène


- L'incrustation d'un objet dans une scène


# 3. Généralités et définitions

## 3.1. Retouches (Inpainting)

L'inpainting (également connu sous le nom d'interpolation d'image ou interpolation de vidéo) est un terme qui désigne la reconstruction de données manquantes dans une image/vidéo contenant de la texture ou des structures en appliquant des algorithmes sophistiqués pour récupérer les parties détériorées ou corrompues des données d'image, qui a pour but de retrouver la couleur des pixels manquants.

## 3.2. Qu'est-ce qu'une texture ?

On entend principalement parler de texture quand on parle d'imagerie 3D. En effet, pour produire des images fixes ou animées, en modélisant des objets en 3D sous la forme de maillages, on plaque sur ceux-ci des images 2D pour leur donner une apparence réaliste ou non. Ce sont ces images 2D que l'on appelle textures. Les jeux vidéo, mais aussi les films d'animation 3D utilisent ainsi massivement des textures. Dans le domaine du traitement d'images, le terme "texture" désigne toute image numérique composée d'éléments répétés. Les textures peuvent être classées selon un large spectre allant de stochastiques à régulières. Les textures stochastiques étant des images dont la couleur de chaque pixel semble avoir été déterminée au hasard. De telles images ressemblent donc a un bruit numérique. Observées à partir d'une certaine distance, beaucoup de textures semblent stochastiques. A l'opposé, les textures régulières sont structurées et constituées d'éléments répétés quasiment à l'identique et selon un certain schéma (exemple : images de carrelage, de parquet, d'un mur de briques ...).

## 3.3. Synthèse de textures

La synthèse de texture est une technique qui permet, à partir d'une image source contenant une texture, de produire une nouvelle image, en général plus grande, ayant les mêmes caractéristiques visuelles. Cependant, il est attendu que cette nouvelle image ne soit pas juste une simple répétition de l'image source afin d'obtenir un bon aspect visuel. C'est une technique utilisée aussi pour reconstruire des données manquantes dans des images. En effet, beaucoup de personnes ont des vieilles photos de famille, de cartes postales de collection, des

films mais, avec les années, celles-ci se sont souvent détériorées, et des rayures, des pliures, des trous ont pu apparaître. L'inpainting permet de restaurer de manière numérique ces images/vidéos une fois numérisées, en réalisant la reconstruction des données manquantes, et ainsi de redonner un coup de jeune à ces précieux souvenirs.

Les algorithmes d'inpainting ne se limitent heureusement pas aux images/vidéos en niveaux de gris, tout type de photographie, film, diapositive pourra donc se voir restauré à l'aide de ceux-ci.

En détournant l'application précédente, on peut obtenir une nouvelle application tout aussi utile et impressionnante. En effet, on voit que si l'on signifie à l'algorithme que des parties de l'image sont des artéfacts à corriger, celui-ci permet alors également la suppression d'objets de notre choix dans cette image. Ainsi, un algorithme d'inpainting est utile si l'on souhaite effacer des parties de l'image. Des personnes, des grillages, des lignes électriques disgracieuses, ... et même du texte incrusté sur l'image/vidéos sont des exemples concrets d'objets qu'un tel algorithme pourra faire disparaître.

### *3.3.1. Principes de fonctionnement*

Différentes méthodes de synthèse de texture existent et ont fait et font toujours l'objet de recherches

#### *3.3.1.1. Tiling : "copier-coller"*

La façon la plus simple de créer une image plus grande à partir d'une image source est le tiling. C'est-à-dire étendre l'image source en la dupliquant par un simple copier-coller. Le résultat n'est pratiquement jamais satisfaisant, sauf si l'image source répond à certaines spécificités, car les jointures entre chaque réplique de l'image source dans la nouvelle image seront pleinement visibles et cette nouvelle image sera de plus extrêmement répétitive.

#### *3.3.1.2. Synthèse stochastique*

Un algorithme de ce type produit une image en choisissant au hasard des couleurs pour chaque pixel, seulement guidé par quelques paramètres tels que la luminosité minimum, la couleur moyenne ou un maximum pour le contraste. Ces algorithmes ne produisent de bons résultats qu'avec des textures stochastiques d'origine, dans les autres cas, les résultats produits sont complètement insatisfaisants du fait qu'ils ne tiennent en aucun compte de la structure éventuelle de l'image source.

