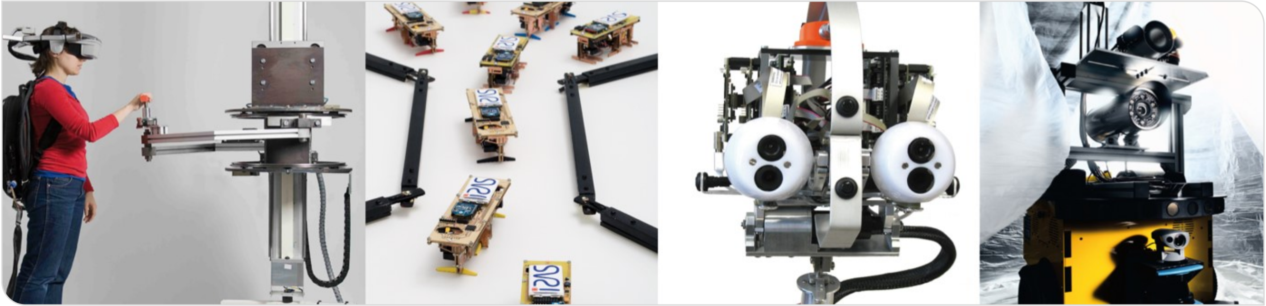
Karlsruher Institut für Technologie

Punkt-Warping: Inversionsmethode mit Hashing

Lukas Knirsch | 12. Januar 2022

Betreuer: Daniel Frisch



Inhaltsverzeichnis

Problemstellung

- Punkte nach Dichtefunktion im Raum verteilen

Problemstellung

- Punkte nach Dichtefunktion im Raum verteilen
- Mehrere existierende Verfahren
  - Halton (Wong, Luk und Heng 1997)
  - Blue Noise (Yan u. a. 2015)

Problemstellung

- Punkte nach Dichtefunktion im Raum verteilen
- Mehrere existierende Verfahren
  - Halton (Wong, Luk und Heng 1997)
  - Blue Noise (Yan u. a. 2015)
- *Erweiterung:* gleichmäßige Verteilung
  - Golden Ratio Sequences + Halton/Blue Noise (Schretter, Kobbelt und Dehaye 2012)
  - Fibonacci Grids (Frisch und Hanebeck 2021)

Problemstellung

- Punkte nach Dichtefunktion im Raum verteilen
- Mehrere existierende Verfahren
  - Halton (Wong, Luk und Heng 1997)
  - Blue Noise (Yan u. a. 2015)
- *Erweiterung:* gleichmäßige Verteilung
  - Golden Ratio Sequences + Halton/Blue Noise (Schretter, Kobbelt und Dehaye 2012)
  - Fibonacci Grids (Frisch und Hanebeck 2021)
- **Ziel 1:** effizientes Vorgehen

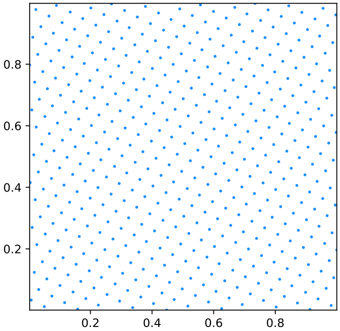
Problemstellung

- Punkte nach Dichtefunktion im Raum verteilen
- Mehrere existierende Verfahren
  - Halton (Wong, Luk und Heng 1997)
  - Blue Noise (Yan u. a. 2015)
- *Erweiterung:* gleichmäßige Verteilung
  - Golden Ratio Sequences + Halton/Blue Noise (Schretter, Kobbelt und Dehaye 2012)
  - Fibonacci Grids (Frisch und Hanebeck 2021)
- **Ziel 1:** effizientes Vorgehen
- **Ziel 2:** mehrdimensional anwendbar

