Visualization

December 5, 2024

1 Data analyzis of the MGZ mission

This Jupiter Notebook makes basic analyzis and refinement on the measured datas of our CanSat when our data sources are updated and our notebook is restarted. Dou to the fact that our mission is a one time flight, this technique is a bit overcomplicated and unnescessarily robust. However, this method of data analyzis makes reusability a possibility and testing less time consuming. Our scripts are designed to be easily readable and usable on other windows computers as well. This makes collaboration accessible and less problematic. These are crucial aspects on an offical mission, which we aim to replicate to the best of our ability.

For more information, read the README.md document.

1.1 Setup and sorting

Before we start the analyzis we have to include the necessary modules and libraries. We will try to make sure that our code is bulletproof aganist all types of errors and easily fixable by using error handling. Let's organize our measurements using OOP. In addition, we are able set test_data_mode = True which generates test values for the notebook. If an organized database is avalible, we are able to set only_analizys_mode = True. This drastically reduces execution time.

[ ]:
```
try:
    import program_files.cansattools as cansattools
    logger = cansattools.logger_creator("jupyter_notebook")
except ImportError:
    logger = None
    print("Error importing cansattools module. Logging is disabled.")

try:
    from mpl_toolkits import mplot3d
    %matplotlib inline
    import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.cm as cm
    import os
    import program_files.classes as classes
    import sqlite3
    from sqlite3 import Error
except ImportError as e:
    logger.error(f"Error importing modules: {e}")
```

```
txt_name = "raw_data.txt"
database_name = "datas/raw_data.db"
test_data_mode = True
only_analyzis_mode = False

#create or replace database
cansattools.create_db(database_name, replace_mode=True)

# Store the data in classes
bmp280: list[classes.BMP280] = []
dht11: list[classes.DHT11] = []
gpses: list[classes.GPS] = []
mpu6050: list[classes.MPU6050] = []

if not only_analyzis_mode:
    if test_data_mode:
        bmp_current = classes.BMP280(0, 30, 4000.0, 100.0)
        dht_current = classes.DHT11(0, 50.0)
        for i in range(0, 2000):
            bmp280.append(bmp_current)
            dht11.append(dht_current)
            bmp_current.time = cansattools.test_data_generator(bmp_current.
 ↪time, 3000000, 0, 20, True)
            bmp_current.temperature = cansattools.
 ↪test_data_generator(bmp_current.temperature, 100, -100, 5)
            bmp_current.pressure = cansattools.test_data_generator(bmp_current.
 ↪pressure, 100000, 0, 20)
            bmp_current.height = cansattools.test_data_generator(bmp_current.
 ↪height, 1000, 20, 20)
            dht_current.time = cansattools.test_data_generator(dht_current.
 ↪time, 3000000, 0, 5, True)
            dht_current.humidity = cansattools.test_data_generator(dht_current.
 ↪humidity, 100, 0, 5)
            bmp_current = classes.BMP280(bmp_current.time, bmp_current.
 ↪temperature, bmp_current.pressure, bmp_current.height)
            dht_current = classes.DHT11(dht_current.time, dht_current.humidity)

        gps_current = classes.GPS(0, 0.0, 0.0, 0.0)
        for i in range(0, 80):
            gpses.append(gps_current)
            gps_current.time = cansattools.test_data_generator(gps_current.
 ↪time, 3000000, 0, 400, True)
            gps_current.latitude = cansattools.test_data_generator(gps_current.
 ↪latitude, 90, -90, 20)
```

```
            gps_current.longitude = cansattools.test_data_generator(gps_current.
 ↪longitude, 180, -180, 20)
            gps_current.altitude = cansattools.test_data_generator(gps_current.
 ↪altitude, 1000, 20, 20)
            gps_current = classes.GPS(gps_current.time, gps_current.latitude,␣
 ↪gps_current.longitude, gps_current.altitude)

    else:
        # Open the txt file and read the data
        try:
            with open(f"datas/{txt_name}", "r") as file:
                data = file.readlines()
        except FileNotFoundError:
            try:
                with open(f"{txt_name}", "r") as file:
                    data = file.readlines()
            except FileNotFoundError:
                logger.error(f"File {txt_name} not found")

        for line in data:
            try:
                if "BMP280" in line:
                    bmp280.append(classes.BMP280(line.split()[1:]))
                elif "DHT11" in line:
                    dht11.append(classes.DHT11(line.split()[1:]))
                elif "GPS" in line:
                    gpses.append(classes.GPS(line.split()[1:]))
                elif "MPU6050" in line:
                    mpu6050.append(classes.MPU6050(line.split()[1:]))
            except Exception as e:
                logger.error(f"Error inserting data into the classes: {e}")

        # Insert the data into the SQLite database
        cansattools.txt_to_db(data, database_name)

        # Clear the data
        data = None
else: #it generates tuples instead of objects
    # Read the data from the SQLite database
    bmp280_from_db = classes.BMP280.read_from_db(database_name, "BMP280")
    dht11_from_db = classes.DHT11.read_from_db(database_name, "DHT11")
    gps_from_db = classes.GPS.read_from_db(database_name, "GPS")
    mpu6050_from_db = classes.MPU6050.read_from_db(database_name, "MPU6050")

    for bmp in bmp280_from_db:
        bmp280.append(classes.BMP280(bmp[0], bmp[1], bmp[2], bmp[3]))
    for dht in dht11_from_db:
```

```
        dht11.append(classes.DHT11(dht[0], dht[1]))
    for gps in gps_from_db:
        gpses.append(classes.GPS(gps[0], gps[1], gps[2], gps[3]))
    for mpu in mpu6050_from_db:
        mpu6050.append(classes.MPU6050(mpu[0], mpu[1], mpu[2], mpu[3], mpu[4],␣
↪mpu[5]))
```

