Lecture 09

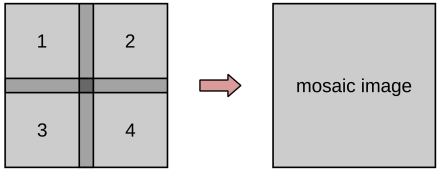
GEE Mosaicking and Compositing

2024-05-06

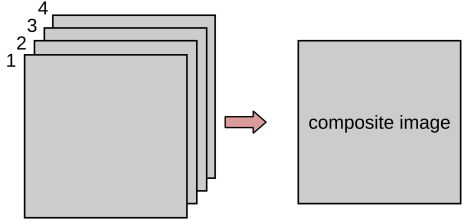
Sébastien Valade

VNIVERSIDAD NACIONAL
AVTÓNOMA DE
MEXICO

- **Mosaicking**: process of spatially assembling image datasets to produce a spatially continuous image

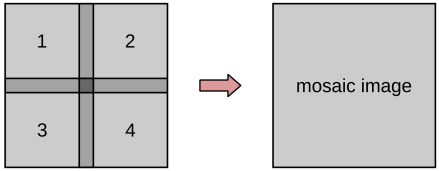


- **Compositing**: process of combining spatially overlapping images into a single image based on an aggregation function

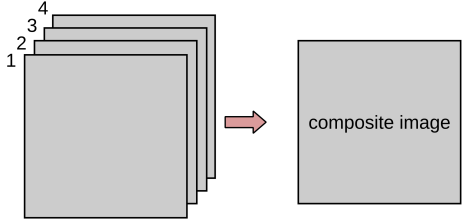


⇒ in GEE, these terms are used interchangeably (both operations are often performed together), see reference

- **Mosaicking**: process of spatially assembling image datasets to produce a spatially continuous image



- **Compositing**: process of combining spatially overlapping images into a single image based on an aggregation function



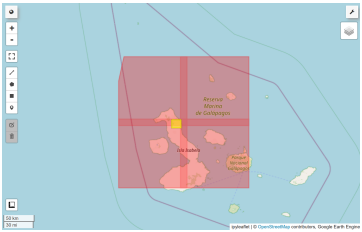
$\Rightarrow$ in GEE, these terms are used interchangeably (both operations are often performed together), see reference

Mosaicking images: **make a single image from multiple images**

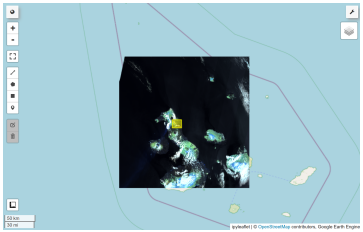
EX: make a single image from a lava flow whose extent overlaps several Sentinel-2 products

⇒ **mosaicking** will reduce the ImageCollection to a single image

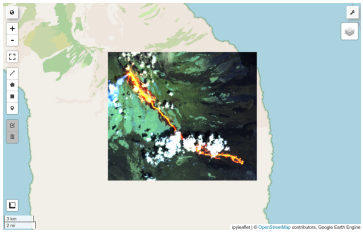
Footprints + AOI



Mosaic image + AOI



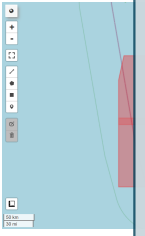
Mosaic image clipped to AOI



Mosaicking images: **make a single image from multiple images**

EX: make a single image from a lava flow whose extent overlaps several Sentinel-2 products

⇒ **mosaicking** will reduce the ImageCollection to a single image



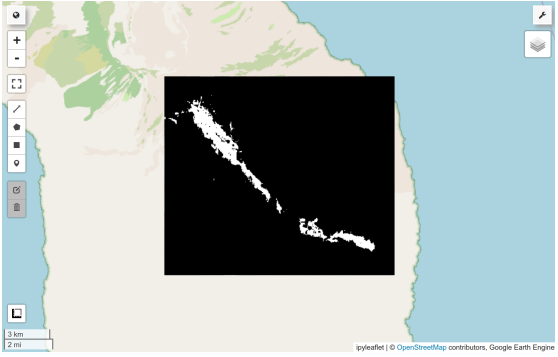
```python
# Get aoi and ImageCollection
aoi = ee.Geometry.Rectangle([-91.3348, -0.1229, -91.2003, -0.0069])
ic = (ee.ImageCollection('COPERNICUS/S2_HARMONIZED')
        .filterBounds(aoi)
        .filterDate('2022-01-11', '2022-01-12')
       )

# Mosaic and clip
image_mosaic = ic.mosaic()
image_mosaic_clip = image_mosaic.clip(aoi)
ic_footprints = ic.map(lambda img: ee.Feature(img.geometry()))

# Plot
vis_params = {'bands': ['B12', 'B11', 'B8A'], 'min': 0, 'max': 3000}  # Swir bands composition

Map = geemap.Map()
Map.addLayerControl()
Map.centerObject(image_mosaic_clip, 12)
Map.addLayer(image_mosaic, vis_params, 'image mosaic')
Map.addLayer(image_mosaic_clip, vis_params, 'image mosaic (clipped)')
Map.addLayer(ic_footprints.draw(color='red'), {'opacity': 0.5}, 'footprints')
Map.addLayer(aoi, {'color': 'yellow', 'opacity': 1}, 'AOI')
Map
```

