Telecom Paris

Report for project IMA 201

In-painting of images

Supervised by:

Mr. Alasdair James Newson

Written by :

Saifeddine Barkia
Taher Romdhane

2020/2021

Contents

List of Figures

1 Introduction

Image in-painting refers to the method of filling the missing details in the specified region of the visual input. The purpose of the operation is to recreate the missing parts of the affected image in such a way that the unpainted region can not be identified by a casual observer. Applications vary from reversing impairments such as holes, and scratches in scanned photos and digitized artwork images, to removing/introducing image artifacts such as icons, stamped dates, writing, individuals, and special effects on that image.

Image painting is an ancient art that was painful since it required human supervision to do the work by hand. Today, however, researchers have suggested a variety of automated painting methods. In addition to the graphic, most of these methods often involve a mask showing the regions that require painting as input.

Many approaches, including sequential algorithms or deep learning techniques, have been suggested for this reason. To this end, we categorize current strategies for painting pictures into three categories: sequential-based approaches, CNN-based approaches, and GAN-based approaches. Sequential-based methods are the proposed strategy without deep learning using neural networks. Where CNN-based methods are algorithms that use neural networks and automated deep learning. GAN-based techniques are the strategies used by Generative Adversarial Networks (GANs) to train representations of painting models. As a first step, and during our project, we will focus on the sequential-based approaches. Actually, there are mainly two approaches to tackle this problem:

- Texture synthesis techniques: which is the process of repeating the textual pattern of a picture stochastically. This technique is cheap and easy to implement but it has difficulties in filling the boundaries and propagating the linear structures and some composite textures.
- Diffusion techniques: Diffusion-based strategies fill the missing region (the "hole") by efficiently propagating the content of the picture from the border to the interior of the missing region. This method is based on the heat flow equation in physics. However, it creates a noticeable blur especially when we are going to fill large regions.

The algorithm presented in the paper that we have implemented combines the strength of both these approaches because most regions of the image consist of texture and structure, and the border between the image regions includes a significant amount of structural details.

2 Algorithm presentation

In this section, we are going to present and explain the algorithm implemented in the paper. For ease of explanation, we are going to adapt the notations presented in the literature (as well as in this paper).
As we mentioned earlier, this paper is based on exemplar-based approaches and it will handle the problem of propagation of the linear structures efficiently without treating the problem of isophotes apart. The region to be filled is denoted The region to be filled, i.e., the target region is indicated by Ω, and its contour is denoted $\partial\Omega$. the contour evolves inward as the algorithm progresses, and so we also refer to it as the "fill front" . The source region, Φ, which remains fixed throughout the algorithm, provides samples used in the filling process. We will also denote by Ψ_p the patch centered at pixel p

2.1 One iteration of the algorithm

For that purpose we are going to consider this photo and we will see what is going to happen at each step.

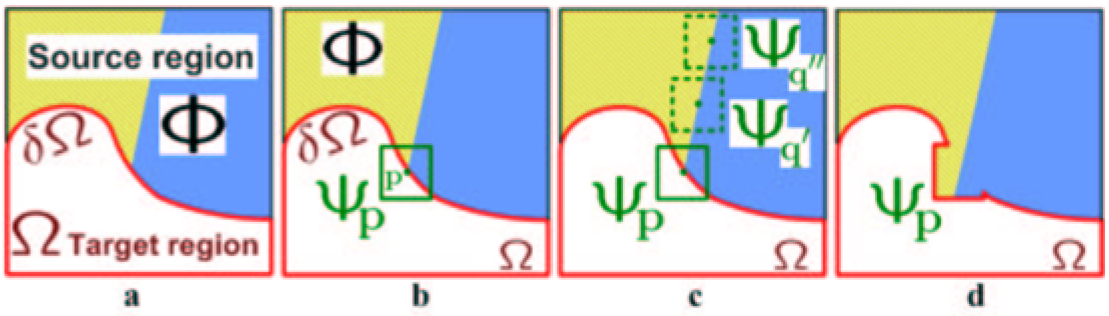


Figure 1: Structure propagation by exemplar-based texture synthesis.

a. Original image, with the target region Ω, its contour $\partial\Omega$ and the source region Φ clearly marked.

b. We want to synthesize the area delimited by the patch Ψ_p centered on the point p $\in \partial\Omega$

c. The most likely candidate matches for Ψ_p lie along the boundary between the two textures in the source region, e.g., $\Psi_{p\prime}$ and $\Psi_{p\prime\prime}$.

d. The best matching patch in the candidates set has been copied into the position occupied by Ψ_p, thus achieving partial filling of Ω. The target region has, now, shrank and its front has assumed a different shape.

Suppose that the square template Ψ_p is centered at point p (b) is to be filled in. The best-match sample from the source region originates from the patch $\Psi_q \in \Phi$, which is more close to those sections that are already filled in Ψ_p . In figure (b), if Ψ_p lies on the boundary of the image edge, the best matches would lie along the same edge $\Psi_{p\prime}$ or $\Psi_{p\prime\prime}$. All that is needed to spread the isophote inward is a simple transfer of the pattern from the best-match source patch (d).

2.2 Details of the algorithm.

After that, the user has chosen the target value that he wants to remove and after choosing also the size of the window (patch size) that has to be larger than the largest distinguishable texture element in the source region, the process of filling the target region is automatic.
Actually, during the filling process, the patches along the Ω are given a ***priority value*** that will determine the order in which they are filled.

