Dec 2020 | By: Ming Gong, Hongyu Chen

PROJECT REPORT

CSC420 Team: 4090 TI

Abstract

In the project we will focus on the inpainting problem which is one of the biggest problems in computer vision.

We will mainly focus on Region Filling and Object Removal by Exemplar-Based Image Inpainting by A. Criminisi*, P. P´erez and K. Toyama.

In the report we will do a clear and detailed explanation about the algorithm and some explanation of how we implement the algorithm.

There are some output images in the result session of the report to fully shows how good or bad this algorithm and our implementation are.

In the conclusion we will discuss the pro and cons of our algorithm compare to other famous inpainting algorithm. Also some thoughts after finished the project.

Introduction

Inpainting

What is inpainting?

Inpainting was originally a traditional graphical problem. The problem itself is straightforward: how to create a blank hole in an image, and how to use other information in the image to fill the hole in such a way that the human eye cannot distinguish the missing part. The problem seems easy to us humans, for example, it is easy to imagine a hole with a window and a door in it and background is a wall, and if you have some drawing talent, you can probably imagine how to fill it out and you might be able to draw it out by hand easily. But this task is particularly difficult for computers. First, there is no perfect solution to this problem, and second, how can we use the rest information in the image? How do I determine if the complementary results are sufficiently realistic? These are the problems we need to think and solve in this project.

Inpainting algorithms

Most of the algorithms can be addressed into two classes of algorithms. The first one is texture synthesis. “texture synthesis” algorithms for generating large image regions from sample textures. It is the process of algorithmically constructing a large digital image from a small digital sample image by taking advantage of its structural content. The second one is inpainting techniques. “inpainting” techniques for filling in small image gaps. In this project we will focus on Region Filling and Object Removal by Exemplar-Based Image Inpainting by A. Criminisi*, P. P´erez and K. Toyama[1].

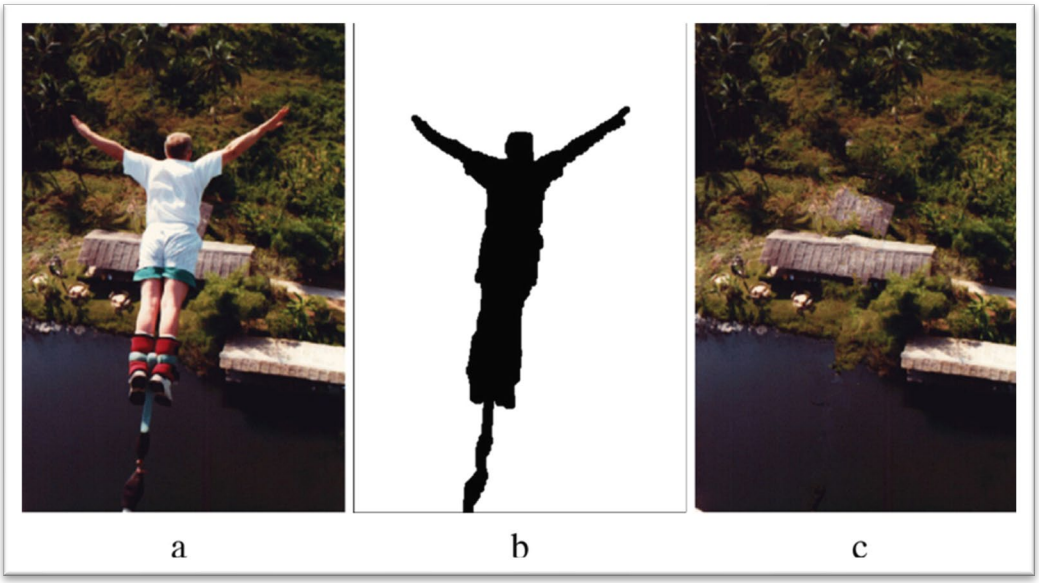
Other than that, we also do a research in Scene Completion Using Millions of Photographs which requires a huge amount of data. It is not actually the best choice of the project. Also, the Context Encoders: Feature Learning by Inpainting. The core idea of this paper, published in CVPR last year, is to use convolutional neural networks to learn high-level features in images and use these features to guide the generation of missing parts of images. The network structure proposed in the article is as follows, including 3 parts: Encoder, Channel-wise fully connected layer, Decoder. This required some ML skill, but we do not have enough time to understand it.

