```
import numpy as np
import cv2
import scipy.io
import os
from numpy.linalg import norm
from matplotlib import pyplot as plt
from numpy.linalg import det
from numpy.linalg import inv
from scipy.linalg import rq
from numpy.linalg import svd
import matplotlib.pyplot as plt
import numpy as np
import math
import random
import sys
from scipy import ndimage, spatial
from tqdm.notebook import tqdm, trange
```

Importing Drive (Dataset-University)

[ ] ↳ *8 cells hidden*

Reading all Files from Folder

[ ] ↳ *1 cell hidden*

Reading GPS and Metdata information

```
from PIL import Image, ExifTags
img = Image.open(f"{all_files_path[0]}")
exif = { ExifTags.TAGS[k]: v for k, v in img._getexif().items() if k in ExifTags.TAGS }
```

```
from PIL.ExifTags import TAGS

def get_exif(filename):
    image = Image.open(filename)
    image.verify()
    return image._getexif()

def get_labeled_exif(exif):
    labeled = {}
    for (key, val) in exif.items():
        labeled[TAGS.get(key)] = val

    return labeled

exif = get_exif(f"{all_files_path[0]}")
labeled = get_labeled_exif(exif)
print(labeled)
```

```
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
```

```
print(TAGS)
```

```
from PIL.ExifTags import GPSTAGS

def get_geotagging(exif):
    if not exif:
```

```
    if not exif:
        raise ValueError("No EXIF metadata found")

    geotagging = {}
    for (idx, tag) in TAGS.items():
        if tag == 'GPSInfo':
            if idx not in exif:
                raise ValueError("No EXIF geotagging found")

            for (key, val) in GPSTAGS.items():
                if key in exif[idx]:
                    geotagging[val] = exif[idx][key]
    return geotagging
```

```
#all_files_path = left_files_path[::-1] + right_files_path[1:]
for file1 in all_files_path:
  exif = get_exif(f"{file1}")
  geotags = get_geotagging(exif)
  print(geotags)
  print(ok)
```

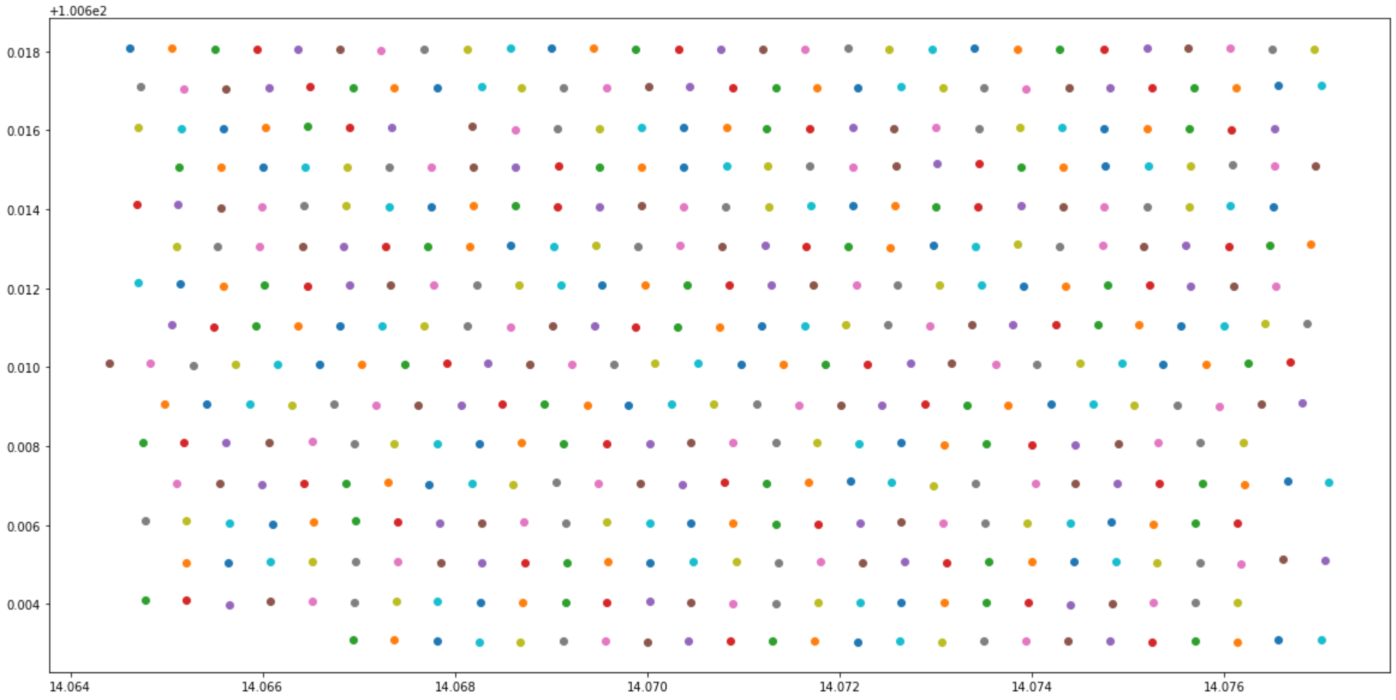
```
    1000000)), 'GPSLongitudeRef': 'E', 'GPSLongitude': ((100, 1), (37, 1), (5068784, 1000000)), 'GPSAltitudeRef': b'\x00', 'GPSAltitude': (2548340, 10000), 'GPSTimeStamp': ((5, 1), (23, 1), (43139, 1000)), 'GPSStatus': 'A', 'GPSMapDatum':
```

```
def get_decimal_from_dms(dms, ref):

    degrees = dms[0][0] / dms[0][1]
    minutes = dms[1][0] / dms[1][1] / 60.0
    seconds = dms[2][0] / dms[2][1] / 3600.0

    if ref in ['S', 'W']:
        degrees = -degrees
        minutes = -minutes
        seconds = -seconds

    return round(degrees + minutes + seconds, 5)

def get_coordinates(geotags):
    lat = get_decimal_from_dms(geotags['GPSLatitude'], geotags['GPSLatitudeRef'])

    lon = get_decimal_from_dms(geotags['GPSLongitude'], geotags['GPSLongitudeRef'])

    return (lat,lon)
```