### *3.3.1.3. Algorithmes spécifiques*

De tels algorithmes utilisent une fonction spécifique pour créer une nouvelle image, c'est-à-dire qu'ils sont limités à un seul type de texture structurée. Ainsi, non seulement ces algorithmes peuvent uniquement être appliqués à des textures structurées, mais ils ne peuvent traiter que des textures ayant des structures très similaires. Par exemple des murs de briques, ou de l'herbe, mais ils ne seront certainement pas capables de produire un bon résultat si la texture sur laquelle on les applique ne correspond pas à la structure pour laquelle ils sont faits.

### *3.3.1.4. Synthèse pixel par pixel*

Ce type d'algorithme réalise la synthèse d'une nouvelle texture à partir d'une image source, pixel par pixel. Pour cela, ils comparent le voisinage du pixel en cours avec les voisinages des pixels de l'image source afin de déterminer quel pixel sera le meilleur candidat à la recopie.

### *3.3.1.5 Synthèse blocs par blocs*

Ce type d'algorithme réalise la synthèse d'une nouvelle texture à partir d'une image source, bloc par bloc. Ils recopient donc d'un seul coup des structures entières de la texture source, ce qui tend à les rendre plus efficaces et plus rapides que les algorithmes de synthèse pixel par pixel.

# *4. Inpainting et EDP de diffusion*

## *4.1 Mécanisme de poisson inpaiting*

Le but de l'opération est de compléter le domaine détérioré Ω de telle sorte que l'image/vidéo résultante soit la plus pertinente possible ; en utilisant le masque qui indique les pixels à restaurer.

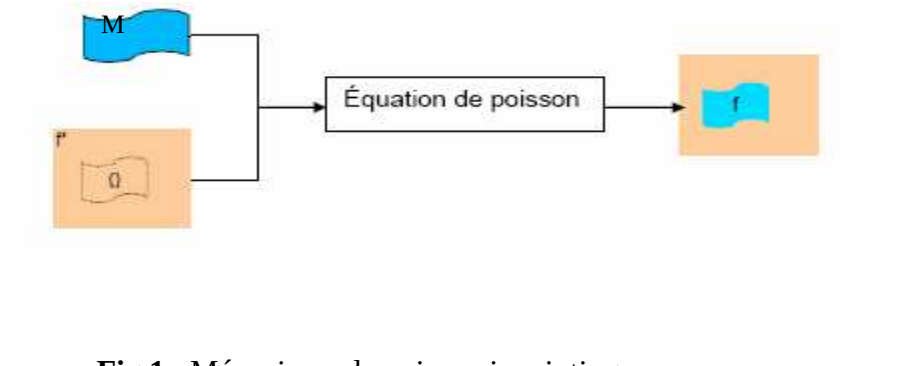



**Fig 1** : Mécanisme de poisson inpainting

Le problème revient donc à trouver f sur Ω donc à résoudre l'équation :

$$\min_f \iint_\Omega \left| \nabla f \right|^2 \; avec \; f_{d\Omega} = f^*_{d\Omega}$$

Ce problème a pour solution la solution de l'équation de Poisson avec les conditions limites de Dirichlet. Cette condition impose que la solution $f$ doit avoir les mêmes valeurs sur son contours que les valeurs du contour sur Ω . Le mécanisme va donc effectuer une diffusion de contours d Ω sur Ω

Reconstitution après 5 itérations

**Fig 2** : exemple d'application de l'équation de poisson

## 4.2. Résolution de l'équation de Poisson

### *4.2.1. Discrétisation de l'équation de la chaleur*

Il est assez naturel de penser à discrétiser Poisson pour trouver sa solution et donc celle du problème de minimisation. L'équation de la chaleur est très proche de l'équation de Poisson du point de vue de l'implémentation.

