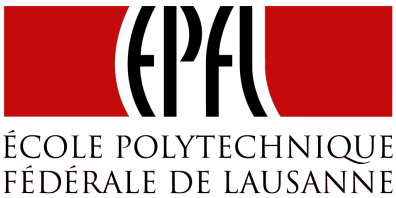


SEMESTER PROJECT

Major in Information Technologies

GIIN

Graph-based Image Inpainting

Student
Michaël Defferrard

Professor
Pierre Vandergheynst

Supervisors
Johan Paratte
Nathanaël Perraudin
Yann Schoenenberger

Proposed by and conducted at the EPFL LTS2 laboratory.

February 12, 2015

Contents

1 Introduction

This report presents the work accomplished during an eleven ECTS semester project at EPFL LTS2 laboratory[1]. It is part of the major in Information Technologies curriculum. The project goal was to explore the applications of graphs for image inpainting. The main idea of this work was to use a non-local patch graph of the image as the underlying structure to address the inpainting problem of large missing chunks.

The objectives of the project were the following:

1. First a literature study was to be conducted. The following papers were used: [5] for patch-based methods, [6] for image inpainting using a non-local patch graph, [3] and [4] for the priority scheme and [1] for a non-linear dimensionality reduction (NLDR) tool which was finally not used but nonetheless considered. [7] was used as an introduction to signal processing on graphs.

2. We then wanted to show that a non-local patch graph indeed possesses enough information about the image to convincingly inpaint it.

3. Once we proved it, we could work out a way to reconstruct the original graph from a partial observation. We devised and implemented an inpainting scheme based on the reconstruction of a non-local patch graph. The implementation was done with MATLAB using the Graph Signal Processing Toolbox[2] and the UNLocBoX[3], a convex optimization toolbox. The actual implementation of the complete algorithm can be found on our GitLab[4].

4. We finally wanted to assess the quality of the reconstructed graph.

This writing is meant to be a report of the work accomplished during the project as well as a base for a following paper to be published. It is organized as follow: we restate the problem and give some background information in section 2. Our method is presented in section 3. Section 4 presents some experimental results, followed by a discussion in section 5. We finally conclude in section 6.

[1] The laboratory homepage is accessible at http://lts2www.epfl.ch/.

[2] The GSPbox is developed and maintained by the LTS2 group. Code and documentation are available at https://lts2research.epfl.ch/gsp/.

[3] The UNLocBoX is developed and maintained by the LTS2 group. Code and documentation are available at http://unlocbox.sourceforge.net/.

[4] Our GitLab is accessible through https://lts2srv1.epfl.ch/gitlab/. The code is private for now but we intend to publish it soon on GitHub under a free software license.

2 Background

Image inpainting is the problem of finding missing parts of an image using only the available content and some prior information. The hallucinated area should look visually plausible to the human eye. Inpainting is thus a difficult image processing problem involving knowledge about image models and regularization techniques. It has gained in importance with the growth of the digital photography market as it allows users to remove disturbing elements from their pictures.

Several researchers have previously considered texture synthesis as a way to fill large regions with repetitive two-dimensional textural patterns. They developed and applied alternative exemplar-based techniques which tackle the lack of an explicit mathematical texture model. Theses textures could be repeated ad infinitum given a small sample of pure texture. While these methods are effective to inpaint a hole in a single texture, they fail to reconstruct transitions between such textures. Region boundaries are indeed a complex product of mutual influences between different textures.

Geometry-driven algorithms try to find a good continuation of the hole surrounding, effectively propagating linear structures, called isophotes, into the target region. Theses techniques however lack support for texture information and are consequently unable to reproduce them. The blur introduced by the diffusion also becomes noticeable when filling larger regions. They are thus hardly suitable for wide area inpainting.

Few attempts were made to conciliate both geometry and texture-based approaches in an unified framework. [4] presented an algorithm which combines the strengths of both approaches into a single, efficient algorithm. They use linear structures to influence the fill order of their exemplar-based texture synthesis algorithm. The result is an algorithm that has the efficiency and qualitative performance of exemplar-based texture synthesis, but which also respects the image constraints imposed by surrounding linear structures.

Natural images have an important property: they are highly redundant. For small enough blocks, chances are high that a bunch of similar patches can be found all over the image. This redundancy is widely exploited by compression algorithms. It can also be used to infer the pixel values of an unknown patch, given that we have some information to match it to a cluster of similar patches.

A framework for image inpainting proposed by [6] relies on information diffusion on non-local patch graph. The recent trend in non-local methods is due to the success of the Non-Local Means denoising filter [2]. Their intuition is simple : similar-looking patches are more likely to represent the same phenomenon, and hence should be used

when averaging pixels without considerations to their spatial distance to obtain an efficient denoising process that also respects image textures. The use of graphs makes the framework more flexible than other non-local formulations, allowing for example to mix spatial and non-local constraints. The strength of the method is that it combines exemplar-based (similarity between patches) and diffusion-based (heat flow on graph) techniques for image inpainting and exploits the redundancy of image data, finding a sparse representation (as for compression).

