Bài 5: Ảnh số và xử lý ảnh

**Mục đích**: Mục đích của bài thực hành này là giới thiệu những khái niệm cơ bản về xử lý ảnh. Bạn sẽ được tìm hiểu các hàm giúp nâng cao chất lượng ảnh như smoothing/làm mịn, sharpening/làm sắc nét, edge detection/dò biên ảnh, morphological processing/xử lý hình thái học, texture analysis/phân tích kết cấu ảnh, resampling/thay đổi độ phân giải và reprojection/nắn ảnh. Sau khi hoàn thành bài thực hành, bạn sẽ có thể xác định các công cụ xử lý ảnh phù hợp để trích xuất thông tin bạn cần trong quá trình phân tích ảnh.

## Ảnh Số Là Gì?

Ảnh số, còn được gọi là [raster](https://en.wikipedia.org/wiki/Raster_graphics), là một tập hợp các pixel ([Besag 1986](http://www.jstor.org/stable/2345426)). Mỗi pixel có một vị trí, được tính trong theo trục của một hệ tọa độ nhất định (coordinate reference system - CRS), cụ thể là một hệ tọa độ địa lý. Một hệ tọa độ trong Earth Engine gắn liền với một phép chiếu, phép chiếu là cách biểu thị mặt cong của trái đất lên một mặt phẳng bản đồ dựa trên một gốc tọa độ.

Ví dụ:

1. Tạo một điểm/point sử dụng [geometry drawing tools](https://developers.google.com/earth-engine/playground#geometry-tools) (công cụ vẽ đối tượng hình học) để xác định vùng nghiên cứu của bạn. Đặt tên cho điểm/point đó.
2. Đưa ảnh Sentinel vào bằng công cụ tìm kiếm với từ khóa 'Sentinel' và chọn tập dữ liệu raster 'Sentinel 2'. Đặt tên cho ảnh Sentinel vừa được đưa vào.
3. Chon một ảnh Sentinel gần đây, bao phủ toàn bộ khu vực nghiên cứu của bạn, xem và kiểm tra ảnh.

// Get a single NAIP image over the area of interest.

var image = ee.Image(Sentinel

.filterBounds(point)

.sort('CLOUDY\_PIXEL\_PERCENTAGE', true)

.first());

//subset by the bands. Here we chose NIR, red and green

var bands = ['B8','B4','B3'];

image = image.select(bands);

// Print the image to the console.

print('Inspect the image object:', image);

// Display the image with the default visualization.

Map.centerObject(point, 14);

Map.addLayer(image, {min:1300, max:3000}, 'Original image');

1. Mở rộng phần thông tin về đối tượng ảnh vừa được in trong console. Mở rộng phần thuộc tính của các band/kênh phổ và một trong các band/kênh phổ (ví dụ band số 0). Chú ý rằng phép chuyển hệ trục tọa độ được lưu trong thuộc tính của crs\_transform và hệ trục tọa độ được lưu trong thuộc tính crs, tham khảo EPSG code. Bạn có thể học thêm về EPSG code tại [the spatialreference.org site](http://spatialreference.org/). Phép chuyển hệ trục tọa độ được liệt kê như sau: [m00, m01, m02, m10, m11, m12] trong phần chú giải của tài liệu tham khảo này.
2. Truy nhập những dữ liệu này theo thứ tự sử dụng hàm .projection():

// Display the projection of band 0.

print('Inspect the projection of band 0:', image.select(0).projection());

1. Chú ý rằng phép chiếu có thể khác nhau với từng band, do đó việc kiểm tra phép chiếu cho từng band ảnh là rất cần thiết. (Nếu bạn gọi .projection() cho từng ảnh hoặc cho những đối tượng mà phép chiếu các band khác nhau, bạn sẽ nhận được báo lỗi.) Tìm hiểu ghi chú của ee.Projection để biết thêm về các hàm thường được dùng cho đối tượng Projection. (Để tìm hiểu thêm về các phép chiếu, thử công cụ này.)