Methodology

Theoretical part

How it works

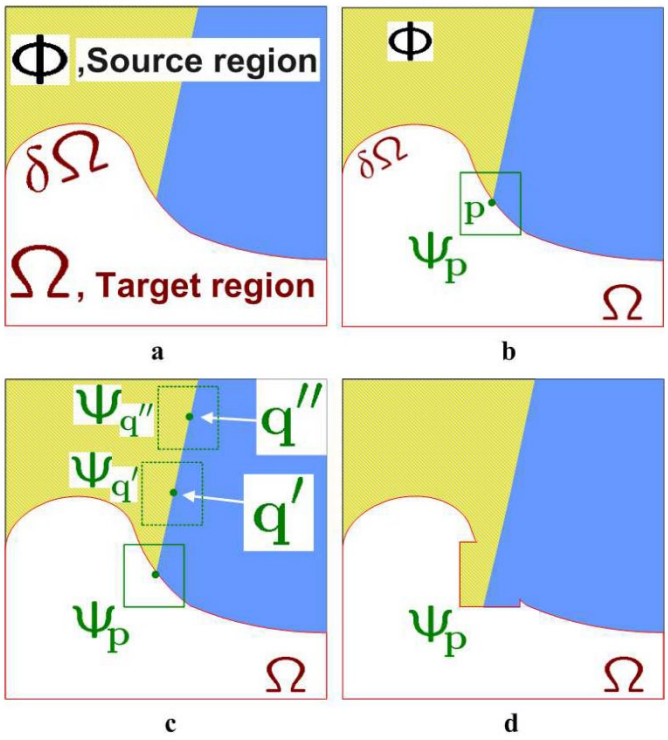
We need two images, first is the original image which you want to do the inpainting (a) and second is the mask image with white background and the black area covers the inpainting required regions (b). Then the last image (c) is supposed to be the output from our program that the person has been removed and the regions has been filled with visually plausible background.



Exemplar-based synthesis

In the figure a, the target region which is the region we want to do the inpainting is indicated as omega. The target region's contour is denoted as $\Delta\Omega$. Then the rest region we want to keep in the image is called source region represented as Φ. Let us move on to figure b, there is a green squared patch whose center is located on a point P and we call this patch Ψp. We want this patch to be filled first, so we will find a best-match patch from the source region Φ. Best match means this picked patch is most like those areas that are already filled in Ψp. As shown in figure c, $\Psi q'$ and $\Psi q''$ are two examples that satisfy this situation (best-match patch). Then in

figure d, we can replace Ψp with the copy of the best matching patch in the candidates set.



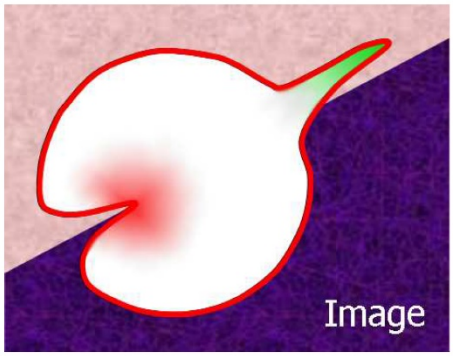
Region-filling algorithm

We can sperate it into three parts. Computing patch priorities, propagating texture and structure information and updating confidence values.

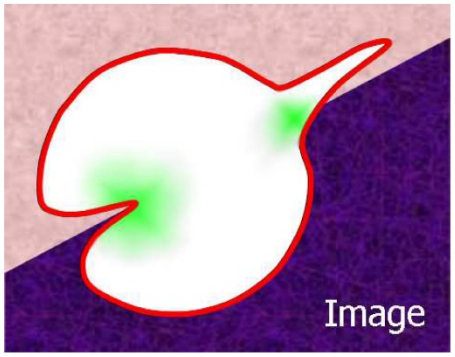
Firstly, we need to compute patch priorities to decide which one is going to be replaced and filled first. The priority of the point P is determined by two parts, confidence term and data term.

