Improving resolution-robust Large Mask Inpainting with Fourier Convolutions

Mohamed Ali SRIR, Louis MARTINEZ, Youness BAHOUS, Benoît RIDEAU
Supervisor : LADJAL Saïd

June 26, 2023

Abstract

Image inpainting methods face challenges when dealing with large regions, as they often struggle to accurately fill the missing areas due to the limited perceptual field in the loss function. In this report, we propose an improved approach called LaMa (Large mask inpainting) based on the architecture introduced in [SLM+21]. LaMa utilizes Fourier convolutional blocks to address this issue. Despite its effectiveness, the results of LaMa still exhibit certain shortcomings, particularly in handling color and texture region filling. Consequently, our research focuses on enhancing the performance of LaMa in these aspects, aiming to provide more accurate and visually pleasing inpainting results.

1 Introduction

The objective of this project was to thoroughly examine the outcomes of implementing LaMa and propose additional methods that can significantly enhance or supplement its performance, ultimately leading to more satisfactory results. Throughout our investigation, we focused on several key aspects that required improvement.

One of the main areas of concern was the texture filling capability of LaMa. While LaMa generally produced impressive results, we noticed instances where the generated fills did not align geometrically as expected. This issue was particularly evident in certain cases, as illustrated in 5. To address this, we aimed to develop innovative techniques that would ensure more accurate and visually pleasing texture fills, enhancing the overall quality of the inpainted images.

Furthermore, we observed discrepancies in color correspondence for the desired filling of our regions of interest. It was crucial for the generated fills to align with the expected colors, maintaining consistency and realism in the inpainted areas. To tackle this challenge, we explored various strategies, to ensure that the generated fills accurately represented the intended colors.

Another aspect we investigated was the adaptability of LaMa to multi-scale scenarios. While LaMa demonstrated exceptional performance on low-resolution images, we sought to determine its effectiveness in handling images of varying resolutions. By exploring the potential of LaMa in multi-scale inpainting, we aimed to extend its capabilities and provide a more versatile solution for a wider range of image sizes and complexities .

To streamline our processes and make them more efficient, we also delved into automation techniques for our treatments. By automating certain steps of the inpainting process, we aimed to reduce manual intervention and increase the overall speed and accuracy of the system. By exploring block matching techniques to find the best suited patches for texture filling and for color transfer.

Through our comprehensive investigation and experimentation, we aimed to not only identify the limitations of LaMa but also propose innovative solutions and enhancements that would significantly improve its performance and make it a more robust and reliable image inpainting system.

2 LAMA

LaMa is a novel single-stage image inpainting system that aims to address the limitations of existing image inpainting systems. The main components of LaMa are the high receptive field architecture, the high receptive field loss function, and the aggressive algorithm of training masks generation . It uses fast Fourier convolutions (FFCs), which have an image-wide receptive field, to improve the effectiveness of the receptive field in both the inpainting network and the loss function. The high receptive field perceptual loss is used to ensure that the generated fills accurately represent the intended colors and textures. Additionally, large training masks are used to unlock the potential of the first two components, allowing LaMa to handle large missing areas and complex geometric structures . LaMa has been shown to improve the state-of-the-art across a range of datasets and achieve excellent performance even in challenging scenarios, such as the completion of periodic structures. It can also generalize surprisingly well to resolutions that are higher than those seen at train time, achieving this at lower parameter and time costs than competitive baselines .

LaMa was trained using a subset of the Places-Challenge dataset, which contains over 10 million images. However, to reduce the computational cost of training, LaMa was trained only on low-resolution 256 x 256 crops of approximately 512 x 512 images. The training process involved generating large training masks using an aggressive algorithm of training masks generation, which ensured that the contours of the training masks were diverse enough to handle a wide range of irregular masks

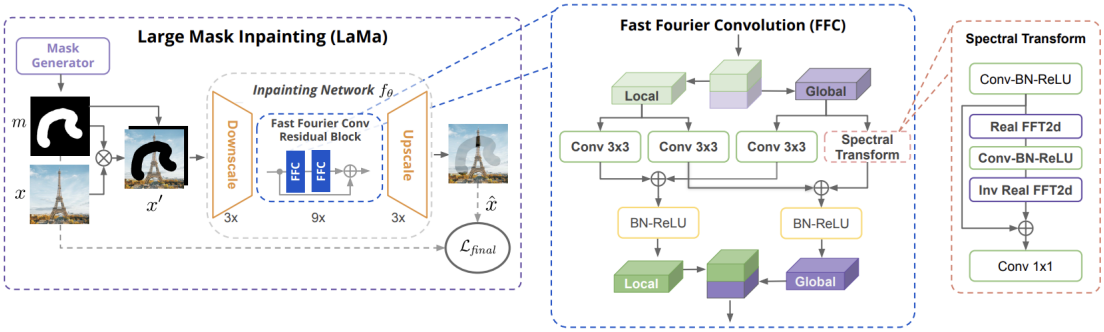

Figure 1: LaMa as proposed by [SLM+21]

2.1 LaMa Results

To check out more results of this method the authors check out [SLM+]

3 Improvements in texture filling

From the example provided in [SLM+21] we notice that LaMa struggles often with texture filling. To improve upon this we use the architecture proposed by [GEB15] with slight modications to adapt it to inpainting tasks. It is a method of generating textures using convolutional neural networks (CNNs) The method involves using the feature spaces of CNNs optimized for object recognition (We used VGG19)

to represent textures by the correlations between feature maps in several layers of the network . The resulting model can generate high-quality textures that capture the statistical properties of natural images while making object information more and more explicit.