In this work, we present an algorithm which combines the strengths of the aforementioned methods. We use a non-local patch graph representation of the image for the flexibility it offers. As [4], we pay special attention to linear structures and use them to influence the fill order. Instead of using a conventional edge detector, we will introduce a novel approach which takes advantage of the non-local graph representation. While [6] recomputes the graph after each numerical iteration (heat flow), we continuously update our graph, leading to a faster algorithm.

3 Proposed method

A sketch of our algorithm goes as follow: first, given an input image I, the user selects a target region, Ω, to be removed and filled. We then construct a non-local patch graph from the known patches of the source region, Φ, defined as the entire image minus the target region ($\Phi = I - \Omega$). The connection weight between two patches is defined by some measure of their similarity. Unknown patches are completely disconnected. Then start a sequential process: those among the disconnected patches who possess a sufficient amount of information (i.e. some fraction of the pixels they represent have a value) are compared against all known and connected patches and inserted into the graph with appropriate weights. The graph is thus sequentially completed, while no pixels have actually been inpainted. To allow further graph completion, we must indeed inpaint unknown pixels to recover enough information about a patch to compare it to the others. A basic approach is to select an unknown patch and copy its most similar known patch over. The selection of the patch is given by a priority measure which objective is to favor the most constrained areas. Our algorithm is thus an iterating process between node insertion and pixel inpainting which indeed combine an exemplar-based synthesis with a geometry-based priority assignment. We know present in greater details the aforementioned steps.

3.1 Patches

Assume we are given an image I defined on a domain $\Omega \subset \mathbb{R}^2$, usually a rectangle, with values in $\mathbb{R}$ (for grayscale images) or $\mathbb{R}^3$ (for color images). We write $\mathbf{x}, \mathbf{y} \in \Omega \times \Omega$ two pixel locations. We define a patch extraction operator R who returns the pixels into a squared $\sqrt{d} \times \sqrt{d}$ neighborhood around $\mathbf{x}$ and stacks them in a vector:

$$R(\mathbf{x}) = (I(\mathbf{x}_1), ..., I(\mathbf{x}_d))^T \in \mathbb{R}^d. \tag{1}$$

Given two patches, we define the distance between them as the usual squared Euclidean distance

$$d(R(\mathbf{x}), R(\mathbf{y})) = \|R(\mathbf{x}) - R(\mathbf{y})\|_2^2 = \sum_{i=1}^{d} (R(\mathbf{x})_i - R(\mathbf{y})_i)^2 . \tag{2}$$

According to [5], the visual similarity between two patches normalized in [0,1] (ranging from no similarity to identical) can be computed given the distance between them by a simple exponential filtering:

$$w(\mathbf{x}, \mathbf{y}) = e^{-\frac{d(\mathbf{x},\mathbf{y})}{h^2}}. \tag{3}$$

The parameter h controls the decay of the exponential function. A small h defines a selective filters (only very similar patches will have a significant similarity score) while a large h will attribute a more uniform importance to all the patches. This parameter should be chosen such to flatten the weights distribution.

3.2 Non-local graph

The first step of our method is to construct a graph representation which models the relationship between the patches extracted from an image. A weighted graph $G = \{E, V, w\}$ consists of a set of vertices V, a set of edges E, and a weight function $w : E \to R^+$ which assigns a positive weight to each edge. Each pixel $\mathbf{x}$ of the image is represented by a vertex u whose coordinates in the non-local space are given by the patch $R(\mathbf{x})$. Each pair of vertices u, v is connected by an edge $e = (u, v)$ whose weight is obtained by computing the similarity function $w(\mathbf{x}, \mathbf{y}) := w(u, v)$. This weight is positive and tends to zero when patches are highly dissimilar. In practice we only connect the K nearest neighbors (KNN)[5] to reduce the number of edges,

[5]Another option is to set weights that are smaller to some ϵ to zero.

thus the complexity of further filtering. As this approximation could potentially create a directed graph, we further symmetrize the weight matrix: $W = \frac{W+W^T}{2}$.

The generated graph is termed non-local as it encodes the non-local relationship between patches. The similar patches should be well connected together while weakly connected to the others, effectively forming clusters of patches belonging to a particular texture. The patches located at the transition between textures are ideally mildly connected to both clusters.

There will be three kind of signals on this graph:

1. The pixel position, or coordinates, $\mathbf{x} \in \mathbb{R}^2$.
2. The value $I(\mathbf{x}) \in \mathbb{R}$ of the pixel located at $\mathbf{x}$.
3. The pixel values $R(\mathbf{x}) \in \mathbb{R}^d$ of a patch centered at $\mathbf{x}$.
4. The priorities $P_{data}(i) \in \mathbb{R}$, $P_{confidence}(i) \in \mathbb{R}$ and $P(i) \in \mathbb{R}$.