1.2 Raw data visualization

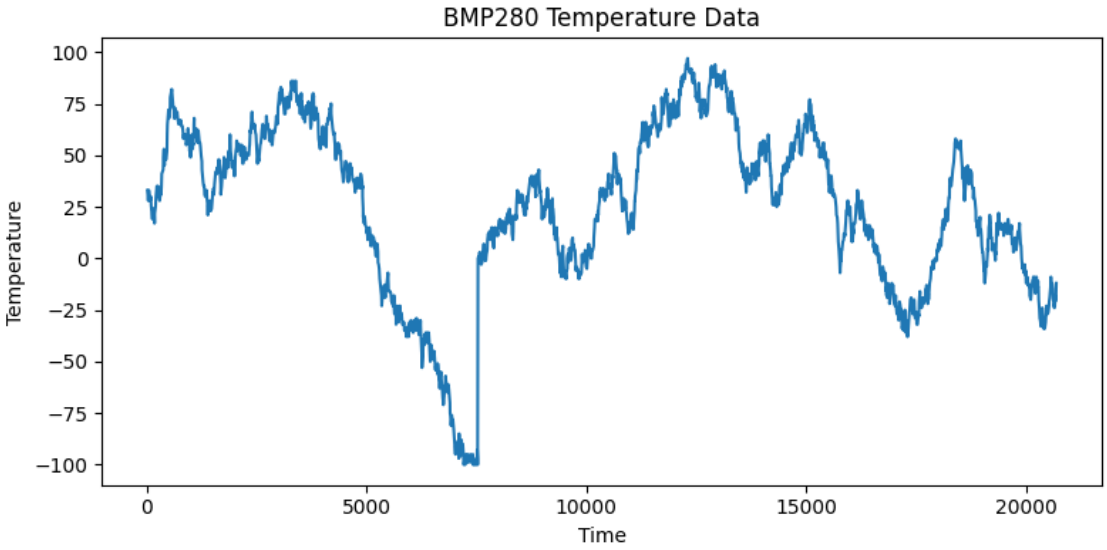
First of all, let's have a look at our measured datas by each sensor on graphs. Due to the sensors' inaccuracies there may be outliers.

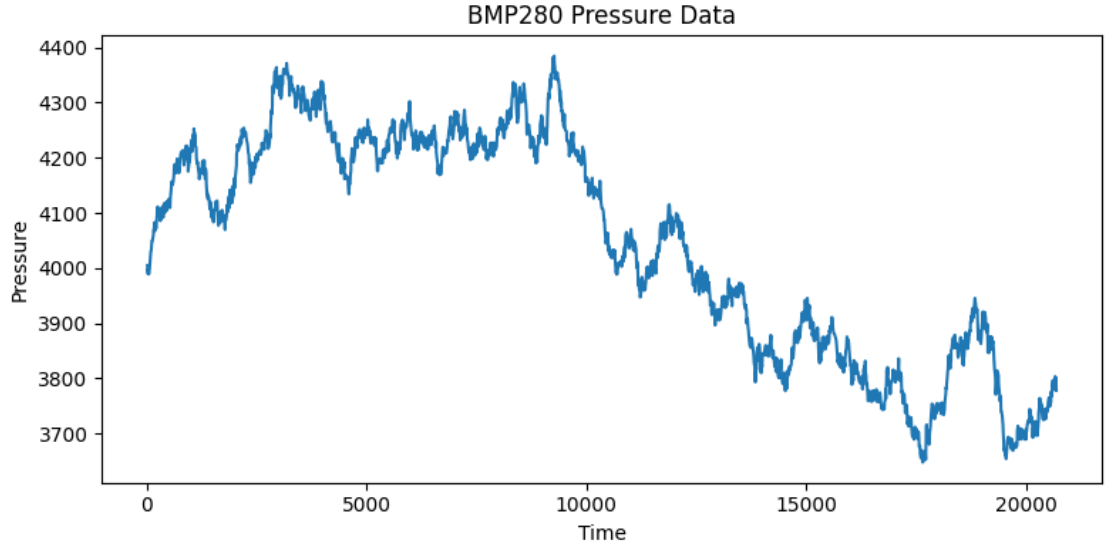
1.2.1 BMP280

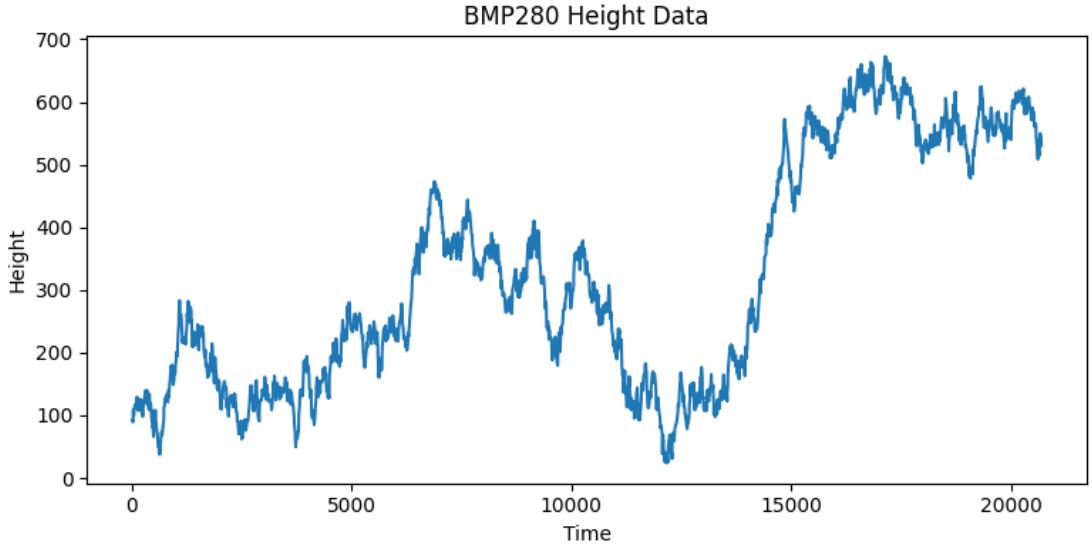
[ ]:
```
if len(bmp280) > 0:
    print("Fortunately, the BMP280 measurements are available. Therefore, we␣
↪can plot the data.")
    print("The amount of data provided by the BMP280 sensor is: ", len(bmp280))
    fig: plt.Figure
    axs: plt.Axes
    fig, axs= plt.subplots(3, figsize=(8, 12))

    axs[0].plot([bmp.time for bmp in bmp280], [bmp.temperature for bmp in␣
↪bmp280], '-')
    axs[0].set_xlabel('Time')
    axs[0].set_ylabel('Temperature')
    axs[0].set_title('BMP280 Temperature Data')

    axs[1].plot([bmp.time for bmp in bmp280], [bmp.pressure for bmp in bmp280],␣
↪'-')
    axs[1].set_xlabel('Time')
    axs[1].set_ylabel('Pressure')
    axs[1].set_title('BMP280 Pressure Data')

    axs[2].plot([bmp.time for bmp in bmp280], [bmp.height for bmp in bmp280],␣
↪'-')
    axs[2].set_xlabel('Time')
    axs[2].set_ylabel('Height')
    axs[2].set_title('BMP280 Height Data')

    plt.tight_layout()
    plt.show()
else:
    print("Unfortunately, the BMP280 measurements are not available. Therefore,␣
↪we cannot plot the data as expected.")
```

```
Fortunately, the BMP280 measurements are available. Therefore, we can plot the
data.
```

```
The amount of data provided by the BMP280 sensor is:  2000
```


