An Efficient Segmentation Based Pipeline For Image Inpainting

Report submitted in fulfillment of the requirements
for the Exploratory Project of

Second Year B.Tech.

by

Abhinav Gupta, Suyash Shukla, Vaishnav S. Menon

Under the guidance of

Dr. Rajeev Srivastava

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
May 2018

Dedicated to

Our fellow people who have their memories kept as images.

Declaration

We certify that

1. The work contained in this report is original and has been done by ourselves and the general supervision of our supervisor.

2. The work has not been submitted for any project.

3. Whenever we have used materials (data, theoretical analysis, results) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever we have quoted written materials from other sources, we have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date:

Abhinav Gupta, Suyash Shukla, Vaishnav S. Menon
B.Tech./IDD Students
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**An Efficient Segmentation Based Pipeline For Image Inpainting**" being submitted by **Abhinav Gupta, Suyash Shukla, Vaishnav S. Menon (Roll No. 16075003, 16074014, 16074015**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date:

Dr. Rajeev Srivastava
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to express our sincere gratitude to our supervisor Dr. Rajeev Srivastava for providing all the necessary support and guidance for the completion of this project.

Place: IIT (BHU) Varanasi
Date:

Abhinav Gupta, Suyash Shukla, Vaishnav S. Menon

Abstract

Image Inpainting is the process of filling in a region of an image based on the information available in the rest of the image. The goal of inpainting is to produce a revised image in which the inpainted region is seamlessly merged into the image, such that it is not detectable by a typical viewer. The major challenges associated with this problem are that of consistency of the filled area and the computation time required to generate a fill. In this project, we first do a comparative analysis of various image inpainting techniques viz. exemplar based methods, diffusion based methods, patch based approaches and the recently proposed generative adversarial neural networks. We then observe that there are peculiarities in these algorithms that make some of them excellent for certain purposes while completely inept at others. Using this insight, we then develop a pipeline for improving the accuracy of the fill by using suitable algorithms for the corresponding sections.

Contents

List of Figures

List of Tables

List of Symbols

Symbol

Ψ

Ω

ω

δ

π

Φ

Meanings are mentioned alonside use.

Chapter 1

Introduction

1.1 Overview

Image Inpainting is the technique of filling in a region of an image based on the information available in the rest of the image. The goal of inpainting is to produce a revised image in which the inpainted region is seamlessly merged into the image, such that the region is not detectable by a typical viewer and looks consistent as a whole [1].

Image completion has various applications such as Unwanted Object Removal, restoration of old and worn out documents and artworks and image editing. The goal of this project is to complete or inpaint missing areas of an image while keeping the constraints of computational efficiency and consistency in mind. We want to build a system that, in the most ideal sense, should be able to complete holes of any number and any size in the image.

This report also addresses the difficulty of comparison of results generated by our pipeline and the available methods as to why quantitative metrics like SNR and $pSNR$ do not work and only qualitative comparison can be done.

1.2 Motivation of the Research Work

The design and structure of our pipeline is motivated by the pros and cons of the various available computer vision and image processing architectures that are used for the task of inpainting. General observations are made about various parts of an image and use the state of the art systems to fill corresponding parts. We bridge the gaps between the various fills using various segmentation and tagging algorithms.

Image inpainting can be achieved using various approaches viz. exemplar based methods, diffusion based methods, patch based approaches, context encoders and generative adversarial neural networks. Each of these approaches are unique in their own sense and have benefits as well as shortcomings. All these approaches either are really fast or are able to generate novel segments which contribute to improving the consistency of the fill.

One of the traditional approaches is that of diffusion based image synthesis. This technique propagates the local image appearance around the target holes to fill them in. This approach is dependent on the gradient of the pixel density at the boundary of the hole and is thus fast but the problem is that even if it is able to generate novel segments, they are blurred out when the hole size becomes large. In contrast, patch-based approaches have been able to perform more complicated image completion that can fill large holes in natural images. Since these approaches sample the patch to be completed from the image itself, they work well when the image to be completed have redundant parts. The drawback is that these methods are not able to generate novel segments. These approaches are built upon by methods such as planar structure guidance that fill in patches by keeping the underlying structure intact but are highly dependent on the heuristic constraints of specific types of scenes and thus are limited to specific structures.

The recently proposed approach of Context Encoders makes use of GANs in order to generate a fill that is consistent with the rest of the image, that is, a generator is

trained to fool a discriminator. This approach is able to generate novel segments that are not at all in the image but the problem is that GANs are highly scene specific and the training is tough as it is unstable. Thus, our approach is motivated by the following observations:

1. The foreground in an image is usually non-redundant and hence the algorithm needs to generate novel segments for the foreground.

2. The background in an image is usually redundant and hence can be filled in by fast patch based approaches.

Thus, the pipeline segments the image into foreground and background and then fills them up with the approach best suited for each of the segments.