To favor connections to closer patches we further introduce the center pixel location $\mathbf{x}$ into the patch $R(\mathbf{x})$, effectively introducing some locality into the graph construction:

$$R_\alpha(\mathbf{x}) = (R(\mathbf{x}), \alpha\mathbf{x})^T = (I(\mathbf{x}_1), ..., I(\mathbf{x}_d), \alpha\mathbf{x})^T \in \mathbb{R}^{d+2}. \tag{4}$$

Weights who include both distance and similarity measures between patches may be obtained from Eq. (3) with this updated patch definition. The parametric amount of local information will favor the emergence of grid-like connections on large uniform texture rather than random-like connections that would otherwise arise from the KNN approximation.

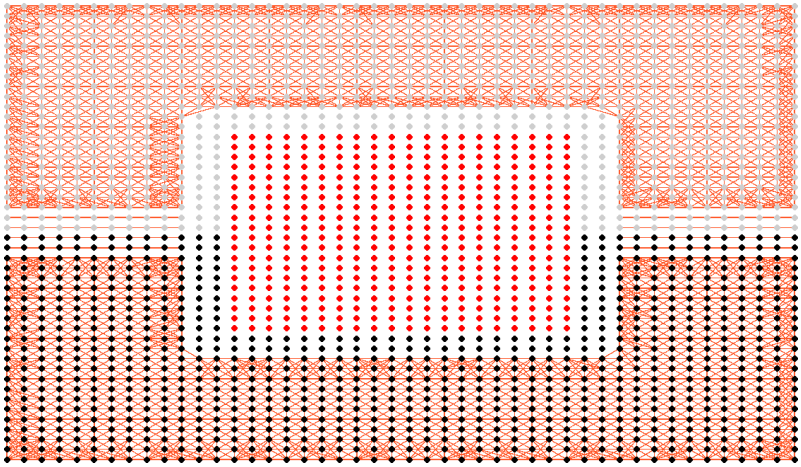


Figure 1: Example of the connections of a non-local patch graph.

Finally, vertices representing patches who contain unknown pixels are not connected to any other vertex during the construction of the graph. This graph is initially disconnected and will ultimately be fully connected as these vertices are sequentially inserted.

Fig. 1 shows the graph connections of an example binary image. Each dot represents a pixel: there are black and white pixels. The red dots mark unknown pixels, i.e. pixels of the target area ω. Note that all vertices i which patch $R(\mathbf{x}(i))$ contains an unknown value are disconnected (patch size $d = 25$). Patches composed solely of white or black pixels are tightly connected, forming two distinct texture clusters. Connections at the boundary between those textures form "lines" as similar patches are arranged horizontally.

3.3 Node insertion

All the unconnected nodes which represent unknown patches (the red dots in Fig. 1) are candidates for insertion into the graph. In the end, all the nodes will be connected. To compute the weights of a node we need some information about the represented patch. We thus select for insertion the patches who at least a fraction L of their area is known, i.e. at least a fraction L of the pixels they represent have a value. We then construct a comparison mask $M \in \{0,1\}^d$ where ones represent known pixels and zeros unknown pixels. Note that $\|M\|_1 \geq L$. Each selected patch $\mathbf{x}$ is compared against all known and connected patches $\mathbf{y}$ (the black and white dots in Fig. 1) by a masked squared Euclidean distance

$$d(R(\mathbf{x}), R(\mathbf{y})) = \|MR(\mathbf{x}) - R(\mathbf{y})\|_2^2 = \sum_{i=1}^{d} \left(M_i R(\mathbf{x})_i - R(\mathbf{y})_i\right)^2 . \tag{5}$$

The selected nodes are then connected to the K closest known nodes. Weights are given by Eq. (3).

3.4 Structure detector

The goal of the structure detector is to discriminate a pixel who is part of a linear structure from a pixel who is part of a texture. The idea is to observe how the vertex u representing the pixel i is connected. If the set $\{E_u | w(u,v) \geq 0\}$ of edges are spread in all directions around the vertex, chances are that the pixel is involved in a texture. If in contrary the vertex connections form a line, the pixel is most probably part of some structure.

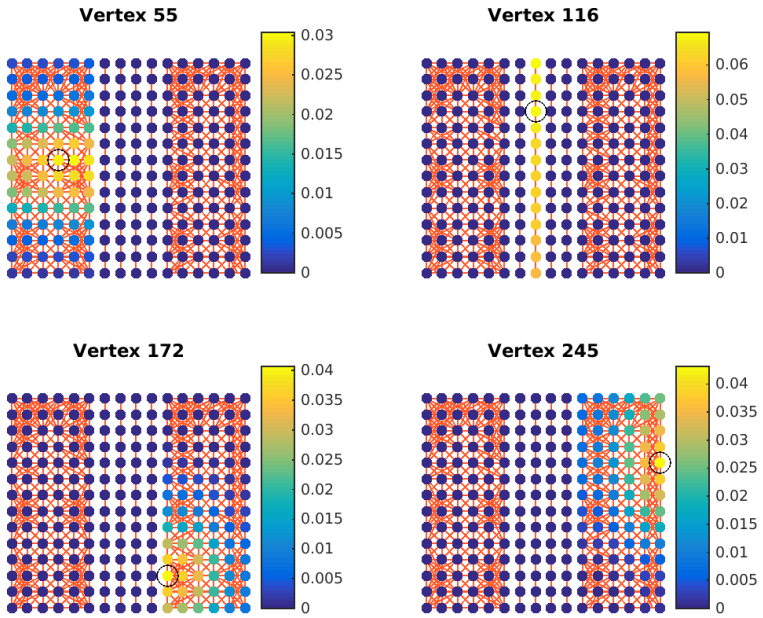


Figure 2: $T_i g$ of four vertices on a binary image with a vertical edge.