The confidence term can be thought of as a measure of the amount of reliable information surrounding the point p. The confidence term assigns high filling priority to out-pointing appendices in green and low priority to in-pointing ones in red in figure a.

The data term is a function of the strength of isophote (direction and intensity) hitting the front delta omega at each iteration. This term boosts the priority of a patch that an isophote “flows” into.



a



b

In figure b data term gives high priority to pixels on the continuation of image structures in green and has the effect of favoring in-pointing appendices in the direction of incoming structures. The combination of the two term produces the desired balance between the two effects.

Equation:

P(p) is the priority equation

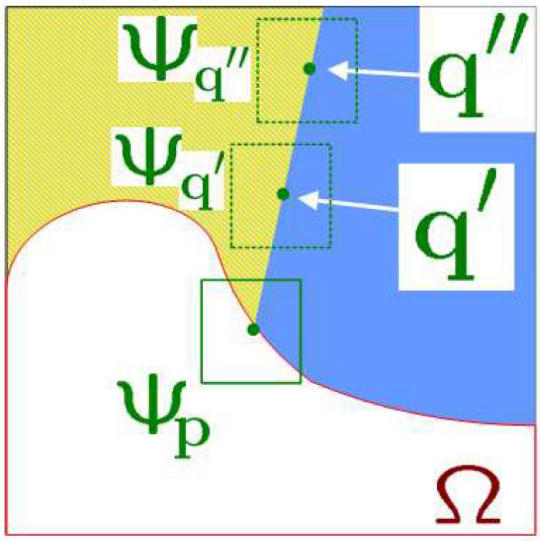
C(p) is the confidence equation

D(p) is the data equation

$$P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p}).$$

$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Psi_{\mathbf{p}}\cap(\mathcal{I}-\Omega)} C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}, \qquad D(\mathbf{p}) = \frac{|\nabla I_{\mathbf{p}}^{\perp} \cdot \mathbf{n}_{\mathbf{p}}|}{\alpha}$$

Once all priorities on the delta omega have been computed, the patch $\Psi\bar{\mathbf{p}}$ with highest priority is found. Then fill it with data extracted from the source region. We need to find the source exemplar $\Psi\bar{\mathbf{q}}$ centered with point q by calculating the square difference between Ψq and $\Psi\bar{\mathbf{p}}$ and consider the sum of squared differences (SSD) of the already filled pixels in patch Ψp and patch Ψq in the source region Φ in order to compute $\bar{\mathbf{q}}$ and $\Psi\bar{\mathbf{q}}$ who has smallest distance. Then we just fill the empty pixels in $\Psi\bar{\mathbf{p}}$ with the value of corresponding pixels in $\Psi\bar{\mathbf{q}}$.



After the priority patch $\Psi_{\bar{\mathbf{p}}}$ has been filled with new pixel values, the confidence C(p) is updated in the area delimited by $\Psi_{\bar{\mathbf{p}}}$.

Equation:

Updating confidence values

$$C(\mathbf{p}) = C(\hat{\mathbf{p}}) \quad \forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega.$$

After this step we will go back to the very beginning and do the same thing again.

Region filling algorithm:

- Extract the manually selected initial front $\delta\Omega^0$.
- Repeat until done:

1a. Identify the fill front $\delta\Omega^t$. If $\Omega^t = \emptyset$, exit.
1b. Compute priorities $P(\mathbf{p}) \quad \forall \mathbf{p} \in \delta\Omega^t$.
2a. Find the patch $\Psi_{\hat{\mathbf{p}}}$ with the maximum priority, *i.e.*, $\hat{\mathbf{p}} = \arg\max_{\mathbf{p} \in \delta\Omega^t} P(\mathbf{p})$.
2b. Find the exemplar $\Psi_{\hat{\mathbf{q}}} \in \Phi$ that minimizes $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$.
2c. Copy image data from $\Psi_{\hat{\mathbf{q}}}$ to $\Psi_{\hat{\mathbf{p}}}$ $\forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$.
3. Update $C(\mathbf{p}) \quad \forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$

This simple update rule allows us to measure the relative confidence of patches on the delta omega. As filling proceeds, the confidence values of the point on the delta omega decrease, indicating that we are less sure of the color values of pixels near the center of the target region. But the algorithm will still fill in the patch that seems plausibly as possible.

