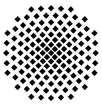

Universität Stuttgart

Institut für Angewandte Analysis und Numerische Simulation

Metrik für sEMG-Daten

metrik_simultaneous_occurence.py

Lisa Dollmann, Maximilian Hack, Lilian Cathérine Lepère

Autor Lisa Dollmann
Maximilian Hack
Lilian Cathérine Lepère

Betreuer Aaron Krämer

Datum Stuttgart
27. Dezember 2021

Das ist die Dokumentation der Metrik *metrik_simultaneous_occurence.py*. Die Metrik misst wie häufig andere Geschwindigkeit gleichzeitig mit einer zu prüfenden Geschwindigkeit aus einem Datensatz von *data_creator_*002.*py* in einem Scenario auftreten. Augrund der Struktur des *data_creator_*001.*py* ist die Metrik für diese Datensätze uninteressant. Für den *data_creator_*002.*py* veranschaulicht die Metrik, dass die Geschwindigkeiten ähnlich oft zusammen auftreten.
In der Dokumentation werden Installation und Nutzung beschrieben, die enthaltenen Features und weitere Ideen aufgelistet, die Ergebnisse kurz interpretiert, relevante Programmausgaben dargestellt und erläutert.

Inhaltsverzeichnis

1 Installation und Nutzung

1.1 Module und Elemente

Das Programm benötigt die bereits an anderen Stellen des Projekts benötigten Module *pickle*, *numpy*, *matplotlib.pyplot* und *sys*. Außerdem wird das Modul *itertools* genutzt, um ein mehrdimensionales Array eindimensional zu machen.

1.2 Parameter

Es gibt vier Eingabeparameter. Der Erste ist entweder x des entsprechenden $data\_creator\_00x.py$ oder ein ganzer Dateiname. die anderen Drei sind die n_speeds, $n\_waveLengths$ und $pruef_speed$. Deren Nutzung wird im folgenden Abschnitt näher erklärt.

1.3 Nutzung

Beispielbilder zur Veranschaulichung der Nutzung sind in Kapitel 3 eingebunden.
Um die Metrik *metrik_speeds.py* zu nutzen hat man zwei Optionen. Man muss entweder den Dateinamen eines Datensatzes übergeben oder dessen 'Kennzahlen'. Will man beispielsweise die Datei $DATA/creator\_001/RawData\_3\_velocities\_\_\_\_15\_waveLengths.pickle$ mit der Metrik analysieren, übergibt man dem Terminal entweder diesen Namen, also *python metrik_speeds.py* $DATA/creator\_001/RawData\_3\_velocities\_\_\_\_15\_waveLengths.pickle$ 10 oder *python metrik_speeds.py* 1 3 15 10. Letzteres ist eine abkürzende Schreibweise,die im folgenden kurz erläutert wird:
Erzeugt man mit $data\_creator\_00x.py$ einen Datensatz, so ist der Dateiname immer nach dem selben Muster aufgebaut. Die Dateinamen sind vom Typ $DATA/creator\_001/RawData\_ + str(n\_speeds) + \_velocities\_\_\_\_ + str(n\_waveLengths) + \_waveLengths.pickle$ (bei $data\_creator\_002.py$ kommt im Dateinamen noch der $grow\_factor$ hinzu; die Metrik nutzt aber immer den Default Wert). Um das Aufrufen eines Datensatzes aus einer *pickle*-Datei zu Erleichtern kann man also auch die 'Kennzahlen' eingeben. Das sind die Zahl x des entsprechenden $data\_creator\_00x.py$, die n_speeds und $n\_waveLengths$. Ein Datensatz mit diesen 'Kennzahlen' muss bereits exisiteren!
Um die Metrik $metrik_simultaneous_occurence.py$ zu nutzen, muss man als vierten Eingabeparameter den sogenannten $pruef_speed$ übergeben. Im Beispiel ist $pruef_speed = 10$. Der $pruef_speed$ ist diejenige Geschwindigkeit für die man untersuchen möchte, wie oft die jeweils anderen Geschwindigkeiten im entsprechenden Datensatz zusammen mit ihr, dem $pruef_speed$, auftreten.
Hat man das Programm ausgeführt zeigt es ein Histogramm, das darstellt, wie häufig die Geschwindigkeiten in den Scenarien des Datensatzes gleichzeitig vorkommen.