To observe the connection of the vertex u we low-pass filter a signal $s_i(j) = \delta_{ij}$, where δ_{ij} denotes the Kronecker delta. The filter is defined by the heat kernel

$$g(x) = e^{-\tau \frac{x}{\lambda_{\max}}} \tag{6}$$

where τ is a scaling factor and $\lambda_{\max}$ denotes the largest Laplacian eigenvalue. The filtered signal $T_i g$, where T_i is a generalized translation operator applied to the filter g, is a representation of the energy diffusion. Fig. 2 shows how the Kronecker delta energy spreads over a simple image.

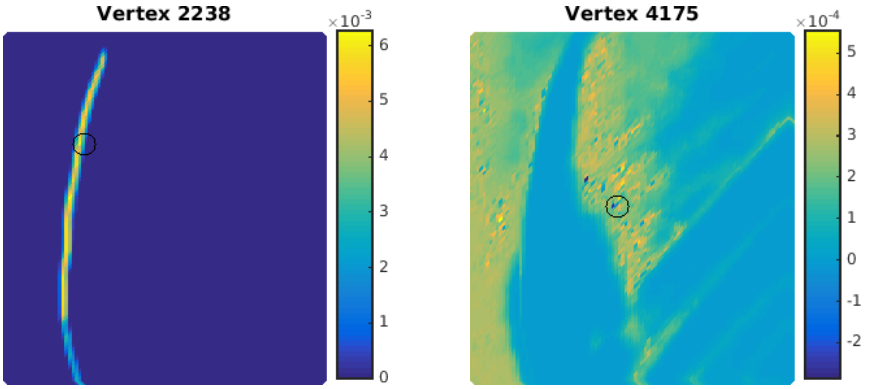
A way to assess the type of connections is to measure the filtered signal sparsity

$$\sigma(i) = \frac{\|T_i g\|_1}{\|T_i g\|_2} = \frac{C}{\|T_i g\|_2} \tag{7}$$

where C is a constant equal to one[6]. A sparse signal means that the energy has diffused along a well-defined direction which is due to strong edges between vertices along the structure direction (see Fig. 3a). A non-sparse signal means that the energy has spread all over the area which is due to the tight network of weak edges in a texture area (see Fig. 3b).

We should synthesize linear structures first, realizing the "Connectivity Principle". The (eventual) texture boundary is the most constrained part of the target region,

[6] $\|T_i g\|_1$ is a measure of the amount of energy. As $\|\delta_{ij}\|_1 = 1$ and heat diffusion only displaces energy, the energy of the filtered signal is still one.



(a) $T_i g$ of a pixel i which is part of a structure.

(b) $T_i g$ of a pixel i which is part of a texture.

Figure 3: Structure detector examples.

it should be filled first. To give higher priority to linear structures we define the data priority term as

$$P_{data}(i) = \frac{1}{\sigma(i)} = \|T_i g\|_2. \tag{8}$$

Fig. 4 shows the value of this priority over all the pixels of a natural image and the influence of the λ parameter.

3.5 Patch priority

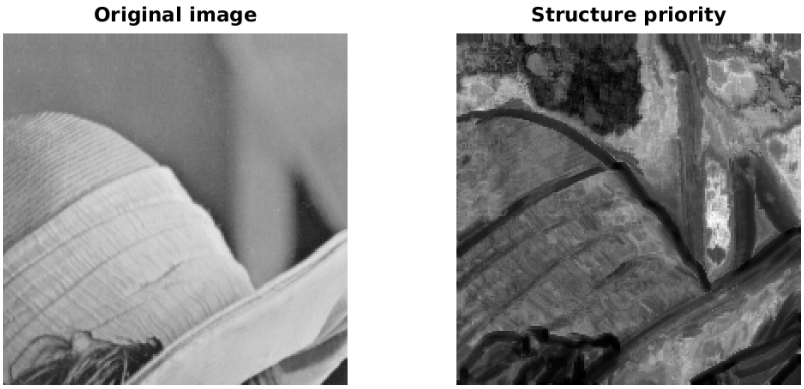
Our algorithm performs the synthesis task through a best-first filling strategy that depends entirely on the priority values that are assigned to each patch on the fill front.