Chapter 2

Literature Review

2.1 Overview

In this chapter, we discuss different approaches for infilling as part of our literature survey. From the observations derived from the results of these approaches, we were able to come up with the idea of the inpainting pipeline.

2.2 Inpainting Using the Navier-Stokes Equations

Bertalmio *et. al.* provided an algorithm to fill images based on the Navier-Stokes equation[5]. The algorithm attempts to imitate basic approaches used by professional restorators. The algorithm also introduces the importance of propagating both the gradient direction (geometry) and gray-values (photometry) of the image in a band surrounding the hole to be filled-in. Bertalmio *et. al.* drew an analogy between the image intensity function for the image inpainting problem and the stream function in a two-dimensional (2D) incompressible fluid. An approximate solution to the inpainting problem is obtained by numerically approximating the steady state solution of the 2D Navier-Stokes Equations' vorticity transport equation, and simultaneously solving the Poisson equation between the vorticity and stream function, in the region to be

inpainted. This elegant approach allows one to produce an approximate solution to the image inpainting problem by using techniques from computational fluid dynamics.

Table 2.1 Comparison of Fluid Dynamics and Image Processing, both using the Navier-Stokes Method[5]

Fluid Dynamics	**Image Processing**
Stream Function Ψ	Image Intensity I
Fluid Velocity $\vec{v} = \nabla^\top \Psi$	Isophote Direction $\nabla^\top I$
Vorticity $\omega = \Delta\Psi$	Smoothness $\omega = \Delta I$
Viscosity ν	Anisotropic Diffusion ν

2.3 An Image Inpainting Technique Based on the Fast Marching Method

Alexandra Telea gave the Fast Marching Method[4] which is a technique for producing distance maps of the points in a region from the boundary of the region. This method combined with a way to paint the points inside the boundary, according to the increasing distance from the boundary of the region. The convention that we follow is given below:

1. $\partial\Omega$ denotes the boundary of the region Ω to be inpainted.

2. $B_\varepsilon(p)$ is the ε-ball of the known image around some unknown point p

For ε small enough, we consider a first order approximation $I_q(p)$ of the image in point p , given the image $I(q)$ and gradient $\nabla I(q)$ values of point q. $I_q(p) = I(q) + \nabla I(q)(pq)$. Next, we inpaint point p as a function of all points q in $B_\varepsilon(p)$ by summing the estimates of all points q, weighted by a normalized weighting function $w(p,q)$: $I(p,q_i) = w(p,q_i)[I(q) + \nabla I(q)(p-q)]$ $I(p)$ is the sum of $I(p,q_i)$ over all q_i belonging to the epsilon ball $B_\varepsilon(p)$ about point p which is to be inpainted. The values of $w(p,q_i)$

depends on the image to be inpainted and hence has to be computed for each point to be inpainted[4].

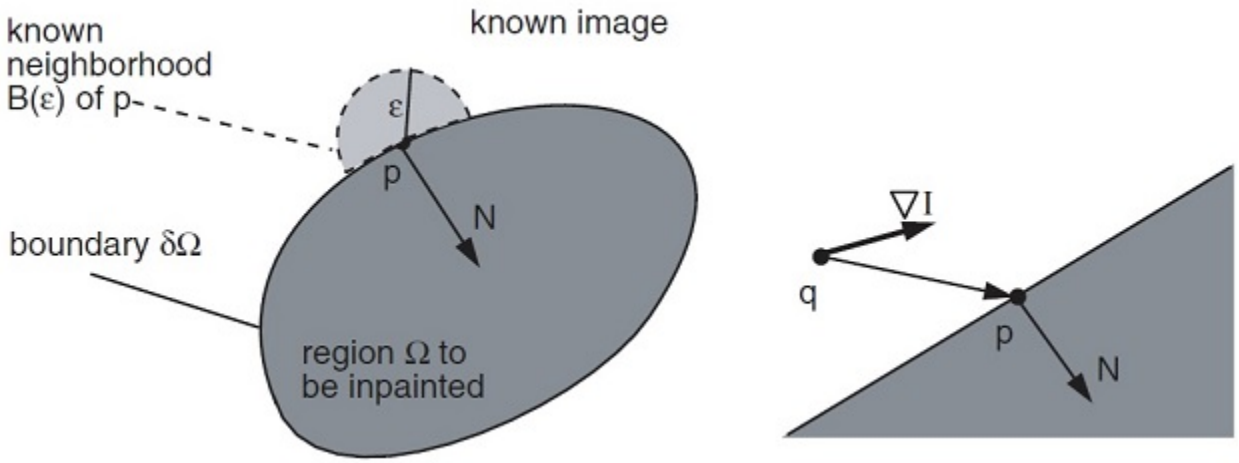

Figure 2.1 Illustration of Fast March Method of inpainting [4]

2.4 PatchMatch

Barnes *et. al.* provided a fast patch based approach called the PatchMatch[2]. This approach provides a randomized nearest neighbour algorithm to find the most similar patch corresponding to the missing region and fills up the region iteratively. The algorithm finds the most appropriate patch that can fill up the missing region by using a distance metric i.e. the error between the input and the output patches. This algorithm is based on the intuition that:

1. Brute force search over the entire image space computationally very expensive and for a user interactive interface, this is a major bottleneck.