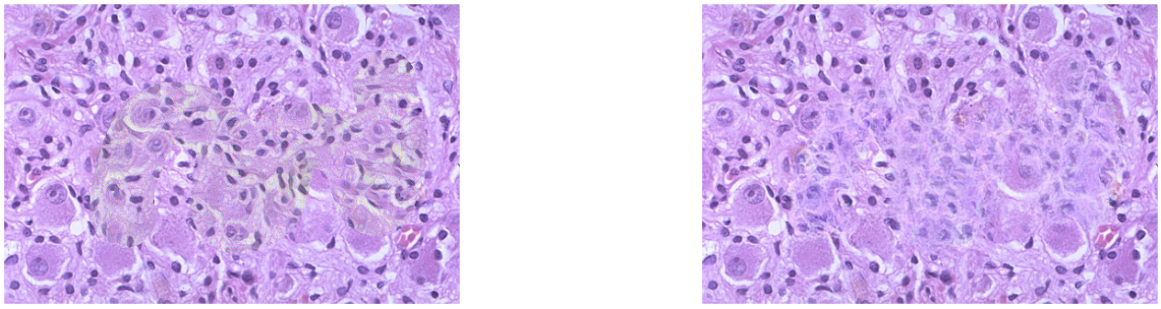

Figure 2: To the left : Filling using our adapted method based on [GEB15], result issued by [SLM+21], As can be seen on the figures the result we provided yields much more convincing textures than those produced by LaMa

To adapt the filling texture for the inpainting task, we employed an example texture that is suitable for filling the region of interest. From this example, we generated the texture to be used for filling. Throughout the filling process, we experimented with various initializations, including our LaMa output, random Gaussian noise, the masked LaMa output, and a homogeneous region with the average of the LaMa filling. It is worth noting that initializing with LaMa yielded favorable results within a reasonable number of epochs.

However, we encountered a problem with broken borders during the process. To address this issue, we incorporated a term into the loss function, where we compute the sum of the squares of the gradient along the border. this is particularly effictive when we initialise with noise, but when working with lama output , this term seems to be useless since the border is already continuous.

In our implementation the loss was written as follows :

$$\mathcal{L} = \sum_{i=0}^{L} (\hat{G}^i - G^i)^2 + \sum_{i \in \partial\Omega} {grad_x^i}^2 + {grad_y^i}^2$$

we denote by $\partial\Omega$ the frontire of the mask Ω and by $\hat{G}^i$ and G^i respectively are the gram matrices during optimisation and the ones given by the VGG layers (see 3)

3.1 Some results

Since the best way to assess the quality of texture generation and Image inpainting is how they are percieved by the uman eye, here are some examples of this approach :

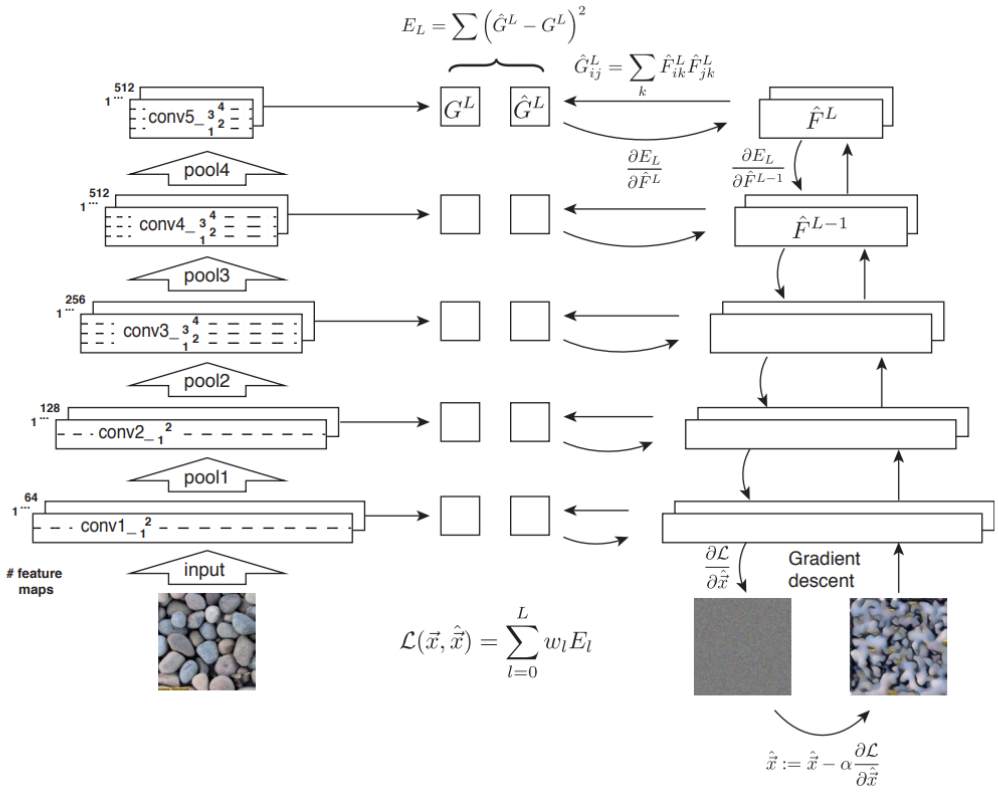

Figure 3: The proposed texture generation model, as outlined by Gatys et al. (2015) [GEB15], consists of two main components: the VGG architecture on the left side and the optimization process on the right side. On the left side, the VGG architecture is employed. The input to this architecture is the example texture. Within the VGG architecture, Gram matrices (G^L) are computed. These Gram matrices capture the style information present in the input texture. On the right side, an initialization step is performed. In the provided figure, white noise is used as the initial input. At each iteration, the Gram matrices are computed for the current input, and the loss is calculated. The loss represents the discrepancy between the current input texture and the desired style captured by the Gram matrices. To optimize the generated texture, gradient descent is employed. The gradient descent algorithm aims to minimize the total loss. By iteratively updating the input texture based on the computed gradients, the model progressively refines the texture generation process.

