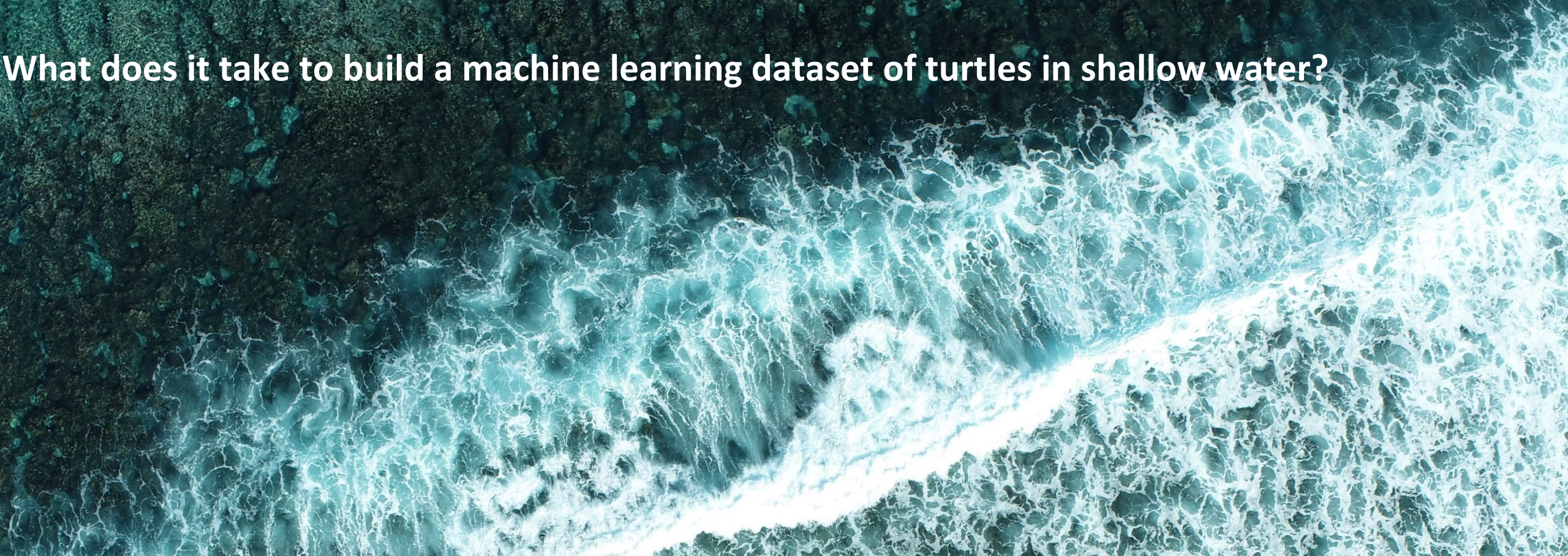


# What does it take to build a machine learning dataset of turtles in shallow water?



Nick Mortimer | 1-Feb-2024





 Flight time  
**30 MINS** <sup>[1]</sup>

 Control range  
**7 KM** <sup>[2]</sup>

 Speed  
**72 KM/H**

 Video resolution  
**4K 60fps**

 Sensor range  
**30 M** <sup>[3]</sup>

 Obstacle sensing  
**5 DIRECTION**

Over 80 Surveys

Over 120km<sup>2</sup> Surveyed

Over 90,000 Images

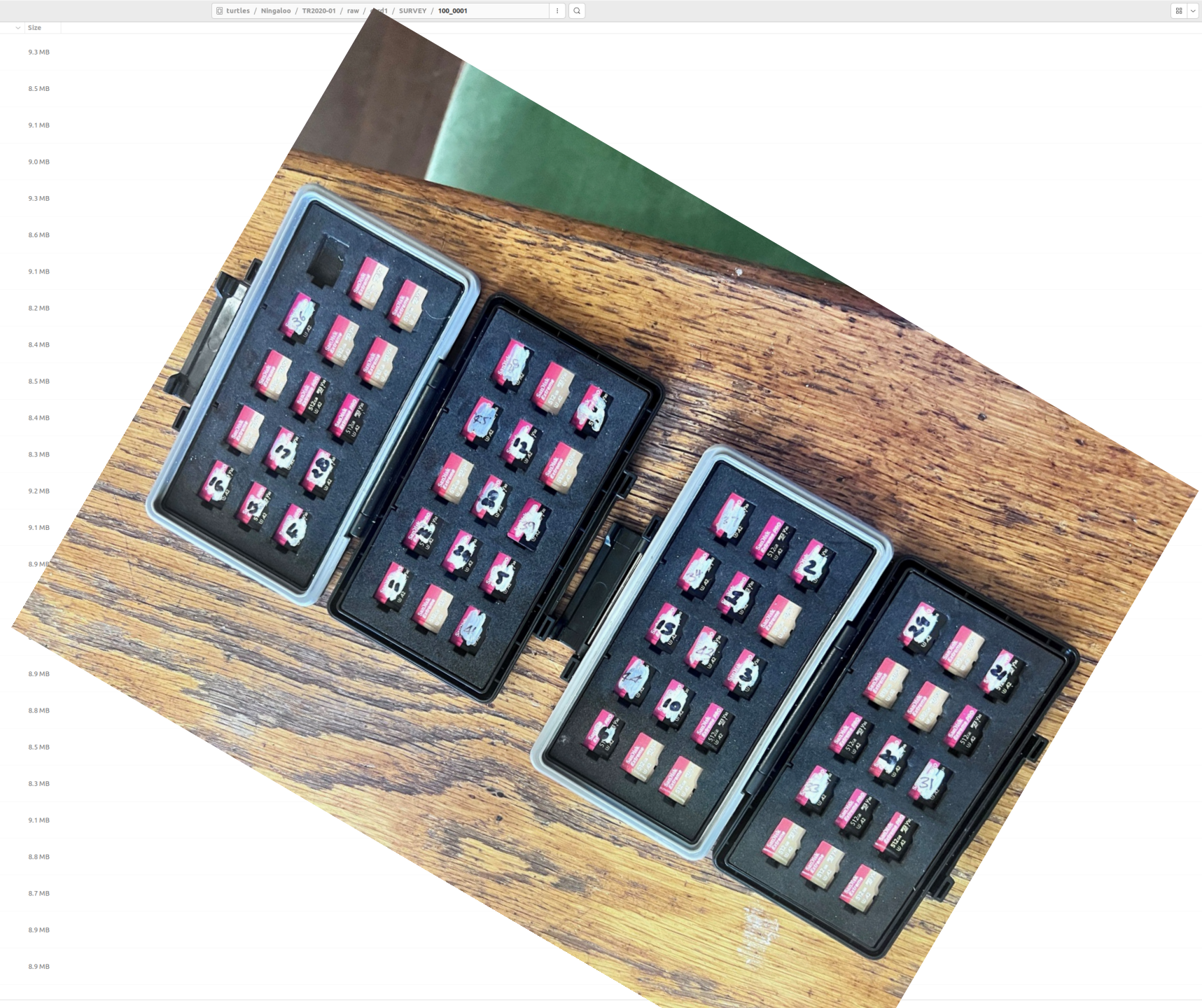
