Video inpainting

MVA 2023-2024

Yann Gousseau
Telecom Paris

Video inpainting

Inpainted video

Video inpainting

Original video

Introduction

What are the challenges of video inpainting?

- Temporal coherency (possibly over a large number of frames)
- Reconstruction of moving objects / movement should be realistic
- Simultaneous foreground/background reconstruction
- Inpainting with moving background
- Dynamic textures / temporal clutter
- Extremely long computational times

Inpainting example (from Wexler *et al.* 2007)

Some approaches to video inpainting

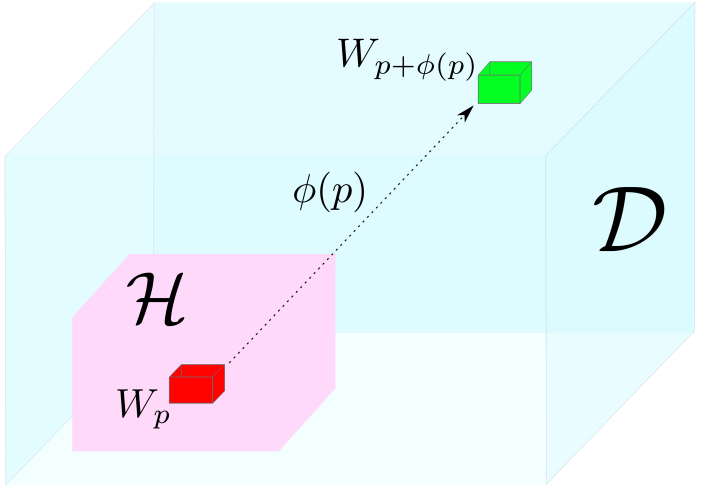
- Greedy patch-based
  Patwardan et al. 2005, 2007, Daisy et al. 2016
- Specific motions (cyclic, etc.)
  Shen et al. 2006, Shiratori et al. 2006, Raimbault et al. 2012
- Shift maps + optimization
  Granados et al. 2012, Ebdelli et al. 2015
- Global patch optimization
  Wexler et al. 2004, 2007, Newson et al. 2014, Le et al. 2017, 2019
- Optical flow driven
  Huang et al. 2016, Bokov et al. 2018
- Deep neural networks
  Oh et al. 2019, Xu et al. 2019, Li et al. 2020, Gao et al. 2020, etc.

From now on : we focus on patch-based methods relying on a global optimization
(Wexler et al. 2004, Newson et al. 2014, Le et al. 2019)

Video inpainting algorithm

Notations

- Target patch
- Nearest neighbour patch

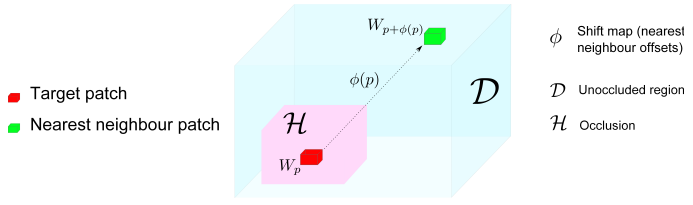


- ϕ Shift map (nearest neighbour offsets)
- $\mathcal{D}$ Unoccluded region
- $\mathcal{H}$ Occlusion

W_p: a patch centered at p

Video inpainting algorithm

Video inpainting notation



W_p: a patch centered at p

Inpainting Principle

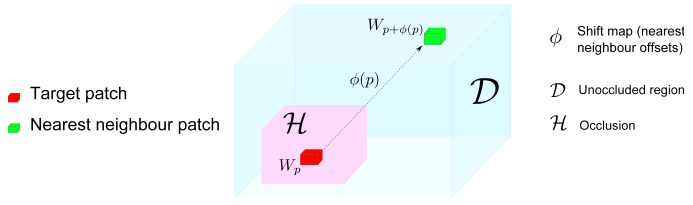
Input: $u|_{\mathcal{D}}$ Output: $u|_{\mathcal{H}}$

Find $u|_{\mathcal{H}}$ by minimizing

$$E(u,\phi) = \sum_{p\in\mathcal{H}} ||W_p^u - W_{p+\phi(p)}^u||_2^2$$

Video inpainting algorithm

Video inpainting notation



W_p: a patch centered at p

Inpainting Principle

Input: $u|_{\mathcal{D}}$ Output: $u|_{\mathcal{H}}$

Find $u|_{\mathcal{H}}$ by minimizing

$$E(u,\phi) = \sum_{p\in\mathcal{H}} ||W_p^u - W_{p+\phi(p)}^u||_2^2$$

Denoising Principle

Input: noisy $\tilde{u}$ Output: denoised $\hat{u}$

Find u by minimizing

$$E(u,w) = \sum_{p,q} w(p,q)||W_p^{\tilde{u}} - W_q^u||_2^2 - h\sum_p H(w(p,\cdot))$$

‡ *P. Arias, G. Facciolo, V. Caselles, G. Sapiro,* **A Variational Framework for Exemplar-Based Image Inpainting**, *IJCV 2011*

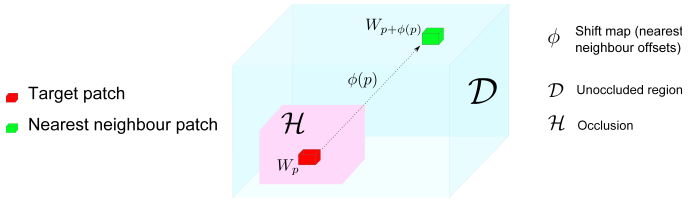
Video inpainting algorithm

Inpainting Principle

Input: $u|_{\mathcal{D}}$ Output: $u|_{\mathcal{H}}$

Find $u|_{\mathcal{H}}$ by minimizing

$$E(u,\phi) = \sum_{p\in\mathcal{H}} ||W_p^u - W_{p+\phi(p)}^u||_2^2$$



W_p: a patch centered at p

Challenges

- non-convex energy
- high dimensionality (dimension $= 5 \times 5 \times 5 \times 3 \approx 500$)

Solutions

- alternate minimizations w.r.t. u and ϕ
- coarse-to-fine processing
- approximate nearest neighbours

Video inpainting algorithm

Core of the approach

Principle:	Algorithm (inspired by Wexler *et al.*)
Minimise global patch-based functional.	Alternate minimizations on u and ϕ:
$$E(u,\phi) = \sum_{p \in \mathcal{H}} \|\|W_p^u - W_{p+\phi(p)}^u\|\|_2^2$$	$u^0 \leftarrow$ Initialisation($u\|_{\mathcal{D}}, \mathcal{H}$) 1/ $\phi^{k+1} \leftarrow$ NearestNeighbourSearch(u^k) 2/ $u^{k+1} \leftarrow$ VideoReconstruction(ϕ^{k+1}) (aggregation of patches)
Analogous to *non-local regularisation*	Carried out in a multi-resolution scheme

Y. Wexler, E. Schechtman, M. Irani, **Space-Time Completion of Video**, *PAMI 2007*
A. Newson, A. Almansa, M. Fradet, Y. Gousseau and P. Pérez, **Video Inpainting of Complex Scenes**, *SIAM IS 2014*

Approximate Nearest Neighbour (ANN) search

Approximate Nearest Neighbour (ANN) search

High dimensionality of problem means NN search is *very* slow

Approximate Nearest Neighbour (ANN) search

High dimensionality of problem means NN search is *very* slow

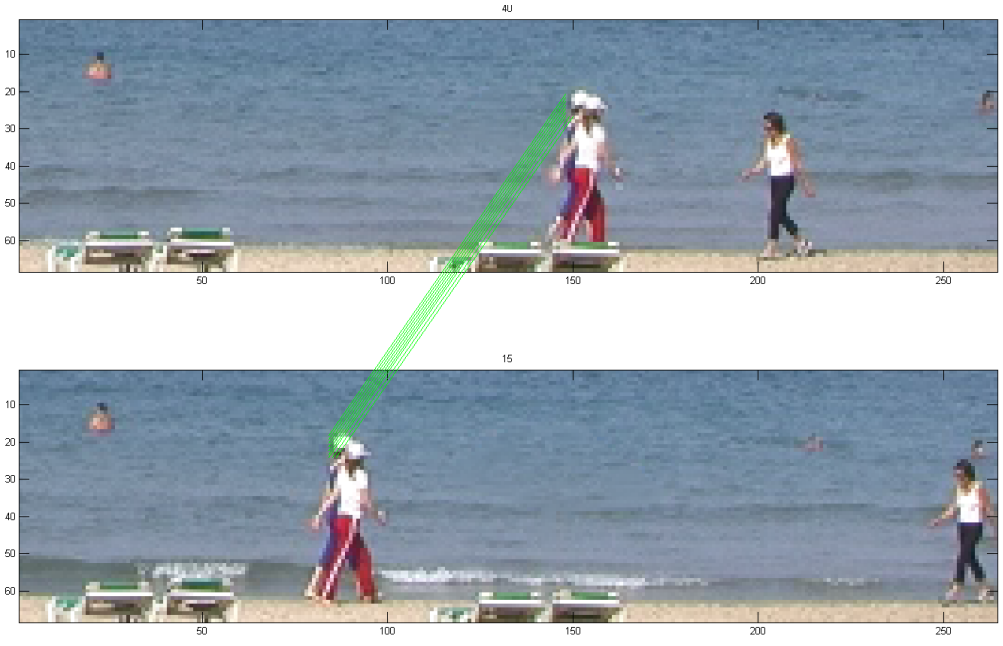
- Previously used ANN search algorithm (kdTrees) very slow
- Extend the PatchMatch (Barnes *et al.* 2009[†]) algorithm to spatio-temporal case.
- PatchMatch: ANN search algorithm for image patches