BMP280 Temperature Data
Temperature
100
75
50
25
0
−25
−50
−75
−100
0
5000
10000
15000
20000
Time

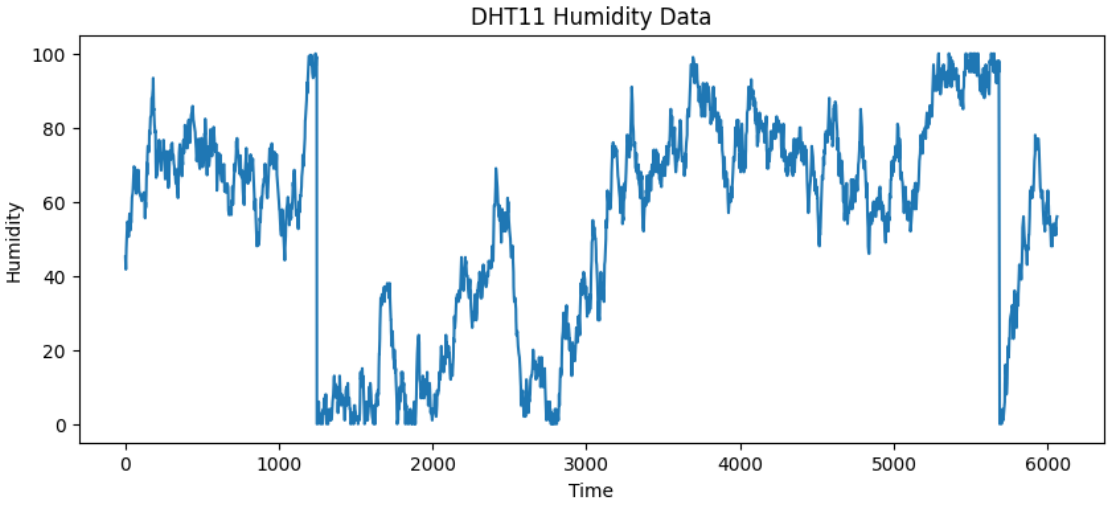

BMP280 Pressure Data
Pressure
4400
4300
4200
4100
4000
3900
3800
3700
0
5000
10000
15000
20000
Time


BMP280 Height Data
Height
700
600
500
400
300
200
100
0
0
5000
10000
15000
20000
Time

1.2.2 DHT11

```
[ ]: if len(dht11) > 0:
        print("Fortunately, the DHT11 measurements are available. Therefore, we can␣
      ↪plot the data.")
        print("The amount of data provided by the DHT11 sensor is: ", len(dht11))
        plt.figure(figsize=(10, 4))
        plt.plot([dht.time for dht in dht11], [dht.humidity for dht in dht11], '-')
        plt.xlabel('Time')
        plt.ylabel('Humidity')
        plt.title('DHT11 Humidity Data')
        plt.show()
    else:
        print("Unfortunately, the DHT11 measurements are not available. Therefore,␣
      ↪we cannot plot the data as expected.")
```

```
Fortunately, the DHT11 measurements are available. Therefore, we can plot the
data.
The amount of data provided by the DHT11 sensor is:  2000
```



1.2.3 GPS

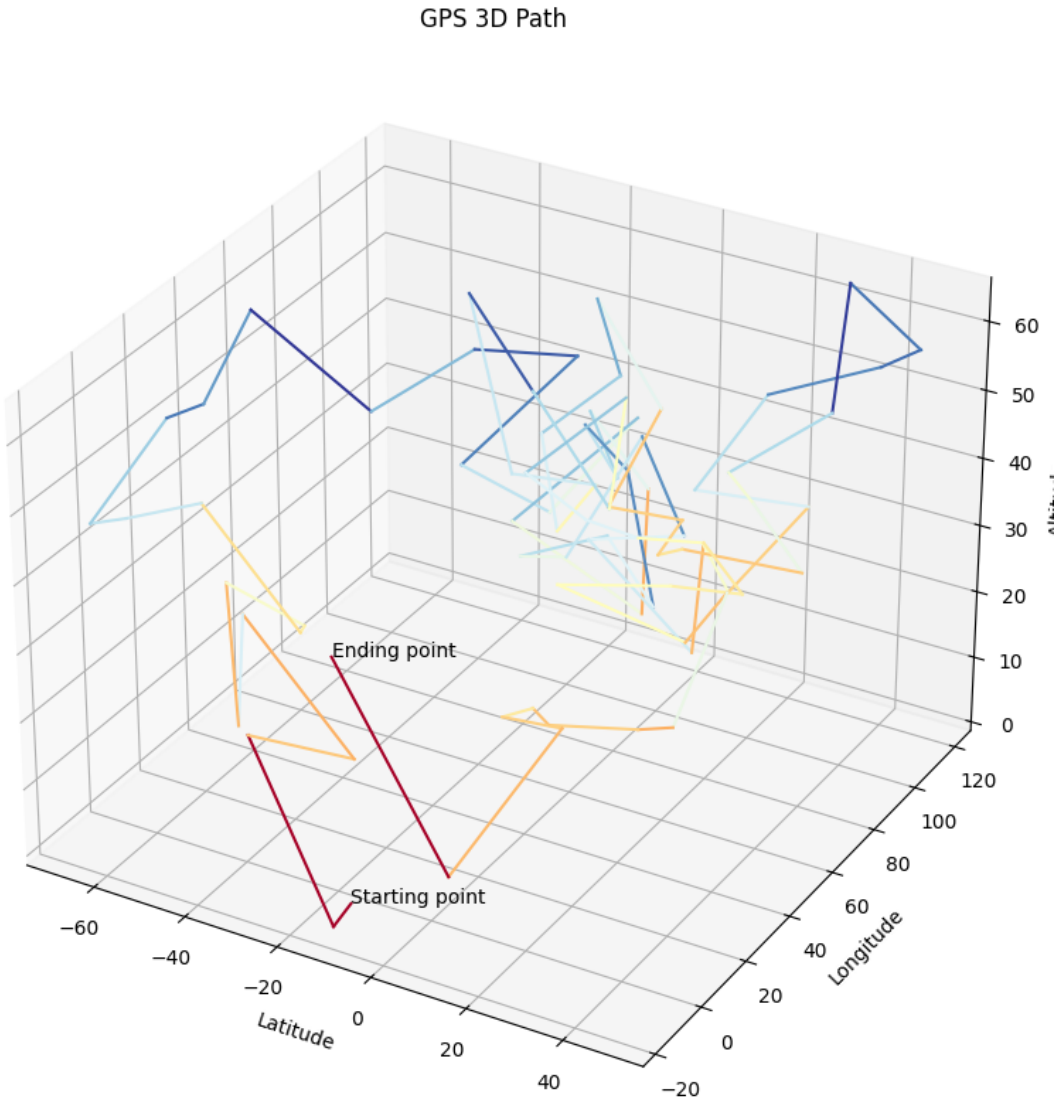
We can create a 3 dimentional map of the gps measurements. Unfortunetly, the altitude data cannot always be measured due to communicational problems. This can cause inaccuracies in the illustration.