2.2.1 Patch Priority

The main idea of the algorithm is to use exemplar-based texture synthesis for both replicating the textures that exist in the picture and propagating the linear structure present in the picture. To do so, the algorithms rely completely on the patch priority assigned along the ' fill front' $\partial\Omega$.

Given a patch Ψ_p centered at the point p for some $p \in \partial\Omega$, the priority $P(p)$ of this patch is defined as :

$$\boxed{P(p) = C(p) * D(p)}$$

Where the ***C(p)*** is the confidence value of the pixel $p \in \partial\Omega$.

$$C(p) = \frac{\sum C(q)}{|\Psi_p|} \qquad \forall q \in \Psi_p \cap \Phi$$

This term indicates the measure of ***reliable information*** that is present around the pixel p.

At the beginning ,$C(p)$ is set to $C(p) = 0\ \ \forall p \in \Omega$, and $C(p) = 1 \forall p \in \Phi$. This term indicates that the patches that have already more pixel filled are the ones that we are going to fill first. This automatically gives a certain preference to certain shapes and structures. For example, patches that have corners are prioritized by this term because they are surrounded by more pixels from the original image. So, pixels are being filled from the outside layers to the inside layers thanks to this term.

And ***D(p)*** is the data term having the following formula

$$D(p) = \frac{|\nabla I_p \perp . n_p|}{\alpha}$$

D(p) is the data term that indicates the strength of the isophotes that are present in this patch. This term is fundamental in this algorithm because it replaces the diffusion approach study that has to be made to propagate efficiently linear structures. It promotes the synthesis of linear structures first.

So, we have a sort of balance between the data term D(p) that pushes the isophotes inside the target region and the confidence term C(p) that tends to make the propagation of the information as precise as possible.

2.2.2 Propagating texture

When all the priority fill front are computed, the most prioritizing patch $\Psi_{\widehat{p}}$ is found and we will fill it with data derived from the source region Φ . Actually, we search in Φ for the patch who have is the most similar to our target patch $\Psi_{\widehat{p}}$. In other words, we are going to compute the sum of squared differences (SSD) of the already filled pixels in all the candidate patches and the target patch and we will take the candidate patch that minimizes this distance. The formula is the following :

$$\Psi_{\widehat{q}} = \underset{\Psi_q \in \Phi}{\operatorname{argmin}}\, d(\Psi_{\widehat{p}}, \Psi_q)$$

Where d is the sum of squared differences (SSD) :

$$d(y, x) = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} (y_i - x_i)^2$$

By computing this, we are confident that both texture and structure information are being propagated from the source Φ to the target region Ω, one patch at a time.

2.2.3 Updating the confidence values

After each iteration (after each target patch that has been filled) , we will update the pixels that have been filled with the same confidence at the selected pixel with the highest priority in order to maintain the information that we have filled these pixels.

$$C(q) = C(\widehat{p}) \quad \forall q \in \Psi_{\widehat{p}} \cap \Omega$$

3 Implementation discussion

In this section, we will discuss the various problems that we have encountered during the implementation of the algorithm and the solutions that we have adopted to overcome them.

- When the mask is on the edge of the picture, we decided not to consider those patches at all (we could have applied the zero-padding method)but this solution is more coherent and more robust.
- Since it's a patch-based methods, we have spent a lot of time debugging and testing each function with special cases apart to create a robust implementation model.
- The data term is the most complicated to implement. We have considered various approaches to compute it but in the end, we think that we have found the most efficient one. First of all, to calculate the normal vector of the border of the mask , we considered at first applying erosion and dilatation to generate the border of the mask, and then we would have taken the 2 neighbors points to our selected pixel and then computed the normal on this line. The complexity of this approach lies in the choice of the neighbors of each pixel at each iteration since the border shape can take many forms and orientations and the choice will be hard to determine if the selected

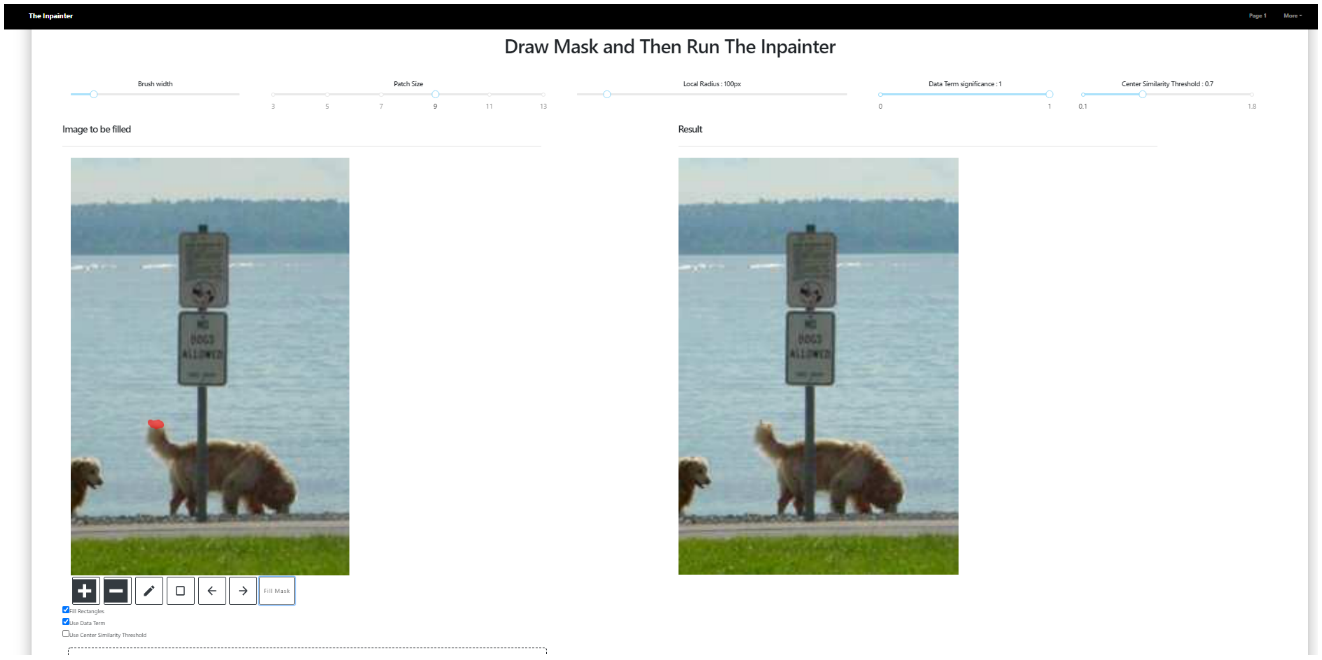
pixel has more than 2 neighbors. By doing that, we will lose a lot of information especially when the selected points are irrelevant. Given the structure of the mask, we have decided to compute its gradient at the selected point. In fact, the mask is composed of black and white pixels so the gradient at the border pixels will give us the direction of the transition from black to white since the gradient follows intensity. The vector obtained after normalization is then the same as the normal of this mask at this point.

