
# Parallelprojektionen

## Semesterarbeit Teil I

Fernfachhochschule Schweiz

**Semesterarbeit**

im Studiengang BSc Informatik

von

**Arber Osmani**

Bern, 22. September 2021

**Eingereicht bei:** Prof. Dr. Jörg Osterrieder

# Inhaltsverzeichnis

# 1 Einleitung

Die Parallelprojektion gehört zu den Abbildungen von Punktmengen des Anschauungsraumes in eine seiner Ebenen [1]. Eine charakteristische Eigenschaft der Parallelprojektion ist, dass die Projektionsstrahlen parallel zueinander verlaufen. Verlaufen die Projektionsstrahlen zusätzlich parallel zu der Projektionsebene, handelt es sich dann um eine Orthogonalprojektion [4]. Die Parallelprojektion kann auch als eine Variante der Zentralprojektion angesehen werden, bei der das Projektionszentrum unendlich weit weg liegt [2].

Die Projektion eines Punktes im dreidimensionalen Raum, ist also dadurch bestimmt, dass der Projektionsstrahl in eine bestimmte Richtung auf die Ebene zugeht und dort „eintrifft“. Bei der Projektionsebene handelt es sich dann meist um die $xy$-Ebene wobei $z \neq 0$ ist. An dieser Stelle kann man sich leicht überlegen, dass es sich bei dieser Art Projektion, nicht um eine lineare Abbildung handeln kann. Der Nullpunkt im dreidimensionalen Raum, würde ebenso wie jeder andere Punkt, auf die $xy$-Ebene projiziert und damit verschoben werden. Letzteres verletzt ein Kriterium, die eine lineare Abbildung erfüllen muss.

Damit der Bildpunkt $P'$ eines Punktes $P$ mittels einer Matrix berechnet werden kann, müssen homogene Koordinaten verwendet werden.

In dieser Semesterarbeit werden die theoretischen Grundlagen behandelt, mittels derer wir Schrittweise das Problem der Parallelprojektion im dreidimensionalen Raum lösen können. Das Problem wird zunächst im $\mathbb{R}^2$ beschrieben um dann im $\mathbb{R}^3$ zu übergehen. In jedem Fall sind die homogenen Koordinaten anzuwenden und damit Teil der behandelten theoretischen Grundlagen.

Im letzten Kapitel wird der Python-Code, zur visuellen Darstellung der Parallelprojektion im $\mathbb{R}^3$, vorgestellt.

Diese Semesterarbeit, der Python-Code und weitere Ressourcen, stehen online unter der nachfolgenden Adresse zur Verfügung.

`https://linalg.arberosmani.ch`

# 2 Mathematische Grundlagen

## 2.1 Homogene Koordinaten

In Anwendungen der Computergrafik kommt es oft vor, dass ein Objekt bewegt werden soll. Beispielhaft kann an ein Flugzeug gedacht werden, dass sich entlang einer Fluglinie bewegt. Die Bewegung entspricht einer Folge von Bildern, bei denen die Punkte im Bild $n$ auf die Punkte im Bild $n+1$ abgebildet werden. Die Folge der Abbildungen macht dann die ganzheitliche Bewegung, gemäss der beispielhaften Anschauung, aus.

Solche Abbildungen, genannt *Translationen*, stellen jedoch keine linearen Abbildungen dar, da sie den Nullpunkt bewegen. Die Translation verschiebt zwar den Nullpunkt, die Abstände zwischen den Punkten bleiben aber gleich. Solche isometrischen Abbildungen[1], die Abstände zwischen Punkten nicht ändern, können immer als Komposition einer Translation und einer linearen Abbildung dargestellt werden [4]. Ein anderes Beispiel einer nicht-linearen Abbildung, die als eine solche Komposition dargestellt werden kann, ist die Drehung um den Winkel $\alpha$ eines Objektes um einen Punkt $p$, der verschieden vom Ursprung ist. Die Berechnung der Bildpunkte besteht aus drei Operationen. Zuerst werden die Punkte um den Vektor $-\vec{p}$ verschoben, sodass der Drehpunkt im Ursprung landet. Danach kann die Drehmatrix $R_\alpha$ angewendet werden. Als letzter Schritt muss die Verschiebung wieder rückgängig gemacht werden, indem die Punkte um den Vektor $\vec{p}$ verschoben werden. Die drei Operationen können in einer Funktion $f$ verknüpft werden (2.1).

