Appendix A. Supplementary Material

Understanding the spatial distribution of urban forests in China using Sentinel-2 images with Google Earth Engine

Qianwen Duan, Minghong Tan*, Yuxuan Guo, Xue Wang and Liangjie Xin

* tanmh@igsnrr.ac.cn

Contents

1. Top 100 prefecture-level cities in China of the highest urban forest cover in 2016

Table S1 Top 100 prefecture-level cities in China of the highest urban forest cover in 2016

Province	Prefecture-level city	Urban forest cover (%)	Province	Prefecture-level city	Urban forest cover (%)
Sichuan	Zigong	40.14	Anhui	Wuhu	25.00
Sichuan	Leshan	38.26	Anhui	Tongling	24.87
Sichuan	Bazhong	37.21	Sichuan	Panzhihua	24.78
Hubei	Shiyan	35.27	Guangxi	Chongzuo	24.77
Liaoning	Benxi	34.72	Hunan	Zhuzhou	24.39
Jiangsu	Nanjing	34.33	Shanghai	Shanghai	24.39
Sichuan	Ya'an	34.24	Shanxi	Linfen	24.36
Chongqing	Chongqing	33.38	Jiangsu	Huai'an	24.36
Sichuan	Luzhou	33.11	Guizhou	Tongren	24.34
Shaanxi	Tongchuan	33.02	Zhejiang	Jiaxing	24.01
Jiangsu	Yangzhou	32.95	Sichuan	Neijiang	24.00
Sichuan	Yibin	32.90	Liaoning	Fushun	23.81
Guangxi	Hechi	32.84	Zhejiang	Wenzhou	23.63
Anhui	Huangshan	32.41	Guangxi	Liuzhou	23.53
Sichuan	Meishan	31.65	Anhui	Huainan	23.35
Shaanxi	Ankang	31.44	Guangdong	Jiangmen	23.34
Shaanxi	Yan'an	31.18	Fujian	Longyan	23.28
Zhejiang	Ningbo	30.42	Jiangsu	Taizhou	23.26
Guangdong	Guangzhou	30.36	Anhui	Ma'anshan	22.62
Guangdong	Zhuhai	30.13	Liaoning	Tieling	22.54
Hunan	Zhangjiajie	30.09	Hubei	Yichang	22.23
Guangdong	Shenzhen	29.99	Hebei	Chengde	22.16
Sichuan	Chengdu	29.70	Xinjiang Uygur	Karamay	22.11
Zhejiang	Hangzhou	29.22	Hunan	Xiangtan	22.11
Sichuan	Mianyang	29.11	Liaoning	Dandong	21.98
Hainan	Sanya	28.94	Hunan	Huaihua	21.91
Guangxi	Guilin	28.85	Sichuan	Nanchong	21.91
Shaanxi	Shangluo	28.77	Beijing	Beijing	21.76
Jiangsu	Zhenjiang	28.75	Guangxi	Laibin	21.58
Guangxi	Baise	28.60	Guangxi	Wuzhou	21.53
Sichuan	Deyang	28.56	Jiangxi	Pingxiang	21.47
Fujian	Sanming	28.51	Guangxi	Qinzhou	21.44
Liaoning	Liaoyang	27.95	Sichuan	Dazhou	21.43
Zhejiang	Lishui	27.64	Fujian	Fuzhou	21.38
Zhejiang	Taizhou	27.56	Guangdong	Qingyuan	21.35
Fujian	Ningde	27.44	Henan	Zhengzhou	21.28
Hubei	Huangshi	27.14	Jiangsu	Nantong	21.25
Jiangsu	Wuxi	26.96	Jiangxi	Shangrao	21.16
Sichuan	Suining	26.90	Hainan	Hainan	21.08
Guangdong	Yunfu	26.87	Henan	Hebi	20.84

Sichuan	Guang'an	26.77	Liaoning	Shenyang	20.80
Fujian	Nanping	26.54	Guangdong	Foshan	20.67
Zhejiang	Shaoxing	26.53	Anhui	Chizhou	20.66
Guizhou	Zunyi	26.48	Guangdong	Zhaoqing	20.65
Jiangsu	Suzhou	26.46	Guangdong	Dongguan	20.64
Guangdong	Shaoguan	26.14	Shaanxi	Xi'an	20.45
Guizhou	Guiyang	26.11	Zhejiang	Huzhou	20.41
Zhejiang	Quzhou	25.96	Qinghai	Xining	20.29
Hunan	Chenzhou	25.94	Guangdong	Huizhou	20.21
Jiangsu	Changzhou	25.92	Guangxi	Yulin	20.16

2. Download link of urban area of China in 2016

https://code.earthengine.google.com/?asset=users/qwduan2013/2016new

We also uploaded this file in the Google Drive, the download link is:

https://drive.google.com/drive/folders/1Ehx1WA9QOTdjwnINcsN1tMwniX2_Dn04?usp=sharing

3. Scripts for image pre-processing, classification, and accuracy assessment

https://code.earthengine.google.com/e89b3c20e67562534cd3adae620655a5

Note: The validation points in this example was for zone A.