While the most constrained part should be prioritized, a patch which half the pixels are known is more likely to be correctly identified that one with only one known pixel. The known pixels of a patch on the border of the fill region are certainly more reliable that the hallucinated pixels in the middle of the fill region. In order to fill first the parts we are most confident about, we should take into account the confidence of the information we have about a patch.

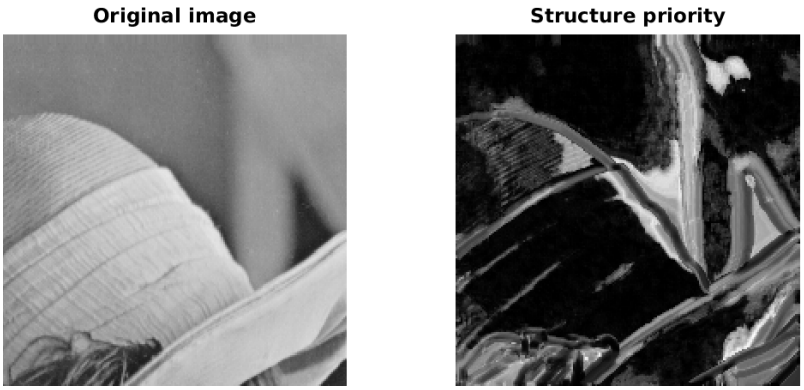
Our confidence measure is defined as

$$P_{confidence}(\Psi_{\mathbf{x}}) = \frac{\sum\limits_{\mathbf{y}\in\Psi_{\mathbf{x}}} P_{confidence}(\mathbf{y})}{|\Psi_{\mathbf{x}}|} \tag{9}$$

where $\Psi_{\mathbf{x}}$ is the patch centered at the point $\mathbf{x}$ and $|\Psi_{\mathbf{x}}|$ its area. During initialization, the function $P_{confidence}(\mathbf{x})$ is set to $P_{confidence}(\mathbf{x}) = 1\ \forall \mathbf{x} \in \Phi$ and $P_{confidence}(\mathbf{x}) =$



(a) Heat kernel width $\lambda = 100$.



(b) Heat kernel width $\lambda = 500$.

Figure 4: $\|T_i g\|_2$ of all the pixels of a natural image.

$0\ \forall \mathbf{x} \in \Omega$. When the patch Φ_x is filled, the pixel confidences are updated by

$$P^{t+1}_{confidence}(\mathbf{y}) = \begin{cases} P_{confidence}(\Psi_{\mathbf{x}}) & \text{for updated pixels} \\ P^{t}_{confidence}(\mathbf{y}) & \text{for untouched pixels} \end{cases} \quad \forall \mathbf{y} \in \Psi_x. \tag{10}$$

Note that depending on the chosen information propagation scheme (see section 3.6), not all pixels of a patch are updated when the patch is filled. As filling proceeds, confidence values decay, indicating that we are less sure of the colour values of pixels near the center of the target region.

The fill priority of the patch i, which is biased toward patches which are on the continuation of strong edges (data term) and which are surrounded by reliable pixels (confidence term), is given by

$$P(i) = P_{data}(i) \cdot P_{confidence}(i). \tag{11}$$

Note that if we drop the structure-based priority $P_{data}(i)$, we default to the "onion peel" strategy which consist of filling the missing region from the edge inward.

3.6 Information propagation

After computation of the priorities, the highest priority patch $\Psi_{\hat{\mathbf{x}}}$ is selected and filled with data extracted from the source region Φ.

To avoid the blur inevitably introduced by diffusion, we propagate information by direct sampling of the source region, i.e. we leverage exemplar-based synthesis. Using our graph representation, we search for that patch which is the most similar to $\Psi_{\hat{\mathbf{x}}}$:

$$R = R(\mathbf{y}) \mid \mathbf{y} = \arg\max_{\mathbf{y}} w(\hat{\mathbf{x}}, \mathbf{y}). \tag{12}$$

In the spirit of the NL-means algorithm presented in [2], we could produce a denoised example patch of the texture by a weighted average of the connected, thus similar, patches:

$$R = \sum_{\mathbf{y}} w(\hat{\mathbf{x}}, \mathbf{y}) \cdot R(\mathbf{y}). \tag{13}$$

We however observed that the non-local averaging does introduce some blur.

Having found the source exemplar $\Psi_{\hat{\mathbf{y}}}$, we update the pixels of the target $\Psi_{\hat{\mathbf{x}}}$. We have here two options:

1. **Overwrite**: Overwrite all the pixels, even those who already have a color value. $R(\hat{\mathbf{x}}) = R$.

2. **Preserve**: Only give a value to unknown pixels. Preserve the values of the pixels that were already hallucinated. $R(\hat{\mathbf{x}}) = M \cdot R + \overline{M} \cdot R(\hat{\mathbf{x}})$ where M is a mask which indicates unknown pixels by ones.

This is a parameter of the algorithm. From our experience, the overwrite mode often gives better visual results.

This suffices to achieve the propagation of both structure and texture information from the source Φ to the target region Ω, one patch at a time.

