MIT EECS 6.815/6.865: Assignment 13:

Make your own Assignment

Due Thursday May 17 at 9pm

1 Summary

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

2 Deliverables

Please turn in a PDF file (you will lose points for any other format) describing your work and showing results. Give a short description (half a page to a page) of the algorithm that could be understood by someone who has taken the class but has not heard of that particular technique before.

Describe your struggles and solutions.

Unless there is a good reason, you should have run your algorithm on at least two different inputs, including at least one that you created. The more, the better.

Include running times and other stats when appropriate. In general, anything that sheds light on the technique and its performance if good.

The total document should probably be about 2-4 pages.

Include your name at the beginning.

6.865 students To get full credit for 6.865, you need to implement a little more than what is described below or provided additional analysis. In the rest of the document, when I say "to get full credit", I mean to get full credit in 6.815, except in the "harder" section. In many cases, additional components are described and you can just pick from there. It's even easier for assignments from previous offerings: just do the grad version. In general, if you're unsure, email me.

3 Easy

3.1 texture synthesis

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

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

http://en.wikipedia.org/wiki/Texture_synthesis
http://www.ics.uci.edu/~fowlkes/class/cs116/hwk3/index.html
http://www.cs.ubc.ca/~woodham/cpsc425/assignments/hw4/hw4.html
http://cs.nyu.edu/~fergus/teaching/comp_photo/assign3.pdf
For full credit, perform hole filling.

3.2 Flash no flash photography

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

Very similar to the tone mapping assignment.

We recommend you use the bilateral grid code. It won't take much to make it do cross bilateral filtering.

Just implementing Petschnigg's approach won't give you full credit. For that, implement either a shadow fix or an alignment procedure.

Data is available on Elmar's page http://maverick.inria.fr/Publications/2004/ED04/index.php but we highly encourage you to capture your own. You can even borrow a camera that takes a flash and a no-flash image in succession.

3.3 Hybrid images

Generate images that look different from a close vs. a large distance. http://cvcl.mit.edu/hybridimage.htm

http://www.cs.illinois.edu/class/fa11/cs498dh/projects/hybrid/ComputationalPhotography_ProjectHybrid.html

To get full credit, show a plot of the frequency content of the two input images, the hybrid image, and its two components, and experiment with color.

Include at leads two examples.

You might want to use your warping code to align the two images.

4 Normal, with instructions

4.1 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/

4.2 Bayesian matting

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

4.3 Inpaining with big database

Replace a masked area in an image by content found in a similar image from a big database of pictures.

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

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

4.4 Tour into the Picture

Create 3D animations from a single image and a few clicks!

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.

http://graphics.cs.cmu.edu/courses/15-463/2010_fall/hw/proj4g/

5 Normal

5.1 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.

5.2 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.

Or implement the Richardson-Lucy version `http://en.wikipedia.org/wiki/Richardson%E2%80%93Lucy_deconvolution`

5.3 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.

That is, store an array of z values. For each brush stroke, the idea is to render a 3D cone center at the stroke and extending away from the canvas. That is, the z value for a pixel is equal to its distance from the center of the stroke. Only the closest cone is visible at a point, so you need to test for each pixel if the new cone is closer than the stored value. If yes, update both your z array and the color array.

Vary the narrowness of the regions by applying a different scale factor to compute depth for each cone.

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

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

Forget about the relaxation part, unless you're very motivated.

5.4 Denoising by wavelet coring

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

5.5 Video texture

Create videos that loop perfectly, and even graphs of transition for non-repetitive playing.

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

5.6 Color 2 gray

Turn a color image into a black and white one while preserving edges as well as possible. Start from your Poisson code and the max of the gradient across the three channels. Then be smarter. `http://www.cs.northwestern.edu/~ago820/color2gray/`

5.7 NL means denoting (easy but slow)

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

5.8 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/

5.9 Pyramid image alignment

Implement a coarse-to-fine version of image alignment.

Extend it to the median pyramid by Greg Ward http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5

Or go full Lucas-Kanade, e.g. http://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method

5.10 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.

For full credit, implement at least the dynamic programming version with two different metrics.

Be wary of computation time if using python. Computing your results could take a while unless you manage to cast most operations as built-in numpy functions.

5.11 Patch match

Probably slow in python

Called content-aware fill in Photoshop.

http://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/

5.12 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

To get full credit, extend to detecting Poisson image cloning. Easy but probably slow. Try to accelerate using convolution/correlation.

5.13 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.

You'll get full credit if you get the transfer of global contrast and textureness. Post-processing effects and gradient preservation are extra credit (although it shouldn't be too hard given your code from pset 11).

5.14 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.

You can always do old-style image mosaics, but at least try to match the edge structure of the individual super pixel images to that of the target photo. `http://en.wikipedia.org/wiki/Photographic_mosaic`

To get full credit, create at least two of these (e.g. the color wheel and the landmark), or one with extra bells and whistle (e.g. landmark+clustering, multilingual color wheel).

5.15 Anisotropic diffusion

`http://en.wikipedia.org/wiki/Anisotropic_diffusion`

Alternative to the bilateral filter, but based on PDEs.

5.16 Laplacian pyramids

The generalization of the 2-scale blending that we did, which can also be used for the apple/orange trick or the focal stack fusion.

`http://persci.mit.edu/pub_pdfs/RCA85.pdf` `http://www.cs.princeton.edu/courses/archive/spr04/cos429/papers/burt_adelson.pdf`

6 Normal, but requires hardware

6.1 Separation of direct and indirect lighting effects

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

You can borrow a projector.

6.2 Dual photography

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

You can borrow a projector.