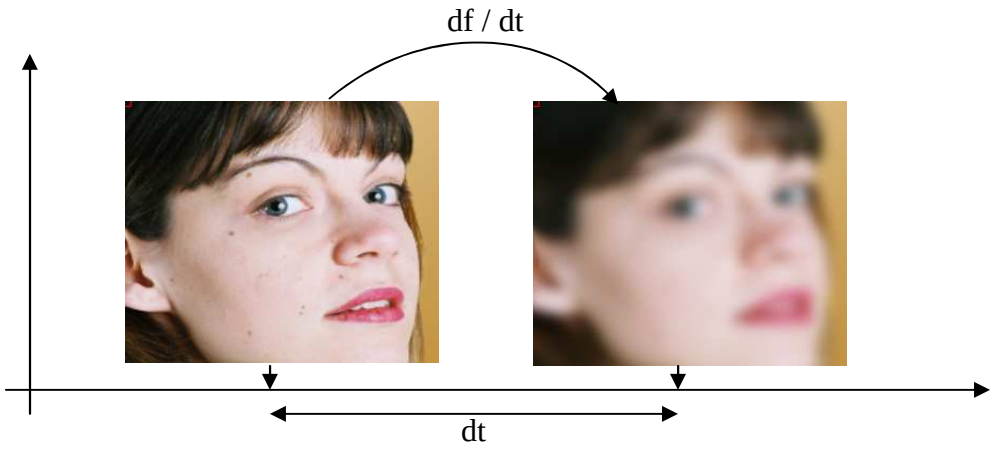
$$\frac{df}{dt} = \Delta f$$

Tout comme Poisson, elle modélise parfaitement les phénomènes de diffusion. Son avantage est la présence d'un terme d'évolution temporelle qui permet d'atteindre la solution $f$ par itérations successives. Enfin la résolution de l'équation de la chaleur est assez intuitive.

### *4.2.2. De la simple diffusion au Poisson inpainting*

#### *4.2.2.1. La diffusion*

Une solution générale de l'équation de la chaleur est celle rencontrée dans le cas de la diffusion, c'est-à-dire une gaussienne qui s'étale au cours du temps. La diffusion intervient pour disperser l'information d'une image au delà de ses contours ce qui se traduit par l'apparition de flou.




L'illustration suivante montre au cours du temps l'effet de la diffusion sur une image.

### *4.2.3. Implémentation de la résolution de l'équation de la Chaleur*

Les principales variables d'entrée du processus sont : l'image source *f* , le masque *m* qui spécifie la partie de *f* que l'on souhaite restaurer.

1) Extraction de la partie de *f à* compléter à l'aide du masque *m* binaire réalisé avec une image en noire et blanc.

2) parcours du masque m et pour chaque pixel noir qui représente le pixel détérioré dans l'image source f .

3) Diffusion de $f$ dans $\Omega$ jusqu'à ce que il n'y ait plus de pixels noirs.

### *4.2.4. Discrétisation de l'équation de la chaleur*

L'implémentation nécessite le passage de l'équation de la chaleur du domaine continu au domaine discret.

$$\frac{df}{dt} = \Delta f$$

$$\frac{\left(f^{n+1} - f^{n}\right)}{dt} = \left(\Delta f^{n}\right)$$

$$\left(f^{n+1} - f^{n}\right) = \left(\Delta f^{n}\right)dt$$

$$f^{n+1} = f^{n} + \left(\Delta f^{n}\right)dt$$

Le terme $\left(\Delta f^{n}\right)$ est à recalculer à chaque itération. Voici comment celui-ci est défini en Discret en partant de son modèle continu :

$$\Delta f = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2}$$

$$or \ \frac{d^2 f}{dx^2} = \frac{d}{dx}\frac{d\ f}{dx} \ \ et \ \ \frac{d\ f}{dx} = f_{x+1,y} - f_{x,y}$$

$$d'ou \quad \frac{d^2 f}{dx^2} = f_{x+1,y} - 2 f_{x,y} + f_{y-1,y}$$

$$soit\ au\ \ final \quad \Delta f = f_{x+1,y} + f_{x-1,y} + f_{y+1,x} + f_{y-1,x} - 4 f_{x,y}$$

# 5. Expérimentation et Résultat

## 5.1. Outils utilisés

Pour réaliser ce projet, j'ai utilisé la bibliothèque Pandore. Pandore est une bibliothèque d'opérateurs de traitement d'images. Elle est développée au sein de l'équipe Image du laboratoire GREYC. La version actuelle (6.3.3, datant de janvier 2009), regroupe des opérateurs capables de traiter des images 1D, 2D et 3D et des cartes de région ; en niveaux de gris, en couleurs et multispectrales.

Cette bibliothèque se compose d'une collection d'opérateurs exécutables ainsi que d'un environnement de programmation en C++. Ainsi, cette bibliothèque se conçoit comme une collection de programmes exécutables travaillant directement sur des fichiers images. La construction d'une application de traitement d'images, se fait par l'activation successive d'opérateurs, les images de sortie des uns servant d'images d'entrée aux autres. Ces images étant dans le format spécifique à Pandore (.pan) mais des opérateurs de conversion de formats existent.

C'est l'environnement de programmation en C++ et ses possibilités de programmer de nouveaux opérateurs qui m'ont servi dans mon projet.