## Hiển Thị Ảnh Số

Chúng ta đã học về cách mà mỗi ảnh lưu trữ dữ liệu pixel trong mỗi band/kênh ảnh với các giá trị số (DNs) và các các pixel được sắp xếp theo không gian. Khi bạn đưa ảnh vào cửa sổ hiển thị, Earth Engine thực hiện hiển thị ảnh qua việc nhận diện phép chiếu và đặt tất cả các pixel vào đúng vị trí của nó theo không gian. Truy nhiên, bạn cần định rõ biên độ của DNs để hiển thị một ảnh 16-bit (ví dụ các thông số hiển thị min và max). Định rõ min và max áp dụng công thức sau (trong đó DN' là giá trị hiển thị):

DN' = (DN - min) \* 65536 / (max - min)

1. Để áp dụng [hiệu chỉnh gamma (gamma correction](https://en.wikipedia.org/wiki/Gamma_correction)) (DN' = DN𝛾), sử dụng:

// Display gamma stretches of the input image.

Map.addLayer(image.visualize({gamma: 0.5}), { min:1300, max:3000}, 'gamma = 0.5');

Map.addLayer(image.visualize({gamma: 1.5}), { min:1300, max:3000}, 'gamma = 1.5');

* 1. Chú ý rằng gamma được đưa ra như một đối số/argument cho image.visualize() vì vậy bạn có thể click vào ảnh để xem sự khác nhau trong các giá trị pixel (hãy thử!). Bạn cũng có thể định rõ gamma, min và max để đạt được các hiển thị đặc trưng khác.

1. (*Không bắt buộc*) Để áp dụng phương pháp [histogram equalization](https://en.wikipedia.org/wiki/Histogram_equalization) (cân bằng histogram), sử dụng hàm [sldStyle()](https://devsite.googleplex.com/earth-engine/image_visualization#styled-layer-descriptors)

// Define a RasterSymbolizer element with '\_enhance\_' for a placeholder.

var histogram\_sld =

'<RasterSymbolizer>' +

'<ContrastEnhancement><Histogram/></ContrastEnhancement>' +

'<ChannelSelection>' +

'<RedChannel>' +

'<SourceChannelName>B8</SourceChannelName>' +

'</RedChannel>' +

'<GreenChannel>' +

'<SourceChannelName>B4</SourceChannelName>' +

'</GreenChannel>' +

'<BlueChannel>' +

'<SourceChannelName>B3</SourceChannelName>' +

'</BlueChannel>' +

'</ChannelSelection>' +

'</RasterSymbolizer>';

// Display the image with a histogram equalization stretch.

Map.addLayer(image.sldStyle(histogram\_sld), {}, 'Equalized');

* 1. Hàm [sldStyle() method](https://devsite.googleplex.com/earth-engine/image_visualization#styled-layer-descriptors) yêu cầu các thống kê của ảnh phải được thực hiện trong phạm vi một vùng (để xác định biểu đồ histogram).

## Lọc Tuyến Tính

Trong ngữ cảnh hiện tại, lọc tuyến tính/linear *filtering* (hay [convolution](http://www.dspguide.com/ch24/1.htm) (tích chập) là sự kết hợp tuyến tính của các giá trị pixel trong một vùng lân cận. Vùng lân cận được xác định bằng một [kernel](https://en.wikipedia.org/wiki/Kernel_(image_processing)) (cửa sổ lọc/ma trận lọc), trong đó các trọng số/weight của cửa sổ lọc/ma trận lọc quyết định các hệ số/coefficient của liên kết tuyến tính. (Trong bài thực hành này, các thuật ngữ *kernel/cửa sổ lọc* và *filter/lọc* là một) Lọc một ảnh có thể hữu ích cho việc trích xuất thông tin ảnh tại [tần số không gian/spatial frequencies](http://www.dspguide.com/ch24/5.htm) khác nhau. Do đó, bộ lọc làm mịn ảnh/smoothing filter được gọi là lọc tần số thấp/*low-pass* filters (dữ liệu đi qua cửa sổ lọc với tuần số thấp) và lọc edge detection filter/dò biên ảnh được gọi là lọc số suất cao/*high-pass* filter. Để dùng các bộ lọc trong Earth Engine sử dụng [image.convolve()](https://devsite.googleplex.com/earth-engine/image_convolutions) với đối số/argument ee.Kernel.