- For the computation of the gradient for the selected pixel, since the gradient there is unknown, we decided to take the patch centered around the selected pixel. We replaced the undefined values inside the patch with 'None' and then we have computed the gradient for the remaining pixels in this patch (that are known because they are located in Φ). Then we took the maximum value of the gradient that exists in this particular patch.
- For big resolution images, we have noticed that the algorithm takes ages to run since it has to search for the candidate patch in all the source region Φ. So we have created an extra parameter (radius) to limit the search and so to speed up the execution of the algorithm. After all, most probably, the candidate patch is located around the selected pixel.
- Also, we have implemented the PCA approach in order to speed the execution of the algorithm by two times.

4 Dash framework

In order to enhance our work and to make it interactive, we decided to make an application where the user can select the image and the target region that he wants to remove manually. So we decided to use the dash framework for its simplicity and efficiency.

Actually, Dash is a productive Python framework for building web applications. It's created and maintained by Plotly. It is implemented on top of Flask,Poltly.js and React.js. It's an open source an environment where you can easily deploy your application. Here is a screenshot from our interactive plate-form:



Brush width

Patch Size: 3 5 7 9 11 13

Local Radius : 100px

Data Term significance : 1 (0 – 1)

Center Similarity Threshold : 0.7 (0.1 – 1.8)

- [✓] Fill Rectangles
- [✓] Use Data Term
- [ ] Use Center Similarity Threshold

Figure 2: Dashboard that we have implemented.

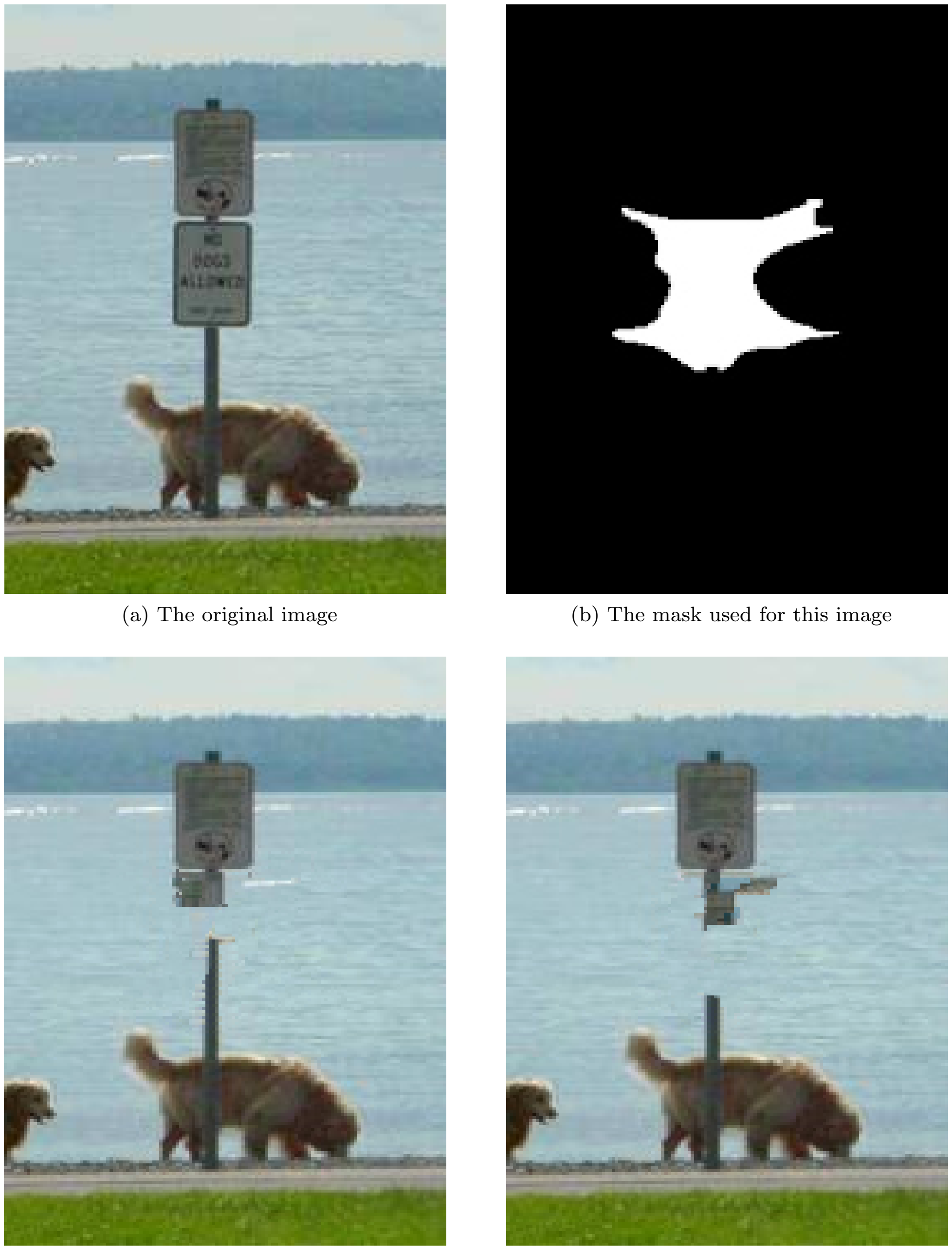
After choosing the image that he wants to modify, the user have the possibility to choose the size of the window(patch size), the radius of local area where to search and the width of the brush with which he will paint the mask.Also, the user can choose either or not he wants to use the data term.