[†] *C. Barnes, E. Schechtman, A. Finkelstein, D. B. Goldman,* **PatchMatch: a randomized correspondence algorithm for structural image editing**, *ACM Transactions on Graphics (2009)*

Approximate Nearest Neighbour (ANN) search

High dimensionality of problem means NN search is *very* slow

- Previously used ANN search algorithm (kdTrees) very slow
- Extend the PatchMatch (Barnes *et al.* 2009[†]) algorithm to spatio-temporal case.
- PatchMatch: ANN search algorithm for image patches



[†] *C. Barnes, E. Schechtman, A. Finkelstein, D. B. Goldman,* **PatchMatch: a randomized correspondence algorithm for structural image editing**, *ACM Transactions on Graphics (2009)*

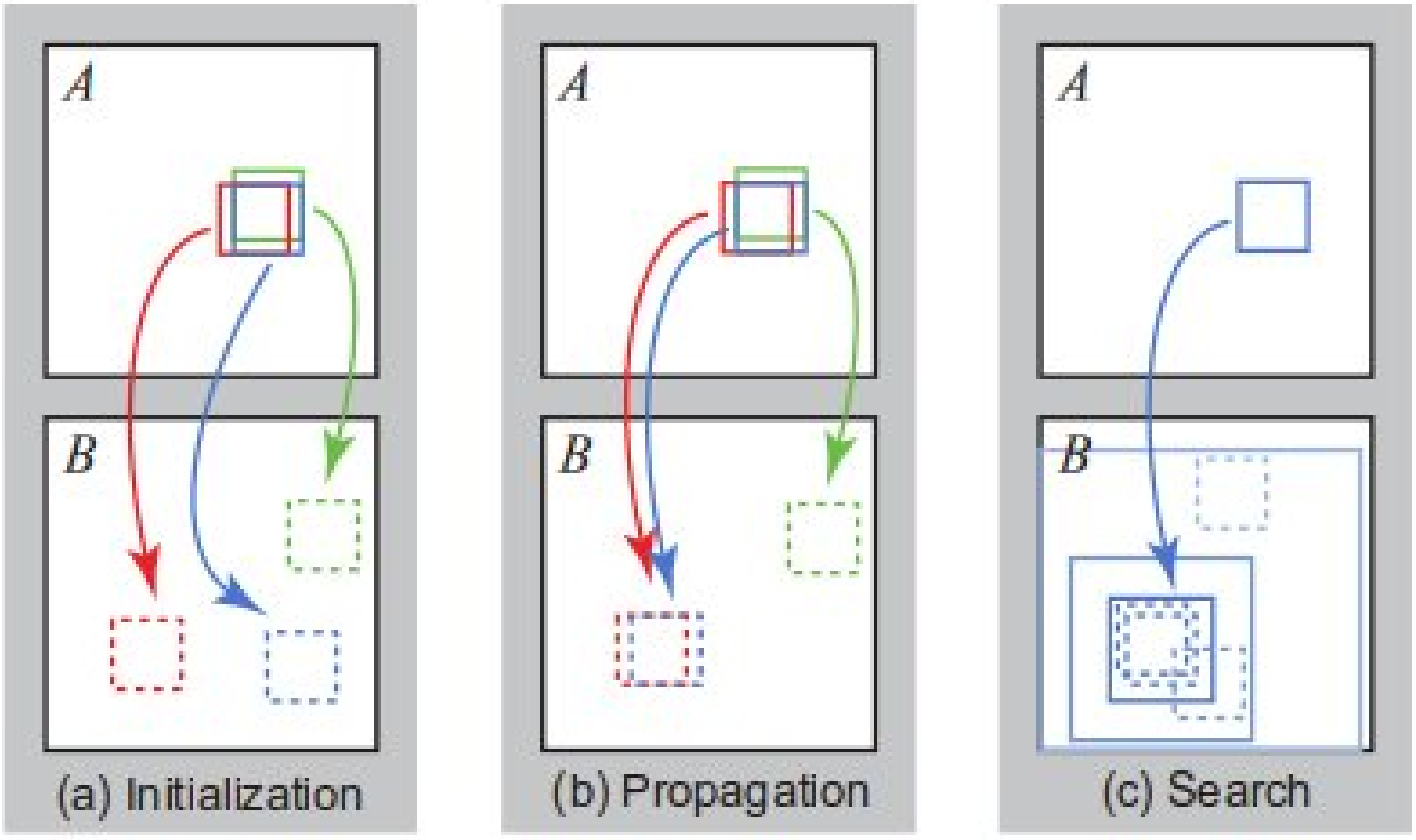


Figure 2: *Phases of the randomized nearest neighbor algorithm: (a) patches initially have random assignments; (b) the blue patch checks above/green and left/red neighbors to see if they will improve the blue mapping, propagating good matches; (c) the patch searches randomly for improvements in concentric neighborhoods.*

From Barnes et al. 2009

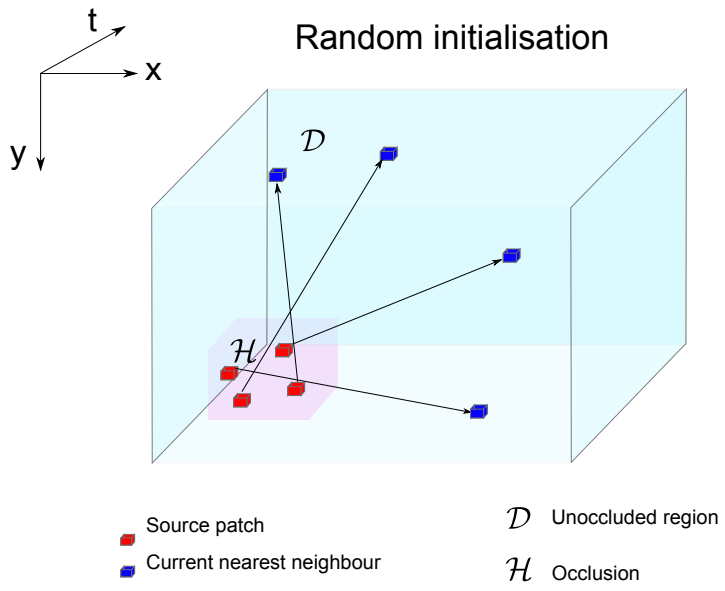
3D extension

Extension of the three steps in 3D:

3D extension

Extension of the three steps in 3D:

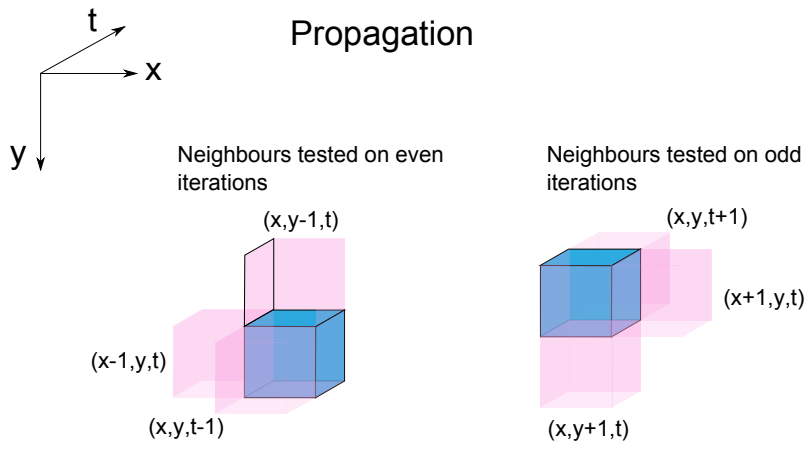
1. Random initialisation of ϕ



3D extension

Extension of the three steps in 3D:

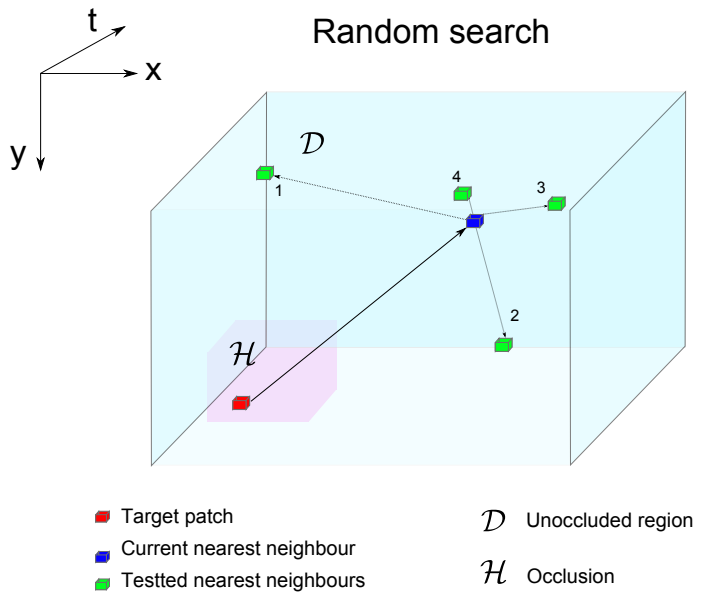
1. Random initialisation of ϕ
2. Propagation of good values of ϕ



3D extension

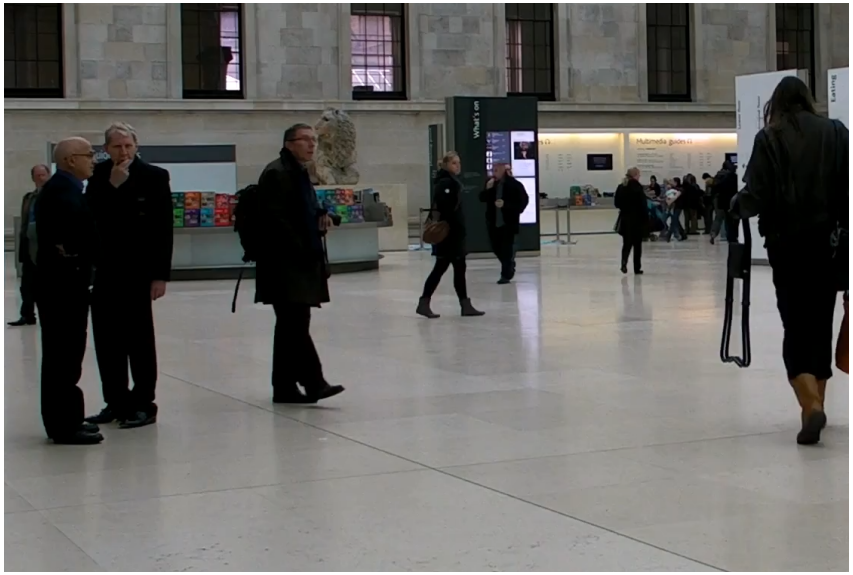
Extension of the three steps in 3D:

1. Random initialisation of ϕ
2. Propagation of good values of ϕ
3. Random search to improve shifts

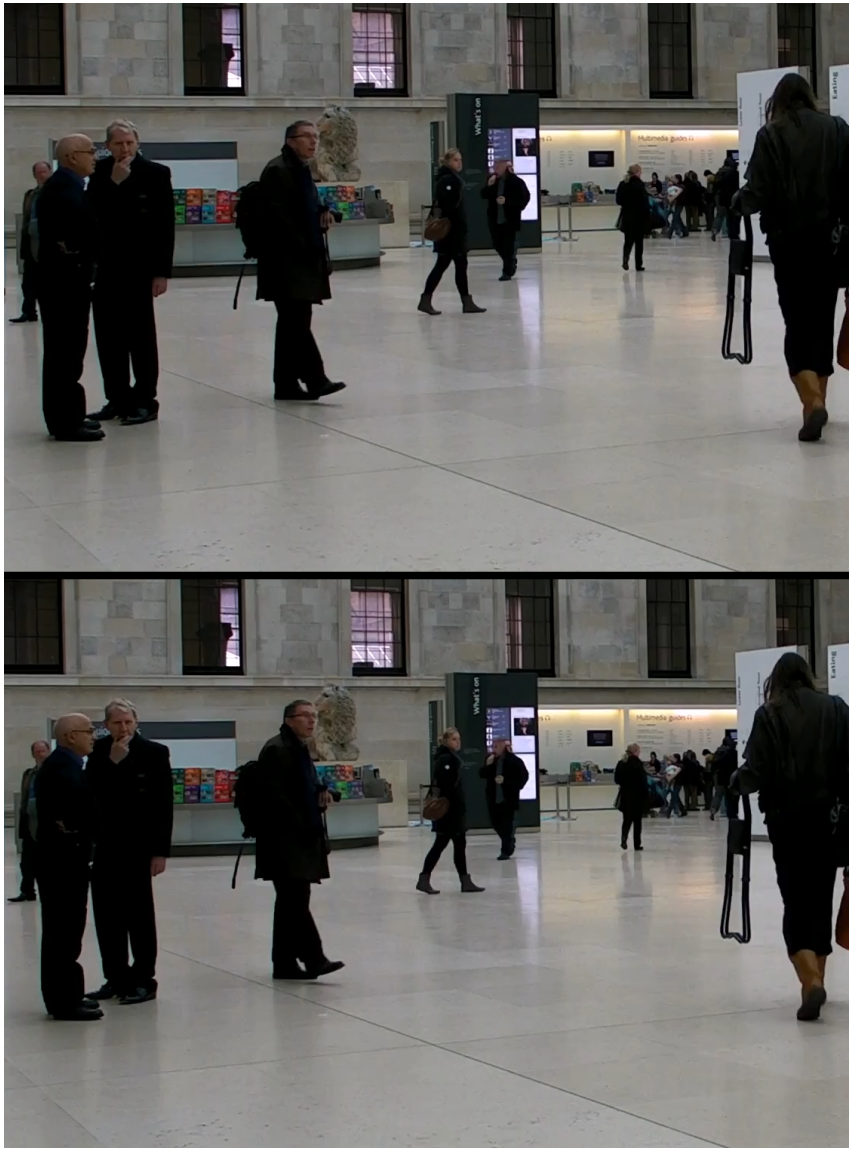


Computational time

High definition example (1120×754)



- 10-50 times speedup with 3D PatchMatch
- Still about 4 hours for 10 seconds of HD (1280 x 720)

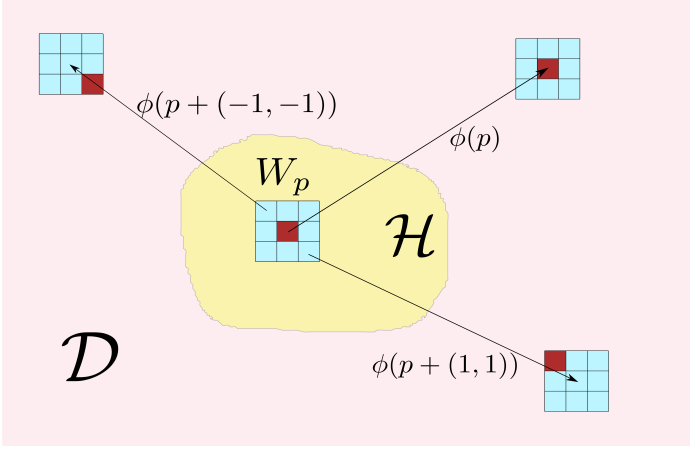


Reconstruction step

Video Reconstruction

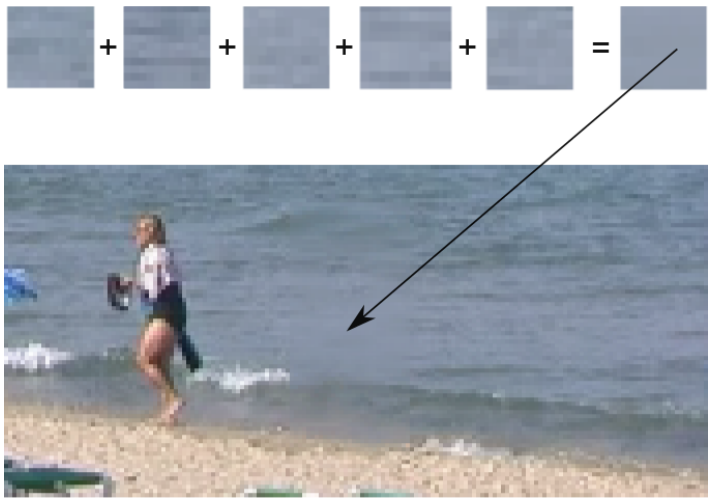
Each pixel p is reconstructed using the values predicted by the nearest neighbours of the patches which contain p.