© Mapbox © OpenStreetMap © Maxar

© Mapbox © OpenStreetMap © Maxar

© Mapbox © OpenStreetMap © Maxar

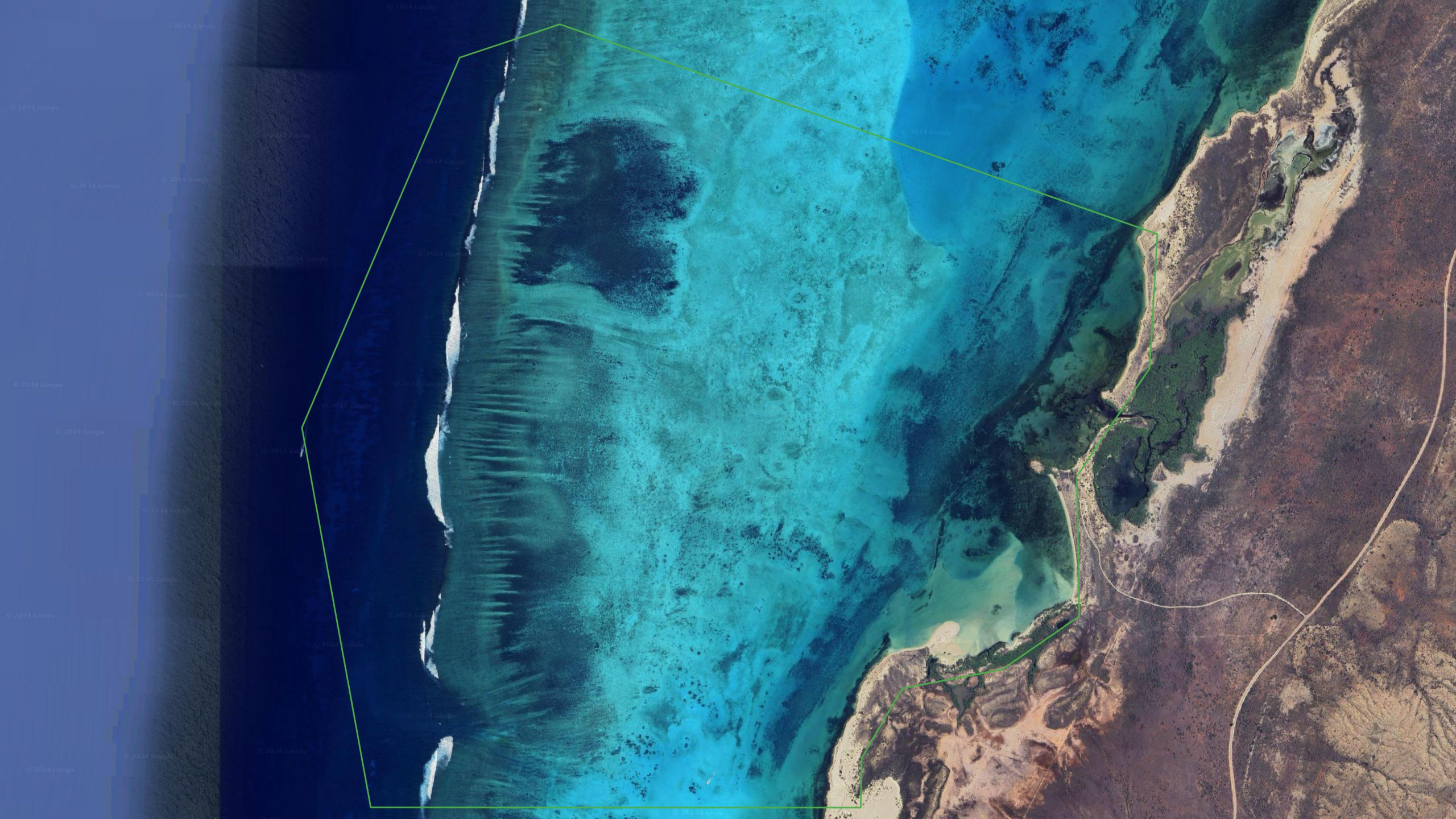
- Recent
- Starred
- Home
- Documents
- Downloads
- Pictures
- Videos
- Trash
- turtles
- MANGB\_20210425T1216
- work
- thailand\_2023\_11
- DR2023-01
- indonesia\_sulawesi\_2023\_06
- field\_trips
- 20230423T175208
- Music
- fielddata
- Drone\_footage
- NO2\_Field\_Data
- AS2022-01
- fielddata0
- 2023-11-18
- turtles
- + Other Locations

