Abschlusspräsentation Wissenschaftliches Individualprojekt:

„Evaluation of Point Cloud Geometry Compression on Sparse and Non-uniform Data“

1 Einleitung und Motivation

- Punktwolken sind flexibel, vielseitige Anwendungsfälle
  - zB. Autonomes fahren, AR/VR, Ditigales Modellieren
- Größe der übertragenen Daten wächst schnell mit Aufnahmedauer
  - z.B. FIN Hörsaal: ~ 2M Punkte nach 300s

→ Kompression der Punktwolken notwendig

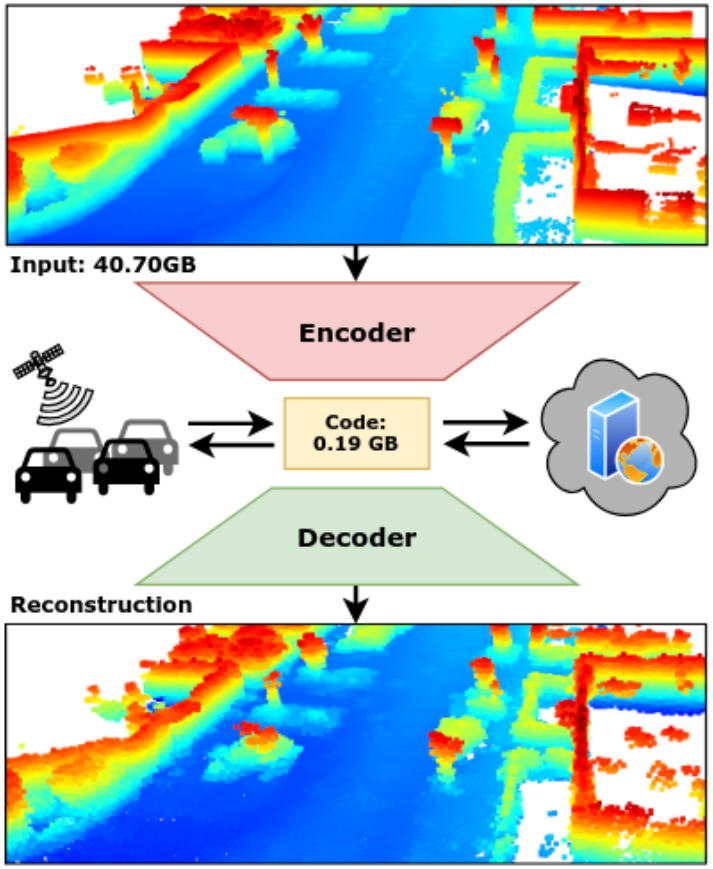

Fig. 1: Kompressionsworkflow [1]

1.1 Punktwolkenkompression

- Übliche Datensätze:
  - Kitti[1][2], Shapenet[2][3], 8iVFB[2][4]
- Lücke in aktueller Literatur:
  - Dünnbesetzte, ungleichmässig dichte Daten von Innenräumen

Fig. 2: Oben nach unten: Kitti[2],Shapenet[3], 8iVFB[4]

[1] Dünnbesetzt, Outdoor
[2] Dichtbesetzt, Objekte

2 Ziel des Projekts

- Evaluierung von Punktwolkenkompression auf bisher unerforschter Art von Daten
  - Einheitlicher Vergleich von Algorithmen
  - „Welcher Algorithmus ist am besten geeignet?“
- Fokus:
  - Innenräume
  - Dünnbesetzt und ungleichmässige Dichte
  - Veröffentlichter Quellcode
  - verlustfreie („lossless“) Algorithmen

3 Konzept

- Kompression sollte zeitlich effizient sein
- Schnelle Übermittlung, aber langsame Kompression ist nicht zielführend
- Rekonstruierte Punktwolke sollte möglichst ähnlich sein
- **Problem**:
  - Bisher wurde keine Evaluation auf dieser Art Daten ausgeführt

  $\rightarrow$ Auswahl des Algorithmus nicht trivial

3 Konzept

- Erster, einheitlicher Vergleich von Kompressionsalgorithmen auf dieser Art von Daten
- Dafür benötigt:
  1. Auswahl von Algorithmen
  2. Auswahl des Datensatzes
  3. Bewertungsmetriken

3.1 Algorithmen in der Literatur

- Quellcode muss verfügbar sein
  - KI-basierte Verfahren: Inkl. Vortrainiertes Modell
- Muss auf Punktwolken im `.ply` Dateiformat mit 32bit Präzision arbeiten

Algorithmus	Quellcode (& Modell) vorhanden?	Dateiformat	Präzision
draco[5]	✓	PLY, OBJ, STL	float32
pccomp[6]	✓	PLY, OBJ, STL,PCD	Numpy[3]
tmc3[7]	✓	PLY	PLY[3]
sparsePCGC[8]	✓	PLY, H5, BIN	float32
DPCC[9]	×	PLY, XYZ, BIN	float32
Unicorn[10], [11]	×	PLY, H5, BIN	float32
Depoco[1]	×	PLY, BIN	float32
Mpeg Anchor[12]	×	?	?

[3]Alle Präzisionen (inkl. float32)

3.1.1 Auswahl der Algorithmen

- Exkludiert:
  - `DPCC`:
    - Trainieren des Modells notwendig
    - Datenvorbereitung erwartet KITTI oder ShapeNet
  - `Unicorn`:
    - Kein Quellcode, keine Modelle
  - `Depoco`:
    - Trainieren des Modells notwendig
  - `Mpeg Anchor`:
    - Kein Quellcode
- Inkludiert: `draco, pccomp, tmc3, sparsePCGC`

3.1.2 Benutzung der ausgewählten Algorithmen

- `Draco`:
  - Binaries aus Evaluationsskript aufrufen
- `pccomp`:
  - Python Funktionen aus Evaluationsskript aufrufen
- `tmc3`:
  - Binaries aus Evaluationsskript aufrufen
  - Benutzter Datensatz zu klein, setze `--inputScale=100` und `--outputUnitLength=100`
- `sparsePCGC`:
  - Docker Entwicklungsumgebung der Autoren
  - Benutzter Datensatz zu klein, manuelle Skalierung nötig (open3D[13])
  - Verschieben des Mittelpunktes

3.2 Ausgewählter Datensatz: FIN[14]

- Aufnahmen in der Fakultät für Informatik der OvGU
- Vier Szenen:
  - Hörsaal (307)
  - Flur
  - Büro (425)
  - Konferenzraum (333)
- Konvertierung: `.pcd` → `.ply` und 64bit → 32bit

Fig. 3: FIN Datensatz, Hörsaal Szene [14]

4 Evaluierung

- Hardware:
  - Intel Core i7-12700KF CPU
  - Nvidia 4070 RTX GPU
  - 32 GB DDR5 RAM
- Bewertungsmetriken
  - Berechnungsdauer
    - Enkodierung, Dekodierung

$$t_{\text{enc/dec}} = \frac{\text{Berechnungszeit}}{|\text{Points in Original Cloud}|}$$

  - Kompressionsrate bpp (bits per point)

$$\text{bpp} = \frac{\text{Encoded File size (bits)}}{|\text{Points in Original Cloud}|}$$

  - Enkodierte Dateigröße
  - Qualität der Rekonstruierten Punktwolke
    - Strukturelle Ähnlichkeitsmetrik `PointSSIM` [15]