$$u(p) = \frac{\sum_{q \in W_p} \omega_q u(p+\phi(q))}{\sum_{q \in W_p} \omega_q}$$



Weighted mean reconstruction scheme

Video Reconstruction



Problem: weighted average produces blurry results

$$u(p) = \frac{\sum_{q \in W_p} \omega^q u(p + \phi(q))}{\sum_{q \in W_p} \omega^q}$$

Video Reconstruction

Solution: use *best* patch at *end* of algorithm

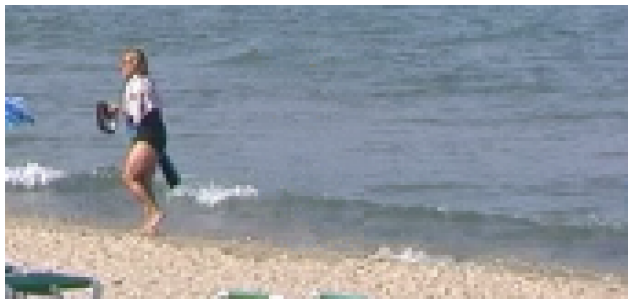
$$u(p) = u(p + \phi(q)),$$
$$\text{with } q = \arg\max_{j \in W_p}(\omega^j)$$

Video Reconstruction

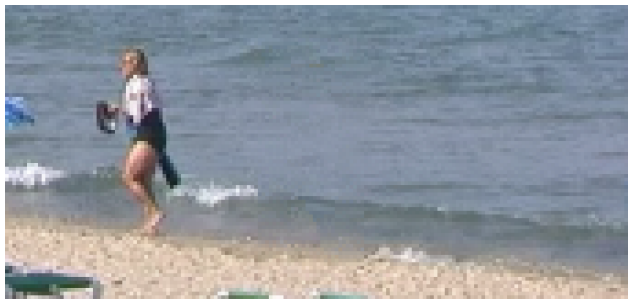
Solution: use *best* patch at *end* of algorithm

$$\begin{aligned} u(p) &= u(p + \phi(q)), \\ \text{with } q &= \arg\max_{j \in W_p}(\omega^j) \end{aligned}$$

A more complex solution relies on mean shift (Wexler *et al.*)



Inpainting with mean shift (Wexler *et al.*)



Inpainting with best patch at the last iteration

Multi-scale scheme

- The whole scheme is performed in a multi-scale manner
- The final result is obtained sequentially, starting at a coarse resolution
- Each time, the result at a given scale is upscaled and then taken as initialization at the next scale
- At the coarsest scale, the results is obtained using an onion peel method (greedy as the ones seen in the first part of the lecture).

The *multi-resolution* scheme is necessary to correctly inpaint structures.

Occluded image

Result with one pyramid level

Result with three pyramid levels

Textures in image/video inpainting

Dealing with textures in images and videos

Why do textures pose a problem ?

Original image

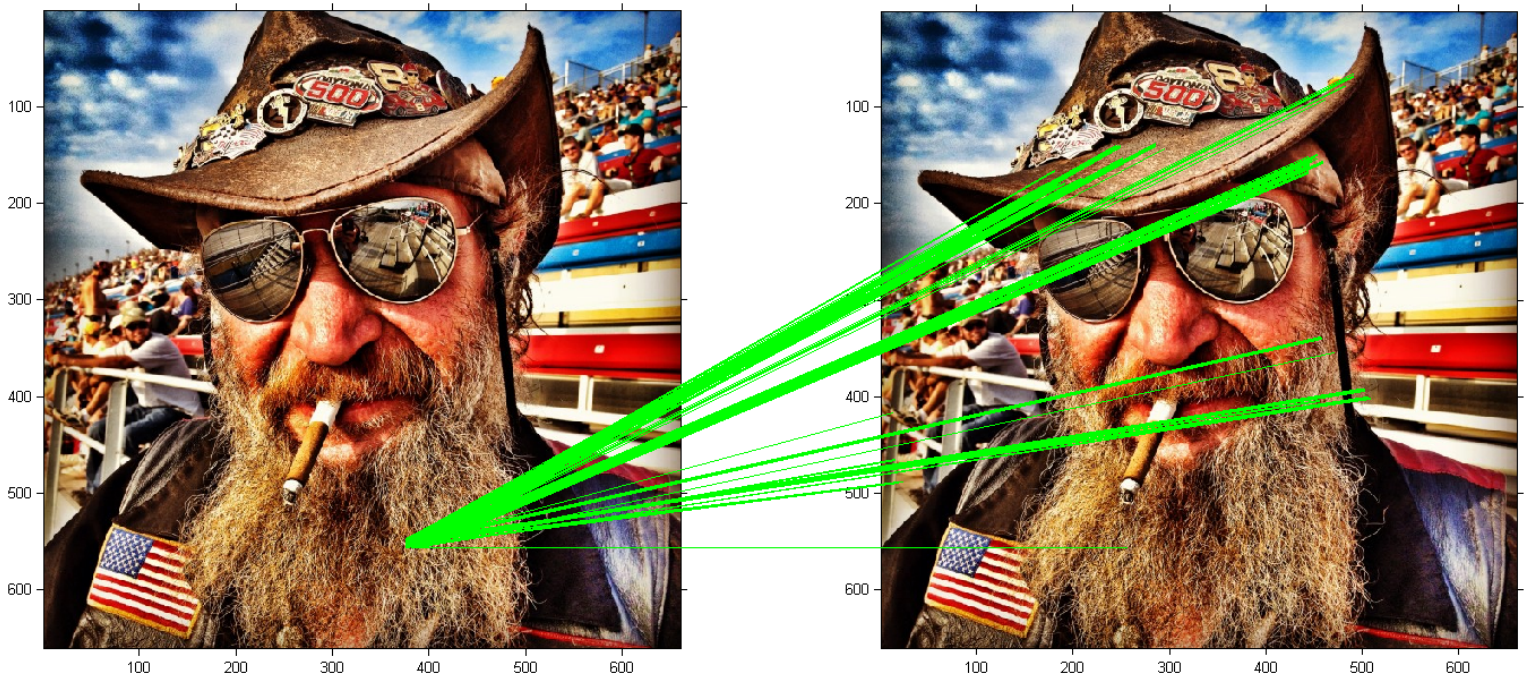
Dealing with textures in images and videos

Why do textures pose a problem ?

Inpainted image

Dealing with textures in images and videos

Why do textures pose a problem ?



Incorrect approximate nearest neighbours

Dealing with textures in images and videos

Why do we identify incorrect patches ???

Dealing with textures in images and videos

Why do we identify incorrect patches ???

Imagine we want to find the ANN of a *random* patch:

X

$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$
$$i = 1 \cdots N$$

Total patch size $= N$

?

Which patch is most similar to X ?

Y

$$y_i \sim \mathcal{N}(\mu, \sigma^2)$$
$$i = 1 \cdots N$$

Z

$$z_i = \mu$$
$$i = 1 \cdots N$$

Dealing with textures in images and videos

Why do we identify incorrect patches ???

Imagine we want to find the ANN of a *random* patch:

X

$x_i \sim \mathcal{N}(\mu, \sigma^2)$
$i = 1 \cdots N$

Total patch size $= N$

?

Which patch is most similar to X ?

Y

$y_i \sim \mathcal{N}(\mu, \sigma^2)$
$i = 1 \cdots N$

Z

$z_i = \mu$
$i = 1 \cdots N$

Patch distance Sum of Squared Differences (SSD):

$$d(X, Y) = \sum_{i=1 \cdots N} (x_i - y_i)^2.$$

Dealing with textures in images and videos

Why do we identify incorrect patches ???

Imagine we want to find the ANN of a *random* patch:

X

$x_i \sim \mathcal{N}(\mu, \sigma^2)$
$i = 1 \cdots N$

Total patch size $= N$

?

Which patch is most similar to X ?

Y

$y_i \sim \mathcal{N}(\mu, \sigma^2)$
$i = 1 \cdots N$

Z

$z_i = \mu$
$i = 1 \cdots N$

Patch distance Sum of Squared Differences (SSD):

$$d(X, Y) = \sum_{i=1 \cdots N} (x_i - y_i)^2.$$

$$x_i - y_i \sim \mathcal{N}(0, 2\sigma^2)$$
$$x_i - z_i \sim \mathcal{N}(0, \sigma^2)$$

Dealing with textures in images and videos

Why do we identify incorrect patches ???