Figure 4: To the left : Filling using our adapted method based on [GEB15], result issued by [SLM+21], As can be seen on the figures the result we provided Is much more structured and close to the texture of whet herbs would look like, however it is not perfect since we have a slight problem with color induced by the texture example we got our inspiration from.

3.2 Limitations and other alternatives

One potential drawback of employing this approach is its dependence on having an accessible example texture, which may not always be feasible. Furthermore, we have observed that the outcome is significantly influenced by the chosen initialization. Specifically, initializing with noise tends to be effective when dealing with less complex textures (e.g., not intricate grilles or bricks). On the other hand, initializing with the Lama output can introduce biases in the filling process, leading to incorrect colors. To rectify this, we apply color histogram transfer to correct any color discrepancies later on.

Figure 5: To the left : Filling using our adapted method based on [GEB15] , result issued by [SLM+21]

Sometimes when we fill structured textures like the rocks in 5 the result issued by our approuach is visually better but still lacks the correct notion of periodicity since it introduced smaller rockswith some ruptures of borders that dont make sense , for instance we can use a spectrum constraint based approach to enforce the periodicity of the structures as done in [LGX16].

An alternative to the aforementioned method is the more recent texture generation approach called "Texture Generation with Neural Cellular Automata" which was proposed in the paper by Mordvintsev et al. (2021) [MNR21].

4 Improvements In color

In this part, we assume that, given an input image, the output generated by LaMa network has no major outlier. In other words, we applied the process described below to inpainted images where the reconstruction is coherent to human perception. The criterions to assert that an output image looks coherent is emprirical, based on our perception, as there is no metric to quantify the coherence of the inpainted image.

Figure 6: Well vs. badly inpainted images by LaMa

We noticed that color defects appeared on the reconstructed area of "well" inpainted images (in terms of coherence). In most of the cases, colors are blander than on the rest of the image.

Formally speaking, although the coherence in texture is widely captured by LaMa reconstruction, the color distribution is altered.

For ease of explanation, we'll be focusing on a single image, as there is no way to generalize the process. The hyperparemeters values are highly dependant on the area to handle. Hence we use the image below for the whole explanation.

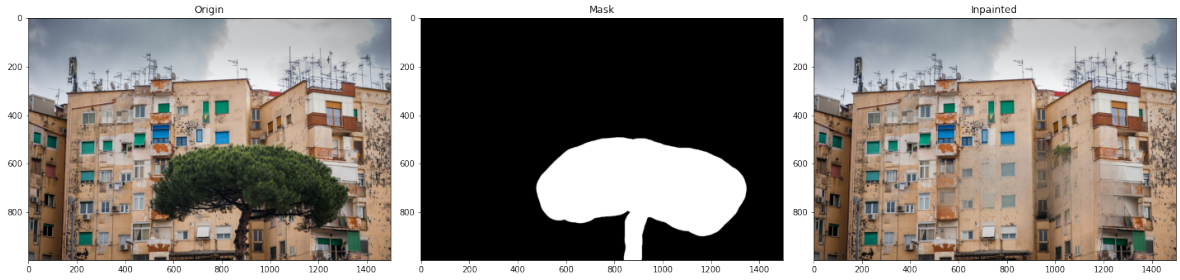

Figure 7: Image used for color transfer

4.1 Key ideas of color tranfer

To tackle the color issue, we used the color transfer method described in [RJ11]. This article proposes to minimize the following loss function :

$$\min_{w\in\mathbb{R}^{d\times N}} \{\mathcal{E}(w) = \underbrace{\sum_{i\in\Omega}\left(\frac{\lambda_L}{2}\|w_i-u_i\|^2 - \lambda_{LS}\left\langle \nabla w_i, \frac{\nabla u_i}{\|\nabla u_i\|}\right\rangle\right)}_{\text{Fidelity term}} + \lambda_R \underbrace{\|w\|_{TV}}_{\text{Regularization term}} + \frac{\lambda_S}{2}\underbrace{SW_2([w],[v])^2}_{\text{Wasserstein distance}}\}$$

where, λ_L,λ_{LS},λ_R and λ_S are hyperparameters, u is the image whose color statistics we want to transfer, and w is the image to recolor. Moreover, $\Omega \subset \mathbb{Z}^2$ represents the spatial grid of the images (pixels).

As this optimization problem is non convex, the article solves it with a proximal gradient descent.

But the major difference between this optimization problem and conventional ones lies in the addition of a further statistical constraint term based on the Wasserstein distance. This term quantifies "how far" is the statistical color distribution of the image to recolor from a given reference image. We will see below that this term is absolutely essential to ensure satisfactory results.

In this particular case, we chose the reference image below :

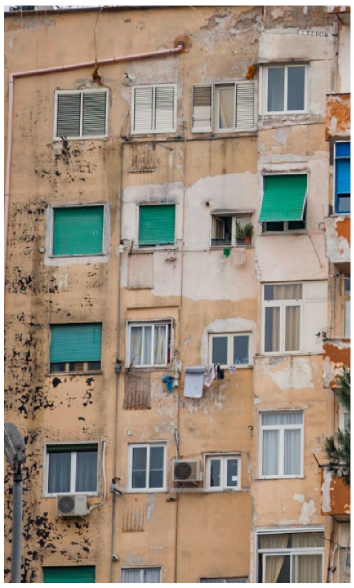

Figure 8: Reference image for color transfer

The color tranfer pipeline is as follows :