4.1 Strukturelle Ähnlichkeitsmetrik `PointSSIM`[4]

- Feature Extrahierung:
  - Nutze lokale Nachbarschaften um jeden punkt
  - Quantitäten: Euklidische Distanz & Krümmung
  - Wende Streuungsschätzer[5] auf Quantitäten an
- Strukturelle Ähnlichkeit:
  - Berechne relativen Unterschied zwischen Features
  - Berechne Gesamtwert über Fehler Pooling:

$$S_Y = \frac{1}{N} \sum_{p=1}^{N} S_Y(p)$$

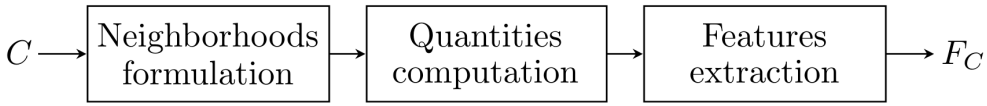

Fig. 4: Feature Extrahierung [15, Fig. 1]

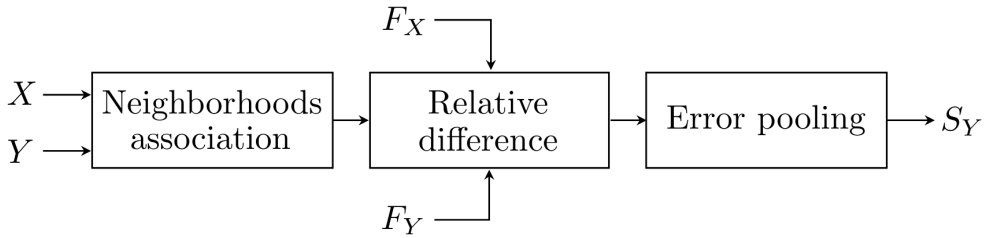

Fig. 5: Strukturelle Ähnlichkeitsberechnung [15, Fig. 2]