Imagine we want to find the ANN of a *random* patch:

X

$x_i \sim \mathcal{N}(\mu, \sigma^2)$
$i = 1 \cdots N$

Total patch size $= N$

?

Which patch is most similar to X ?

Y

$y_i \sim \mathcal{N}(\mu, \sigma^2)$
$i = 1 \cdots N$

Z

$z_i = \mu$
$i = 1 \cdots N$

Patch distance Sum of Squared Differences (SSD):

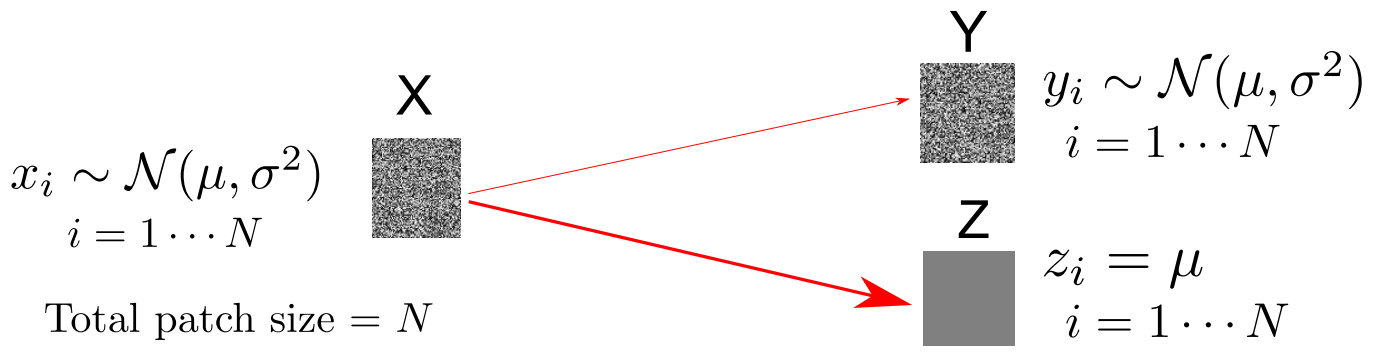
$$d(X, Y) = \sum_{i=1 \cdots N} (x_i - y_i)^2.$$

$$E[d(X, Y)] = 2N\sigma^2$$
$$E[d(X, Z)] = N\sigma^2$$

Dealing with textures in images and videos

Why do we identify incorrect patches ???

Imagine we want to find the ANN of a *random* patch:



Patch distance Sum of Squared Differences (SSD):

$$d(X, Y) = \sum_{i=1 \cdots N} (x_i - y_i)^2.$$

On average, $d(X, Y)$ is *twice* as large as $d(X, Z)$.

On average, constant patch Z is preferred !

Modified patch distance

We wish to include some information pertaining to the texture.

Idea : include an estimation of the local variance

Modified patch distance

We wish to include some information pertaining to the texture.

Idea : include an estimation of the local variance

A solution (from Liu and Caselles 2013) :

$$\text{SSD: } [\text{R}, \text{G}, \text{B}, \alpha g_\nu * |\nabla_x I|, \alpha\, g_\nu * |\nabla_y I|]$$

α: a weighting scalar

g_ν a gaussian kernel of size ν .

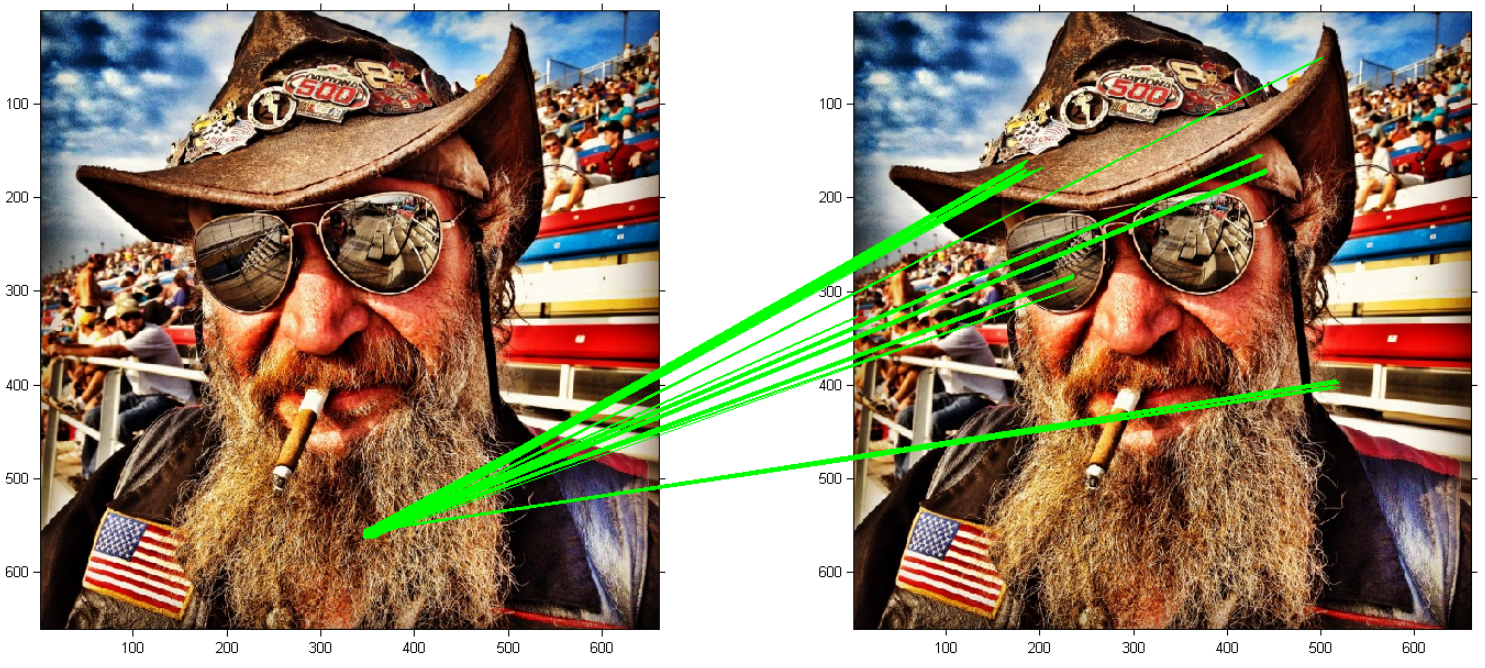
Y. Liu, V. Caselles, **Exemplar-Based Image Inpainting Using Multiscale Graph Cuts**, *IEEE TIP (2013)*
J. Bruna & S. Mallat (2013). **Invariant scattering convolution networks.** *IEEE TPAMI, 35(8), 187286*