3.7 Global optimization

An optional final step would be a global optimization which minimizes the TV or Thikonov norm (the prior term) under the constraint of resemblance to the observed image (the data term). This is the idea of [6], which interleaved this global optimization step with a graph construction step. We however observed that any further manipulation of the pixel values that does not explicitly depend upon statistics of

the source region is more likely to degrade visual similarity between the filled region and the source region, than to improve it.

3.8 Algorithm

To resume, our algorithm is composed of the following steps:

1. Construct the non-local patch graph from the entirely known patches Λ. The unknown patches Υ are ignored, their vertices are disconnected.
2. Loop until all unknown patches are processed:
   (a) Select the patches from the set Υ who contain enough information to be matched.
   (b) Compare those patches to all known patches. Select the K strongest $e(u, v)$ and insert the new node in the graph.
   (c) Update the priorities P_{data} and $P_{confidence}$.
   (d) Select the highest priority patch $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} P(\mathbf{x}) = \arg\max P_{data}(\mathbf{x}) \cdot P_{confidence}(\mathbf{x})$.
   (e) Update the pixel values of the selected patch $\hat{\mathbf{x}}$ using one the presented scheme.
3. Optionally perform a global optimization.

4 Results

The first result we present is the assumption our method makes about the non-local patch graph representation. We show that it indeed possesses enough information about the image, which was an objective of this work. We then show some reconstructions performed by our algorithm and discuss the effects of parameters and the reasons of failures. We finally expose a quality measure of the reconstructed graph.

4.1 Assumption validation

We first wanted to verify the assumption that a non-local patch graph indeed possess enough information about the image to allow a convincing reconstruction. While [6] already showed that non-local patch graphs indeed possess the necessary information, we wanted to try it on our own with our tools. For this purpose, we generated a patch graph from the original image. We then masked some part of the image without disconnecting any vertices. We then tried to inpaint the missing region by minimizing the norm (the prior term) of the pixel signal under the constraint of resemblance to the known parts of the image (the data term). See [6] for further details. Energy will flow from the known patches toward the unknown ones by diffusion, effectively solving the heat equation. Fig. 5 shows an example of such a reconstruction, which indeed demonstrate that the graph contain enough information to convincingly hallucinate the missing pixels. We are obviously not going to be able to generate this graph in a real setting. The problem is to reconstruct a graph as close as possible as the graph derived from the original image.

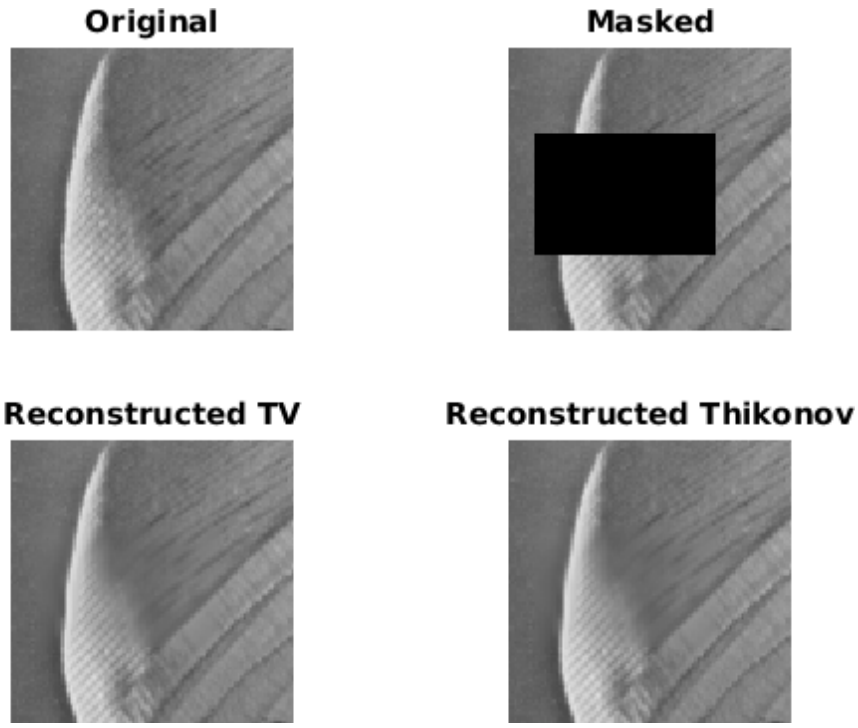
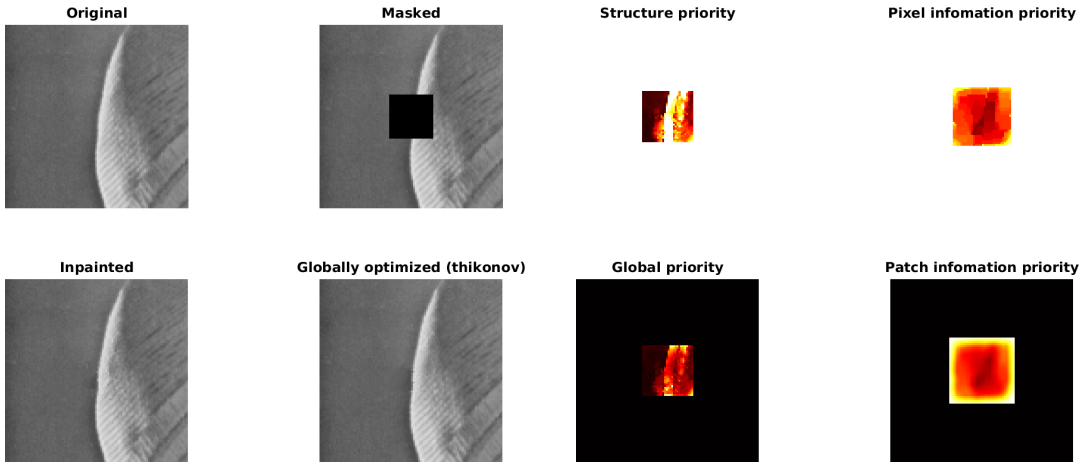