$$f = \tau_p \circ R_\alpha \circ \tau_{-p} \tag{2.1}$$

Der Nachteil dieser Methode gegenüber einer Matrix, wie sie für lineare Abbildungen zur Verfügung steht, ist der erhöhte Rechenaufwand [4]. Es müssen nämlich alle drei Operationen einzeln auf einen Punkt angewendet werden. Nehmen wir ein Beispiel aus dem $\mathbb{R}^2$ für $P = (x, y)$ der um den Drehpunkt $P_r = (a, b)$ um den Winkel $\alpha$ gedreht werden soll. Der Bildpunkt $P'$ berechnete sich dann wie folgt:

$$P' = \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x-a \\ y-b \end{pmatrix} \tag{2.2}$$

[1] Abstandserhaltenden und längentreue Abbildungen.

Die Idee der homogenen Koordinaten ist, die Betrachtung des Problems aus einer höheren Dimension heraus, wodurch die Abbildung dann einer linearen Abbildung entspricht. Eine Translation im $\mathbb{R}^2$ entspricht dann im $\mathbb{R}^3$ einer Scherung [4]. Bei der Scherung handelt es sich um eine lineare Abbildung. Um so eine Matrix für die Berechnung aus (2.2) anzugeben, muss der Punkt $p$ und die Matrix $R_\alpha$ in homogene Koordinaten angegeben werden. Dadurch könnten für die Translationen $\tau_p$ und $\tau_{-p}$ ebenfalls Matrizen angegeben werden. Die Berechnung der Matrix entspricht dann der aus (2.3).

$$M = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -a \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

$$\vec{p'} = M \cdot \vec{p} = M \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

Die $z$ Koordinate von $p'$ können wir dann einfach weglassen[2].

Eine weitere Eigenschaft, die aus der Berechnung mit Hinzunahme homogener Koordinaten folgt, ist, dass ein Punkt durch unendlich viele homogene Koordinaten angegeben werden kann [3].

## 2.2 Parallelprojektion

### 2.2.1 Parallelprojektion im 2D-Raum

Wir nähern uns dem eigentlichen Problem, der Parallelprojektion im 3D-Raum, mit einer ähnlichen Projektion im 2D-Raum. Wenn es im 2D verstanden ist, ist der Übergang im 3D ein Leichtes.

Punkte sollen auf eine zur $x$-Achse parallellaufende Linie projiziert werden. Diese Projektionslinie ist durch ihre Höhe $h$ definiert. Die Punkte in der Ebene werden dabei in eine bestimmten Richtung auf die Projektionslinie projiziert. Die Richtung ist durch den Richtungsvektor $\vec{v}$ definiert. Diese Problemstellung ist sehr ähnlich der vorgegebenen[3].

Das Problem kann auf das Finden des Schnittpunktes zweier Geraden reduziert werden.

[2] Es wäre auch möglich gewesen, die Matrix $M$ so zu generieren, dass bei der Berechnung $M \cdot \vec{p}$, die $z$ Koordinate automatisch rausfliegen würde.

[3] Teil 1 der Semesterarbeit: Parallelprojektionen `https://moodle.ffhs.ch/mod/assign/view.php?id=3891999`

Die erste Gerade $G_h$, ist gegeben durch die Höhe $h$ der zur $x$-Achse parallellaufenden Projektionslinie. $G_h$ ist in (2.4) in der Normalenform definiert.

$$G_h : \langle \vec{n} \ , \ \vec{x} - \overrightarrow{OP} \rangle = \langle \begin{pmatrix} 0 \\ 1 \end{pmatrix} \ , \ \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} 0 \\ h \end{pmatrix} \rangle = 0 \tag{2.4}$$

Die zweite Gerade $G_p$ ist durch den zu projizierenden Punkt $p$ und dem Richtungsvektor $\vec{v}$ bestimmt. $G_p$ ist in (2.5) in der Punkt-Richtungs-Form definiert.

