
Lab 3 – Digital Image Processing

Introduction

Digital image analysis refers to the manipulation of digital images with the aid of a computer. The field of digital image analysis is broad, but fundamentally focuses on the use of algorithms to modify pixel values from an image in order to extract information or enhance later processing. This lab focuses on procedures for image enhancement including techniques for contrast manipulation, smoothing, and edge extraction.

Pre-lab requirement: i.e. you must complete this before lab starts

Read the following pages in Lillesand et al. (2014):

- P500 Image enhancement
- P501 Gray-level thresholding and level slicing
- P507–511 Spatial filtering and convolution
- P511–512 Edge enhancement
- P530–536 Intensity-hue-saturation color space transformation

Read the following pages in Szeliski (2011):

- P111–114 Image convolution and linear filtering
- P238–249 Edge detection and edge linking
- P251–254 Hough transforms

Objectives


This lab focuses on guiding you through several image processing techniques. At the end of this lab you will be able to use GEE to perform the following tasks:

- Grey-level thresholding
- Level slicing
- Image convolution using boxcar kernel
- Edge detection
- Pan sharpening
- Texture analysis

Selecting an image

Google Earth Engine provides a search tool to identify image collections of interest. However, having identified an image collection, there is typically a desire to narrow down spatially, temporally, or based on some image quality such as cloud content.

```
//Create a variable to look at the Landsat 8 Top-of-atmosphere image collection
var l8 = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA');
```

Use the point tool in the map window to create a point in Syracuse . By default the feature will be stored in the variable called geometry.

```
//Filter the image collection spatially to select images that include the point created
var spatialFiltered = l8.filterBounds(geometry);
```

```
//Narrow the data range of interest.
var temporalFiltered = spatialFiltered.filterDate('2017-01-01', '2018-12-31');
```

```
// This may still lead to a large number of scenes so sort from least to most cloudy.
var sorted = temporalFiltered.sort('CLOUD_COVER');
```

```
// And then select the first (i.e. least cloudy) image.
var bestscene = ee.Image(sorted.first());
```

```
/*If you print the selected scene to the console you can see the image details
including the id, which can be used to directly select that image in the future */
print('Best scene', bestscene);
```

```
//Center map display on the point created with a zoom level of 12
Map.centerObject(geometry,12);
```

```
//Display image as color infrared composite.
Map.addLayer(bestscene, {bands: ['B5', 'B4', 'B3'], max: 0.5}, 'input image');
```

Performing image enhancement

The main objective of image enhancement is to improve the interpretability of an image by increasing the distinction between features in the scene. Image enhancement is normally applied after image pre-processing. Image enhancement techniques fall into three main categories: contrast manipulation, spatial feature manipulation and multi-image manipulation. This lab will focus on several approaches in these broad topics.

Grey-level Thresholding


Grey-level thresholding can be used to segment an input image. In the example below a threshold is used to separate remove water pixels from an image based on a threshold defined in one image band.

```
//Import a top-of-atmosphere reflectance Landsat 5 image.
var image = ee.Image('LANDSAT/LT05/C01/T1_TOA/LT05_015030_20100531');

//Center map display on Syracuse with a zoom level of 12
Map.setCenter(-76.1467, 43.0458,12);

//Display image as color infrared composite.
Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], max: 0.5}, 'input image');

//Define a study area that captures Onondaga Lake and the surrounding area
var studyarea = ee.Geometry.Rectangle(-76.454, 43.692,-76.40, 42.82);
```

Alternately, you can create a polygon geometry on the fly using the geometry creation tool in the map window. 

```
//Select out the NIR band from the input image
var NIRband = image.select('B4');

//Create a histogram of the spectral response in the NIR band using the geometry identified
var histogram2 = ui.Chart.image.histogram(NIRband, studyarea);

// Display the histogram.
print(histogram2);
```

Note that the histogram has two peaks with the low one corresponding to water. Since the reflectance value of 0.07 divides the two peaks we can use that to threshold out the water pixels.

```
//Create a binary mask to filter out water based on the reflectance information
var watermask = NIRband.gte(0.07);

//Use the water mask to create an image with water pixels removed.
var nowaterimage= image.updateMask(watermask);

//Display the output water-masked image using a CIR composite
Map.addLayer(nowaterimage, {bands: ['B4', 'B3', 'B2'], max: 0.5}, 'water-masked');
```

Note: unless you commented out the first Map.addLayer line above, the original image will still be visible underneath the masked version. Use the Layers controls in the map window to verify that the thresholding has effectively eliminated water.