Getting and Storing all Geolocations

```
all_geocoords = []
plt.figure(figsize = (20,10))
for file1 in tqdm(all_files_path):
  exif = get_exif(f"{file1}")
  geotags = get_geotagging(exif)
  #print(get_coordinates(geotags))
  geocoord = get_coordinates(geotags)
  all_geocoords.append(geocoord)
  plt.scatter(x=geocoord[0], y=geocoord[1])
```

```
100%                                        443/443 [00:02<00:00, 168.35it/s]
```



```
!pip install pyproj
```

```
Collecting pyproj
  Downloading https://files.pythonhosted.org/packages/11/1d/1c54c672c2faf08d28fe78e15d664c048f786225bef95ad87b6c435cf69e/pyproj-3.1.0-cp37-cp37m-manylinux2010_x86_64.whl (6.6MB)
     |████████████████████████████████| 6.6MB 3.2MB/s
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from pyproj) (2020.12.5)
Installing collected packages: pyproj
Successfully installed pyproj-3.1.0
```

```
!pip install gmplot
```

```
Collecting gmplot
  Downloading https://files.pythonhosted.org/packages/2f/2f/45399c0a3b75d22a6ece1a1732a1670836cf284de7c1f91379a8d9b666a1/gmplot-1.4.1-py3-none-any.whl (164kB)
     |████████████████████████████████| 174kB 14.9MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from gmplot) (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->gmplot) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->gmplot) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->gmplot) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->gmplot) (1.24.3)
Installing collected packages: gmplot
Successfully installed gmplot-1.4.1
```

```
print(np.min(np.array(all_geocoords)[:len1,0]),np.max(np.array(all_geocoords)[:len1,0]))
```

```
14.06462 14.077
```

```
print(np.min(np.array(all_geocoords)[:len1,1]),np.max(np.array(all_geocoords)[:len1,1]))
```

```
100.61506 100.61808
```

```
print(all_geocoords[int(len1/2)][0],all_geocoords[int(len1/2)][1])
```

```
14.06782 100.61706
```

Getting Bounds for plotting Polygon

This is still under-progress (almost completed) due to partial plotting of polygon by gmplot, so this will not be seen in the current plot, will be working on finishing this.

```
def get_geoloc_bounds(l, n):
    index_list = [None] + [i for i in range(1, len(l)) if abs(l[i] - l[i - 1]) > n] + [None]
    return [l[index_list[j - 1]:index_list[j]] for j in range(1, len(index_list))]
```

```
example =list(np.array(all_geocoords)[:,1])
```

```
print(list(np.array(all_geocoords)[:40,1]))
```

```
.61807, 100.61807, 100.61804, 100.61804, 100.61804, 100.61806, 100.61806, 100.61807, 100.61806, 100.61805, 100.61807, 100.61806, 100.61806, 100.61806, 100.61808, 100.61808, 100.61807, 100.61805, 100.61806, 100.61712, 100.61712, 100.6170
```

```
print(len(example))
```

```
101
```

```
for i in range(90,91):
  num = 1*i*1e-5
  split =get_geoloc_bounds(example, num)
```

```
print(len(split))
```

```
16
```