Problemstellung

- Punkte nach Dichtefunktion im Raum verteilen
- Mehrere existierende Verfahren
  - Halton (Wong, Luk und Heng 1997)
  - Blue Noise (Yan u. a. 2015)
- *Erweiterung:* gleichmäßige Verteilung
  - Golden Ratio Sequences + Halton/Blue Noise (Schretter, Kobbelt und Dehaye 2012)
  - Fibonacci Grids (Frisch und Hanebeck 2021)
- **Ziel 1:** effizientes Vorgehen
- **Ziel 2:** mehrdimensional anwendbar
- **Ziel 3:** unabhängig von bestimmten Eigenschaften der Funktion einsetzbar

Ziel



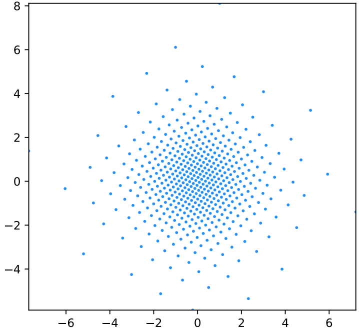
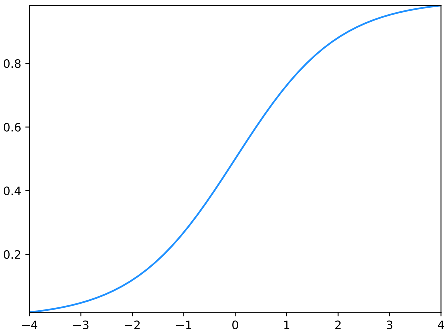
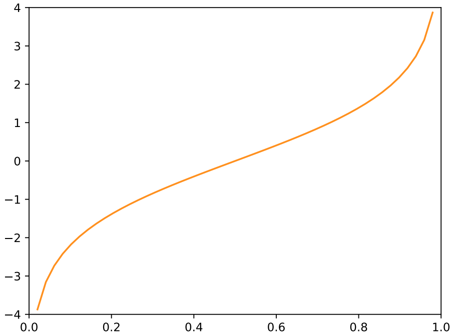


Abbildung: Schretter, Kobbelt und Dehaye 2012

Inversionsmethode



(a) F mit Werten von −4 bis 4



(b) F^{-1} mit Werten von 0 bis 1

Inversionsmethode

- Inverse einer Dichtefunktion benötigt
- Entweder bekannt oder numerisch integrierbar/annäherbar

Name	Funktion F	Zufällige Variable F^{-1}
Exponentiell	$1 - e^{-x}$	$\log(1/U)$
Logistisch	$1/(1 + e^{-x})$	$-\log(\frac{1-U}{U})$
Cauchy	$1/2 + (1/\pi)\arctan(x)$	$\tan(\pi U)$

Tabelle: Devroye 1986

Inversionsmethode

- Inverse einer Dichtefunktion benötigt
- Entweder bekannt oder numerisch integrierbar/annäherbar

Name	Funktion F	Zufällige Variable F^{-1}
Exponentiell	$1 - e^{-x}$	$\log(1/U)$
Logistisch	$1/(1+e^{-x})$	$-\log(\frac{1-U}{U})$
Cauchy	$1/2 + (1/\pi)\arctan(x)$	$\tan(\pi U)$

Tabelle: Devroye 1986

$\rightarrow$ **Problem:** einfache Inversionsmethode ineffizient

Hash-basierte Inversionsmethode

Initialisierung.
Funktion C_F und Wahrscheinlichkeiten $v_j,\ j \in [1,\ n]$

Hash-basierte Inversionsmethode

Initialisierung.
Funktion C_F und Wahrscheinlichkeiten $v_j,\ j \in [1,\ n]$

- $C_F(v_{j-1}) < U \leq C_F(v_j)$ (1) benötigt naiv $O(n)$

Hash-basierte Inversionsmethode

Initialisierung.
Funktion C_F und Wahrscheinlichkeiten $v_j,\ j \in [1,\ n]$

- $C_F(v_{j-1}) < U \leq C_F(v_j)$ (1) benötigt naiv $O(n)$
- Berechne $I_j = \lfloor C_F(v_j) * d \rfloor + 1$ (2)

Hash-basierte Inversionsmethode

Initialisierung.
Funktion C_F und Wahrscheinlichkeiten $v_j,\ j \in [1,\ n]$

- $C_F(v_{j-1}) < U \leq C_F(v_j)$ (1) benötigt naiv $O(n)$
- Berechne $I_j = \lfloor C_F(v_j) * d \rfloor + 1$ (2)
- für Tabelleneintrag $T(I_j) = k$, sodass $I(k) = I(j)$.