[4] https://github.com/mmspg/pointssim
[5] u.a. median, Varianz, MAD

4.2 Kompressionsrate & Berechnungszeiten

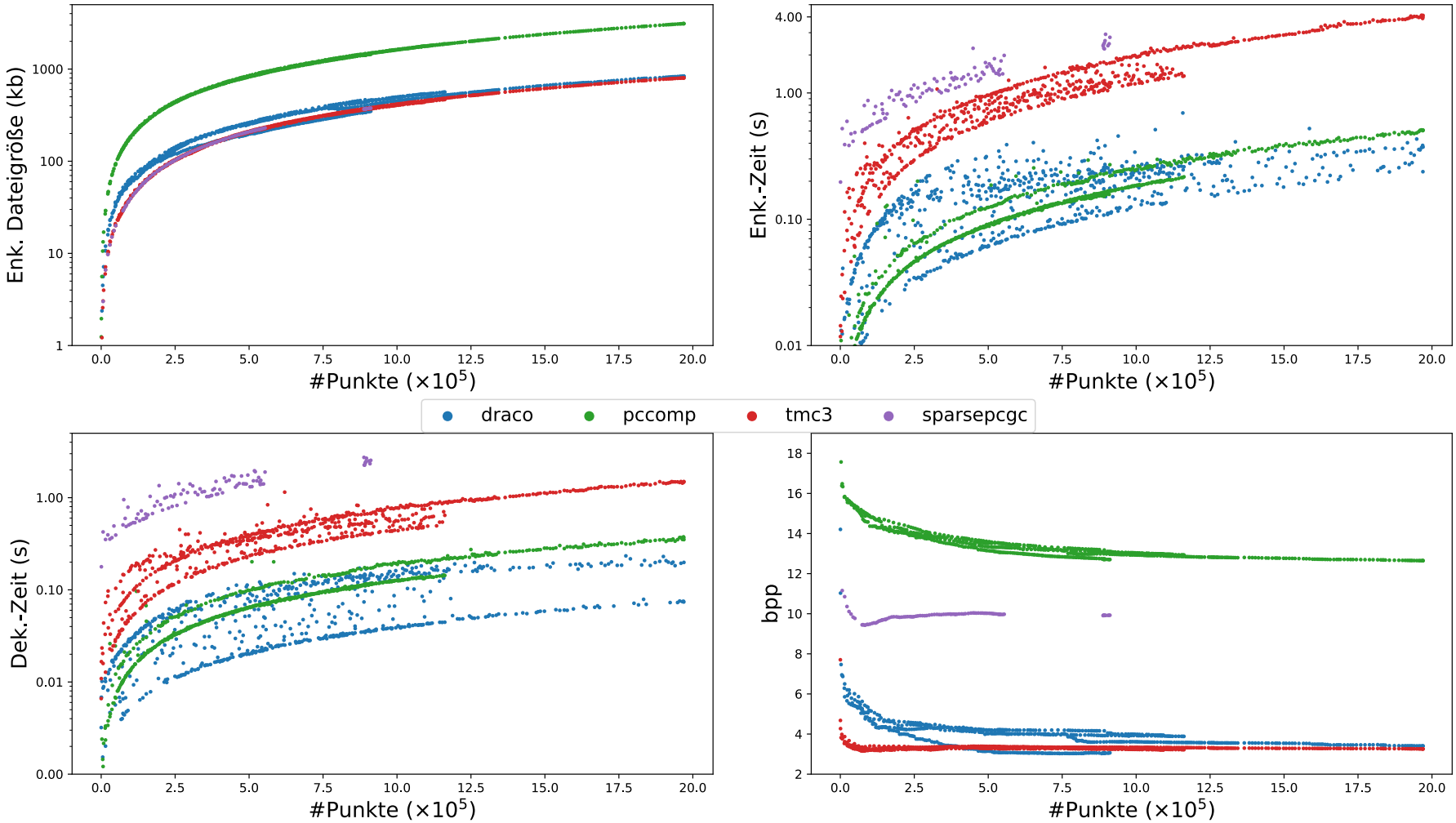

Fig. 6: Kompressionsrate & Berechnungszeiten

4.3 Strukturelle Ähnlichkeit

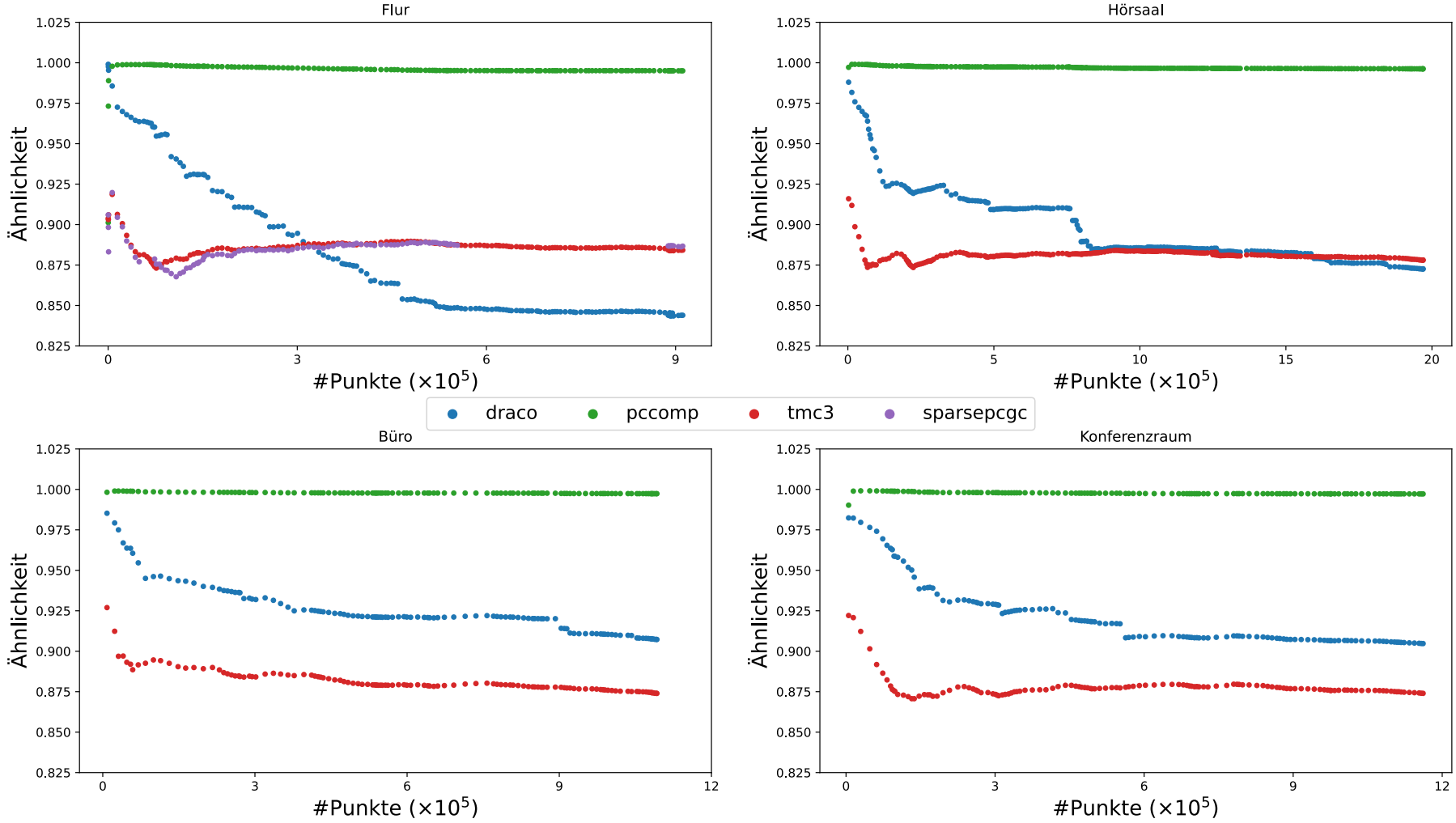

Fig. 7: Strukturelle Ähnlichkeit

5 Fazit

- `tmc3`, `sparsePCGC`:
  - Im Vergleich deutlich langsamer
  - niedrigste Ähnlichkeit
  - Datensatzskalierung ist ein Faktor
- `pccomp`:
  - Höchste Ähnlichkeit, geringste Berechnungszeiten
  - **aber**: geringste Kompressionsrate
- `draco`:
  - Knapp schlechter als `pccomp` in Ähnlichkeit und Berechnungszeit
  - **aber**: deutlich bessere Kompressionsrate
- `draco` überzeugt insgesamt am meisten
- Quellcode und Ergebnisse: https://github.com/lupeterm/pc-compression

Algorithmus	PSSIM	bpp	Enk.-Zeit[1]	Dek.-Zeit[1]
`tmc3`	0.882	**3.31**	1925ns	938ns
`sparsePCGC`	0.885	9.86	6238ns	5780ns
`pccomp`	**0.997**	13.54	**281ns**	**173ns**
`draco`	0.903	4.025	396ns	220ns

[1]Durchschnittliche Zeit pro Punkt

References

[1] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, und J. Behley, „Deep Compression for Dense Point Cloud Maps“, *IEEE Robotics and Automation Letters (RA-L)*, Bd. 6, Nr. 2, S. 2060–2067, 2021, doi: 10.1109/LRA.2021.3059633.

[2] J. Behley *u. a.*, „SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences“, in *Proc. of the IEEE/CVF International Conf.~on Computer Vision (ICCV)*, 2019.

[3] A. X. Chang *u. a.*, „ShapeNet: An Information-Rich 3D Model Repository“, *ArXiv*, 2015, [Online]. Verfügbar unter: https://api.semanticscholar.org/CorpusID:2554264

[4] „Pleno Database: 8i Voxelized Full Bodies (8iVFB v2) - A Dynamic Voxelized Point Cloud Dataset“. [Online]. Verfügbar unter: http://plenodb.jpeg.org/pc/8ilabs

[5] google, „GitHub - google/draco: Draco is a library for compressing and decompressing 3D geometric meshes and point clouds. It is intended to improve the storage and transmission of 3D graphics.“. [Online]. Verfügbar unter: https://github.com/google/draco

[6] Szppaks, „GitHub - szppaks/pccomp_oct: Octree-based lossy point-cloud compression with open3d and numpy“. [Online]. Verfügbar unter: https://github.com/szppaks/pccomp_oct

[7] MPEGGroup, „GitHub - MPEGGroup/mpeg-pcc-tmc13: Geometry based point cloud compression (G-PCC) test model“. [Online]. Verfügbar unter: https://github.com/MPEGGroup/mpeg-pcc-tmc13

[8] D. Ding, Z. Li, X. Feng, C. Cao, und Z. Ma, „Sparse Tensor-based Multiscale Representation for Point Cloud Geometry Compression“, 2022, doi: 10.48550/arXiv.2111.10633.

[9] Y. He, X. Ren, D. Tang, Y. Zhang, X. Xue, und Y. Fu, „Density-preserving Deep Point Cloud Compression“, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[10] J. Wang, R. Xue, J. Li, D. Ding, Y. Lin, und Z. Ma, „A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding – Part I: Geometry“, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 47, Nr. 1, 2025, doi: 10.1109/TPAMI.2024.3462938.

[11] J. Wang, R. Xue, J. Li, D. Ding, Y. Lin, und Z. Ma, „A Versatile Point Cloud Compressor Using Universal Multiscale Conditional Coding – Part II: Attribute“, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 47, Nr. 1, 2025, doi: 10.1109/TPAMI.2024.3462945.

[12] R. Mekuria, K. Blom, und P. Cesar, „Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video“, *IEEE Transactions on Circuits and Systems for Video Technology*, Bd. 27, Nr. 4, S. 828–842, 2017, doi: 10.1109/TCSVT.2016.2543039.

[13] Q.-Y. Zhou, J. Park, und V. Koltun, „Open3D: A Modern Library for 3D Data Processing“. [Online]. Verfügbar unter: https://arxiv.org/abs/1801.09847

[14] L. Petermann und F. Ortmeier, „Comparison of Real-Time Plane Detection Algorithms on Intel RealSense“, 2023. [Online]. Verfügbar unter: https://api.semanticscholar.org/CorpusID:259343618

[15] E. Alexiou und T. Ebrahimi, „Towards a Point Cloud Structural Similarity Metric“, in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 2020, S. 1–6. doi: 10.1109/ICMEW46912.2020.9106005.

[16] I. Armeni, S. Sax, A. R. Zamir, und S. Savarese, „Joint 2D-3D-Semantic Data for Indoor Scene Understanding“. [Online]. Verfügbar unter: https://arxiv.org/abs/1702.01105

[17] E. Alexiou, I. Viola, T. M. Borges, T. A. Fonseca, R. L. de Queiroz, und T. Ebrahimi, „A comprehensive study of the rate-distortion performance in MPEG point cloud compression“, *APSIPA Transactions on Signal and Information Processing*, Bd. 8, Nr. 1, S. –, 2019, doi: 10.1017/ATSIP.2019.20.

