# IMAGE INPAINTING

**João Fontes Gonçalves**
Final project - INF573
École Polytechnique
joao.fontes-goncalves@polytechnique.edu

January 5, 2020

## 1 Introduction

This project is about filling in an unknown region in an image. The notation used here is the same as the one used by (Newson et al., 2016) which is:

- $\Omega$ denotes the image support.
- $H$ is the occluded region.
- $D$ is the unnocluded region.
- $\widetilde{D}$ is the set of positions such that for all $p \in \widetilde{D}$, $N_p \subset D$.
- $\widetilde{H}$ is the complement of $\widetilde{D}$ in $\Omega$.
- $p = (x, y)$ is a position in the support.
- $u : \Omega \rightarrow R^3$ denotes the image content.
- $N_p$ is the patch neighborhood of a pixel p.
- $W_p = (u(x_1), ..., u_(x_N))$ denotes the patch itself.
- The NN of a $W_p$ is a patch $W_q$ which minimizes a certain patch distance $d(W_p, W_q)$.
- The shift map $\phi : \Omega \rightarrow N^2$ is a vector field which shows where the NN of a patch is located. That is, the NN of $W_p$ is $W_{p+\phi(p)}$

## 2 Method

### 2.1 Variational framework

Initially, I tried to minimize the following functional: $E(u, \phi) = \sum_{p \in H} d^2(W_p, w_{p+\phi(p)})$.

Where $d^2(W_q, W_{q+\phi(q)}) = \sum_{q \in N_p} \|u(q) - u(q + \phi(p))\|_2^2$.

Following the heuristic proposed by (Wexler et al.), a solution is obtained using an iterated alternating minimization, as well as a multi-scale framework.

At each scale, we minimize firstly with respect to the shift map $\phi$, then with respect to the image content $u$. These two steps corresponds to the following operations:

- Nearest neighbor search
- Image reconstruction

**Algorithm 1:** ANN search with PatchMatch.

**Data**: Current inpainting configuration $u$ (height: $m$, width: $n$), $\phi$, $\tilde{\mathcal{H}}$
**Parameters**: $r_{max}$ ($\max(m,n)$), $\rho$ (0.5)
**Result**: ANN shift map $\phi$

**for** $k = 1$ **to** $k_{max}$ **do**
  **for** $i = 1$ **to** $|\tilde{\mathcal{H}}|$ **do**
    **if** $k$ *even* **then** `/* Propagation on even iteration (lexicographical order) */`
      $p = p_i$;
      $a = p - (1,0)$, $b = p - (0,1)$;
      $q = \arg\min_{r \in \{p,a,b\}} d(W_p^u, W_{p+\phi(r)}^u)$;
      **if** $p + \phi(q) \in \tilde{\mathcal{D}}$ **then** $\phi(p) \leftarrow \phi(q)$
    **else** `/* Propagation on odd iteration (inverted order) */`
      $p = p_{|\tilde{\mathcal{H}}|-i+1}$;
      $a = p + (1,0)$, $b = p + (0,1)$;
      $q = \arg\min_{r \in \{p,a,b\}} d(W_p^u, W_{p+\phi(r)}^u)$;
      **if** $p + \phi(q) \in \tilde{\mathcal{D}}$ **then** $\phi(p) \leftarrow \phi(q)$
    **end**
    $z_{max} \leftarrow \lceil -\frac{\log(\max(m,n))}{\log(\rho)} \rceil$;
    **for** $z = 1$ **to** $z_{max}$ **do**
      $q = p + \phi(p) + \lfloor r_{\max} \rho^z \text{RandUniform}([-1,1]^2) \rfloor$;
      **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**
**end**

Figure 1: Pseudo-code for PatchMatch algorithm

### 2.2 Approximate Nearest neighbor Search

NN search can be quite computationally expensive here. So it's necessary to use approximate nearest neighbors (ANN) instead of exact nearest neighbors. PatchMatch algorithm (introduced by (Barnes et al)) was used for this task. It relies on the idea that good shift maps tend to be piecewise constant: a relative shift which leads to a good NN for a patch $W_p$ has a good chance of leading to a good NN to the patches situated around $W_p$. It consists of three steps:

- Random initialization
- Propagation
- Random search

The pseudo-code for PatchMatch algorithm is given in figure 1.

### 2.3 Reconstruction

It was used the weighted mean scheme proposed by (Wexler et al.).

Given a shift map, each pixel is reconstructed as:
$u(p) = \frac{\sum_{q \in N_p} s_q^p u(p+\phi(q))}{\sum_{q \in N_p} s_q^p}$, where $s_q^p$ is the weight assigned to the pixel value for $p$ indicated by the ANN of $W_q$:
$s_q^p = exp(\frac{-d^2(W_q, W_{q+\phi(q)})}{2\sigma^2})$, where $\sigma$ is the $75th$ percentile of all the current patch distances, as is done by (Wexler et al.). This operation is carried out for all pixels $p \in H$.