Hash-basierte Inversionsmethode

Generierung.
Hashtabelle mit Einträgen $T(i)$ und Zufallszahl U.

Hash-basierte Inversionsmethode

Generierung.
Hashtabelle mit Einträgen $T(i)$ und Zufallszahl U.

- Berechne I_U mit (2)

Hash-basierte Inversionsmethode

Generierung.
Hashtabelle mit Einträgen $T(i)$ und Zufallszahl U.

- Berechne I_U mit (2)
- für Index $i = T(I_U)$.

Hash-basierte Inversionsmethode

Generierung.
Hashtabelle mit Einträgen $T(i)$ und Zufallszahl U.

- Berechne I_U mit (2)
- für Index $i = T(I_U)$.
- Überprüfe Menge der Teilmengen $\{C_F(v_i), \ldots, C_F(v_{r-1})\}$ mit (1), sodass $I(i) \neq I(r)$ und $r > i$.

Hash-basierte Inversionsmethode

Generierung.
Hashtabelle mit Einträgen $T(i)$ und Zufallszahl U.

- Berechne I_U mit (2)
- für Index $i = T(I_U)$.
- Überprüfe Menge der Teilmengen $\{C_F(v_i), \ldots, C_F(v_{r-1})\}$ mit (1), sodass $I(i) \neq I(r)$ und $r > i$.
- Sobald Ungleichung (1) erfüllt, ist $X = v_j$

Hash-basierte Inversionsmethode: Beispiel

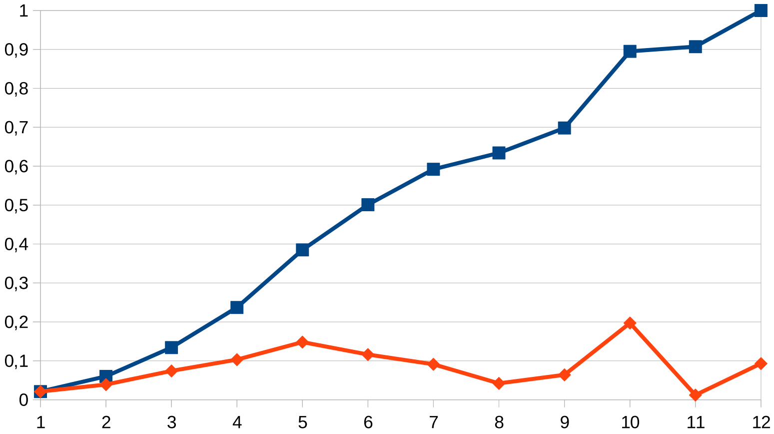


Abbildung: Verteilungsfunktion F (orange) und ihre kumulative Dichte C_F

Hash-basierte Inversionsmethode: Beispiel

X_j	$\mathrm{C_F}(X_j)$
v_1	0.021
v_2	0.060
v_3	0.134
v_4	0.237
v_5	0.385
v_6	0.501

X_j	$\mathrm{C_F}(X_j)$
v_7	0.592
v_8	0.634
v_9	0.698
v_{10}	0.895
v_{11}	0.907
v_{12}	1.000

Tabelle: Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

X_j	$C_F(X_j)$	I_j
v_1	0.021	1
v_2	0.060	1
v_3	0.134	2
v_4	0.237	3
v_5	0.385	4
v_6	0.501	6

X_j	$C_F(X_j)$
v_7	0.592
v_8	0.634
v_9	0.698
v_{10}	0.895
v_{11}	0.907
v_{12}	1.000

