

wygładzanie GPS, przetwarzanie danych

1.0

Generated by Doxygen 1.9.6

1 Namespace Index	1
1.1 Namespace List	1
2 File Index	3
2.1 File List	3
3 Namespace Documentation	5
3.1 kolumnowe Namespace Reference	5
3.1.1 Function Documentation	6
3.1.1.1 autolabel()	6
3.1.2 Variable Documentation	6
3.1.2.1 ax1	6
3.1.2.2 ax2	7
3.1.2.3 figsize	7
3.1.2.4 med_1	7
3.1.2.5 med_2	7
3.1.2.6 med_3	7
3.1.2.7 med_4	7
3.1.2.8 mediana_1	7
3.1.2.9 mediana_2	7
3.1.2.10 mediana_3	8
3.1.2.11 mediana_4	8
3.1.2.12 methods_1	8
3.1.2.13 methods_2	8
3.1.2.14 methods_3	8
3.1.2.15 methods_4	8
3.1.2.16 mse_1	8
3.1.2.17 mse_2	8
3.1.2.18 mse_3	9
3.1.2.19 mse_4	9
3.1.2.20 rects1_1	9
3.1.2.21 rects1_2	9
3.1.2.22 rects1_3	9
3.1.2.23 rects1_4	9
3.1.2.24 rects2_1	9
3.1.2.25 rects2_2	9
3.1.2.26 rects2_3	10
3.1.2.27 rects2_4	10
3.1.2.28 rects3_1	10
3.1.2.29 rects3_2	10
3.1.2.30 rects3_3	10
3.1.2.31 rects3_4	10
3.1.2.32 rects4_1	10

3.1.2.33 rect4_2	10
3.1.2.34 rect4_3	11
3.1.2.35 rect4_4	11
3.1.2.36 rmse_1	11
3.1.2.37 rmse_2	11
3.1.2.38 rmse_3	11
3.1.2.39 rmse_4	11
3.1.2.40 width	11
3.1.2.41 x	11
3.2 main Namespace Reference	12
3.2.1 Function Documentation	13
3.2.1.1 calculate_groups_errors()	13
3.2.1.2 calculate_mean_euclidean_error()	14
3.2.1.3 calculate_mse()	14
3.2.1.4 calculate_rmse()	14
3.2.1.5 euclidean_distance()	16
3.2.1.6 find_closest_point()	16
3.2.1.7 group_data()	17
3.2.1.8 group_data_by_original()	17
3.2.1.9 haversine_distance()	17
3.2.1.10 interpolate_reference()	18
3.2.2 Variable Documentation	18
3.2.2.1 ax	18
3.2.2.2 axs	18
3.2.2.3 bbox_to_anchor	18
3.2.2.4 c	19
3.2.2.5 color	19
3.2.2.6 config	19
3.2.2.7 config_path	19
3.2.2.8 data_files	19
3.2.2.9 errors	19
3.2.2.10 errors2	19
3.2.2.11 fig	19
3.2.2.12 figsize	20
3.2.2.13 full_path	20
3.2.2.14 group	20
3.2.2.15 group_colors	20
3.2.2.16 grouped_original_data	20
3.2.2.17 grouped_smoothed_data	20
3.2.2.18 handles	20
3.2.2.19 hb	21
3.2.2.20 interpolated_ref_data	21

3.2.2.21 interpolated_ref_data_np	21
3.2.2.22 label	21
3.2.2.23 lat	21
3.2.2.24 loc	21
3.2.2.25 lon	21
3.2.2.26 mae	21
3.2.2.27 mae2	22
3.2.2.28 match	22
3.2.2.29 max_distance	22
3.2.2.30 mean_error	22
3.2.2.31 mean_error3	22
3.2.2.32 mean_error_no_groups	22
3.2.2.33 mean_error_no_groups2	22
3.2.2.34 median	22
3.2.2.35 median2	23
3.2.2.36 mse	23
3.2.2.37 mse2	23
3.2.2.38 mse_no_groups	23
3.2.2.39 mse_no_groups2	23
3.2.2.40 original_data	23
3.2.2.41 original_path	23
3.2.2.42 point_index	23
3.2.2.43 points	24
3.2.2.44 ref_data	24
3.2.2.45 ref_path	24
3.2.2.46 rmsd	24
3.2.2.47 rmsd2	24
3.2.2.48 rmse	24
3.2.2.49 rmse_no_groups2	24
3.2.2.50 smoothed_coords	24
3.2.2.51 smoothed_data	25
3.2.2.52 smoothed_path	25
3.2.2.53 time	25
4 File Documentation	27
4.1 kolumnowe.py File Reference	27
4.1.1 Detailed Description	28
4.2 main.py File Reference	28
4.2.1 Detailed Description	30
Index	31

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

kolumnowe	5
main	12

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

kolumnowe.py	Skrypt do wizualizacji danych błędów dla różnych metod wygładzania w formie wykresów kolumnowych	27
main.py	Główny skrypt obliczający błędy i generujący wykresy	28

Chapter 3

Namespace Documentation

3.1 kolumnowe Namespace Reference

Functions

- def [autolabel](#) (ax, rects)

Funkcja dodająca wartości błędów nad kolumnami na wykresach.