Methodology

Empirical work (Quality of Analyses)

Computing patch priorities

1) Initialization

   Before we start the algorithm, we need to first do the initialization of the confidence. We will need a 2D array with the same size of the image(mask) and all the point in the source region will have a confidence value 1 and all of the point in the target region will have a confidence value 0. We can do this using Boolean index arrays.

2) Get Front Fill

   The first thing we need to do to start this algorithm is to get the front fill which is also the boundary of the mask. So we use skimage.feature.canny function with a default sigma value 1.0 to first do the edge detection in order to find the boundary of the mask (front fill). Then what we want is the coordinate of each point on the boundary. We use np.where function to get the coordinates of these point.

3) Patch

   Since this algorithm is a patch-based filling, we will need to compute the patch all the time. The only thing we need to be aware of is the patches may reach outside of the boundary and we do not allow out of bound patches since we only consider the information inside the image not outside the image. So, we need to combine the size of the image and the window size to reduce the size of the patches when it reaches at the boundary. Otherwise, we just need to get the range of the patch which size is the same as window size and centered at point.

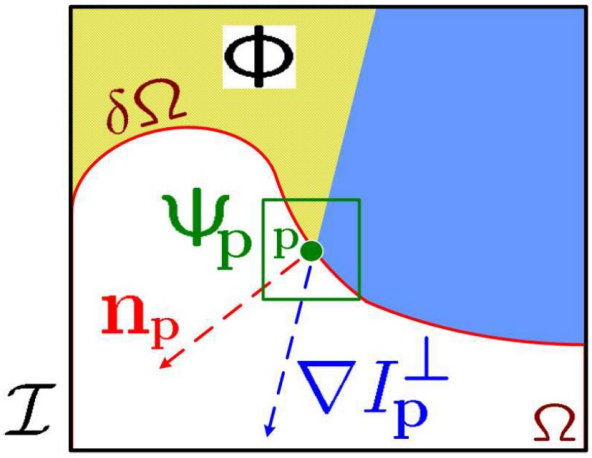
4) Compute confidence

   Then we need to get the confidence value of the point on the front fill (boundary of mask). First, we need to get a window size patch centered at that point using the patch method. Then we need to add all the confidence value in this region together. And divide by the area of that patch region to get the confidence value per pixel. We treat this value as the confidence value of the center point.

5) Compute normal to the contour of the target region and the isophote (direction and intensity) at point p

   To compute the isophote at point p, we will need to first get the gradient corresponding to original image's x and the gradient corresponding to original image's y. And since we only care about source region here. We need to set the

index where the mask is covered to 0. Also, to compute the normal we will need to get the gradient corresponding to the mask's x and the gradient corresponding to mask's y. And then we need to compute the norm corresponding to both gradients. We will get the normal by gradient divide the norm. Finally, according to the formula to compute data, we need to calculate the norm with both isophote and the normal. Then divide by the alpha to do the normalization. (alpha is a normalization factor) We will choose 255 since we are dealing with grey-level image in this problem. And we finally finish the data term computation.



6) Compute the priority

According to the formula, we can compute the priority of each point on the front fill by multiply the confidence of the point and the data term of that point.

Propagating texture and structure information

1) Find suitable patch in source region

In this step, we need to find the suitable patch which has the same patch size with the given patch Ψp. In order to make sure the patch is located in source region we can use mask image to locate the desired patch area and sum the pixels value up to check if the result is equal to zero.

2) Compute the sum of squared differences

After finding a patch Ψq, we need to calculate the SSD between this patch Ψq and the patch Ψp. Same as above, we use mask image to locate all the pixels which is filled in source region then use this location to compute the squared differences of the pixels in Ψp and Ψq.

