MIT EECS 6.815/6.865: Assignment 13:

Make your own Assignment

Due Thursday May 17 at 9pm

This is a document in progress. I'll refine it in the next few days.

1 Summary

I tried to rate the difficulty of the various techniques. It is just an estimate.

2 Normal

2.1 Image resizing using seam carving

Make an image smaller by removing pixels that are less significant.

A little bit of Sobel gradient to create an energy function that says which pixels can be removed, then standard dynamic programming to find "seams" that connect easily-removable pixels.

`http://www.faculty.idc.ac.il/arik/site/seam-carve.asp`

`https://stellar.mit.edu/S/course/6/sp11/6.815/homework/assignment6/`

2.2 Dehazing

Remove haze in photography. Locally compute the minimum (scipy has a min filter) and use it to guestimate what to subtract from the image.

`http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5206515&tag=1`

Ignore the soft matting from that paper. Replace it by a cross-bilateral filter, which is easier to implement.

2.3 Deconvolution (easy to implement, requires a little bit of math to understand)

Given a blurry image, invert the blur process to yield a sharp image. If the blur process is described by the convolution operator A and your input blurry image is y, you want to solve for $Ax = b$, which is very similar to the Poisson equation we solved. You only need to replace the Laplacian operator by the blur kernel.

Make the process more stable by adding a gradient-based regularization. To avoid amplifying the noise, minimize:

$$\min ||Ax - y||^2 + \lambda ||\nabla x||^2$$

where λ is a parameter. This is essentially the weighted solution of the original convolution and the Laplace equation. All you need is modify the computation of your residual.

Extra-credit (easy to implement, hard to understand): Use reweighted least square to simulate an L1 regularization.

2.4 texture synthesis

Given input texture example, generate a similar-looking but potentially bigger texture.

http://graphics.cs.cmu.edu/people/efros/research/EfrosLeung.html

2.5 Stainglass by numbers

Similar to the painterly rendering, except that we store some notion of depth for each pixel (z buffer) and splat 3D cones centered at y, x.

Use bumpy operations and logical indexing (such as `z[z>newZ]=newZ[ z>newZ]]`) to make things tractable.

http://dl.acm.org/citation.cfm?id=97902

2.6 Denoising by wavelet coring

`http://www.cns.nyu.edu/pub/lcv/simoncelli96c.pdf`

2.7 Video texture

`http://www.cc.gatech.edu/cpl/projects/videotexture/SIGGRAPH2000/index.htm`

2.8 Flash no flash photograpjy

Implement Petshnigg's version first, it is simpler because it doesn't seek to deal with shadows.

`http://dl.acm.org/citation.cfm?id=1015777` `http://people.csail.mit.edu/fredo/PUBLI/flash/index.htm`

2.9 Separation of direct and indirect lighting effects

`http://www1.cs.columbia.edu/CAVE//projects/separation/`

You can borrow a projector.

2.10 In paining with big database

http://www.cs.brown.edu/courses/csci1950-g/asgn/proj4/
http://www.cs.brown.edu/courses/csci1950-g/asgn/proj4/resources/SceneCompletion.pdf

2.11 Hybrid images

http://cvcl.mit.edu/hybridimage.htm
http://www.cs.illinois.edu/class/fa11/cs498dh/projects/hybrid/ComputationalPhotography_ProjectHybrid.html

2.12 Color 2 gray

http://www.cs.northwestern.edu/~ago820/color2gray/

2.13 NL means denoting (easy but slow)

http://bengal.missouri.edu/~kes25c/nl1.pdf http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

2.14 Tour into the Picture

http://graphics.cs.cmu.edu/courses/15-463/2005_fall/www/Papers/TIP.pdf
http://graphics.cs.cmu.edu/courses/15-463/2010_fall/hw/proj4g/
Start with just your homography code

2.15 Morphable face models and caricatures

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.49.9275
http://web.mit.edu/emeyers/www/face_databases.html
http://vasc.ri.cmu.edu/idb/html/face/

2.16 Pyramid image alignment

2.17 Time lapse manipulation

http://dl.acm.org/citation.cfm?id=1276505 or http://www.csbio.unc.edu/mcmillan/pubs/sig07_Bennett.pdf
Median or min across time, dynamic programming.

2.18 Patch match

Probably slow in python
Called content-aware fill in Photoshop.
http://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/

2.19 Detecting copy-pasting

http://www.cs.dartmouth.edu/farid/Hany_Farid/Papers/Entries/2011/3/20_Exposing_Digital_Forgeries_by_Detecting_Duplicated_Image_Regions.html

Extend to detecting Poisson image cloning. Easy but probably slow.