## 5.2. Parcours du masque

Deux manières possibles pour le parcours du trou afin de diffuser

### 5.2.1. Parcours linière

Pour chaque pixel détérioré faire une diffusion et passer au pixel se trouvant sur la même ligne ou colonne suivant le parcours de l'image.

I

**Fig 3 :** masque après étiquetage

### *5.2.3. Parcours par couche*

Le parcours se fait selon l'algorithme suivant

Parcours de l'image masque

Calcul gradient

Tanque (gradient !=0) alors

Ajout de ce pixel à la liste

Mise à jour du pixel

Fin tanque

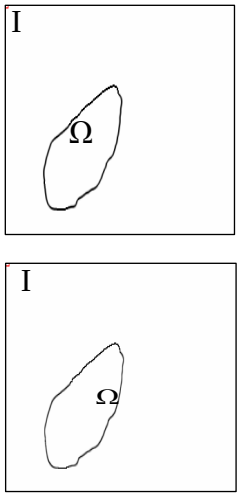
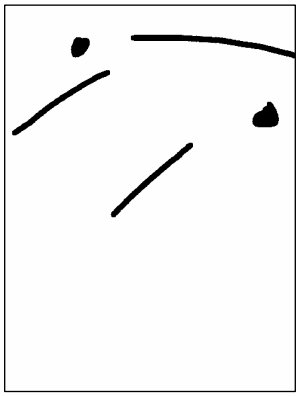



**Fig 4** : parcours de l'image par couche

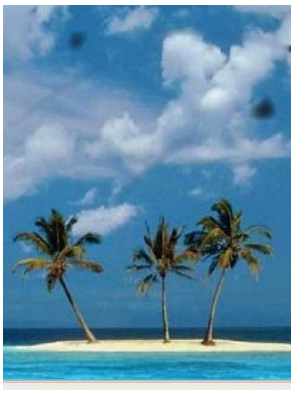
## *5.3. Application à la retouche d'image*

Nous avons appliqué notre algorithme à la retouche d'image avec différentes valeurs des paramètres dt défini précédemment dans l'équation de poisson et le nombre d'itération (it) avec l'utilisation d'un masque binaire pour déterminer la région à retoucher

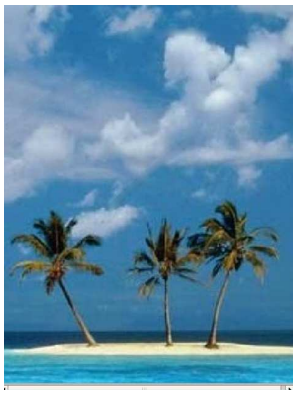
Image détériorée



masque



dt = 0.5,  it = 100



dt = 0.5,  it = 200

**Fig 5 :** reconstruction d'une image par poisson inpainting

## *5.4. Application aux images texturées*

Dans cette section nous allons appliquer l'algorithme de poisson inpainting sur une image texturée avec comme paramètre d'entrée l'image, le masque, dt et it

Image texturée détériorée

Masque

dt = 0.5  et  it = 2000

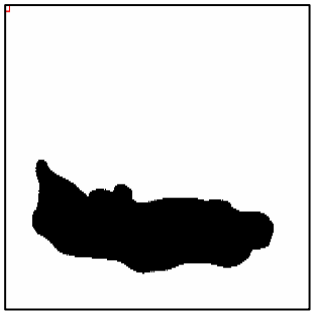
**Fig 13**: application de poisson inpaiting à une image texturé

## *5.4. Application à l'élimination d'objet dans une scène*

Le principe d'élimination d'objet dans une scène reste le même que celui utilisé dans poisson inpainting ; donc l'algorithme est le même sauf que le masque utilisé représente les objets que l'on veut supprimer de la scène

Image originale

objet à supprimer



Masque

objet supprimer

**Fig 6 :** suppression d'objet dans une image par poisson inpainting

# *6. Incrustation d'objet dans une scène*

## *6.1. Mécanisme d'incrustation d'objet dans une scène*

Le but de l'opération est d'incruster une image source *g* dans une image destination $f^*$ sur le domaine Ω de telle sorte que l'image résultante *f* soit la plus pertinente possible.