$$G_p : \vec{p} + \lambda \vec{v} \tag{2.5}$$

Die Lösung besteht darin, die Geradengleichungen in Abhängigkeit einer Gegebenen Höhe $h$ und eines gegebenen Richtungsvektor $\vec{v}$, zu lösen. Aus Abschnitt 2.1 wissen wir, dass es möglich ist, die Parallelprojektion in $\mathbb{R}^2$ in der höheren Dimension in $\mathbb{R}^3$ als eine lineare Abbildung anzugeben und damit auch eine Abbildungsmatrix zu finden.

Abbildung 2.1 veranschaulicht die Parallelprojektion in der Ebene. Die Punkte $p_1$ und $p_2$ werden auf der Projektionslinie projiziert. Der Richtungsvektor $\vec{v}$ ist als Pfeil dargestellt und der Projektionsstrahl als gestrichelte Gerade. Die Bildpunkte $p_1'$ und $p_2'$ liegen exakt auf der Projektionslinie und entsprechen dem Schnittpunkten zwischen den Geraden $G_h$ und $G_{p_1}$ bzw. $G_{p_2}$.

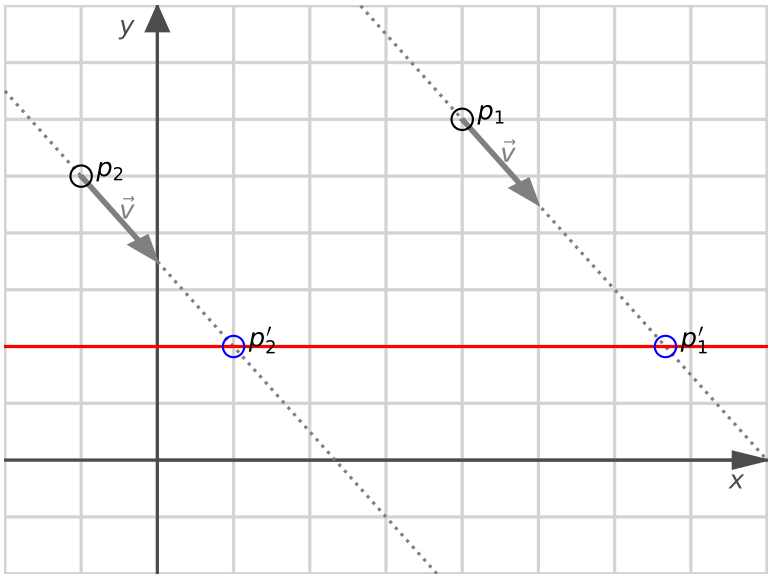


Abbildung 2.1: Parallelprojektion im 2D-Raum

Nachfolgend wird die Projektionsmatrix hergeleitet. Dafür lösen wir nach $\lambda$ auf.

$$\begin{aligned} p' &= \langle \vec{n}, \vec{p} + \lambda\vec{v} - \overrightarrow{OP} \rangle \\ &= \langle \vec{n}, \vec{p} \rangle + \lambda \langle \vec{n}, \vec{v} \rangle - \langle \vec{n}, \overrightarrow{OP} \rangle = 0 \end{aligned}$$

$$\lambda = \frac{\langle \vec{n}, \overrightarrow{OP} \rangle - \langle \vec{n}, \vec{p} \rangle}{\langle \vec{n}, \vec{v} \rangle} = \frac{\langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle}{\langle \vec{n}, \vec{v} \rangle}$$