2. For coherency of images, a patch by patch matching approach would be better than pixel by pixel matching since independent search for each pixel does not cater to the natural structure of the images.

3. The law of large numbers. Finally, whereas any one random choice of patch

assignment is very unlikely to be a good guess, some nontrivial fraction of a large field of random assignments will likely be good guesses. As this field grows larger, the chance that no patch will have a correct offset becomes vanishingly small.[2]

This algorithm completes images in three major steps:

1. **Initialization**: It first initializes offsets for a plenty of candidate patches randomly i.e. for the start, it chooses a plenty of nearest patches randomly.

2. **Propagation**: Then it tries to improve the error $f(x, y)$ of a patch from the input patch using the known offsets of $f(x-1, y)$ and $f(x, y-1)$. For example, if there is a good mapping at $(x-1, y)$, we try to use the translation of that mapping one pixel to the right for our mapping at (x, y). Let $D(v)$ denote the patch distance ($error$) between the patch at (x, y) in A and patch $(x, y) + \mathbf{v}$ in B. We take the new value for $f(x, y)$ to be the arg min of $\{D(f(x, y)), D(f(x-1, y)), D(f(x, y-1))\}$.[2]

3. **Random Search**: Finally, the algorithm randomly searches for a better patch in some k-distant space of the new position of the candidate patch.

2.5 Exemplar Based Inpainting

Criminisi *et. al.* provided an algorithm that implements an exemplar-based approach to the problem of image completion. The core of this algorithm is an isophote-driven image-sampling process[3]. Exemplar-based texture synthesis is sufficient for propagating extended linear image structures. Suppose that the square template Ψ_p centred at the point p, is to be filled. The best-match sample from the source region comes from the patch Ψ_q, which is most similar to those parts that are already filled in Ψ_p. All that is required to propagate the isophote inwards is a simple transfer

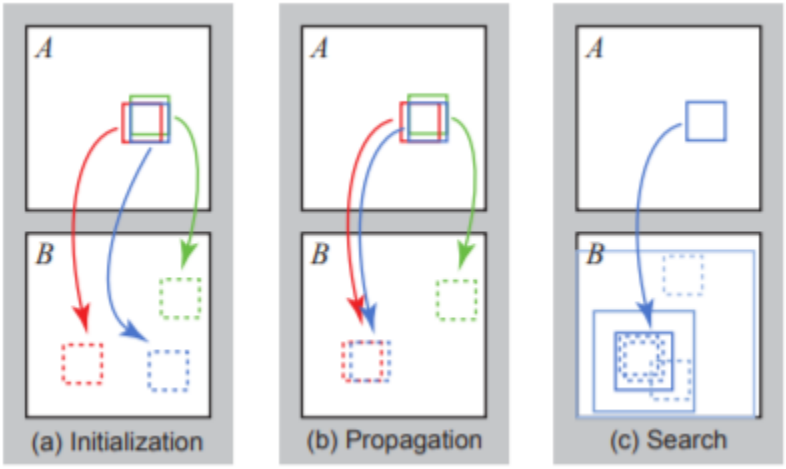

Figure 2.2 Phases of PatchMatch:(a) patches initially have random assignments; (b) the blue patch checks above(green) and left(red) neighbors to see if they will improve the blue mapping, propagating good matches; (c) the patch searches randomly for improvements in concentric neighborhoods. [2]

of the pattern from the best-match source patch. The quality of the output image synthesis is highly inuenced by the order in which the filling process proceeds. The filling algorithm proposed in this paper overcomes the issues that characterize the traditional concentric-layers filling approach and achieves the desired properties of:

1. Correct propagation of linear structures
2. Robustness to changes in shape of the target region
3. Balanced simultaneous structure and texture propagation

In this algorithm, each pixel maintains a colour value (or "empty", if the pixel is unfilled) and a confidence value, which reflects our confidence in the pixel value, and which is frozen once a pixel has been filled[3]. During the algorithm, patches along the fill front are also given a temporary priority value, which determines the order in which they are filled. The algorithm iterates the following three steps until all pixels have been filled:

1. **Computing Patch Priorities**: Given a patch Ψ centred at the point p for

some p, we dene its priority $P(p)$ as the product of two terms:

$$P(p) = C(p)D(p)$$

$$C(p) \equiv \text{the confidence term}$$

$$D(p) \equiv \text{the data term}$$

2. **Propagating texture and structure information**:We propagate image texture by direct sampling of the source region. We search in the source region for that patch which is most similar to Ψ.