Name	Size	Modified
 100_0001_0001.JPG	9.3 MB	29 Oct 2020
 100_0001_0002.JPG	8.5 MB	29 Oct 2020
 100_0001_0003.JPG	9.1 MB	29 Oct 2020
 100_0001_0004.JPG	9.0 MB	29 Oct 2020
 100_0001_0005.JPG	9.3 MB	29 Oct 2020
 100_0001_0006.JPG	8.6 MB	29 Oct 2020
 100_0001_0007.JPG	9.1 MB	29 Oct 2020
 100_0001_0008.JPG	8.2 MB	29 Oct 2020
 100_0001_0009.JPG	8.4 MB	29 Oct 2020
 100_0001_0010.JPG	8.5 MB	29 Oct 2020
 100_0001_0011.JPG	8.4 MB	29 Oct 2020
 100_0001_0012.JPG	8.3 MB	29 Oct 2020
 100_0001_0013.JPG	9.2 MB	29 Oct 2020
 100_0001_0014.JPG	9.1 MB	29 Oct 2020
 100_0001_0015.JPG	8.9 MB	29 Oct 2020
 100_0001_0016.JPG	8.9 MB	29 Oct 2020
 100_0001_0017.JPG	8.9 MB	29 Oct 2020
 100_0001_0018.JPG	8.9 MB	29 Oct 2020
 100_0001_0019.JPG	8.8 MB	29 Oct 2020
 100_0001_0020.JPG	8.5 MB	29 Oct 2020
 100_0001_0021.JPG	8.3 MB	29 Oct 2020
 100_0001_0022.JPG	9.1 MB	29 Oct 2020
 100_0001_0023.JPG	8.8 MB	29 Oct 2020
 100_0001_0024.JPG	8.7 MB	29 Oct 2020
 100_0001_0025.JPG	8.9 MB	29 Oct 2020
 100_0001_0026.JPG	8.9 MB	29 Oct 2020





```
63 "ExposureMode": "Auto",
64 "WhiteBalance": "Manual",
65 "DigitalZoomRatio": "undef",
66 "FocalLengthIn35mmFormat": "24 mm",
67 "SceneCaptureType": "Standard",
68 "GainControl": "None",
69 "Contrast": "Normal",
70 "Saturation": "Normal",
71 "Sharpness": "Normal",
72 "SubjectDistanceRange": "Unknown",
73 "SerialNumber": "6e753047f0441c55b34839a8777c2765",
74 "GPSVersionID": "2.3.0.0",
75 "GPSLatitudeRef": "South",
76 "GPSLongitudeRef": "East",
77 "GPSAltitudeRef": "Above Sea Level",
78 "XPComment": "Type=N, Mode=P, DE=None",
79 "XPKeywords": "v01.09.1755;1.3.0;v1.0.0",
80 "Compression": "JPEG (old-style)",
81 "ThumbnailOffset": 10240,
82 "ThumbnailLength": 11219,
83 "About": "DJI Meta Data",
84 "Format": "image/jpeg",
85 "AbsoluteAltitude": "+59.57",
86 "RelativeAltitude": "+75.00",
87 "GPSLongitude": "113 deg 53' 33.06\" E",
88 "GimbalRollDegree": "+0.00",
89 "GimbalYawDegree": -76.70,
90 "GimbalPitchDegree": -85.00,
91 "FlightRollDegree": -4.70,
92 "FlightYawDegree": -72.80,
93 "FlightPitchDegree": -16.40,
94 "FlightXSpeed": -0.20,
95 "FlightYSpeed": -0.20,
96 "FlightZSpeed": "+0.00",
97 "CamReverse": 0,
98 "GimbalReverse": 0,
99 "SelfData": "Undefined",
100 "CalibratedFocalLength": 3666.666504,
```



```
src > exifdata.py > task_create_json
42 for item in config.geturl('imagesource').rglob('.'):
43     file_dep = list(item.glob(config.cfg['imagewild']))
44     if len(file_dep)>0:
45         target = item / 'exif.json'
46         if file_dep:
47             if which('exiftool'):
48                 yield {
49                     'name':str(target),
50                     'actions':[f'exiftool -ext JPG -ext jpg -json "{item.resolve()}" > "{target.resolve()}"'],
51                     'targets':[target],
52                     'uptodate':[True],
53                     #
54                     'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
55                     'clean':True,
56                 }
57             else:
58                 yield {
59                     'name':str(target),
60                     'actions':[f'exiftool -ext JPG -ext jpg -json "{item.resolve()}" > "{target.resolve()}"'],
61                     'targets':[target],
62                     'uptodate':[True],
63                     #
64                     'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
65                     'clean':True,
66                 }
67
68 @create_after(executed='create_json', target_regex='.*\sexif.json')
69 def task_process_json():
70     def process_json(dependencies, targets):
71         source_file =dependencies[0]
72         print('source file is: {0}'.format(source_file))
73         print('output dir is: {0}'.format(list(targets)[0]))
74         if os.stat(source_file).st_size > 0:
75             drone = pd.read_json(source_file)
76             if 'GPSLongitude' in drone.columns:
77                 def get_longitude(item):
78                     longitude =float(item[0]) + float(item[2][0:-1])/60 + float(item[3][0:-1])/3600
79                     return (longitude)
80             drone['Longitude'] = np.nan
81             drone['Latitude'] = np.nan
82             drone.loc[ ~drone['GPSLongitude'].isna(), 'Longitude']=drone.loc[ ~drone['GPSLongitude'].isna(), 'GPSLongitude'].str.split(' ',expand=True).
83             drone.loc[ ~drone['GPSLatitude'].isna(), 'Latitude']=drone.loc[ ~drone['GPSLatitude'].isna(), 'GPSLatitude'].str.split(' ',expand=True).appl
84             drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude'] =drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude']*-1
85             drone = drone[drone.columns[drone.columns.isin(wanted)]]
86             if 'SubSecDateTimeOriginal' in drone.columns:
87                 drone['TimeStamp'] = pd.to_datetime(drone.SubSecDateTimeOriginal,format='%Y:%m:%d %H:%M:%S.%f')
88             else:
89                 drone['TimeStamp'] = pd.to_datetime(drone.DateTimeOriginal,format='%Y:%m:%d %H:%M:%S')
90             sourcepath = config.CATALOG_DIR
91             drone['SourceRel'] =drone.SourceFile.apply(lambda x: os.path.relpath(x,start=sourcepath))
92             drone['Sequence'] =drone.SourceFile.str.extract('(.*Sequence>.+)(.jpg)')['Sequence']
```

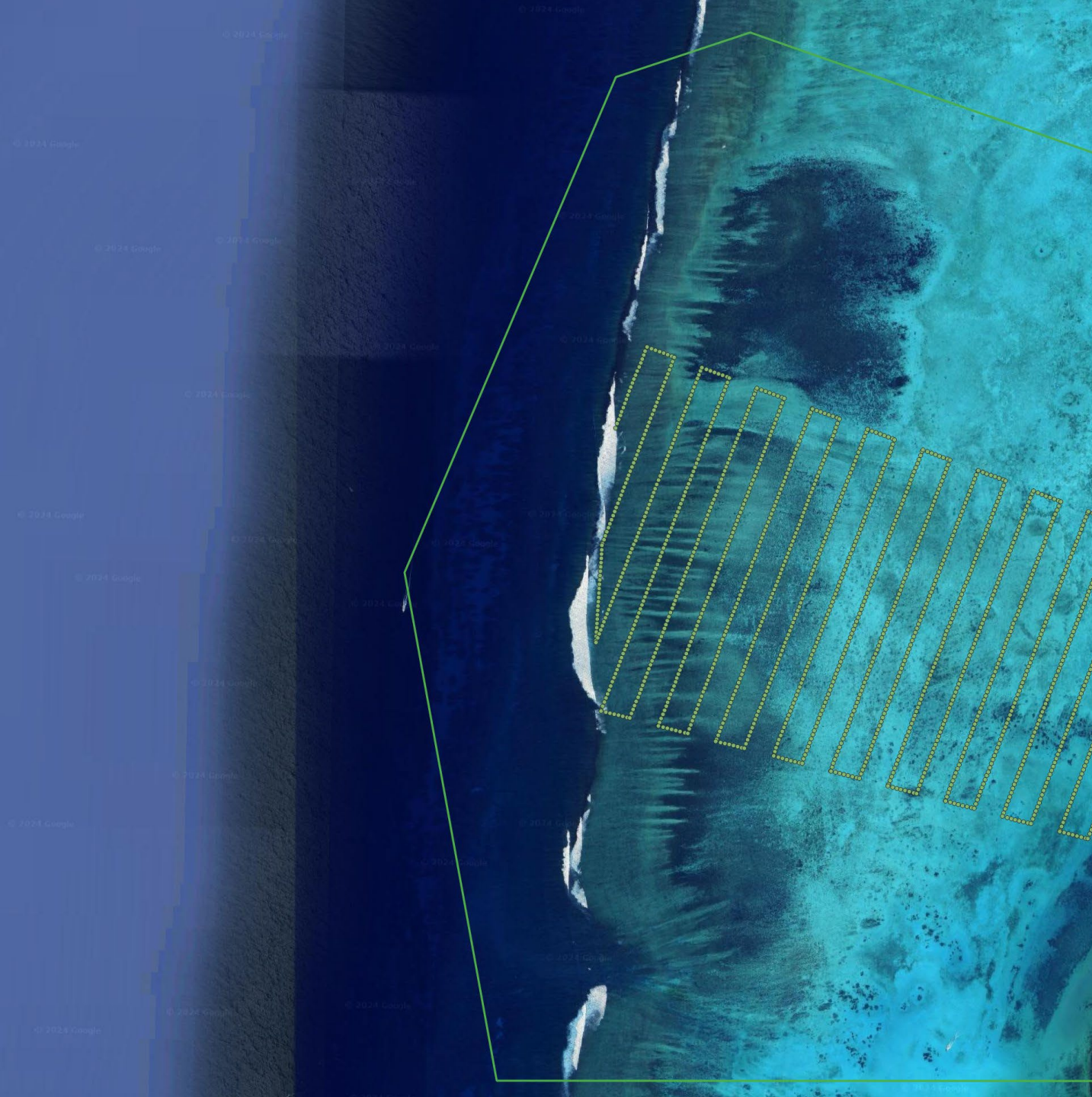
# Sanity Check data and assign to survey area