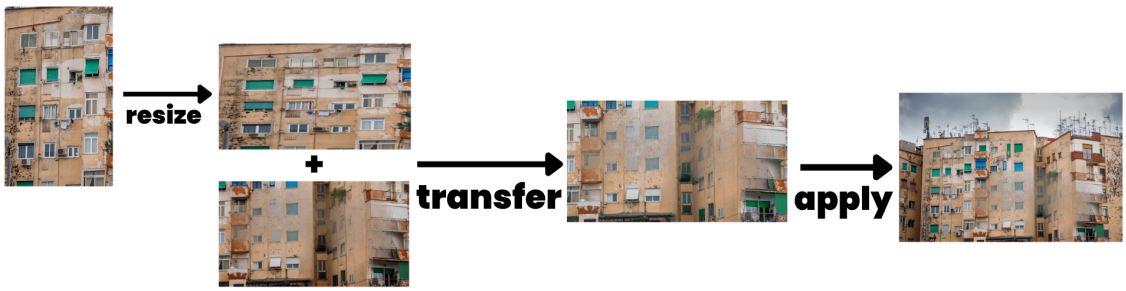

Figure 9: Color transfer pipeline

4.2 Hyperparameters

Here we'll discuss on the influence of the hyperparameters mentioned above. To understand their role, we have varied each hyperparameter over several orders of magnitude, keeping the others to the default value mentioned in the article.

As the Wasserstein distance used in the article is actually an approximation of the real one (projection on a few directions of the unit ball) we could also have modified the number of directions (10 by default). But the results were hardly perceptible and this greatly increased program execution time. So we decided to keep this parameter to its default value and to handle the ones having more visual effects on the result.

- λ_L : When it increases, transitions between colors are getting smoother. But as we want to keep clean transitions, like the reference image, we can consider that $\lambda_L = 0.01$ provides the best result.

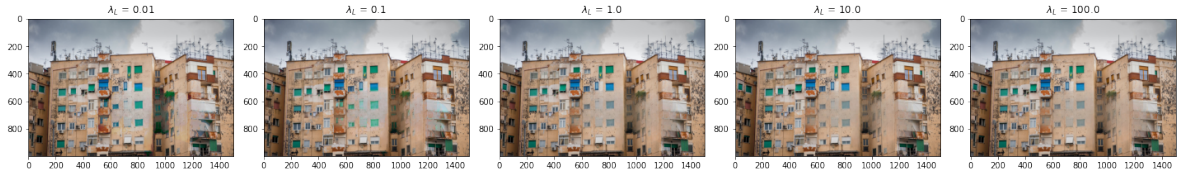

Figure 10: Result images by varying λ_L

- λ_R : This parameter controls the TV regularization term. Logically, when it increases, the inpainted area gets blurrier, as it tries not to avoid wide color variations. But once again, as we want to keep these variations, we chose to keep $\lambda_R = 0$.

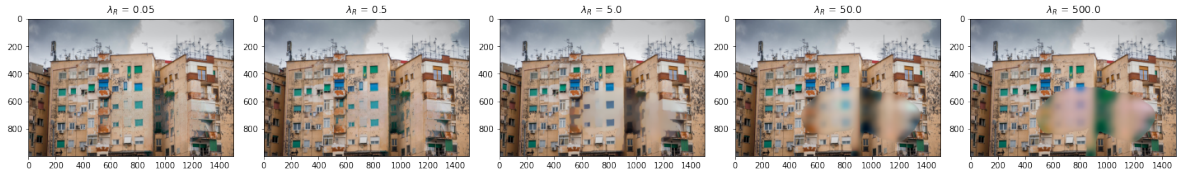

Figure 11: Result images by varying λ_R

- λ_{LS} : This parameter encourages gradients between the reference image and the image to recolor to be similar, or at least varying in the same direction, as this term in the loss function is substracted and not added. (It means that if the gradients evolve in opposite directions, the loss increases). We notice that for small values of λ_{LS} (smaller than 1), differences are hardly perceptible. But on the one hand, as it increases to much, the result becomes incoherent. On the other hand, according to the values chosen for the other parameters, results are more perceptible with small values of λ_{LS}. After several attempts, $\lambda_{LS} = 0.005$ provides a satisfying result.

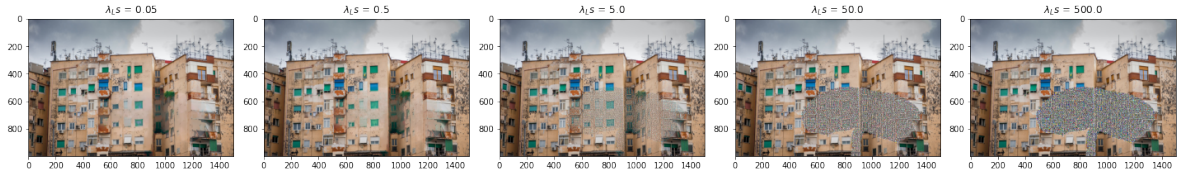

Figure 12: Result images by varying λ_{LS}

- λ_S : This parameter controls the Wasserstein distance term. Its value is of little importance. Whatever the order of magnitude, the result is essentially the same. However, if it is set to 0, we notice a color shift between the inpainted area and the rest of the image.

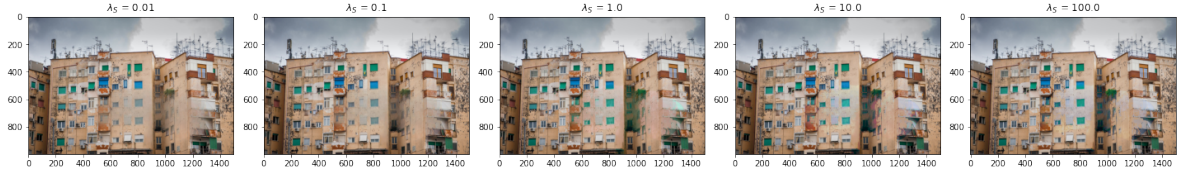

Figure 13: Result images by varying λ_S

Figure 14: Result with $\lambda_S = 0$. We clearly notice the color shift mentioned above. That's why this extra term is essential, event though the value of λ_S doesn't really matter.