Der Bildpunkt $p'$ kann jetzt in homogenen Koordinaten wie wie folgt angegeben werden:

$$p' = \left( p_x + \frac{\langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle}{\langle \vec{n}, \vec{v} \rangle} \cdot v_1 \ , \ p_y + \frac{\langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle}{\langle \vec{n}, \vec{v} \rangle} \cdot v_2 \ , \ 1 \right)$$

$$\begin{aligned} \langle \vec{n}, \vec{v} \rangle \cdot p' &= \left( \langle \vec{n}, \vec{v} \rangle \cdot p_x + \langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle \cdot v_1 \ , \ \langle \vec{n}, \vec{v} \rangle \cdot p_y + \langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle \cdot v_2 \ , \ \langle \vec{n}, \vec{v} \rangle \right) \\ &= \left( \langle \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle \cdot p_x + \langle \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ h \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \end{pmatrix} \rangle \cdot v_1 \ , \right. \\ &\quad \langle \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle \cdot p_y + \langle \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ h \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \end{pmatrix} \rangle \cdot v_2 \ , \\ &\quad \left. \langle \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \rangle \right) \\ &= (p_x v_2 + v_1 h - p_y v_1 \ , \ p_y v_2 + v_2 h - p_y v_2 \ , \ v_2) \\ &= (p_x v_2 - p_y v_1 + v_1 h \ , \ v_2 h \ , \ v_2) \end{aligned}$$

Nach der Berechnung in homogenen Koordinaten, lässt sich erkennen, dass dafür eine Matrix angegeben werden kann. Die erste Komponente hängt von $x$, $y$ und einer Konstante ab. Die zweite und dritte Komponente hängen nur von konstanten Werten ab. Die Matrix (2.6) ist die Abbildungsvorschrift für die Parallelprojektion im 2D-Raum. Bei Multiplikation mit (2.7), erhielten wir sogar gleich den Punkt ohne homogene Koordinate.

$$M = \begin{pmatrix} v_2 & -v_1 & v_1 h \\ 0 & 0 & v_2 h \\ 0 & 0 & v_2 \end{pmatrix} \tag{2.6}$$

$$M \cdot \frac{1}{v_2} = \begin{pmatrix} 1 & -v_1/_v 2 & v_1 h / v_2 \\ 0 & 0 & h \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & -v_1/_v 2 & v_1 h / v_2 \\ 0 & 0 & h \end{pmatrix} \tag{2.7}$$

### 2.2.2 Parallelprojektion im 3D-Raum

Die Herleitung der Abbildungsmatrix für den 3D Fall, ist ähnlich dem Vorgehen für den 2D Fall aus dem vorigen Kapitel. Der wesentliche Unterschied besteht darin, dass die Projektion nicht auf einer Projektionslinie sondern auf einer Projektionsebene geschieht. Die Aufgabenstellung beschränkt sich dabei auf die $xy$-Ebene.

Das Problem kann auf das Finden des Schnittpunktes einer Ebene und einer Geraden reduziert werden. Die Ebene $\epsilon_h$, verläuft parallel zur $xy$ Ebene in der Höhe $h$, wobei $h \neq 0$ sein muss. $\epsilon_h$ ist in (2.8) in der Normalenform definiert.

$$\epsilon_h : \langle \vec{n} \ , \ \vec{x} - \overrightarrow{OP} \rangle = \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \ , \ \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ h \end{pmatrix} \rangle = 0 \tag{2.8}$$

Die Gerade $G_p$ ist durch den zu projizierenden Punkt $p$ und dem Richtungsvektor $\vec{v}$ bestimmt. $G_p$ ist in (2.9) in der Punkt-Richtungs-Form definiert.

$$G_p : \vec{p} + \lambda\vec{v} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \lambda \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \tag{2.9}$$

Erneut haben wir es mit einem Gleichungssystem zu tun, bei dem zunächst nach $\lambda$ aufgelöst werden soll, indem $G_p$ für $\vec{x}$ in $\epsilon_h$ eingesetzt wird.

$$\begin{aligned} p' &= \langle \vec{n}, \vec{p} + \lambda\vec{v} - \overrightarrow{OP} \rangle \\ &= \langle \vec{n}, \vec{p} \rangle + \lambda\langle \vec{n}, \vec{v} \rangle - \langle \vec{n}, \overrightarrow{OP} \rangle = 0 \end{aligned}$$

$$\lambda = \frac{\langle \vec{n}, \overrightarrow{OP} \rangle - \langle \vec{n}, \vec{p} \rangle}{\langle \vec{n}, \vec{v} \rangle} = \frac{\langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle}{\langle \vec{n}, \vec{v} \rangle}$$