2 Enthaltene Features und deren Ausgaben

2.1 Dateiname aus 'Kennzahlen'

Wie in 1.3 bereits angesprochen, werden Dateien entweder über deren Name oder über sogenannte Kennzahlen aufgerufen. Das funktioniert mithilfe des Moduls *sys*. Das Meiste zu diesem Vorgehen ist in 1.3 bereits erklärt. An dieser Stelle soll noch auf zwei Punkte eingegangen werden. Das Programm soll anhand der eingegebenen *sys*-Argumente herausfinden, ob ein Dateiname oder nur die Kennzahlen der Datei übergeben wurden. Dazu wird mittels einer if-Abfrage überprüft wie lange der *string* des ersten *sys*-Arguments ist. Da im Moment mehr als 1000 *data_creators* nicht absehbar sind, sollte ein *string*, der mehr als 4 Zeichen beinhaltet ein Dateiname sein. Der zweite Punkt bezieht sich auf die Methode, die aus den Kennzahlen den Dateinamen zusammenfügt. Der Dateiname für Daten aus dem *data_creator_*002*.py* enthält den Zusatz $grow\_factor\_\_\_4$. Dies muss also im Fall x=2 geprüft und angepasst werden. Das wurde mithilfe einer if-Abfrage umgesetzt.
Daraus ergeben sich drei @ToDos:

2.1.1 @ToDo: $str(sys.argv[5])$

Man kann ein fünftes Argument $str(sys.argv[5])$ übergeben um den $grow\_factor$ ebenfalls variabel zu halten. Eventuell kann man diesem *sys*-Argument schlicht den Default-Wert 4 zuordnen, damit lässt man dem Nutzer die Entscheidung.

2.1.2 @ToDo: if-Abfrage

Eventuell kann man die erste if-Abfrage, ob ein Dateiname oder nur dessen Kennzahlen übergeben wurden, kompakter fassen.
Man könnte beispielsweise nur den Dateinamen als Ganzes übergeben und die Option mit den Kennzahlen streichen. Wenn es mehr als die bisherigen beiden *data_creator_*00x*.py* gibt, können die Kennzahlen aufgrund möglicher Zusätze im Dateinamen, wie das beim $grow\_factor$ in *data_creator_*002*.py* der Fall ist, kompliziert werden. Man müsste entweder bei all diesen Zusätzen auf einen Default-Wert hoffen, oder 2.1.1 für all diese Zusätze umsetzen. Das ist entweder sehr steif oder ab spätestens 7 *sys*-Argumenten sehr verwirrend und Tippfehler-anfällig.
Bisher erleichtert die Kennzahlen-Schreibweise aber die Eingabe und bleibt vorerst erhalten.

2.1.3 @ToDo: *creator*_001 uninteressant

Wie in der Zusammenfassung bereits angesprochen, sind die Daten vom *creator_*001 uninterssant. Man könnte die Metrik also ausschließlich für die Daten aus dem *creator_*002 nutzen un das bereits in diesem Feature/ Codeblock einbauen. Dann kann man die Metrik aber auch nicht für andere *creator_*00x nutzen, die in Zukunft noch geschrieben werden. Ob die *metrik_simultaneous_occurence.py* mit den Daten eines solchen

creator_00x umgehen kann, ist ohnehin fragwürdig, da die erzeugten Daten eine anderes Format haben können. Bei der Szenarien Auswahl (2.3) entsteht dann eine Fehlermeldung.

2.2 Pickle

Nachdem der Dateiname aus den *sys*-Argumenten erstellt wurde, wird die Datei mithilfe des *pickle*-Moduls entpickelt. Das läuft standardmäßig ab.