EXPLORER

- TURTLE
- \_\_pycache\_\_
- dodo.cpython-39.pyc
- .vscode
- launch.json
- binder
- scripts
- src
  - \_\_pycache\_\_
  - flask
  - templates
  - app.py
  - .doit.db.bak
  - .doit.db.dat
  - .doit.db.dir
  - build\_DJIMavic2.bat
  - build\_DJIP4Pro\_clean.sh
  - build\_DJIP4Pro.bat
  - build\_DJIP4Pro.sh
  - build\_DJIP4RTK.sh
  - build\_DJIP4RTKPro.bat
  - buildzarr.py
  - clean.py
  - combine\_surveys.py
  - concat\_labelme.py
  - concat\_turtles.py
  - config.py
  - count\_turtles.py
  - datacopy.py
  - DJIMavic2.py
  - DJIMavicPro2.py
  - DJIP4Pro.py
  - DJIP4RTK.py
  - dodo.py
  - drone.py
  - exifdata.py
  - getbase.py
  - initalize.py
  - labelme\_matchup.py
  - labelme.py
  - makegeo.py
  - makepdf.py
  - read\_rtk.py
- OUTLINE
- TIMELINE

jbs Aa ab \* No results



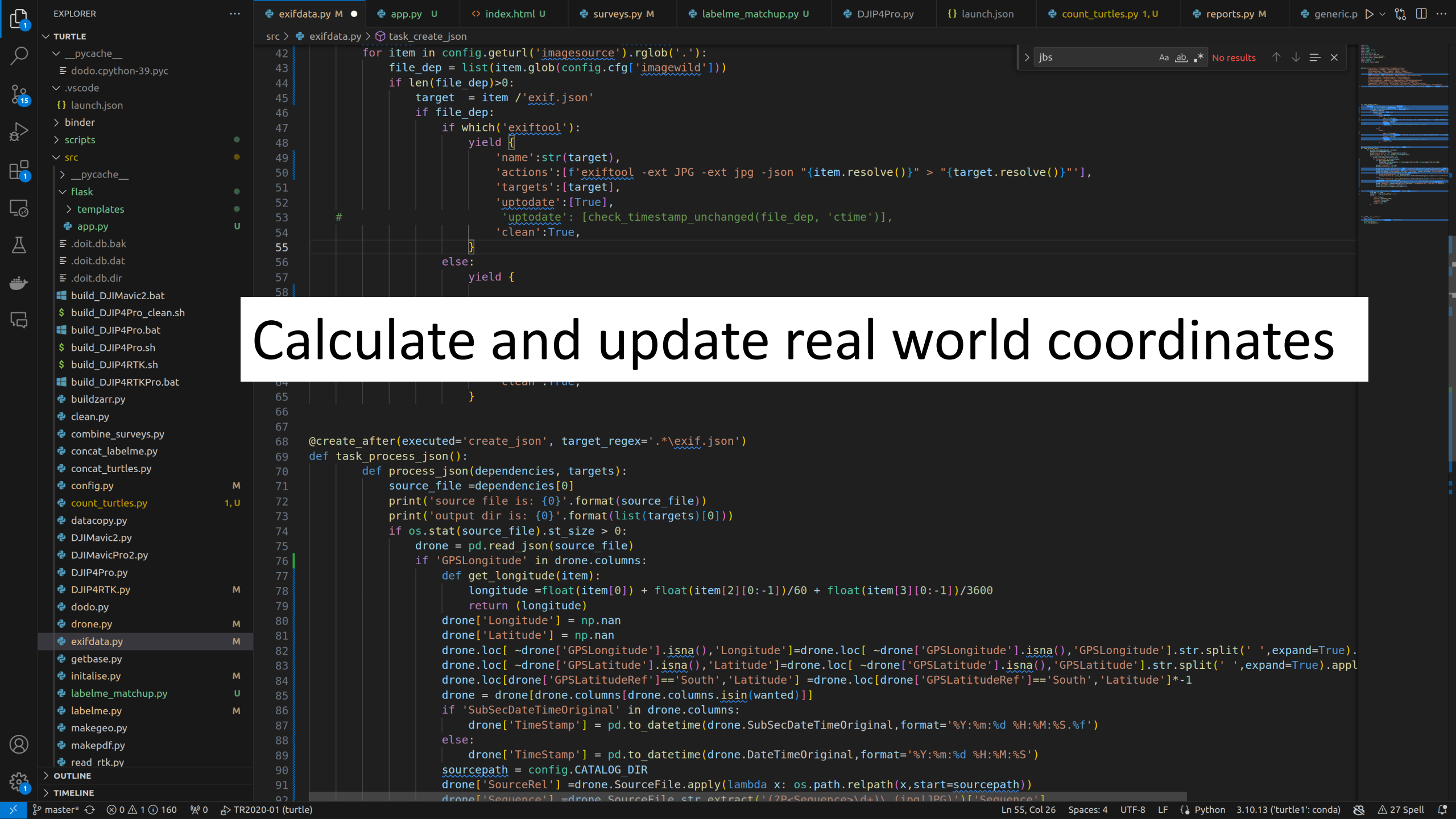


	DJIP4_RGB_AU_MANGB_20210426T085923_0031.JPG	8.7 MB
	DJIP4_RGB_AU_MANGB_20210426T085920_0030.JPG	8.7 MB
	DJIP4_RGB_AU_MANGB_20210426T085917_0029.JPG	8.6 MB
	DJIP4_RGB_AU_MANGB_20210426T085914_0028.JPG	8.5 MB
	DJIP4_RGB_AU_MANGB_20210426T085912_0027.JPG	8.5 MB
	DJIP4_RGB_AU_MANGB_20210426T085909_0026.JPG	8.7 MB
	DJIP4_RGB_AU_MANGB_20210426T085907_0025.JPG	8.7 MB
	DJIP4_RGB_AU_MANGB_20210426T085904_0024.JPG	8.7 MB
	DJIP4_RGB_AU_MANGB_20210426T085902_0023.JPG	8.5 MB
	DJIP4_RGB_AU_MANGB_20210426T085859_0022.JPG	8.5 MB
	DJIP4_RGB_AU_MANGB_20210426T085858_0021.JPG	8.6 MB
	DJIP4_RGB_AU_MANGB_20210426T085855_0020.JPG	8.6 MB