5 Results and discussions

In this section we will show the results obtained by the algorithm and discuss its results.

We would be changing as parameters , the patch size , the use or not of the window and the radius of search in the source image.

For the first 3 images, we tried to compare the results that we have got, with the ones obtained in the paper. For that reason, we tried our best to imitate the mask that they have used so that the comparison would be fair also we can't tell the size of the patch size that they used so each time we try our best to guess it.



(a) The original image

(b) The mask used for this image

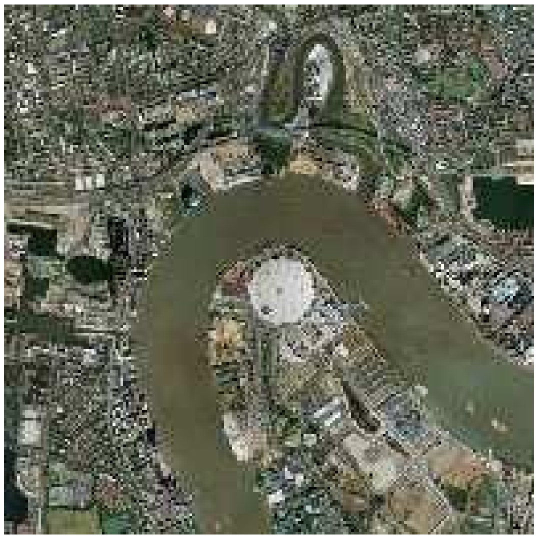
(c) The result using the data term

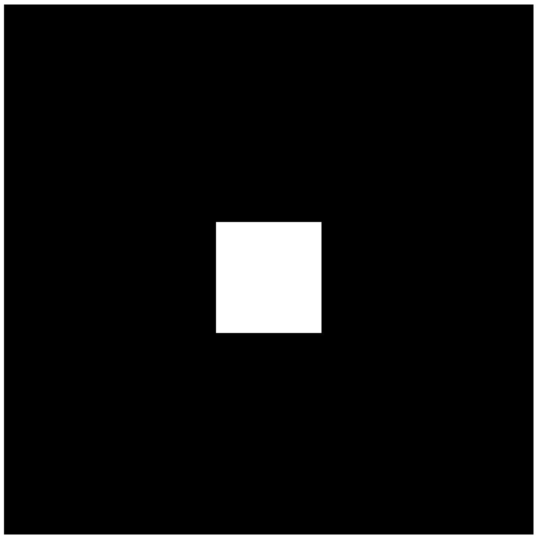
(d) The result without using the data term image

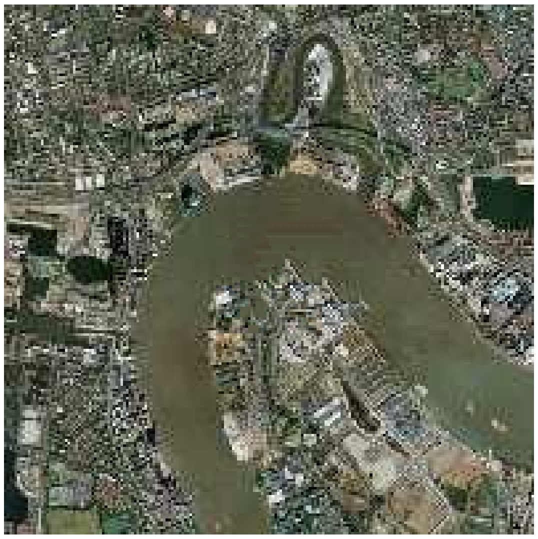
Figure 3: Results for image n2

We have used the a window of 9x9 because we believe it's the size of the lowest distinguishable detail.

As we can see, using the data term here also gave us better results. At a certain level, we have propagated the line that exists in the image. But it failed to go further more and that's the limitation of the algorithm. In fact, in a certain moment, the algorithm considered that another pixel has more priority to be filled first. So it ended the propagation of the line and filled the other pixel with patches from the water and the sign.


