



(10) **DE 11 2022 003 547 T5** 2024.05.29

(12) **Veröffentlichung**

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2023/044001**
in der deutschen Übersetzung (Art. III § 8 Abs. 2
IntPatÜbkG)
(21) Deutsches Aktenzeichen: **11 2022 003 547.1**
(86) PCT-Aktenzeichen: **PCT/US2022/043800**
(86) PCT-Anmeldetag: **16.09.2022**
(87) PCT-Veröffentlichungstag: **23.03.2023**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **29.05.2024**

(51) Int Cl.: **G06T 17/20** (2006.01)

(30) Unionspriorität:
63/245,155 **16.09.2021** **US**

(71) Anmelder:
NVIDIA Corporation, Santa Clara, CA, US

(74) Vertreter:
Kraus & Lederer PartGmbB, 80539 München, DE

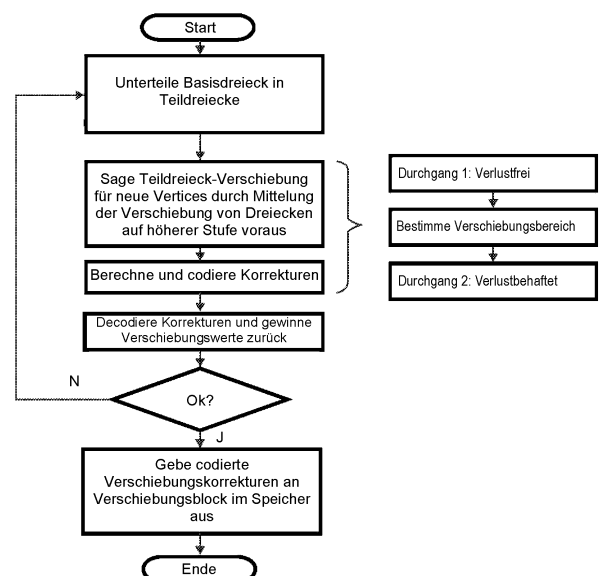
(72) Erfinder:
**Salvi, Marco, Santa Clara, CA, US; Moreton, Henry,
Santa Clara, CA, US; Bickford, Neil, Santa Clara,
CA, US; Muthler, Gregory, Santa Clara, CA, US**

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Verschobene Mikronetz-Kompression**

(57) Zusammenfassung: Ein Algorithmus und ein zugehöriger Satz von Regeln ermöglichen es, dass ein gegebener Polygon-Mikronetztyp immer in der Lage ist, einen stärker komprimierten Mikronetztyp darzustellen. Diese Regeln in Verbindung mit zusätzlichen Beschränkungen für die Reihenfolge, in der die verschobenen Mikronetze codiert werden, ermöglichen verlustbehaftete Kompressionstechniken zur effizienten Speicherung geometrischer Verschiebungen als paralleler Algorithmus, bei dem nur wenig Kommunikation zwischen unabhängig komprimierten verschobenen Mikronetzen erforderlich ist, während gleichzeitig qualitativ hochwertige, wasserdichte (rissfreie) Ergebnisse für Vektorverschiebungen, Dreieckstexturen sowie Strahlen- und Pfadverfolgung sichergestellt werden.



Beschreibung

Querverweis auf verwandte Anwendungen

[0001] Diese Anmeldung beansprucht Priorität für die am 16. September 2021 eingereichte vorläufige US-Patentanmeldung Nr. 63/245,155, deren gesamter Inhalt hier durch Bezugnahme aufgenommen ist,

[0002] Diese Anmeldung steht im Zusammenhang mit den folgenden Patentanmeldungen, die sich im gemeinsamen Besitz befinden und von denen jede durch Bezugnahme für alle Zwecke einbezogen wird, als ob sie hier ausdrücklich aufgeführt wäre:

[0003] US-Patentanmeldung Nr. 17946235 eingereicht am 16.9.2022 mit dem Titel Micro-Meshes, A Structured Geometry For Computer Graphics (21-SC-1926US02; 6610-126)

[0004] US-Patentanmeldung Nr. 17946221, eingereicht am 16.9.2022 mit dem Titel Accelerating Triangle Visibility Tests For Real-Time (22-DU-0175US01; 6610-124)

[0005] US-Patentanmeldung Nr. xxxxxx eingereicht am 16.9.2022 mit dem Titel Displaced Micro-meshes for Ray and Path Tracing (22-AU-0623US01/6610-125).