Tabelle: Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

X_j	$\mathrm{C_F}(X_j)$	I_j
v_1	0.021	1
v_2	0.060	1
v_3	0.134	2
v_4	0.237	3
v_5	0.385	4
v_6	0.501	6

X_j	$\mathrm{C_F}(X_j)$	I_j
v_7	0.592	6
v_8	0.634	7
v_9	0.698	7
v_{10}	0.895	9
v_{11}	0.907	10
v_{12}	1.000	11

Tabelle: Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

X_j	$\mathrm{C_F}(X_j)$	I_j
v_1	0.021	1
v_2	0.060	1
v_3	0.134	2
v_4	0.237	3
v_5	0.385	4
v_6	0.501	6

X_j	$\mathrm{C_F}(X_j)$	I_j
v_7	0.592	6
v_8	0.634	7
v_9	0.698	7
v_{10}	0.895	9
v_{11}	0.907	10
v_{12}	1.000	11

Tabelle: Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_1	0.021	1	1
v_2	0.060	1	1
v_3	0.134	2	3
v_4	0.237	3	4
v_5	0.385	4	5
v_6	0.501	6	6

X_j	$\mathrm{C_F}(X_j)$	I_j
v_7	0.592	6
v_8	0.634	7
v_9	0.698	7
v_{10}	0.895	9
v_{11}	0.907	10
v_{12}	1.000	11

Tabelle: Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_1	0.021	1	1
v_2	0.060	1	1
v_3	0.134	2	3
v_4	0.237	3	4
v_5	0.385	4	5
v_6	0.501	6	6

X_j	$\mathrm{C_F}(X_j)$	I_j
v_7	0.592	6
v_8	0.634	7
v_9	0.698	7
v_{10}	0.895	9
v_{11}	0.907	10
v_{12}	1.000	11

Tabelle: Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

X_j	$C_F(X_j)$	I_j	$T(I_j)$
v_1	0.021	1	1
v_2	0.060	1	1
v_3	0.134	2	3
v_4	0.237	3	4
v_5	0.385	4	5
v_6	0.501	6	6

X_j	$C_F(X_j)$	I_j	$T(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Tabelle: Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

u = 0.642

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

u = 0.642

- $I_u = \lfloor u * 10 \rfloor + 1 = 7$

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

u = 0.642

- $I_u = \lfloor u * 10 \rfloor + 1 = 7$
- $i = \mathrm{T}(I_u) = \mathrm{T}(7) = 8$

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

u = 0.642

- $I_u = \lfloor u * 10 \rfloor + 1 = 7$
- $i = \mathrm{T}(I_u) = \mathrm{T}(7) = 8$
- r ist nächstgrößerer Index, sodass $\mathrm{T}(v_r) \neq \mathrm{T}(v_i)$, also $r = 10$

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

u = 0.642

- $I_u = \lfloor u * 10 \rfloor + 1 = 7$
- $i = \mathrm{T}(I_u) = \mathrm{T}(7) = 8$
- r ist nächstgrößerer Index, sodass $\mathrm{T}(v_r) \neq \mathrm{T}(v_i)$, also $r = 10$
- $\mathrm{F}(v_{i-1}) < u \leq \mathrm{F}(v_r)$

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

u = 0.642

- $I_u = \lfloor u * 10 \rfloor + 1 = 7$
- $i = \mathrm{T}(I_u) = \mathrm{T}(7) = 8$
- r ist nächstgrößerer Index, sodass $\mathrm{T}(v_r) \neq \mathrm{T}(v_i)$, also $r = 10$
- $\mathrm{F}(v_{i-1}) < u \leq \mathrm{F}(v_r)$

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Chen und Asau 1974

Hash-basierte Inversionsmethode: Beispiel

u = 0.642

- $I_u = \lfloor u * 10 \rfloor + 1 = 7$
- $i = \mathrm{T}(I_u) = \mathrm{T}(7) = 8$
- r ist nächstgrößerer Index, sodass $\mathrm{T}(v_r) \neq \mathrm{T}(v_i)$, also $r = 10$
- $\mathrm{F}(v_{i-1}) < u \leq \mathrm{F}(v_r)$
- $u < \mathrm{F}(v_9) \rightarrow X = v_9$