(a) The original image


(b) The mask used for this image


(c) The result using the data term

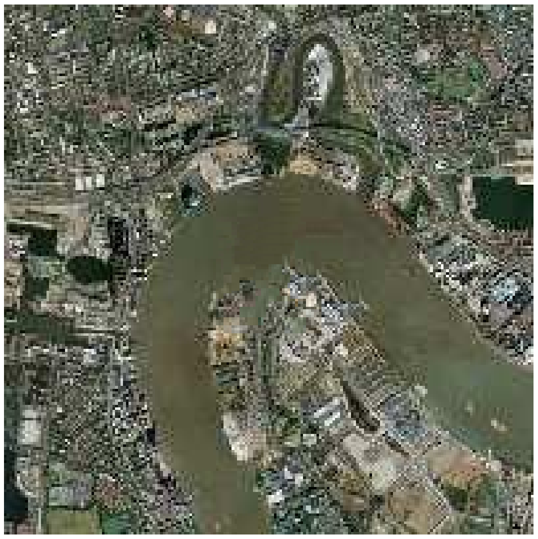
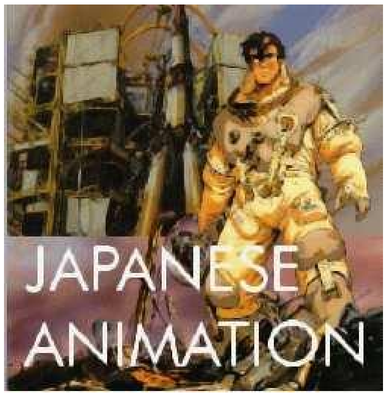

(d) The result without using the data term image

Figure 4: Results for image n3

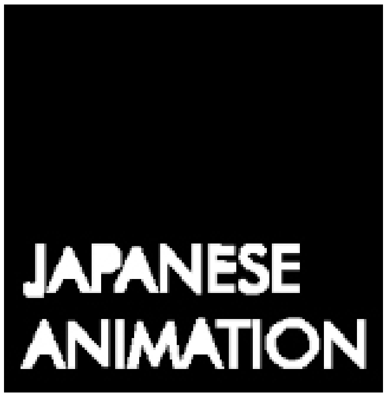
We have used the a window of 7x7 because we believe it's the size of the lowest distinguishable detail.

The results are good, but as we can see in the middle, when we have used the data term, they are much more better. As we can see, the reconstruction of the land is compact when we used the data term(this case is more realistic) unlike when we ignored it, there is water inside it.
the results that we have obtained are very similar to the ones they obtained in the paper despite not

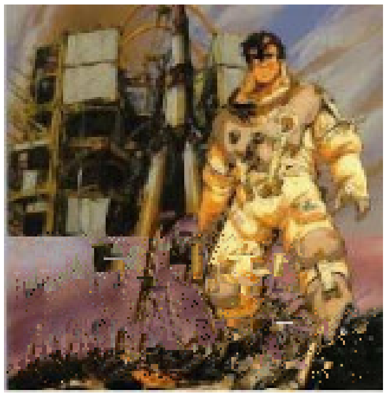
using the exact mask.

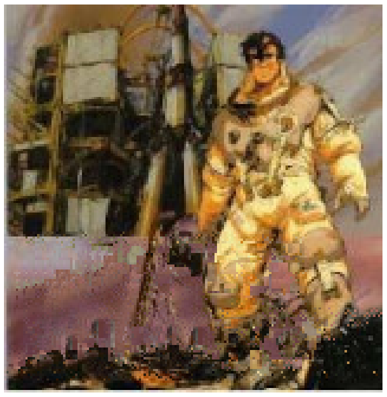


(a) The original image



(b) The mask used for this image


(c) The result using the data term and patch size of 9


(d) The result without using the data term image

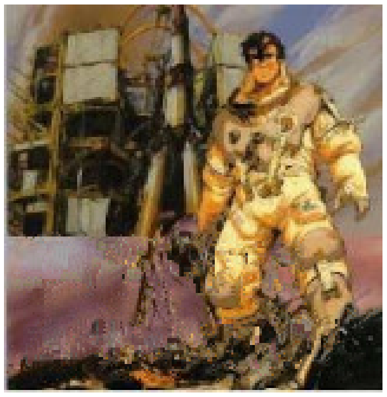
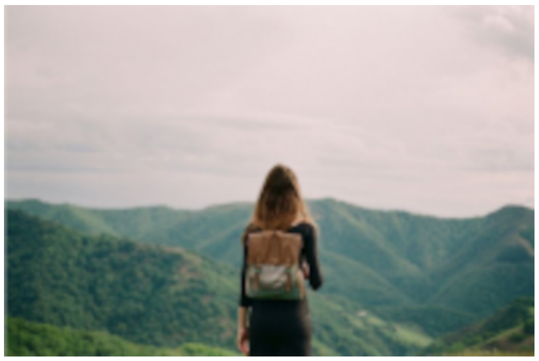
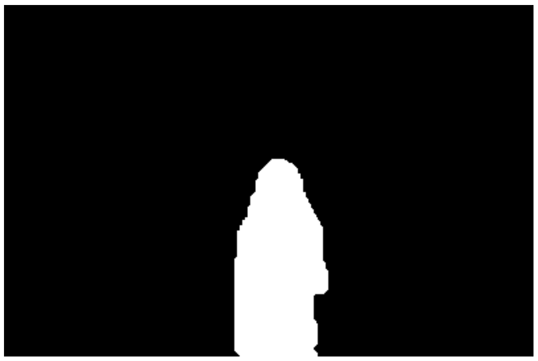