Variables

- list [methods_1](#) = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
- list [med_1](#) = [2.426e-5, 2.405e-5, 2.422e-5, 1.79e-5, 2.252e-5, 2.5073e-5]
- list [mse_1](#) = [7.712e-10, 7.627e-10, 7.569e-10, 4.457e-10, 6.931e-10, 8.63e-10]
- list [rmse_1](#) = [2.777e-5, 2.762e-5, 2.751e-5, 2.111e-5, 2.632e-5, 2.9378e-5]
- list [mediana_1](#) = [2.208e-5, 2.207e-5, 2.198e-5, 1.623e-6, 2.014e-5, 2.2815e-5]
- list [methods_2](#) = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
- list [med_2](#) = [0.0001206, 0.0001204, 0.000121, 0.000107, 0.000119, 0.0001075]
- list [mse_2](#) = [2.149e-8, 2.146e-8, 2.2e-8, 1.585e-8, 2.113e-8, 1.591e-8]
- list [rmse_2](#) = [0.0001466, 0.0001462, 0.0001481, 0.0001259, 0.0001453, 0.0001261]
- list [mediana_2](#) = [9.843e-5, 9.83e-5, 0.00010004, 9.704e-5, 0.00010073, 0.0001008]
- np [x](#) = np.arange(len([methods_1](#)))
- float [width](#) = 0.2
- [ax1](#)

Tworzenie wykresów kolumnowych dla zestawów danych z Tabeli 1 i Tabeli 2.

- [ax2](#)
- [figsize](#)
- [ax1](#) [rects1_1](#) = [ax1](#).bar([x](#) - 1.5*[width](#), [med_1](#), [width](#), label='MED')
- [ax1](#) [rects1_2](#) = [ax1](#).bar([x](#) - 0.5*[width](#), [mse_1](#), [width](#), label='MSE')
- [ax1](#) [rects1_3](#) = [ax1](#).bar([x](#) + 0.5*[width](#), [rmse_1](#), [width](#), label='RMSE')
- [ax1](#) [rects1_4](#) = [ax1](#).bar([x](#) + 1.5*[width](#), [mediana_1](#), [width](#), label='Mediana')
- [ax2](#) [rects2_1](#) = [ax2](#).bar([x](#) - 1.5*[width](#), [med_2](#), [width](#), label='MED')
- [ax2](#) [rects2_2](#) = [ax2](#).bar([x](#) - 0.5*[width](#), [mse_2](#), [width](#), label='MSE')
- [ax2](#) [rects2_3](#) = [ax2](#).bar([x](#) + 0.5*[width](#), [rmse_2](#), [width](#), label='RMSE')
- [ax2](#) [rects2_4](#) = [ax2](#).bar([x](#) + 1.5*[width](#), [mediana_2](#), [width](#), label='Mediana')
- list [methods_3](#) = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
- list [med_3](#) = [2.7245e-5, 2.5288e-5, 2.5782e-5, 1.3462e-5, 2.5942e-5, 3.0398e-5]
- list [mse_3](#) = [8.8167e-10, 7.8341e-10, 7.7622e-10, 2.3869e-10, 8.1655e-10, 1.1928e-9]

- list `rmse_3` = [2.9692e-5, 2.7989e-5, 2.786e-5, 1.5449e-5, 2.8575e-5, 3.4536e-5]
- list `mediana_3` = [2.7446e-5, 2.661e-5, 2.7342e-5, 1.4534e-5, 2.3687e-5, 2.8072e-5]
- list `methods_4` = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
- list `med_4` = [0.0001258, 0.0001174, 0.0001245, 6.505e-5, 0.0001299, 5.986e-5]
- list `mse_4` = [2.2271e-8, 1.9782e-8, 2.2558e-8, 6.7572e-9, 2.4743e-8, 4.3091e-9]
- list `rmse_4` = [0.0001492, 0.0001406, 0.0001501, 8.2202e-5, 0.0001573, 6.5647e-5]
- list `mediana_4` = [0.0001086, 0.000105, 0.0001074, 5.0468e-5, 0.0001117, 5.7155e-5]
- `ax1 rects3_1` = `ax1.bar(x - 1.5*width, med_3, width, label='MED')`
- `ax1 rects3_2` = `ax1.bar(x - 0.5*width, mse_3, width, label='MSE')`
- `ax1 rects3_3` = `ax1.bar(x + 0.5*width, rmse_3, width, label='RMSE')`
- `ax1 rects3_4` = `ax1.bar(x + 1.5*width, mediana_3, width, label='Mediana')`
- `ax2 rects4_1` = `ax2.bar(x - 1.5*width, med_4, width, label='MED')`
- `ax2 rects4_2` = `ax2.bar(x - 0.5*width, mse_4, width, label='MSE')`
- `ax2 rects4_3` = `ax2.bar(x + 0.5*width, rmse_4, width, label='RMSE')`
- `ax2 rects4_4` = `ax2.bar(x + 1.5*width, mediana_4, width, label='Mediana')`

3.1.1 Function Documentation

3.1.1.1 autolabel()

```
def autolabel (
    ax,
    rects )
```

Funkcja dodająca wartości błędów nad kolumnami na wykresach.

Parameters

<code>ax</code>	Oś, na której umieszczony jest wykres.
<code>rects</code>	Lista kolumn, do których będą dodawane wartości błędów.

3.1.2 Variable Documentation

3.1.2.1 ax1

`ax1`

Tworzenie wykresów kolumnowych dla zestawów danych z Tabeli 1 i Tabeli 2.

Ustawienia osi X, Y oraz etykiet. Wykresy przedstawiają wartości błędów w skali logarytmicznej.