Modified patch distance

Example of image created by $|\nabla_x I|_\nu$

Modified patch distance

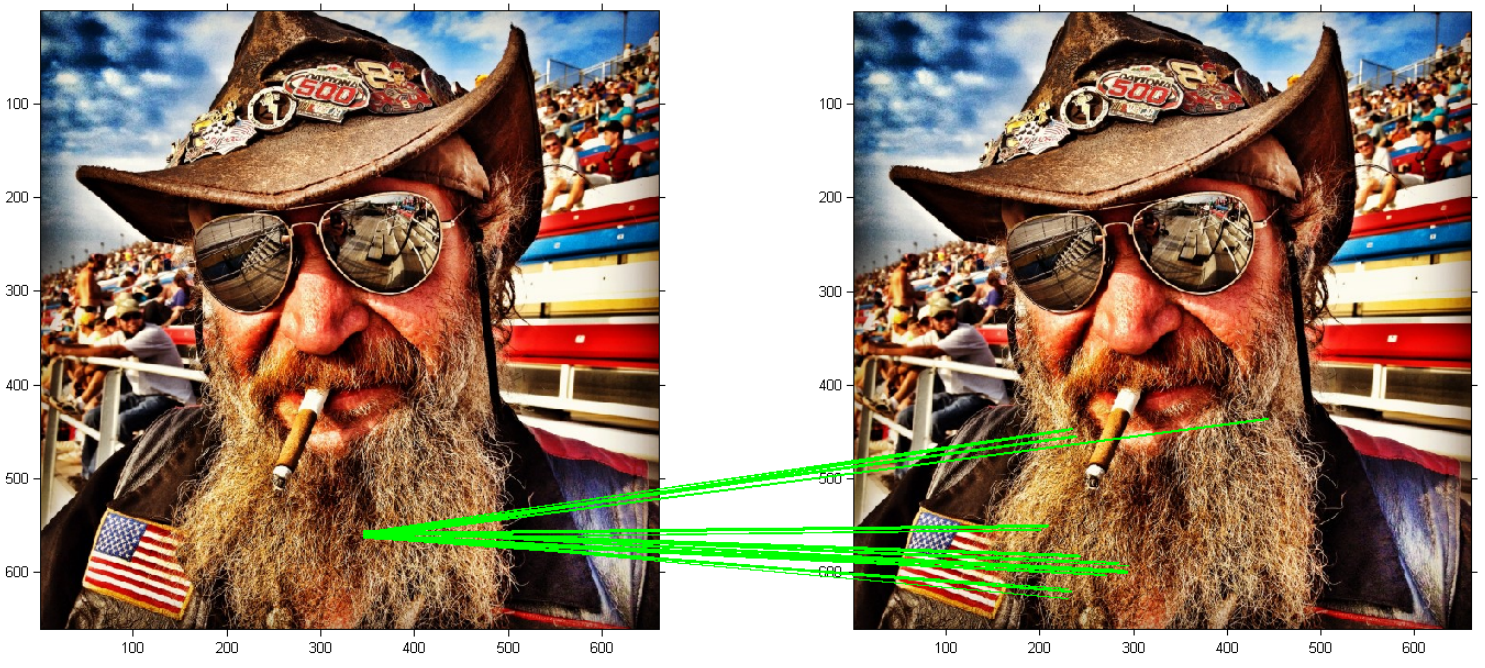
Example of the impact of the modified distance



PatchMatch with regular SSD

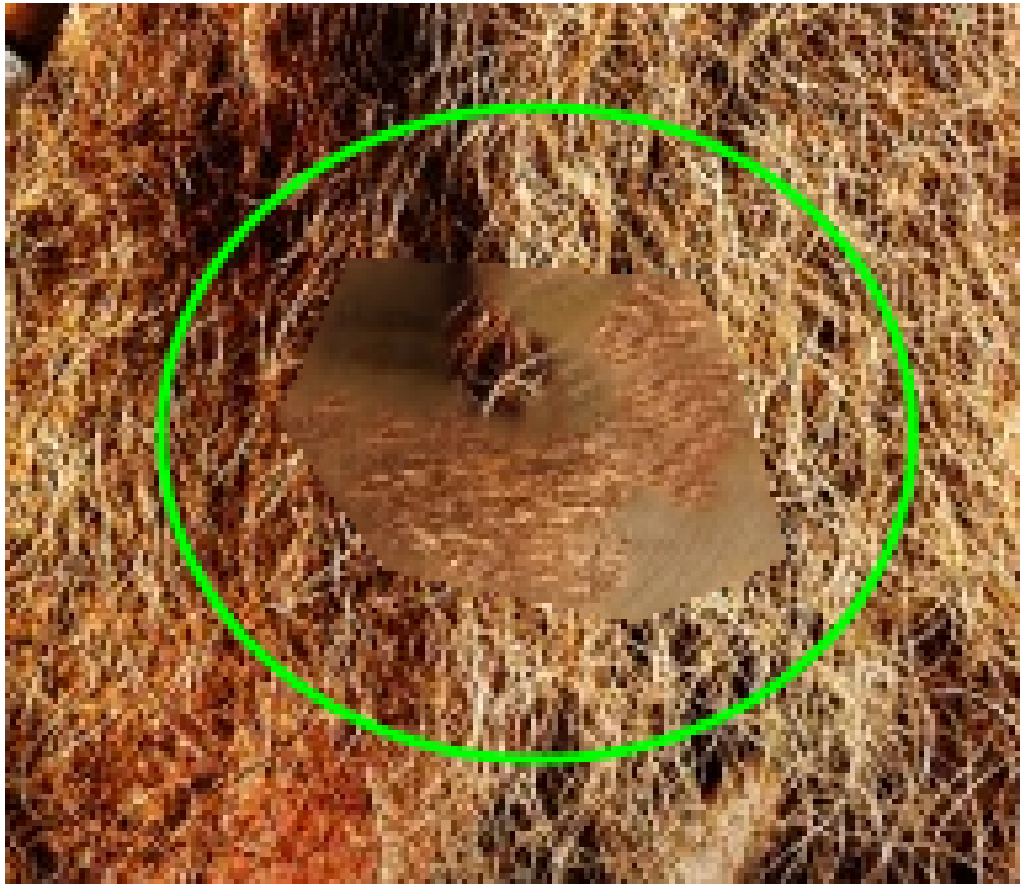
Modified patch distance

Example of the impact of the modified distance



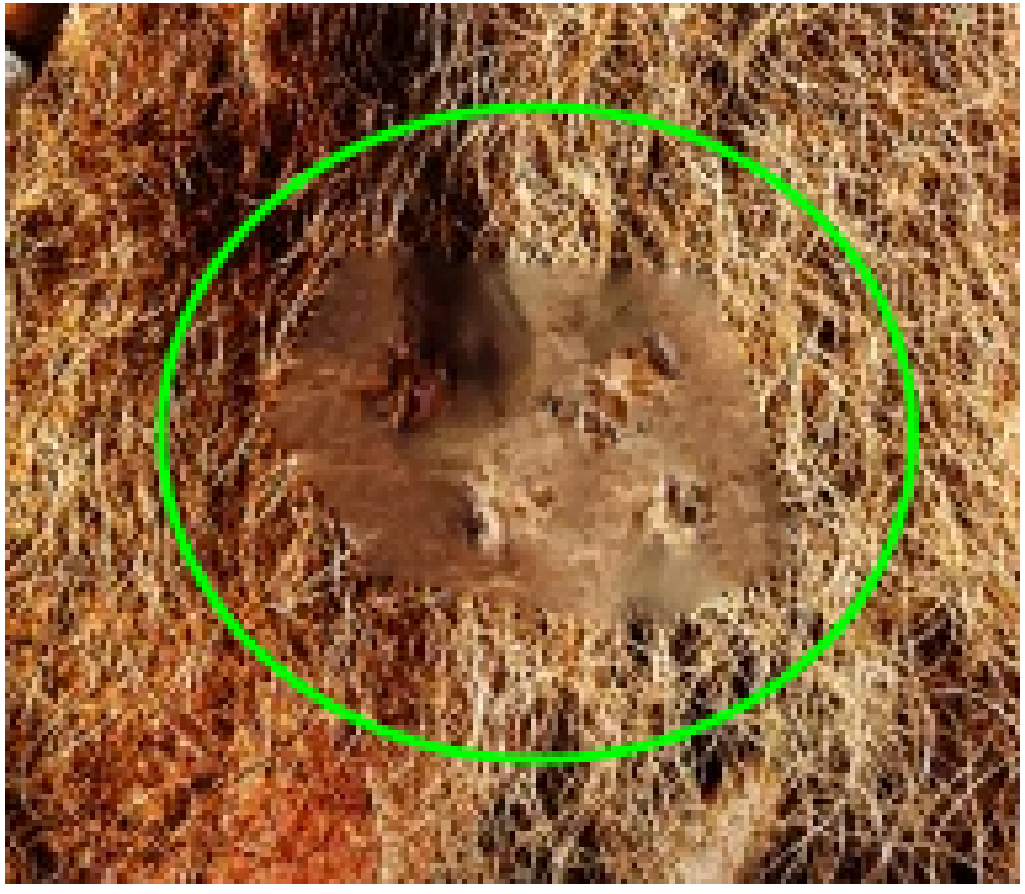
PatchMatch with modified SSD

Image example



Inpainting with unmodified patch distance

Image example



Inpainting with "Image Melding" (Darabi *et al.* 2012)