(e) The result using the data term and patch size of 5

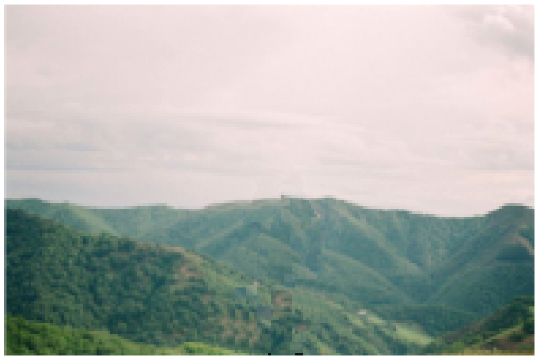
Figure 5: Results for image n4

The results obtained by applying this mask to this image are convenient. We have succeeded to eliminate the written words from the image.

We can clearly see in this example the effect of the patch size on the results. As we can clearly see, when we reduced the size of the window, the image is smoother and we reduced the noise that existed before.


(a) The original image


(b) The mask used for this image


(c) The result using the data term

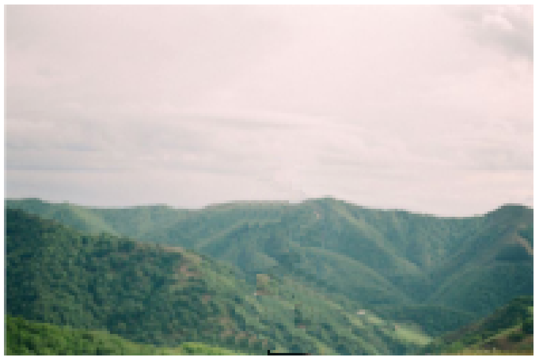
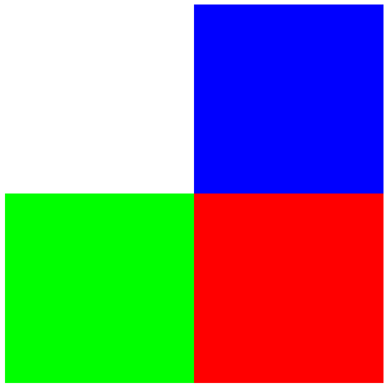

(d) The result without using the data term image

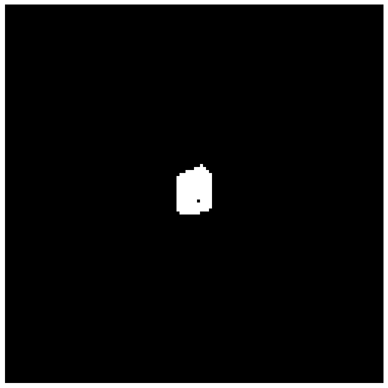
Figure 6: Results for image n1

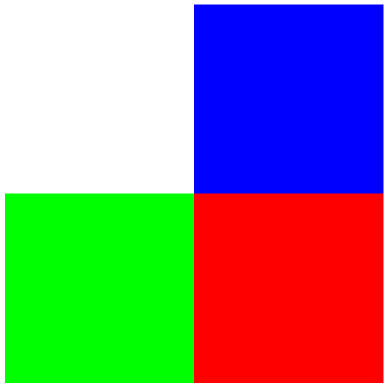
As we can see, we selected the person in the image and we wanted to remove it, as we can see, the algorithm fill the target region (the one that we wanted to remove smoothly with patches from the source image).

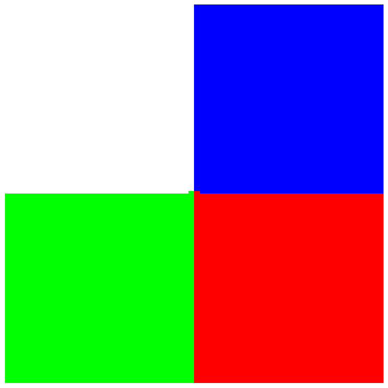
We have used the a window of 9x9 because we believe it's the size of the lowest distinguishable detail.

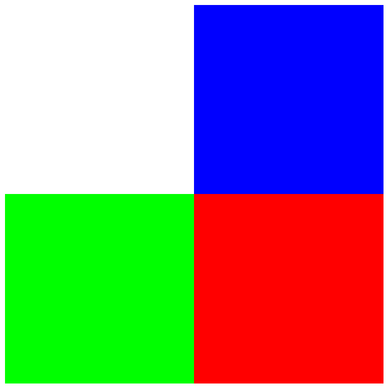
Let's compare the results with and without the data term. If we focus on the bottom of the image, we can clearly see that there is a propagation of the 'flower' structure when we used the Data term. The data term here replaced the diffusion approach. Add to that for this image, since that the patch that we are going to replace the image with are not far, we decided to limit the radius of our search for 50 pixels to accelerate the execution of our algorithm.