Figure 5: Inpainting using a graph constructed from the original image.

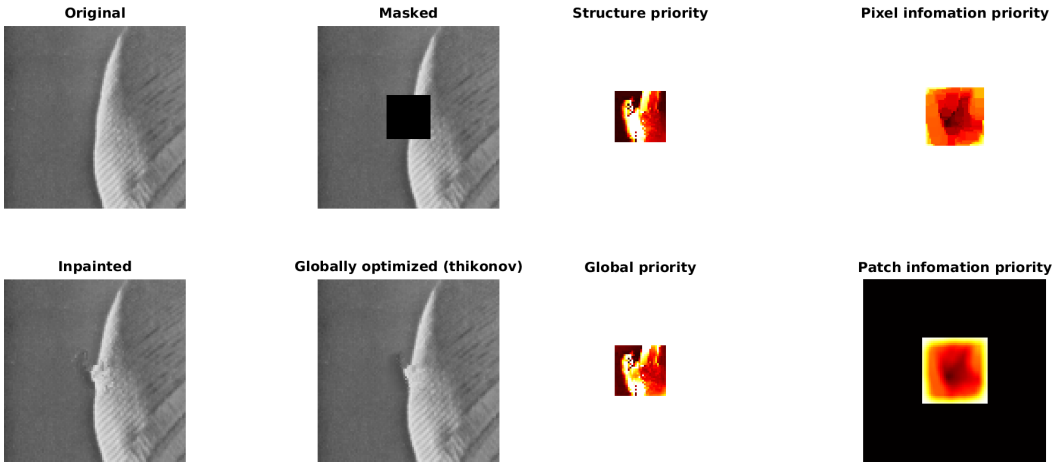
4.2 Inpainting

Fig. 6 shows a good example of inpainting with its associated priorities while Fig. 7 shows a bad one. A comparison of the hallucinated area with the shape of the structure priority suggests that the fill order has a great impact on the quality of the reconstructed image. While we only show a single example here, we made several such observations during experimentations with parameter values. More experiences should be conducted to determine the optimal set of parameters and their impact.



(a) Images. (b) Priorities.

Figure 6: Good example.



(a) Images. (b) Priorities.

Figure 7: Good example.

4.3 Graph quality

A useful indication of the performance of our reconstruction algorithm would be a measure of the reconstructed graph quality. An idea is to compare this graph with the one derived from the original image and some other graphs. The regularity of the graph against the image signal may be such a comparison measure, given by

$$xLx^T = \|\nabla f\|_2^2 \tag{14}$$

where L is the graph laplacian and x the signal. Smaller the regularity, better the graph represents the signal. The regularity difference between the reconstructed graph and the original graph would then be an indication of the performance of our method.

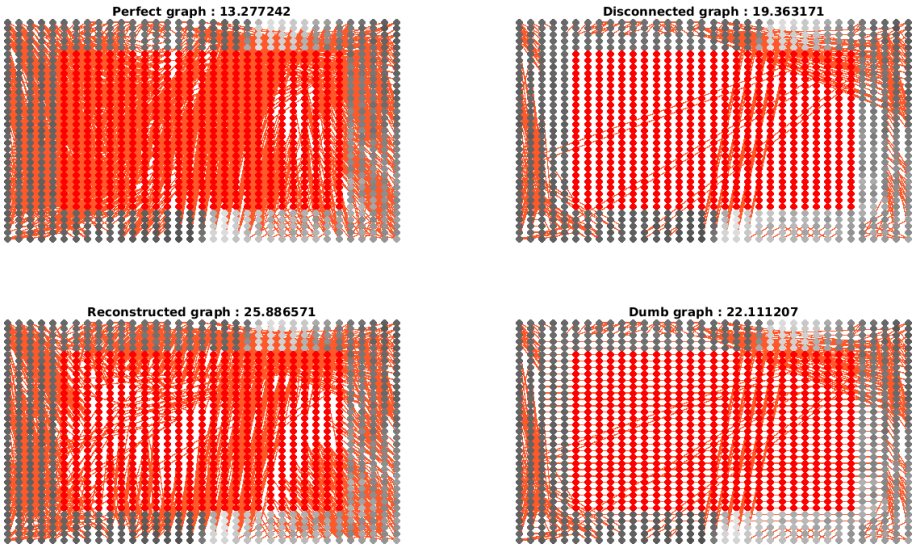


Figure 8: Regularity measure of some graphs against the original image signal.