Gebiet

[0006] Die vorliegende Technologie betrifft die Kompression von Polygonnetz-Verschiebungsdaten für Computergrafiken, einschließlich, aber nicht beschränkt auf Strahlen- und Pfadverfolgung. Die vorliegende Technologie bietet einen benutzerdefinierten Kompressionsalgorithmus zur Erzeugung qualitativ hochwertiger, rissfreier Verschiebungs-Mikronetze („DMMs“) für Computergrafiken, der gleichzeitig schnell genug ist, um dynamische Inhalte in modernen Echtzeitanwendungen zu verarbeiten

[0007] Insbesondere betrifft die hier beschriebene Technologie ein Verfahren zur Berechnung einer komprimierten Darstellung von dichten Dreiecksnetzen, z. B. für die Strahlenverfolgung, und zur Verwendung verlustbehafteter Kompressionstechniken, um geometrische Verschiebungen von Polygonnetzen, z. B. für die Strahlen- und Pfadverfolgung, effizienter zu speichern und gleichzeitig die Wasserdichtigkeit zu erhalten.

Hintergrund & Zusammenfassung

[0008] Da die Genauigkeit des Grafik-Renderings zugenommen hat und die Grafikindustrie große Fortschritte bei der Modellierung des Verhaltens von Licht und seiner Interaktion mit Objekten in virtuellen Umgebungen gemacht hat, besteht nun eine große Nachfrage nach sehr detaillierten, realistischeren virtuellen Umgebungen. Dies hat zu einem enormen Anstieg der Menge an Geometrie geführt, die Entwickler modellieren und abbilden möchten. Die Speicherbandbreite ist jedoch nach wie vor ein Engpass, der die Menge der Geometrie begrenzt, die die Grafikhardware für das Rendering aus dem Speicher holen kann.

[0009] In der Vergangenheit haben Tessellierungs-Shader das Problem der Speicherbandbreite gelöst, indem sie - im laufenden Betrieb - ein Polygonnetz (siehe **Fig. 1A, 1B**) ohne Überlappungen und ohne Lücken zwischen den geometrischen Formen oder Polygonen erzeugten, um eine zu rasternde und zu rendernde Oberfläche mit einem gewünschten Detailgrad abzudecken. Siehe z. B., Lee et al, „Displaced subdivision surfaces“, SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques July 2000 Pages 85-94 //doi.org/10.1145/344779.344829; Cantlay, „DirectX 11 Terrain Tessellation“, Nvidia (Januar 2011); khronos.org/opengl/wiki/Tessellation#Tessellation_control_shader; Moreton et al, (2001); Moreton, Tessellation and Geometry Shaders: Trends CMU 15-869 (Nvidia Corp. 2011); US 10825230; US 9437042; US 8860742; US 8698802; US 8570322; US 8558833; US 8471852; US 20110085736; US 7324105; US 7196703; US 6,597,356; US 6738062; US 6504537; Dudash, „My Tessellation Has Cracks!“, Game Developer's Conference (2012); Sfarti et al, „New 3D Graphics Rendering Engine Architecture for Direct Tessellation of Spline Surfaces“, V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 224-231 (2005); N. Pietroni et al, „Almost Isometric Mesh Parameterization through Abstract Domains“, IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 4, pp. 621-635, July-Aug. 2010, doi: 10.1109/TVCG.2009.96. Bereits 2001 wurde im NVIDIA GeForce3 ein Oberflächentessellator in Hardware implementiert, der eine garantiert wasserdichte Tessellierung und unterschiedliche Detaillierungsgrade (LOD) ohne Überlappung ermöglichte.