(a) The original image


(b) The mask used for this image


(c) The result using the data term and patch size of 9


(d) The result without using the data term image


(e) The result using the data term and patch size of 11

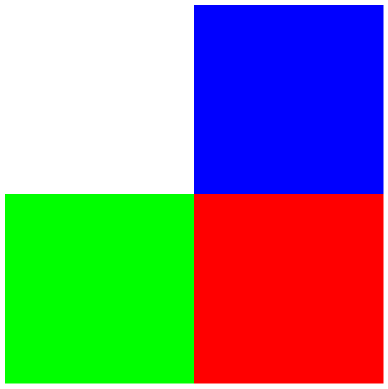
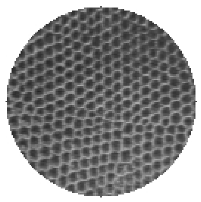
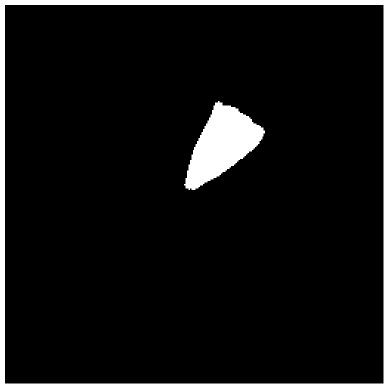

(f) The result without using the data term image and patch size of 11

Figure 7: Results for image n5

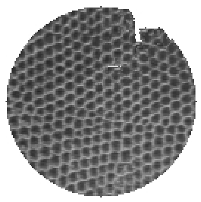
In this example we applied a mask in the centre of the image, masking all the colors at the same time and we will analyze the result. First of all, for the first 2 images, we used the patch size of 9. When we activated the data term we obtained exactly the same image as the original one. When we didn't apply the data term, we can see that the edges are not perfect and the colors interfered between each others. When we augmented the size of the patch, we obtained the same original image.



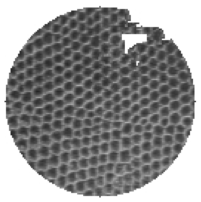
(a) The original image



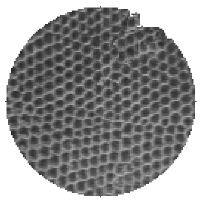
(b) The mask used for this image



(c) The result using the data term and patch size of 9



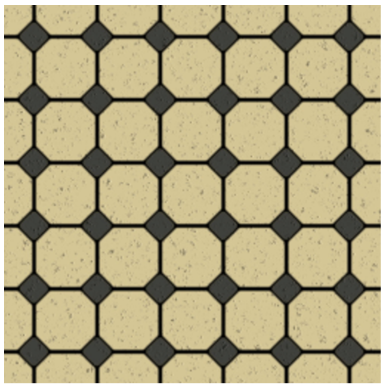
(d) The result without using the data term image nd patch size of 9

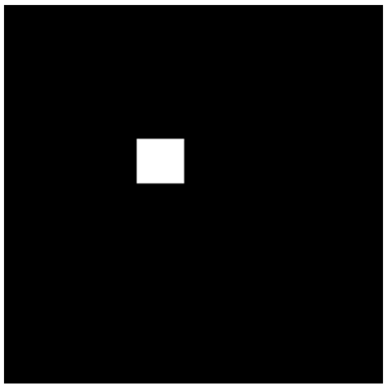


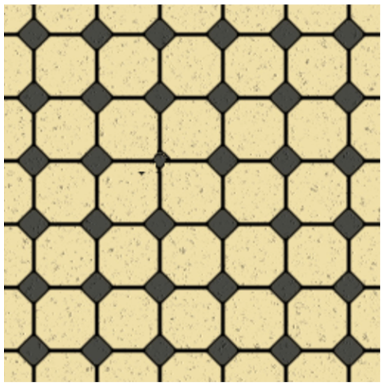
(e) The result using the data term and patch size of 5

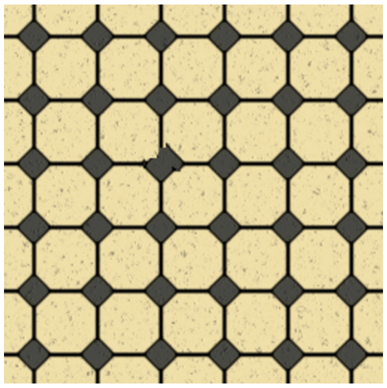
Figure 8: Results for image n6

In this example, we masked a piece of the circle.First of all, we used a window of size 9x9.when we didn't use the data term, the in-painting is not good at all. As we can see the recovered part was not compact and there is a blank in the middle. When we applied the data term, the result got better a lot better but we didn't get the curved shape of the circle at the end. When we used a window of 5*5 and we used the data term, we have almost recovered the curved shape of the circle. It's the limit of the algorithm.


(a) The original image


(b) The mask used for this image


(c) The result using the data term and patch size of 7


(d) The result without using the data term image nd patch size of 7

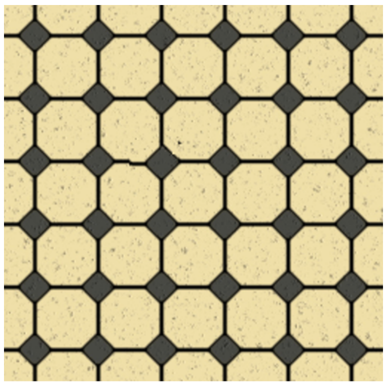

(e) The result using the data term and patch size of 13

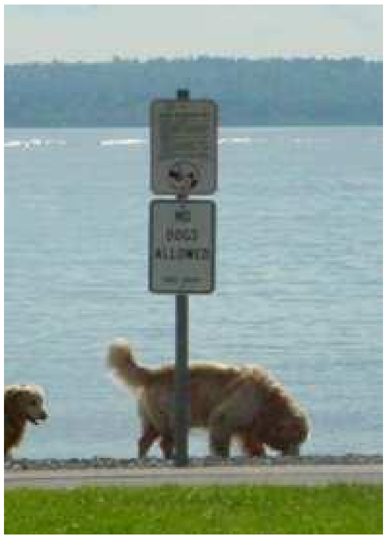
Figure 9: Results for image n7

In this example we masked a diamond of the repeated structure.At first, we used a patch size of 7 to treat this problem. As we can see, when we didn't use the data term the results are much more better than when we used this term. When we increased the size of the patch, we remark that the inpainting did improved and we barely see the difference between the result and the original image. Unlike the other results where the data term has helped in propagating the structure, in some cases not using this term get us better results.