3.1.2.2 ax2

```
ax2
```

3.1.2.3 figsize

```
figsize
```

3.1.2.4 med_1

```
list med_1 = [2.426e-5, 2.405e-5, 2.422e-5, 1.79e-5, 2.252e-5, 2.5073e-5]
```

3.1.2.5 med_2

```
list med_2 = [0.0001206, 0.0001204, 0.000121, 0.000107, 0.000119, 0.0001075]
```

3.1.2.6 med_3

```
list med_3 = [2.7245e-5, 2.5288e-5, 2.5782e-5, 1.3462e-5, 2.5942e-5, 3.0398e-5]
```

3.1.2.7 med_4

```
list med_4 = [0.0001258, 0.0001174, 0.0001245, 6.505e-5, 0.0001299, 5.986e-5]
```

3.1.2.8 mediana_1

```
list mediana_1 = [2.208e-5, 2.207e-5, 2.198e-5, 1.623e-6, 2.014e-5, 2.2815e-5]
```

3.1.2.9 mediana_2

```
list mediana_2 = [9.843e-5, 9.83e-5, 0.00010004, 9.704e-5, 0.00010073, 0.0001008]
```

3.1.2.10 mediana_3

```
list mediana_3 = [2.7446e-5, 2.661e-5, 2.7342e-5, 1.4534e-5, 2.3687e-5, 2.8072e-5]
```

3.1.2.11 mediana_4

```
list mediana_4 = [0.0001086, 0.000105, 0.0001074, 5.0468e-5, 0.0001117, 5.7155e-5]
```

3.1.2.12 methods_1

```
list methods_1 = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
```

3.1.2.13 methods_2

```
list methods_2 = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
```

3.1.2.14 methods_3

```
list methods_3 = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
```

3.1.2.15 methods_4

```
list methods_4 = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
```

3.1.2.16 mse_1

```
list mse_1 = [7.712e-10, 7.627e-10, 7.569e-10, 4.457e-10, 6.931e-10, 8.63e-10]
```

3.1.2.17 mse_2

```
list mse_2 = [2.149e-8, 2.146e-8, 2.2e-8, 1.585e-8, 2.113e-8, 1.591e-8]
```

3.1.2.18 mse_3

```
list mse_3 = [8.8167e-10, 7.8341e-10, 7.7622e-10, 2.3869e-10, 8.1655e-10, 1.1928e-9]
```

3.1.2.19 mse_4

```
list mse_4 = [2.2271e-8, 1.9782e-8, 2.2558e-8, 6.7572e-9, 2.4743e-8, 4.3091e-9]
```

3.1.2.20 rects1_1

```
ax1 rects1_1 = ax1.bar(x - 1.5*width, med_1, width, label='MED')
```

3.1.2.21 rects1_2

```
ax1 rects1_2 = ax1.bar(x - 0.5*width, mse_1, width, label='MSE')
```

3.1.2.22 rects1_3

```
ax1 rects1_3 = ax1.bar(x + 0.5*width, rmse_1, width, label='RMSE')
```

3.1.2.23 rects1_4

```
ax1 rects1_4 = ax1.bar(x + 1.5*width, mediana_1, width, label='Mediana')
```

3.1.2.24 rects2_1

```
ax2 rects2_1 = ax2.bar(x - 1.5*width, med_2, width, label='MED')
```

3.1.2.25 rects2_2

```
ax2 rects2_2 = ax2.bar(x - 0.5*width, mse_2, width, label='MSE')
```

3.1.2.26 rects2_3

```
ax2 rects2_3 = ax2.bar(x + 0.5*width, rmse_2, width, label='RMSE')
```

3.1.2.27 rects2_4

```
ax2 rects2_4 = ax2.bar(x + 1.5*width, mediana_2, width, label='Mediana')
```

3.1.2.28 rects3_1

```
ax1 rects3_1 = ax1.bar(x - 1.5*width, med_3, width, label='MED')
```

3.1.2.29 rects3_2

```
ax1 rects3_2 = ax1.bar(x - 0.5*width, mse_3, width, label='MSE')
```

3.1.2.30 rects3_3

```
ax1 rects3_3 = ax1.bar(x + 0.5*width, rmse_3, width, label='RMSE')
```

3.1.2.31 rects3_4

```
ax1 rects3_4 = ax1.bar(x + 1.5*width, mediana_3, width, label='Mediana')
```

3.1.2.32 rects4_1

```
ax2 rects4_1 = ax2.bar(x - 1.5*width, med_4, width, label='MED')
```

3.1.2.33 rects4_2

```
ax2 rects4_2 = ax2.bar(x - 0.5*width, mse_4, width, label='MSE')
```


3.1.2.34 rects4_3

```
ax2 rects4_3 = ax2.bar(x + 0.5*width, rmse_4, width, label='RMSE')
```

3.1.2.35 rects4_4

```
ax2 rects4_4 = ax2.bar(x + 1.5*width, mediana_4, width, label='Mediana')
```

3.1.2.36 rmse_1

```
list rmse_1 = [2.777e-5, 2.762e-5, 2.751e-5, 2.111e-5, 2.632e-5, 2.9378e-5]
```

3.1.2.37 rmse_2

```
list rmse_2 = [0.0001466, 0.0001462, 0.0001481, 0.0001259, 0.0001453, 0.0001261]
```

3.1.2.38 rmse_3

```
list rmse_3 = [2.9692e-5, 2.7989e-5, 2.786e-5, 1.5449e-5, 2.8575e-5, 3.4536e-5]
```

3.1.2.39 rmse_4

```
list rmse_4 = [0.0001492, 0.0001406, 0.0001501, 8.2202e-5, 0.0001573, 6.5647e-5]
```

