Transformation dreidimensionaler Koordinaten in zweidimensionale Koordinaten (German version)

Transformation dreidimensionaler Koordinaten in zweidimensionale Koordinaten (German Version)

Edward Gerhold

August 27, 2015

Deutsche Version 0.0.98 von 0.1.0

Contents

1 Einleitung

Auf einem Blatt Papier sehen wir ein dreidimensionales Koordinatensystem in den Raum zeigen. In der Realität sind die drei Basisvektoren der Abbildung zweidimensional. Denn sie zeigen in drei Richtungen, flach auf dem Papier, und gar nicht in den reellen Raum.

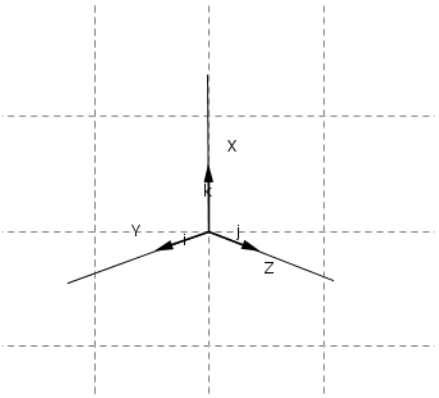


Figure 1: Bild eines rechtshändigen Koordinatensystems mit ijk-Basisvektoren auf den Achsen, in drei dimensionen zeigend. Schauen sie in (**?**) für eine Einführung.

In diesem Dokument entwerfen wir ein $\mathbb{R}^{2\times3}$ Koordinatensystem um 3-D Punkte in 2-D Punkte umzuwandeln. Mit den Kosinus- und Sinusfunktionen errechnen wir die exakten Anteile der horizontalen und vertikalen Teilverschiebungen.

Was wir in diesem Dokument machen werden.

1. Winkel auswählen für unsere Koordinatenachsen
2. Einheiten der Achsen auswählen oder sich für 1 entscheiden
3. Die drei zweidimensionalen Achsvektoren des Koordinatensystems aufschreiben
4. Die Achsen zu einer Matrix, einer Funktion, oder anstelle der Basis zusammenbauen
5. Den JavaScript Code der Transformation lesen
6. In den Folgeversionen die noch nicht eingetragenen bereits bekannten Fakten kennenlernen

2 Entwurf eines $\mathbb{R}^{2\times3}$ Koordinatensystems für unsere Koordinatentransformation vom $\mathbb{R}^3$ auf den $\mathbb{R}^2$

In der englischen Version schreibe ich alles etwas anders. Diese deutsche Fassung ist speziell neu angefangen und ich verfolge das Ziel, mich auch etwas kürzer zu fassen. Mathematik, die ich in der deutschen Fassung auslassen werde, ist aber problemlos in der englischen Version zu verfolgen. Wer das rechnen kann, kann bestimmt auch ein wenig Englisch und versteht auch deutsches Englisch.

3 Definitionen

3.1 Koordinatensystem

3.1.1 Links- und rechts

Ein linkshändiges Koordinatensystem auf der Ebene hat die dritte Achse zwischen den beiden normalen Achsen in die gleiche Richtung zeigend.

Bei einem rechtshändigen Koordinatensystem auf der Ebene zeigt die dritte Achse in die entgegengesetzte Richtung.

3.1.2 Einheitskreis

Der Einfachheit halber kann man die drei Achsen am Einheitskreis ausrichten. Dazu bedienen wir uns gleich einer hilfreichen Formel aus dem Polarkoordinatensystem.

$$(x, y) := (r\cos\varphi, r\sin\varphi)$$

Figure 2: Ein linkshändiges Koordinatensystem

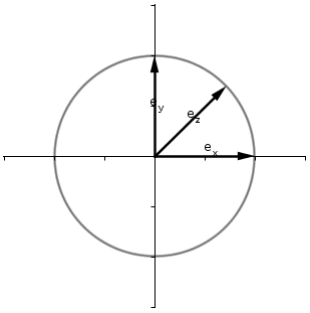
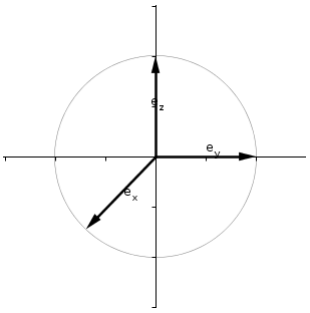


Figure 3: Ein rechtshändiges Koordinatensystem



(x, y) stellen die Spitze eines Ortsvektors aus Zentrum des Koordinatensystems dar. Wenn wir drei Achsen haben wollen, sollten wir uns drei solche Vektoren erzeugen. Das werden wir auch gleich tun.