[0010] Ein solches tesselliertes Netz wird als „wasserdicht“ bezeichnet, wenn es keine Lücken zwischen den Polygonen gibt. Das Netz ist nicht „wasserdicht“, wenn - unter der Annahme, dass das Netz ein echtes, in Wasser getauchtes Objekt wäre - Wasser durch Nähte oder Löcher zwischen den geometrischen Formen oder Polygonen, die das Netz bilden, eindringen würde. Selbst winzige Lücken zwischen Polygonen können zu fehlenden Pixeln führen, die in einem gerenderten Bild zu sehen sind. Siehe beispielsweise **Fig. 2A**.

[0011] Eine Ursache für solche Lücken war die Durchführung von Gleitkommaoperationen in unterschiedlicher Reihenfolge, was nicht immer zu den gleichen Ergebnissen führte. Leider kann die Anordnung von Shader-Berechnungen, um sie für benachbarte Patches identisch zu machen, eine Menge Leistung kosten. T-Sprünge - ein weiteres wasserdichtes Tessellierungsproblem - treten auf, wenn ein Patch geteilt wird, obwohl eine oder mehrere seiner Kanten flach sind. Wenn das Feld, das die flachen Kanten teilt, nicht auf die gleiche Weise geteilt wird, entsteht ein Riss. Siehe **Fig. 2B**; und siehe auch Bunnell, Kapitel 7. Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping, GPU Gems 2 (NVIDIA 2005).

[0012] Es war bekannt, dass Risse und Pixelausfälle durch unterschiedliche Tessellierungsgrade, durch die Bildung von T-Verbindungen, durch Berechnungsprobleme und aus anderen Gründen entstehen. Da jedes praktische System die Lage eines bestimmten Vertex mit endlicher Genauigkeit darstellt, liegen die Vertices (für die detaillierte Berechnungs- und Verarbeitungshardware) nicht immer genau auf angrenzenden Segmenten zwischen Polygonen. Obwohl dieses Problem durch die geringere Genauigkeit einiger Hardware-Rasterizer und anderer Grafikhardware verschärft werden kann, besteht es bei jeder Darstellung mit endlicher Genauigkeit, einschließlich IEEE-Gleitkomma.

[0013] Bisherige Ansätze erforderten oft die Lösung eines komplexen globalen Optimierungsproblems, um die Qualität zu maximieren, ohne Risse zu erzeugen.

[0014] Die einzige Möglichkeit, eine fehlerfreie Darstellung zu gewährleisten, besteht in der präzisen Darstellung der Beziehungen; Eckpunkte, die logisch gleich sind, müssen exakt gleich sein. Siehe Moreton et al. (2001). Darüber hinaus müssen Echtzeit-Grafikanwendungen häufig neu generierte Daten (z. B. die Ausgabe einer Physiksimulation) pro Frame komprimieren, bevor sie gerendert werden können. Um den Anforderungen aktueller Grafiksysteme gerecht zu werden, muss man also sehr vorsichtig und gleichzeitig schnell sein, wenn es um die Verarbeitung von Daten geht, die einer Flut von Informationen gleichkommen.

[0015] Die Leistung des Raytracing skaliert gut mit zunehmender geometrischer Komplexität, was es zu einem guten Kandidaten für die Visualisierung solcher komplexeren und realistischeren Umgebungen macht. Beispielsweise ist es möglich, mit Raytracing die Menge der Geometrie, die eine Szene modelliert, um den Faktor 100 zu erhöhen, ohne dass sich die Leistung wesentlich verschlechtert (beispielsweise könnte sich die Tracing-Zeit verdoppeln - aber im Allgemeinen nicht annähernd ver Hundertfachen).

[0016] Das Problem: Obwohl die Echtzeit- oder echtzeitnahe Verarbeitung einer großen Anzahl von Dreiecken inzwischen praktikabel ist, können die Beschleunigungsdatenstrukturen, die zur Unterstützung der Verfolgung einer derartig komplexen Geometrie erforderlich sind, linear mit der Menge der Geometrie anwachsen, und die Erstellung könnte eine Zeitspanne in Anspruch nehmen, die ebenfalls linear mit der Menge der Geometrie steigt. Komplexe 3D-Szenen, die aus Milliarden von Dreiecken bestehen, lassen sich nur schwer im Speicher speichern und in die Rendering-Hardware übertragen. Ziel ist es, eine drastische Erhöhung der Geometriemenge zu ermöglichen und gleichzeitig einen proportionalen Anstieg der Zeit, die für den Aufbau einer Beschleunigungsdatenstruktur benötigt wird, oder des Platzes, der für die Speicherung der Beschleunigungsdatenstruktur im Speicher benötigt wird, zu vermeiden.