# Calculate and update real world coordinates

```
src > exifdata.py > task_create_json
42     for item in config.geturl('imagesource').rglob('.'):
43         file_dep = list(item.glob(config.cfg['imagewild']))
44         if len(file_dep)>0:
45             target = item / 'exif.json'
46             if file_dep:
47                 if which('exiftool'):
48                     yield {
49                         'name':str(target),
50                         'actions':[f'exiftool -ext JPG -ext jpg -json "{item.resolve()}" > "{target.resolve()}"'],
51                         'targets':[target],
52                         'uptodate':[True],
53                         'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
54                         'clean':True,
55                     }
56             else:
57                 yield {
58
64
65     }
66
67
68 @create_after(executed='create_json', target_regex='.*\sexif.json')
69 def task_process_json():
70     def process_json(dependencies, targets):
71         source_file =dependencies[0]
72         print('source file is: {0}'.format(source_file))
73         print('output dir is: {0}'.format(list(targets)[0]))
74         if os.stat(source_file).st_size > 0:
75             drone = pd.read_json(source_file)
76             if 'GPSLongitude' in drone.columns:
77                 def get_longitude(item):
78                     longitude =float(item[0]) + float(item[2][0:-1])/60 + float(item[3][0:-1])/3600
79                     return (longitude)
80             drone['Longitude'] = np.nan
81             drone['Latitude'] = np.nan
82             drone.loc[ ~drone['GPSLongitude'].isna(), 'Longitude']=drone.loc[ ~drone['GPSLongitude'].isna(), 'GPSLongitude'].str.split(' ',expand=True).
83             drone.loc[ ~drone['GPSLatitude'].isna(), 'Latitude']=drone.loc[ ~drone['GPSLatitude'].isna(), 'GPSLatitude'].str.split(' ',expand=True).appl
84             drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude'] =drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude']* -1
85             drone = drone[drone.columns[drone.columns.isin(wanted)]]
86             if 'SubSecDateTimeOriginal' in drone.columns:
87                 drone['TimeStamp'] = pd.to_datetime(drone.SubSecDateTmeOriginal,format='%Y:%m:%d %H:%M:%S.%f')
88             else:
89                 drone['TimeStamp'] = pd.to_datetime(drone.DateTmeOriginal,format='%Y:%m:%d %H:%M:%S')
90             sourcepath = config.CATALOG_DIR
91             drone['SourceRel'] =drone.SourceFile.apply(lambda x: os.path.relpath(x,start=sourcepath))
92             drone['Sequence'] =drone.SourceFile.str.extract('(?!Sequence>.+)(.+(?!\.jpg))$')
```

jbs Aa ab \* No results





- Open
- Open Dir
- Next Image
- Prev Image
- Save
- Delete File
- Create Polygons
- Edit Polygons
- Duplicate Polygons
- Copy Polygons
- Paste Polygons
- Delete Polygons
- Undo
- Brightness Contrast
- 327 %
- 2 of 36
- Fit Width



Flags

---

Label List

- done
- turtle\_surface
- ray
- turtle\_jbs

---

Polygon Labels

- turtle\_jbs

---

File List

Search filename

- rveys/AU/MANCB/MANCB\_20210426T0958/DJIP4\_RGB\_AU\_MANCB\_20210426T095735\_1165.JPG
- rveys/AU/MANCB/MANCB\_20210426T095737\_1166.JPG
- rveys/AU/MANCB/MANCB\_20210426T095740\_1167.JPG
- rveys/AU/MANCB/MANCB\_20210426T095742\_1168.JPG
- rveys/AU/MANCB/MANCB\_20210426T095745\_1169.JPG
- rveys/AU/MANCB/MANCB\_20210426T095747\_1170.JPG
- rveys/AU/MANCB/MANCB\_20210426T095750\_1171.JPG
- rveys/AU/MANCB/MANCB\_20210426T095752\_1172.JPG
- rveys/AU/MANCB/MANCB\_20210426T095755\_1173.JPG
- rveys/AU/MANCB/MANCB\_20210426T095757\_1174.JPG
- rveys/AU/MANCB/MANCB\_20210426T095800\_1175.JPG
- rveys/AU/MANCB/MANCB\_20210426T095802\_1176.JPG
- rveys/AU/MANCB/MANCB\_20210426T095805\_1177.JPG
- rveys/AU/MANCB/MANCB\_20210426T095807\_1178.JPG
- rveys/AU/MANCB/MANCB\_20210426T095810\_1179.JPG
- rveys/AU/MANCB/MANCB\_20210426T095812\_1180.JPG
- rveys/AU/MANCB/MANCB\_20210426T095815\_1181.JPG
- rveys/AU/MANCB/MANCB\_20210426T095817\_1182.JPG
- rveys/AU/MANCB/MANCB\_20210426T095820\_1183.JPG
- rveys/AU/MANCB/MANCB\_20210426T095823\_1184.JPG
- rveys/AU/MANCB/MANCB\_20210426T095825\_1185.JPG
- rveys/AU/MANCB/MANCB\_20210426T095830\_1187.JPG
- rveys/AU/MANCB/MANCB\_20210426T095833\_1188.JPG
- rveys/AU/MANCB/MANCB\_20210426T095835\_1189.JPG
- rveys/AU/MANCB/MANCB\_20210426T095838\_1190.JPG
- rveys/AU/MANCB/MANCB\_20210426T095840\_1191.JPG
- rveys/AU/MANCB/MANCB\_20210426T095843\_1192.JPG
- rveys/AU/MANCB/MANCB\_20210426T095845\_1193.JPG

- Open
- Open Dir
- Next Image
- Prev Image
- Save
- Delete File
- Create Polygons
- Edit Polygons
- Duplicate Polygons
- Copy Polygons
- Paste Polygons
- Delete Polygons
- Undo
- Brightness Contrast
- 327 %
- 2 of 36
- Fit Width