1. **Smoothing/Làm mịn**. Smoothing/làm mịn có nghĩa là nhân chập ảnh với cửa sổ lọc smoothing.
   1. Một phép lọc smoothing đơn giản là một cửa sổ lọc hình vuông có cùng trọng số mà tổng của bằng 1. Nhân chập với cửa sổ lọc này, mỗi pixel sẽ có một giá trị mới bằng giá trị trung bình của các pixel lân cận nằm trong cửa sổ lọc. In một cửa sổ lọc hình vuông với cùng một trọng số (đôi khi còn được gọi là "pillbox" hay "boxcar" filter):

// Print a uniform kernel to see its weights.

print('A uniform kernel:', ee.Kernel.square(2));

Mở rộng phần thông tin về đối tượng cửa sổ lọc/kernel tại console để xem các trọng số. Cửa sổ lọc này được định nghĩa bằng số lượng pixel nằm trong cửa sổ lọc (hay nói cách khác kích thước cửa sổ lọc có đơn vị là 'pixel'). Một cửa sổ lọc với kích thước theo mét nghĩa là kích thước của cửa sổ lọc đó đã được điều chỉnh từ pixel, do đó bạn không thể thấy được trọng số của nó, nhưng nó lại linh động hơn khi phải làm việc với các ảnh đầu vào ở tỷ lệ khác nhau. Trong phần sau, sử dụng các cửa sổ lọc với kích thước được định nghĩa theo mét (bỏ qua bước kiểm tra trọng số).

* 1. Định nghĩa một cửa sổ lọc có kích thước 20 mét (Tương ứng với bao nhiêu pixel của ảnh Sentinel? Convolve/Nhân chập ảnh với cửa sổ lọc đó và so sánh ảnh đầu vào và ảnh sau khi đã được làm mịn/smooth)

// Define a square, uniform kernel.

var uniformKernel = ee.Kernel.square({

radius: 20,

units: 'meters',

});

// Filter the image by convolving with the smoothing filter.

var smoothed = image.convolve(uniformKernel);

Map.addLayer(smoothed, { min: 1123, max: 2069}, 'smoothed image');

* 1. Để ảnh được mịn hơn, thử thay đổi kích thước của vùng lân cận bằng cách tăng kích thước pixel.
  2. Cửa sổ lọc Gaussian (lọc nhiễu) cũng có thể được sử dụng để làm mịn ảnh. Lọc với cửa sổ lọc Gaussian bằng cách tính trung bình lượng giá của các pixel lân cận. Ví dụ như:

// Print a Gaussian kernel to see its weights.

print('A Gaussian kernel:', ee.Kernel.gaussian(2));

// Define a square Gaussian kernel:

var gaussianKernel = ee.Kernel.gaussian({

radius: 20,

units: 'meters',

});

// Filter the image by convolving with the Gaussian filter.

var gaussian = image.convolve(gaussianKernel);

Map.addLayer(gaussian, {min: 1123, max: 2069}, 'Gaussian smoothed image');

1. **Edge detection/Dò biên ảnh**. Nhân chập với cửa sổ lọc dò biên ảnh/edge-detection kernel để tìm những biến đổi trong DNs, thường được phát hiện ở mép của các đối tượng trong ảnh dữ liệu.
   1. Cửa sổ lọc dò biên ảnh cổ điển là ma trận [Laplacian](https://en.wikipedia.org/wiki/Discrete_Laplace_operator). Xem kỹ trọng số trong của sổ lọc và ảnh kết quả sau khi nhân chập với cửa sổ Laplacian:

// Define a Laplacian filter.

var laplacianKernel = ee.Kernel.laplacian8();

// Print the kernel to see its weights.

print(laplacianKernel);

// Filter the image by convolving with the Laplacian filter.