### 2.4 Multiscale scheme

The functional we want to minimize is highly non-convex. So, the use of a multiscale scheme is of crucial importance in avoiding local minima of the energy functional.

To implement this multiscale scheme I used a Gaussian pyramid with 5 x 5 Gaussian kernel as showed in the figure 2.

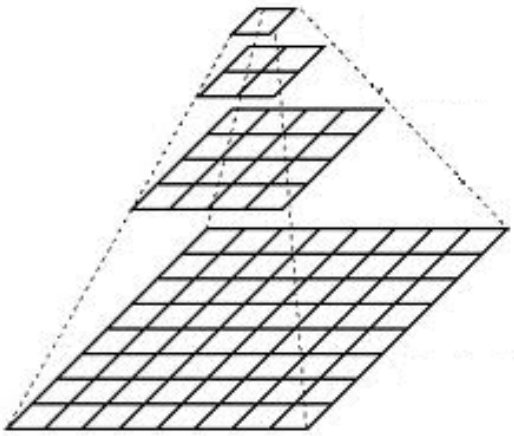

Figure 2: Multiscale scheme

### 2.5 Details of the multiscale scheme

The inpainting solution is passed from a coarser pyramid level to a finer one by upsampling the shift map $\phi$ rather than the inpainting solution itself, as proposed by (Newson et al.). This is done to avoid the inevitable smoothing which would happen if we took the second option.

The shift map is upsampled using a simple nearest neighbors interpolation with non-integer coordinates resulting from this interpolation being rounded down to the nearest integer.

To determine the number of pyramid levels, I used a relation between the occlusion size and the patch size, as proposed by (Newson et al.). The occlusion is iteratively eroded with a square structuring element of width three pixels until it disappears.

Let $N_o$ be the number of times that the occlusion needs to be eroded for it to disappear and let $N$ be the maximum between the two sizes of the patch rectangle. Then, the number of pyramid levels is: $L = log_2\left(\frac{2N_o}{N}\right)$.

### 2.6 Initialization

For initialization of the inpainting solution, it was implemented an "onion-peel" approach which inpaints the occlusion one layer at a time, eroding the occlusion progressively, as proposed by (Newson et al.). The layers are one-pixel thick, and are located at the border of the occlusion. Each pixel in a layer is processed using the information of the inpainting solution of the previous layer, and thus there is no processing order for pixels within a single layer.

It's necessary a partial patch comparison, since the content of the patches will not be completely known.

If we define $H' \subset H$ as the current occlusion during the erosion and $\partial H' \subset H'$ the current layer to inpaint, then a partial neighborhood can be defined as:
$N'_p = \{q \in N_p, q \notin H'\}$

And we change $N_p$ by $N'_p$ during the computation of patch distance and pixel reconstruction.

### 2.7 Inpainting with textures

When several regions with a similar average color but differing texture contents exist, three main problems can arise with multi-scale and iterative algorithms:

- Ambiguities in patch comparisons at coarse pyramid levels.
- Failure of the regular $l_2$ patch distance to correctly compare textures.
- The weighted mean reconstruction tends to smooth textures during the algorithm, which in turn leads to smooth patches being chosen as NNs.

**Algorithm 2:** Complete proposed inpainting algorithm.

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

$\{u^\ell\}_{\ell=1}^L \leftarrow$ImagePyramid$(u)$;
$\{T^\ell\}_{\ell=1}^L \leftarrow$TextureFeaturePyramid$(u)$; // Equation (6)
$\{\mathcal{H}^\ell\}_{\ell=1}^L \leftarrow$OcclusionPyramid$(\mathcal{H})$;
$\phi^L \leftarrow$Random;
$(u^L, T^L, \phi^L) \leftarrow$Initialization$(u^L, T^L, \phi^L, \mathcal{H}^L)$; // Equation (11)
**for** $\ell = L$ **to** $1$ **do**
  $k = 0$, $e = 1$;
  **while** $e > 0.1$ **and** $k < 10$ **do**
    $v = u^\ell$;
    $\phi^\ell \leftarrow$ANNsearch$(u^\ell, T^\ell, \phi^\ell, \mathcal{H}^\ell)$ ; // Algorithm 1
    $u^\ell \leftarrow$Reconstruction$(u^\ell, \phi^\ell, \mathcal{H}^\ell)$ ; // Equation (3)
    $T^\ell \leftarrow$Reconstruction$(T^\ell, \phi^\ell, \mathcal{H}^\ell)$;
    $e = \frac{1}{3|\mathcal{H}^\ell|}\|u^\ell_{\mathcal{H}^\ell} - v_{\mathcal{H}^\ell}\|_1$;
    $k \leftarrow k + 1$;
  **end**
  **if** $\ell = 1$ **then**
    $u \leftarrow$FinalReconstruction$(u^1, \phi^1, \mathcal{H})$ ; // Equation (5)
  **else**
    $\phi^{\ell-1} \leftarrow$UpSample$(\phi^\ell, 2)$ ; // Section 3.5
    $u^{\ell-1} \leftarrow$Reconstruction$(u^{\ell-1}, \phi^{\ell-1}, \mathcal{H}^{\ell-1})$;
    $T^{\ell-1} \leftarrow$Reconstruction$(T^{\ell-1}, \phi^{\ell-1}, \mathcal{H}^{\ell-1})$;
  **end**
**end**

Figure 3: Complete proposed inpainting algorithm

To address these issues I introduced texture features into the patch distance. Inspired by the work of (Liu and Caselles) I tryed the following features: $T_x(p) = \frac{1}{card(v)} \sum_{q \in v} |I_x(p)|$ and $T_y(p) = \frac{1}{card(v)} \sum_{q \in v} |I_y(p)|$.

