Python – Übung 11

1 Scatter Plot

In der Natur gibt es viele Beispiele, wo der Goldene Schnitt

$$\Phi = \frac{1+\sqrt{5}}{2} \approx 1.618 \tag{1}$$

vorkommt. Zudem gibt es den Goldenen Winkel, g. Man erhält den Goldenen Winkel indem der Vollwinkel durch den Goldenen Schnitt geteilt wird und zum Vollwinkel ergänzt wird:

$$g = 2\pi - \frac{2\pi}{\Phi} \approx 2.39996 \approx 137.51^\circ \tag{2}$$

Der Goldene Winkel kommt ebenso häufig in der Natur vor, wie z.B. im Blütenstand einer Sonnenblume, wie dies in Abb. 1 dargestellt wird. In der Abbildung gibt es total N Punkte, wobei die xy-Koordinaten des n-ten Punktes wie folgt definiert sind:

$$\boldsymbol{p}_n = \begin{bmatrix} x \\ y \end{bmatrix} = \sqrt{N-n} \begin{bmatrix} \cos(ng) \\ \sin(ng) \end{bmatrix}, \ \forall\, n = 0, 1, \ldots, N-1\ . \tag{3}$$

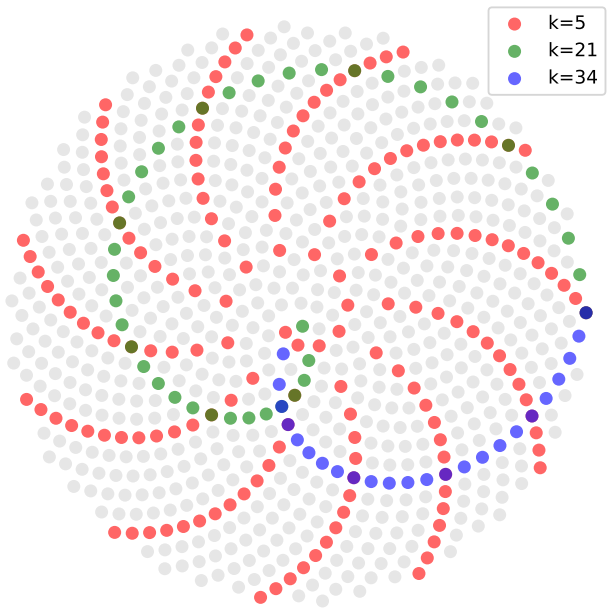

Abbildung 1: Scatterplot eines Sonnenblumen-Blütenstands.

☞ Schreiben Sie ein Python-Programm, welches den Scatterplot mit $N = 800$ Punkte erzeugt, ähnlich wie in Abb. 1 dargestellt. Färben Sie zudem alle Punkte ein, deren Nummer n ein Vielfaches von k ist, d.h. $n = ki\,|_{n<N},\ \forall\, i \in \mathbb{N}_0,\ \forall\, k = 5, 21, 34$.

Hinweis: Benutzen Sie die `plt.scatter()`-Funktion[1] mit dem `c`-Argument, um die Farbe (`c=color`) der Punkte zu spezifizieren und mit dem `alpha`-Argument, um die Transparenz der Punkte zu steuern.

[1] `https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html`

2 Elektrisches Feld

Das elektrische Feld, $\vec{E}$, einer Punktladung Q an einem bestimmten Ort (gegeben durch den Richtungsvektor $\vec{R}$) ist wie folgt definiert:

$$\vec{E}\left(Q, \varepsilon_r, \vec{R}\right) = \frac{Q\vec{R}}{4\pi\varepsilon_0\varepsilon_r R^3} \,, \tag{4}$$

wobei Q die Ladung in Amperesekunden (As) ist, $\varepsilon_0 = 8.854 \times 10^{-12}\,\mathrm{As/Vm}$ die elektrische Feldkonstante ist und $\varepsilon_r = 1$ (im Vakuum) die relative Permittivität ist.

Zwei Punktladungen

$$Q_1 = 10^{-9}\,\mathrm{As} \quad \text{und} \tag{5}$$
$$Q_2 = -10^{-9}\,\mathrm{As} \tag{6}$$

werden an den Positionen

$$\vec{P}_1 = \begin{bmatrix} P_{1,x} \\ P_{1,y} \\ P_{1,z} \end{bmatrix} = \begin{bmatrix} 1\,\mathrm{m} \\ 0\,\mathrm{m} \\ -0.5\,\mathrm{m} \end{bmatrix} \quad \text{und} \tag{7}$$

$$\vec{P}_2 = \begin{bmatrix} P_{2,x} \\ P_{2,y} \\ P_{2,z} \end{bmatrix} = \begin{bmatrix} -1\,\mathrm{m} \\ 0\,\mathrm{m} \\ -0.5\,\mathrm{m} \end{bmatrix} \tag{8}$$

