MIT EECS 6.815/6.865: Assignment 10:

Make your own assignment

Due Wednesday December 13 at 9pm

1 Summary

You will choose your own assignment for this last problem set. This should be about the difficulty of an average problem set.

You can choose a problem set from a previous year (that was not covered this year). Alternatively, you can create your own problem set (possibly from the given list). In addition to the coding component, you will need to produce a short write-up.

2 Make Your Own Assignment

1 Make your own assignment! Turn in your code and write-up to the submission system. See below for details. Choose an assignment from:

- A previous year problem set - section 3.1.
- Make your own from the list in sections 3.2 - 3.6.
- Or literally do your own thing. (If you choose this, let us know on piazza, so we can tell you if your idea is reasonable.)

2.1 Deliverables

This assignment will have two deliverables: (1) the code and (2) a write-up of the assignment. See below for details.

2.1.1 Write-Up

Please turn in a PDF file (you will lose points for any other format) describing your work and showing results. Place the write-up in the folder `asst/write-up` of your submission. Your write-up should contain the following:

- Include your name at the beginning.
- Clearly state what you did and why you chose it. For example, if you choose a pset from a previous year indicate its name and why you thought it would be interesting to work on it.

- Background section with a short summary of at least 3 papers related to what you are working on.
- 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 what you implemented, what it does and what was difficult about implementing it.
- Add appropriate figures to the write-up about your test cases (*i.e.,* intermediate steps).
- Include useful statistics if applicable, *e.g.* running times, averages, *etc.* In general, anything that sheds light on the technique and its performance is good.
- Run your algorithm on at least two different inputs including at least one you created. The more the better. Add appropriate figures to your write-up showing the inputs and the results.
- The total document should be between 2-4 pages.

2.1.2 Code

As usual, include your code. We provided you with a skeleton zip file containing some code from previous assignments and a `Makefile`. Add your function(s) signature in `a10.h` and implement them in `a10.cpp`. As before, your test cases should be in `a10_main.cpp`. Feel free to add additional source and header files; if you do, you'll need to add them to the `Makefile`. See previous assignment `Makefile` for details on how to do that.

Submit the code to the submission system and make sure that your code runs on it. If you have any trouble or feel like the submission system is not sufficient for your needs (*e.g.*, you need to use an external library or you want to use Halide), let us know.

Your code should contain the following:

- At least 5 well-documented test cases testing intermediate parts of your algorithm. This should be similar to the test cases we have provided you on previous assignments. Include your test cases in the `a10_main.cpp` file. Include figures in the write-up if appropriate.
- Run your algorithm on at least two different inputs including at least one you created. The more the better.
- Print out, useful statistics if appropriate, *e.g.* running times, averages, *etc.*

3 Assignment Lists

Here we provide a few choices on assignments. Section 3.1 contains the previous year's assignment. Sections 3.2 - 3.6 contains additional ideas, which we have (approximately) separated according to difficulty.

6.865 Students. To get full credit for 6.865, you need to implement a little more than what is described below or provide additional analysis (for example, try to extend or generalize the method in your own way). In the rest of the document, when we say "to get full credit", we mean to get full credit in 6.815, except in the "harder" section (3.6). In many cases, additional components are described and you can just pick from there. If you are selecting from previous year's offering, you should implement the grad version. In general, if you're unsure, ask us.

3.1 Previous Years' Assignments

- Bayesian Matting: http://stellar.mit.edu/S/course/6/sp11/6.815/homework/assignment5/
- Seam Carving: http://stellar.mit.edu/S/course/6/sp11/6.815/homework/assignment6/ (paper link: http://www.faculty.idc.ac.il/arik/site/seam-carve.asp)
- Deconvolution and Poisson Editing: https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment10/
- Non-photorealistic Rendering: https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment11/
- Light Field (Lytro): https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment12/
- Video Magnification: https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment15/

3.2 Additional - Easy

3.2.1 Texture synthesis

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

http://en.wikipedia.org/wiki/Texture_synthesis
http://graphics.cs.cmu.edu/people/efros/research/EfrosLeung.html
http://www.ics.uci.edu/~fowlkes/class/cs116/hwk3/index.html
http://cs.nyu.edu/~fergus/teaching/comp_photo/assign3.pdf
For full credit, perform hole filling.
Also see:

http://lgg.epfl.ch/publications/2015/Texture/index.php
https://arxiv.org/abs/1505.07376

3.2.2 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. 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.

3.2.3 Hybrid images

Generate images that look different from a close vs. a large distance.

http://cvcl.mit.edu/publications/publications.html
https://courses.engr.illinois.edu/cs498dh/fa2011/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 least two examples. You might want to use your warping code to align the two images.