4.3 Results and points of improvement

As illustrated above, choosing the right value for each hyperparameter independently and then tuning them to each other is a key step to generate a satisfactory color reconstruction.
However, there is a high variability between the value and the order of magnitude of these parameters, depending on the type of area to handle.

Moreover, for efficiency's sake, the image to recolor is downsampled by a factor 2, then processed and finally upsampled to match the original size of the whole image. But the upsampling step creates a blurry effect on the reconstructed area. With more computing power, it would be sufficient not to subsample this portion to avoid this issue.

Finally, as we use only a part of the recolored image, there is necessarily a tiny difference between the original image and the inpainted recolored part. It also depends of the reference image. But as a way of improvement, we could add as well as for Gatys texture synthesis, we could add an extra term to the loss in order to ensure continuity at the frontiers of the inpainted area. It also implies that we need to adapt the algorithm to provide the original image.

Figure 15: Result of the method with $\lambda_L = 0.1$,$\lambda_R = 0.1$,$\lambda_{LS} = 0.1$,$\lambda_S = 0.1$. Even though the appearance is not perfect, the inpainted area is now textured and the transition with the rest of the image is smoother than before.

Figure 16: Result of the method with $\lambda_L = 0.5$,$\lambda_R = 0.5$,$\lambda_{LS} = 0.5$,$\lambda_S = 100$. There are some defaults on the ground because the reference image was a plain wall of bricks. Therefore, the brick colors of the recolored image (left) are closer to the ground truth than the original one (right).

To conlude, the method proposed by [RJ11] works well thanks to the Wasserstein distance extra term on the loss, for the reasons mensioned above. However, as the areas we wish to handle are completely different from one image to another, the optimal value of hyperparameters depends a lot on the context. Hence it would be really hard to automate this method, or at least to find an algorithm to choose the parameters. Even though we know their individual utility, they cannot be changed independently in a human-predictive way.

5 Using Block Matching

We have seen that we can transfer color from a image to another. Moreover, we have seen that with a adapted architexture, we can improve the texture filling using a neural network.

Nevertheless in both of these cases, we have so far choose the block of the image to transfer the color or the texture by hand. Then, an automatic way to do it will be much more adapted in a real pipeline.

Thus, in this part, we want to adress this issue by finding blocks corresponding the best at the part covered by the mask in order to reconstruct a more precise image.

5.1 Grid creation

First of all, we have to create a grid of blocks from the image and determine which blocks we need to assign to the mask. The main idea is very simple : the grid divides the image in evenly sized blocks. If within a block there is at least one pixel which belongs to the mask, then this block is considered to be part of the mask blocks. We will then search corresponding blocks to the mask blocks so that

we can transfer texture or color from a non-mask block to each mask block to reconstruct the image after LaMa in an even more accurate way.

The image from which we take all the informations is the one obtained after LaMa as we supposed that it is not too far from the reality. Then we can use the mask blocks as a (wrong) approximation of the true block that should be here if the reconstruction was perfect. Indeed, we used the features of the reconstructed mask block to choose the corresponding non-mask block.

5.2 Similarity measure

To obtain the most similar non-mask block to one mask block, we will use the correlation measure. The correlation measure can be resumed as the convolution of this one mask block over the whole image and then we obtain a heatmap if the correlation of the image corresponding to this mask block (see figure 17).

Hence, the correlation heatmap according to a block is computed as :

$$heatmap_{block} = image * block$$

We can fasten this operation manifolds by using the FFT :

$$heatmap_{block} = FFT^{-1}[FFT(image).FFT(block)]$$

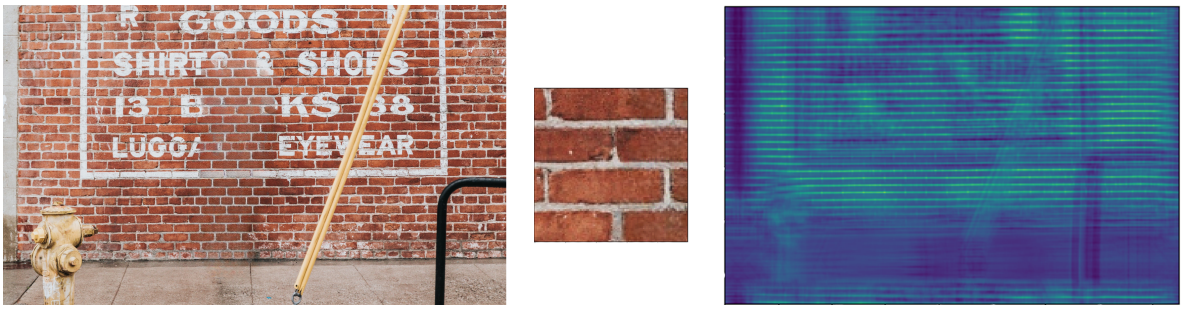

Figure 17: On the left : Reconstructed Image from LaMa, we see that we have to fix the middle part of it. On the center : a block of the image. On the right : the heatmap of the reconstructed image according to the block, the brighter the value is in a certain point, the more correlated the block centered at this point is with the block.

Nevertheless, since the LaMa reconstructed part of the mask is not the wanted result, the correlation can mismatch a very different non-mask block to a mask block, this can be seen in figure 18.

To fix it, we can add another measure so that we can exclude extreme cases of matching. We can add a measure related to the histogram of the three color channels who compute three more distances (one distance by channels between a potential non-mask block and the mask block). If one of the three distances is superior at a certain threshold, then the corresponding block is rejected. We have much better results but still with few artifacts (see figure 19).

Finally, a third distance can be added (not done here) to limit the distance between the matching non-mask block and the mask block.

5.3 Results

After severals experimentations, we saw that the size of the blocks can affect the results : if the size is too small, the texture will not be reconstructed and if they are too big, we will not find a satisfying block to match to each mask block.