3.1.2.40 width

```
float width = 0.2
```

3.1.2.41 x

```
np x = np.arange(len(methods_1))
```

3.2 main Namespace Reference

Functions

- def `euclidean_distance` (p1, p2)
Oblicza odległość euklidesową pomiędzy dwoma punktami.
- def `haversine_distance` (coord1, coord2)
Oblicza odległość Haversine pomiędzy dwoma współrzędnymi geograficznymi.
- def `interpolate_reference` (data, num_points)
Interpoluje dane referencyjne, aby uzyskać określoną liczbę punktów.
- def `group_data` (data)
Grupuje dane wygładzone według grup.
- def `group_data_by_original` (original_data, smoothed_data)
Grupuje dane oryginalne na podstawie grup z danych wygładzonych.
- def `calculate_groups_errors` (grouped_smoothed_data, grouped_original_data, ref_data)
Oblicza błędy dla grup danych wygładzonych i oryginalnych w odniesieniu do danych referencyjnych.
- def `find_closest_point` (target_point, reference_points)
Znajduje najbliższy punkt do zadanego punktu docelowego wśród punktów referencyjnych.
- def `calculate_mean_euclidean_error` (smoothed_data, ref_data)
Oblicza średni błąd euklidesowy między danymi wygładzonymi a referencyjnymi.
- def `calculate_mse` (smoothed_data, ref_data)
Oblicza średni błąd kwadratowy (MSE).
- def `calculate_rmse` (smoothed_data, ref_data)
Oblicza pierwiastek z średniego błędu kwadratowego (RMSE).

Variables

- str `config_path` = 'config.yaml'
- yaml `config` = yaml.safe_load(config_file)
- yaml `data_files` = config['data_files']
- list `original_data` = []
- list `smoothed_data` = []
- list `ref_data` = []
- os `full_path` = os.path.join(os.getcwd(), file_path)
- re `match` = re.search(r'Point (\d+):Latitude:([\d.]+)\sLongitude:([\d.]+)', line)
- int `point_index` = int(match.group(1))
- float `lat` = float(match.group(2))
- float `lon` = float(match.group(3))
- float `mae` = float(match.group(4))
- int `group` = int(match.group(5))
- int `time` = int(match.group(6))
- np `interpolated_ref_data` = np.array(interpolate_reference(ref_data, len(smoothed_data)))
- list `smoothed_coords` = smoothed_data[:, 1:3]
- def `grouped_smoothed_data` = group_data(smoothed_data)
- def `grouped_original_data` = group_data_by_original(original_data, smoothed_data)
- `mean_error`
- `mean_error3`
- `mse`
- `mse2`
- `mae2`
- `rmsd`

- `rmsd2`
- `median`
- `median2`
- `errors`
- `mean_error_no_groups`
- `errors2`
- `def mse_no_groups = calculate_mse(smoothed_data, interpolated_ref_data)`
- `def rmse = calculate_rmse(smoothed_data, interpolated_ref_data)`
- `def mean_error_no_groups2 = calculate_mean_euclidean_error(original_data, interpolated_ref_data)`
- `def mse_no_groups2 = calculate_mse(original_data, interpolated_ref_data)`
- `def rmse_no_groups2 = calculate_rmse(original_data, interpolated_ref_data)`
- `np interpolated_ref_data_np = np.array(interpolated_ref_data)`
- `list group_colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black', 'orange', 'purple', 'brown']`
- `fig`
- `ax`
- `figsize`
- `np points = np.array(points)`
- `label`
- `color`
- `bbox_to_anchor`
- `loc`
- `axs`
- `float max_distance = 0.00018`
- `list original_path = original_data[:, 1:3]`
- `list smoothed_path = smoothed_data[:, 1:3]`
- `np ref_path = interpolated_ref_data`
- `c`
- `handles`
- `axs hb = axs[1].hexbin(smoothed_data[:, 1], smoothed_data[:, 2], gridsize=50, cmap='inferno')`

3.2.1 Function Documentation

3.2.1.1 `calculate_groups_errors()`

```
def calculate_groups_errors (
    grouped_smoothed_data,
    grouped_original_data,
    ref_data )
```

Oblicza błędy dla grup danych wygładzonych i oryginalnych w odniesieniu do danych referencyjnych.

Parameters

<code>grouped_smoothed_data</code>	Zgrupowane dane wygładzone.
<code>grouped_original_data</code>	Zgrupowane dane oryginalne.
<code>ref_data</code>	Dane referencyjne.

Returns

Krotka zawierająca średnie błędy, MSE, MAE, RMSE i mediany błędów dla danych wygładzonych i oryginalnych.

3.2.1.2 calculate_mean_euclidean_error()

```
def calculate_mean_euclidean_error (
    smoothed_data,
    ref_data )
```

Oblicza średni błąd euklidesowy między danymi wygładzonymi a referencyjnymi.

Parameters

<i>smoothed_data</i>	Dane wygładzone.
<i>ref_data</i>	Dane referencyjne.

Returns

Średni błąd euklidesowy i lista błędów dla każdego punktu.

3.2.1.3 calculate_mse()

```
def calculate_mse (
    smoothed_data,
    ref_data )
```

Oblicza średni błąd kwadratowy (MSE).

Parameters

<i>smoothed_data</i>	Dane wygładzone.
<i>ref_data</i>	Dane referencyjne.

Returns

Wartość MSE.

3.2.1.4 calculate_rmse()

```
def calculate_rmse (
    smoothed_data,
    ref_data )
```

Oblicza pierwiastek z średniego błędu kwadratowego (RMSE).

Parameters

<i>smoothed_data</i>	Dane wygładzone.
<i>ref_data</i>	Dane referencyjne.

Returns

Wartość RMSE.

3.2.1.5 euclidean_distance()

```
def euclidean_distance (
    p1,
    p2 )
```

Oblicza odległość euklidesową pomiędzy dwoma punktami.

Parameters

<i>p1</i>	Pierwszy punkt w formacie (x, y).
<i>p2</i>	Drugi punkt w formacie (x, y).

Returns

Odległość euklidesowa pomiędzy p1 a p2.

3.2.1.6 find_closest_point()

```
def find_closest_point (
    target_point,
    reference_points )
```

Znajduje najbliższy punkt do zadanego punktu docelowego wśród punktów referencyjnych.

Parameters

<i>target_point</i>	Punkt docelowy (x, y).
<i>reference_points</i>	Lista punktów referencyjnych (x, y).