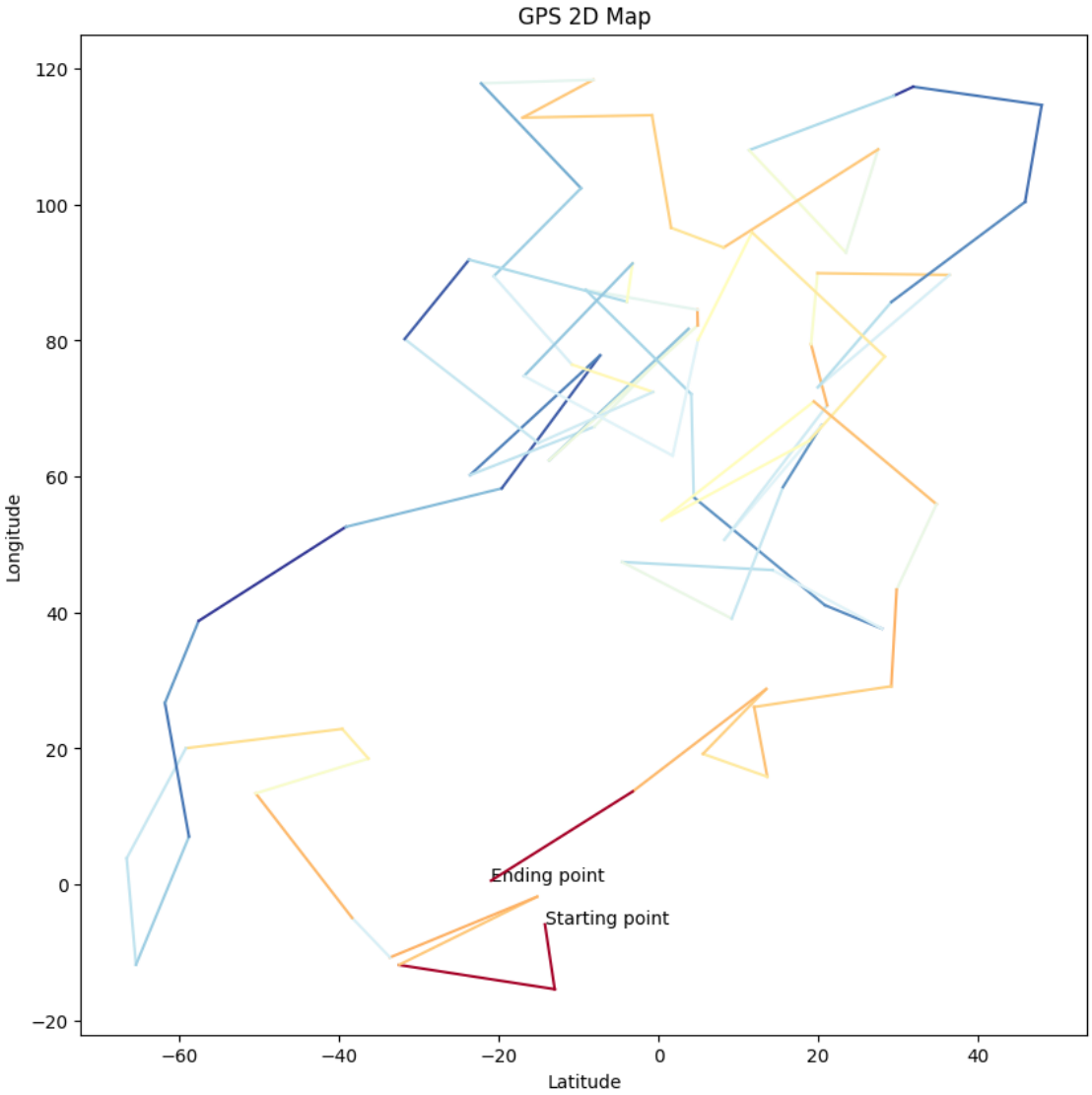
[ ]:
```
if len(gpses) > 0:
    print("Fortunately, the GPS measurements are available. Therefore, we can␣
↪plot the data.")
    print("The amount of data provided by the GPS sensor is: ", len(gpses))
    # Create a color map
    cmap = cm.get_cmap('RdYlBu')  # Red to Blue color map

    # Normalize altitude values to [0, 1] range
    altitudes = np.array([gps.altitude for gps in gpses])
    normalized_altitudes = (altitudes - altitudes.min()) / (altitudes.max() -␣
↪altitudes.min())

    # Map altitude values to colors
    colors = cmap(normalized_altitudes)

    # Plot the GPS data
    fig = plt.figure(figsize=(10, 14))
    # 3D path map
    ax = plt.axes(projection='3d')
    for i in range(len(gpses) - 1):
        ax.plot3D([gpses[i].latitude, gpses[i+1].latitude],
                  [gpses[i].longitude, gpses[i+1].longitude],
                  [gpses[i].altitude, gpses[i+1].altitude],
                  color=colors[i])
    ax.text(gpses[0].latitude, gpses[0].longitude, gpses[0].altitude, 'Starting␣
↪point', size=10, zorder=1, color='k')
    ax.text(gpses[-1].latitude, gpses[-1].longitude, gpses[-1].altitude,␣
↪'Ending point', size=10, zorder=1, color='k')
    ax.set_xlabel('Latitude')
    ax.set_ylabel('Longitude')
    ax.set_zlabel('Altitude')
    ax.set_title('GPS 3D Path')
    plt.show()
    # 2D map
    fig = plt.figure(figsize=(10, 10))
    ax = fig.add_subplot(111)
    for i in range(len(gpses) - 1):
        ax.plot([gpses[i].latitude, gpses[i+1].latitude], [gpses[i].longitude,␣
↪gpses[i+1].longitude], '-', color=colors[i])
    ax.text(gpses[0].latitude, gpses[0].longitude, 'Starting point', size=10,␣
↪zorder=1, color='k')
    ax.text(gpses[-1].latitude, gpses[-1].longitude, 'Ending point', size=10,␣
↪zorder=1, color='k')
    ax.set_xlabel('Latitude')
    ax.set_ylabel('Longitude')
    ax.set_title('GPS 2D Map')
```