[0017] In der Vergangenheit wurden bereits Arbeiten zur Kompression von Polygonnetzen für die Strahlen- und Pfadverfolgung durchgeführt. Siehe beispielsweise Thonat et al, Tessellation-free displacement mapping for ray tracing, pp 1-16 ACM Transactions on Graphics Volume 40 Issue 6 No.: 282 (Dezember 2021) doi.org/10.1145/3478513.3480535, //dl.acm.org/doi/abs/10.1145/3478513.3480535; Wang et al, View-dependent displacement mapping, ACM Transactions on Graphics Volume 22 Issue 3, Juli 2003 pp 334-339, doi.org/10.1145/882262.882272; Lier et al, „A high-resolution compression scheme for ray tracing subdivision surfaces with displacement“, Proceedings of the ACM on Computer Graphics and Interactive Techniques Volume 1 Issue 2 August 2018 Article No.: 33 pp 1-17, doi.org/10.1145/3233308; Chun et al, „Multiple layer displacement mapping with lossless image compression“, International Conference on Technologies for E-Learning and Digital Entertainment Edutainment 2010: Entertainment for Education. Digital Techniques and Systems pp 518-528; Szirmay-Kalos et al, Displacement Mapping on the GPU-State of the Art, Computer Graphics Forum Volume 27, Issue 6 September 2008 Pages 1567-1592.

[0018] Es gibt jedoch noch viel Raum für Verbesserungen bei der Darstellung von Polygonnetzen für Anwendungen, einschließlich, aber nicht beschränkt auf Strahlen- und Pfadverfolgung, in kompakteren, komprimierten Formen, die „Wasserdichtigkeit“ erreichen. Insbesondere gibt es mehrere Gründe, warum eine konsistente Netzgenerierung und -darstellung nicht einfach ist. Beispielsweise kann es bei der Vorwärtsdifferenzierung zu Rundungsfehlern kommen, wenn eine lange Folge von Vertices eines mosaikartigen Netzes ausgewertet wird. Dieses Problem wird manchmal noch verschärft, wenn der Kompressor und der Dekompressor unterschiedliche Berechnungshardware verwenden. Selbst wenn die Implementierungen identisch wären, könnten die gleichen Eingaben mit unterschiedlichen Rundungsmodi zu ungleichen Ergebnissen führen. Wenn verschiedene Patches unabhängig voneinander verarbeitet werden, ist es einfach nicht möglich, die Dinge während der Arbeit abzugleichen oder kleine Unstimmigkeiten im Nachhinein zu bereinigen - vielmehr sollte eine konsistente Dreiecksnetzdarstellung, Kompression, Dekompression und Verarbeitung von Anfang an als Teil des Entwurfs erfolgen. Es ist wichtig, sich darüber im Klaren zu sein, dass es keine Fehler oder Inkonsistenzen geben darf - nicht einmal ein einziges Bit -, wenn man eine Garantie für ein perfektes, wasserdichtes Rendering haben will. Siehe Moreton et al, Watertight Tessellation using Forward Differencing, EGGH01: SIGGRAPH/Eurographics Workshop on Graphics Hardware (2001).

Kurzbeschreibung der Zeichnungen

Fig. 1A und 1B zeigen beispielhafte Mikronetze.

Fig. 2A und 2B zeigen beispielhafte Risse in Mikronetzen.

Fig. 3 zeigt eine beispielhafte Tabelle, die Bits pro Dreieck zeigt.

Fig. 4 zeigt eine beispielhafte Tabelle mit beispielhaften Tesselierungsstufen.

Fig. 5 zeigt ein Beispiel eines Spektrums von Tesselierungsstufen von mehr bis weniger komprimiert von links nach rechts.

Fig. 6 zeigt einen beispielhaften unkomprimierten Verschiebungsblock.

Fig. 7 zeigt ein Prisma-toid-konvexes Hüllenmodell für ein verschobenes Mikronetz.