3. **Updating confidence values**: After the patch Ψ has been lled with new pixel values, the condence $C(p)$ is updated in the area delimited by Ψ as follows:

$$C(p) = C(\hat{p})$$

See Figure 2.3 for more details.

2.6 Generative Adversarial Networks

Iizuka *et. al.* provided a fully-convolutional Generative Adversarial Network(GAN)[1]. This approach can complete images of arbitrary resolutions by filling in missing regions of any shape. To train this image completion network to be consistent, this model uses global and local context discriminators that are trained to distinguish real images from completed ones. The global discriminator looks at the entire image to assess if it is coherent as a whole, while the local discriminator looks only at a small area centered at the completed region to ensure the local consistency of the generated patches[1]. The image completion network is then trained to fool the both

context discriminator networks, which requires it to generate images that are indistinguishable from real ones with regard to overall consistency as well as in details. This approach can generate fragments that do not appear elsewhere in the image, which allows us to naturally complete the images of objects with familiar and highly specific structures, such as faces. There is unstable training as the discriminator network adds to the loss and the generative network subtracts from it. In other words, we have to maximize GAN loss and minimize Mean Squared Error(MSE) loss at the same time. The major details of the architecture are:

1. The architecture is composed of three networks: a completion network, a global context discriminator, and a local context discriminator.

2. The completion network is fully convolutional and used to complete the image, while both the global and the local context discriminators are auxiliary networks used exclusively for training.

3. The completion network makes use of dialated convolutional layers which allow increasing the area each layer can use as input.This is done without increasing the number of learnable weights by spreading the convolution kernel across the input maps. This helps in filling in large holes.

4. During each training iteration, the discriminators are updated first so that they correctly distinguish between real and completed training images. Afterwards, the completion network is updated so that it fills the missing area well enough to fool the context discriminator networks.[1]

See figure 2.4 showing the fully connected convolutional architecture.

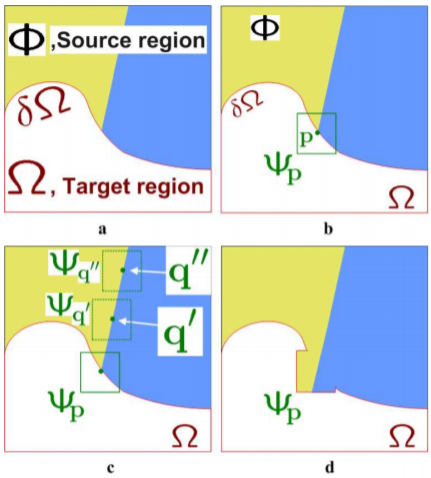

Figure 2.3 Structure propagation by exemplar-based texture synthesis: (a) Original image, with the target region Ω, its contour $\delta\Omega$, and $\delta\Omega$the source region ϕ clearly marked. (b) We want to synthesize the area delimited by the patch Ψ_p centred on the point $p \epsilon \delta\Omega$. (c) The most likely candidate matches for Ψ_p lie along the boundary between the two textures in the source region, e.g., $\Psi_{q'}$ and $\Psi_{q''}$. (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. Notice that both texture and structure (the separating line) have been propagated inside the target region. The target region Ω has, now, shrank and its front $\delta\Omega$ has assumed a different shape.[3]

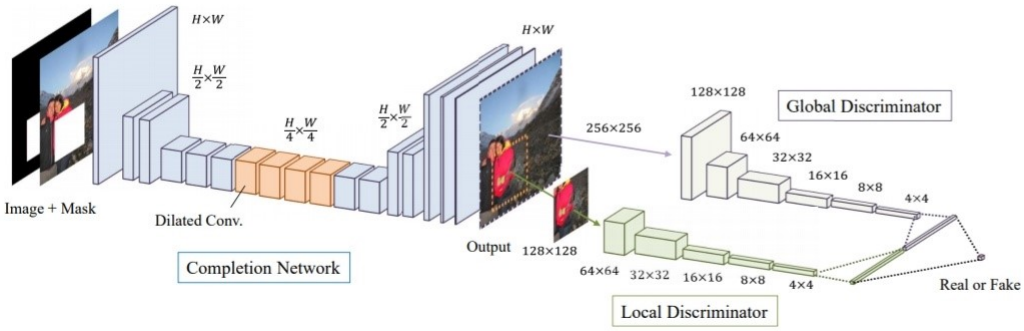

Figure 2.4 The fully connected convolutional GAN for Inpainting[1]

2.7 Results and Conclusions

From our observations we see that GANs are pretty good in generating novel segments and fill objects like faces brilliantly. Even though they take some time to fill, the quality of fill is very good. We also see that GANs are very scene specific and need to be trained on specialized datasets. We also see that approaches such as PatchMatch are really fast and do not require any training. They also fill redundant segments pretty nicely. Thus these can be used in filling up of the background.