Die Gleichung entspricht im allgemeinen exakt der aus $\mathbb{R}^2$. Wir fahren genau gleich fort und geben den Bildpunkt $p'$ in homogene Koordinaten an.

$$
\begin{aligned}
\langle \vec{n}, \vec{v} \rangle \cdot p' = \Big( & \langle \vec{n}, \vec{v} \rangle \cdot p_x + \langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle \cdot v_1 \ , \\
& \langle \vec{n}, \vec{v} \rangle \cdot p_y + \langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle \cdot v_2 \ , \\
& \langle \vec{n}, \vec{v} \rangle \cdot p_z + \langle \vec{n}, \overrightarrow{OP} - \vec{p} \rangle \cdot v_3 \ , \ \langle \vec{n}, \vec{v} \rangle \Big) \\
= \Bigg( & \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \begin{pmatrix} v_1 \\ v_2 \\ v_2 \end{pmatrix} \rangle \cdot p_x + \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \begin{pmatrix} 0 \\ 0 \\ h \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \rangle \cdot v_1 \ , \\
& \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \rangle \cdot p_y + \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \begin{pmatrix} 0 \\ 0 \\ h \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \rangle \cdot v_2 \ , \\
& \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \rangle \cdot p_z + \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \begin{pmatrix} 0 \\ 0 \\ h \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \rangle \cdot v_3 \ , \\
& \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} , \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \rangle \Bigg) \\
= & (p_x v_3 - p_z v_1 + v_1 h \ , \ p_y v_3 - p_z v_2 + v_2 h \ , \ p_z v_3 - p_z v_3 + v_3 h \ , \ v_3) \\
= & (p_x v_3 - p_z v_1 + v_1 h \ , \ p_y v_3 - p_z v_2 + v_2 h \ , \ v_3 h \ , \ v_3)
\end{aligned}
$$

Nun kann die Abbildungsmatrix angegeben werden, welche die geforderte Python Funktion `projektionsmatrix(richtung, h)` ausmacht.

$$
\epsilon_{\vec{v},h} = \begin{pmatrix} v_3 & 0 & -v_1 & v_1 h \\ 0 & v_3 & -v_2 & v_2 h \\ 0 & 0 & 0 & v_3 h \\ 0 & 0 & 0 & v_3 \end{pmatrix} \tag{2.10}
$$

$$
\epsilon_{\vec{v},h} \cdot \frac{1}{v_3} = \begin{pmatrix} 1 & 0 & -v_1/v_3 & v_1 h/v_3 \\ 0 & 1 & -v_2/v_3 & v_2 h/v_3 \\ 0 & 0 & 0 & h \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & -v_1/v_3 & v_1 h/v_3 \\ 0 & 1 & -v_2/v_3 & v_2 h/v_3 \\ 0 & 0 & 0 & h \end{pmatrix} \tag{2.11}
$$

Die Projektion $P'$ eines beliebigen Punktes $P = (3, 5, 2)$ bei vorgegebenem Richtungsvektor $\vec{v} = (2\ 4\ -4)^T$ und vorgegebener Höhe $h = 2$ ist nun ganz einfach (2.12).

$$
P' = \begin{pmatrix} 1 & 0 & -2/(-4) & 2 \cdot 2/(-4) \\ 0 & 1 & -4/(-4) & 4 \cdot 2/(-4) \\ 0 & 0 & 0 & 2 \end{pmatrix} \cdot (3\ 5\ 2\ 1)^T = \begin{pmatrix} 3 \\ 5 \\ 2 \end{pmatrix} \tag{2.12}
$$

# 3 Python-Code

Vorgestellt wird die Funktion `projektionsmatrix(richtung, h)` und Visualisierungen, die die Parallelprojektionen in 3D zeigen.