Fig. 8 zeigt eine beispielhafte Zusammenfassung für Vorhersage und Korrektur.

Fig. 9 zeigt einen beispielhaften Codierer-Vorhersage- und Korrekturprozess.

Fig. 10 zeigt einen beispielhaften Decodierungsprozess.

Fig. 10A zeigt ein beispielhaftes unterteiltes Teildreieck.

Fig. 11 zeigt eine beispielhafte Anwendung von vorzeichenbehafteten Korrekturen mit Tesselierungsstufe.

Fig. 12 zeigt ein beispielhaftes generisches Verschiebungsblockformat.

Fig. 13 zeigt ein beispielhaftes detailliertes Verschiebungsblockformat.

Fig. 14 zeigt einen beispielhaften Kompressionsprozess.

Fig. 15 zeigt einen beispielhaften Zahlenbereich, der ein Korrekturproblem für den Fall $p=100$, $r=1900$, $s=0$, $b=7$ darstellt.

Fig. 16 zeigt einen Beispiel-Zahlenbereich, der den Korrekturproblemfall $p=100$, $r=1900$, $s=6$, $b=3$ darstellt.

Fig. 17 zeigt eine Beispielverteilung der Differenzen zwischen Referenz- und Vorhersagewerten mod 2048 für ein bestimmtes Teildreieck, eine Korrekturstufe und einen Vertextyp. Da sich die Differenzen um 0 und 2047 ansammeln, kann die Verwendung des Wraparound-Verhaltens hier eine höhere Präzision ermöglichen.

Fig. 18 zeigt, wie die Auswahl von Verschiebungen, die den gesamten Bereich möglicher Differenzen abdecken, ohne Berücksichtigung des Wrapping zu großen Fehlern führen kann (Abstand zwischen jeder gestrichelt-gepunkteten Linie und der ihr am nächsten liegenden vertikalen durchgezogenen Linie).

Fig. 19 - Auf den Bereich $-1024 \dots 1023$ umgebrochene Verschiebungsdifferenzen. Die umgebrochenen Werte sammeln sich um 0 an, und ihr Minimum und Maximum ergeben Verschiebungen, so dass die darstellbaren Differenzen aus verschobenen Korrekturen (durchgezogene vertikale Linien) näher an den Differenzen liegen, wenn man die Abstandsmetrik von $\mathbb{Z}/2^{11} \mathbb{Z}$ verwendet (wobei zu beachten ist,

dass dies nicht die Metrik ist, die letztendlich für die Korrektur verwendet wird), was die Qualität verbessert.

Fig. 20A, 20B und 20C zeigen verschiedene Situationen, in denen Dreiecke aneinandergrenzen.

Fig. 21, 21A, 21B zeigen einen beispielhaften Pseudocode.

Fig. 22 zeigt einen beispielhaften Kompressionsalgorithmus.

Fig. 23A-23F zeigen eine Flipchart-Animation, die die Unterteilung eines Basisdreiecks zeigt.

Fig. 24 zeigt ein Beispielsystem.

Fig. 25A, 25B und 25C zeigen verschiedene Systemkonfigurationen.

Fig. 26 zeigt eine beispielhafte Raytracer-Hardware-Implementierung.

Detaillierte Beschreibung der nicht-einschränkenden Ausführungsformen

[0019] Die hier vorgestellten Ausführungsformen verwenden ein schnelles Kompressionsverfahren, das die parallele Codierung von Teildreiecken eines Dreiecksnetzes mit minimaler Synchronisierung ermöglicht und gleichzeitig qualitativ hochwertige Ergebnisse liefert, die frei von Rissen sind.