Mosaicking layers: make a single image from multiple image layers

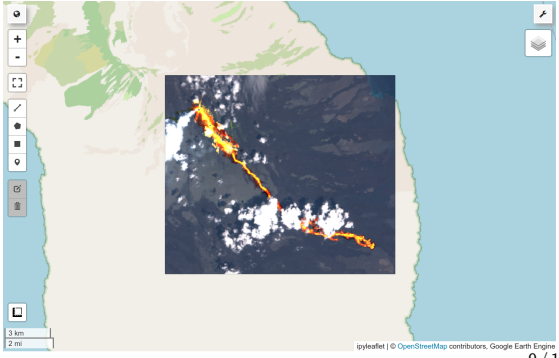
EX: make a single image from multiple image layers with different visualization parameters
⇒ using *masking* operations we can visualize the hot lava pixels with SWIR bands composition, and the background pixels with RGB bands composition

NB: The mosaic() method composites overlapping images according to their order in the collection: the last image in the collection is on top.

Hot pixels mask



Layer mosaic



Mosaicking layers: make a single image from multiple image layers

EX: make a single image from multiple image layers with different visualization parameters
⇒ using *masking* operations we can visualize the hot lava pixels with SWIR bands composition, and the background pixels with RGB bands composition

NB: The mosaic() method composites overlapping images according to their order in the collection: the last image in the collection is on top.

Hot pixels mask

Layer mosaic

```
# Detect hot pixels
mask_hotpixels = detect_hot_pixels(image_mosaic_clip)

# Mask
image_masked_bkgrd = image_mosaic_clip.updateMask(mask_hotpixels.eq(0))
image_masked_lava = image_mosaic_clip.updateMask(mask_hotpixels)

# Construct ImageCollection and mosaic  (last layer is on top)
vis_params_rgb = {'bands': ['B4', 'B3', 'B2'], 'min': 0, 'max': 3000, 'gamma': 1.25}
vis_params_swir = {'bands': ['B12', 'B11', 'B8A'], 'min': 0, 'max': 3000}
layer_mosaic = ee.ImageCollection([
            image_masked_bkgrd.visualize(**vis_params_rgb),
            image_masked_lava.visualize(**vis_params_swir),
            ]).mosaic()
```

Compositing example

EX: make a cloud-free (almost) image of Mexico

⇒ **compositing** using the *median* function will reduce the ImageCollection by taking the median value of each pixel over time

⇒ in regions where clouds are transient, the median value will tend to be cloud-free

Median composite image

Median composite image clipped

Compositing example

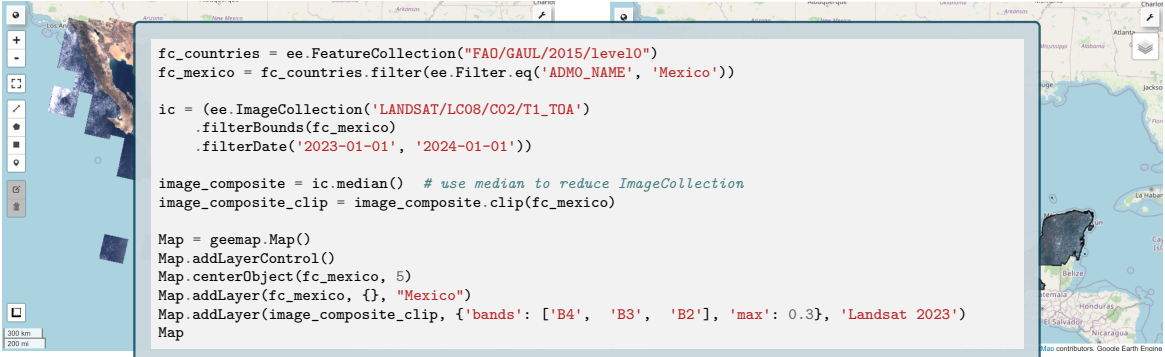
EX: make a cloud-free (almost) image of Mexico

⇒ **compositing** using the *median* function will reduce the ImageCollection by taking the median value of each pixel over time

⇒ in regions where clouds are transient, the median value will tend to be cloud-free

Median composite image

Median composite image clipped




```
fc_countries = ee.FeatureCollection("FAO/GAUL/2015/level0")
fc_mexico = fc_countries.filter(ee.Filter.eq('ADM0_NAME', 'Mexico'))

ic = (ee.ImageCollection('LANDSAT/LC08/C02/T1_TOA')
    .filterBounds(fc_mexico)
    .filterDate('2023-01-01', '2024-01-01'))

image_composite = ic.median()  # use median to reduce ImageCollection
image_composite_clip = image_composite.clip(fc_mexico)

Map = geemap.Map()
Map.addLayerControl()
Map.centerObject(fc_mexico, 5)
Map.addLayer(fc_mexico, {}, "Mexico")
Map.addLayer(image_composite_clip, {'bands': ['B4',  'B3',  'B2'], 'max': 0.3}, 'Landsat 2023')
Map
```