## 3.1 Implementierungsidee

Für die Visualisierung wird die Python-Bibliothek `matplotlib`[1] eingesetzt werden. Für das Rechnen mit Vektoren und Matrizen, sind die Klassen `linalg.Vector` und `linalg.Matrix` implementiert. Beide Klassen unterstützen diverse Operationen für das Arbeiten mit Vektoren und Matrizen. Die Funktion `projektionsmatrix`, ist aufgrund der im Kapitel 2 behandelten mathematischen Grundlagen implementiert. Für die Visualisierung eines Streckenzuges, ist ein Würfel gewählt worden. Die Berechnung der Kanten ist im Modul `linalg.plot.figures.cube` gekapselt. Diverse weitere Hilfsfunktionen, die spezifisch für die Semesterarbeit sind, sind im Modul `sema.teil1.pp3d` untergebracht. Darunter auch die Funktion `plotParallelprojektion` die es sehr einfach macht, Parallelprojektionen im 3D zu plotten.

Im Rahmen des Moduls *LinAlg* ist noch viel weiterer Programmcode entstanden. Besonders auch um Visualisierungen im 2D zu erstellen (bspw. lineare Transformationen). Beispielsweise die Klasse `linalg.plot.CoordinateSystem` ist speziell dafür erstellt. Die Abbildung 2.1 ist mithilfe dieser Klasse, im Modul `sema.teil1.pp2d` erstellt worden. Auf diese Komponenten wird jedoch nicht weiter eingegangen.

## 3.2 Programmcode

### 3.2.1 `projektionsmatrix`

Für die Berechnung der Bildpunkte ist die Funktion `projektionsmatrix` zuständig. Diese Funktion ist im Modul `sema.teil.pp3d` definiert. Die Funktion generiert abhängig vom Parameter `richtung` und `h` eine Matrix für die Parallelprojektion im 3D. Die generierte

[1] `https://matplotlib.org`

Matrix ist eine $(3 \times 4)$-Matrix. Ein Vektor der damit multipliziert wird, muss in homogenen Koordinaten angegeben sein. Das Resultat ist aber aber ein Vektor aus dem $\mathbb{R}^3$. Damit ersparen wir uns, das Entfernen der letzten Komponente. Das funktioniert aber nur deshalb, weil die Matrix so gestaltet ist, dass die Komponente in der letzten Zeile und letzten Spalte, eine 1 ist [2]. Demnach konnte die letzten Zeile weggelassen werden.

```
def projektionsmatrix(richtung, h):
    v1, v2, v3 = richtung
    return Matrix([1, 0, -v1 / v3, v1 * h / v3],
                  [0, 1, -v2 / v3, v2 * h / v3],
                  [0, 0, 0, h])
```

### 3.2.2 `coordinateSystem3d`

Diese Funktion erstellt ein 3D Plot. Die Grösse der Achsen kann mittels dem Parameter `range` angegeben werden. Wobei der Parameter ein Array mit zwei Werten ist (Start– und Endwert der Achsen). Die selbe Range wird für alle drei Achsen verwendet. Zusätzlich kann mit dem Parameter `axisOff` gewählt werden, ob die Achsen überhaupt eingeblendet werden sollen (Standardmässig ist der Wert `False` gewählt). Die Funktion gibt eine Subklasse von `~.axes.Axes`, aus der `matplotlib` Bibliothek, zurück.

### 3.2.3 `plotParallelprojektion`

Die Funktion `sema.teil1.pp3d.plotParallelprojektion` fasst das meiste zusammen und erlaubt es, sehr einfach einen Würfel auf eine Ebene zu projizieren. Dabei kann der Richtungsvektor `richtung` und der Abstand `h` gewählt werden. Für die visuelle Darstellung verwendet die Funktion dann die Hilfsfunktion `drawXyPlane` für die Zeichnung der $xy$ Ebene, `drawCube` für die Zeichnung des Würfels und `drawLines` bzw. `drawLine` zur Zeichnung der Würfelkanten und der Projektionslinien. Die Bildpunkte werden alle mithilfe einer Matrix berechnet.

Um die Visualisierung möglichst gut zur veranschaulichen, werden die Achsen ausgeblendet und der Betrachtungswinkel optimal eingestellt. Letzteres kann im `matplotlib` mittels `ax.view_init`[3] eingestellt werden. Leider erlaubt die Funktion aber nur Drehungen um zwei Achsen womit die gewünschte Ansicht nicht ohne Weiteres eingestellt werden kann[4]. Um das Problem zu umgehen, werden in den für die Zeichnung zuständigen Funktionen

[2] Siehe dafür (2.11).