Returns

Najbliższy punkt referencyjny do target_point.

3.2.1.7 group_data()

```
def group_data (
    data )
```

Grupuje dane wygładzone według grup.

Parameters

<i>data</i>	Dane wygładzone w formacie (indeks, szerokość, długość, MAE, grupa, czas).
-------------	--

Returns

Słownik, gdzie kluczem jest numer grupy, a wartością lista punktów należących do tej grupy.

3.2.1.8 group_data_by_original()

```
def group_data_by_original (
    original_data,
    smoothed_data )
```

Grupuje dane oryginalne na podstawie grup z danych wygładzonych.

Parameters

<i>original_data</i>	Dane oryginalne w formacie (indeks, szerokość, długość).
<i>smoothed_data</i>	Dane wygładzone w formacie (indeks, szerokość, długość, MAE, grupa, czas).

Returns

Słownik, gdzie kluczem jest numer grupy, a wartością lista punktów oryginalnych należących do tej grupy.

3.2.1.9 haversine_distance()

```
def haversine_distance (
    coord1,
    coord2 )
```

Oblicza odległość Haversine pomiędzy dwoma współrzędnymi geograficznymi.

Parameters

<i>coord1</i>	Pierwsza współrzędna w formacie (szerokość, długość).
<i>coord2</i>	Druga współrzędna w formacie (szerokość, długość).

Returns

Odległość w kilometrach pomiędzy coord1 a coord2.

3.2.1.10 interpolate_reference()

```
def interpolate_reference (
    data,
    num_points )
```

Interpoluje dane referencyjne, aby uzyskać określoną liczbę punktów.

Parameters

<i>data</i>	Lista punktów referencyjnych w formacie (szerokość, długość).
<i>num_points</i>	Liczba punktów do uzyskania po interpolacji.

Returns

Lista interpolowanych punktów w formacie (szerokość, długość).

3.2.2 Variable Documentation**3.2.2.1 ax**

ax

3.2.2.2 axs

axs

3.2.2.3 bbox_to_anchor

bbox_to_anchor

3.2.2.4 c

c

3.2.2.5 color

color

3.2.2.6 config

```
yaml config = yaml.safe_load(config_file)
```

3.2.2.7 config_path

```
str config_path = 'config.yaml'
```

3.2.2.8 data_files

```
yaml data_files = config['data_files']
```

3.2.2.9 errors

errors

3.2.2.10 errors2

errors2

3.2.2.11 fig

fig

3.2.2.12 `figsize`

```
figsize
```

3.2.2.13 `full_path`

```
os full_path = os.path.join(os.getcwd(), file_path)
```

3.2.2.14 `group`

```
int group = int(match.group(5))
```

3.2.2.15 `group_colors`

```
list group_colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black', 'orange',  
'purple', 'brown']
```

3.2.2.16 `grouped_original_data`

```
def grouped_original_data = group\_data\_by\_original(original_data, smoothed_data)
```

3.2.2.17 `grouped_smoothed_data`

```
def grouped_smoothed_data = group\_data(smoothed_data)
```

3.2.2.18 `handles`

```
handles
```

3.2.2.19 hb

```
axs hb = axs[1].hexbin(smoothed_data[:, 1], smoothed_data[:, 2], gridsize=50, cmap='inferno')
```

3.2.2.20 interpolated_ref_data

```
np interpolated_ref_data = np.array(interpolate_reference(ref_data, len(smoothed_data)))
```

3.2.2.21 interpolated_ref_data_np

```
np interpolated_ref_data_np = np.array(interpolated_ref_data)
```

3.2.2.22 label

```
label
```

3.2.2.23 lat

```
float lat = float(match.group(2))
```

3.2.2.24 loc

```
loc
```

3.2.2.25 lon

```
float lon = float(match.group(3))
```

3.2.2.26 mae

```
mae = float(match.group(4))
```

3.2.2.27 mae2

mae2

3.2.2.28 match

```
re match = re.search(r'Point (\d+):Latitude:([\d.]+)\sLongitude:([\d.]+)', line)
```

3.2.2.29 max_distance

```
float max_distance = 0.00018
```

3.2.2.30 mean_error

mean_error

3.2.2.31 mean_error3

mean_error3

3.2.2.32 mean_error_no_groups

mean_error_no_groups

3.2.2.33 mean_error_no_groups2

```
def mean_error_no_groups2 = calculate_mean_euclidean_error(original_data, interpolated_ref_data)
```

3.2.2.34 median

median

3.2.2.35 median2

```
median2
```

3.2.2.36 mse

```
mse
```

3.2.2.37 mse2

```
mse2
```

3.2.2.38 mse_no_groups

```
def mse_no_groups = calculate_mse(smoothed_data, interpolated_ref_data)
```

3.2.2.39 mse_no_groups2

```
def mse_no_groups2 = calculate_mse(original_data, interpolated_ref_data)
```

3.2.2.40 original_data

```
np original_data = []
```

3.2.2.41 original_path

```
list original_path = original_data[:, 1:3]
```

3.2.2.42 point_index

```
int point_index = int(match.group(1))
```

3.2.2.43 points

```
np points = np.array(points)
```

3.2.2.44 ref_data

```
np ref_data = []
```

3.2.2.45 ref_path

```
np ref_path = interpolated_ref_data
```

3.2.2.46 rmsd

```
rmsd
```

3.2.2.47 rmsd2

```
rmsd2
```

3.2.2.48 rmse

```
def rmse = calculate_rmse(smoothed_data, interpolated_ref_data)
```

3.2.2.49 rmse_no_groups2

```
def rmse_no_groups2 = calculate_rmse(original_data, interpolated_ref_data)
```

3.2.2.50 smoothed_coords

```
list smoothed_coords = smoothed_data[:, 1:3]
```

3.2.2.51 smoothed_data

```
list smoothed_data = []
```

3.2.2.52 smoothed_path

```
list smoothed_path = smoothed_data[:, 1:3]
```

3.2.2.53 time

```
int time = int(match.group(6))
```


Chapter 4

File Documentation

4.1 kolumnowe.py File Reference

Skrypt do wizualizacji danych błędów dla różnych metod wygładzania w formie wykresów kolumnowych.