Get upper and lower bound indices of each section

```
len_tot_split = 0
indx_lst = []
for num,each in enumerate(split):
  len_each_split = len(each)
  first_index = len_tot_split
  len_tot_split += len_each_split
  last_index = len_tot_split-1
  print(first_index,last_index)
  if num==0:
    continue
  indx_lst.append(first_index)
  indx_lst.append(last_index)
  #indx_lst_all.append(indx_lst)
```

```
0 28
29 57
58 84
85 112
113 140
141 168
169 196
197 224
225 253
254 281
282 308
309 336
337 363
364 391
392 418
419 442
```

```
lon_bounds = [list(np.array(all_geocoords)[:,1])[i] for i in indx_lst]
```

```
lat_bounds = [list(np.array(all_geocoords)[:,0])[i] for i in indx_lst]
```

Ideas for Image Registration of Geo-tagged Images

1) Online Method using Google Maps API through GmPlot

(Not useful when internet connection is weak/remote locations)

Creating Google Map Object using API Key and Gmplot

```
import gmplot
len1 = len(all_files_path)

# Create the map plotter:
apikey = '' # (It's hidden because it's a private key)

mid_lat = all_geocoords[int(len1/2)][0]
mid_lon = all_geocoords[int(len1/2)][1]

latMax = np.max(np.array(all_geocoords)[:len1,0])
latMin = np.min(np.array(all_geocoords)[:len1,0])



lngMax = np.max(np.array(all_geocoords)[:len1,1])
lngMin = np.min(np.array(all_geocoords)[:len1,1])


bounds = {'north':latMax, 'south':latMin, 'east':lngMax, 'west':lngMin}

gmap = gmplot.GoogleMapPlotter(mid_lat, mid_lon, 19, apikey=apikey,fit_bounds = bounds,tilt=45)


# Mark a hidden gem:
#gmap.marker(all_geocoords[0][0], all_geocoords[0][1], color='cornflowerblue')
```

Creating Marker object as well as embedding link of each image on your desktop as each marker

```
for count,file1 in enumerate(all_files_path[:len1]):
  fname = file1.split('/')[-1]
  img_tag = f"C:/Users/User%20Default/Downloads/RGB-img/Uni_img/{fname}"
  gmap.marker(all_geocoords[count][0], all_geocoords[count][1], color='purple',info_window =f'<a href={img_tag}>{fname} <br/> {(all_geocoords[count][0],all_geocoords[count][1])} </a>')
```

```
#gmap.polygon(np.array(all_geocoords)[:len1,0], np.array(all_geocoords)[:len1,1], face_color='green', edge_color='cornflowerblue', edge_width=5,face_alpha=0.6)
```

```
gmap.polygon(lat_bounds, lon_bounds, face_color='blue', edge_color='cornflowerblue', edge_width=5,face_alpha=0.6)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-23-886adf5ce43a> in <module>()
----> 1 gmap.polygon(lat_bounds, lon_bounds, face_color='blue', edge_color='cornflowerblue', edge_width=5,face_alpha=0.6)

NameError: name 'lat_bounds' is not defined
```

SEARCH STACK OVERFLOW

Saving the GMap plot

[ ] ↳ 1 cell hidden

Video Link of Output

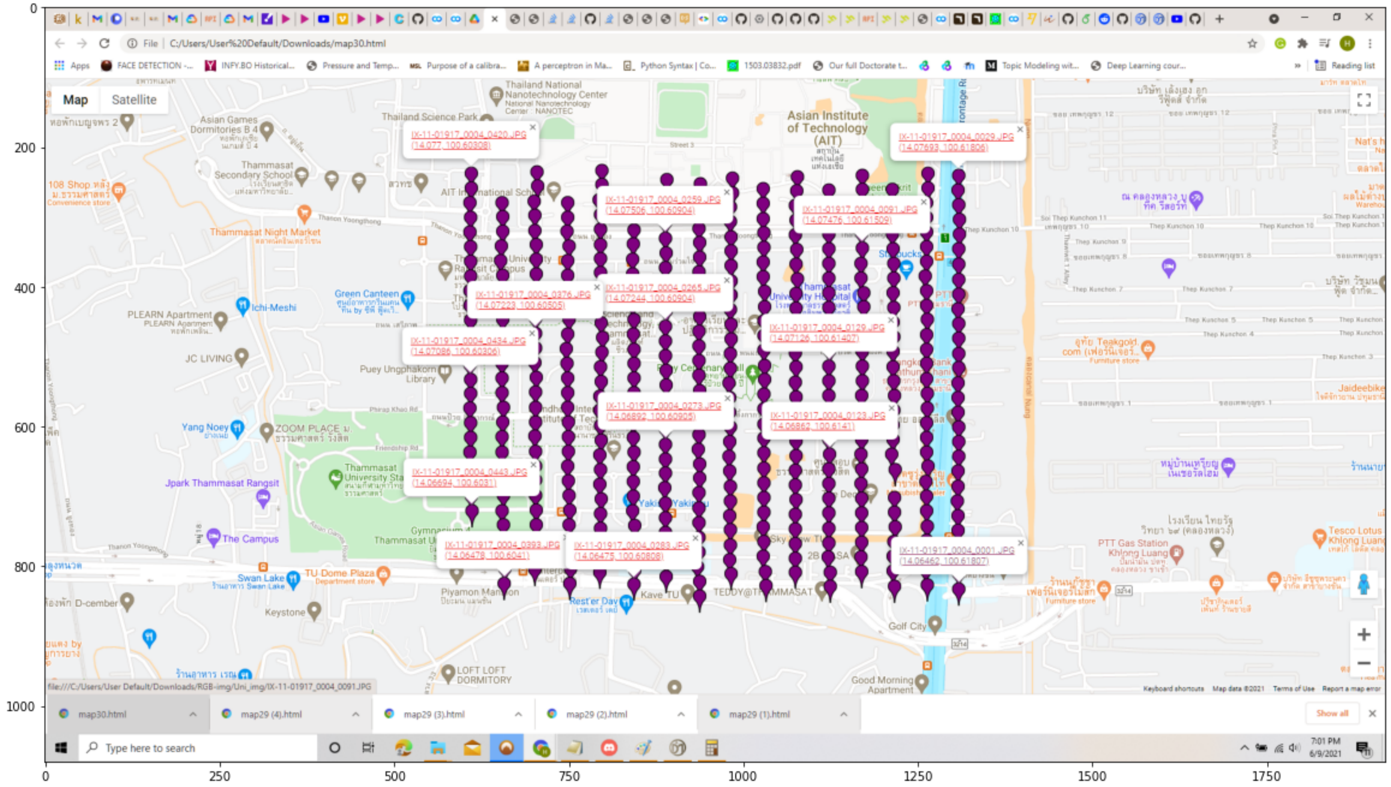
https://www.loom.com/share/f7534dbe837541e7b2ea9611580c6ce6