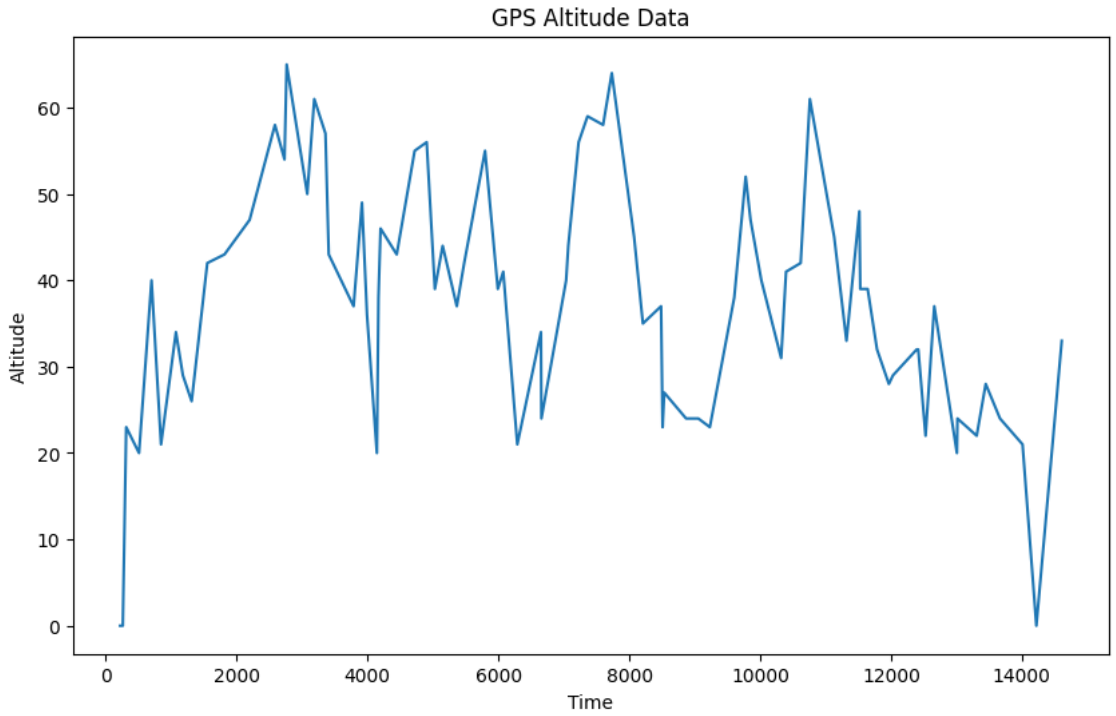
```
    plt.show()
    # 2D altitude-time graph
    fig = plt.figure(figsize=(10, 6))
    ax = fig.add_subplot(111)
    ax.plot([gps.time for gps in gpses], [gps.altitude for gps in gpses], '-')
    ax.set_xlabel('Time')
    ax.set_ylabel('Altitude')
    ax.set_title('GPS Altitude Data')
    plt.show()
else:
    print("Unfortunately, the GPS measurements are not available. Therefore, we␣
 ↪cannot plot the data as expected.")
```

```
Fortunately, the GPS measurements are available. Therefore, we can plot the
data.
The amount of data provided by the GPS sensor is:  80
```

```
C:\Users\Siket Arnold Ádám\AppData\Local\Temp\ipykernel_12848\3711809889.py:5:
MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib
3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or
``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
  cmap = cm.get_cmap('RdYlBu')  # Red to Blue color map
```


GPS 3D Path
Ending point
Starting point
Latitude
Longitude
−60
−40
−20
0
20
40
−20
0
20
40
60
80
100
120
0
10
20
30
40
50
60


GPS 2D Map
Longitude
Latitude
120
100
80
60
40
20
0
−20
−60
−40
−20
0
20
40
Ending point
Starting point



However

1.2.4 MPU6050

1.3 Refining datas

For proper analyzis we should refine the measured datas. This includes the removal of outliers, the detection of lacking data provision and the reorganizement of data blocks. The refined values will be stored in a database.

[ ]:
```
# Fill objects with test data and insert them into the database
try:
    conn = sqlite3.connect(database_name)
    c = conn.cursor()
except Error as e:
    logger.error(f"Error connecting to the database: {e}")

#We should set different values for the launch and for the descent

for i, bmp in enumerate(bmp280[1:], start=1):
    previous_bmp = bmp280[i-1]
    bmp.refine(previous_object=previous_bmp, outlier_iqr_multiplier=1.5,␣
 ↪lacking_data_threshold=100)
    bmp.insert_into_db("BMP280", c)
for i, dht in enumerate(dht11[1:], start=1):
```