Level Slicing

Level slicing is an enhancement technique where the DN's or reflectance values distributed along the x-axis of an image histogram are divided into a series of analyst-specified intervals (i.e. slices). This technique is accomplished by assigning different color palettes to each data interval.

```
//Load and display a top-of-atmosphere reflectance Landsat 5 image
var image = ee.Image('LANDSAT/LT05/C01/T1_TOA/LT05_015030_20100531');

//Center map display on Syracuse with a zoom level of 12
Map.setCenter(-76.1467, 43.0458,12);

//Display image as color infrared composite.
Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], max: 0.5}, 'input image');

//Create an NDWI image using band 4 (NIR) and 5 (mid-IR) from the selected Landsat 5 image,
var ndwi = image.normalizedDifference(['B4', 'B5']); //create a NDWI image

/*Create an image visualization parameter to recolor the NDWI image based on pixels values:
  NDWI <= 0, display as white (FFFFFF color)
  NDWI > 0 & < 0.5, display in cyan (00FFFF color)
  NDWI >=0.5 & <= 1, display as blue (0000FF color)
Higher NDWI suggest higher water content, the three slices are represented with different colors
that show no water content, low water content and high water content pixels.*/
var ndwislices = ndwi.expression(
  "(b('nd') < 0) ? 3" + //nd refer to the values of each pixel in the NDWI image
  ": (b('nd') < 0.5) ? 2" +
  ": (b('nd') < 1) ? 1" +
  ": 0"
);

//Display final image based on level sliced image.
Map.addLayer(ndwislices,
  {min: 0, max: 3, palette: ["0000FF", "00FFFF", "FFFFFF"]},
  'NDWI Slices');
```

Image Convolution

Image convolution, also called neighborhood filtering, uses pixel values in the vicinity of a given pixel to determine its final output value. In addition to performing local tone adjustment, convolution can be used to filter images in order to smooth, sharpen details, accentuate edges, or remove noise (Szeliski 2011). This exercise uses a low-pass filter to smooth an image. Low-pass

filters are designed to emphasize low spectral frequency features (large areas of change in brightness) and deemphasize the high frequency components (local details) of an image.

```
//Load and display a top-of-atmosphere reflectance Landsat 5 image
var image = ee.Image('LANDSAT/LT05/C01/T1_TOA/LT05_015030_20100531');

//Center map display on Syracuse with a zoom level of 12
Map.setCenter(-76.1467, 43.0458, 12);

//Display image as color infrared composite.
Map.addLayer(image, {bands: ['B4', 'B3', 'B2'], max: 0.5}, 'input image');

/*Define a low-pass kernel. This kernel creates a 7x7 square, the normalize parameter means the
sum of values in the kernel will be 1 */
var lowpass = ee.Kernel.square({radius: 7, units: 'pixels', normalize: true});

//Smooth the image by convolving with the low-pass kernel.
var smooth = image.convolve(lowpass);
Map.addLayer(smooth, {bands: ['B4', 'B3', 'B2'], max: 0.5}, 'smoothed');
```

Note: modifying the radius will change the amount of smoothing e.g. a radius of 3 will result in less smoothing or a radius of 11 would result in more smoothing.