Danke für ihre Aufmerksamkeit!

Fragen?

Appendix

Vergleich: Datensatz Dichte

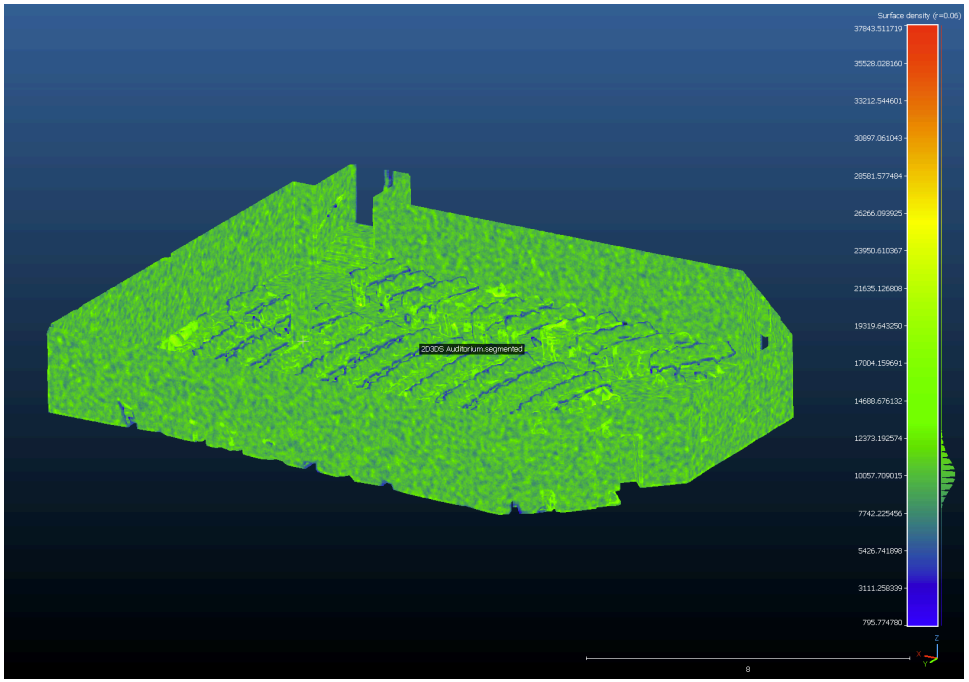

Fig. 8: 2D3DS [16] Hörsaal Dichte

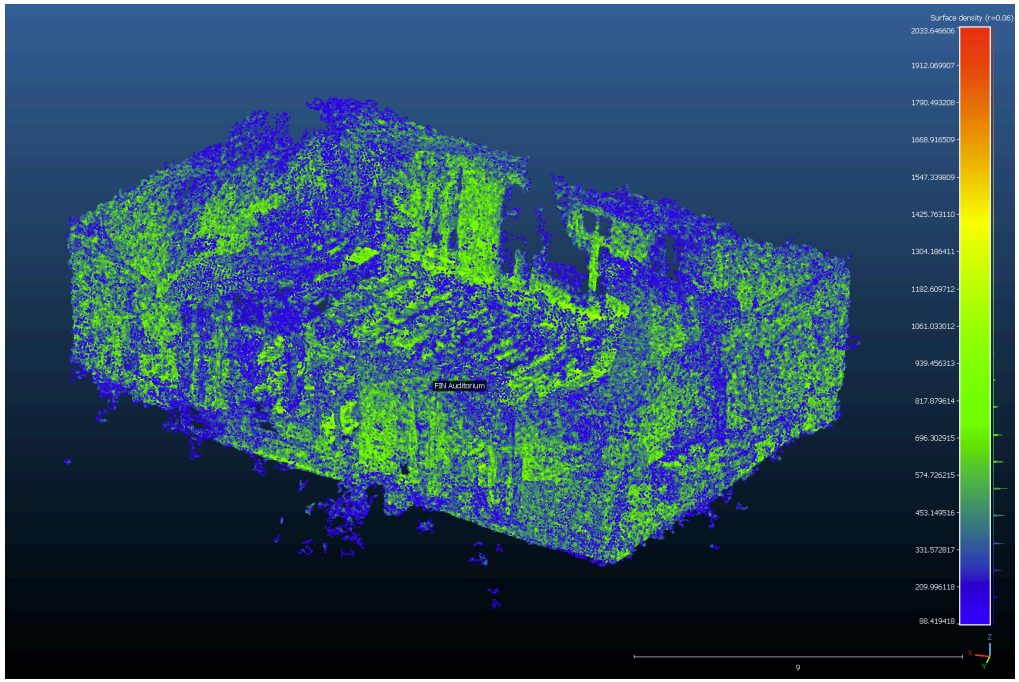

Fig. 9: FIN [14] Hörsaal Dichte

PointSSIM Gleichungen

$$\mu\mathrm{AD}_A = E(A - \mu_A)$$

$$m\mathrm{AD}_A = E(A - m_A)$$

$$\mathrm{COV}_A = \frac{\sigma_A}{\mu_A}$$

$$\mathrm{QCD}_A = \frac{Q_A(3) - Q_A(1)}{Q_A(3) + Q_A(1)}$$

Fig. 10: Streuungsschätzer [15, Eqs.1-4]

$$S_Y(p) = \frac{|\, F_X(q) - F_Y(p) \,|}{\max\{|\, F_X(q) \,|, |\, F_Y(p) \,|\} + \varepsilon}$$

Fig. 11: Berechnung Relativer Fehler der Features [15, Eq. 5]

$$S_Y = \frac{1}{N_p} \sum_{p=1}^{N_p} S_Y(p)^k$$

Fig. 12: Berechnung Fehler der Ganzen Punktwolke. $k \in \{1, 2\}$ [15, Eq. 6]