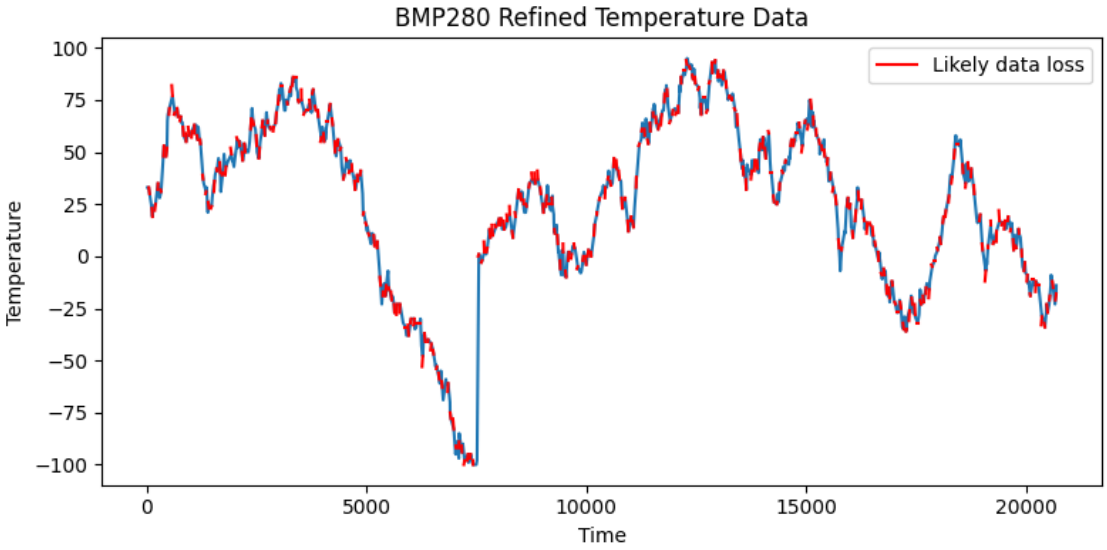
```
    previous_dht = dht11[i-1]
    dht.refine(previous_object=previous_dht, outlier_iqr_multiplier=1.5,␣
 ↪lacking_data_threshold=100)
    dht.insert_into_db("DHT11", c)
for i, gps in enumerate(gpses[1:], start=1):
    previous_gps = gpses[i-1]
    gps.refine(previous_object=previous_gps, outlier_iqr_multiplier=1.5,␣
 ↪lacking_data_threshold=500)
    gps.insert_into_db("GPS", c)
for i, mpu in enumerate(mpu6050[1:], start=1):
    previous_mpu = mpu6050[i-1]
    mpu.refine(previous_object=previous_mpu, outlier_iqr_multiplier=1.5,␣
 ↪lacking_data_threshold=100)
    mpu.insert_into_db("MPU6050", c)
try:
    conn.commit()
    conn.close()
except Error as e:
    logger.error(f"Error committing changes to the database: {e}")
```

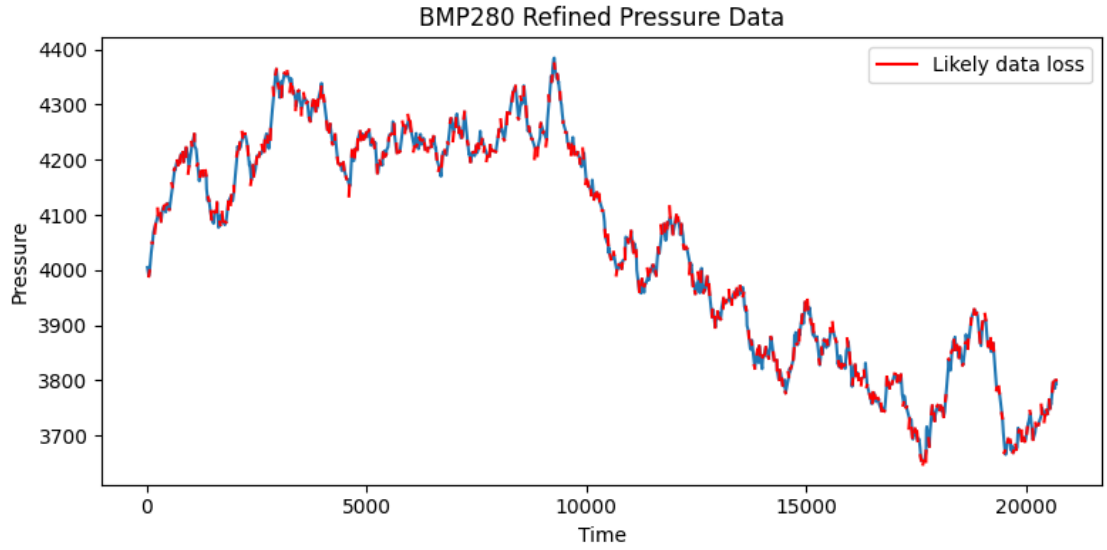
1.3.1 Visualization of refined datas

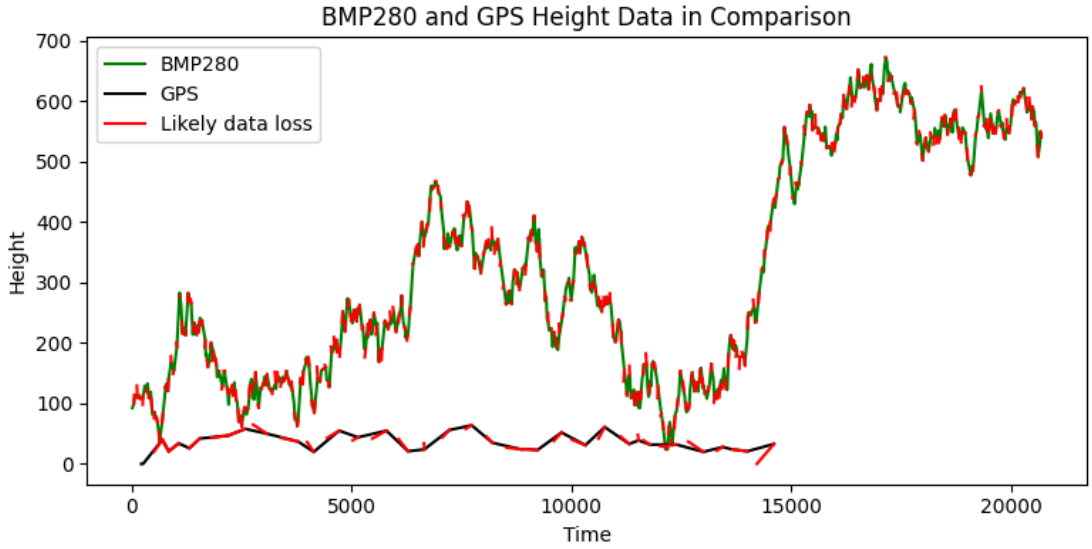
Let's visualize the refined datas for comparition aganist the raw datas.