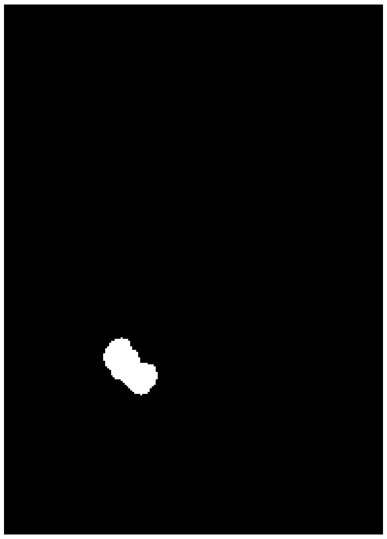
6 PCA and central pixel approaches

In order to fasten up the execution of the algorithm, we had the possibility either to write our code again in C++ because it's faster there or to learn new technique, like the PCA, for the search of patches instead of computing each time the minimum distance.
We found it more interesting to learn something new and we decided to go for the PCA approach. We read the documentation about it and we started computing the code but we faced a problem in finding the best patch since the patches that we are filling contain null values and we can't think of a solution to select the best patch since we have to select specific pixels in each patch.

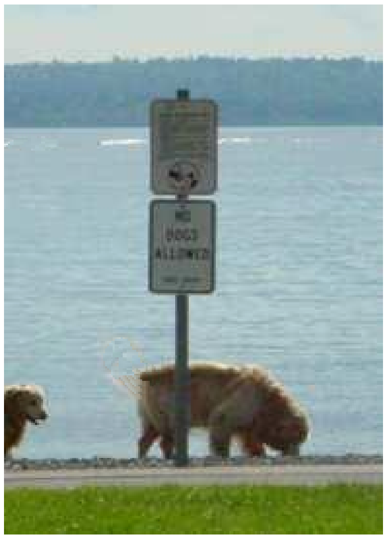
As an alternative to speed up the algorithm a little bit, we decided to fix a certain threshold and we compare the distance between the central pixels of the target patch and each candidate patch to it. We will be then comparing only the patches for which the difference between the central pixels of the candidate patch and the target patch is less than the threshold that we have selected. Overall, there is less comparisons, which means less computation time. By doing that, we have reduced the execution time of the algorithm by almost the half and we have obtained reasonable results.



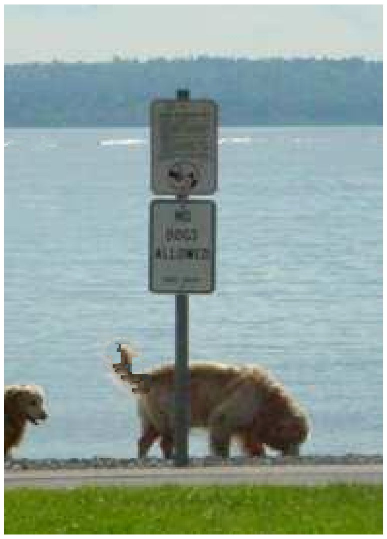
(a) The original image



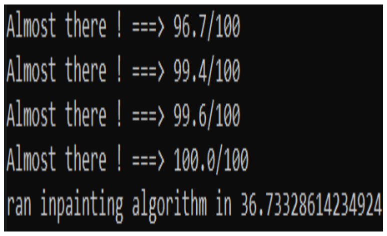
(b) The mask used for this image



(c) The result using the new apporach

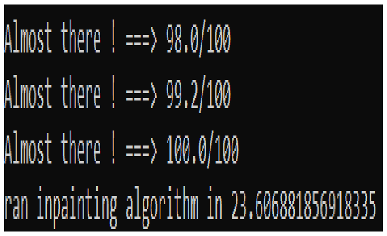


(d) The result without using the new approach



```
Almost there ! ===> 96.7/100
Almost there ! ===> 99.4/100
Almost there ! ===> 99.6/100
Almost there ! ===> 100.0/100
ran inpainting algorithm in 36.73328614234924
```

(e) Execution time without using the new approach



```
Almost there ! ===> 98.0/100
Almost there ! ===> 99.2/100
Almost there ! ===> 100.0/100
ran inpainting algorithm in 23.606881856918335
```

(f) Execution time using using the new approach image

Figure 10: Results when we applied the central pixel approach

7 Conclusion

The algorithm proposed in the paper combines the strengths of the well known approaches in the literature for the sequential models.However, as we saw in the examples, the algorithm has its limitation in propagating the linear structures and its very sensible to the size of the window that we are using. Add to that,the complexity of the algorithm is very high , that's why we proposed some techniques to speed up the execution time.

8 Code

You find in this github repository our implementation for this project:
`https://github.com/taherromdhane/inpainting-images`
`https://the-inpainter.herokuapp.com/`

References

[1] Object Removal by Exemplar-Based Inpainting ,A. Criminisi, P. Perez, K. Toyama

[2] https://dash.plotly.com/

[3] https://towardsdatascience.com/introduction-to-image-inpainting-3fbc60373341