platziert. Nun soll das elektrische Feld $\vec{E}_{\text{total}}$, als Summe der beiden Feldern (wegen Q_1 und Q_2), in der xy-Ebene bei $z = 0\,\mathrm{m}$ berechnet werden:

$$\vec{E}_{\text{total}} = \vec{E}_1 + \vec{E}_2 \,. \tag{9}$$

Die einzelnen Felder werden wie folgt berechnet:

$$\vec{E}_1 = \begin{bmatrix} E_{1,x} \\ E_{1,y} \\ E_{1,z} \end{bmatrix} = \frac{Q_1}{4\pi\varepsilon_0\varepsilon_r} \frac{\vec{R}_1}{\left\|\vec{R}_1\right\|^3} = \frac{Q_1}{4\pi\varepsilon_0\varepsilon_r \left\|\vec{R}_1\right\|^3} \begin{bmatrix} R_{1,x} \\ R_{1,y} \\ R_{1,z} \end{bmatrix} \quad \text{und} \tag{10}$$

$$\vec{E}_2 = \begin{bmatrix} E_{2,x} \\ E_{2,y} \\ E_{2,z} \end{bmatrix} = \frac{Q_2}{4\pi\varepsilon_0\varepsilon_r} \frac{\vec{R}_2}{\left\|\vec{R}_2\right\|^3} = \frac{Q_2}{4\pi\varepsilon_0\varepsilon_r \left\|\vec{R}_2\right\|^3} \begin{bmatrix} R_{2,x} \\ R_{2,y} \\ R_{2,z} \end{bmatrix} \,, \tag{11}$$

wobei $\|.\|$ die Norm bezeichnet (welche mit der `np.linalg.norm()`-Funktion berechnet wird) und die Richtungsvektoren $\vec{R}_1$ und $\vec{R}_2$ wie folgt definiert sind:

$$\vec{R}_1 = \begin{bmatrix} R_{1,x} \\ R_{1,y} \\ R_{1,z} \end{bmatrix} = \vec{P}_E - \vec{P}_1 = \begin{bmatrix} P_{E,x} - P_{1,x} \\ P_{E,y} - P_{1,y} \\ P_{E,z} - P_{1,z} \end{bmatrix} \quad \text{und} \tag{12}$$

$$\vec{R}_2 = \begin{bmatrix} R_{2,x} \\ R_{2,y} \\ R_{2,z} \end{bmatrix} = \vec{P}_E - \vec{P}_2 = \begin{bmatrix} P_{E,x} - P_{2,x} \\ P_{E,y} - P_{2,y} \\ P_{E,z} - P_{2,z} \end{bmatrix} \,. \tag{13}$$

Die Ortsvektoren $\vec{P}_E$ stellen die Punkte in der xy-Ebene dar.

☞ Schreiben Sie ein Programm, welches das elektrische Feld $\vec{E}_{\text{total}}$ für die Punkte in der xy-Ebene berechnet und ähnlich wie in Abb. 2 darstellt.

Hinweise: Erstellen Sie ein dreidimensionales NumPy-Array, d.h. `shape=(m, n, 3)`, für den $\vec{P}_E$-Vektor, indem Sie die `np.meshgrid()`-Funktion[2] und die `np.stack()`[3]-Funktion benutzen. Das NumPy-Array soll ein $m \times n$ Stack von 3D-Ortsvektoren (x, y, z) sein. Somit besitzen dann automatisch auch alle anderen Arrays $(\vec{R}_i, \vec{E}_i)$ die gleiche Form, d.h. `shape=(m, n, 3)`.

Benutzen Sie für die Norm der Richtungsvektoren, $\left\|\vec{R}_i\right\|$, die `np.linalg.norm()`-Funktion mit dem Argument `keepdims=True`, damit das resultierende Array direkt mit dem Array der Richtungsvektoren verrechnet werden kann (siehe Broadcasting).

Erstellen Sie den Konturplot mittels der `plt.contourf()`-Funktion[4] und benutzen Sie die Norm des elektrischen Feldes $\left\|\vec{E}_{\text{total}}\right\|$ als Intensitätswert für das `Z`-Argument.

Zeichnen Sie die Feldlinien mittels der `plt.streamplot()`-Funktion[5] und benutzen Sie die x- und y-Komponenten des elektrischen Feldes $\vec{E}_{\text{total}}$ für die Argumente `u` und `v`.