Image à incruster

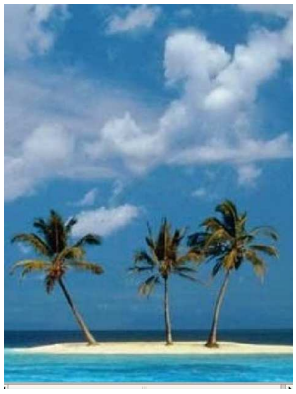


Image de destination

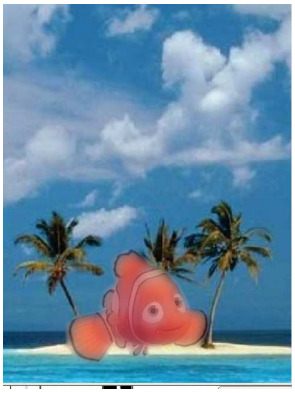


**Fig 7 :** Exemple d'incrustation d'objet dans une scène

## *6.2. Mise en équation du problème*

Le processus a pour but de trouver la fonction optimale sur Ω telle que la différence entre le gradient de $f$, $\nabla f$ et le guide $v$ soit minimale. Ce processus d'interpolation aboutit donc au problème de minimisation suivant :

$$\min_f \iint_\Omega \left| \nabla f - v \right|^2 \; avec \; f_{d\Omega} = f^*_{d\Omega}$$

avec $\Delta f = div\ v\ sur\ \Omega,$ avec $f_{d\Omega} = f^*_{d\Omega}$

dans notre cas $v = \nabla g$, donc le problème revient à minimiser

$$\min_f \iint_\Omega \left| \nabla f - \nabla g \right|^2 \; avec \; f_{d\Omega} = f^*_{d\Omega}$$

C'est-à-dire qu'on cherche à trouver $f$ sur Ω tels que les gradients de source et de destination soient identiques. Ce qui revient à une incrustation optimale.

$$\Delta f = \Delta g \; sur \; \Omega, \text{ avec } f_{d\Omega} = f^*_{d\Omega}$$

## *Remarque*

La résolution de cette équation est la même que celle de poisson inpainting ; on a juste ajouté un terme qui représente le laplacien de l'image à incruster ; donc la discrétisation de l'équation de la chaleur serra de la forme suivante :

$$\frac{df}{dt} = \Delta f - \Delta g$$
$$f^{n+1} = f^{n} + (\Delta f^{n} - \Delta g)dt$$

## *6.3. Algorithme*

Les principales variables d'entrée du processus sont : l'image source g , l'image $f^*$ de destination et le masque m qui spécifie la partie de g que l'on souhaite incruster.

1) Extraction de la partie de g à insérer à l'aide du masque m binaire réalisé avec une image en noir et blanc.

2) Calcul du gradient de g puis de sa divergence pour former Δg .

3) Insertion de l'image g dans l'image de destination $f^*$ .Cette étape constitue un simple copier/coller

4) Diffusion de $f^*$ dans Ω jusqu'aux contours de l'image g par évolution de l'équation de la chaleur.

5) L'image *f* finale satisfait $\Delta f = \Delta g$.

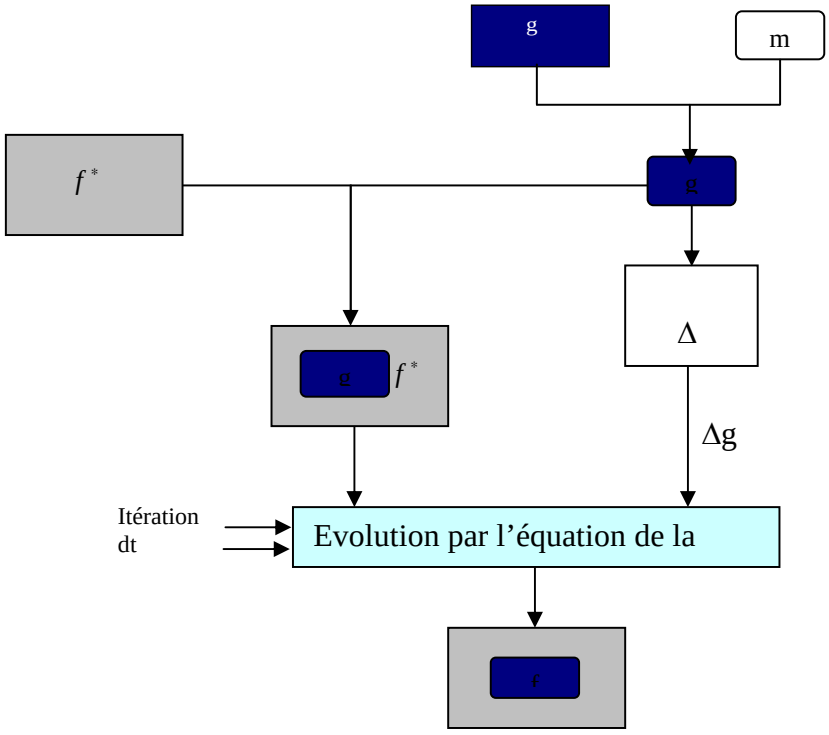



**Fig 8** : Mécanisme d'incrustation d'objet dans une scène

## *6.4. Expérimentation et Résultat*

L'algorithme a été appliqué avec deux paramètres d'entrées : dt et le nombre d'itération it avec les coordonnées où on veut incruster l'objet