Finally we can conclude that, from these observations, we can build our pipeline upon GANs for foreground and Patch Based approaches for background, both stiched using segmentation and tagging algorithms.

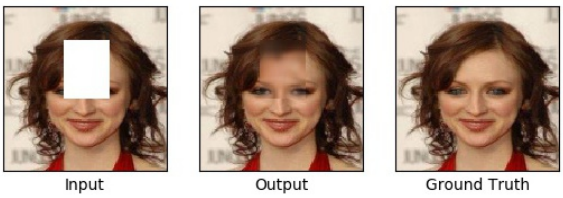

Figure 2.5 Result of GAN Inpaining

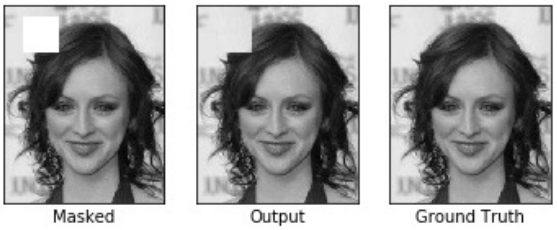

Figure 2.6 Result of PatchMatch Inpaining

Figure 2.7 Result of Fast Marching Method and Navier-Stokes Based Inpainting

Figure 2.8 Result of Exemplar Based Inpaining

Chapter 3

The Pipeline

3.1 Introduction

As we concluded from the last chapter, some algorithms are better than others at certain tasks. In order to leverage this, we device a pipeline that takes an image and processes it such that the region to be inpainted is inpainted with the apt algorithm. The pipeline is broken into 4 major parts:

1. Segment the image into Foreground and Background: Using Saliency Detection and Foreground Extraction via GrabCut, we segment the image. This is done so that we can pick from the foreground and background algorithms (viz. Generative Adversarial Networks (GANs) and patch based algorithms).

2. Fill in the background using a patch based approach like PatchMatch.

3. Tag the different foreground segments into classes and use a specialized GAN to fill them up.

4. Stitch the background and the foreground to get the final infilled image.

We have already discussed the GAN and Patch based approaches above. We now look at the rest of the pipeline, and its effectiveness.

3.2 Foreground and Background Segmentation

3.2.1 Salient Object Detection using Minimum Barrier Distance Transform

What we want to do is to generate a saliency map using the approch provided by Zhang *et. al.* that suppresses the objects in the background and highlights the one in the foreground. To do this, we use a salient object detection method based on the Minimum Barrier Distance (MBD) Transform[6]. The MBD transform is robust to pixel value fluctuation, and thus can be effectively applied on raw pixels without region abstraction. The original MBD is slow and an approximate speedup is done by using a raster and inverse raster based convergence technique that provied a 100X computational speedup. Saliency detection is based on the Image Boundary Connectivity Cue which states that the background regions are usually connected to image borders. Therefore, to leverage this cue using something like a Geodesic Distance, region abstraction is needed which is a serious computational bottleneck. Thus, MBD works on raw pixel values to remove this bottleneck. The Minimum Barrier Distance algorithm works in the following way:

1. We apply a distance transform technique for each pixel in the image. Formally, we consider a 2-D single-channel digital image I. A path $\pi = (\pi(0), \cdots, \pi(k))$ on image I is a sequence of pixels where consecutive pairs of pixels $(\pi(0), pi(1), ...)$ are adjacent. Given a path cost function F and a seed set S, the distance transform problem entails computing a distance map D, such that for each pixel t:

$$D(t) = \min_{\pi \epsilon \Pi_{S,t}} F(\pi)$$

where $\Pi_{S,t}$ is set of all points which connect the seed pixel in S and t. The

function F is the Minimum Barrier Distance given by:

$$F(\pi) = \max_{i=0}^{k} I(\pi(i)) - \min_{i=0}^{k} I(\pi(i))$$

where I(.) denotes the pixel value.

2. This is also computationally expensive and thus we apply a fast raster scan method in which we calculate the the distance transform dynamically along the raster and the inverse raster for some fixed number of iterations. The error analysis in the cited paper shows that the distance transform converges after some number of iterations. The dynamic distance is calculated using:

$$D(x) = min(D(x), F(P(y) \otimes (y, x)))$$

where x in the current pixel, y is the set of neighbouring pixels, P(y) denotes the path currently assigned to the pixel y, (y, x) denotes the edge from y to x, and $P(y) \otimes (y, x)$ is a path for x that appends edge (y, x) to P(y).