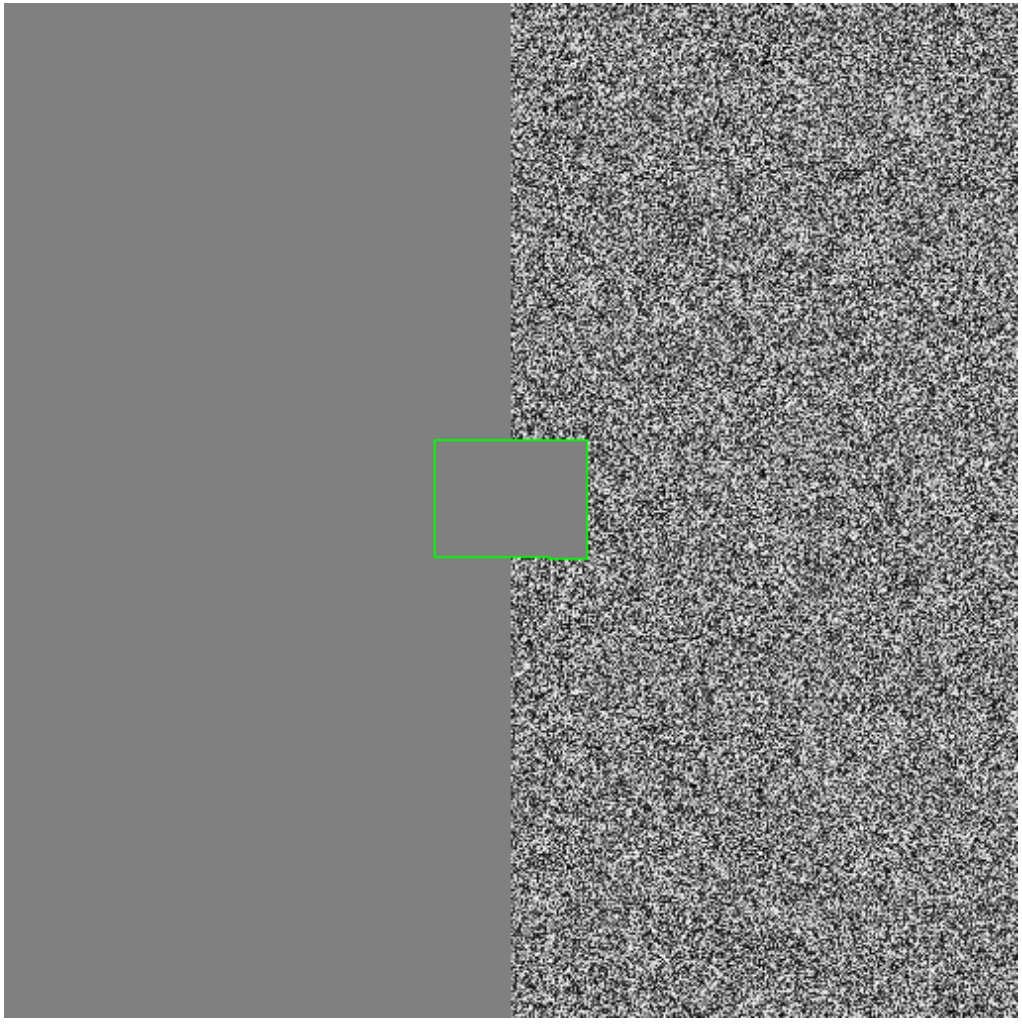
Image example

Inpainting with modified patch distance

Image example

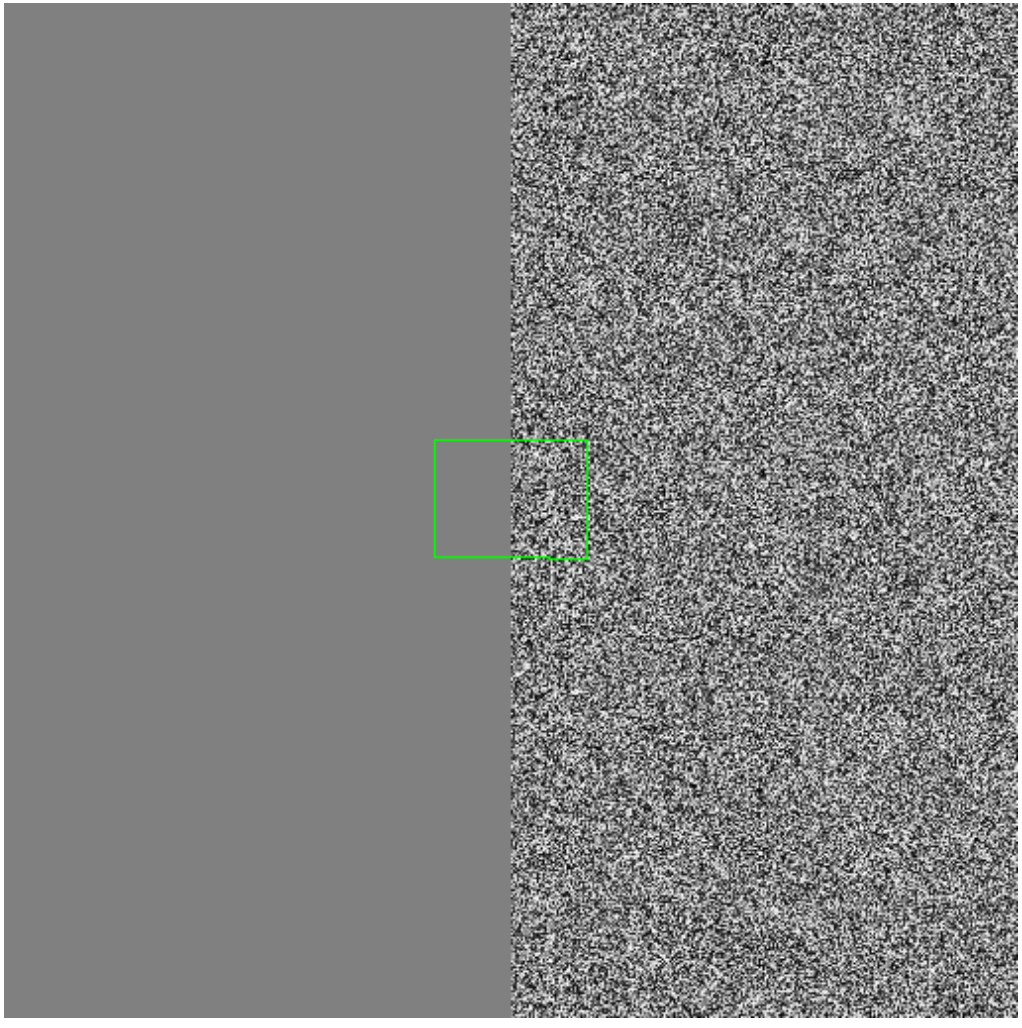
Original image

Noise example



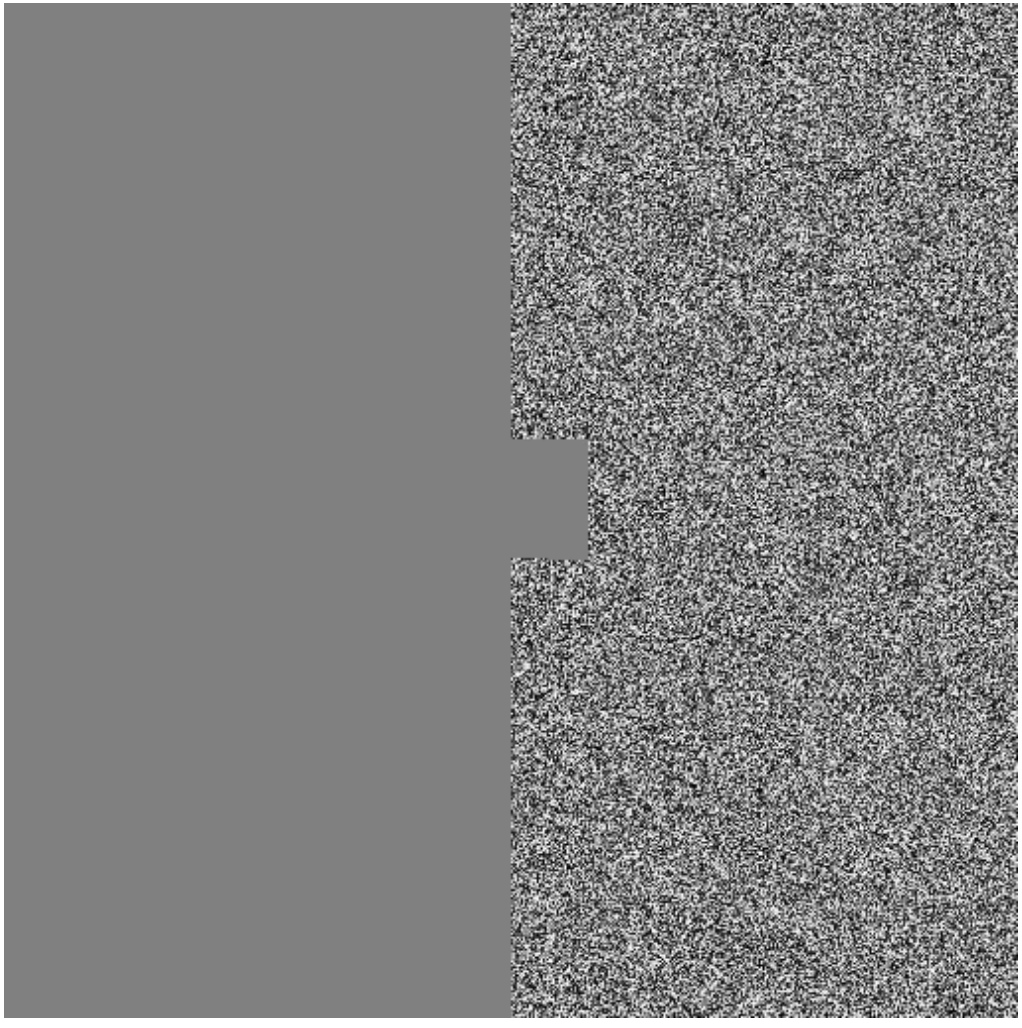
Inpainting with unmodified patch distance