var edges = image.convolve(laplacianKernel)

.reproject('EPSG:32647', null, 10);

Map.addLayer(edges, {min: 1123, max: 2069}, 'Laplacian filtered image');

* 1. (bỏ qua reproject(). Nó sẽ được giải thích trong phần 6.)
  2. Các cửa sổ lọc dò biên ảnh khác gồm có [Sobel](https://en.wikipedia.org/wiki/Sobel_operator), [Prewitt](https://en.wikipedia.org/wiki/Prewitt_operator) và [Roberts](https://en.wikipedia.org/wiki/Roberts_cross). [Tìm hiểu thêm về các phương pháp dò biên ảnh khác trong Earth Engine](https://devsite.googleplex.com/earth-engine/image_edges).

1. (*Không bắt buộc*) Lọc theo đạo hàm **Gradients**. Gradient ảnh là sự thay đổi giá trị pixel theo không gian (tương tự như tạo slope/dốc cho DEM).
   1. Trong Earth Engine, dùng [image.gradient()](https://devsite.googleplex.com/earth-engine/image_gradients) để tính gradient/độ dốc của các band/kênh phổ trong ảnh. Ví dụ, gradient/độ dóc của kênh phổ hồng ngoại NIR chỉ thị sự biến đổi về thực vật:

// Compute the image gradient in the X and Y directions.

var xyGrad = image.select('B8').gradient();

// Compute the magnitude of the gradient.

var gradient = xyGrad.select('x').pow(2)

.add(xyGrad.select('y').pow(2)).sqrt()

.reproject('EPSG: 32647', null, 10);

// Compute the direction of the gradient.

var direction = xyGrad.select('y').atan2(xyGrad.select('x'))

.reproject('EPSG:32647', null, 10);

// Display the results.

Map.addLayer(direction, {min: -3, max: 3, format: 'png'}, 'direction');

Map.addLayer(gradient, {min: -10, max: 50, format: 'png'}, 'gradient');

* 1. (Bỏ qua reproject() sẽ được giải thích trong 6.)
  2. Để hiểu sâu hơn về gradients/độ dốc trong ảnh đa phổ, đọc [Di Zenzo (1986)](http://www.sciencedirect.com/science/article/pii/0734189X86902239).

1. (*Không bắt buộc*) **Tăng cường độ sắc nét/Sharpening**. Tăng cường độ sắc nét của ảnh hay [edge enhancement](http://www.dspguide.com/ch24/2.htm) (lọc tăng cường biên), có liên quan đến đạo hàm bậc hai của ảnh. Cụ thể, mô phỏng lại quan điểm về [Hiệu ứng ảo ảnh Mach bands](https://en.wikipedia.org/wiki/Mach_bands) trong phản xạ quang học của mắt người bằng cách cộng ảnh với đạo hàm bậc hai của nó.
   1. Môt ứng dụng của ý tưởng này là nhân chập ảnh với một toán tử Laplace của một hàm Gauss (Laplacian-of-a-Gaussian) hoặc lọc [Difference-of-Gaussians](https://en.wikipedia.org/wiki/Difference_of_Gaussians) (Hàm sai khác DoG) (đọc [Schowengerdt 2007](http://www.sciencedirect.com/science/book/9780123694072) để biết thêm chi tiết), và đưa nó vào ảnh đầu vào:

// Define a "fat" Gaussian kernel.

var fat = ee.Kernel.gaussian({

radius: 30,

sigma: 3,

magnitude: -1,

units: 'meters'

});

// Define a "skinny" Gaussian kernel.

var skinny = ee.Kernel.gaussian({

radius: 30,

sigma: 0.5,

units: 'meters'

});

// Compute a difference-of-Gaussians (DOG) kernel.

var dog = fat.add(skinny);

// Add the DoG filtered image to the original image.

var sharpened = image.add(image.convolve(dog));

Map.addLayer(sharpened, {min: 1123, max: 2069}, 'Edges enhanced');

* 1. Các khái niệm liên quan gồm có [*spectral inversion*](http://www.dspguide.com/ch14/5.htm) *(nghịch đảo phổ)* từ xử lý ảnh số và *unsharp masking (mặt nạ không sắc nét)* ([Burger and Burge 2008](http://imagingbook.com/)).