3. The the set of seeds S is defined as the boundary pixels and raster MBD is applied to get the distance transform of each of the pixels from the boundary seeds. The pixel values are further scaled so that the maximum value is 1. Then according to the Image boundary connectivity cue, we set a certain threshold and the distance transforms greater than the threshold are marked as salient.[6]

3.2.2 GrabCut and Foreground Extraction

An algorithm was needed for foreground extraction with minimal user interaction, and the result was GrabCut[7].

1. User inputs the rectangle. Everything outside this rectangle will be taken as sure background (That is the reason it is mentioned before that your rectangle should

include all the objects). Everything inside rectangle is unknown. Similarly any user input specifying foreground and background are considered as hard-labelling which means they won't change in the process.

2. Computer does an initial labelling depending on the data we gave. It labels the foreground and background pixels (or it hard-labels)

3. Now a Gaussian Mixture Mode l(GMM) is used to model the foreground and background.

4. Depending on the data we gave, GMM learns and create new pixel distribution. That is, the unknown pixels are labelled either probable foreground or probable background depending on its relation with the other hard-labelled pixels in terms of color statistics (It is just like clustering).

5. A graph is built from this pixel distribution. Nodes in the graphs are pixels. Additional two nodes are added, Source node and Sink node. Every foreground pixel is connected to Source node and every background pixel is connected to Sink node.

6. The weights of edges connecting pixels to source node/end node are defined by the probability of a pixel being foreground/background. The weights between the pixels are defined by the edge information or pixel similarity. If there is a large difference in pixel color, the edge between them will get a low weight.

7. Then a mincut algorithm is used to segment the graph. It cuts the graph into two separating source node and sink node with minimum cost function. The cost function is the sum of all weights of the edges that are cut. After the cut, all the pixels connected to Source node become foreground and those connected to Sink node become background.

8. The process is continued until the classification converges.

However, GrabCut needs human input in the form of the seeds and bounded boxes to find foreground and background. Instead, we can use the salient object detection algorithm above to find this seed. This is a way to automate the process of foreground detection.

3.3 Tagging The Foreground and Categorization

The basic idea behind tagging the foreground is to use an appropriately trained GAN to fill in the section. This makes the pipeline extremely scalable, as new GANs for appropriate foreground objects can be added without any change to the pipeline at all, except for registering this new GAN into the pipeline. This is a three step process:

3.3.1 Preprocessing: Fill Foreground Hole Using Fast Marching Method

Any holes in the foreground pose a problem to the image tagger, as it considers the hole to be part of the image, and thus reports the category incorrectly. In order to combat this, we fill in the hole prematurely using the Fast-Marching Method. We use the FMM as it is both fast and keeps local consistency, which improves the tagging result. (See Figures 3.6 - 3.8)

3.3.2 Image Tagging

We use Google's pre-trained image tagger called the Inception-V3[8] to tag the different sections of the foreground into different classes. This neural network was developed by Google as part of the ImageNet Large Scale Visual Recognition Challenge(ILSVRC) and classifies images into 1000 classes specified by ImageNet. We only consider the section suggested by the tagger that has the highest confidence.

Figure 3.1 Original Image with Hole

Figure 3.2 Full Saliency Map by Minimum Barrier Distance algorithm

Figure 3.3 Saliency Map mask after thresholding

Figure 3.4 GrabCut mask using previous saliency mask

Figure 3.5 GrabCut Foreground Section

3.3.3 Categorization

We trained GANs on the *Celeb-A* Dataset for facial filling, the *Places2* dataset for general scene structures and *ImageNet* as a generalized GAN so as to fill images that did not fall in the above two categories.

To pick which category the foreground entity belongs to, we take the cross-product of the categories of GANs and the tags to form an ordered pair of words $(tag_word, category_word)$. We then find the Wu-Palmer Similarity of the each pair of words on the WordNet corpus. This is basically the depth of the two words in the taxonomy and that of their Least Common Subsumer (most specific ancestor node). We pick the best such pair and use it to pick the GAN. If the confidence score falls below a threshold, we use the generalized GAN.

Importantly, to extend this system, we simply need to extend the categories list of words for each new GAN, making the system scalable.

Table 3.1 Results of Classifier and Tagger of Figure 3.6

Classifier Output	**Result of Tagging**
{"wig": "0.08326399", "quill, quill pen": "0.05241453", "stage": "0.06860219", "feather boa, boa": "0.07666192", "microphone, mike": "0.059891075"}	fallback

3.4 Results

Figures 3.9 - 3.13 show results of the integrated pipeline.