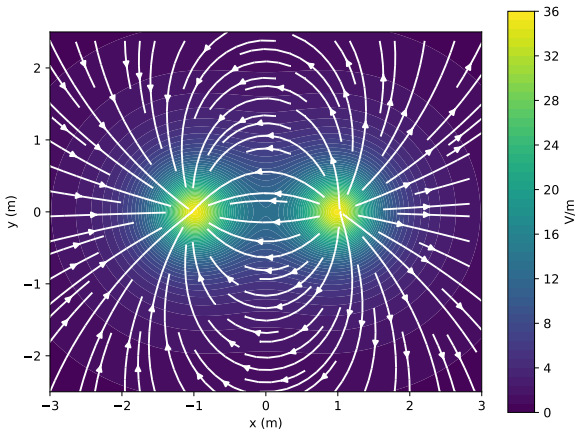

Abbildung 2: Elektrisches Feld in der xy-Ebene.

3 Image Processing

In der Bildverarbeitung werden Bilder für die weitere Verarbeitung in dreidimensionalen Arrays gespeichert. Mit Matplotlib kann ein solches Bild wie folgt geladen und dargestellt werden:

```
import matplotlib.pyplot as plt

B = plt.imread("rapperswil.jpg")
fig, ax = plt.subplots()
ax.imshow(B)
plt.show()
```

[2] https://numpy.org/doc/stable/reference/generated/numpy.meshgrid.html
[3] https://numpy.org/doc/stable/reference/generated/numpy.stack.html
[4] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.contourf.html
[5] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.streamplot.html

Wie in Abb. 3 ersichtlich ist, hat das Array $\boldsymbol{B}$ die Form $(m, n, 3)$, wobei m und n die Anzahl Zeilen und Spalten im Bild sind und in der letzten Dimension jeweils die Intensitäten der drei Farbkanäle (Rot, Grün und Blau) enthalten sind.

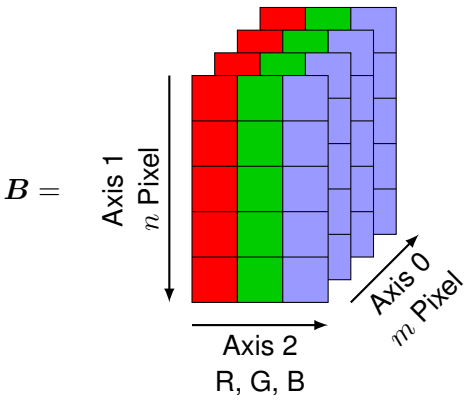

Abbildung 3: Ein farbiges Bild als dreidimensionales NumPy-Array.

Ist ein Bild als NumPy-Array gegeben, lassen sich auch entsprechende Transformationen auf eben jene anwenden. Einige dieser Transformationen werden im folgenden vorgestellt.

Graustufen Ein farbiges Bild kann mit Hilfe der nachfolgend gezeigten Matrix

$$\boldsymbol{T}_{\text{gray}} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \tag{14}$$

in ein graustufiges Bild transformiert werden, indem jeder Pixelvektor

$$\boldsymbol{p} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{15}$$

mit ihr multipliziert wird:

$$\tilde{\boldsymbol{p}} = \boldsymbol{T}_{\text{gray}}\, \boldsymbol{p} \tag{16}$$

$$\begin{bmatrix} \tilde{r} \\ \tilde{g} \\ \tilde{b} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} . \tag{17}$$

Durch die Matrixmultiplikation wird die Summe der Produkte (Skalarprodukt) über der letzten Dimension von $\boldsymbol{T}_{\text{gray}}$ und dem Pixelvektor $\boldsymbol{p}$ gebildet. In diesem Fall wird dreimal der Mittelwert der drei Farbkomponenten gebildet.

Sepia-Effekt Ein farbiges Bild kann auch sehr einfach in ein Bild mit Sepia-Effekt transformiert werden. Die lineare Transformationsmatrix

$$\boldsymbol{T}_{\text{sepia}} = \begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.686 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix} \tag{18}$$

verleiht dem Bild eine rötlich-braune Farbe, die an die monochromen Fotografien des 20. Jahrhunderts erinnert.

3.1 Bild transformieren

- Lesen Sie das Bild `rapperswil.jpg` mittels der `plt.imread()`-Funktion ein. Die Funktion liefert ein NumPy-Array mit dem `uint8`-Datentyp, der die Werte 0 bis 255 abbilden kann. Für die weitere Verarbeitung wird empfohlen den Wertebereich auf 0 bis 1 zu komprimieren, indem die Array-Werte durch 255 geteilt werden. Der Datentyp des resultierenden Arrays wird dadurch automatisch auf `float` eingestellt.
- Berechnen Sie mit Hilfe der oben gezeigten Transformationsmatrizen sowohl die Graustufen- als auch die Sepia-Version des Bildes. Verwenden Sie den @-Operator für die Matrixmultiplikationen. Sie benötigen dazu keine for-Schleifen.