X_j	$\mathrm{C_F}(X_j)$	I_j	$\mathrm{T}(I_j)$
v_7	0.592	6	6
v_8	0.634	7	8
v_9	0.698	7	8
v_{10}	0.895	9	10
v_{11}	0.907	10	11
v_{12}	1.000	11	12

Chen und Asau 1974

Vergleich der Invertierungsmethoden

Größe	Methode	Anzahl an generierten Zufallsvariablen					
		200	300	500	1000	3000	7000
5	Standard	0.0623	0.0914	0.1511	0.3062	0.9498	2.1846
	Binäre Suche	0.08493	0.1245	0.2152	0.4230	1.2352	3.01873
	Hash-basiert	0.0826	0.1233	0.2117	0.4296	1.1698	2.48563
50	Standard	0.1834	0.2959	0.5357	0.9973	3.1 698	7.1161
	Binäre Suche	0.1549	0.2245	0.3745	0.7491	2.3863	5.3035
	Hash-basiert	0.0865	0.1288	0.2161	0.4401	1.4096	2.970
100	Standard	0.3465	0.5 196	0.8968	1.8139	5.4206	11.2695
	Binäre Suche	0.17373	0.261 2	0.4399	0.8641	2.4420	6.23313
	Hash-basiert	0.09593	0.1 469	0.2488	0.5282	1.5539	3.51973

Tabelle: Chen und Asau 1974

Vergleich der Invertierungsmethoden

Größe	Methode	Anzahl an generierten Zufallsvariablen					
		200	300	500	1000	3000	7000
5	Standard	0.0623	0.0914	0.1511	0.3062	0.9498	2.1846
	Binäre Suche	0.08493	0.1245	0.2152	0.4230	1.2352	3.01873
	Hash-basiert	0.0826	0.1233	0.2117	0.4296	1.1698	2.48563
50	Standard	0.1834	0.2959	0.5357	0.9973	3.1 698	7.1161
	Binäre Suche	0.1549	0.2245	0.3745	0.7491	2.3863	5.3035
	Hash-basiert	0.0865	0.1288	0.2161	0.4401	1.4096	2.970
100	Standard	0.3465	0.5 196	0.8968	1.8139	5.4206	11.2695
	Binäre Suche	0.17373	0.261 2	0.4399	0.8641	2.4420	6.23313
	Hash-basiert	0.09593	0.1 469	0.2488	0.5282	1.5539	3.51973

Tabelle: Chen und Asau 1974

Vergleich der Invertierungsmethoden

Größe	Methode	Anzahl an generierten Zufallsvariablen					
		200	300	500	1000	3000	7000
5	Standard	0.0623	0.0914	0.1511	0.3062	0.9498	2.1846
	Binäre Suche	0.08493	0.1245	0.2152	0.4230	1.2352	3.01873
	Hash-basiert	0.0826	0.1233	0.2117	0.4296	1.1698	2.48563
50	Standard	0.1834	0.2959	0.5357	0.9973	3.1 698	7.1161
	Binäre Suche	0.1549	0.2245	0.3745	0.7491	2.3863	5.3035
	Hash-basiert	0.0865	0.1288	0.2161	0.4401	1.4096	2.970
100	Standard	0.3465	0.5 196	0.8968	1.8139	5.4206	11.2695
	Binäre Suche	0.17373	0.261 2	0.4399	0.8641	2.4420	6.23313
	Hash-basiert	0.09593	0.1 469	0.2488	0.5282	1.5539	3.51973

Tabelle: Chen und Asau 1974

Beispielcode

function SEARCH_SINGLE(*self*, U)
    $Z \leftarrow \mathit{self}.\mathit{hash}(\mathrm{U})$
    while $\mathit{self}.T[Z] \leq \mathrm{U}$ **do** $Z \leftarrow Z + 1$
    end while
    return $\mathit{self}.\mathit{PS}[Z]$
end function

Knirsch 2021

Kurz & Knapp

- Schnelle Generierung von großen Datenmengen

Kurz & Knapp

- Schnelle Generierung von großen Datenmengen
- Beibehaltung aller Eigenschaften der ursprünglichen Funktion

Kurz & Knapp

- Schnelle Generierung von großen Datenmengen
- Beibehaltung aller Eigenschaften der ursprünglichen Funktion
- einfaches Prinzip

Verbesserungsmöglichkeit

- **Problem:** Funktion mit sehr hoher Dichte an wenigen Stellen, sonst nur geringe Dichte.