Screenshot of the Output

```
img_scrnsht = cv2.imread('drive/MyDrive/Screenshot_gmaps_gelocation_marker_embed_443_images.png')
```

```
plt.figure(figsize = (20,20))

plt.imshow(img_scrnsht)
```

```
<matplotlib.image.AxesImage at 0x7fa5381bfe90>
```



0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750
C:/Users/User%20Default/Downloads/map30.html
Map
Satellite
Thailand National Nanotechnology Center
Thailand Science Park
Asian Institute of Technology (AIT)
AIT International School
Thammasat University Rangsit Campus
Thammasat Night Market
Puey Ungphakorn Library
Thammasat University Hospital
Golf City
IX-11-01917_0004_0420.JPG (14.077, 100.60308)
IX-11-01917_0004_0029.JPG (14.07693, 100.61806)
IX-11-01917_0004_0259.JPG (14.07506, 100.60904)
IX-11-01917_0004_0091.JPG (14.07476, 100.61509)
IX-11-01917_0004_0376.JPG (14.07223, 100.60505)
IX-11-01917_0004_0265.JPG (14.07244, 100.60904)
IX-11-01917_0004_0434.JPG (14.07086, 100.60306)
IX-11-01917_0004_0129.JPG (14.07126, 100.61407)
IX-11-01917_0004_0273.JPG (14.06892, 100.60905)
IX-11-01917_0004_0123.JPG (14.06862, 100.6141)
IX-11-01917_0004_0443.JPG (14.06694, 100.6031)
IX-11-01917_0004_0393.JPG (14.06478, 100.6041)
IX-11-01917_0004_0283.JPG (14.06475, 100.60808)
IX-11-01917_0004_0001.JPG (14.06462, 100.61807)
file:///C:/Users/User Default/Downloads/RGB-img/Uni_img/IX-11-01917_0004_0091.JPG
map30.html
map29 (4).html
map29 (3).html
map29 (2).html
map29 (1).html
Show all
Type here to search
7:01 PM
6/9/2021