Here figures 20 and 21 that show that the algorithm gives great results in another cases.

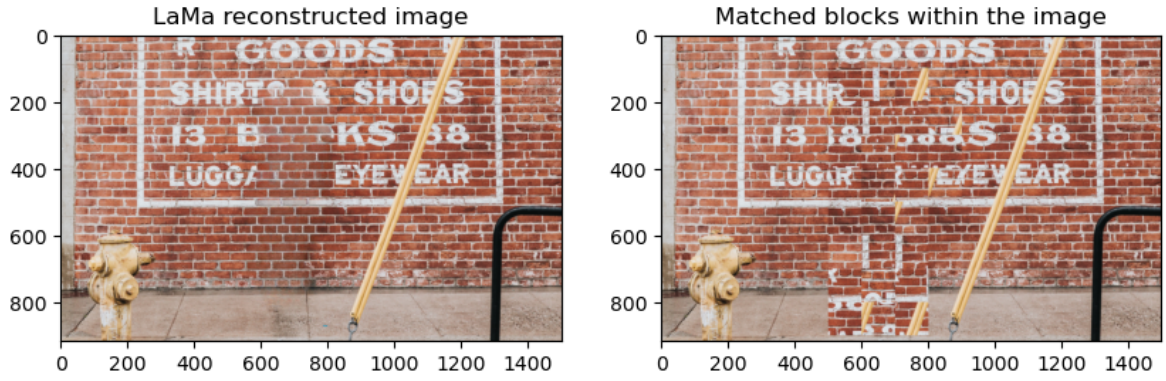

Figure 18: On the left : Reconstructed Image from LaMA. On the right : the image but we replaced each mask block by the corresponding non-mask block found by the block-matching algorithm but here with only the correlation measure in the algorithm.

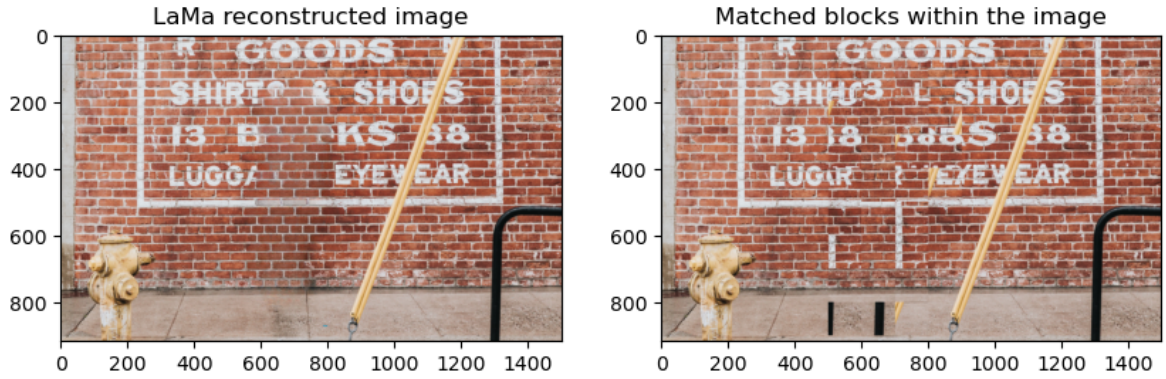

Figure 19: On the left : Reconstructed Image from LaMA. On the right : the image but we replaced each mask block by the corresponding non-mask block found by the block-matching algorithm but here with the histogram comparaison in the algorithm.

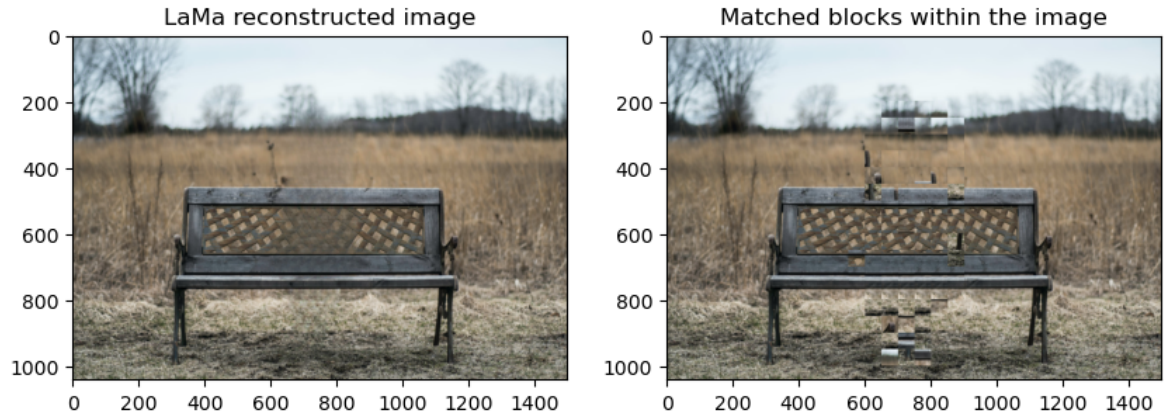

Figure 20: On the left : Reconstructed Image from LaMA. On the right : the result but with matched blocks

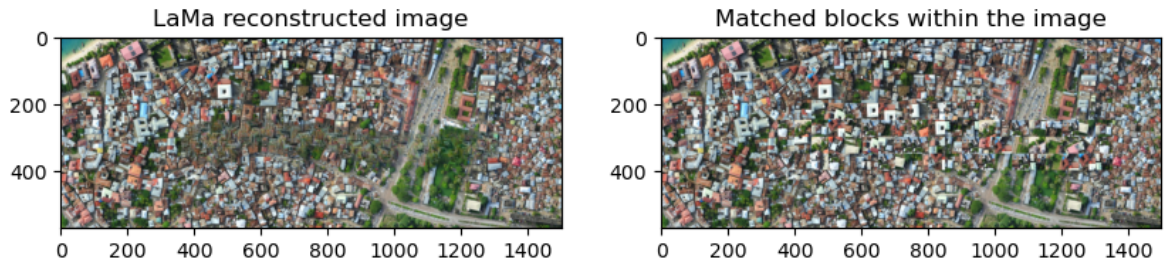

Figure 21: On the left : Reconstructed Image from LaMA. On the right : the result but with matched blocks, here the result is impressive.

