Math 354: Final Project Installment II

Fred Park

December 2019

1 Image Inpainting

Image inpainting is the problem of filling in missing information into an image. This amounts to restoring an image that has been scratched, degraded, or even missing pixels lost during compression and transmission.

Figure 1: Example of a degraded image and a result from inpainting

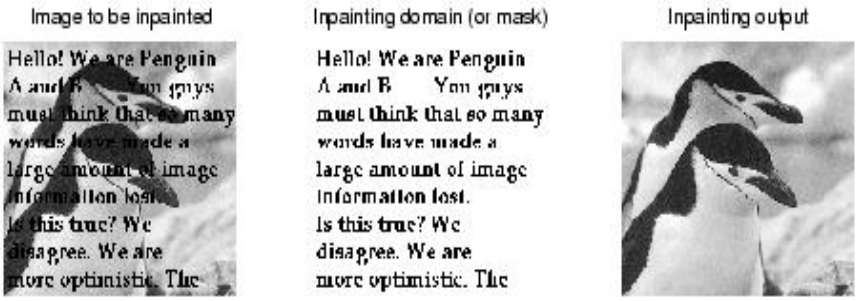

Figure 2: Example of a degraded image and a result from inpainting

Examples of inpainting can be seen in the above figures where a family photo is scratched and then recovered in Fig. 1. In Fig. 2, someone has added graffiti to the penguins! Luckily, computer vision is to the rescue and the penguins have been restored! Your goal in this project is to degrade your image by scratches and/or text and then restore the image. The location of the scratches or text, also known as the inpainting domain or mask must be supplied. Let f be a given image of size $m \times n$. Then the mask representing the inpainting region is in the form of an indicator function image of size $m \times n$ where missing pixels have value 0. They have value 1 if they correspond to non-missing pixels.

If you are given a degraded image f that has missing values, we can consider the missing regions as being domain D_I and the pixels that are not missing as D_O. Thus, the image domain $D = D_O \cup D_I$. The

Total-Variation Inpainting model then becomes:

$$\min_u J[u] = \min_u \frac{1}{2} \int_{D_O} (f-u)^2 \, dx \, dy + \lambda \int_{D=D_O \cup D_I} |\nabla u| \, dx \, dy. \tag{1}$$

Here, we only fit data where it exists, i.e. on D_O, but require that TV be minimized at all pixel values, including missing ones!

Follow the steps below on a grayscale image of your choice.

1. Let f_c denote your clean original image. Create an inpainting mask g by making an image of same size as f_c, but with pixel values 1. Use either MS paint or any other image manipulation program to draw scratches (not too wide) or text onto g. Make sure they are done in black. Import this image g to matlab and make sure it is grayscale. Use 'rgb2gray' command if it is not. Double check that the values are 0 in inpainting regions and 1 elsewhere.

2. Degrade f_c to create f by the following operation $f = f_c .* g$. This should make inpainting regions have value 0 e.g. black, and leave other regions unchanged.

3. To process this image via the TV model, you will need to fill in the missing information with a random guess. Create a random matrix R with values between 0 and 255 in matlab as 'R = randi([0,255],m,n)' where f_c is size $m \times n$.

4. Next, set values in inpainting domain to random values by the following process in matlab:

   - $g2 = -1 * (g - 1)$
   - $f_0 = g2 .* R + f$.

   Now, f_0 will be your initial image. You can then use the TV model to inpaint. Note the following slick way of adjusting the inpainting domain:

   $$\min_u \frac{1}{2} \int_{D_O} (f-u)^2 \, dx \, dy + \lambda \int_{D=D_O \cup D_I} |\nabla u| \, dx \, dy = \tag{2}$$

   $$\min_u \chi_{D \setminus D_I} * \frac{1}{2} \int_{D} (f-u)^2 \, dx \, dy + \lambda \int_{D} |\nabla u| \, dx \, dy. \tag{3}$$

   Where $\chi_{D \setminus D_I} = 1$ if away from the inpainting domain and 0 otherwise. Note, this indicator function is just g that you created in matlab. Thus to enforce data fitting away from the inpainting domain i.e. where data is present, you need only multiply by g.

5. You will need to implement the following PDE which is gradient descent on the TV energy:

   $$u_t = g * (f_0 - u) + \lambda * \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) \tag{4}$$

   where you use $u(x, y, 0) = f_0$ as an initial guess. Use this method to inpaint your given image. Display the results.

6. Extend your inpainting to the color version of your image by inpaining each of the R,G, and B channels independently and putting them back into a color image.

7. Extra credit: can you think of a way to extend the TV inpainting model to digital-zooming/super-resolution? If so, implement it and display the results!