Extra Stuff

[ ] ↳ 30 cells hidden

Ideas for Image Registration of Geo-tagged Images

2) Offline Method using Matplotlib

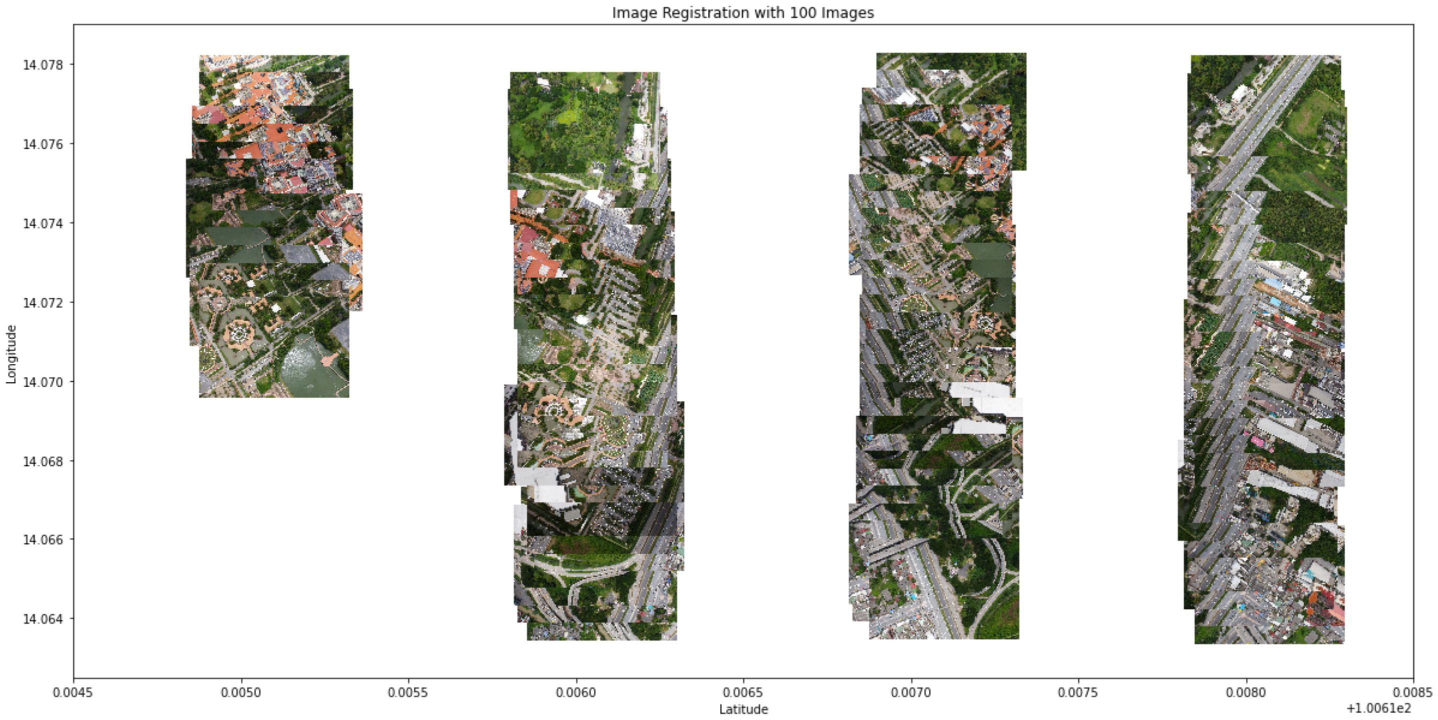
(Works offline but not overlayed on a map)

```
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
```

2 a.) Plotting Images on respective geo-locations

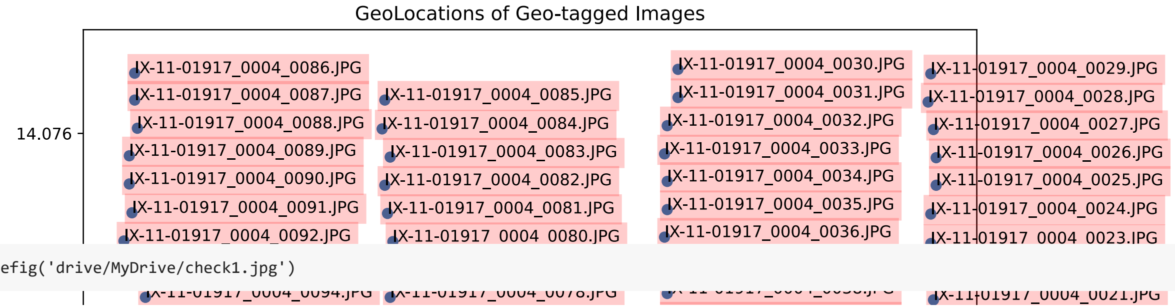
(Obscures images, different to decipher if images are missing/blurred,etc.)

```
fig, ax = plt.subplots()
fig.set_size_inches(20,10)
ax.set_xlabel('Latitude')
ax.set_ylabel('Longitude')
ax.set_xlim(100.6145,100.6185)
len1 = 100
ax.set_title(f'Image Registration with {len1} Images')
ax.set_ylim(14.0625,14.079)

ax.plot(np.array(all_geocoords)[:len1,0], np.array(all_geocoords)[:len1,1],linestyle='None')

def aerial_images_register(x, y,ax=None):
    ax = ax or plt.gca()
    for count,points in enumerate(zip(x,y)):
        lat,lon = points
        image = plt.imread(all_files_path[count])
        #print(ax.figure.dpi)
        im = OffsetImage(image, zoom=1.5/ax.figure.dpi)
        im.image.axes = ax
        ab = AnnotationBbox(im, (lat,lon), frameon=False, pad=0.0,)

        ax.add_artist(ab)

aerial_images_register( np.array(all_geocoords)[:len1,1],np.array(all_geocoords)[:len1,0], ax=ax)
```