Fig. 8 compares the regularity of four graphs :

1. **The perfect graph** : constructed with the complete original image.
2. **The disconnected graph** : constructed by leaving the unknown vertices alone.
3. **The reconstructed graph** : constructed during the inpainting process.
4. **A dumb graph** : unknown vertices are connected by a grid.

As expected, the perfect graph is the most regular of the four. Surprisingly, the disconnected and dumb graphs are more regular than the reconstructed graph. This would indicate that our reconstruction scheme does not produce an ideal graph.

This measure however needs to be put into perspective as the regularity depends on the number and weights of the connections. Some normalization term is thus needed for a fair comparison. Such a normalization is still under investigation.

Note also that the relative importance α (see Eq. (4)) of locality in the graph construction has a great impact on the regularity. From our experience, the regularity measure prefers graphs who are more local. The lowest regularity (on the perfect graph) was achieved with $\alpha = 0.1$. If the graph is too local, the regularity increases because there is connections across edges. If it is too non-local, the regularity increases for no apparent reason.

5 Discussion

Some inherent limitations of our algorithm are:

1. The synthesis of regions for which similar patches do not exist will obviously be problematic.

2. The algorithm does not handle depth ambiguities (i.e., what is in front of what in the occluded area?).

As opposed to the work presented in [4], our structure detector is not inherently limited to linear structures. It can handle curved structures.

A principal component analysis on a typical sets of patches showed that 92% of the variance is contained in the first two components (depicted by Fig. 9) which suggests that the data resides on a lower dimensional space. The wavelet and DCT transforms should be investigated as a sparse patch representation which would enable an embedding of the graph in lower dimension. Laplacian Eigenmaps should also be considered as a non-linear dimensionality reduction tool as it would preserve the similarity between patches by minimizing the regularity. It ensures that close points on the original manifold stay close.

The graph weights are actually given by a measure of similarity between patches. A superior method may be to use the normalized cross-correlation (NCC) which would measure the correlation between patches, corrected for brightness. We may finally use graph features.

Before further improvements, we should however compare our technique to competing algorithms using classical images.

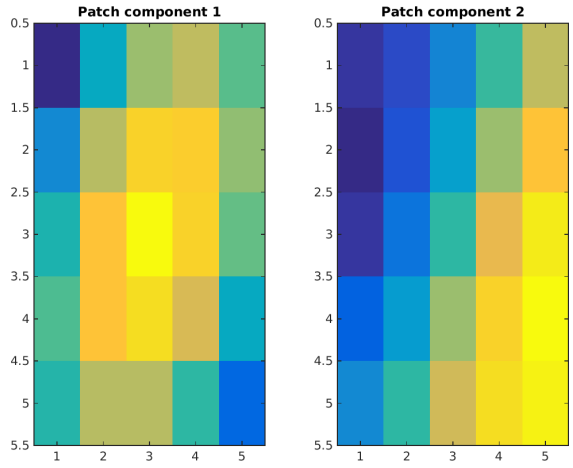


Figure 9: The two principal components of patches.

6 Conclusion

This report presented and analyzed a new technique for image inpainting along with a novel approach to structure detection. The algorithm is an exemplar-based texture synthesis technique which uses the image structure to determine the fill order, leveraging the non-local patch graph representation of an image. The technique is able to propagate both linear structure and two-dimensional texture into the target region. The result is a plausible hallucination of the missing pixels that mimics the appearance of the source region. While there is a lot more to experiment, we showed that a non-local patch graph is an efficient representation of images for the inpainting problem. We further showed that bad results are linked to bad estimation of the structure priority, which reinforces the idea that the reconstruction quality is highly dependent on the order in which the filling proceeds. As the result heavily depends on the ability to extract meaningful structure, the structure detector, a major contribution of this work, has a great role to play.

References

[1] Mikhail Belkin and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396.

[2] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. "A review of image denoising algorithms, with a new one". In: *Multiscale Modeling & Simulation* 4.2 (2005), pp. 490–530.

[3] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. "Object removal by exemplar-based inpainting". In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2003, pp. II–721.

[4] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting". In: *Image Processing, IEEE Transactions on* 13.9 (2004), pp. 1200–1212.

[5] Emmanuel d'Angelo. "Patch-based methods for variational image processing problems". PhD thesis. École polytechnique fédérale de Lausanne, 2013.

[6] Emmanuel d'Angelo and Pierre Vandergheynst. "Towards unifying diffusion and exemplar-based inpainting". In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE. 2010, pp. 417–420.

[7] David I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *Signal Processing Magazine, IEEE* 30.3 (2013), pp. 83–98.