Verbesserungsmöglichkeit

- **Problem:** Funktion mit sehr hoher Dichte an wenigen Stellen, sonst nur geringe Dichte.
  $\rightarrow$ Dort fallen alle Punkte zusammen.

Verbesserungsmöglichkeit

- **Problem:** Funktion mit sehr hoher Dichte an wenigen Stellen, sonst nur geringe Dichte.
  $\rightarrow$ Dort fallen alle Punkte zusammen.
  $\rightarrow$ Hohe Ungenauigkeit an wichtigen Stellen.

Verbesserungsmöglichkeit

- **Problem:** Funktion mit sehr hoher Dichte an wenigen Stellen, sonst nur geringe Dichte.
  $\rightarrow$ Dort fallen alle Punkte zusammen.
  $\rightarrow$ Hohe Ungenauigkeit an wichtigen Stellen.
- **Lösung:** Hashtabelle nicht gleichmäßig populieren, sondern für diese Stellen höhere Genauigkeit durch mehr Einträge ermöglichen.

Verbesserungsmöglichkeit

- **Problem:** Funktion mit sehr hoher Dichte an wenigen Stellen, sonst nur geringe Dichte.
  $\rightarrow$ Dort fallen alle Punkte zusammen.
  $\rightarrow$ Hohe Ungenauigkeit an wichtigen Stellen.
- **Lösung:** Hashtabelle nicht gleichmäßig populieren, sondern für diese Stellen höhere Genauigkeit durch mehr Einträge ermöglichen.
- **Allerdings:** stark erhöhter Initialisierungsaufwand

Literatur I

[1] Hui-Chuan Chen und Yoshinori Asau. „On Generating Random Variates from an Empirical Distribution“. In: *A I I E Transactions* 6.2 (1974), S. 163–166. DOI: 10.1080/05695557408974949. eprint: https://doi.org/10.1080/05695557408974949. URL: https://doi.org/10.1080/05695557408974949.

[2] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer New York, 1986.

[3] Daniel Frisch und Uwe D. Hanebeck. „Deterministic Gaussian Sampling With Generalized Fibonacci Grids“. In: *Proceedings of the 24th International Conference on Information Fusion (Fusion 2021)*. South Africa, Nov. 2021.

[4] Lukas Knirsch. *Punkt-Warping: Inversionsmethode mit Hash-Tabelle*. GitHub-Repository. 2021. URL: https:github.com/knirschl/Proseminar-Anthropomatik.

[5] Colas Schretter, Leif Kobbelt und Paul-Olivier Dehaye. „Golden Ratio Sequences for Low-Discrepancy Sampling“. In: *Journal of Graphics Tools* 16 (Juni 2012), S. 9. DOI: 10.1080/2165347X.2012.679555.

Literatur II

[6] Tien-Tsin Wong, Wai-Shing Luk und Pheng-Ann Heng. „Sampling with Hammersley and Halton Points“. In: *Journal of Graphics Tools* 2.2 (1997), S. 9–24. DOI: `10.1080/10867651.1997.10487471`. eprint: `https://doi.org/10.1080/10867651.1997.10487471`. URL: `https://doi.org/10.1080/10867651.1997.10487471`.

[7] Dong-Ming Yan u. a. „A survey of blue-noise sampling and its applications“. In: *Journal of Computer Science and Technology* 30.3 (2015), S. 439–452.

Vielen Dank für Ihre Aufmerksamkeit!