2 b.) Embed the Images on respective geo-locations markers so as to take care of problem in 2 a)

```
import matplotlib.pyplot as plt
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("svg")

len1 = 100
fig, ax = plt.subplots()
fig.set_size_inches(10,10)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('GeoLocations of Geo-tagged Images')

ax.scatter(np.array(all_geocoords)[:len1,1], np.array(all_geocoords)[:len1,0])
#text = ax.annotate("Link", xy=(2,5), xytext=(2.2,5.5),
#                    url='http://matplotlib.org',
#                    bbox=dict(color='w', alpha=1e-6, url='http://matplotlib.org'))
def hover(event):
  vis = annot.get_visible()
  if event.inaxes == ax:
      cont, ind = sc.contains(event)
      if cont:
          update_annot(ind)
          annot.set_visible(True)
          fig.canvas.draw_idle()
      else:
          if vis:
              annot.set_visible(False)
              fig.canvas.draw_idle()
for count,file1 in enumerate(all_files_path[:len1]):
  fname = file1.split('/')[-1]
  img_tag = f"C:/Users/User%20Default/Downloads/RGB-img/Uni_img/{fname}"
  txt = plt.text(all_geocoords[count][1], all_geocoords[count][0],f'{fname}' , url=file1)
  txt.set_bbox(dict(color='r', alpha=0.2, url=txt.get_url()))
  fig.canvas.mpl_connect("motion_notify_event", hover)
```



```
fig.savefig('drive/MyDrive/check1.jpg')
```

Ideas for Image Registration of Geo-tagged Images

3) Offline Method using Folium

(Works offline and and overlayed on map)

```
!pip install folium
```

```
    Requirement already satisfied: folium in /usr/local/lib/python3.7/dist-packages (0.8.3)
    Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from folium) (2.11.3)
    Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from folium) (1.19.5)
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from folium) (1.15.0)
    Requirement already satisfied: branca>=0.3.0 in /usr/local/lib/python3.7/dist-packages (from folium) (0.4.2)
    Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from folium) (2.23.0)
    Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2->folium) (2.0.1)
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->folium) (1.24.3)
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->folium) (2.10)
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->folium) (2020.12.5)
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->folium) (3.0.4)
```

```
import folium
from folium import features
from scipy.spatial import ConvexHull
```

```
#Reference: https://nbviewer.jupyter.org/github/python-visualization/folium/blob/master/examples/Polygons_from_list_of_points.ipynb


def create_convexhull_polygon(
    map_object, list_of_points, layer_name, line_color, fill_color, weight, text
):

    # Since it is pointless to draw a convex hull polygon around less than 3 points check len of input
    if len(list_of_points) < 3:
        return

    # Create the convex hull using scipy.spatial
    form = [list_of_points[i] for i in ConvexHull(list_of_points).vertices]

    # Create feature group, add the polygon and add the feature group to the map
    fg = folium.FeatureGroup(name=layer_name)
    fg.add_child(
        folium.vector_layers.Polygon(
            locations=form,
            color=line_color,
            fill_color=fill_color,
            weight=weight,
            popup=(folium.Popup(text)),
        )
    )
    map_object.add_child(fg)
    return map_object
```

Creating Folium Map Object

```
len1 = len(all_files_path)
mid_lat = all_geocoords[int(len1/2)][0]
mid_lon = all_geocoords[int(len1/2)][1]

latMax = np.max(np.array(all_geocoords)[:len1,0])
latMin = np.min(np.array(all_geocoords)[:len1,0])



lngMax = np.max(np.array(all_geocoords)[:len1,1])
lngMin = np.min(np.array(all_geocoords)[:len1,1])

SJER_map = folium.Map([mid_lat,mid_lon],
                  zoom_start=15,min_lat=latMin, max_lat = latMax, min_lon = lngMin, max_lon=lngMax)
```

Creating Marker object as well as embedding link of each image on your desktop on each marker and adding to the Map Object

```
for count,file1 in enumerate(all_files_path[:len1]):
  fname = file1.split('/')[-1]
  img_tag = f"C:/Users/User%20Default/Downloads/RGB-img/Uni_img/{fname}"
  mk = features.Marker([all_geocoords[count][0], all_geocoords[count][1]],popup=f'<a href={img_tag}>{fname} <br/> {(all_geocoords[count][0],all_geocoords[count][1])} </a>',icon=folium.Icon(color="darkpurple"))
  SJER_map.add_child(mk)
```