Namespaces

- namespace [kolumnowe](#)

Functions

- def [autolabel](#) (ax, rects)
Funkcja dodająca wartości błędów nad kolumnami na wykresach.

Variables

- list [methods_1](#) = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
- list [med_1](#) = [2.426e-5, 2.405e-5, 2.422e-5, 1.79e-5, 2.252e-5, 2.5073e-5]
- list [mse_1](#) = [7.712e-10, 7.627e-10, 7.569e-10, 4.457e-10, 6.931e-10, 8.63e-10]
- list [rmse_1](#) = [2.777e-5, 2.762e-5, 2.751e-5, 2.111e-5, 2.632e-5, 2.9378e-5]
- list [mediana_1](#) = [2.208e-5, 2.207e-5, 2.198e-5, 1.623e-6, 2.014e-5, 2.2815e-5]
- list [methods_2](#) = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']
- list [med_2](#) = [0.0001206, 0.0001204, 0.000121, 0.000107, 0.000119, 0.0001075]
- list [mse_2](#) = [2.149e-8, 2.146e-8, 2.2e-8, 1.585e-8, 2.113e-8, 1.591e-8]
- list [rmse_2](#) = [0.0001466, 0.0001462, 0.0001481, 0.0001259, 0.0001453, 0.0001261]
- list [mediana_2](#) = [9.843e-5, 9.83e-5, 0.00010004, 9.704e-5, 0.00010073, 0.0001008]
- np [x](#) = np.arange(len(methods_1))
- float [width](#) = 0.2
- [ax1](#)
Tworzenie wykresów kolumnowych dla zestawów danych z Tabeli 1 i Tabeli 2.
- [ax2](#)
- [figsize](#)
- ax1 [rects1_1](#) = ax1.bar(x - 1.5*width, med_1, width, label='MED')
- ax1 [rects1_2](#) = ax1.bar(x - 0.5*width, mse_1, width, label='MSE')
- ax1 [rects1_3](#) = ax1.bar(x + 0.5*width, rmse_1, width, label='RMSE')

- `ax1 rects1_4 = ax1.bar(x + 1.5*width, mediana_1, width, label='Mediana')`
- `ax2 rects2_1 = ax2.bar(x - 1.5*width, med_2, width, label='MED')`
- `ax2 rects2_2 = ax2.bar(x - 0.5*width, mse_2, width, label='MSE')`
- `ax2 rects2_3 = ax2.bar(x + 0.5*width, rmse_2, width, label='RMSE')`
- `ax2 rects2_4 = ax2.bar(x + 1.5*width, mediana_2, width, label='Mediana')`
- `list methods_3 = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']`
- `list med_3 = [2.7245e-5, 2.5288e-5, 2.5782e-5, 1.3462e-5, 2.5942e-5, 3.0398e-5]`
- `list mse_3 = [8.8167e-10, 7.8341e-10, 7.7622e-10, 2.3869e-10, 8.1655e-10, 1.1928e-9]`
- `list rmse_3 = [2.9692e-5, 2.7989e-5, 2.786e-5, 1.5449e-5, 2.8575e-5, 3.4536e-5]`
- `list mediana_3 = [2.7446e-5, 2.661e-5, 2.7342e-5, 1.4534e-5, 2.3687e-5, 2.8072e-5]`
- `list methods_4 = ['SMA', 'LOWESS', 'FMA', 'Filtr Kalmana', 'WMA', 'Dane oryginalne']`
- `list med_4 = [0.0001258, 0.0001174, 0.0001245, 6.505e-5, 0.0001299, 5.986e-5]`
- `list mse_4 = [2.2271e-8, 1.9782e-8, 2.2558e-8, 6.7572e-9, 2.4743e-8, 4.3091e-9]`
- `list rmse_4 = [0.0001492, 0.0001406, 0.0001501, 8.2202e-5, 0.0001573, 6.5647e-5]`
- `list mediana_4 = [0.0001086, 0.000105, 0.0001074, 5.0468e-5, 0.0001117, 5.7155e-5]`
- `ax1 rects3_1 = ax1.bar(x - 1.5*width, med_3, width, label='MED')`
- `ax1 rects3_2 = ax1.bar(x - 0.5*width, mse_3, width, label='MSE')`
- `ax1 rects3_3 = ax1.bar(x + 0.5*width, rmse_3, width, label='RMSE')`
- `ax1 rects3_4 = ax1.bar(x + 1.5*width, mediana_3, width, label='Mediana')`
- `ax2 rects4_1 = ax2.bar(x - 1.5*width, med_4, width, label='MED')`
- `ax2 rects4_2 = ax2.bar(x - 0.5*width, mse_4, width, label='MSE')`
- `ax2 rects4_3 = ax2.bar(x + 0.5*width, rmse_4, width, label='RMSE')`
- `ax2 rects4_4 = ax2.bar(x + 1.5*width, mediana_4, width, label='Mediana')`

4.1.1 Detailed Description

Skrypt do wizualizacji danych błędów dla różnych metod wygładzania w formie wykresów kolumnowych.

Skrypt wykorzystuje dane z tabeli, tworzy wykresy kolumnowe dla różnych parametrów błędów (MED, MSE, RMSE, Mediana) oraz wizualizuje je na wykresach w skali logarytmicznej. Wykresy przedstawiają porównanie metod dla różnych zestawów danych.