3.2.4 Style transfer using convolutional neural networks

https://arxiv.org/abs/1508.06576

3.3 Additional - Normal (with instructions)

3.3.1 Inpainting 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

3.3.2 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
http://graphics.cs.cmu.edu/courses/15-463/2010_fall/hw/proj4g/

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.

3.4 Additional - Normal

3.4.1 Convolutional Neural Network with Halide

Implement a convolutional neural network with backpropagation in Halide (http://cs231n.github.io/convolutional-networks/). Train the network on simple tasks like image denoising or deconvolution.

3.4.2 Demosaicking++

Implement advanced demosaicking algorithms.
http://ieeexplore.ieee.org/document/1395991

https://groups.csail.mit.edu/graphics/demosaicnet/

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

For this one, you may want to refer to https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment10/.

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 $Ax = b$.

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.

Extra-credit (easy to implement, hard to understand): Use reweighted least square to simulate an L1 regularization. That is, rather than minimizing the gradients with uniform weights, reqesight the constraint of each gradient by the magnitude of the gradient.

Careful: you need to reweight the gradient, not the Laplacian. You need to decompose the Laplacian as the divergence of the gradient. As discussed in class, to compute the divergence of the gradient, you should use kernels [-1, 1] but you need to use forward difference for one and backward differences for the other one, so that the overall kernel gets centered.

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

3.4.4 Dehazing

Remove haze in photography. Locally compute the minimum and use it to guestimate what to subtract from the image.

http://ieeexplore.ieee.org/abstract/document/5206515/

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

3.4.5 Super-resolution

Given a low resolution image, output a high resolution image (Enhance!).

http://www.wisdom.weizmann.ac.il/~vision/SingleImageSR.html
http://vllab.ucmerced.edu/wlai24/LapSRN/

3.4.6 Approximating image filters with bilateral grid

See https://people.csail.mit.edu/jiawen/bgu/bgu.pdf.

3.4.7 Rectangling Panoramas with Warping

Making the panoramas as rectangular as possible. See http://kaiminghe.com/sig13/index.html.

3.4.8 Stainglass by numbers

For this assignment, you may want to refer to:

https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment11/

This is 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, *i.e.*, 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
Forget about the relaxation part, unless you're very motivated.

3.4.9 Denoising by wavelet coring

Denoising using Bayesian estimator.

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

3.4.10 NL means denoising

Non-Local means denoising. Easy but slow (Halide might be useful here).

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

3.4.11 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

3.4.12 Color 2 gray

Turn a color image into a black and white one while preserving edges as well as possible. Start from the Poisson code and the max of the gradient across the three channels.

Then be smarter: https://dl.acm.org/citation.cfm?doid=1073204.1073241

3.4.13 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/

A good starting point is with your morphing code. You can create the same segments for a lot of different faces. We can give you what we have from the prior problem set for all the students, but it might work better with one of the standard datasets linked above, because subjects are all in exactly the same pose and their hair usually doesn't get as much in the way.

Things we would like to see: (1) average face, (2) average male, average female, (3) caricature of a face, and (4) make a face more male or more female.

3.4.14 Pyramid image alignment

Implement a coarse-to-fine version of image alignment (more sophisticated than the slow brute-force alignment you implemented in earlier problem set)

Extend it to the median pyramid by Greg Ward http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.277

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

3.4.15 Time lapse manipulation

http://dl.acm.org/citation.cfm?id=1276505

Median or min across time, dynamic programming. For full credit, implement at least the dynamic programming version with two different metrics.

3.4.16 Patchmatch

Otherwise referred to as content-aware fill in Photoshop.

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

3.4.17 Detecting copy-pasting

http://www.cs.dartmouth.edu/farid/downloads/publications/tr04.pdf

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

3.4.18 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. 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 (unless your in 6.865).

3.4.19 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).

3.4.20 Anisotropic diffusion

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

Alternative to the bilateral filter, but based on PDEs.

3.4.21 Laplacian pyramids

The generalization of the 2-scale blending that we did in panorama, 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

3.4.22 Photobios

Implement photobios

http://grail.cs.washington.edu/photobios/paper.pdf

http://grail.cs.washington.edu/photobios/video.mp4

You can try getting data from this website: http://daily.jasonfletcher.info/. Or find your own collection of lots of faces.

3.4.23 Guided image filtering

http://kaiminghe.com/eccv10/

3.5 Additional - Normal (but requires hardware)

3.5.1 Separation of direct and indirect lighting effects

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

You can borrow a projector.

3.5.2 Dual photography

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

You can borrow a projector.