Flags

---

Label List

- done ●
- turtle\_surface ●**
- ray ●
- turtle\_jbs ●

---

Polygon Labels

- ✓ done ●
- ✓ turtle\_jbs ●**

---

File List

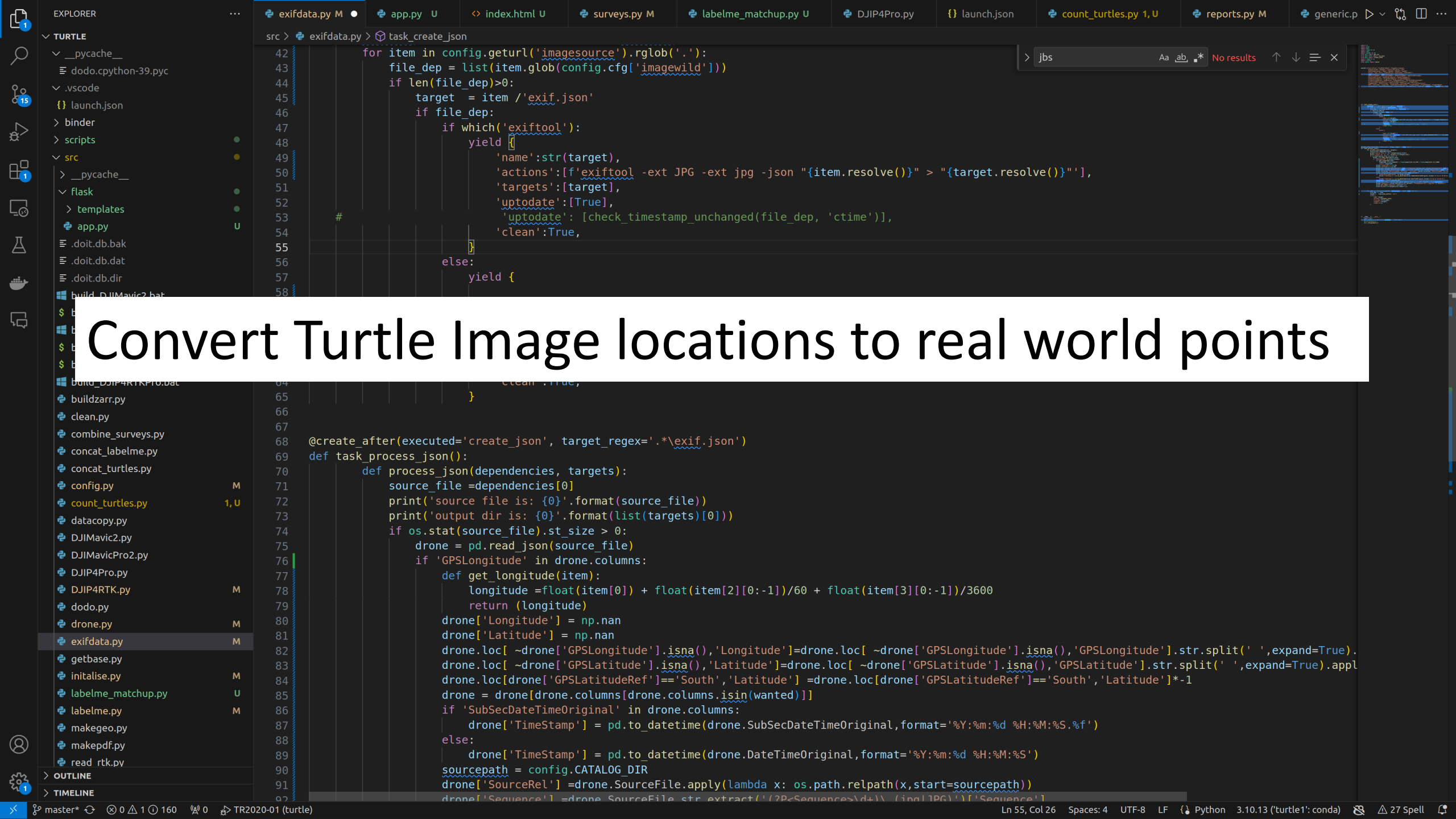
Search Filename

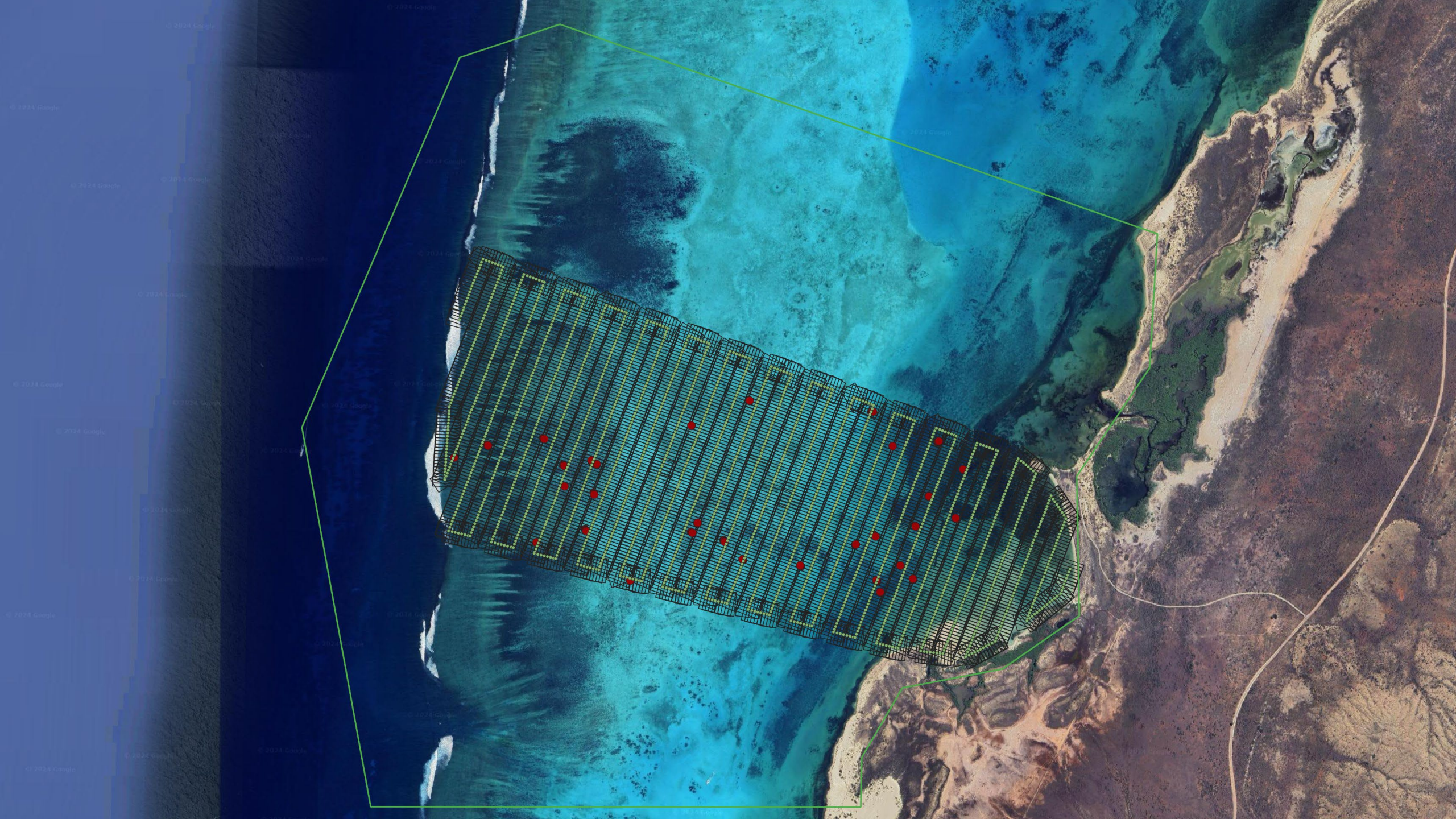
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095735\_1165.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095737\_1166.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095740\_1167.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095742\_1168.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095745\_1169.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095747\_1170.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095750\_1171.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095752\_1172.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095755\_1173.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095757\_1174.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095800\_1175.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095802\_1176.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095805\_1177.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095807\_1178.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095810\_1179.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095812\_1180.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095815\_1181.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095817\_1182.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095820\_1183.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095823\_1184.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095825\_1185.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095828\_1186.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095830\_1187.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095833\_1188.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095835\_1189.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095838\_1190.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095840\_1191.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095843\_1192.JPG
- rveys/AU/MANGB/MANGB\_20210426T0858/DJIP4\_RGB\_AU\_MANGB\_20210426T095845\_1193.JPG