4.2 main.py File Reference

Główny skrypt obliczający błędy i generujący wykresy.

Namespaces

- namespace `main`

Functions

- def `euclidean_distance` (p1, p2)
Oblicza odległość euklidesową pomiędzy dwoma punktami.
- def `haversine_distance` (coord1, coord2)
Oblicza odległość Haversine pomiędzy dwoma współrzędnymi geograficznymi.
- def `interpolate_reference` (data, num_points)
Interpoluje dane referencyjne, aby uzyskać określoną liczbę punktów.
- def `group_data` (data)
Grupuje dane wygładzone według grup.
- def `group_data_by_original` (original_data, smoothed_data)
Grupuje dane oryginalne na podstawie grup z danych wygładzonych.
- def `calculate_groups_errors` (grouped_smoothed_data, grouped_original_data, ref_data)
Oblicza błędy dla grup danych wygładzonych i oryginalnych w odniesieniu do danych referencyjnych.
- def `find_closest_point` (target_point, reference_points)
Znajduje najbliższy punkt do zadanego punktu docelowego wśród punktów referencyjnych.
- def `calculate_mean_euclidean_error` (smoothed_data, ref_data)
Oblicza średni błąd euklidesowy między danymi wygładzonymi a referencyjnymi.
- def `calculate_mse` (smoothed_data, ref_data)
Oblicza średni błąd kwadratowy (MSE).
- def `calculate_rmse` (smoothed_data, ref_data)
Oblicza pierwiastek z średniego błędu kwadratowego (RMSE).

Variables

- str `config_path` = 'config.yaml'
- yaml `config` = yaml.safe_load(config_file)
- yaml `data_files` = config['data_files']
- list `original_data` = []
- list `smoothed_data` = []
- list `ref_data` = []
- os `full_path` = os.path.join(os.getcwd(), file_path)
- re `match` = re.search(r'Point (\d+):Latitude:([\d.]+)\sLongitude:([\d.]+)', line)
- int `point_index` = int(match.group(1))
- float `lat` = float(match.group(2))
- float `lon` = float(match.group(3))
- float `mae` = float(match.group(4))
- int `group` = int(match.group(5))
- int `time` = int(match.group(6))
- np `interpolated_ref_data` = np.array(interpolate_reference(ref_data, len(smoothed_data)))
- list `smoothed_coords` = smoothed_data[:, 1:3]
- def `grouped_smoothed_data` = group_data(smoothed_data)
- def `grouped_original_data` = group_data_by_original(original_data, smoothed_data)
- `mean_error`
- `mean_error3`
- `mse`
- `mse2`
- `mae2`
- `rmsd`
- `rmsd2`
- `median`
- `median2`

- `errors`
- `mean_error_no_groups`
- `errors2`
- `def mse_no_groups = calculate_mse(smoothed_data, interpolated_ref_data)`
- `def rmse = calculate_rmse(smoothed_data, interpolated_ref_data)`
- `def mean_error_no_groups2 = calculate_mean_euclidean_error(original_data, interpolated_ref_data)`
- `def mse_no_groups2 = calculate_mse(original_data, interpolated_ref_data)`
- `def rmse_no_groups2 = calculate_rmse(original_data, interpolated_ref_data)`
- `np interpolated_ref_data_np = np.array(interpolated_ref_data)`
- `list group_colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black', 'orange', 'purple', 'brown']`
- `fig`
- `ax`
- `figsize`
- `np points = np.array(points)`
- `label`
- `color`
- `bbox_to_anchor`
- `loc`
- `axs`
- `float max_distance = 0.00018`
- `list original_path = original_data[:, 1:3]`
- `list smoothed_path = smoothed_data[:, 1:3]`
- `np ref_path = interpolated_ref_data`
- `c`
- `handles`
- `axs hb = axs[1].hexbin(smoothed_data[:, 1], smoothed_data[:, 2], gridsize=50, cmap='inferno')`

4.2.1 Detailed Description

Główny skrypt obliczający błędy i generujący wykresy.

Główny skrypt odpowiedzialny za wczytanie danych, przetworzenie ich i wygenerowanie wszystkich wykresów, a także obliczenie wartości błędów.

Index

- autolabel
 - kolumnowe, 6
- ax
 - main, 18
- ax1
 - kolumnowe, 6
- ax2
 - kolumnowe, 6
- axs
 - main, 18
- bbox_to_anchor
 - main, 18
- c
 - main, 18
- calculate_groups_errors
 - main, 13
- calculate_mean_euclidean_error
 - main, 14
- calculate_mse
 - main, 14
- calculate_rmse
 - main, 14
- color
 - main, 19
- config
 - main, 19
- config_path
 - main, 19
- data_files
 - main, 19
- errors
 - main, 19
- errors2
 - main, 19
- euclidean_distance
 - main, 16
- fig
 - main, 19
- figsize
 - kolumnowe, 7
 - main, 19
- find_closest_point
 - main, 16
- full_path
 - main, 20
- group
 - main, 20
- group_colors
 - main, 20
- group_data
 - main, 16
- group_data_by_original
 - main, 17
- grouped_original_data
 - main, 20
- grouped_smoothed_data
 - main, 20
- handles
 - main, 20
- haversine_distance
 - main, 17
- hb
 - main, 20
- interpolate_reference
 - main, 18
- interpolated_ref_data
 - main, 21
- interpolated_ref_data_np
 - main, 21
- kolumnowe, 5
 - autolabel, 6
 - ax1, 6
 - ax2, 6
 - figsize, 7
 - med_1, 7
 - med_2, 7
 - med_3, 7
 - med_4, 7
 - mediana_1, 7
 - mediana_2, 7
 - mediana_3, 7
 - mediana_4, 8
 - methods_1, 8
 - methods_2, 8
 - methods_3, 8
 - methods_4, 8
 - mse_1, 8
 - mse_2, 8
 - mse_3, 8
 - mse_4, 9
 - rects1_1, 9
 - rects1_2, 9