[3] `https://matplotlib.org/2.0.2/mpl_toolkits/mplot3d/api.html#mpl_toolkits.mplot3d.axes3d.Axes3D.view_init`

[4] `https://stackoverflow.com/a/56457693/9816335`

`drawLine`, `drawCube` und `drawXyPlane`, die $y$ und $z$ Werte vertauscht. Die Berechnungen bleiben unverändert, einzig die visuelle Darstellung ist so gewählt, dass der Betrachtungswinkel für die Ansicht, optimal gewählt werden kann.

Nachfolgend beispielhaft die Funktion `drawXyPlane` wo die Vertauschung zu erkennen ist.

```
def drawXyPlane(ax, range, h, color='darkgray', alpha=0.2, **kwargs):
    x, y, z = xyEbene(range, h)
    ax.plot_surface(x, z, y, color=color, alpha=alpha, **kwargs)
```

### 3.2.4 `main`

In der `main` Funktion wird die Funktion `plotParallelporjektion` aufgerufen. Im Google-Colab damit selber herumprobiert werden[5].

## 3.3 Visualisierungen

Nachfolgen werden zwei 3D Visualisierungen vorgestellt. Die graue Fläche stellt die $xy$ Projektionsfläche dar. Der Würfel mit den grünen Punkten befindet sich im $\mathbb{R}^3$ und wird auf die Projektionsfläche projiziert. Die gestrichelten Linien, laufend zu der Projektionsfläche, sind die Projektionslinien. Die Projektion auf der Ebene stellt eine 2D Ansicht des Würfels dar.

In Abbildung 3.1 ist der Richtungsvektor $(1\ 2\ 3)^T$ und die Höhe 20 gewählt. In Abbildung 3.2 sind zwei Ebenen, aber die selbe Projektion zu sehen. Einmal ist die Höhe 10 und einmal die Höhe 25 gewählt. Der Richtungsvektor ist in beiden Fällen $(4\ 2\ 5)^T$. Daran ist gut zu erkennen, was mit der letzten Aussage aus Abschnitt 2.1 gemeint ist.

Eine animierte Version der Parallelprojektion findest du online.

`https://linalg.arberosmani.ch/pp3d.gif`

1

[5] `https://colab.research.google.com/drive/1dOfhAthenCPuNzb8mWv9xIahp55-cQQh?usp=sharing`

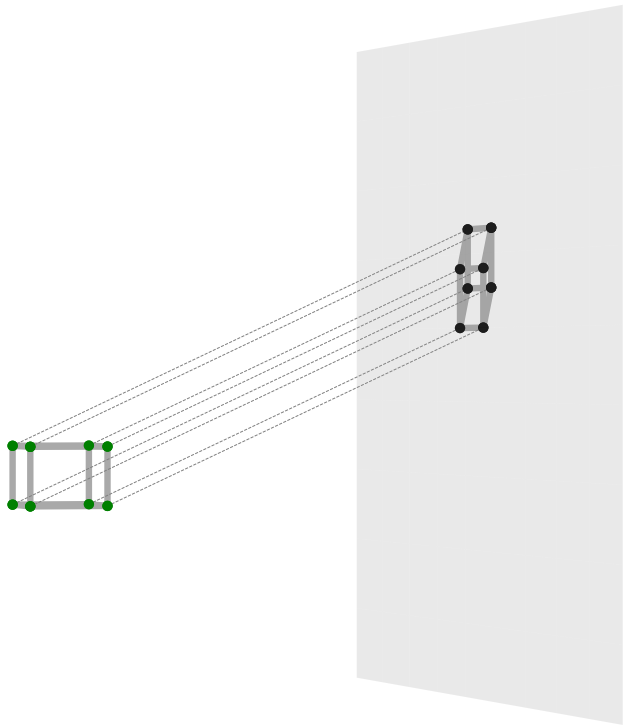


Abbildung 3.1: Parallelprojektion eines Würfels im $\mathbb{R}^3$ auf der $xy$-Ebene