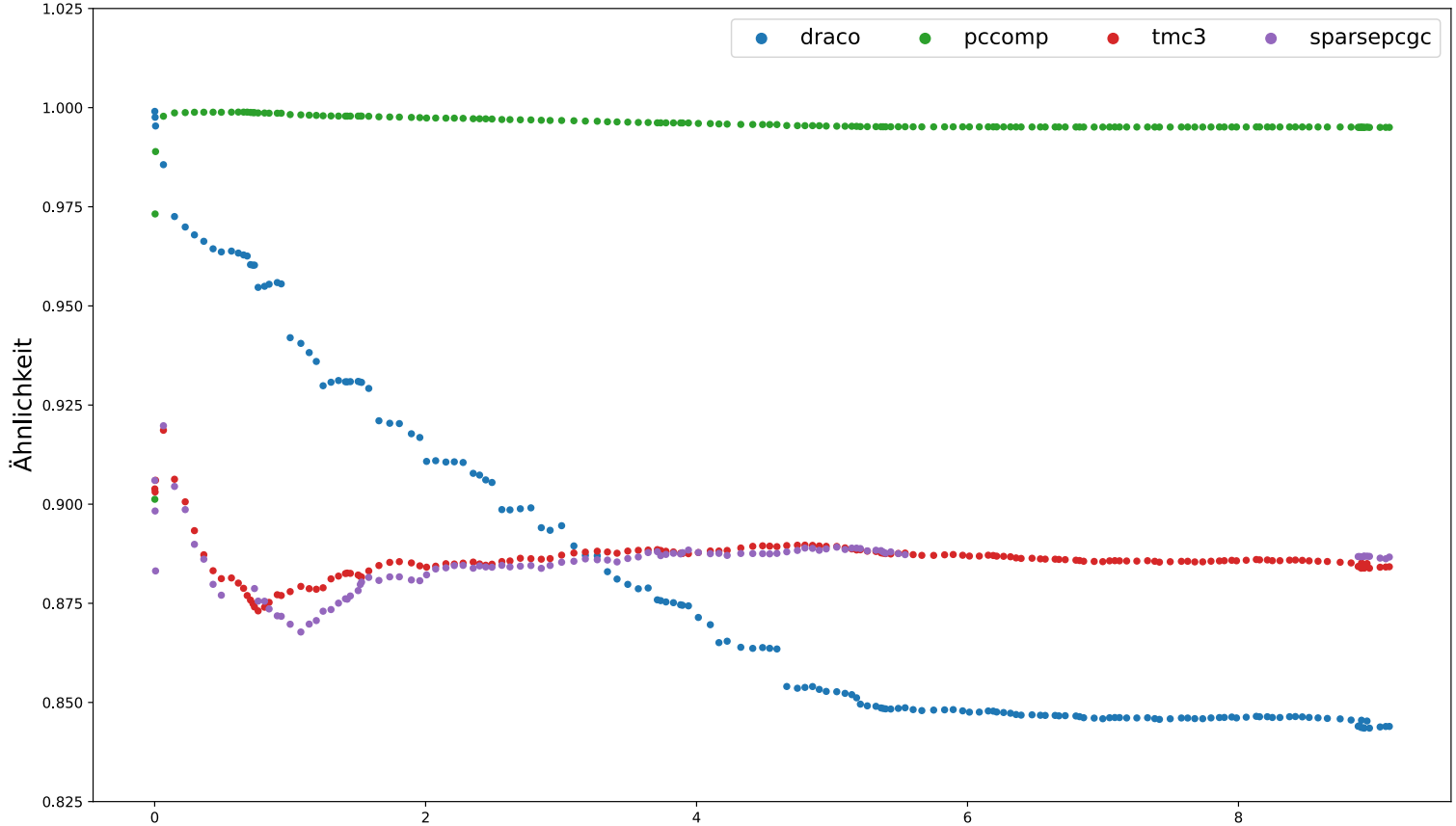

Fig. 13: Ähnlichkeitswert Flur

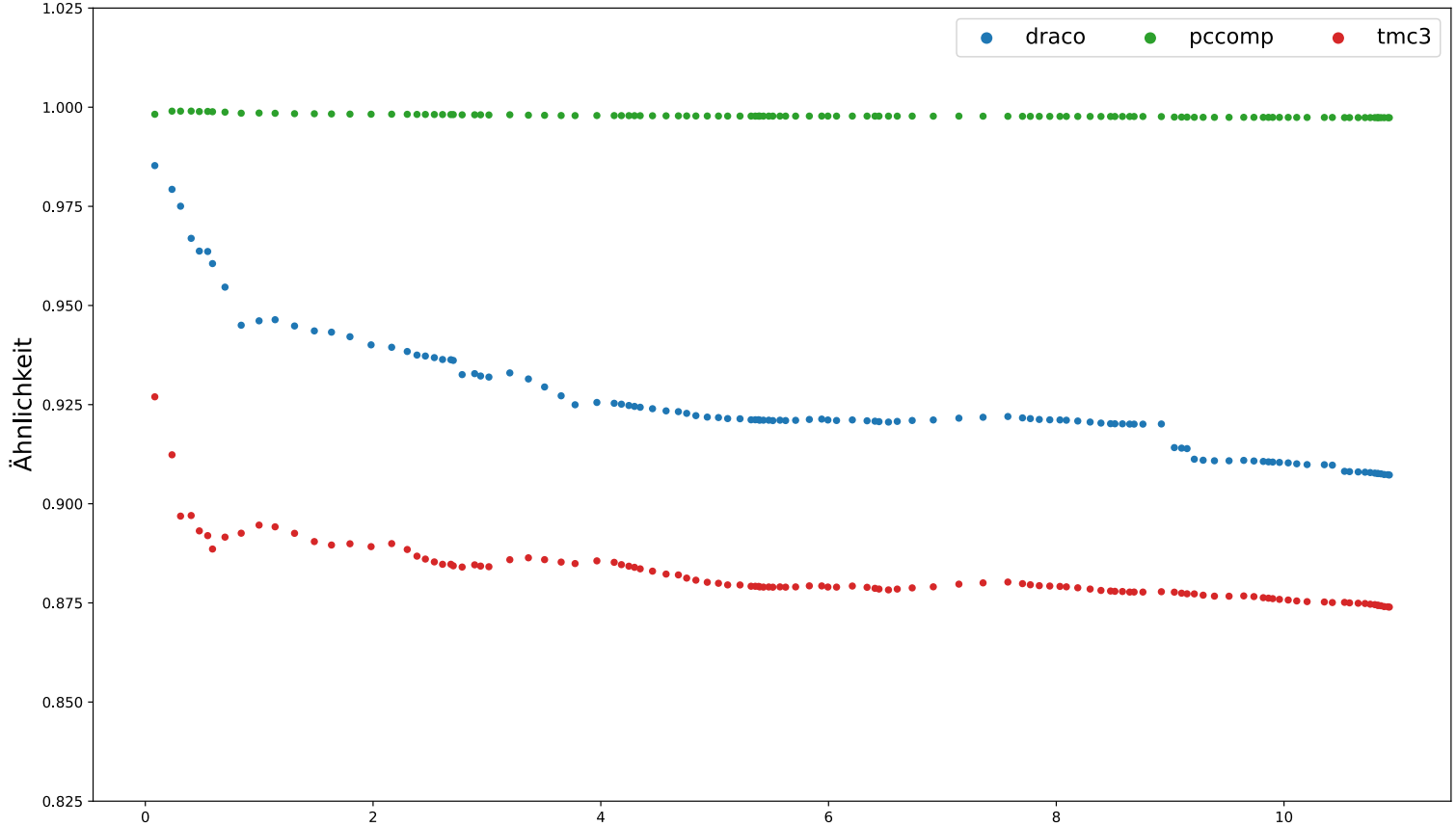

Fig. 14: Ähnlichkeitswert Büro