We do a qualitative comparison because metrics like SNR fail because they are dependent of the variance of pixel density in the pixel energy graph. But the thing with infilling is that you can have a high pixel variation but the quality of fill can also be better, but high pixel pixel variance is indicated as a poor result by these metric. Take for example, infilling at boundaries requires high pixel variation to generate a

Figure 3.6 Original Image

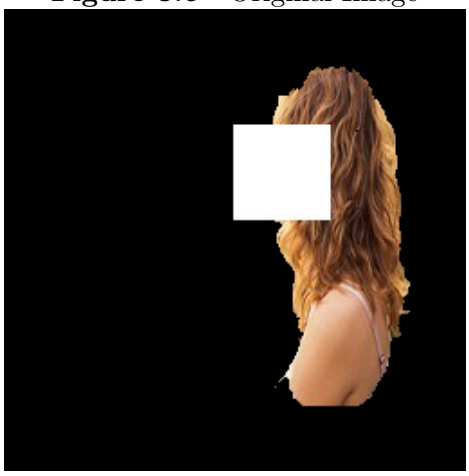

Figure 3.7 GrabCut Foreground Extraction

Figure 3.8 Telea Infill Foreground Extraction

good fill. The other problem is that the inpainted region is usually small and these values do not change much from the ground truth whatever may be the fill. $pSNR$ cannot be used as it requires a ground truth image to take comparisons, something which is not usually available. However, the $pSNR$ results we obtained using the original ground truth as the base do show that our method is significantly better than the holed image every time.

Table 3.2 SNR and pSNR values of the 5 result images. Highest SNR values in **bold** and smallest values *italicized*. Note the inconsistent pattern.

Image	SNR(Holed Image)	SNR(Filled Image)	SNR(Ground Truth)	pSNR(Holed Image)	pSNR(Filled Image)
Malala Yousafzai	**1.3626712**	1.3448751	*1.3348264*	21.5417369	33.0114100
Man Holding Cigar	**0.9734610**	*0.7929946*	0.9627069	19.8388269	29.7881497
Girl's Portrait	**1.6282203**	*1.4654143*	1.6047214	18.5025386	30.0990372
Deer in Forest	1.8614853	**1.9807348**	*1.8445147*	21.2103386	32.0667446
Starfish	**1.5182899**	*1.4625321*	1.5128821	22.4422056	28.8908365

Table 3.3 SNR values of the 5 result images with various methods.

Image	Holed Image	Telea FMM	Exemplar Based	GAN(CelebA)	Proposed Method	Ground Truth
Malala Yousafzai	1.3626712	1.3480339	1.3493832	1.3325245	1.3448751	1.3348264
Man Holding Cigar	0.9734610	0.7969325	0.7960292	0.7832322	0.7929946	0.9627069
Girl's Portrait	1.6282203	1.4704976	1.4612858	1.4566601	1.4654143	1.6047214
Deer in Forest	1.8614853	1.9857905	1.9919077	1.9693860	1.9807348	1.8445147
Starfish	1.5182899	1.4641308	1.4578623	1.4484318	1.4625321	1.5128821

Table 3.4 pSNR values of the 5 result images with various methods.

Image	Holed Image	Telea FMM	Exemplar Based	GAN(CelebA)	Proposed Method
Malala Yousafzai	21.5417369	31.7454942	30.2731403	32.4125792	33.0114100
Man Holding Cigar	19.8388269	29.9342887	27.8561026	29.7596782	29.7881497
Girl's Portrait	18.5025386	32.2555852	29.6110473	31.2756809	30.0990372
Deer in Forest	21.2103386	32.9485120	31.4105526	33.0321043	32.0667446
Starfish	22.4422056	30.2476409	26.4108725	28.9019111	28.8908365

Figure 3.9 Malala Yousafzai - viz. Holed Image, Filled Image, Ground Truth; GAN Based, Exemplar Based and Telea FMM.

Figure 3.10 Man With Cigar - viz. Holed Image, Filled Image, Ground Truth; GAN Based, Exemplar Based and Telea FMM.

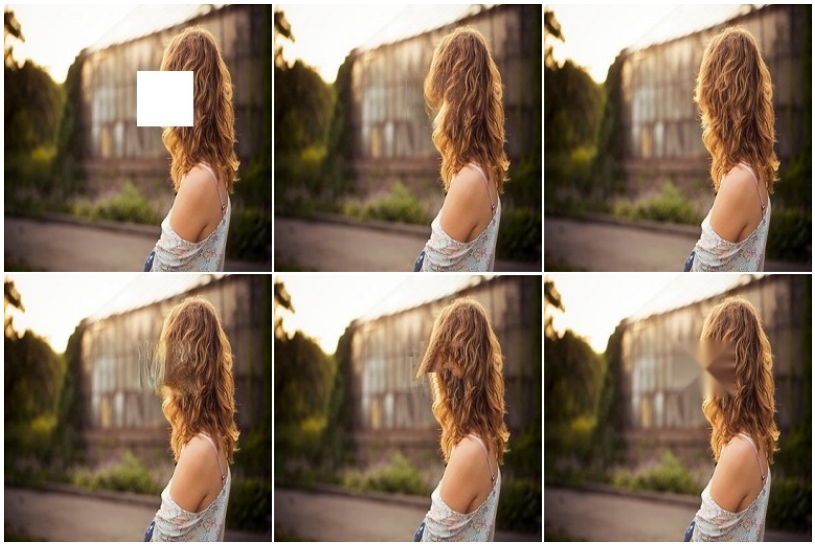

Figure 3.11 Girl's Portrait - viz. Holed Image, Filled Image, Ground Truth; GAN Based, Exemplar Based and Telea FMM.

