Advanced Topics in Computer Vision and Deep Learning (EEEM071)

Week 4: Image Completion and Compositing

Dr Xiatian Zhu

Introduction

In this lab you will experiment with techniques to remove objects from / repair images (image completion) and to add objects into images (image compositing).

Please visit the **lab sheets section on SurreyLearn** and download **EEEM071_labcode_weeks2-4.zip** into your home space.

You should ensure you attended the lectures on image compositing and completion in Week 2 before attempting this worksheet. If you didn't then you should watch on Panopto first.

1. Getting Started (you may already have done this if you completed the warping labs)

Unzip the lab code into a folder in your home space. Open up a terminal window using Ctrl-Alt-T.

Assuming you downloaded the file into the top folder of your homespace, type

```
unzip EEEM071_labcode_weeks2-4.zip
```

This will create a folder EEEM071 and within that, a subfolder warping. Start MATLAB and ensure that this folder is added to your Matlab search path (Use File -> Set Path.. -> Add with subfolders and pick the EEEM071 folder). If you are unsure how to do this then check with the lab tutors. If MATLAB is complaining that it cannot find functions when you perform these lab exercises it is probably because your path is not set correctly. You can also try typing directly within MATLAB:

```
addpath(genpath('~/EEEM071'))
```

```
cd ~/EEEM071
```

2. Image Completion – Patch based (Freeman Efros)

Open up the code for the demo implementation of Freeman Efros (patch based) inpainting

edit ipv_freeman_efros.m

As configured, the demo will load up an image of a dog and a binary mask specifying the location of the dog region. It will inpaint that masked region using patches sampled from elsewhere in the image.

Line 20 controls the size of the patches (it is currently 11).

Explore the impact of varying patch size by varying the number in this line e.g. change from 11 to 31 or 71.

The bigger the patch size the fewer the patches that can be sampled from non-masked regions of the image. Bigger patches can preserve structure in the hole being in-painting, but can produce edge artifacts. Smaller patch size can lose structure but avoids such artifacts – it is also slower.

3. Image Completion – Poisson Image Completion (Image Inpainting)

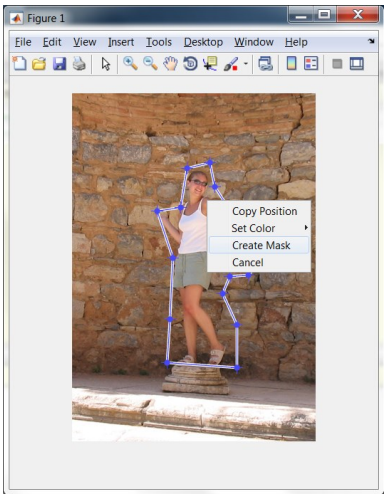
Open up the code for the demo of poisson image completion, which solves for the Laplacian constraint across the image to ensure smooth inpainting.

edit ipv_inpaint_image.m

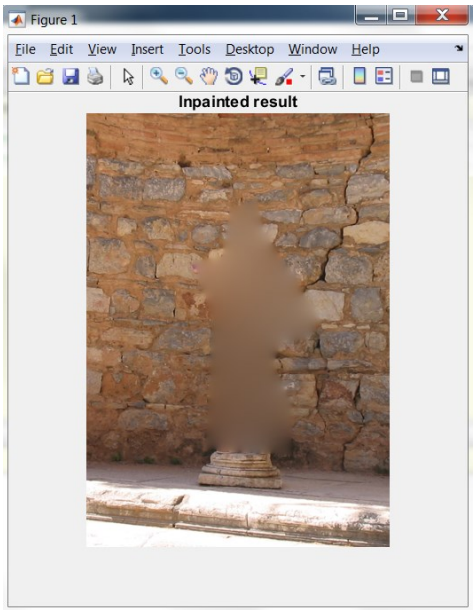
As configured, the code will load up a picture of a person and invite you to draw around the person to identify the region to be filled. On lab machines this can be slow given the size of the image you may wish to downsample it first by adding the following line after the imread

```
img=imresize(img,0.5);
```

Run the code and mark the region to inpainting by left clicking. When you are done right click to close the selection polygon. In earlier versions of MATLAB this will begin the inpainting, but in recent versions you will need to right click inside the polygon you drew and select 'Create Mask'.



