📚 GIS Labs 5–7 Summary with Scripts

🧪 Lab 5: Images, Exif Data & GPS Info

Goal: Extract GPS info from images and convert it to shapefiles.

📜 Topics:

- Using Python `Pillow` to extract EXIF metadata
- Parsing GPS coordinates
- Converting degrees-minutes-seconds (DMS) to decimal
- Lat & Long reference handling

🧰 Steps & Scripts:

1. **Import Modules & Read Image Paths**

```
import os
from PIL import Image
from PIL.ExifTags import TAGS, GPSTAGS

img_folder = r"F:\GIS\Application\Images"
img_contents = os.listdir(img_folder)
```

2. **Get Full Path for Each Image**

```
for image in img_contents:
    full_path = os.path.join(img_folder, image)
```

3. **Read EXIF Metadata**

```
pillow_img = Image.open(full_path)
exif = {
    TAGS.get(k): v
    for k, v in pillow_img._getexif().items()
    if k in TAGS
}
```

4. **Extract GPS Info**

```
gps_info = {}
if "GPSInfo" in exif:
    for key in exif["GPSInfo"].keys():
```

```
        decode = GPSTAGS.get(key)
        gps_info[decode] = exif["GPSInfo"][key]
```

5. **Convert DMS to Decimal Degrees**

```
def convert_to_degrees(value):
    d = value[0][0] / value[0][1]
    m = value[1][0] / value[1][1]
    s = value[2][0] / value[2][1]
    return d + (m / 60.0) + (s / 3600.0)

lat = convert_to_degrees(gps_info["GPSLatitude"])
lon = convert_to_degrees(gps_info["GPSLongitude"])

if gps_info["GPSLatitudeRef"] == "S":
    lat = -lat
if gps_info["GPSLongitudeRef"] == "W":
    lon = -lon
```

ALL Script

```
import os
from PIL import Image
from PIL.ExifTags import TAGS, GPSTAGS

def convert_to_degrees(value):
    d = value[0][0] / value[0][1]
    m = value[1][0] / value[1][1]
    s = value[2][0] / value[2][1]
    return d + (m / 60.0) + (s / 3600.0)

img_folder = r"F:\GIS\Application\Images"
img_contents = os.listdir(img_folder)

for image in img_contents:
    full_path = os.path.join(img_folder, image)
    pillow_img = Image.open(full_path)
    exif_data = pillow_img._getexif()

    if not exif_data:
        continue

    exif = {
        TAGS.get(k): v
        for k, v in exif_data.items()
        if k in TAGS
    }

    gps_info = {}
    if "GPSInfo" in exif:
        for key in exif["GPSInfo"].keys():
            decode = GPSTAGS.get(key)
            gps_info[decode] = exif["GPSInfo"][key]

        try:
            lat = convert_to_degrees(gps_info["GPSLatitude"])
```

```
            lon = convert_to_degrees(gps_info["GPSLongitude"])

            if gps_info["GPSLatitudeRef"] == "S":
                lat = -lat
            if gps_info["GPSLongitudeRef"] == "W":
                lon = -lon

            print(f"{image} → Latitude: {lat}, Longitude: {lon}")
        except Exception as e:
            print(f"Skipping {image}: Missing or invalid GPS info")
```

🌍 Lab 6: Geotagged SHP, Add Fields & Records

Goal: Create shapefile from image GPS data and add fields.

🧰 Steps & Scripts:

1. **Create a List of Coordinates**

```
shp_list = []
shp_list.append([lon, lat])
```

2. **Create Point Objects**

```
import arcpy

pt = arcpy.Point()
ptGeoms = []
spatial_ref = arcpy.SpatialReference(4326)

for p in shp_list:
    pt.X = p[0]
    pt.Y = p[1]
    ptGeoms.append(arcpy.PointGeometry(pt, spatial_ref))
```

3. **Generate Shapefile**

```
out_shapefile = r"D:\GIS\Application\Images\out_shape_GeoImages.shp"
arcpy.CopyFeatures_management(ptGeoms, out_shapefile)
```

4. **Add XY Coordinates Fields**

```
arcpy.management.AddXY(out_shapefile)
```

5. **Add Custom Fields (Path & Timestamp)**

```
arcpy.management.AddField(out_shapefile, "ImagePath", "TEXT")
arcpy.management.AddField(out_shapefile, "Timestamp", "TEXT")
```