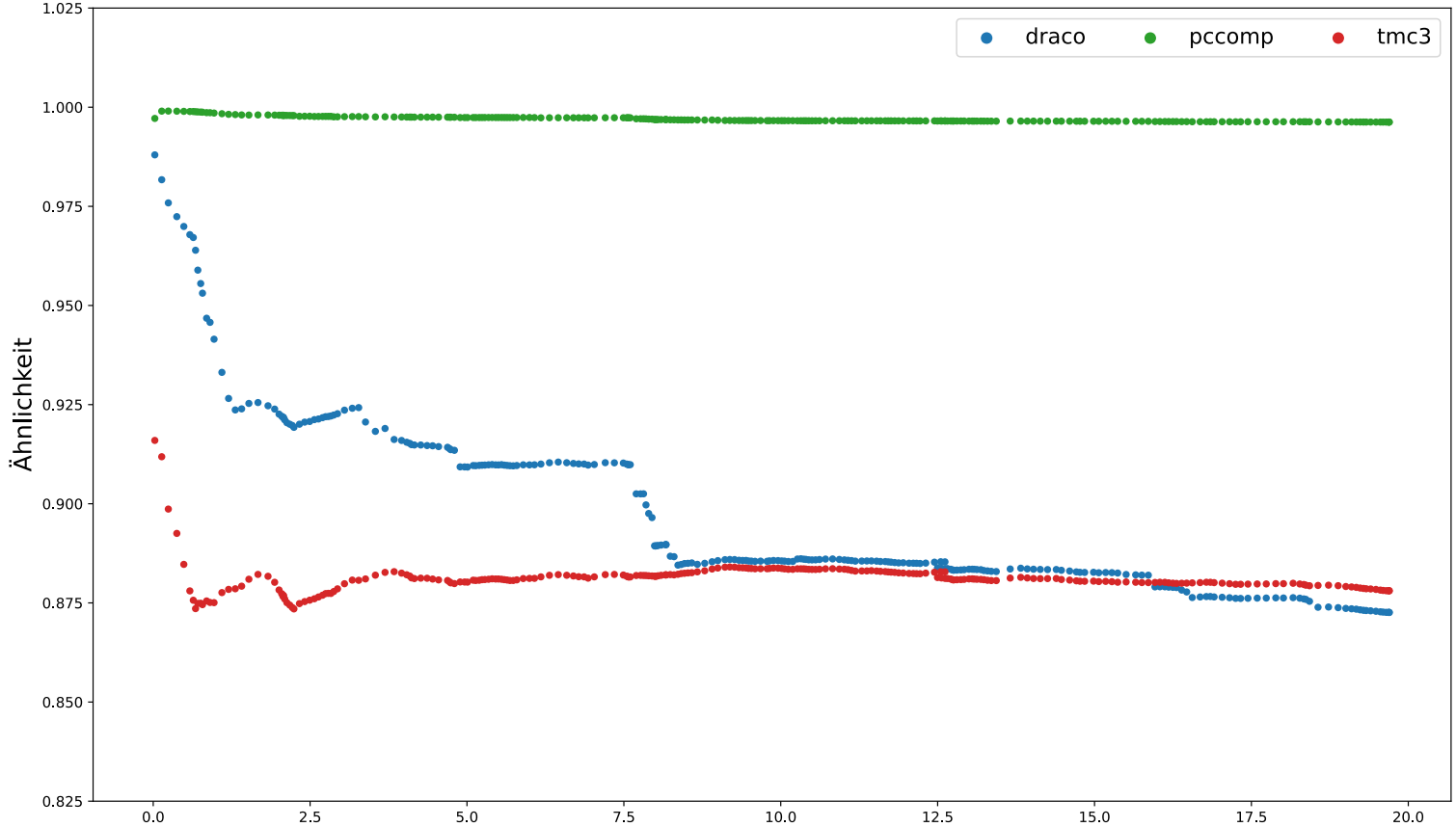

Fig. 15: Ähnlichkeitswert Hörsaal

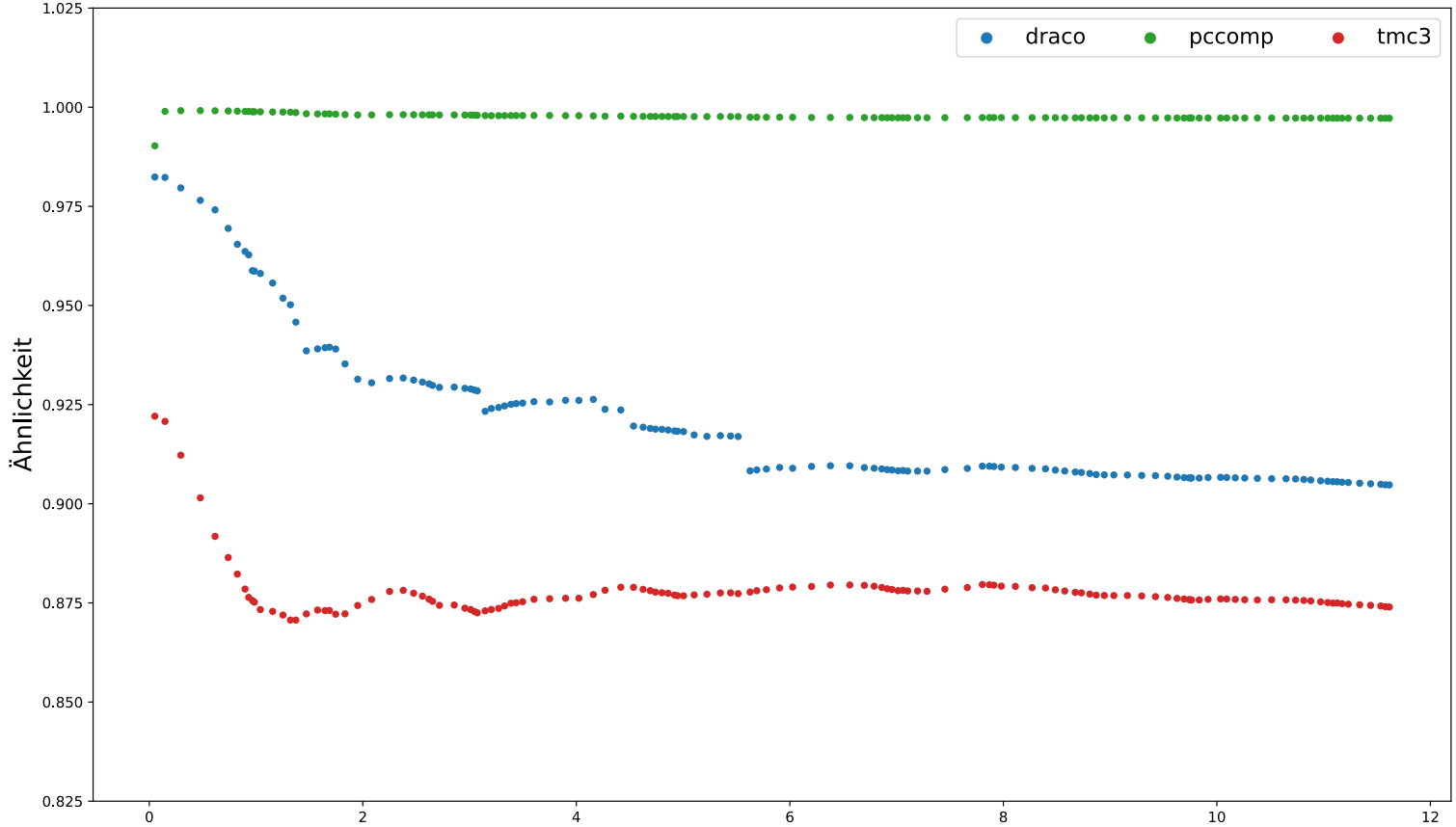

Fig. 16: Ähnlichkeitswert Konferenzraum