2.20 Tour into the Picture

http://graphics.cs.cmu.edu/courses/15-463/2007_fall/Papers/TIP.pdf

Just start with your homography code. It's OK if you have to manually indicate where the four corners should got. Then add the notion of vertical billboard.

2.21 Dual photography

http://graphics.stanford.edu/papers/dual_photography/

You can borrow a projector.

2.22 Hockney collage from a single image

Create a collage in the style of David Hockney http://www.hockneypictures.com/works_photos.php

See an example of software at http://bighugelabs.com/hockney.php

I'd start with the NPR algorithm to scatter a bunch of window locations across the image according to the importance map.

2.23 Photographic style transfer

http://people.csail.mit.edu/soonmin/photolook/

Focus on histogram matching of the bilateral filter components, and in particular the notion of textureness, which is not very different from the sharpness in pset 11. Don't worry too much about the post-processing and the gradient preservation unless you have time.

2.24 Bayesian matting

See last year's assignment description. https://stellar.mit.edu/S/course/6/sp11/6.815/homework/assignment5/

2.25 Salovon-style art

Jason saloon does amazing algorithmic art, usually based on the combination of many photos. http://salavon.com/work/. See also http://blag.xkcd.com/2010/05/03/color-survey-results/

Use the flickr API http://www.flickr.com/services/api/ to reproduce images such as his color wheel http://salavon.com/work/color-wheel/image/

409/ by querying flickr for images with color names such as "red". Start with a flat rectangular version. See http://en.wikipedia.org/wiki/Color_term for more inspiration and create a multilingual comparison.

Aggregate many photos of a given landmark ("Statue of Liberty") or type of image ("landscape") or ("portrait") in the spirit of http://salavon.com/work/Homes/grid/2/, http://salavon.com/work/Portrait/grid/1/. Maybe cluster the results somehow to create multiple composites.

3 Harder

3.1 View morphing

Combine homographies and morphing!

http://www.cs.washington.edu/homes/seitz/papers/sigg96.pdf The paper is not completely easy to read but it's cool.

3.2 inpainting (not so hard to implement but not easy to understand)

Given an image and a masked region, reconstruct plausible values inside the mask by interpolation. http://en.wikipedia.org/wiki/Inpainting http://www.tecn.upf.es/~mbertalmio/restoration0.html http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5593835

I suggest you combine the Poisson solver and the structure tensor.

3.3 Colorization using least square optimization

Given a greyscale image and a sparse set of color indications given by the user, propagate these colors to the full image.The interpolation takes into account the content of the greyscale image and tends to have color changes only where the intensity changes.

See http://www.cs.huji.ac.il/~yweiss/Colorization/

You can do a full linear algebra version by forming the sparse matrix and use your Poisson code but with BF

You can alternatively modify your Poisson image editing code and replace the Laplacian kernel by an input-dependent kernel.

3.4 Perceptual metric for photo retouching

http://www.cs.dartmouth.edu/farid/Hany_Farid/Papers/Entries/2011/6/6_A_Perceptual_Metric_for_Photo_Retouching.html

Use your morphing code for computing the warp field.

The tricky part is to get data.

3.5 Hockney collage from multiple images

`http://webee.technion.ac.il/~lihi/Demos/AutoJoiners.html`

Modify your automatic panorama matching and perform automatic layout using least square optimization, trying to use the average translation vector between pairs of images.

Speed could be an issue.

3.6 Local Laplacian

`http://people.csail.mit.edu/sparis/publi/2011/siggraph/`

Probably very slow in Python.

3.7 Image deformation using moving least squares

Probably slow. Use numpy to solve each least square problem.

`http://faculty.cs.tamu.edu/schaefer/research/mls.pdf`

3.8 Non-Photorealistic rendering using extended differences of Gaussians

optical flow + retiming

NPR decarlo

Gradient tone mapping

guided image filtering

graph cut / grab cut http://www.csd.uwo.ca/ yuri/Abstracts/iccv01-abs.html

Ramesh multiflash

Shree's LCD in front of sensor? Play with film/transparency

3.9 Interactive digital photomontage

http://grail.cs.washington.edu/projects/photomontage/

Relighting with multiple photographs

compressive sensing

`http://graphics.stanford.edu/papers/glare_removal/`

Laplacian matting

Inpainting with Heeger and Bergen

Fake tilt shift

Photometric stereo

view morphing

Auto perspective correction Image mosaics

magic scissors

Rolling shutter correction

Eulerian video magnification