3) Find the best match patch

In this step, we have collected all the suitable Ψq and their corresponding SSD. We need to pick up the smallest distance between generic patch Ψp and Ψq and use this source exemplar $\Psi\bar{\mathbf{q}}$ to fill the image.

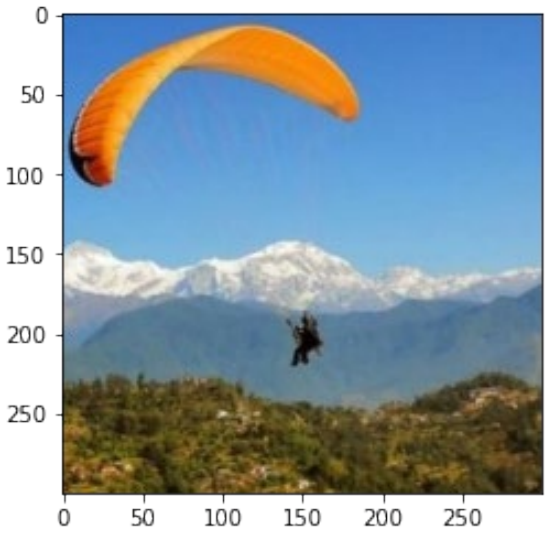
4) Fill the image

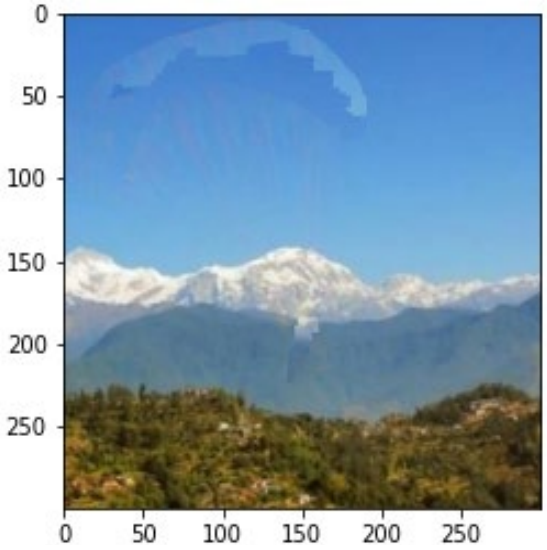
We use mask image to locate all the patch Ψp pixels which is unfilled in target region then we copy the matched pixels in $\Psi \bar{\mathbf{q}}$ to fill this unfilled area. Since the image is three channels, we need to do this channel by channel.

Updating confidence values

After the patch has been filled with new pixel values. We need to update the confidence in that patch. The confidence of the patch area (each pixel in the patch area) will be the same as the point p.