## Xóa Script/Tập Lệnh

1. Tạo một đối tượng dạng Điểm/Point bằng [geometry drawing tools](https://developers.google.com/earth-engine/playground#geometry-tools) (công cụ vẽ vật thể hình học) để xác định vùng nghiên cứu của (chú ý, phải nằm trong Hoa Kỳ).  Đặt tên cho điểm/point đó.
2. Đưa ảnh NAIP vào bằng cách thanh công cụ tìm kiếm với từ khóa 'naip' và chọn tập dữ liệu raster 'NAIP: National Agriculture Imagery Program' raster dataset.  Đặt tên cho ảnh naip.
3. Lấy một ảnh NAIP gần đây bao phủ toàn bộ khu vực nghiên cứu của bạn và xem kỹ nó:

// Get a single NAIP image over the area of interest.

var image = ee.Image(naip

   .filterBounds(point)

   .sort('system:time\_start', false)

   .first());

// Print the image to the console.

print('Inspect the image object:', image);

// Display the image with the default visualization.

Map.centerObject(point, 18);

Map.addLayer(image, {}, 'Original image');

// Define a square, uniform kernel.

var uniformKernel = ee.Kernel.square({

radius: 20,

units: 'meters',

});

## Lọc Phi Tuyến Tính

Các ví dụ nhân chập/convolution vừa được giới thiệu được thực hiện như những tổ hợp tuyến tính của các giá trị pixel tại một vùng lân cận (lọc theo đạo hàm gradient cần thêm một vài bước, nhưng không đáng bận tâm).  Các hàm phi tuyến tính/Non-linear functions được áp dụng cho vùng lân cận cũng rất hữu ích.  Sử dụng những hàm này cho ảnh trong Earth Engine thông qua [reduceNeighborhood()](https://devsite.googleplex.com/earth-engine/reducers_reduce_neighborhood).

1. **Median/Lọc Trung vị**.  Lọc trung vị/median filter rất hiệu quả để giảm độ nhiễu của ảnh.  Cụ thể, giả dụ rằng các pixel bất kỳ trong ảnh của bạn bị vỡ màu do những giá trị cao hoặc thấp hơn bình thường gây ra nhiễu.  Lọc ảnh với lọc trung bình/mean filter (như trong phần 3.a.i) có thể gây ra các pixel bị ảnh hưởng bởi nhiễu.  Để tránh điều đó, làm mịn ảnh với lọc trung vị (sử dụng lại cửa sổ lọc đồng nhất 5x5 như ở trên):

var median = image.reduceNeighborhood({

 reducer: ee.Reducer.median(),

 kernel: uniformKernel

});

Map.addLayer(median, {min: 0, max: 255}, 'Median');

1. **Mode/Lọc mode**.   Đối với các bản đồ phân loại, các phương pháp lọc trung vị/median hoặc lọc trung bình/mean thường ít có ý nghĩa để tập hợp dữ liệu định danh (nominal data).  Trong những trường hợp này, sử dụng mode (giá trị xuất hiện nhiều nhất) của vùng lân cận để có được giá trị xuất hiện thường xuyên nhất.
   1. Với mục đích giới thiệu, thử tạo một bản đồ phân loại bằng cách đặt ngưỡng giới hạn cho kênh phổ cận hồng ngoại NIR.  Với hai lớp kết quả được gán nhãn là 1 và 0:

// Create and display a simple two-class image.

var veg = image.select('N').gt(200);

// Display the two-class (binary) result.

var binaryVis = {min: 0, max: 1, palette: ['black', 'green']};

Map.addLayer(veg, binaryVis, 'veg');

* 1. Tính giá trị mode trong cửa sổ lọc kích thước 5x5:

// Compute the mode in each 5x5 neighborhood and display the result.

var mode = veg.reduceNeighborhood({

 reducer: ee.Reducer.mode(),

 kernel: uniformKernel

});

Map.addLayer(mode, binaryVis, 'mode');

* 1. So sánh hình dạng của các patch (khoanh vi, mảng rời rạc) trong ảnh mode và ảnh nguyên gốc veg.  Chú ý hình dạng của các patch để thấy hiệu ứng làm mịn mà mode thực hiện.