Edge Detection/Enhancement

Edge detection or enhancement attempts to preserve both local contrast and low frequency brightness information. The Canny edge detection algorithm (Canny 1986) uses four separate filters to identify the diagonal, vertical, and horizontal edges. The calculation extracts the first derivative value for the horizontal and vertical directions and computes the gradient magnitude. Gradients of smaller magnitude are suppressed. For line extraction from an edge detector, Earth Engine implements the Hough transform (Duda and Hart 1972). This well-known technique has edges “vote” for plausible line locations.

```
/*Load a Landsat 8 image and select only the panchromatic band. The panchromatic band
includes reflectance in the 0.503–0.676 µm wavelength range with a ground sample distance of
15 m. */
var image = ee.Image('LANDSAT/LC08/C01/T1/LC08_015030_20140307').select("B8");

//Center map display on Syracuse with a zoom level of 12
Map.setCenter(-76.1467, 43.0458, 12);

//Display image as color infrared composite.
Map.addLayer(image, "", 'input image');
```

```

/* Perform Canny edge detection and display the result. This function uses the threshold parameter
to determine the minimum gradient magnitude; the sigma parameter is the standard deviation (SD)
of a Gaussian pre-filter to remove high-frequency noise. A sigma of 0 means apply no filtering.*/
var canny = ee.Algorithms.CannyEdgeDetector({
  image: image, threshold: 200, sigma: 1 });

//Display the outcome of the canny edge detection.
Map.addLayer(canny, {}, 'canny');

// Perform a Hough transform of the Canny result. In this function gridSize means size of grid for
analysis (default is 256), inputThreshold is the value threshold for the input image (pixels at or
above this value are considered active) and lineThreshold is the threshold for line detection
(values at or above this threshold on the Hough transform are considered detected lines). Each of
these parameters can be determined experimentally. The defaults provided in GEE for the
parameters are intended as a starting point for use with unsigned 8-bit integers.
var hough = ee.Algorithms.HoughTransform(canny, 256, 7000, 80);

//Display the outcome of the Hough transformed canny edge detection.
Map.addLayer(hough, {}, 'hough');

```

Pan Sharpening using HSV space

Pan-sharpening techniques are used to merge panchromatic (high spatial, low spectral resolution) and multispectral (lower spatial, higher spectral resolution) imagery. One method for pan-sharpening is through transformations between RGB and Intensity-Hue-Saturation (IHS) or Hue-Saturation-Value (HSV) spaces. For more information on HSV space, please refer back to Lab 2.

```

// Import a Landsat 8 top-of-atmosphere reflectance image
var image = ee.Image('LANDSAT/LC08/C01/T1_TOA/LC08_015030_20140307');

//Display image as normal color composite.
Map.addLayer(image,{bands: ['B4', 'B3', 'B2']}, 'RGB image');

// Convert the visible bands from the input image to HSV color space.
var hsv = image.select(['B4', 'B3', 'B2']).rgbToHsv();

/* Create a new image by concatenating the hue and saturation bands (HSV image) with the panchr
omatic band from the original image and converting back to RGB. */

//Concatenate bands
var combined = ee.Image.cat([hsv.select('hue'), hsv.select('saturation'), image.select('B8')]);

//Convert back to RGB space
var sharpened = ee.Image(combined).hsvToRgb();

//Set the map display to center at City of Syracuse and a zoom level of 14

```

```
Map.setCenter(-76.14, 43.04, 14);

// Display the pan-sharpened result.
Map.addLayer(sharpened, {'pan-sharpened'});
```

Performing visual interpretation through texture analysis

Texture refers to the frequency of tonal change on an image. It is produced by the aggregation of unit features that may be too small to be discerned individually within the image. An example would be a smooth texture of green grass vs. the rough texture of green tree crowns on an aerial photo. Google Earth Engine has multiple ways to compute texture. One of these is entropy, which provides a measure of the disorder within a space.

```
//Import a high-resolution NAIP image.
var image = ee.Image('USDA/NAIP/DOQQ/m_4307663_se_18_1_20130619');

//Zoom map to Oakwood Cemetery at level 16 and display input image.
Map.setCenter(-76.13, 43.03, 16);
//The catch with entropy is that it needs discrete values, hence the image is rescaled.
Map.addLayer(image, {max: 255}, 'Input image');

//Select NIR band from input image.
var nir = image.select('N');

//Define size and shape of neighborhood for deriving the texture information.
var square = ee.Kernel.square({radius: 3});

//Compute entropy (texture) using the defined neighborhood information.
var entropy = nir.entropy(square);

//Display computed image texture outcome.
Map.addLayer(entropy, {min: 1, max: 5, palette: ['0000CC', 'CC0000']}, 'entropy');
```

Note: you can toggle between the layers in the map to see the texture differences between the grass and trees within the cemetery.