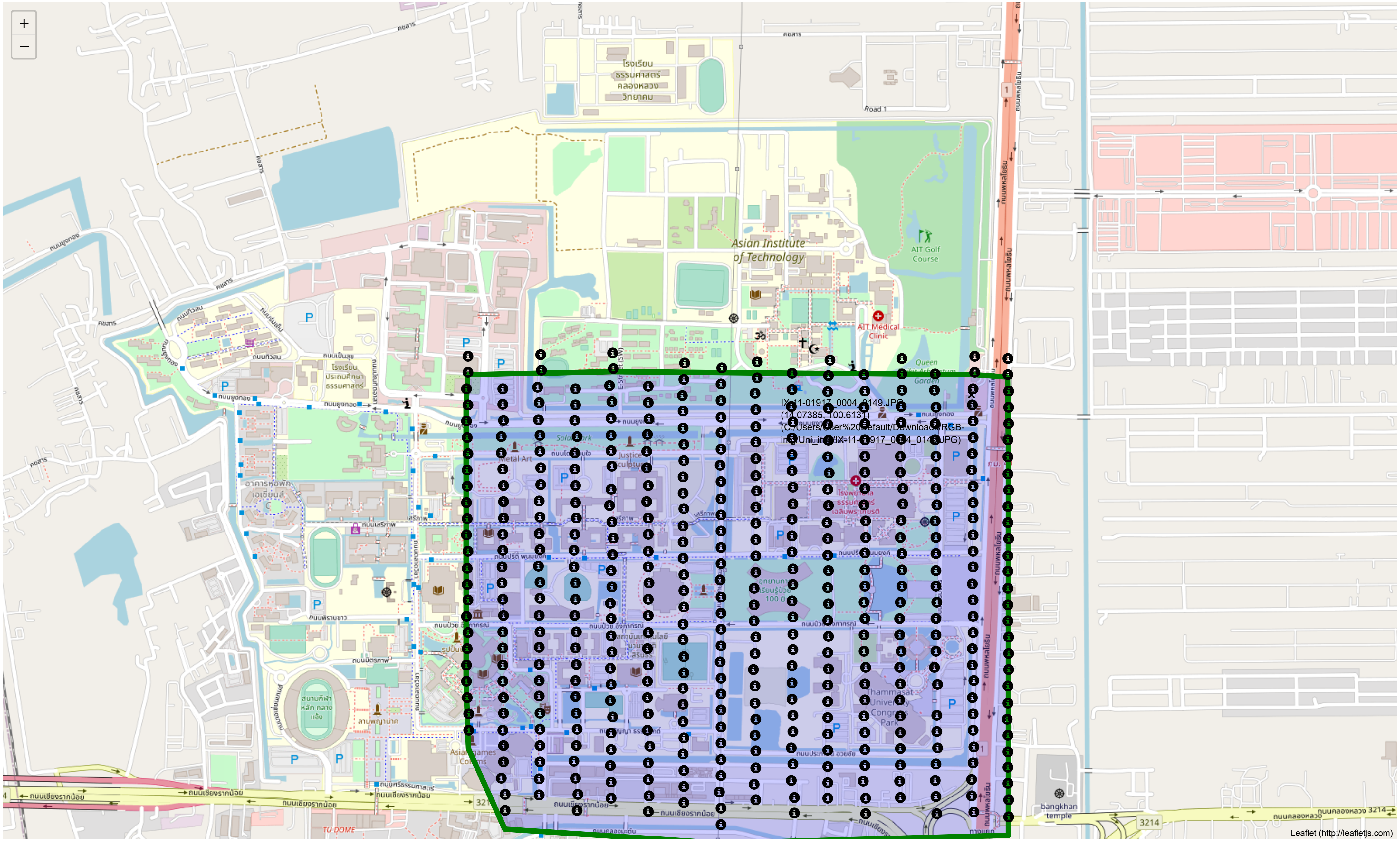
```
list_of_points = list(zip(np.array(all_geocoords)[:len1,0], np.array(all_geocoords)[:len1,1]))
```

Creating and Drawing Polygon on the list of (lat,lon) points

```
SJER_map = create_convexhull_polygon(
    SJER_map,
    list_of_points,
    layer_name="Boundary",
    line_color="green",
    fill_color="blue",
    weight=7,
    text="Boundary",
)
```

Output Map

```
SJER_map
```



IX-11-01917_0004_0149.JPG
(14.07385, 100.6131)
(C:/Users/User%20Default/Downloads/RGB-img/Uni_img/IX-11-01917_0004_0149.JPG)

+
−
Asian Institute of Technology
AIT Golf Course
AIT Medical Clinic
Queen Sirikit Arboretum Garden
Thammasat University Congress Park
Asian games Columns
bangkhan temple
Road 1
TU DOME
Solar Park
Metal Art
Justice sculpture
3214
Leaflet (http://leafletjs.com)

```
SJER_map.save('drive/MyDrive/off_map2.html')
```