1. **Morphological processing/Xử lý hình thái học**.  Ý tưởng về xử lý ảnh theo hình thái học gắn liền với khái niệm về các đối tượng trong ảnh.  Ví dụ, giả định rằng các patch của giá trị 1 trong ảnh veg từ phần thực hành trên trên đại diện cho các patch/mảng rời rạc của lớp thực vật.
   1. **Phép giãn ảnh/Dilation** (max).  Nếu sự phân bố patch thấp hơn thực tế phân bố của thực vật, hoặc có chứa các holes/khe trống, phép lọc max có để được dùng để giãn tăng cường các vùng thực vật

// Dilate by takaing the max in each 5x5 neighborhood.

var max = veg.reduceNeighborhood({

 reducer: ee.Reducer.max(),

 kernel: uniformKernel

});

Map.addLayer(max, binaryVis, 'max');

* 1. Thử tăng độ giãn bằng cách tăng kích thước của cửa sổ lọc (tăng radius) hoặc áp dụng reduceNeighborhood() nhiều lần.
  2. **Phép co ảnh/Erosion** (min).  Trái ngược với phép giãn ảnh là [co ản](https://en.wikipedia.org/wiki/Erosion_(morphology))h, để giảm kích thước của các patch/mảng rời rạc:

// Erode by takaing the min in each 5x5 neighborhood.

var min = veg.reduceNeighborhood({

 reducer: ee.Reducer.min(),

 kernel: uniformKernel

});

Map.addLayer(min, binaryVis, 'min');

* 1. Kiểm tra kết quả và so sánh với ảnh đầu vào. Chú ý rằng hình dạng của cửa sổ lọc có ảnh hưởng đển hình dáng của các patch đã được giảm kích (tương tự với phép giãn ảnh).  Tìm hiểu thêm về những ảnh hưởng này bằng cách thử thay đổi hình dạng của cửa sổ lọc.  Cũng như phép giãn ảnh, chú ý rằng bạn có thể tăng độ co hơn bằng cách tăng kích thước của cửa sổ lọc hoặc lặp quy trình lại nhiều lần.
  2. **Mở/Opening**.  Để "mở" các "khe trống" có thể xuất hiện tại các patch/mảng rời rạc, thực hiện phép co ảnh trước, sau đó là giãn ảnh. Quy trình này được gọi là [Mở/opening](https://en.wikipedia.org/wiki/Opening_(morphology)).  Thử bằng cách thực hiện phép giãn ảnh từ một ảnh đã qua phép co ảnh:

// Perform an opening by dilating the eroded image.

var opened = min.reduceNeighborhood({

 reducer: ee.Reducer.max(),

 kernel: uniformKernel

});

Map.addLayer(opened, binaryVis, 'opened');

* 1. **Đóng/Closing**.  Trái ngược với opening là [Đóng/cosing](https://en.wikipedia.org/wiki/Closing_(morphology)), hoặc phép giãn ảnh được thực hiện trước, sau đó là phép giãn ảnh.  Dùng để "đóng" các "khe trống" có thể xuất hiện trong các patch/mảng rời rạc trong ảnh đầu vào:

// Perform a closing by eroding the dilated image.

var closed = max.reduceNeighborhood({

 reducer: ee.Reducer.min(),

 kernel: uniformKernel

});

Map.addLayer(closed, binaryVis, 'closed');

* 1. Kiểm tra kỹ sự khác biện giữa mỗi phép xử lý hình thái học và đầu vào veg.  Điều chỉnh các toán tử hình thái học/toán tử morphological này bằng cách thay đổi kích thước và hình dạng của cửa sổ lọc (trong trường hợp này còn được gọi là [*phần tử cấu t*](https://en.wikipedia.org/wiki/Structuring_element)*ruc*, do tác động của nó đến hình thái đối tượng trong kết quả đầu ra), hoặc lặp đi lặp lại các quy trình.