2.3 Szenarien Auswahl

Für das Histogramm sind nur diejenigen Scenarien interessant in denen der $pruef\_speed$ tatsächlich vorkommt. Das wird in einer Schleife vorsortiert:
Dafür wird zunächst eine leere Liste $number\_simultaneous\_occurance$ angelegt. Dann wird mit einer for-Schleife über die Länge des Datensatzes $len(daten)$ iteriert. In jedem Schritt i werden dann die Geschwindigkeiten, die in dem Scenario $daten[i]$ auftreten, extrahiert. Das funktioniert mittels $daten[i].meta\_.get\_vecs()[2]$. Über eine if-Schleife wird dann geprüft, ob $pruef\_speed$ unter diesen Geschwindigkeiten ist. Wenn ja, wird das Array mit diesen Geschwindigkeiten an die Liste $number\_simultaneous\_occurance$ gehängt.
@ToDo: Man könnte den $pruef\_speed$ selbst erst aus dem Array entfernen und nur die anderen Geschwindigkeiten anhängen. Der Vorteil wäre, dass die y-Achse eine geringere Spanne hätte und somit die Häufigkeit der anderen Geschwindigkeiten esser lesbar ist. Allerdings ist die absolute Anzahl, in wie vielen Szenarien der $pruef\_speed$ auftritt eine Vergleichszahl. Sie wird vorerst beibehalten.
Die fertige Liste wird noch eindimensional gemacht, da das das Erstellen des Histogramms erleichtert.

2.4 Das Histogramm

Das Histogramm bildet dann die Häufigtkeit ab, wie oft andere Geschwindigkeiten mit dem $pruef\_speed$ gleichzeitig in einem Szenario auftreten im eingegebenen Datensatz. Da die genaue Häufigkeit auf dem Bild schwer abzulesen ist, wird die Häufigkeit der jeweiligen Geschwindigkeit zusätzlich in Zahlenform in der Konsole ausgegeben.

3 Interpretation

3.1 $data\_creator\_001$

Die Metrik ergibt keinen Sinn für den *data_creator_001*, da in jedem Szenario genau eine Geschwindigkeit auftritt. Deswegen wurden diesbezüglich keine Fehler behoben (@ToDo?).

3.2 *data_creator_*002

Für Daten aus dem *data_creator_*002 wird ersichtlich, dass Geschwindigkeiten ähnlich häufig in einem gemeinsamen Szenario vorkommen (vgl. 1). Das könnte an dem Prozess liegen, der beim Erstellen der Daten mit dem *data_creator_*002 abläuft. Das Zusammenlegen der Ausschnitte aus den Arrays der einzel Geschwindigkeiten könnte dafür sorgen.
Interessant wäre noch, wie viele Wellen gleichzeitig in einem Scenario auftreten. Das könnte man mit echten sEMG-Daten vergleichen.

4 Relevante Programmausgaben

In diesem Abschnitt werden im Text referenzierte Programmausgaben veranschaulicht. Es werden von der *metrik_simultaneous_occurence.py* erzeugt Bilder aus Daten, die mit dem *data_creator*_002 erzeugt wurden, gezeigt.