3.2 Winkel der Achsen

Die Achsen werden von der horizontalen x-Achse des zweidimensionalen Bilds gegen den Uhrzeigersinn gezählt. Da wir drei Achsen haben, brauchen wir auch drei Winkel.

$$\varphi_n := \{\varphi_x, \varphi_y, \varphi_z | \text{ die Winkel der drei Achsen}\}$$

Die Winkel kann man selbst in Grad oder Radians festlegen. Die Programmiersprache JavaScript zum Beispiel nimmt, für die Funktionen Kosinus und Sinus die Winkel, in Radians an. Die sind mit einer einfachen Formel berechenbar.

$$\text{rad(deg)} := \frac{\pi}{180} \times \text{deg}$$

$$\text{deg(rad)} := \frac{180}{\pi} \times \text{rad}$$

3.3 Einheit der Achsen

3.3.1 Der r-Wert der Achsen

$$r_n := \{r_x, r_y, r_z | \text{ die Einheit jeder der drei Achsen}\}$$

3.3.2 Mathematische Vorsicht mit dem r-Wert

Der r-Wert verkompliziert die Berechnungen und Abschätzungen natürlich, weil die Koordinaten mit multipliziert werden.

Wenn man bewegte Bilder produzieren will, sollte der r-Wert grundsaetzlich gleich auf allen Achsen sein, weil Rotationen und Translationen sonst daneben gehen, da die Punkte dann plötzlich ihre Einheiten wechseln. Das sieht unrealistisch aus, ist aber bei Standbildern kein Problem.

Die Formel verkompliziert sich bei drei verschiedenen r-Werten natürlich und zum Rechnen sollte zuerst das vereinfachte Modell herangezogen werden, wo der r-Wert auf allen drei Achsen gleich ist, oder gleich 1 ist und komplett entfällt.

3.4 (Basis)vektoren der Linearkombination

Wir benötigen für das Koordinatensystem drei Achsen. Jede Achse bekommt einen kanonischen Einheitsvektor.

$$\vec{e}_n := \{\vec{e}_x, \vec{e}_y, \vec{e}_z | \vec{e}_n = \begin{pmatrix} r_n \cos\varphi_n \\ r_n \sin\varphi_n \end{pmatrix}, \text{ die drei Achsen des Koordinatensystems } \}$$

Wenn der r-Wert gleich 1 ist, haben diese Vektoren gleich normalisierte Einheitslänge im Sinne der Orthonormalbasis. Der Unterschied zur Orthonormalbasis ist, dass wir mindestens eine linear abhängige Achse haben. Je nach Arrangement um den Kreis können dabei bis zu drei linear abhängige Achsen entstehen, in Bezug zur 2-D Standardbasis $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ auf die das Ergebnis abgebildet wird.

4 Transformationswerkzeuge

In diesem Kapitel stelle ich dann vor, wie man die Transformation durchführt.

4.1 Anstelle der Basis

Wir definieren mit $x\vec{i} + y\vec{j} + z\vec{k}$ in der Regel einen Vektor mit einer 3x3 Basis. Man kann die 2x3 Basis anstelle der 3x3 Basis einsetzen und erhält eine saubere Orthogonalprojektion.

$$x\vec{e}_x + y\vec{e}_y + z\vec{e}_z = \vec{w}$$

4.2 Funktional

Die Funktion nimmt einen Vektor mit drei Koordinaten an und gibt einen Vektor mit zwei Koordinaten zurück. Den richtigen Punkt.

$$\vec{f}(\vec{x}) := x \begin{pmatrix} r_x \cos\varphi_x \\ r_x \sin\varphi_x \end{pmatrix} + y \begin{pmatrix} r_y \cos\varphi_y \\ r_y \sin\varphi_y \end{pmatrix} + z \begin{pmatrix} r_z \cos\varphi_z \\ r_z \sin\varphi_z \end{pmatrix}$$

4.3 Matrix

Die Multiplikation der Matrix mit einem dreidimensionalen Vektor gibt einen zweidimensionalen Vektor zurück. Den richtigen Punkt.

$$\boldsymbol{A} := \begin{pmatrix} r_x \cos\varphi_x & r_y \cos\varphi_y & r_z \cos\varphi_z \\ r_x \sin\varphi_x & r_y \sin\varphi_y & r_z \sin\varphi_z \end{pmatrix}$$

Bemerkung. Über die drei Werkzeuge sind mir bereits mehr Fakten bekannt und ich werde sie in der kommenden Version berücksichtigen und freilassen.

5 Transformationsverhalten

Die Transformation ist eine lineare Transformation und erfüllt damit die grundsätzlichen Bedingungen der linearen Funktionen.