[0020] Die Einführung von verschobenen Mikronetzen (DMMs) füllt die oben genannte Lücke, indem sie zur Lösung des Problems der Speicherbandbreite beiträgt. Siehe die Mikronetz-Patentanmeldungen. Qualitativ sehr hochwertige, hochauflösende Inhalte sind oft sehr kohärent oder lokal ähnlich. Zur Erzielung drastisch erhöhter geometrischer Mengen können wir ein μ -Netz (auch „Mikronetz“) verwenden - eine strukturierte Darstellung der Geometrie, die die Kohärenz für Kompaktheit (Kompression) und ihre Struktur für effizientes Rendering mit intrinsischem Detailgrad (LOD) und Animation nutzt. Mikronetze sind ein leistungsfähiges Konzept, das zu erheblichen Geschwindigkeits- und Effizienzsteigerungen führen kann. Ein großer Vorteil der Mikronetz-Verfolgung ist beispielsweise die Möglichkeit, große Teile des Netzes schnell und effizient zu entfernen. Die μ -Netzstruktur kann beispielsweise verwendet werden, um einen starken Anstieg der Kosten für die Konstruktion der BegrenzungsvolumenHierarchie (BVH) (Zeit und Raum) zu vermeiden und gleichzeitig eine hohe Effizienz zu gewährleisten. Bei der Rasterisierung kann die dem μ -Netz innewohnende LOD verwendet werden, um Primitive in der richtigen Größe zu rastern.

[0021] Während die Anwendung einer Verschiebungsabbildung auf Mikronetze eine effiziente Darstellung von hochkomplexen 3D-Objekten ermöglicht, können komprimierte numerische Darstellungen für die Verschiebungskarte bei unvorsichtiger Implementierung Probleme mit der „Wasserdichtigkeit“ verursachen. Insbesondere kann jede verlustbehaftete Kompression, die zur Darstellung von lokalisierten numerischen Darstellungen der Verschiebungskarte verwendet wird, bei unsachgemäßer Handhabung zu Rissen im visualisierten/gerenderten Mikronetz führen.

[0022] Beispielhafte Ausführungsformen bieten einen benutzerdefinierten Kompressionsalgorithmus zur Erzeugung qualitativ hochwertiger, rissfreier verschobener Mikronetze („DMMs“), der gleichzeitig schnell genug ist, um dynamische Inhalte in modernen Echtzeitanwendungen zu verarbeiten. Mit der hier vorgestellten Technologie gelingt es, ein rissfreies Mikronetz in Form einer strukturierten Darstellung bereitzustellen, die es ermöglicht, es in sehr kompakten, komprimierten Formaten zu speichern. In einigen Beispielimplementierungen wird der durchschnittliche Speicherplatz pro Dreieck von typischerweise ~100 Bits pro Dreieck auf eine Größenordnung von nur 1 oder 2 Bits pro Dreieck reduziert.

[0023] In einer Ausführungsform wird eine solche Kompression durch ein neuartiges hierarchisches Codierungsschema erreicht, das linear interpolierte Vertex-Verschiebungsbeträge zwischen minimalen und maximalen Dreiecken verwendet, die ein Prisma bilden. Um die oben genannten Anforderungen zu erfüllen, haben wir ein schnelles Kompressionsverfahren entwickelt, das die parallele Codierung von Teildreiecken mit minimaler Synchronisierung ermöglicht und gleichzeitig qualitativ hochwertige, rissfreie Ergebnisse liefert.

[0024] In einer Ausführungsform können Verschiebungsbeträge in einem flachen, unkomprimierten Format gespeichert werden, so dass beispielsweise auf einen vorzeichenlosen normalisierten Wert (wie UNORM11) für jeden Mikrovertex direkt zugegriffen werden kann. Die Verschiebungsbeträge können auch in einem neuen Kompressionsformat gespeichert werden, das einen Vorhersage- und Korrekturmechanismus verwendet.

[0025] Eine Ausführungsform unseres Kompressionsalgorithmus schränkt die Korrekturbreiten so ein, dass die Menge der mit einem bestimmten μ -Netz-Typ darstellbaren Verschiebungswerte eine strenge Obergrenze aller Werte ist, die mit einem stärker komprimierten μ -Netz-Typ darstellbar sind. Indem der Codierer die μ -Netz-Typen von der höchsten zur niedrigsten Kompression ordnet, können wir die Teildreiecke direkt in der Reihenfolge des Kompressionsverhältnisses codieren, indem wir ein Vorhersage- und Korrekturschema (P&C) verwenden, beginnend mit dem am stärksten komprimierten μ -Netz-Typ, bis ein gewünschtes Qualitätsniveau erreicht ist. Dieses Schema ermöglicht eine parallele Codierung bei gleichzeitiger Maximierung des Kompressionsverhältnisses und ohne die Einführung nicht übereinstimmender Verschiebungswerte entlang von Kanten, die von Teildreiecken gemeinsam genutzt werden.