6 Using multi-scaling

6.1 Observations

An intriguing aspect worth noting about LaMa is its training methodology, wherein it has exclusively undergone training on images of small proportions, specifically those with dimensions of 256x256 pixels. Despite this inherent limitation, LaMa's underlying structure exhibits a remarkable scalability and applicability, allowing it to effectively process images of diverse sizes. This attribute underscores the adaptive nature of LaMa, showcasing its ability to generalize its learned knowledge to images beyond its initial training scope.An intriguing aspect worth noting about LaMa is its training methodology, wherein it has exclusively undergone training on images of small proportions, specifically those with dimensions of 256x256 pixels. Despite this inherent limitation, LaMa's underlying structure exhibits a remarkable scalability and applicability, allowing it to effectively process images of diverse sizes. This attribute underscores the adaptive nature of LaMa, showcasing its ability to generalize its learned knowledge to images beyond its initial training scope. However, we can see artefacts when inpainting big pictures. Here is an example of inpainting the same duo of image and mask in different scales before getting into the network.

Figure 22: Image and mask

Figure 23: Left : 2x upscale, Center : Original, Right : 0.5 downscale

The artefact in the left is a common artefact generated by this neural network, and even thought it is the same image the result is not just different scales of the same output. The construction is clearly better in the small scale and we can use this fact to output better results.

6.2 Correction Pipeline

Since the reconstruction is better in small scales, we can downscale an image, generate a result, return to the normal scale and fill the empty zone following the pipeline below :

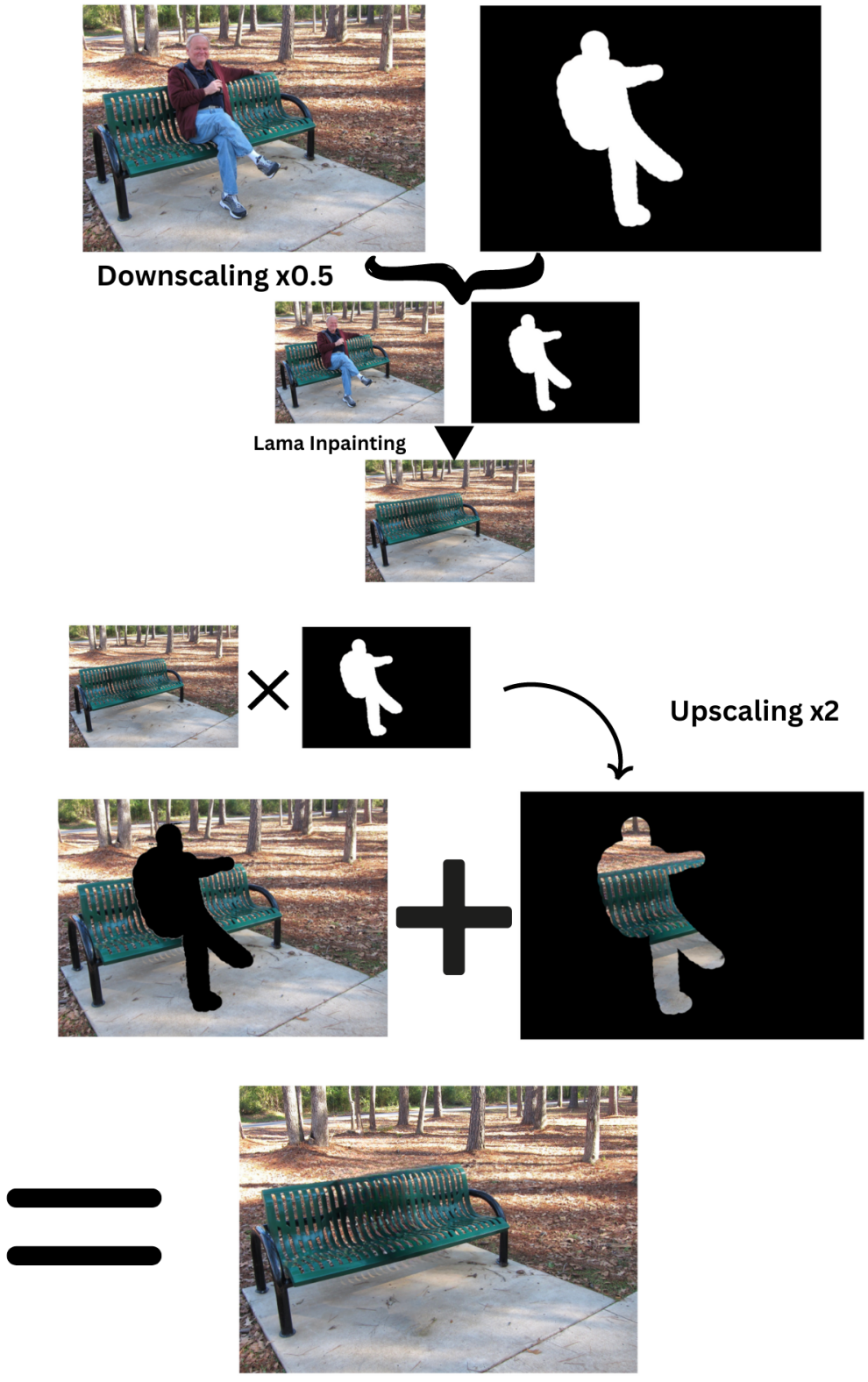

Figure 24: Pipeline of correction using multi-scaling

Here is a comparaison between the original Lama output and our correction.