4.1 Histogramm mit 3 Geschwindigkeiten und 20 Wellenlängen zu $pruef\_speed = 1$

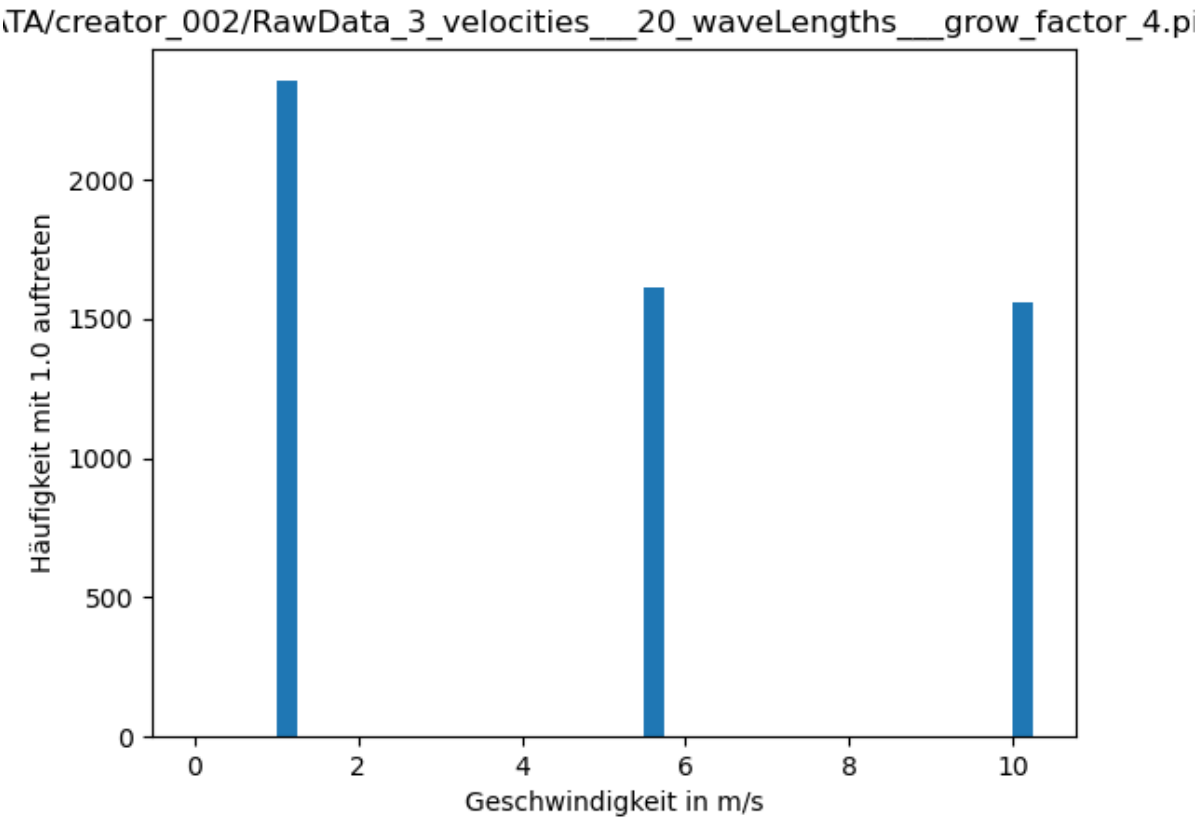

Abbildung 1: Das Histogramm eines Datensatzes, der mithilfe des *data_creator*_002 erzeugt wurde. Dieser Datensatz enthält zwei von 1.0 verschiedene Geschwindigkeiten. Sie kommen ähnlich oft mit 1.0 vor (vgl. 3.2).

4.2 Terminalausgabe bei 3 Geschwindigkeiten und 20 Wellenlängen bei $pruef\_speed = 1$

```
(base) PS D:\sandbox\github\aarons> python .\metrik_simultaneous_occurance.py 2 3 20 2
[Geschwindigkeit,Häufigkeit mit 2.0 auftreten]
[]
(base) PS D:\sandbox\github\aarons> python .\metrik_simultaneous_occurance.py 2 3 20 1
[Geschwindigkeit,Häufigkeit mit 1.0 auftreten]
[[1.0, 2353.0], [5.5, 1614.0], [10.0, 1556.0]]
(base) PS D:\sandbox\github\aarons> python .\metrik_simultaneous_occurance.py 2 3 20 10
[Geschwindigkeit,Häufigkeit mit 10.0 auftreten]
[[1.0, 1556.0], [5.5, 1574.0], [10.0, 2309.0]]
(base) PS D:\sandbox\github\aarons> python .\metrik_simultaneous_occurance.py 2 3 20 10
[Geschwindigkeit,Häufigkeit mit 10.0 auftreten]
[[1.0, 1556.0], [5.5, 1574.0], [10.0, 2309.0]]
(base) PS D:\sandbox\github\aarons> python .\metrik_simultaneous_occurance.py 2 3 20 5.5
[Geschwindigkeit,Häufigkeit mit 5.5 auftreten]
[[1.0, 1614.0], [5.5, 2363.0], [10.0, 1574.0]]
```

Abbildung 2: Die Ausgabe im Terminal unter anderem zum Histogramm aus 1. Man erkennt nach der zweiten Eingabe, dass Wellen mit der Geschwindigkeit 1.0 m/s im gesamten Datensatz 2353 mal vorkommen, die Wellen mit 5.5 m/s kommen 1614 mal in den gleichen Szenarien vor und die mit 10.0 m/s 1565 mal.
Außerdem sieht man an der ersten Eingabe, dass die Metrik mit nicht auftretenden Geschwindigkeiten ebenfalls umgehen kann. Sie gibt eine leere Liste aus.