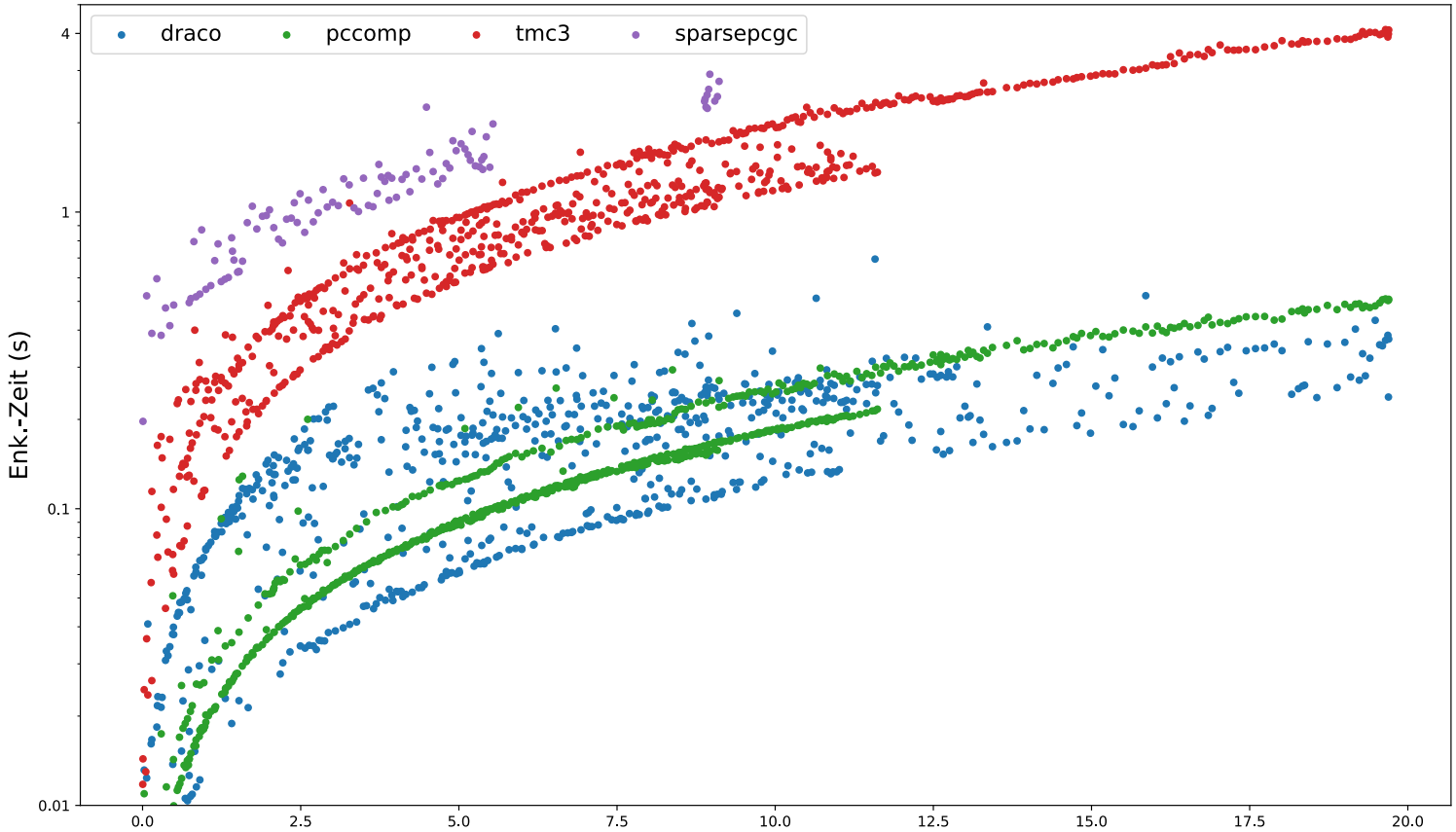

Fig. 17: Enkodierungszeit

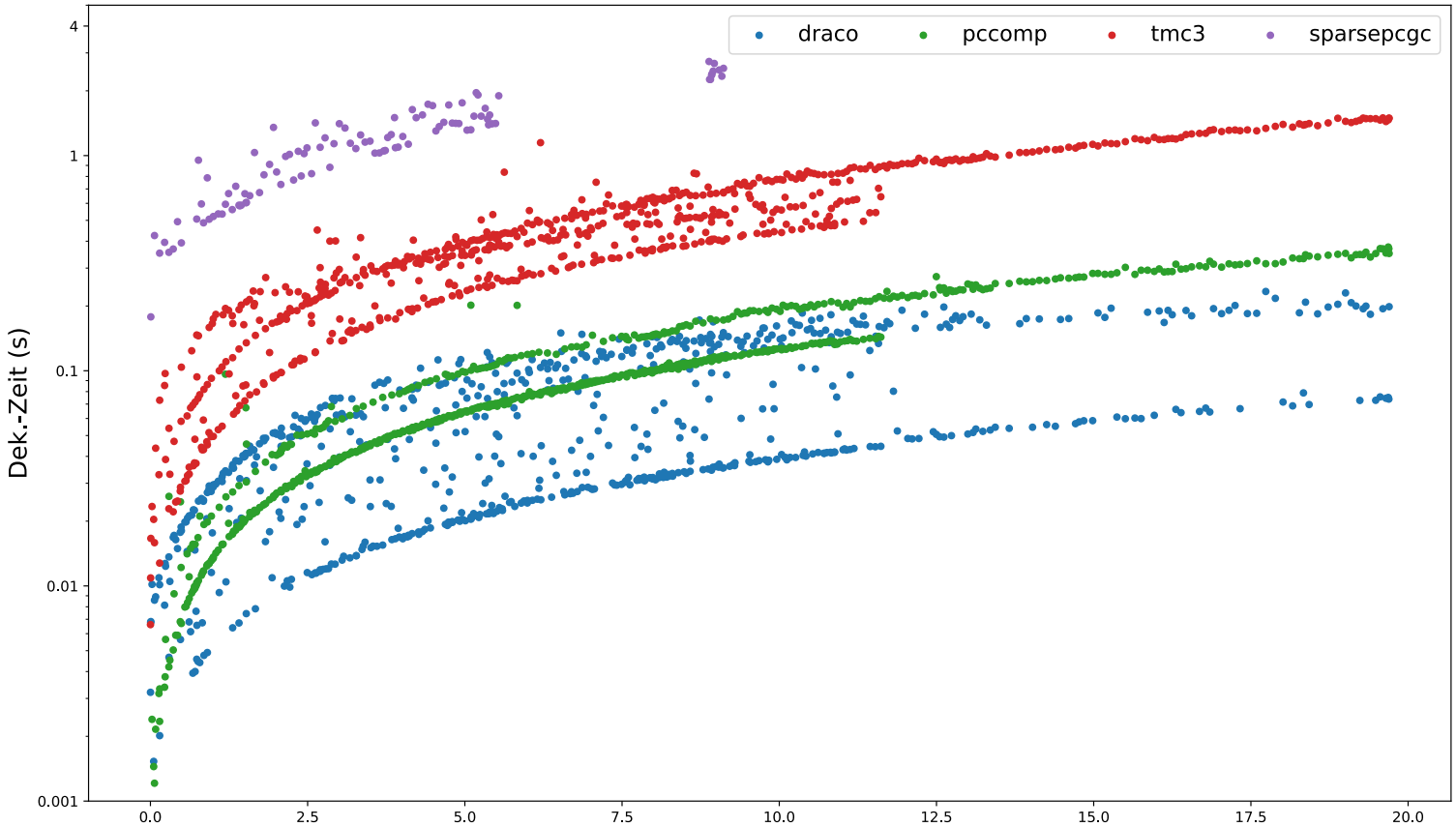

Fig. 18: Dekodierungszeit