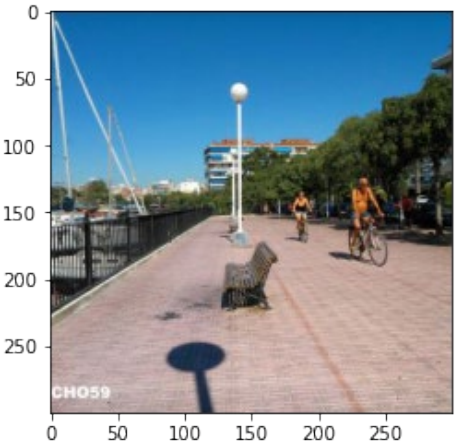
Result

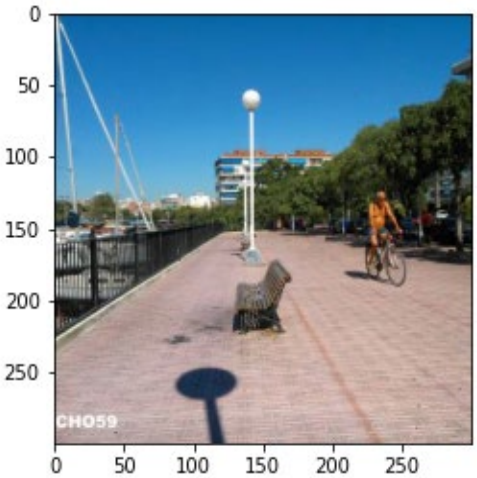

0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250


0
25
50
75
100
125
150
175
0
50
100
150
200
250


0
25
50
75
100
125
150
175
0
50
100
150
200
250


0
50
100
150
200
250
CHO59
0
50
100
150
200
250


0
50
100
150
200
250
CHO59
0
50
100
150
200
250


BRONX ZOO
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600


BRONX ZOO
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600

Conclusion

Region Filling and Object Removal by Exemplar-Based Image Inpainting is a novel and efficient algorithm that combines the advantages of texture synthesis algorithm and inpainting techniques. This algorithm does have any data base requirement and do not require any machine learning and deep learning technique. The cons of this method are it can only be used on images which has a low frequency background (simple background). Since we filled the mask part depends on the information in the source region. So, in order to use this algorithm, there must be some place in the source image contain the information of the missing part (target region). Otherwise, it is not possible for this algorithm to fill the target region correctly. If we want to solve this problem, we will need to follow the paper "Scene Completion Using Millions of Photographs" from 2007. Which tries to use data-driven method to solve 2 main problems of Region Filling and Object Removal by Exemplar-Based Image Inpainting. First problem is when the background is too complicated the algorithm are not able to find similar patches in the source region of an image, this paper uses many images on the Internet to provide as data so we can find similar patches in these images rather than just the source region. The second problem is low efficiency of on the filling, the paper proposes to directly take complete piece from other images which is the similar patches and copy to the target region. It is a way to fill the gaps directly from other images. However, using the images on the internet also brings new

PROBLEMS: HOW TO FIND SIMILAR IMAGES ON THE INTERNET TO BE THE DATA SOURCE OF THE ALGORITHM. HOW TO MAKE SURE WE ARE TAKING THE CORRECT SIMILAR PATCHES; IT WILL BE AWFUL IF WE TAKE A PICTURE OF A DOG AND FILLED IN AS A SIMILAR PATCH OF A MOUNTAIN. AND DIFFERENT IMAGE HAS DIFFERENT LIGHTNING. SOME ARE DARK, AND SOME ARE LIGHT, BUT THE OBJECT IN THE IMAGE MIGHT BE REALLY CLOSE TO THE ONE IN THE TARGET REGION. WE ALSO NEED TO FIND A WAY TO AVOID THESE KINDS OF SITUATION HAPPENED. THIS PAPER ALSO REQUIRED A HUGE NUMBER OF IMAGES SINCE THERE ARE MILLIONS OF KINDS OF OBJECT WE NEED TO THINK ABOUT. THERE ARE MANY INPAINTING METHODS EXIST DEPEND ON THE DESIRED GOAL AND TYPE OF IMAGE BEING TREATED. DO THE INPAINTING ON DIFFERENT KINDS OF IMAGE CAN BE COMPLETELY DIFFERENT. IN MY OPINION THE PHYSICAL INPAINTING WILL ALWAYS BE A GOOD WAY TO DO SINCE HUMAN BRAIN IS MORE POWERFUL THAN COMPUTER. DIGITAL INPAINTING WILL WORK ONLY ON SOME SITUATION AND IT WILL DEFINITELY TAKE TIME AND HUGE NUMBER OF RESOURCES.

Author contribution

Hongyu Chen carried out the research, Computing patch priorities and Updating confidence values in the method. Ming Gong carried out Propagating texture and structure information in the method. All other parts are done together, and all authors participated in each part of the project. All authors read and approved the final manuscript.

References

Region Filling and Object Removal by
Exemplar-Based Image Inpainting
A. Criminisi*, P. P´erez and K. Toyama
Microsoft Research, Cambridge (UK) and Redmond (US)
IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 9, SEP 2004
https://en.wikipedia.org/wiki/Inpainting
https://github.com/Shubham-Sahoo/Image_inpainting (for the images)