Figure 3.12 Deer in Forest - viz. Holed Image, Filled Image, Ground Truth; GAN Based, Exemplar Based and Telea FMM.

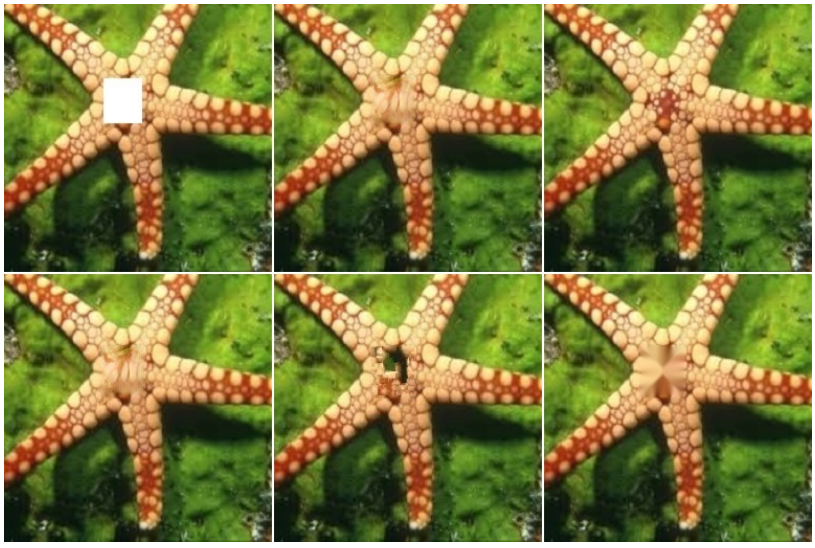

Figure 3.13 Starfish - viz. Holed Image, Filled Image, Ground Truth; GAN Based, Exemplar Based and Telea FMM.

Chapter 4

Conclusions and Discussion

A scalable and effective pipeline for image inpainting has been created, which uses the best possible algorithm for the section of the hole in its repertoire. This is able to solve the major issue with image inpainting algorithms—the fact that different algorithms can only fill certain regions.

Future Directions

- Training more GANs will scale the pipeline and give better results as different categories will be filled better. This requires the use of more GPUs and some data annotation is also required as datasets for rare classes are not readily available.
- The computational efficiency of path based methods, GrabCut and Saliency Detection can be improved by using paradigms of Parallel Computing.
- The Stitching algorithm needs to be improved. Currently, the stitch may give poor results if the holes appear in the interface between the foreground and background. This requires considerable amount of further work.
- A proper result metric must be devised. The only way to currently compare

results, $pSNR$, requires the use of the ground truth image, which may not be available at all. Further, in use cases like unwanted object removal, the ground truth image is useless, as we are trying to generate details not present in the original ground truth image. Metrics like SNR have proven to be unusable, as discussed in Section 3.4

- The various thresholds have to be tuned further. We have only done basic tuning of the pipeline as of now.

- We can add the ability to select from various background-fills as well. This is harder than foreground as the exact images where different algorithms work well are difficult to point out without observation.

- We can use the pipeline to create practical tools now.

Bibliography

[1] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2017. Globally and locally consistent image completion. ACM Trans. Graph. 36, 4, Article 107 (July 2017), 14 pages. [1,9,10,11]

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. 28, 3, Article 24 (July 2009), 11 pages. [6,7,8]

[3] A. Criminisi, P. Perez and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," in IEEE Transactions on Image Processing, vol. 13, no. 9, pp. 1200-1212, Sept. 2004. [7,8,11]

[4] Alexandru Telea (2012) An Image Inpainting Technique Based on the Fast Marching Method, Journal of Graphics Tools, 9:1, 23-34. [5,6]

[5] M. Bertalmio, A. L. Bertozzi, G. Sapiro, Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting, Proceedings of the International Conference on Computer Vision and Pattern Recognition , IEEE, Dec. 2001, Kauai, HI, volume I, pp. I-355-I362. [4,5]

[6] Zhang, J., Sclaroff, S., Lin, Z.L., Shen, X., Price, B.L., & Mech, R. (2015). Minimum Barrier Salient Object Detection at 80 FPS. 2015 IEEE International Conference on Computer Vision (ICCV), 1404-1412. [15,16]

[7] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. 23, 3 (August 2004), 309-314. [16]

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, Zbigniew Wojna; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818-2826. [18]