Image à incruster

masque

dt = 0.5 et it = 300

**Fig 9**: incrustation d'un ballon dans une scène

## *7. Extensions et application à la vidéo*

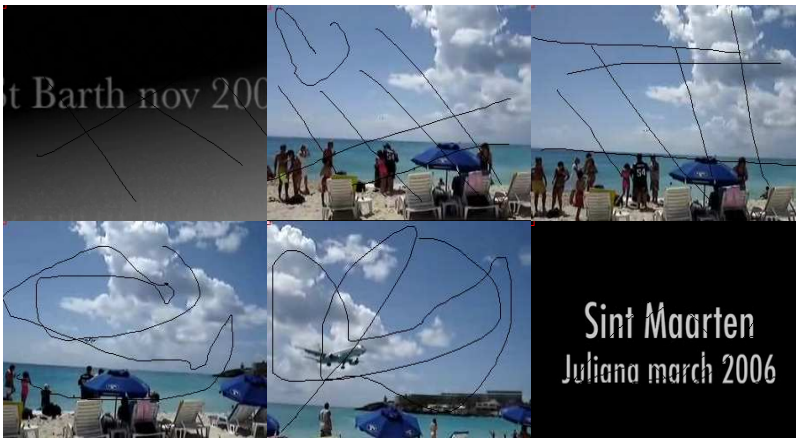
## *7.1. Application à l'inpainting sur vidéo*

Une vidéo est une succession d'images ; donc c'est la représentation spatio-temporelle d'un ensemble de frames. Deux méthodes peuvent être utilisées pour la retouche sur vidéo :

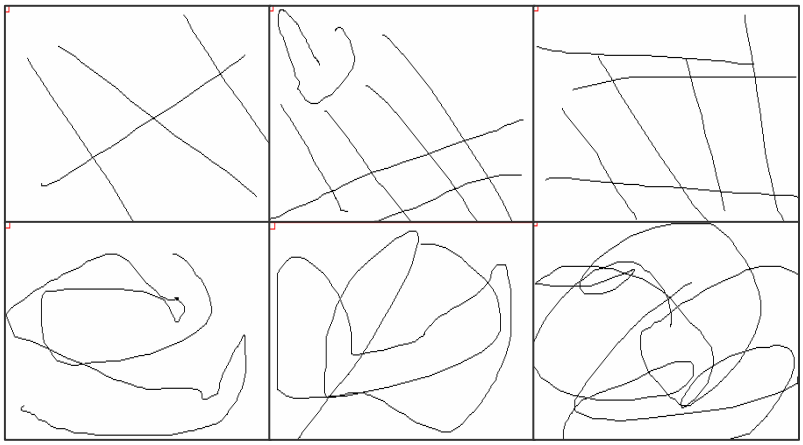
- Remplissage des régions détériorées frames par frames
- ou bien traiter la vidéo comme un seul bloc en 3D

Les deux méthodes on été testées, en exécutant un scripte qui prend :

- en entrée une vidéo
- tracer les régions détériorées afin d'avoir le masque
- application des algorithmes cités ci-dessus
- en sortie on aura une vidéo la plus nette possible
- Exécution de l'algorithme de poisson inpainting sur une vidéo détériorée par des rayures. Avec comme paramètres la vidéo, le masque, dt , it