## Cấu trúc ảnh/Texture

Kết cấu được định nghĩa như phép đo sự phân tán của DN trong khu vực lân cận. Có nhiều cách để [tính toán kết cẩu trong Earth Engine](https://developers.google.com/earth-engine/image_texture).

1. **Độ Lệch Chuẩn/Standard Deviation** (SD).  Độ lệch chuẩn đo độ phân tán của DN trong vùng lân cận. Một vùng lân cận không có kết cấu là vùng chỉ có một giá trị DN, SD=0. Tính độ lệnh chủa trong vùng lân cận cho ảnh NAIP với:

// Define a big neighborhood with a 7-meter radius kernel.

var bigKernel = ee.Kernel.square({

 radius: 7,

 units: 'meters'

});

// Compute SD in a neighborhood.

var sd = image.reduceNeighborhood({

 reducer: ee.Reducer.stdDev(),

 kernel: bigKernel

});

Map.addLayer(sd, {min: 0, max: 70}, 'SD');

1. **Entropy**.  Cho những ảnh đầu vào rời rạc, bạn có thể tính [entropy](https://en.wikipedia.org/wiki/Diversity_index#Shannon_index) cho vùng lân cận, trong đó entropy được xem như một chỉ số thể hiện tính đa dạng của DN diversity trong vùng lân cận:

// Compute entropy in a neighborhood.

var entropy = image.entropy(bigKernel);

Map.addLayer(entropy, {min: 1, max: 5}, 'entropy');

1. (*Không bắt buộc*)  **Ma trận đồng nhất mức xám độ /Gray-level co-occurrence matrices** (GLCM).  GLCM được tính bằng cách tạo một ma trận cỡ *M*x*M* cho một ảnh với *M* các giá trị có thể của DN, sau đó thực hiện nhập *i*,*j* bằng tần suất mà DN=*i* liền kề với DN=*j*.  Có rất nhiều thông số về kết cấu ảnh có thể thu được từ ma trận đó, trong đó có độ tương phản:

// Use the GLCM to compute a large number of texture measures.

var glcmTexture = image.glcmTexture(7);

// Display the 'contrast' results for the red, green and blue bands.

var contrastVis = {

 bands: ['R\_contrast', 'G\_contrast', 'B\_contrast'],

 min: 40,

 max: 2000

};

Map.addLayer(glcmTexture, contrastVis, 'contrast');

1. (*Không bắt buộc*)  **Thống kê không gian/Spatial statistics**.  Hai phương pháp kết cấu sử dụng thống kê không gian bao gồm Local [Moran's I](https://en.wikipedia.org/wiki/Moran%27s_I) và [Local Geary's C](https://en.wikipedia.org/wiki/Geary%27s_C) ([Anselin 1995](http://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.1995.tb00338.x/abstract)).  Tính local Geary's C với ảnh NAIP đầu vào, sử dụng:

// Create a list of weights for a 9x9 kernel.

var list = [1, 1, 1, 1, 1, 1, 1, 1, 1];

// The center of the kernel is zero.

var centerList = [1, 1, 1, 1, 0, 1, 1, 1, 1];

// Assemble a list of lists: the 9x9 kernel weights as a 2-D matrix.

var lists = [list, list, list, list, centerList, list, list, list, list];

// Create the kernel from the weights.