# Convert Turtle Image locations to real world points

```
src > exifdata.py > task_create_json
42 for item in config.geturl('imagesource').rglob('.'):
43     file_dep = list(item.glob(config.cfg['imagewild']))
44     if len(file_dep)>0:
45         target = item / 'exif.json'
46         if file_dep:
47             if which('exiftool'):
48                 yield {
49                     'name':str(target),
50                     'actions':[f'exiftool -ext JPG -ext jpg -json "{item.resolve()}" > "{target.resolve()}"'],
51                     'targets':[target],
52                     'uptodate':[True],
53                     'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
54                     'clean':True,
55                 }
56             else:
57                 yield {
58
59
60
61
62
63
64
65     }
66
67
68 @create_after(executed='create_json', target_regex='.*\exif.json')
69 def task_process_json():
70     def process_json(dependencies, targets):
71         source_file =dependencies[0]
72         print('source file is: {0}'.format(source_file))
73         print('output dir is: {0}'.format(list(targets)[0]))
74         if os.stat(source_file).st_size > 0:
75             drone = pd.read_json(source_file)
76             if 'GPSLongitude' in drone.columns:
77                 def get_longitude(item):
78                     longitude =float(item[0]) + float(item[2][0:-1])/60 + float(item[3][0:-1])/3600
79                     return (longitude)
80                 drone['Longitude'] = np.nan
81                 drone['Latitude'] = np.nan
82                 drone.loc[ ~drone['GPSLongitude'].isna(), 'Longitude']=drone.loc[ ~drone['GPSLongitude'].isna(), 'GPSLongitude'].str.split(' ',expand=True).
83                 drone.loc[ ~drone['GPSLatitude'].isna(), 'Latitude']=drone.loc[ ~drone['GPSLatitude'].isna(), 'GPSLatitude'].str.split(' ',expand=True).appl
84                 drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude'] =drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude']* -1
85                 drone = drone[drone.columns[drone.columns.isin(wanted)]]
86                 if 'SubSecDateTimeOriginal' in drone.columns:
87                     drone['TimeStamp'] = pd.to_datetime(drone.SubSecDateTimeOriginal,format='%Y:%m:%d %H:%M:%S.%f')
88                 else:
89                     drone['TimeStamp'] = pd.to_datetime(drone.DateTimeOriginal,format='%Y:%m:%d %H:%M:%S')
90                 sourcepath = config.CATALOG_DIR
91                 drone['SourceRel'] =drone.SourceFile.apply(lambda x: os.path.relpath(x,start=sourcepath))
92                 drone['Sequence'] =drone.SourceFile.str.extract('(.*Sequence>)\d+\.(jpg|JPG)')['Sequence']
```

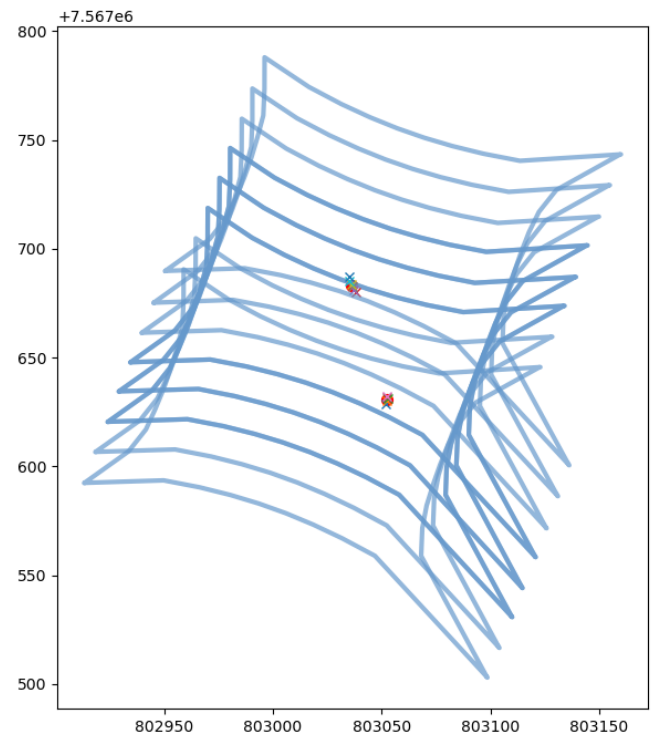
jbs Aa ab\_\* No results





# Group turtle points together

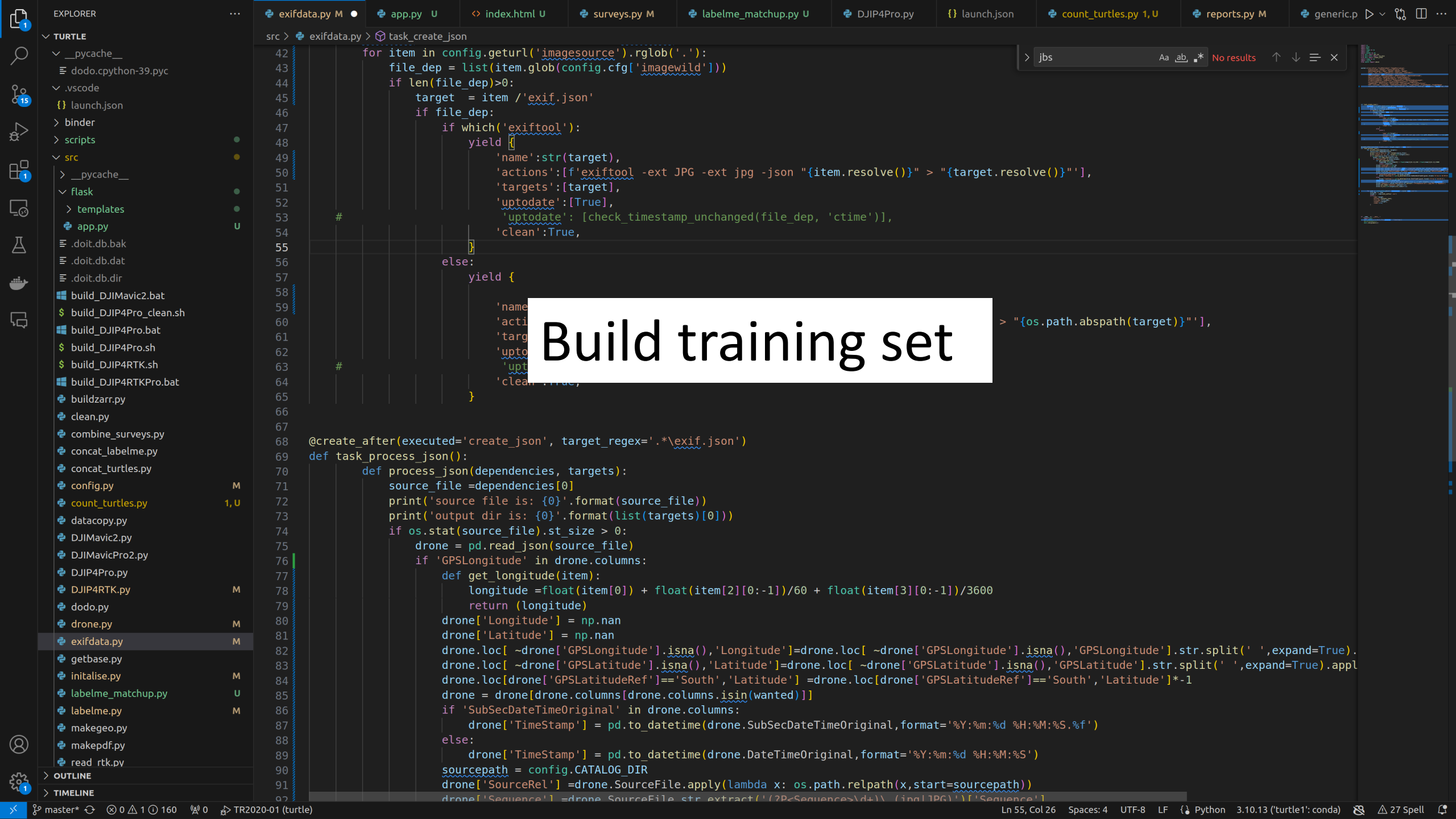
```
src > exifdata.py > task_create_json
42     for item in config.geturl('imagesource').rglob('.'):
43         file_dep = list(item.glob(config.cfg['imagewild']))
44         if len(file_dep)>0:
45             actions = [exiftool -ext jpg -ext json {item.resolve()} > "{target.resolve()}"],
46             'targets':[target],
47             'uptodate':[True],
48             'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
49             'clean':True,
50         else:
51             yield {
52                 'name':str(target),+
53                 'actions':[f"{exifpath}" -exiftool {target.resolve()} > "{target.resolve()}"],
54                 'targets':[target],
55                 'uptodate':[True],
56                 'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
57                 'clean':True,
58             }
59
60 @create_after(executed='create_json', target_regex='.*\.(json|jpg)$')
61 def task_process_json():
62     def process_json(dependencies, targets):
63         source_file =dependencies[0]
64         print('source file is: {0}'.format(source_file))
65         print('output dir is: {0}'.format(list(targets)))
66         if os.stat(source_file).st_size > 0:
67             drone = pd.read_json(source_file)
68             if 'GPSLongitude' in drone.columns:
69                 def get_longitude(item):
70                     longitude =float(item[0]) + float(item[1])
71                     return (longitude)
72                 drone['Longitude'] = np.nan
73                 drone['Latitude'] = np.nan
74                 drone.loc[~drone['GPSLongitude'].isna(), 'Longitude'] = drone['GPSLongitude'].apply(get_longitude)
75                 drone.loc[~drone['GPSLatitude'].isna(), 'Latitude'] = drone['GPSLatitude'].apply(get_latitude)
76                 drone = drone[drone.columns[drone.columns!='GPSLongitudeRef']]
77                 if 'SubSecDateOriginal' in drone.columns:
78                     drone['TimeStamp'] = pd.to_datetime(drone['SubSecDateOriginal'])
79                 else:
80                     drone['TimeStamp'] = pd.to_datetime(drone['SubSecDateOriginal'])
81                 sourcepath = config.CATALOG_DIR
82                 drone['SourceRel'] =drone.SourceFile.apply(lambda x: os.path.relpath(x,start=sourcepath))
83                 drone['Sequence'] =drone.SourceFile.str.extract('(.*Sequence>)\d+\.(jpg|png)')['Sequence']
```