4.3 Histogramm mit 30 Geschwindigkeiten und 30 Wellenlängen

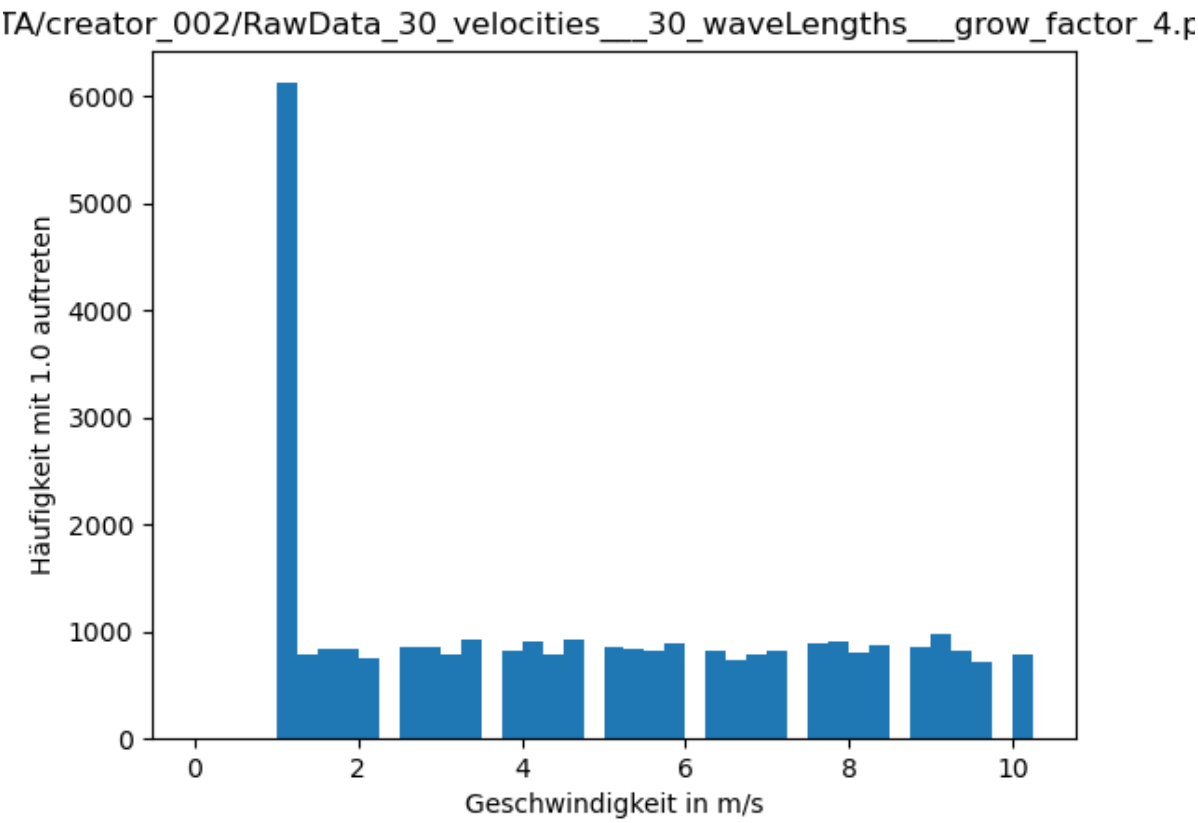

Abbildung 3: Das Histogramm eines Datensatzes, der mithilfe des *data_creator_*002 erzeugt wurde. Dieser Datensatz enthält dreißig verschiedene Geschwindigkeiten. Die y-Achse läuft über 6000, allerdings nur wegen des $pruef\_speed = 1.0$. Die andern Geschwindigkeiten treten wieder ähnlich oft in einem Szenario mit 1.0 auf (vgl. 2.3).

4.4 Terminalausgabe bei 30 Geschwindigkeiten und 30 Wellenlängen

```
(base) PS D:\sandbox\github\aarons> python .\metrik_simultaneous_occurance.py 2 30 30 1
[Geschwindigkeit,Häufigkeit mit 1.0 auftreten]
[[1.0, 6123.0], [1.25, 788.0], [1.5, 841.0], [1.75, 836.0], [2.0, 744.0], [2.5, 853.0], [2.75, 856.0], [3.0, 783.0], [3.
25, 923.0], [3.75, 824.0], [4.0, 911.0], [4.25, 778.0], [4.5, 921.0], [5.0, 861.0], [5.25, 836.0], [5.5, 822.0], [5.75,
899.0], [6.25, 821.0], [6.5, 729.0], [6.75, 790.0], [7.0, 814.0], [7.5, 884.0], [7.75, 908.0], [8.0, 803.0], [8.25, 875.
0], [8.75, 861.0], [9.0, 970.0], [9.25, 816.0], [9.5, 711.0], [10.0, 786.0]]
```

Abbildung 4: Die Ausgabe im Terminal zum Histogramm aus 3.