[ ]:
```
# Visualize data
first_time = True
if len(bmp280) > 0:
    fig: plt.Figure
    axs: plt.Axes
    fig, axs= plt.subplots(3, figsize=(8, 12))

    axs[0].plot([bmp.time for bmp in bmp280 if not bmp.is_outlier], [bmp.
 ↪temperature for bmp in bmp280 if not bmp.is_outlier], '-')
    axs[0].set_xlabel('Time')
    axs[0].set_ylabel('Temperature')
    axs[0].set_title('BMP280 Refined Temperature Data')
    for i, bmp in enumerate(bmp280[1:], start=1):
        if bmp.missing_data:
            axs[0].plot([bmp280[i-1].time, bmp.time], [bmp280[i-1].temperature,␣
 ↪bmp.temperature], '-', color='red', label='Likely data loss' if first_time␣
 ↪else '')
            first_time = False
    axs[0].legend()
    first_time = True

    axs[1].plot([bmp.time for bmp in bmp280 if not bmp.is_outlier], [bmp.
 ↪pressure for bmp in bmp280 if not bmp.is_outlier], '-')
```

```
    axs[1].set_xlabel('Time')
    axs[1].set_ylabel('Pressure')
    axs[1].set_title('BMP280 Refined Pressure Data')
    for i, bmp in enumerate(bmp280[1:], start=1):
        if bmp.missing_data:
            axs[1].plot([bmp280[i-1].time, bmp.time], [bmp280[i-1].pressure,␣
↪bmp.pressure], '-', color='red', label='Likely data loss' if first_time else␣
↪'')
            first_time = False
    axs[1].legend()
    first_time = True

    # Visualize height data(comparing bmp280 and gps with different colors)
    axs[2].plot([bmp.time for bmp in bmp280 if not bmp.is_outlier], [bmp.height␣
↪for bmp in bmp280 if not bmp.is_outlier], '-', color='green', label='BMP280')
    axs[2].plot([gps.time for gps in gpses if not gps.is_outlier], [gps.
↪altitude for gps in gpses if not gps.is_outlier], '-', color='black',␣
↪label='GPS')
    axs[2].set_xlabel('Time')
    axs[2].set_ylabel('Height')
    axs[2].set_title('BMP280 and GPS Height Data in Comparison')
    for i, bmp in enumerate(bmp280[1:], start=1):
        if bmp.missing_data:
            axs[2].plot([bmp280[i-1].time, bmp.time], [bmp280[i-1].height, bmp.
↪height], '-', color='red', label='Likely data loss' if first_time else '')
            first_time = False
    for i, gps in enumerate(gpses[1:], start=1):
        if gps.missing_data:
            axs[2].plot([gpses[i-1].time, gps.time], [gpses[i-1].altitude, gps.
↪altitude], '-', color='red')
    axs[2].legend()

    plt.tight_layout()
    plt.show()
```


BMP280 Refined Temperature Data
Likely data loss
Temperature
Time
100
75
50
25
0
−25
−50
−75
−100
0
5000
10000
15000
20000


BMP280 Refined Pressure Data
Likely data loss
Pressure
Time
4400
4300
4200
4100
4000
3900
3800
3700
0
5000
10000
15000
20000


BMP280 and GPS Height Data in Comparison
BMP280
GPS
Likely data loss
Height
Time
700
600
500
400
300
200
100
0
0
5000
10000
15000
20000

1.4 CanSat path 3D illustration

1.5 Calculating windspeed and comparition

1.5.1 Comparing temperature data with official sources

If we would like to compare our temperature measurements as well with official weather forecast values, we have to webscrape the datas from a proper website. We will use https://koponyeg.hu/elorejelzes/Tat%C3%A1rszentgy%C3%B6rgy for this purpose.