Assignment – Answer the Following Questions

Submit your code and text answers for this assignment by clicking on “Get Link” in the Code Editor and sending the link generated to the TA (your link will look something like <https://code.earthengine.google.com/XXXXXXXXX>). Any written responses should be in comments with each question separated by a line of forward slashes. For example:

```
//Q1. Text answer to Q1
```

```
Code answer to Q1 //All code responses need suitable comments.
```

```
////////////////////////////////////
```

```
//Q2. Text answer to Q2
```

```
Code to Q2
```

```
...
```

1. Select one scene from the Landsat 8 image collection as an input image. Detect and remove all water pixels from the input image using the gray-level thresholding and level slicing techniques. Describe the differences between the resulting water masked images.
2. Select one image scene from the NAIP image collection as an input image. Use the Canny filter to highlights edges of Onondaga Lake, Oneida Lake and the runways at Syracuse International Airport. Select appropriate parameters to zoom in on each of the locations.
3. Will low-pass or high-pass filters help to identify large homogeneous features with low spectral frequency variation, e.g. the Carrier Dome? Select one image from the NAIP image collection as an input and explore the impact of these filters. How does your selected filter enhance the ability to find these features?
4. Select one scene from the Landsat 8 image collection and apply pan sharpening. How does pan sharpening an image help with image interpretation e.g. identifying heterogeneous features?
5. Assuming a fixed kernel (size and shape), describe the image texture within a lake, a football field, and a forest. Select one scene from the NAIP image collection and then apply texture analysis to provide examples to answer this question.

Appendix A - Glossary of Terms

Page numbers below refer to Lillesand et al. (2015).

- Contrast manipulation - P 500 to 506. A common category of image enhancement techniques. Example techniques in contrast manipulation can be gray-level thresholding, level slicing or contrast stretching.
- Spatial feature manipulation - P 500, 507–517. A common category of image enhancement techniques. Example techniques in spatial feature manipulation include spatial filtering, edge enhancement or Fourier analysis.
- Image convolution - P 509–511. Image convolution is a generic image processing operation that includes spatial filtering. Convolution provides a way of multiplying two arrays of numbers of the same dimensionality to produce a third array of numbers. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values.
- Kernel - P 509. A moving window utilized for image convolution. Also known as convolution matrix or mask.
- High-pass and Low-pass filters - P 507–508. Low-pass filters are designed to emphasize low spectral frequency features (large areas of change in brightness) and deemphasize the high frequency components (local details) of an image. High-pass filters do the reverse.
- Edge detection or enhancement - P 511–512. Edge enhancement attempt to preserve both local contrast and low frequency brightness information.
- Pan-sharpening - P 536. Pan-sharpening merges panchromatic and multispectral imagery to enhance the spatial resolution of the multispectral images. It can be done through the transformation between RGB and IHS spaces.
- Spectral mixture analysis - P 564–567. This involves a range of techniques wherein mixed spectral signatures are compared to a set of “pure” reference spectra. The result is an estimate of the approximate proportions of the ground area of each pixel that is occupied by each of the reference classes.
- Texture - P 60. Texture refers to the frequency of tonal change on an image. It is produced by the aggregation of unit features that may be too small to be discerned individually on the image, such as tree leaves and leaf shadows. An example would be a smooth texture of green grass vs. rough texture of green tree crowns on an aerial photo.

Appendix B – Useful Resources

- Google Earth Engine Java Style Guide
<http://google.github.io/styleguide/javascriptguide.xml>
- Google Earth Engine Guides/Cookbook/Dictionary
<https://developers.google.com/earth-engine/>
- Google Earth Engine API Tutorials
<https://developers.google.com/earth-engine/tutorials>
- Google Earth Engine Workshop (Beginning)
<https://docs.google.com/document/d/1ZxRKMic8dfTvBmUNOO0TFMkd7ELGWf3WjX0JvESZdOE>
- Google Earth Engine Workshop (Intermediate)
<https://docs.google.com/document/d/1keJGLN-j5H5B-kQXdwy0ryx6E8j2D9KZVEUD-v9evys>
- Google Earth Engine tutorials on AmericaView
<https://americaview.org/program-areas/education/google-earth-engine-tutorials/>

General references

- Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 679–698.
<https://doi.org/10.1109/TPAMI.1986.4767851>
- Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), 11–15.
<https://doi.org/10.1145/361237.361242>
- Lillesand, T. M., Kiefer, R. W., & Chipman, J. W. (2015). *Remote Sensing and Image Interpretation*. 7th edition. John Wiley & Sons.
- Szeliski, R. (2011). *Computer Vision*. *Computer*, 5, 832.
<https://doi.org/10.1007/978-1-84882-935-0>



This work is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).

This work was produced by Ge (Jeff) Pu and Dr. Lindi Quackenbush at State University of New York-College of Environmental Science and Forestry. Suggestions, questions and comments are all welcome and can be directed to Jeff at gpu100@syr.edu.