5.1 Der Ursprung wird auf den Ursprung abgebildet

Der Nullvektor aus dem dreidimensionalen Raum $\vec{0} \in \mathbb{R}^3$ wird auf den Nullvektor im zweidimensionalen Raum abgebildet $\vec{0} \in \mathbb{R}^2$.

Beweis:

$$\boldsymbol{A}\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0+0+0 \\ 0+0+0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

5.2 Punkte auf einer Achsen

Punkte, die nur auf einer Achse liegen sind ein vielfaches des jeweiligen Achsenvektors.

Beweis:

$$\boldsymbol{A}\begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} ar_x \cos\varphi_x + 0 + 0 \\ ar_x \sin\varphi_x + 0 + 0 \end{pmatrix} = a\vec{e}_x$$

$$\boldsymbol{A}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 + r_y \cos\varphi_y + 0 \\ 0 + r_y \sin\varphi_y + 0 \end{pmatrix} = \vec{e}_y$$

$$\boldsymbol{A}\begin{pmatrix} 0 \\ 0 \\ -b \end{pmatrix} = \begin{pmatrix} 0 + 0 - br_z \cos\varphi_z \\ 0 + 0 - br_z \sin\varphi_z \end{pmatrix} = -b\vec{e}_z$$

5.3 Skalare Multiplikation

Es ist einfach zu zeigen, dass $\boldsymbol{A}(\lambda\vec{x}) = \lambda\boldsymbol{A}\vec{x}$. Man kann vor der Transformation oder nach der Transformation mit dem Skalar multiplizieren. Das Resultat ist identisch.

Beweis:

$$\begin{aligned}
\boldsymbol{A}(\lambda\vec{x}) &= \boldsymbol{A}\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \end{pmatrix} \\
&= \begin{pmatrix} \lambda x r_x \cos(\varphi_x) + \lambda y r_y \cos(\varphi_y) + \lambda z r_z \cos(\varphi_z) \\ \lambda x r_x \sin(\varphi_x) + \lambda y r_y \sin(\varphi_y) + \lambda z r_z \sin(\varphi_z) \end{pmatrix} \\
&= \lambda \begin{pmatrix} x r_x \cos(\varphi_x) + y r_y \cos(\varphi_y) + z r_z \cos(\varphi_z) \\ x r_x \sin(\varphi_x) + y r_y \sin(\varphi_y) + z r_z \sin(\varphi_z) \end{pmatrix} \\
&= \lambda \begin{pmatrix} x' \\ y' \end{pmatrix} \\
&= \lambda\boldsymbol{A}\vec{x}
\end{aligned}$$

5.4 Addition und Subtraktion

Einfach zu zeigen ist auch, dass $\boldsymbol{A}(\vec{v} + \vec{w}) = \boldsymbol{A}\vec{v} + \boldsymbol{A}\vec{w}$. Man kann auch hier die Eingabevektoren vor der Umwandlung addieren, oder die Ausgabevektoren nach der Umwandlung. Die resultierende Summe ist identisch.

Beweis:

$$\begin{aligned}
\boldsymbol{A}\begin{pmatrix} x+u \\ y+v \\ z+w \end{pmatrix} &= \begin{pmatrix} (x+u)r_x\cos(\varphi_x)+(y+v)r_y\cos(\varphi_y)+(z+w)r_z\cos(\varphi_z) \\ (x+u)r_x\sin(\varphi_x)+(y+v)r_y\sin(\varphi_y)+(z+w)r_z\sin(\varphi_z) \end{pmatrix} \\
&= \begin{pmatrix} xr_x\cos(\varphi_x)+yr_y\cos(\varphi_y)+zr_z\cos(\varphi_z) \\ xr_x\sin(\varphi_x)+yr_y\sin(\varphi_y)+zr_z\sin(\varphi_z) \end{pmatrix} + \begin{pmatrix} ur_x\cos(\varphi_x)+vr_y\cos(\varphi_y)+wr_z\cos(\varphi_z) \\ ur_x\sin(\varphi_x)+vr_y\sin(\varphi_y)+wr_z\sin(\varphi_z) \end{pmatrix} \\
&= \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} u' \\ v' \end{pmatrix} \\
&= \boldsymbol{A}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \boldsymbol{A}\begin{pmatrix} u \\ v \\ w \end{pmatrix}
\end{aligned}$$

5.5 Linearität

Durch die letzten zwei Beweise ist es offensichtlich, dass die Transformation die Regeln der Linearität befolgt.

$$\boldsymbol{A}(\lambda\vec{v}+\kappa\vec{w}) = \lambda\boldsymbol{A}\vec{v}+\kappa\boldsymbol{A}\vec{w} = \lambda\begin{pmatrix} x' \\ y' \end{pmatrix} + \kappa\begin{pmatrix} u' \\ v' \end{pmatrix}$$

Auf den Beweis der Kombination beider Abschnitte verzichte ich.