Figure 25: First output vs corrected

6.3 LaMa Fine-tuning

This observation can be effectively utilized to refine the LaMa model. To achieve this, we maintain the existing structure of the model, import the current weights, and introduce an additional term in the loss function. During training, the input will no longer consist of a single image, but rather encompass various scales of the identical image. In the loss function, a weighted difference in the Euclidean norm (no problem with gradient descent) between the two output images, which have been adjusted to the same scale, is incorporated. Then we will train LaMa on bigger images while using the fact that he already knows how to work on smaller images.

7 Region decomposition

The advantage of LaMa is that the filling is made taking into consideration all the information of the image thanks to FFC layer. However, sometimes the filling should be made considering local information only, in that situation globality is more a disadvantage than an advantage. Let's take a look at this example :

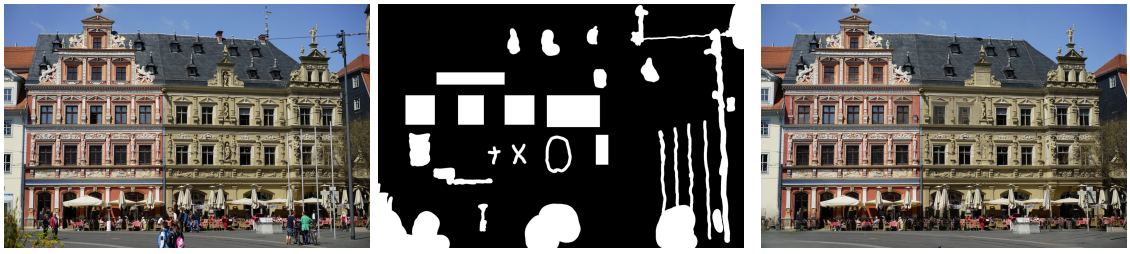

Figure 26: Left : Image, Center : Mask, Right : Output

We can see that the reconstruction of the windows on the right is not very good, and we can notice that the structures are influenced by the structure on the left while it shouldn't be. We can crop the image and the mask in order to force LaMa to only consider local information. Let's see the result.

Figure 27: Left : Image, Center : Mask, Right : Output

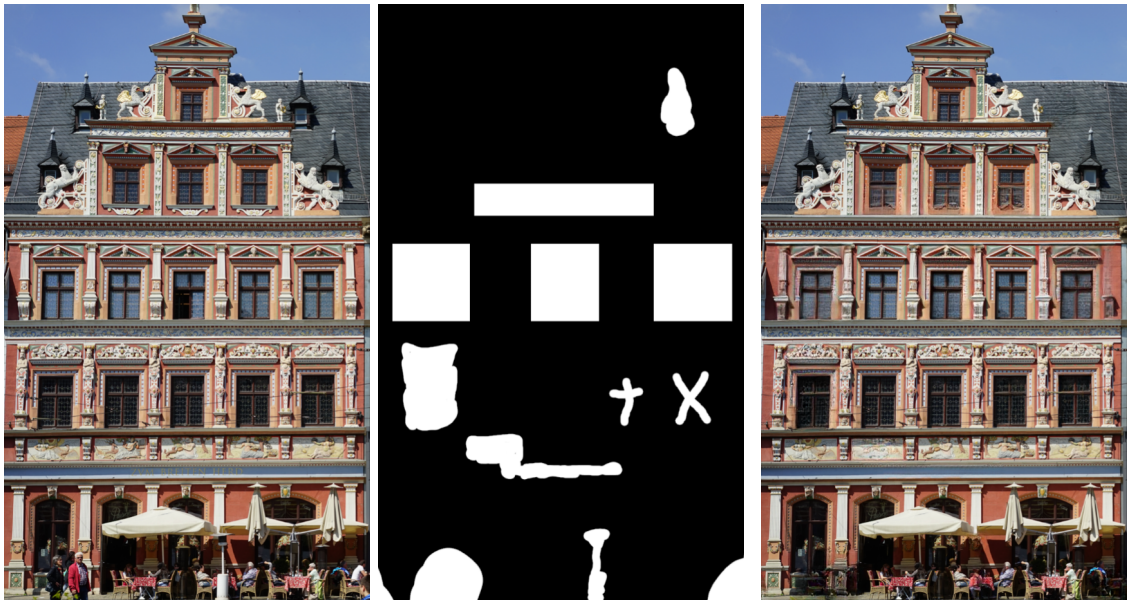

Figure 28: Left : Image, Center : Mask, Right : Output

Here is a comparaison between the original Lama output and our correction.

Figure 29: First output vs corrected

8 Conclusion

In conclusion, this report introduces the LaMa (Large Mask Inpainting with Fourier Convolutions) approach, which addresses the challenge of resolution-robustness in image inpainting. This report has identified certain limitations within the original LaMa model and has put forth innovative solutions and enhancements to moderately enhance its performance. However, it is important to note that further advancements are required to automate the correction process and leverage the proposed techniques. Future research should focus on incorporating these findings into a comprehensive framework for efficient and automated image correction.

References

[GEB15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks, 2015.

[LGX16] Gang Liu, Yann Gousseau, and Gui-Song Xia. Texture synthesis through convolutional neural networks and spectrum constraints, 2016.

[MNR21] Alexander Mordvintsev, Eyvind Niklasson, and Ettore Randazzo. Texture generation with neural cellular automata, 2021.

[RJ11] Gabriel Peyré Rabin Julien. Régularisation de wasserstein. application au transfert de couleur. gretsi'11. *hal-00744813*, 2011.

[SLM+] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. LaMa Examples. https://advimman.github.io/lama-project/lama_supmat_2021.pdf.

[SLM+21] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. *arXiv preprint arXiv:2109.07161*, 2021.