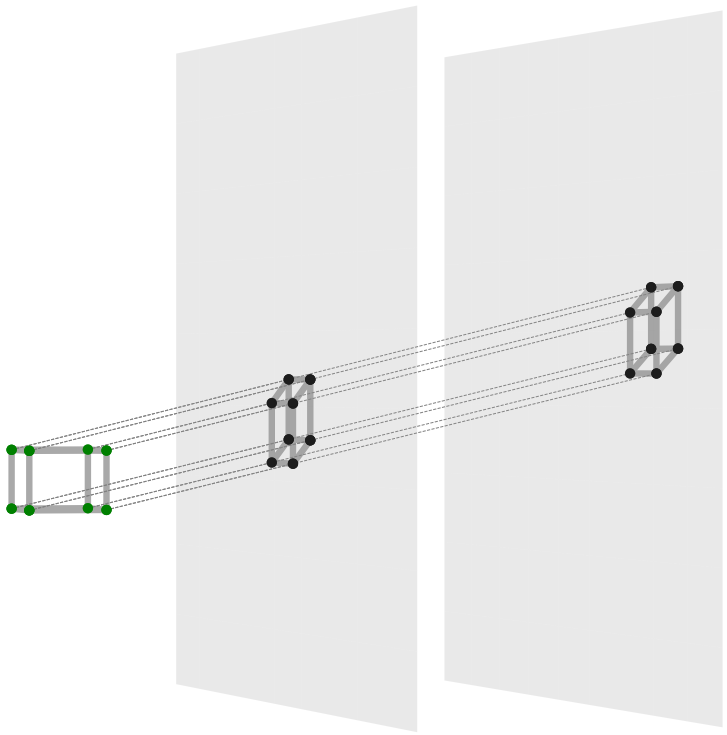

Abbildung 3.2: Parallelprojektion eines Würfels im $\mathbb{R}^3$ auf der $xy$-Ebenen in zwei unterschiedlichen Höhen

# 4 Konklusion

## 4.1 Diskussion der Ergebnisse

Die Generierung einer Matrix mittels homogenen Koordinaten, macht die Berechnung der Parallelprojektion sehr einfach. Es können auch beliebig viele Transformationen in die selbe Matrix gepackt werden. Dafür sind die Matrizen wie gewohnt, der gewünschten Reihenfolge entsprechend, miteinander zu multiplizieren.

Die Projektion im $\mathbb{R}^2$ unterscheidet sich kaum von der im $\mathbb{R}^3$. Eine „Parallelprojektion“ in höheren Dimensionen, mittels den erarbeiteten theoretischen Grundlagen, ist auch kein Problem mehr. Die Visualisierung ist dann aber nicht mehr möglich.

Mit `matplotlib` können Visualisierungen sehr einfach erstellt werden und unterstützen das Verständnis enorm. In der animierten Version der Darstellung[1] ist beispielsweise sehr gut zu erkennen, dass der projizierte Würfen kein 3D Objekt mehr ist. Sobald die Ebene exakt zum Auge gedreht ist, erkennt man, dass die Bildpunkte auf einer Ebene liegen.

[1] `https://linalg.arberosmani.ch/pp3d.gif`

# Literaturverzeichnis


[1] Heinrich Brauner. *Parallelprojektion*. Springer Vienna, 1986. doi: 10.1007/978-3-7091-8778-4_2. URL https://doi.org/10.1007/978-3-7091-8778-4_2.

[2] Cornelie Leopold. *ZENTRALPROJEKTION*. Vieweg+Teubner Verlag, December 2011. doi: 10.1007/978-3-8348-1986-4_13. URL https://doi.org/10.1007/978-3-8348-1986-4_13.

[3] Jürgen Richter-Gebert and Thorsten Orendt. *Homogene Koordinaten der Ebene*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-02530-3. doi: 10.1007/978-3-642-02530-3_1. URL https://doi.org/10.1007/978-3-642-02530-3_1.

[4] Edmund Weitz. *Konkrete Mathematik (nicht nur) für Informatiker*. Springer Fachmedien Wiesbaden, 2018. doi: 10.1007/978-3-658-21565-1. URL https://doi.org/10.1007/978-3-658-21565-1.