- rects1_3, 9
- rects1_4, 9
- rects2_1, 9
- rects2_2, 9
- rects2_3, 9
- rects2_4, 10
- rects3_1, 10
- rects3_2, 10
- rects3_3, 10
- rects3_4, 10
- rects4_1, 10
- rects4_2, 10
- rects4_3, 10
- rects4_4, 11
- rmse_1, 11
- rmse_2, 11
- rmse_3, 11
- rmse_4, 11
- width, 11
- x, 11
- kolumnowe.py, 27
- label
 - main, 21
- lat
 - main, 21
- loc
 - main, 21
- lon
 - main, 21
- mae
 - main, 21
- mae2
 - main, 21
- main, 12
 - ax, 18
 - axs, 18
 - bbox_to_anchor, 18
 - c, 18
 - calculate_groups_errors, 13
 - calculate_mean_euclidean_error, 14
 - calculate_mse, 14
 - calculate_rmse, 14
 - color, 19
 - config, 19
 - config_path, 19
 - data_files, 19
 - errors, 19
 - errors2, 19
 - euclidean_distance, 16
 - fig, 19
 - figsize, 19
 - find_closest_point, 16
 - full_path, 20
 - group, 20
 - group_colors, 20
 - group_data, 16
 - group_data_by_original, 17
 - grouped_original_data, 20
 - grouped_smoothed_data, 20
 - handles, 20
 - haversine_distance, 17
 - hb, 20
 - interpolate_reference, 18
 - interpolated_ref_data, 21
 - interpolated_ref_data_np, 21
 - label, 21
 - lat, 21
 - loc, 21
 - lon, 21
 - mae, 21
 - mae2, 21
 - match, 22
 - max_distance, 22
 - mean_error, 22
 - mean_error3, 22
 - mean_error_no_groups, 22
 - mean_error_no_groups2, 22
 - median, 22
 - median2, 22
 - mse, 23
 - mse2, 23
 - mse_no_groups, 23
 - mse_no_groups2, 23
 - original_data, 23
 - original_path, 23
 - point_index, 23
 - points, 23
 - ref_data, 24
 - ref_path, 24
 - rmsd, 24
 - rmsd2, 24
 - rmse, 24
 - rmse_no_groups2, 24
 - smoothed_coords, 24
 - smoothed_data, 24
 - smoothed_path, 25
 - time, 25
- main.py, 28
- match
 - main, 22
- max_distance
 - main, 22
- mean_error
 - main, 22
- mean_error3
 - main, 22
- mean_error_no_groups
 - main, 22
- mean_error_no_groups2
 - main, 22
- med_1
 - kolumnowe, 7
- med_2
 - kolumnowe, 7
- med_3

- kolumnowe, [7](#)
- med_4
 - kolumnowe, [7](#)
- median
 - main, [22](#)
- median2
 - main, [22](#)
- mediana_1
 - kolumnowe, [7](#)
- mediana_2
 - kolumnowe, [7](#)
- mediana_3
 - kolumnowe, [7](#)
- mediana_4
 - kolumnowe, [8](#)
- methods_1
 - kolumnowe, [8](#)
- methods_2
 - kolumnowe, [8](#)
- methods_3
 - kolumnowe, [8](#)
- methods_4
 - kolumnowe, [8](#)
- mse
 - main, [23](#)
- mse2
 - main, [23](#)
- mse_1
 - kolumnowe, [8](#)
- mse_2
 - kolumnowe, [8](#)
- mse_3
 - kolumnowe, [8](#)
- mse_4
 - kolumnowe, [9](#)
- mse_no_groups
 - main, [23](#)
- mse_no_groups2
 - main, [23](#)
- original_data
 - main, [23](#)
- original_path
 - main, [23](#)
- point_index
 - main, [23](#)
- points
 - main, [23](#)
- rects1_1
 - kolumnowe, [9](#)
- rects1_2
 - kolumnowe, [9](#)
- rects1_3
 - kolumnowe, [9](#)
- rects1_4
 - kolumnowe, [9](#)
- rects2_1
 - kolumnowe, [9](#)
- rects2_2
 - kolumnowe, [9](#)
- rects2_3
 - kolumnowe, [9](#)
- rects2_4
 - kolumnowe, [10](#)
- rects3_1
 - kolumnowe, [10](#)
- rects3_2
 - kolumnowe, [10](#)
- rects3_3
 - kolumnowe, [10](#)
- rects3_4
 - kolumnowe, [10](#)
- rects4_1
 - kolumnowe, [10](#)
- rects4_2
 - kolumnowe, [10](#)
- rects4_3
 - kolumnowe, [10](#)
- rects4_4
 - kolumnowe, [11](#)
- ref_data
 - main, [24](#)
- ref_path
 - main, [24](#)
- rmsd
 - main, [24](#)
- rmsd2
 - main, [24](#)
- rmse
 - main, [24](#)
- rmse_1
 - kolumnowe, [11](#)
- rmse_2
 - kolumnowe, [11](#)
- rmse_3
 - kolumnowe, [11](#)
- rmse_4
 - kolumnowe, [11](#)
- rmse_no_groups2
 - main, [24](#)
- smoothed_coords
 - main, [24](#)
- smoothed_data
 - main, [24](#)
- smoothed_path
 - main, [25](#)
- time
 - main, [25](#)
- width
 - kolumnowe, [11](#)
- x
 - kolumnowe, [11](#)