3.5.3 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.

3.6 Additional - Harder

3.6.1 HDR+

Implement the HDR+ camera pipeline.

http://www.hdrplusdata.org/

3.6.2 Single image HDR

http://hdrv.org/hdrcnn/ http://www.npal.cs.tsukuba.ac.jp/~endo/projects/DrTMO/

3.6.3 More photographic style transfer

https://www.cs.cornell.edu/~fujun/files/style-cvpr17/style-cvpr17.html
http://people.csail.mit.edu/yichangshih/portrait_web/

3.6.4 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.

3.6.5 Lens correction

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

Vignetting is more tricky than it seems.

3.6.6 Inpainting

This one is not hard to implement but it is not easy to understand. You may want to refer to https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment10/.

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

A good solution is to 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.

3.6.7 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/

3.6.8 More inpainting

http://ieeexplore.ieee.org/document/1323101/

3.6.9 Bundle adjustment

Refine your panorama with a global optimization, including radial distortion optimization. This one might be somewhat hard because of the optimization. You can try using http://ab-initio.mit.edu/wiki/index.php/NLopt or http://ceres-solver.org/.

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 below.

Good Resources:

https://pdfs.semanticscholar.org/2b0c/9c57572b156680e10f711b13ae205849493d.pdf

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

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

http://ceres-solver.org/nnls_tutorial.html#bundle-adjustment

3.6.10 Image colorization

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.

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

You may want to refer to https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment10/. You can do a full linear algebra version by forming the sparse matrix and use the Poisson code with other kernels, *e.g.*, replace the Laplacian kernel by an input-dependent kernel or a bilateral filter.

Also see http://richzhang.github.io/colorization/

3.6.11 Perceptual metric for photo retouching

http://www.pnas.org/content/108/50/19907.full.pdf

Use your morphing code for computing the warp field.

The tricky part is getting data.

3.6.12 Hockney collage from a single image

Create a collage in the style of David Hockney's polaroids: http://www.davidhockney.co/works/photos/composite-polaroids

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.

3.6.13 Hockney collage from multiple images

Create a collage in the style of David Hockney's collages: http://www.davidhockney.co/works/photos/photographic-collages

See an example of software at http://lihi.eew.technion.ac.il/files/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.

3.6.14 Local Laplacian

What replaced the bilateral filter in Camera RAW/Lightroom.

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

3.6.15 Adaptive manifolds filter

Yet another fast edge-aware filter. The math could be scary but the implementation is not that complicated.

http://inf.ufrgs.br/~eslgastal/AdaptiveManifolds/

3.6.16 Image deformation using moving least squares

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

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 and needs a solver for the least square problem.

An extension uses biharmonic energies http://igl.ethz.ch/projects/bbw/

3.6.17 Non-Photorealistic rendering using extended differences of Gaussians

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

Implement both adaptive thresholding and flow alignment to get full credit.

For this one, you may want to refer to https://stellar.mit.edu/S/course/6/fa13/6.815/homework/assignment11/. Use the structure tensor and the eigenvector business from that pset to get the flow alignment, or use the Sobel operator if you prefer.

Rather than being circular, Gaussians now have elliptical shapes, and the long axis of the Gaussian follows the directions from the tensor field. To get an anisotropic Gaussian, rather than using $\exp(-(x^2 + y^2)/2\sigma^2)$ you replace the simple square by a general quadratic: $\exp(-V^T SV)$ where V is your vector (x, y) and S is a symmetric positive definite matrix. For example, the diagonal matrix [1, 0; 0, 2] gives you a Gaussian elongated twice as long in the second coordinate.

One way to achieve general anisotropic Gaussians is to generate a canonical one like the one just above and them use the image rotation function we used in the referenced assignment to get other orientations. Alternatively: http://www.science.uva.nl/research/publications/2003/GeusebroekTIP2003/

3.6.18 Multiflash camera

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

3.6.19 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

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.

3.6.20 Interactive Digital Photomontage

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

3.6.21 Reducing veiling glare for higher-dynamic-range imaging

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

3.6.22 Artistic screening

Reproduce shades of grey with micro patterns of your choice.

http://www.iro.umontreal.ca/~ostrom/publications/pdf/SIGGRAPH95_ArtisticScreening.pdf

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

3.6.23 Laplacian matting

Separate foreground and background. http://www.wisdom.weizmann.ac.il/~levina/papers/Matting-Levin-Lischinski-Weiss-CVPR06.pdf

The derivation is a little scary but the implementation can be simplish.

3.6.24 BM3D denoising

http://www.cs.tut.fi/~foi/GCF-BM3D/