You will know the code is running when you start seeing percentages scroll past from 0% to 100%. It may take a few minutes to complete the inpainting.

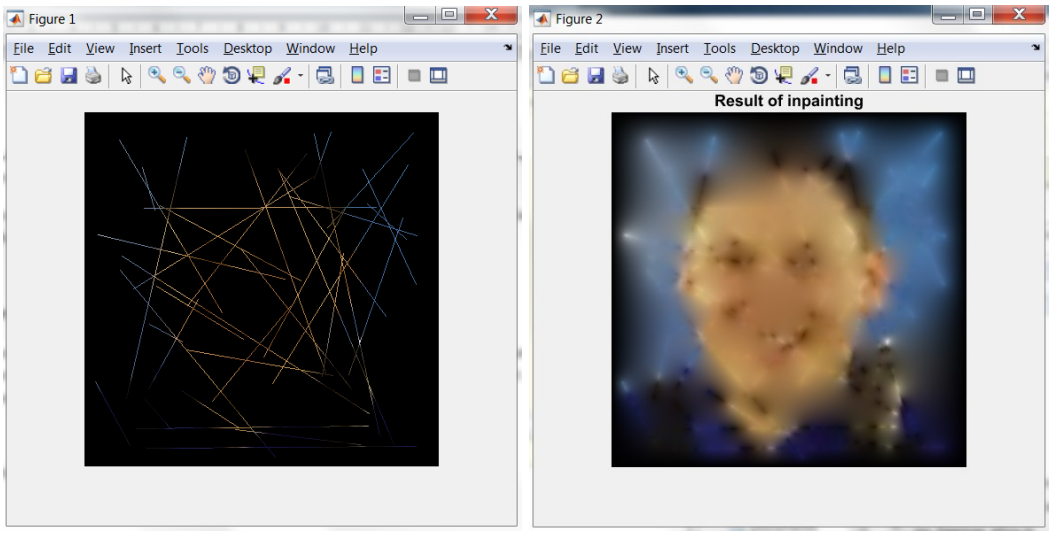


Now load the same image up in the Freeman Efros code from part 1 of this lab sheet and compare the results (or alternatively load the dog image in this code by modifying the imread command).

What can you say about the texture present in the in-painted region with the Poisson/Laplacian inpainting approach vs. the Freeman Efros patch based approach?

4. Image Completion – Poisson Image Completion (Image Reconstruction)

Open up the code for the demo of poisson image reconstruction – it is simply another application of this inpainting algorithm. As configured, an image of a face will be loaded but masked out for but a few pixels (defined by random lines across the image).



The Poisson inpainting function is again called, but this time the black region is considered the mask and the remaining pixels the source data to interpolate from. The process is identical to part 3 of this sheet – but there is a lot more 'mask' and fewer data points to reconstruct from. Nevertheless a reasonable reconstruction of the face is produced – showing how much visual redundancy exists within an image. This code will take several minutes to run. Try the code out on other images from the Internet – ensure that they are no bigger than 200 x 200 pixels (or use imresize in the code to reduce their size). Remember you can check the size of the image using size(img)

5. Poisson Image Compositing (Image Reconstruction)

Open up the code for the demo of poisson image compositing.

edit ipv_poisson_composite_demo.m

Run the code, which will composit the masked region of the dog.jpg image (defined in dogmask.png) onto the pool.jpg image. Examine the linear system constructed within the function ipv_poisson_composite.m and compare this to the system constructed in the image completion demo (ipv_inpaint_image).

What are the key differences (check design matrix A and the output vector b)?

Currently this code does not do a good job at the perimeter of the image (where the Laplacian kernel would overlap the image boundaries). How could the code be fixed to better handle these edge cases?