6 Computer Implementierung

6.1 Generischer Code

Das folgende Beispiel ist der Pseudocode für alle Computersysteme.

```
x_ = x*r*cos(alpha) + y*r*cos(beta) + z*r*cos(gamma)
y_ = x*r*sin(alpha) + y*r*sin(beta) + z*r*sin(gamma)
```

Das sind die einzigen beiden Zeilen Code die man braucht.

6.2 JavaScript Implementierung

Das ist ein komplettes EcmaScript 6 snippet mit allen notwendigen Informationen.

```
let rad = (deg) => Math.PI/180*deg;
let r_x = 1, r_y = 1, r_z = 1;
let phi_x = rad(220), phi_y = rad(330), phi_z = rad(90);
let xAxisCos = r_x*Math.cos(phi_x),
    yAxisCos = r_y*Math.cos(phi_y),
    zAxisCos = r_z*Math.cos(phi_z),
    xAxisSin = r_x*Math.sin(phi_x),
    yAxisSin = r_y*Math.sin(phi_y),
    zAxisSin = r_z*Math.sin(phi_z);
let transform2d = ([x,y,z]) => [
    x*xAxisCos+ y*yAxisCos+ z*zAxisCos,
    x*xAxisSin+ y*yAxisSin+ z*zAxisSin];
let transform2dAll = (P) => P.map(transform2d);

let beispielPunkte = transform2dAll([[1,2,3], [3,4,5], [14,24,15]]);
```

Das ist die realistische Menge an Code für die komplette Transformation von 3-D nach 2-D.

7 Selbständigkeitserklärung

Ich habe dieses Koordinatensystem selbst entwickelt. Es ist keine Formel aus einem Buch oder einer Lehrveranstaltung. Ob es irgendwo eine identische Formel oder eine vergleichbare Definition gibt, ist mir nicht bekannt. Ich habe den Inhalt des Dokuments aus eigenem Ermessen zusammengestellt. Ich habe mir Gedanken zum Thema gemacht und ausserdem Rechnungen mit Stift und Papier angefertigt. Ausserdem habe ich in Lehrbüchern und Veranstaltungen geblättert, um das Koordinatensystem und die definierten Variabeln und Operationen möglichst gut in die reelle Mathematik einzuordnen. Mir mögen Fehler unterlaufen sein, und auch Details entgangen sein. Für beides möchte ich mich entschuldigen.

8 Lizenz

Der produzierte Source Code, um das Koordinatensystem und einige Abbildungen zu zeigen, ist frei für alle, wie auch das Koordinatensystem selbst und die dazugehörigen Definitionen, die ich selbst angefertigt habe. Es ist erlaubt, mir dafür Anerkennung zu gewähren, es ist aber nicht zwingend nötig, mich dafür im eigenen Projekt zu nennen. Allerdings mag auch ich keine Menschen, die diese Arbeit für ihre eigene ausgeben.

References

Corral1 *Michael Corral, Schoolcraft College*, Vector Calculus, GNU Free Documentation License, http://mecmath.net

[1] [1] *Michael Corral, Schoolcraft College*, Trigonometry, GNU Free Documentation License, http://mecmath.net

[2] *Gilbert Strang, MIT*, Linear Algebra and its Applications. Fourth Edition.

[3] *Gilbert Strang, MIT*, Calculus. MIT OpenCourseWare Supplemental Resources. http://ocw.mit.edu

[4] *John Rogues, Lecture Notes on Topology, following J.R.Munkres Textbook, for MAT3500/4500*, Lecture Script, Topology (english), http://

[5] *Roman Vershynin. Lectures in Functional Analysis. Department of Mathematics, University of Michigan*, Lecture Script, http://,

[6] *Dirk Ferus, TU-Berlin, em.*, Lecture Script, Lineare Algebra 1+2, 2000, http://page.math.tu-berlin/ ferus/skripten.html

[7] *Franziska Kühn, Technische Universität Dresden*, Lecture Script, Lineare Algebra und analytische Geometrie I+II, http://fkuehn.de/download/LAAG.pdf

[8] *Petra Wittbold, TU-berlin*, Lecture Script, Funktionalanalysis I, http://www3.math.tu-berlin.de/Vorlesungen/SS09/FA1/Doc/Funkana1-SS06-08.06.09.pd

[9] *Michael Corral, Schoolcraft College*, Latex Mini Tutorial, http://mecmath.net

[10] *Manuela Jürgens, Thomas Feuerstack, Fernuniversität Hagen*, LaTeX, eine Einführung und ein bisschen mehr..., a026_latex_einf.pdf

[11] *Dr.Jan Rudl, Technische Universität Dresden, Fachbereich Mathematik*, Einführung in LaTeX, LaTeX-Kurs.pdf