Where $v$ is a neighborhood centered on $p$ and $I_x$ and $I_y$ are the derivatives of the image in the x and y directions.

The squared patch distance is now redefined as:
$d^2(W_q, W_{q+\phi(q)}) = \sum_{q \in N_p} (\|u(q) - u(q + \phi(p))\|_2^2 + \lambda \|T(q) - T(q + \phi(p))\|_2^2)$

Where $\lambda$ is a weighting scalar to balance the effects of both color and texture information in the patch distance.

These features are calculated at the finest pyramids level for the unnocluded pixels $p \in \widetilde{D}$ and propagated to coarser levels by nearest neighbor subsampling.

To avoid smoothing the result due to the patch averaging process we can create a separate image with the texture features and inpaint this image in parallel to the color image.

The final algorithm is showed in figure 3 (taken from (Newson et al., 2016)).

## 3 Results

The parameters used are the same proposed by (Newson et al., 2016):

- It was used a patch size of 7 x 7.
- The parameters of the PatchMatch algorithm are considered to be fixed. The random search reduction window is set to $\rho = \frac{1}{2}$. It was used ten iterations of propagation/random search in PatchMatch.
- The weight $\lambda$ associated with texture features is set to $50$. The texture neighborhood $v$ is a square neighborhood of size $2^L$.
- Finally, the convergence of the algorithm at a pyramid level is obtained by determining the average absolute pixel value difference for each color channel between the current solution and the previous one. If this value drops below $0.1$ or the number of iterations exceeds ten, we stop iterating at the current level.

My code receives two images as input: the image and the occlusion and gives as output the inpainted image at the base of the pyramid.

Figure 4: Original image

Figure 5: Second level

As example, consider the pyramid obtained for the figure 4. By calculating the number of levels for the pyramid using the erosion process explained in the last section, I got that the ideal pyramid should have two levels for this image. The base is the original image and the second level is given by figure 5.

The occlusion pyramid is given by the figures 6 and 7.

The onion-peel initialization for the top level of the pyramid gave the result showed in figure 8.

The final result is probabilistic, because it depends on the random initialization of the shift map. In figure 9 and 10 is showed some inpainting results after running the code.

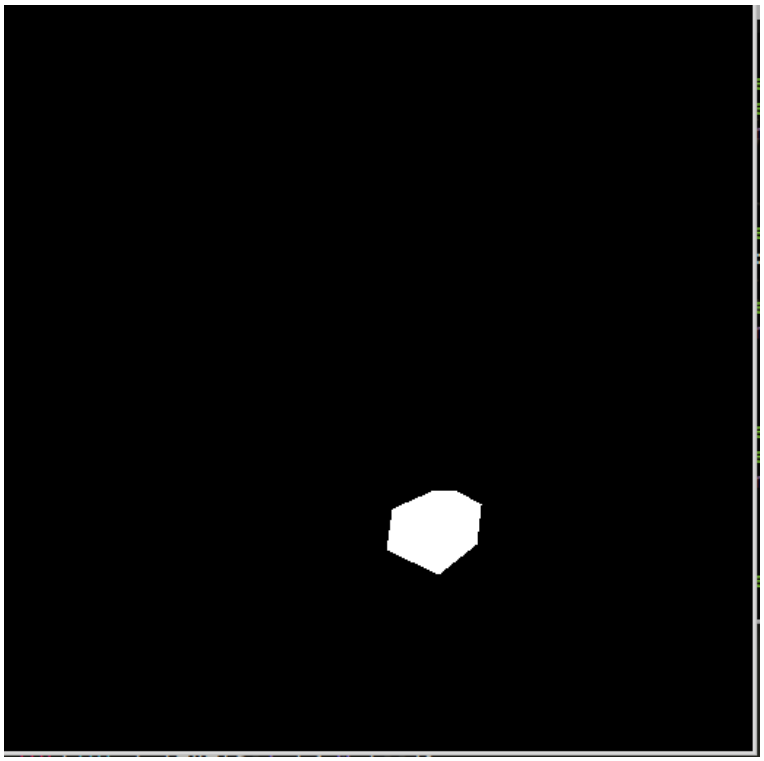

Figure 6: Original occlusion

Figure 7: Second level for occlusion

Figure 8: Onion-peel initialization

Figure 9: Inpainting 1

Figure 10: Inpainting 2