6. **Update Records using Cursor**

```
with arcpy.da.UpdateCursor(out_shapefile, ["ImagePath", "Timestamp"]) as cursor:
    for i, row in enumerate(cursor):
        row[0] = shp_list[i][2]  # Path
        row[1] = shp_list[i][3]  # Timestamp
        cursor.updateRow(row)
```

ALL Script

```
import arcpy

# This should be filled from Lab 5 output
shp_list = [
    [-0.074575, 51.504105, r"D:\GIS\Application\Images\london.jpg", "2018:08:22 13:13:41"],
    [31.2357, 30.0444, r"D:\GIS\Application\Images\cairo.jpg", "2020:02:01 12:00:00"]
]

# Create PointGeometry objects
pt = arcpy.Point()
ptGeoms = []
spatial_ref = arcpy.SpatialReference(4326)

for p in shp_list:
    pt.X = p[0]
    pt.Y = p[1]
    ptGeoms.append(arcpy.PointGeometry(pt, spatial_ref))

# Create shapefile
out_shapefile = r"D:\GIS\Application\Images\geo_images.shp"
arcpy.CopyFeatures_management(ptGeoms, out_shapefile)

# Add XY fields
arcpy.management.AddXY(out_shapefile)

# Add Path and Timestamp fields
arcpy.management.AddField(out_shapefile, "ImagePath", "TEXT")
arcpy.management.AddField(out_shapefile, "Timestamp", "TEXT")

# Update fields with values
with arcpy.da.UpdateCursor(out_shapefile, ["ImagePath", "Timestamp"]) as cursor:
    for i, row in enumerate(cursor):
        row[0] = shp_list[i][2]  # Path
        row[1] = shp_list[i][3]  # Timestamp
        cursor.updateRow(row)
```

🩹 Lab 7: Spatial Join, Symbology, Buffer & Clip, Reports & Graphs

---

📌 Spatial Join in ArcPy

```
target_layer = "countries"
join_layer = "cities"
output = r"C:\GIS\output\joined_countries.shp"

arcpy.analysis.SpatialJoin(
    target_layer, join_layer, output,
    join_type="KEEP_COMMON", match_option="INTERSECT"
)
```

🎨 Symbology in ArcMap

Manual steps:

- Right-click layer → Properties → Symbology tab.
- Select “Quantities” → Field: `Join_Count`.
- Apply color ramp and classes.

📏 Buffer in ArcPy

```
arcpy.Buffer_analysis(
    in_features="cities",
    out_feature_class=r"C:\GIS\output\cities_buffer.shp",
    buffer_distance_or_field="5000 Meters"
)
```

✂️ Clip in ArcPy

```
arcpy.Clip_analysis(
    in_features="world_cities",
    clip_features="egypt_boundary",
    out_feature_class=r"C:\GIS\output\egypt_cities.shp"
)
```

All Script

```
import arcpy

# Set workspace
arcpy.env.workspace = r"C:\GIS"

# ---------- Spatial Join ----------
target_layer = "countries.shp"
join_layer = "cities.shp"
output_join = "joined_countries.shp"
```

```
arcpy.analysis.SpatialJoin(
    target_layer, join_layer, output_join,
    join_type="KEEP_COMMON", match_option="INTERSECT"
)

# ---------- Buffer ----------
buffer_output = "cities_buffer.shp"
arcpy.Buffer_analysis(
    in_features="cities.shp",
    out_feature_class=buffer_output,
    buffer_distance_or_field="10000 Meters"
)

# ---------- Clip ----------
clip_output = "egypt_cities.shp"
arcpy.Clip_analysis(
    in_features="world_cities.shp",
    clip_features="egypt_boundary.shp",
    out_feature_class=clip_output
)
```

📄 Reports (Manual)

1. Right-click layer → Create Report.
2. Select fields (e.g., Continent, Population).
3. Apply grouping and sorting.
4. Export or print.

📊 Graphs in ArcMap

1. View → Graphs → Create.
2. Choose chart type (e.g., Pie).
3. Select field (e.g., POP_EST).
4. Customize and display.

Would you like me to export this as a **PDF** or **Word** file?