Noise example



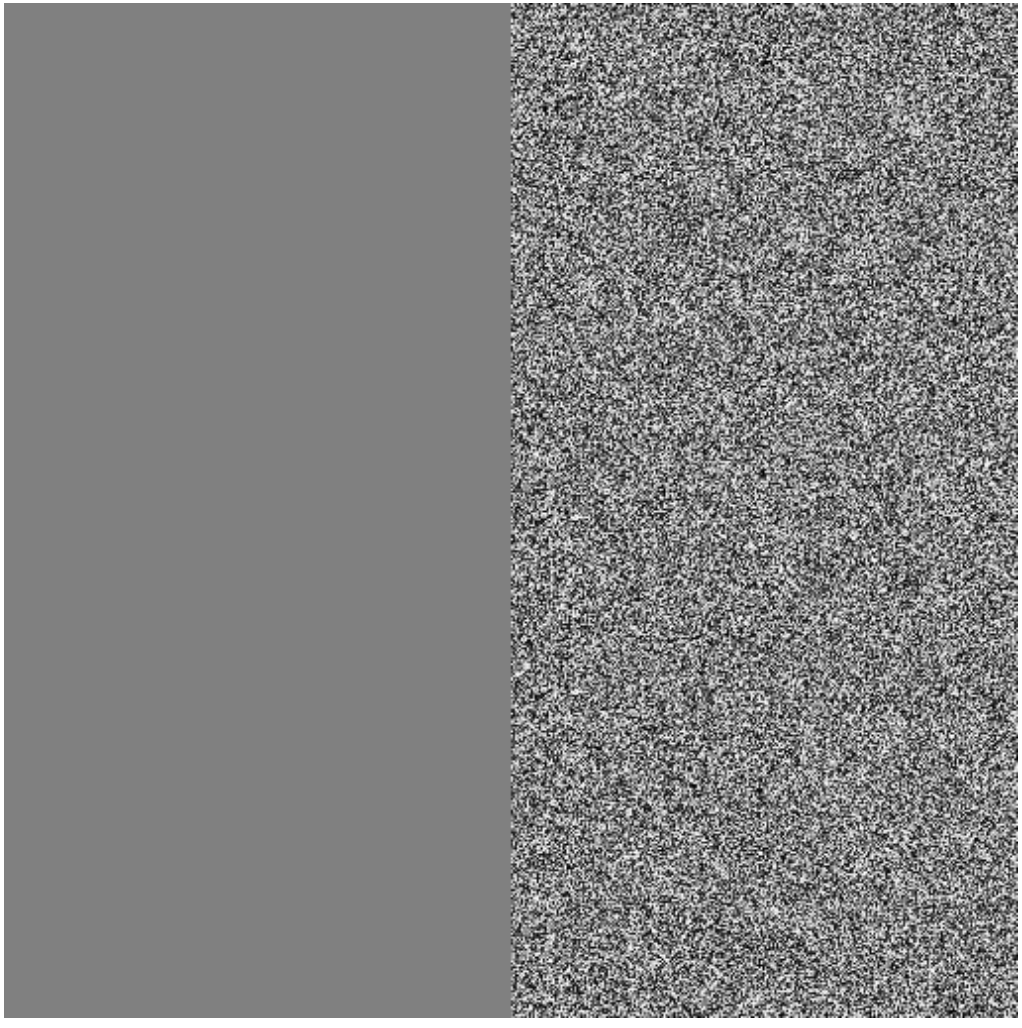
Inpainting with modified patch distance

Noise example



Inpainting with unmodified patch distance

Noise example



Inpainting with modified patch distance

Video example

Original video

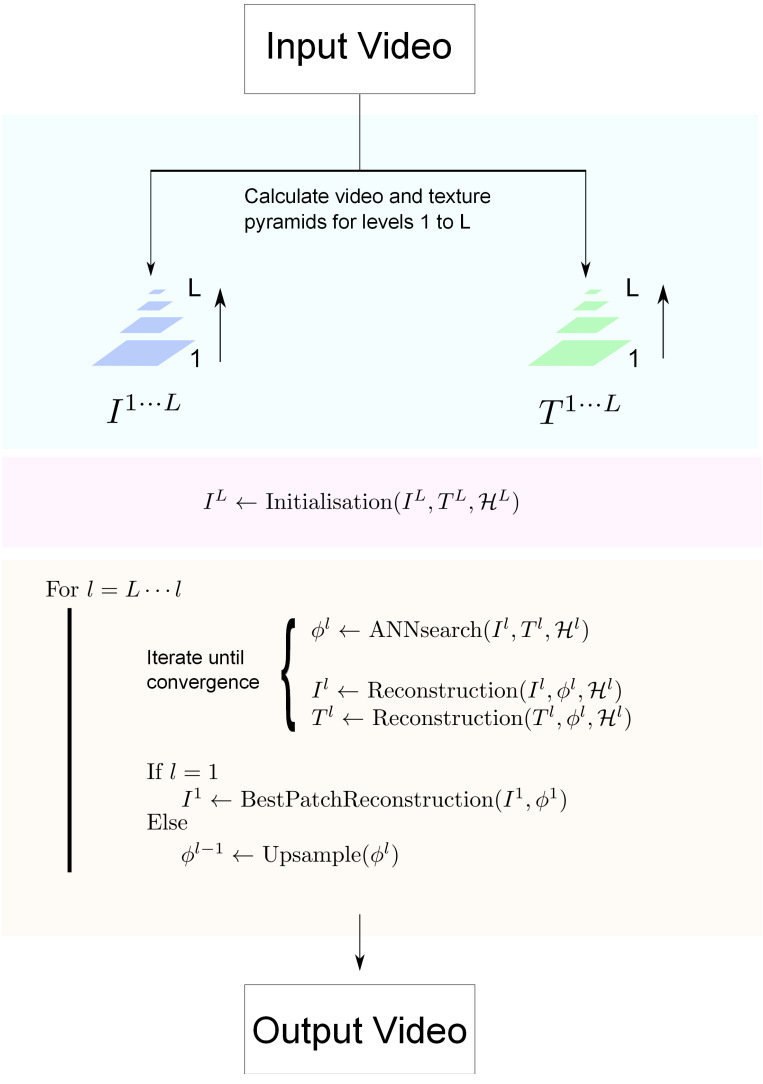
Video example

Unmodified patch distance

Video example

Modified patch distance

Complete algorithm



Inpainting with moving background

Inpainting with moving background

Inpainting may need to be done with moving backgrounds/cameras. Why is this a problem ?

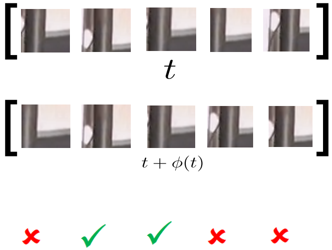
Inpainting with moving background

Inpainting may need to be done with moving backgrounds/cameras. Why is this a problem ?

- Patches are *spatio-temporal*
- The same motion sequence may potentially not be repeated

$W_p =$

$W_{p+\phi(p)} =$



Inpainting with moving background

- One possibility, realign patches: would vastly increase time complexity
- Instead, try realigning the *whole* image (e.g. Odobez and Bouthemy 1995)

Inpainting with moving background

- One possibility, realign patches: would vastly increase time complexity
- Instead, try realigning the *whole* image (e.g. Odobez and Bouthemy 1995)

Original video

Realigned video

Original frame : “Jumping man¨

Inpainting without realignement

Inpainting with realignement

Inpainting with moving background

Result without realignment

Inpainting with moving background

Result with realignment

Inpainting results

Original video

Inpainting results

Result

Inpainting results

Original video

Inpainting results

Result

Application of video inpainting to scratch restoration

Original video (with synthetic scratches)

Application of video inpainting to scratch restoration

Restoration using inpainting

More videos / paper / code : `http://perso.telecom-paristech.fr/~gousseau/video_inpainting/`