6.3 Relighting with multiple photographs

Take images with a static camera (on tripod) but with light coming from different directions. Then use these images to create new images using weighted combinations. See e.g. http://gl.ict.usc.edu/Research/LS3/ for inspiration, but don't try to reproduce all their crazy stuff.

7 Harder

7.1 View morphing

Combine homographies and morphing! Add a homography to make your morphing respect 3D structure better. In particular, given two views of the same 3D object, this method guarantees that the morphing sequence is equal to a 3D rotation around the object.

http://www.cs.washington.edu/homes/seitz/papers/sigg96.pdf

The paper is not completely easy to read but it's cool.

7.2 Lens correction

Calibration and correction of radial distortion and vignetting. See the slides.

Vignetting is more tricky than it seems. replace the polynomial by a table and a piecewise-linear function.

7.3 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. First compute the structure tensor, ignoring pixels in the masked region. (The local weighted average in the second Gaussian blur should ignore pixels in the region. The easiest way to do it is to set them to zero, and keep track of the sum of the weights in the sum by blurring the mask itself.). Then run Poisson with a flat source to interpolate the structure tensor inside the region. Once you have an interpolated structure tensor, use it to interpolate color values, for example using anisotropic diffusion http://en.wikipedia.org/wiki/Anisotropic_diffusion, where the diffusion is guided by the 2x2 structure tensor matrix at each pixel.

7.4 Bundle adjustment

Refine your panorama with a global optimization, including radial distortion optimization. Use numpy's optimization functions and, if you're hardcore, get closed-form derivatives.

http://sse.tongji.edu.cn/linzhang/computervision/projects/image%20alignment%20and%20stitching%20a%20tutorial.pdf

http://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Triggs00.pdf

Given your inlier pairs for the various images you have, optimize free parameters to minimize the reproduction error. Start by writing this error function computation. Make it "robust" by clamping it to a max value (if the reproduction is more than XXX pixels away, only pay the penalty for XXX pixels).

The free parameters are typically the focal length and three 3D rotation angles for each photo, 3D coordinates for each point (of course up to scale, since we can't know the distance to the camera), and some radial distortion parameter (which you should forget about at the beginning). Good initialization is critical. Start from your homographies, guess focal length (e.g. 30mm for a 24x36mm sensor), and look at the maths of projection from the cylindrical panorama assignment.

Start with a brute force approach. This will be enough to get full credit. Demonstrate that your method can, e.g. converge to the correct focal length (use a pano where you know the focal length, but initialize with a wrong one). Your first read should probably be section 5.1 of Szeliski's survey above.

http://web.me.com/dellaert/07F-Vision/Schedule_files/08-Stitching.ppt.pdf

http://research.google.com/pubs/pub37112.html

7.5 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.

7.6 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.

7.7 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.

7.8 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.

7.9 Local Laplacian

What replaced the bilateral filter in Camera RAW/Lightroom.

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

Probably very slow in Python.

7.10 Image deformation using moving least squares

Related to warping. Specify a sparse set of point displacement and interpolate intelligently by solving a least-square problem at each point. In particular, it allows the interpolation to have some underlying notion of class of transformations, such as angle preservation.

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

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

An extension uses biharmonic energies

http://igl.ethz.ch/projects/bbw/

7.11 Non-Photorealistic rendering using extended differences of Gaussians

http://dl.acm.org/citation.cfm?id=2024700

Use the structure tensor and the eigenvector business from pset 12 to get the flow alignment, or use the Sobel operator if you prefer.

Implement both adaptive thresholding and flow alignment and you'll get full credit.

7.12 Multiflash camera

http://web.media.mit.edu/~raskar/NprCamera/

7.13 Guided image filtering

http://research.microsoft.com/en-us/um/people/kahe/eccv10/eccv10supp.pdf

7.14 graph cut / grab cut

Foreground/background extraction

http://www.csd.uwo.ca/~yuri/Abstracts/iccv01-abs.html http://research.microsoft.com/apps/pubs/default.aspx?id=67890

use python's library http://code.google.com/p/python-graph/

Implement graph cut segmentation with a combination of data and edge term and you'll get full credit. Grab cut is extra credit. Don't worry about the mixture of Gaussian part and keep the same histogram approach.

7.15 Interactive digital photomontage

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

7.16 Reducing veiling glare for higher-dynamic-range imaging

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

7.17 Artistic screening

http://www.iro.umontreal.ca/~ostrom/publications/pdf/SIGGRAPH95_ArtisticScreening.pdf reproduce shades of grey with micro patterns of your choice.

really hardcore: color version http://www.iro.umontreal.ca/~ostrom/publications/pdf/SIGGRAPH99_MultiColorDithering_600dpi.pdf

7.18 Laplacian matting

Separate foreground and background. The derivation is a little scary but the implementation can be simplish. http://www.wisdom.weizmann.ac.il/~levina/papers/Matting-Levin-Lischinski-Weiss-CVPR06.pdf

7.19 Inpainting with parametric texture synthesis

http://people.csail.mit.edu/torralba/courses/6.869/lectures/lecture5/heegerbergen.pdf

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

7.20 More inpaining

http://www.ittc.ku.edu/~potetz/EECS_841_Fall08/Readings/Lecture_39_CriminisiPerezToyama_Inpainting_04.pdf

7.21 Eulerian video magnification

Amplify subtle motion in videos. The paper is not even public yet. See `http://people.csail.mit.edu/fredo/tmp/videomag.pdf` and `http://people.csail.mit.edu/fredo/tmp/videomag_new.mov`

For full credit, implement the multi scale Laplacian pyramid. You can make the temporal filtering easier by just using difference of Gaussians.