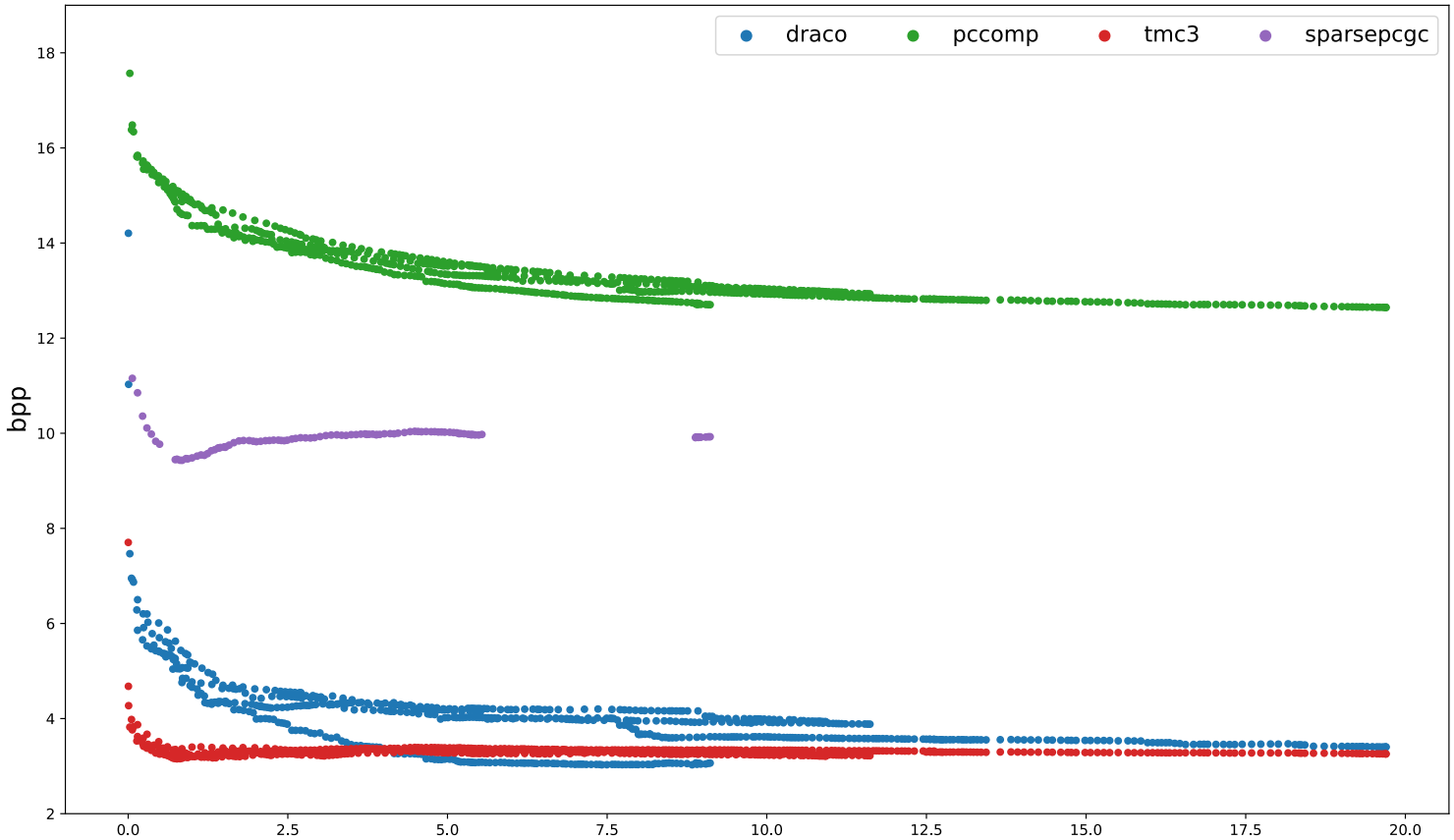

Fig. 19: Bits pro Punkt

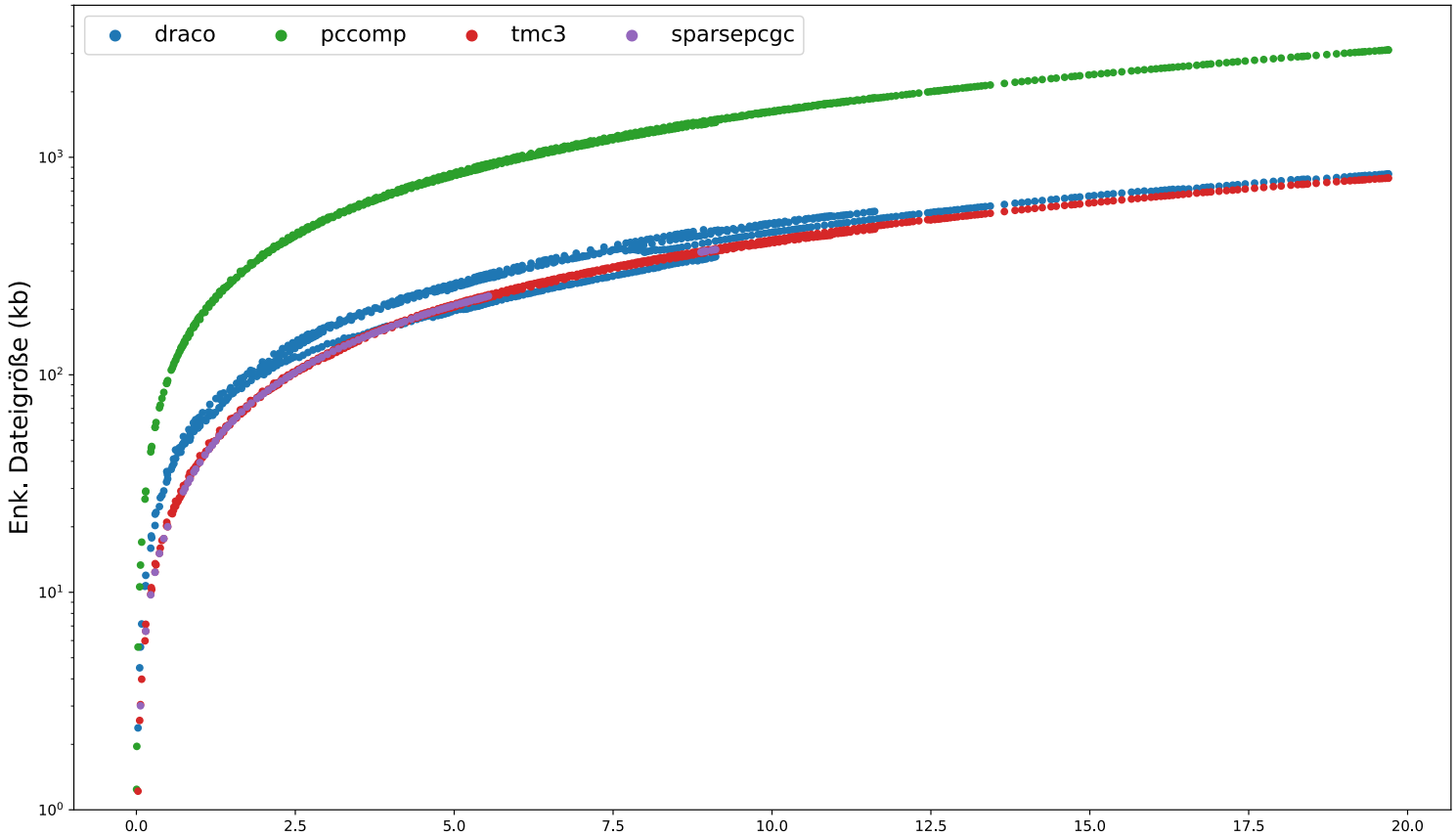

Fig. 20: Enk. Dateigröße