Quelques images d'une vidéo détériorée par des rayures

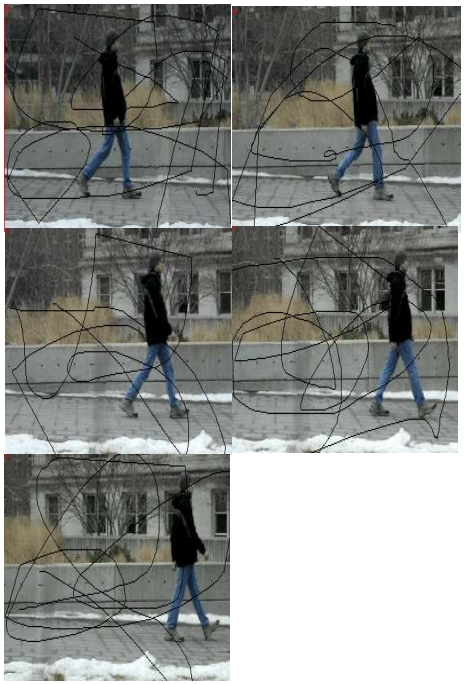


Masque correspondant

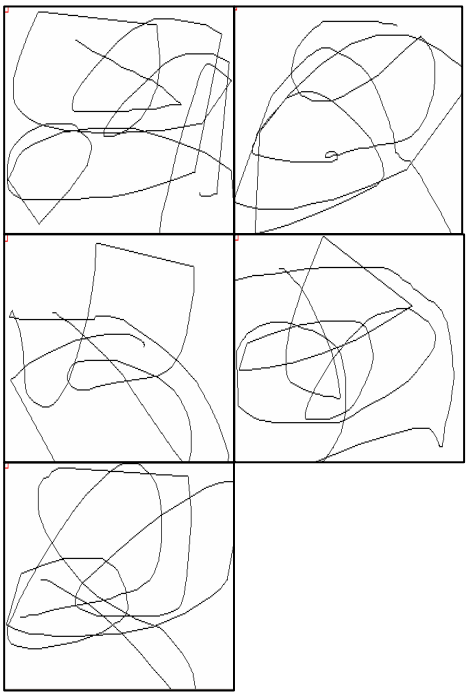

dt = 0.5 et it = 10

**Fig 10** : exemple de quelques frames extraites d'une vidéo détériorée par des rayures et traitée par l'algorithme de poisson inpainting

Application de l'algorithme précèdent sur une autre vidéo frames par frames. Avec comme paramètres la vidéo, le masque, dt , it



Vidéo détériorée



Masque correspondant

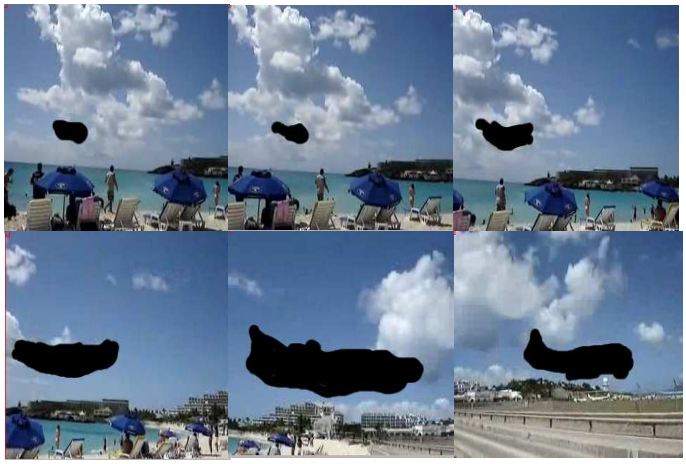
dt = 0.5 et it = 10

**Fig 11**: exemple de quelque frames extraites d'une vidéo détériorée par des rayures et traitée par l'algorithme de poisson inpainting frames par frames

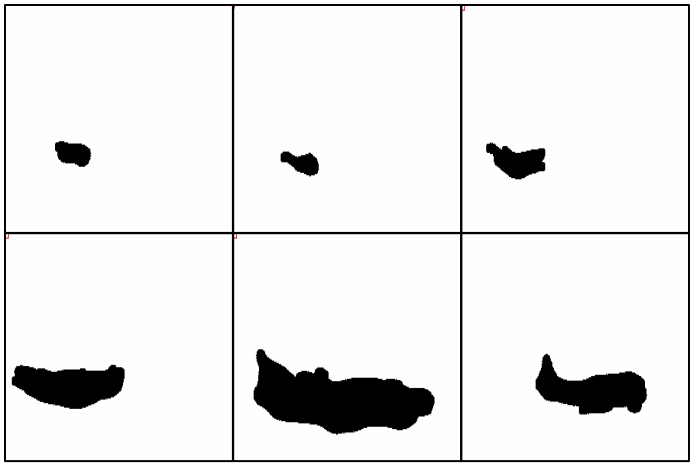
## *7.2. Application à l'élimination d'un objet dans une vidéo*

Nous allons exécuter le même algorithme afin de supprimer un objet d'une scène, dans notre cas un avion. Avec comme paramètre d'entrée la vidéo, le masque de l'objet à supprimer, dt et it

Quelques frames d'une vidéo



Objet à supprimer



Masque

dt = 0.5  et it = 2000

**Fig 12** : suppression d'un  avion  de la vidéo par l'algorithme d'élimination

## *Remarques:*

Comme vu précédemment dans la section 5.4 l'application de EDP (équation de poisson) à l'inpainting sur image/vidéo donne de bons résultats en général, sauf dans le cas des image/vidéos très texturées ou bien lorsque la taille de la partie détériorée dans la vidéo est importante.