// Non-zero weights represent the spatial neighborhood.

var kernel = ee.Kernel.fixed(9, 9, lists, -4, -4, false);

// Use the max among bands as the input.

var maxBands = image.reduce(ee.Reducer.max());

// Convert the neighborhood into multiple bands.

var neighs = maxBands.neighborhoodToBands(kernel);

// Compute local Geary's C, a measure of spatial association.

var gearys = maxBands.subtract(neighs).pow(2).reduce(ee.Reducer.sum())

            .divide(Math.pow(9, 2));

Map.addLayer(gearys, {min: 20, max: 2500}, "Geary's C");

## Thay Đổi Độ Phân Giải và Nắn Ảnh

Earth Engine đã nỗ lực để có thể thực hiện toàn bộ các chức năng liên quan đến lưới chiếu và tỷ lệ do vậy mà bạn không cần phải thực hiện chúng. Tuy nhiên, trong nhiều trường hợp mà việc hiểu về lưới chiếu là rất quan trọng để có được kết quả bạn muốn. Đây là lúc để làm tìm hiểu về hàm reproject() trong các ví dụ trước.  Earth Engine yêu cầu thông tin đầu vào cho việc thực hiện lưới chiếu và tỷ lệ trong kết quả đầu ra.  Bản đồ được sử dụng trong phần thực hành có lưới chiếu [Mercator](http://epsg.io/3857) .  Tỷ lệ/Scale được xác định từ độ phóng của cửa sổ bản đồ.  Khi bạn thêm dữ liệu vào cửa sổ bản đồ, Earth Engine sẽ nắn lại các dữ liệu đầu vào này theo lưới chiếu Mercator, thay đổi độ phân giải (sử dụng nearest neighbor) để điều chỉnh độ phân giải pixel phù hợp với mức độ phóng to thu nhỏ của cửa sổ bản đồ, sau đó thực hiện các thao tác với ảnh đã được nắn chỉnh và thay đổi độ phân giải. Trong những ví dụ trước, khi chúng ta gọi reproject() nghĩa là đã ra lệnh cho thao tác trên được thực hiện các tính toán tại độ phân giải của ảnh đầu vào là: 1 meter.

1. Chạy lại code edge detection/dò biên ảnh cùng với nắn ảnh và không cùng với nắn ảnh (Tạo chú giải/comment lên toàn bộ phần gọi Map.addLayer() trừ phần nguyên gốc ban đầu):

// Zoom all the way in.

Map.centerObject(point, 21);

// Display edges computed on a reprojected image.

Map.addLayer(image.convolve(laplacianKernel), {min: 0, max: 255},

   'Edges with little screen pixels');

// Display edges computed on the image at native resolution.

Map.addLayer(edges, {min: 0, max: 255},

   'Edges with 1 meter pixels');

* 1. Điều xảy ra ở đây đó là phép chiếu được xác định trong reproject() được đẩy lùi lại gần với đầu vào, bắt tất cả các quá trình tính toán thực hiện trên phép chiếu đó.  Nếu bạn không định rõ, các tính toán sẽ được thực hiện theo lưới chiếu và tỷ lệ của bản đồ (Mercator) theo độ phân giải màn hình.

1. Bạn có thể điều khiển thay đổi độ phân giải của Earth Engine tại đầu vào với resample().  Theo mặc định, tất cả thay đổi độ phân giải được thực hiện với nearest neighbor.  Để thay đổi, gọi resample() cho đầu vào.  So sánh ảnh đầu vào, được thay đổi theo độ phân giải màn hình và với phương pháp resampling bilinear và bicubic:

// Resample the image with bilinear instead of nearest neighbor.

var bilinearResampled = image.resample('bilinear');

Map.addLayer(bilinearResampled, {}, 'input image, bilinear resampling');

// Resample the image with bicubic instead of nearest neighbor.

var bicubicResampled = image.resample('bicubic');

Map.addLayer(bicubicResampled, {}, 'input image, bicubic resampling');

* 1. Thử phóng to thu nhỏ, so sánh với ảnh đầu vào được resample với nearest neighbor (ví dụ không goi resample() cho đầu vào).

1. ***Bạn ít khi phải dùng*  reproject() *và* resample()**.  Không nên dùng reproject() hoặc resample() nếu không cần thiết.  Chúng được đưa ra trong bài học này chỉ với mục đích giới thiệu.



Bản quyền bài giảng [Creative Commons Attribution 4.0 International License](http://creativecommons.org/licenses/by/4.0/).

Copyright 2015-2016, Google Earth Engine Team