Video Link of Output

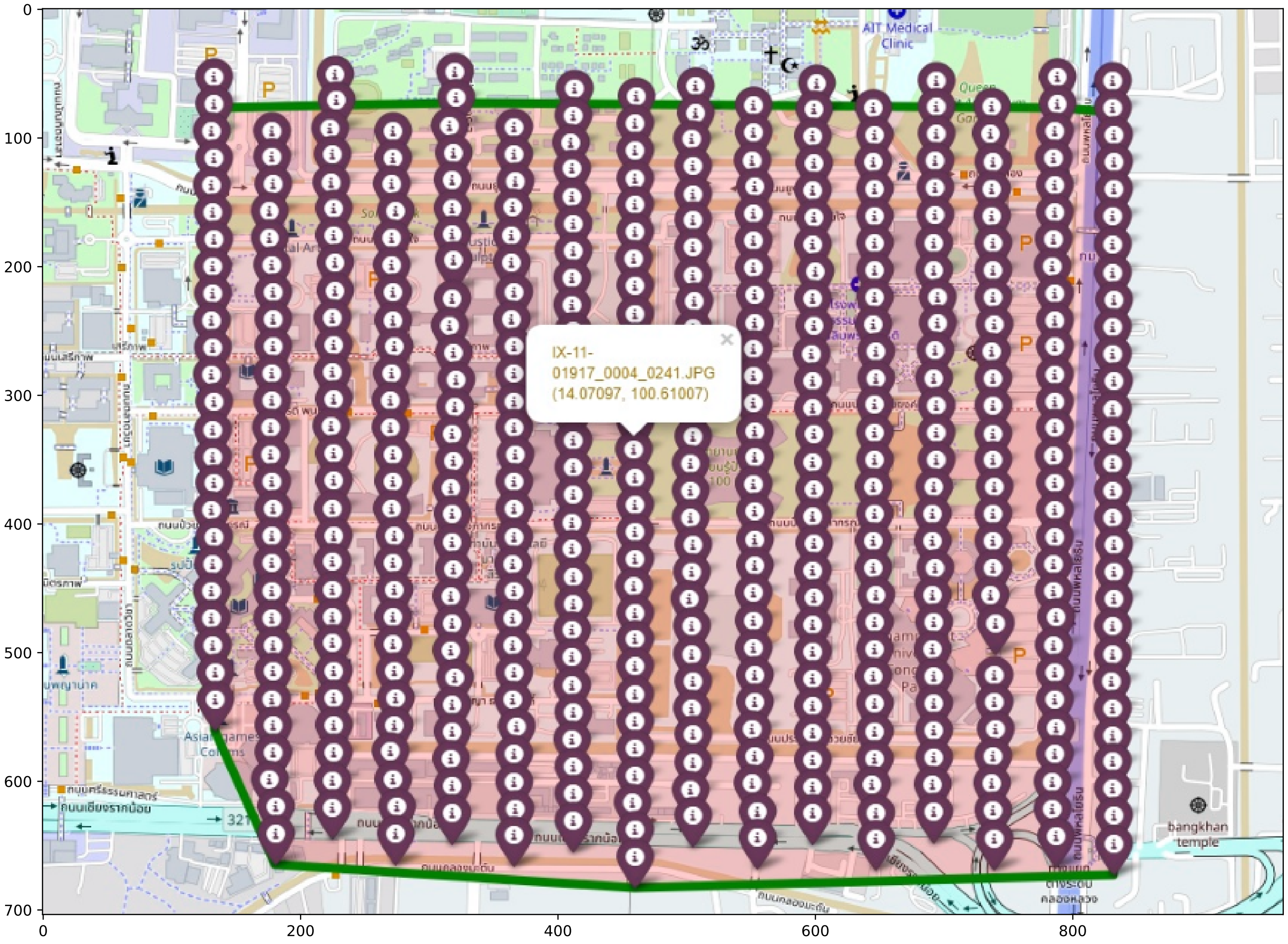
https://www.loom.com/share/2e632e81f49e4578afd7a7512d8b865f

Screenshot of the Output

```
img_scrnsht = cv2.imread('drive/MyDrive/Screenshot_gmaps_gelocation_marker_embed_443_images_offline.jpg')
```

```
plt.figure(figsize = (15,15))

plt.imshow(img_scrnsht)
```

```
<matplotlib.image.AxesImage at 0x7f258f97c890>
```



Reading images and Extracting SuperPoint Keypoints and Descriptors from each image

[ ] ↳ *15 cells hidden*

Loading and Initialing the SuperPoint Pretrained Network

[ ] ↳ *1 cell hidden*

Now Extracting Keypoints and Descriptors from all images and storing them

[ ] ↳ *7 cells hidden*

Image Matching (Robust) through RANSAC and Homography Matrix computation

[ ] ↳ *8 cells hidden*

Auto-Selection/Ordering of Images (Complete)

[ ] ↳ *20 cells hidden*

Perspective Transformation b/w consecutive pairs through the computed Homography Matrices

[ ] ↳ *6 cells hidden*

Final Mosaiced Image (with 22 images)

[ ] ↳ *1 cell hidden*

To-Do Tasks

- Seam Removal
- Improve On this Enhancement
- Extend to 50 images

[ ] ↳ *1 cell hidden*

✓ 5s completed at 4:28 PM