On remarque lors de l'application des EDPs l'apparition d'un flou sur la région à traiter ; pour remédier à ce problème je vous présente un algorithme de synthèse de texture pour la retouche sur image

## *8. Algorithme de synthèse de texture*

Des algorithmes de synthèse de textures, basées sur des mises en correspondance et des recopies de blocs images sont proposés. Les grandes textures sont alors bien recomposées.

Notre algorithme se résume comme suit :

Parcours de l'image masque

ims image détériorée

Tanque gradient du masque !=0

    Mettre les pixels de la frontière dans une pile tmp

Pour chaque pixel de la pile tmp(p) faire

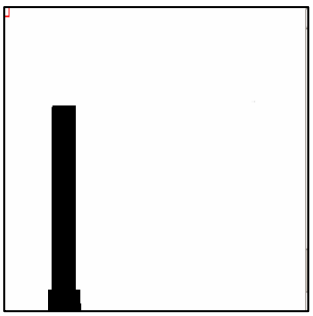
    Prendre un patch $\Psi_p$ carré centré en ce pixel p

    Trouver un autre patch $\Psi_q$ de ims qui minimise $d(\Psi_p, \Psi_q)$

    Copier $\Psi_q$ dans $\Psi_p$

Fin Tanque

Image texturée



Masque

**Fig 14**: application d’un algorithme de synthèse d’image sur une image texturée

## *9. Analyse*

On remarque lors de l'exécution des algorithmes que le temps d'exécution dépend de la taille de la région à traiter :
Lorsque l'image/vidéo est détériorée par des rayures le temps d'exécution du programme est rapide et chaque fois que la région à traiter est plus grande, le temps d'exécution est plus long. La rapidité d'exécution dépend aussi des paramètres d'entrée dt et it (nombre d'itération). Le nombre d'itération dépend de la région à traiter ; chaque fois que la région à traiter est plus grande, cela demande un grand nombre d'itération.

En ce qui concerne l'efficacité, les algorithmes proposés sont en général efficaces que ce soit pour la retouche, l'incrustation ou l'élimination d'objet sauf dans le cas des images/vidéo texturées et la région à traiter est importante, on remarque l'apparition d'un flou. On a essayé de proposer un algorithme de synthèse de texturé mais on n'a pas eu le temps de le perfectionner pour obtenir des résultats acceptables.

## *Conclusion et perspective*

On a commencé à utiliser l'équation de poisson dans le but de la retouche sur vidéo, où on a découvert qu'elle sert à d'autres objectifs tels que l'incrustation d'objet dans une scène, la suppression d'objets dans une image/vidéo ; en utilisant un masque binaire qui représente la partie de la région à retoucher, incruster ou supprimer. Les résultats obtenus pour ses différentes applications sont acceptables.

Mais malheureusement il reste le problème de flou qui apparaît lors de l'application de poisson inpainting sur des images/vidéos texturé donc sur se point l'objectif n'a pas été atteint mais il existe dans la littérature différentes techniques pour remédier a ce problème pour les étudiants qui voudraient travailler sur le sujet.

## *Bibliographie*

[1] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, “Navier-stokes, fluid dynamics, and image and video inpainting,” Proc. IEEE
Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 355–362, 2001.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” Computer Graphics (SIGGRAPH 2000), pp. 417–424, 2000.

[3] V. Do, G. Lebrun, L. Malapert, C. Smet, D. Tschumperlé “Inpainting d’Images Couleurs par Lissage Anisotrope et Synthèse de Textures. Color Image Inpainting by Anisotropic Smoothing and Texture Synthesis” Equipe Image / Laboratoire GREYC (UMR CNRS 6072), 6 Bd du Mar´echal Juin, 14050 Caen CEDEX.

[4] J. Jia, T. Wu, Y. Tai, and C. Tang, “Video repairing under variable illumination using cyclic motions,” Proceedings. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 364–371, 2004.

[5] A.Criminisi, P. Pérez, K. Toyama. Object Removal by Exemplar-Based Inpainting. In Conf. Computer Vision and Pattern Recog, CVPR'03, Volume 2, Pages 721-728, Madison, WI, June 2003.