This image shows a screenshot of a code editor with a terminal window on the left and a plot on the right. The terminal window displays the execution of a Python script named 'exifdata.py', which processes drone data. The plot shows the resulting drone trajectories, which are grouped together. The code in the terminal includes functions for processing JSON files and extracting drone data, such as longitude and latitude. The plot visualizes the data extracted from these files, showing multiple overlapping paths in a specific geographic area.







Build training set

```
src > exifdata.py > task_create_json
42 for item in config.geturl('imagesource').rglob('.'):
43     file_dep = list(item.glob(config.cfg['imagewild']))
44     if len(file_dep)>0:
45         target = item / 'exif.json'
46         if file_dep:
47             if which('exiftool'):
48                 yield {
49                     'name':str(target),
50                     'actions':[f'exiftool -ext JPG -ext jpg -json "{item.resolve()}" > "{target.resolve()}"'],
51                     'targets':[target],
52                     'uptodate':[True],
53                     'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
54                     'clean':True,
55                 }
56             else:
57                 yield {
58                     'name':str(target),
59                     'actions':[f'exiftool -ext JPG -ext jpg -json "{item.resolve()}" > "{target.resolve()}"'],
60                     'targets':[target],
61                     'uptodate':[True],
62                     'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
63                     'clean':True,
64                 }
65         else:
66             yield {
67                 'name':str(target),
68                 'actions':[f'exiftool -ext JPG -ext jpg -json "{item.resolve()}" > "{target.resolve()}"'],
69                 'targets':[target],
70                 'uptodate':[True],
71                 'uptodate':[check_timestamp_unchanged(file_dep, 'ctime')],
72                 'clean':True,
73             }
74
75 @create_after(executed='create_json', target_regex='.*\sexif.json')
76 def task_process_json():
77     def process_json(dependencies, targets):
78         source_file =dependencies[0]
79         print('source file is: {0}'.format(source_file))
80         print('output dir is: {0}'.format(list(targets)[0]))
81         if os.stat(source_file).st_size > 0:
82             drone = pd.read_json(source_file)
83             if 'GPSLongitude' in drone.columns:
84                 def get_longitude(item):
85                     longitude =float(item[0]) + float(item[2][0:-1])/60 + float(item[3][0:-1])/3600
86                     return (longitude)
87                 drone['Longitude'] = np.nan
88                 drone['Latitude'] = np.nan
89                 drone.loc[ ~drone['GPSLongitude'].isna(), 'Longitude']=drone.loc[ ~drone['GPSLongitude'].isna(), 'GPSLongitude'].str.split(' ',expand=True).
90                 drone.loc[ ~drone['GPSLatitude'].isna(), 'Latitude']=drone.loc[ ~drone['GPSLatitude'].isna(), 'GPSLatitude'].str.split(' ',expand=True).appl
91                 drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude'] =drone.loc[drone['GPSLatitudeRef']=='South', 'Latitude']*-1
92                 drone = drone[drone.columns[drone.columns.isin(wanted)]]
93                 if 'SubSecDateTimeOriginal' in drone.columns:
94                     drone['TimeStamp'] = pd.to_datetime(drone.SubSecDateTimeOriginal,format='%Y:%m:%d %H:%M:%S.%f')
95                 else:
96                     drone['TimeStamp'] = pd.to_datetime(drone.DateTimeOriginal,format='%Y:%m:%d %H:%M:%S')
97                 sourcepath = config.CATALOG_DIR
98                 drone['SourceRel'] =drone.SourceFile.apply(lambda x: os.path.relpath(x,start=sourcepath))
99                 drone['Sequence'] =drone.SourceFile.str.extract('(.*Sequence>+)(\.(jpg|JPG))$')
100
```

jbs Aa ab\_\* No results

EXPLORER

- TURTLE
  - \_\_pycache\_\_
    - dodo.cpython-39.pyc
  - .vscode
    - launch.json
  - binder
  - scripts
  - src
    - \_\_pycache\_\_
    - flask
    - templates
    - app.py
    - .doit.db.bak
    - .doit.db.dat
    - .doit.db.dir
    - build\_DJIMavic2.bat
    - build\_DJIP4Pro\_clean.sh
    - build\_DJIP4Pro.bat
    - build\_DJIP4Pro.sh
    - build\_DJIP4RTK.sh
    - build\_DJIP4RTKPro.bat
    - buildzarr.py
    - clean.py
    - combine\_surveys.py
    - concat\_labelme.py
    - concat\_turtles.py
    - config.py
    - count\_turtles.py
    - datacopy.py
    - DJIMavic2.py
    - DJIMavicPro2.py
    - DJIP4Pro.py
    - DJIP4RTK.py
    - dodo.py
    - drone.py
    - exifdata.py
    - getbase.py
    - initalize.py
    - labelme\_matchup.py
    - labelme.py
    - makegeo.py
    - makepdf.py
    - read\_rtk.py
  - OUTLINE
  - TIMELINE

# Thank you

Plus applause for Ocean Atlan (turtle finder!)



<https://github.com/NickMortimer/turtle>