The complete code is following. If you download the file of urban area of China in 2016 and copy the code in GEE platform, this can run.

```
// Function to mask clouds using the Sentinel-2 QA band.
function maskS2clouds(image) {
  var qa = image.select('QA60');

// Bits 10 and 11 are clouds and cirrus, respectively.
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;

// Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0).and(
                 qa.bitwiseAnd(cirrusBitMask).eq(0));
// Return the masked and scaled data, without the QA bands.
  return image.updateMask(mask).divide(10000)
      .select("B.*")
      .copyProperties(image, ["system:time_start"]);
}

//Function add ndvi
function S2ndvi(image) {
  var ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI');
   return image.addBands(ndvi)
                .float()
}
```

```
//Function add ndbi
function S2ndbi(image) {
   var ndbi = image.normalizedDifference(['B11', 'B8']).rename('NDBI');
     return image.addBands(ndbi)
                         .float()
}

//Function add ndwi
function S2ndwi(image) {
   var ndwi = image.normalizedDifference(['B3', 'B8']).rename('NDWI');
     return image.addBands(ndwi)
                         .float()
}

// Map the function over one year of data and take the median.
// Load Sentinel-2 TOA reflectance data.
var collection = ee.ImageCollection('COPERNICUS/S2')
       .filterDate('2016-04-01', '2016-9-30')
       // Pre-filter to get less cloudy granules.
       .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))
       .map(maskS2clouds)
       .map(S2ndvi)
       .map(S2ndwi)
       .map(S2ndbi)
var composite = collection.median()

// Load a table of state boundaries and filter.
var fc = ee.FeatureCollection('users/qwduan2013/2016new');

// Clip urben area.
var china=composite.clip (fc);

//Select the bands for training
var bands = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8','B11','NDVI','NDWI','NDBI'];

//From Fusion Table
var points = ee.FeatureCollection('ft:1xg8pt-RYI7QvuRUD_dQuRkI5cnM3_N7f9aXS_zLk');
var testingPartitionA = ee.FeatureCollection('ft:1eqZlJKW3-rVcbqm5dxXmlrrNcj9W3kjIDM8ITgAW');

// Sample the input imagery to get a FeatureCollection of training data.
var training = china.select(bands).sampleRegions({
   collection: points,
   properties: ['class'],
```

```
    scale: 15
});

// Make a Random Forest classifier and train it.
var classifier = ee.Classifier.randomForest({
    numberOfTrees : 40,
    minLeafPopulation : 1,
    seed : 0
})
    .train({
    features: training,
    classProperty: 'class',
    inputProperties: bands
});

// Classify the input imagery.
//var classified = china.select(bands).classify(classifier);
var classified = china.select(bands).classify(classifier);
// Define a palette for the Land Use classification.
var palette = [
    'D3D3D3', // urban (0)    // grey
    '008000' //    forest (1) // green
];
// Get a confusion matrix representing resubstitution accuracy.
var trainAccuracy = classifier.confusionMatrix();
print('Resubstitution error matrix: ', trainAccuracy);
print('Training overall accuracy: ', trainAccuracy.accuracy());

// Display the classification result and the input image.
Map.setCenter(116.3, 39.90);
Map.addLayer(classified, {min: 0, max: 1, palette: palette}, 'Land Use Classification');
Map.addLayer(china, {bands: ['B4', 'B3', 'B2'], max: 0.5, gamma: 2}, 'S2 Image', false);

// Sample the input imagery to get a FeatureCollection of test data.
var testdata = china.select(bands).sampleRegions({
    collection: testingPartitionA,
    properties: ['class'],
    scale: 15
});

// // Classify the test FeatureCollection.
  var test = testdata.classify(classifier,'classification');

// // Print the confusion matrix.
```

```
var confusionMatrix = test.errorMatrix('class', 'classification');
print('RF testing error matrix', confusionMatrix);
print('RF testing accuraccy', confusionMatrix.accuracy());
```

4. Download link of land cover dataset

It is difficult to upload the mosaic result which is large. Therefore, we provided a download link to the dataset which should be mosaicked before using.

0 – non-urban forest; 1- urban forest

https://drive.google.com/open?id=13grxfNDrsJsWcHPMC_M53y3NK_BA40eS