[ ]:
```
official_datas = cansattools.get_official_data()
official_temperatures = official_datas["temperatures"]
official_times = official_datas["times"]
# Adjusting timestamps
official_times[0] = bmp280[0].time
for i in range(1, len(official_times)):
    official_times[i] = official_times[i-1] + 1000

# Visualize bmp280 and official temperatures
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111)
ax.plot([bmp.time for bmp in bmp280 if not bmp.is_outlier], [bmp.temperature␣
 ↪for bmp in bmp280 if not bmp.is_outlier], '-', color='green', label='BMP280')
ax.plot(official_times, official_temperatures, '-', color='black',␣
 ↪label='Official')
ax.set_xlabel('Time')
ax.set_ylabel('Temperature')
ax.set_title('BMP280 and Official Temperatures in Comparison')
for i, bmp in enumerate(bmp280[1:], start=1):
    if bmp.missing_data:
        ax.plot([bmp280[i-1].time, bmp.time], [bmp280[i-1].temperature, bmp.
 ↪temperature], '-', color='red', label='Likely data loss' if first_time else␣
 ↪'')
        first_time = False
ax.legend()
plt.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\PIL\ImageFile.
 ↪py:554, in _save(im, fp, tile, bufsize)
    553 try:
--> 554     fh = fp.fileno()
    555     fp.flush()
```

```
AttributeError: '_idat' object has no attribute 'fileno'

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
Cell In[10], line 23
     21 ax.legend()
     22 plt.tight_layout()
---> 23 plt.show()

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\pyplot
 ↪py:612, in show(*args, **kwargs)
    568 """
    569 Display all open figures.
    570
   (…)
    609 explicitly there.
    610 """
    611 _warn_if_gui_out_of_main_thread()
--> 612 return _get_backend_mod().show(*args, **kwargs)

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\matplotlib_inline\backend_in .ine.
 ↪py:90, in show(close, block)
     88 try:
     89     for figure_manager in Gcf.get_all_fig_managers():
---> 90         display(
     91             figure_manager.canvas.figure,
     92             metadata=_fetch_figure_metadata(figure_manager.canvas.figur )
     93         )
     94 finally:
     95     show._to_draw = []

File␣
 ↪~\AppData\Roaming\Python\Python312\site-packages\IPython\core\display_functic s.
 ↪py:298, in display(include, exclude, metadata, transient, display_id, raw,␣
 ↪clear, *objs, **kwargs)
    296     publish_display_data(data=obj, metadata=metadata, **kwargs)
    297 else:
--> 298     format_dict, md_dict = format(obj, include=include, exclude=exclude
    299     if not format_dict:
    300         # nothing to display (e.g. _ipython_display_ took over)
    301         continue

File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\formatters.p :
 ↪182, in DisplayFormatter.format(self, obj, include, exclude)
    180 md = None
    181 try:
```

```
--> 182     data = formatter(obj)
    183 except:
    184     # FIXME: log the exception
    185     raise

File ~\AppData\Roaming\Python\Python312\site-packages\decorator.py:232, in␣
 ↪decorate.<locals>.fun(*args, **kw)
    230 if not kwsyntax:
    231     args, kw = fix(args, kw, sig)
--> 232 return caller(func, *(extras + args), **kw)

File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\formatters.py:
 ↪226, in catch_format_error(method, self, *args, **kwargs)
    224 """show traceback on failed format call"""
    225 try:
--> 226     r = method(self, *args, **kwargs)
    227 except NotImplementedError:
    228     # don't warn on NotImplementedErrors
    229     return self._check_return(None, args[0])

File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\formatters.py:
 ↪343, in BaseFormatter.__call__(self, obj)
    341     pass
    342 else:
--> 343     return printer(obj)
    344 # Finally look for special method names
    345 method = get_real_method(obj, self.print_method)

File ~\AppData\Roaming\Python\Python312\site-packages\IPython\core\pylabtools.py:
 ↪170, in print_figure(fig, fmt, bbox_inches, base64, **kwargs)
    167     from matplotlib.backend_bases import FigureCanvasBase
    168     FigureCanvasBase(fig)
--> 170 fig.canvas.print_figure(bytes_io, **kw)
    171 data = bytes_io.getvalue()
    172 if fmt == 'svg':

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\backen␣
 ↪_bases.py:2204, in FigureCanvasBase.print_figure(self, filename, dpi, facecolor,␣
 ↪edgecolor, orientation, format, bbox_inches, pad_inches, bbox_extra_artists,␣
 ↪backend, **kwargs)
   2200 try:
   2201     # _get_renderer may change the figure dpi (as vector formats
   2202     # force the figure dpi to 72), so we need to set it again here.
   2203     with cbook._setattr_cm(self.figure, dpi=dpi):
-> 2204         result = print_method(
   2205             filename,
   2206             facecolor=facecolor,
```

```
   2207             edgecolor=edgecolor,
   2208             orientation=orientation,
   2209             bbox_inches_restore=_bbox_inches_restore,
   2210             **kwargs)
   2211 finally:
   2212     if bbox_inches and restore_bbox:

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\backen l_bases.
 ↪py:2054, in FigureCanvasBase._switch_canvas_and_return_print_method.<locals>.
 ↪<lambda>(*args, **kwargs)
   2050     optional_kws = {  # Passed by print_figure for other renderers.
   2051         "dpi", "facecolor", "edgecolor", "orientation",
   2052         "bbox_inches_restore"}
   2053     skip = optional_kws - {*inspect.signature(meth).parameters}
-> 2054     print_method = functools.wraps(meth)(lambda *args, **kwargs: meth(
   2055         *args, **{k: v for k, v in kwargs.items() if k not in skip}))
   2056 else:  # Let third-parties do as they see fit.
   2057     print_method = meth

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\backen ls\backend_agg
 ↪py:496, in FigureCanvasAgg.print_png(self, filename_or_obj, metadata,␣
 ↪pil_kwargs)
    449 def print_png(self, filename_or_obj, *, metadata=None, pil_kwargs=None)
    450     """
    451     Write the figure to a PNG file.
    452
   (…)
    494         *metadata*, including the default 'Software' key.
    495     """
--> 496     self._print_pil(filename_or_obj, "png", pil_kwargs, metadata)

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\backen ls\backend_agg
 ↪py:445, in FigureCanvasAgg._print_pil(self, filename_or_obj, fmt, pil_kwargs,␣
 ↪metadata)
    440 """
    441 Draw the canvas, then save it using `.image.imsave` (to which
    442 *pil_kwargs* and *metadata* are forwarded).
    443 """
    444 FigureCanvasAgg.draw(self)
--> 445 mpl.image.imsave(
    446     filename_or_obj, self.buffer_rgba(), format=fmt, origin="upper",
    447     dpi=self.figure.dpi, metadata=metadata, pil_kwargs=pil_kwargs)

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\image.
 ↪py:1676, in imsave(fname, arr, vmin, vmax, cmap, format, origin, dpi,␣
 ↪metadata, pil_kwargs)
```

```
   1674 pil_kwargs.setdefault("format", format)
   1675 pil_kwargs.setdefault("dpi", (dpi, dpi))
-> 1676 image.save(fname, **pil_kwargs)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\PIL\Image.py:
 ↪2605, in Image.save(self, fp, format, **params)
   2602     fp = cast(IO[bytes], fp)
   2604 try:
-> 2605     save_handler(self, fp, filename)
   2606 except Exception:
   2607     if open_fp:

File␣
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\PIL\PngImagePlugi .
 ↪py:1488, in _save(im, fp, filename, chunk, save_all)
   1484     single_im = _write_multiple_frames(
   1485         im, fp, chunk, mode, rawmode, default_image, append_images
   1486     )
   1487 if single_im:
-> 1488     ImageFile._save(
   1489         single_im,
   1490         cast(IO[bytes], _idat(fp, chunk)),
   1491         [ImageFile._Tile("zip", (0, 0) + single_im.size, 0, rawmode)],
   1492     )
   1494 if info:
   1495     for info_chunk in info.chunks:

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\PIL\ImageFile.
 ↪py:558, in _save(im, fp, tile, bufsize)
    556     _encode_tile(im, fp, tile, bufsize, fh)
    557 except (AttributeError, io.UnsupportedOperation) as exc:
--> 558     _encode_tile(im, fp, tile, bufsize, None, exc)
    559 if hasattr(fp, "flush"):
    560     fp.flush()

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\PIL\ImageFile.
 ↪py:584, in _encode_tile(im, fp, tile, bufsize, fh, exc)
    581 if exc:
    582     # compress to Python file-compatible object
    583     while True:
--> 584         errcode, data = encoder.encode(bufsize)[1:]
    585         fp.write(data)
    586         if errcode:

KeyboardInterrupt:
```

1.6 Convert Notebook into a pdf

For people who cannot open the Notebook due to the lack of environment setup we should convert our analyzis into a PDF file. This code tries to make two different files: one which contains our calculations and scripts besides the outputs (Visualization.pdf) and one which only contains the outputs and markdowns (Results.pdf).

[ ]:
```
import subprocess
try:
    subprocess.run(f"python -m jupyter nbconvert --to pdf --output␣
 ↪Visualization.pdf Visualization.ipynb", shell=True, check=True, timeout=600)
    print("Exporting to PDF...")
except Exception as e:
    logger.error(f"Error exporting the notebook into pdf format: {e}")
if os.path.exists("Visualization.pdf"):
    print("PDF exported successfully.")
else:
    print("PDF export failed.")
    logger.error("PDF export failed.")
```

```
Exporting to PDF…
PDF exported successfully.
```