Hinweis: In diesem Fall möchte man am liebsten alle Pixelvektoren im $\boldsymbol{B}$-Array auf einen Schlag mit der Matrix $\boldsymbol{T}$ transformieren. Der @-Operator kann automatisch über einen Stack von Vektoren ($\boldsymbol{B}$) iterieren und jeden Vektor mit einer 2D Matrix ($\boldsymbol{T}$) multiplizieren. Aber dies stimmt nur dann, wenn der Stack von Vektoren links und die Matrix rechts steht. Wenn die Matrix $\boldsymbol{T}$ von links nach rechts verschoben wird, muss die Matrix transponiert werden, denn beim @-Operator werden die Summen der Produkte über der letzten Dimension des linken Arrays ($\boldsymbol{B}$) und der zweitletzten Dimension des rechten Arrays ($\boldsymbol{T}^T$) berechnet.

- Stellen Sie die neuen Bilder in einem Plot dar, ähnlich wie in Abb. 4 gezeigt.

Hinweis: Vor dem Darstellen des Bildes mittels der `plt.imshow()`-Funktion muss sichergestellt werden, dass alle Array-Werte innerhalb des Bereichs von 0 bis 1 sind. Die Werte können z.B. mit der `np.clip()`-Funktion[6] limitiert werden.

Abbildung 4: Originales Bild (links) und die zwei transformierten Bilder.

[6] `https://docs.scipy.org/doc/numpy/reference/generated/numpy.clip.html`

4 3D-Landschaft darstellen

Wie in Abb. 5 dargestellt, besteht die Kartendatei `landschaft.csv` aus mehreren kommagetrennten Zahlenwerten, die in drei Bereiche (rot, blau, grün) eingeteilt werden. Die Zahlen im grünen Bereich stellen die z-Werte (Höhe über Meer) für alle Punkte im Raster dar. In der ersten Zeile (roter Bereich) liegen die entsprechenden x-Werte der Punkte (z.B. $-15\,\mathrm{km}, \ldots, 10\,\mathrm{km}$) und in der ersten Spalte (blauer Bereich) liegen die y-Werte der Punkte (z.B. $-15\,\mathrm{km}, \ldots, 15\,\mathrm{km}$).

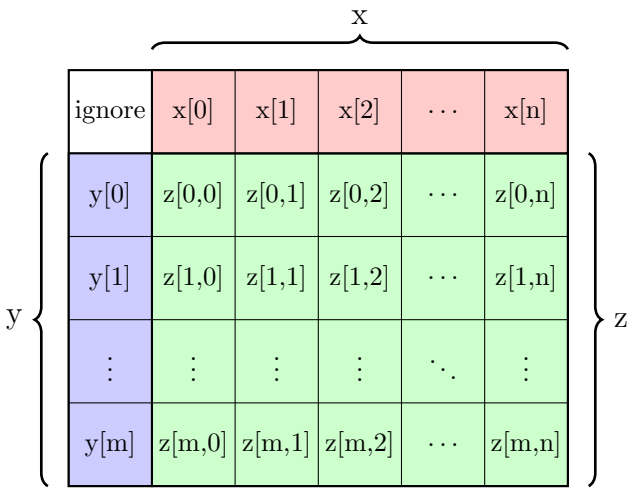

Abbildung 5: Datenstruktur der Kartendatei `landschaft.csv`.

☞ Lesen Sie die Kartendatei `landschaft.csv` ein und extrahieren Sie aus den Daten die drei NumPy-Arrays `x`, `y` und `z`, wie in Abb. 5 spezifiziert.

☞ Plotten Sie die eingelesenen Kartendaten als 3D-Oberfläche, ähnlich wie Abb. 6 dargestellt.

Hinweise: Benutzen Sie die `plot_surface()`-Funktion[7] mit dem Colormap[8] `cmap="terrain"`. Die `plot_surface()`-Funktion erwartet drei 2D-Arrays (für `x`, `y`, `z`) als Argumente. Benutzen Sie die `np.meshgrid()`-Funktion, um aus den beiden 1D-Arrays `x` und `y` je ein 2D-Array zu erstellen.

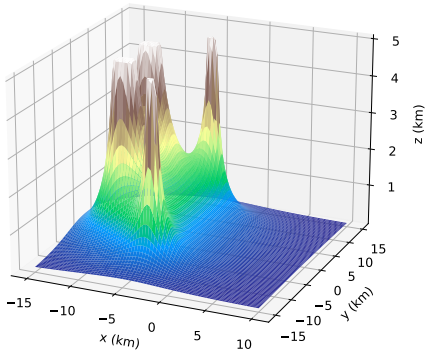

Abbildung 6: 3D-Plot der Landschaft.

[7] `https://matplotlib.org/api/_as_gen/mpl_toolkits.mplot3d.axes3d.Axes3D.html#mpl_toolkits.mplot3d.axes3d.Axes3D.plot_surface`
[8] `https://matplotlib.org/tutorials/colors/colormaps.html`