[0026] Zu den weiteren Aspekten gehören die Bestimmung der Bedingungen, die erfüllt sein müssen, um eine rissfreie Kompression zu gewährleisten, ein schneller Codierungsalgorithmus für ein einzelnes Teildreieck unter Verwendung des Vorhersage- und Korrekturschemas, ein Kompressionsschema für Netze, die eine einheitliche Tessellierungsrate aufweisen (d. h. alle Basisdreiecke enthalten die gleiche Anzahl von μ -Dreiecken), Erweiterungen des Kompressors für adaptiv tessellierte Dreiecksnetze und Techniken, die Wrap-around-Berechnungsmethoden zur Steigerung der Kompressionsleistung nutzen.

[0027] Eine Ausführungsform sieht eine Reihe von Regeln für die DMM-Korrektur und die Bitbreiten für die Verschiebung vor, die es ermöglichen, dass ein bestimmter Mikronetztyp immer in der Lage ist, einen stärker komprimierten Mikronetztyp darzustellen. Diese Regeln ermöglichen in Verbindung mit zusätzlichen Einschränkungen hinsichtlich der Reihenfolge, in der die DMMs codiert werden, ein Kompressionsschema als parallelen Algorithmus, bei dem nur wenig Kommunikation zwischen den unabhängig voneinander komprimierten DMMs erforderlich ist und dennoch qualitativ hochwertige, rissfreie Ergebnisse garantiert werden können. In einer Ausführungsform wandelt die hier beschriebene Technologie eine zuvor globale Optimierung in eine lokale Optimierung um, die eine parallele, rissfreie Kompression von DMMs ermöglicht, wobei zur Kompressionszeit nur sehr wenig Kommunikation zwischen den Dreiecken erforderlich ist.

[0028] Beim Rendering mit Daten aus einer komprimierten Darstellung müssen wir in der Lage sein, effizient auf die benötigten Daten zuzugreifen. Beim Rendern eines Pixels können wir die zugehörigen Texel direkt adressieren, indem wir die Speicheradresse des komprimierten Blocks berechnen, der die benötigten Texel-daten enthält. Texel-Kompressionsverfahren verwenden eine Kompression mit fester Blockgröße, die eine direkte Adressierung von Texelblöcken ermöglicht. Bei der Kompression von Verschiebungskarten (siehe unten) wird in einer Ausführungsform eine Hierarchie von Blöcken fester Größe mit komprimierten Codierungen darin verwendet.

[0029] Weitere neue Funktionen sind:

- Ein robuster Algorithmus mit konstanter Zeit für die Suche nach der nächstmöglichen Korrektur. Es wird ein Algorithmus verwendet, um die Korrektur zu finden, die den decodierten Wert so nah wie möglich an einen Referenzwert bringt. Dies erweist sich als knifflig, da die vorzeichenerweiterte und verschobene Korrektur zur Vorhersage in der Gruppe der ganzen Zahlen modulo 2048 addiert wird, die Fehler jedoch ohne Umbruch berechnet werden (und visuell erscheinen).
- Verbesserung der Berechnung von Verschiebungswerten durch Verwendung von Wrapping. In manchen Fällen können wir die erforderlichen Verschiebungen reduzieren und in Blöcken, in denen sich die Korrekturen um 0 herum gruppieren, effektiv einige zusätzliche Bits an Präzision erhalten.
- Verwendung von Verschiebungsbereichen in der Codierungserfolgsmetrik. Wir können jedem Vertex eine Wichtigkeit zuweisen; Regionen mit höherer Wichtigkeit neigen dazu, qualitativ hochwertigere Formate zu verwenden, während Regionen mit geringerer Wichtigkeit dazu neigen, qualitativ schlechtere Formate zu verwenden.

Rissfrei-Garantie

[0030] Fig. 2A und 2B zeigen beispielhafte Netze, die Risse aufweisen, d. h. sichtbare Nähte zwischen den Innenkanten des Netzes. Die Vertices benachbarter Dreiecke sollten sich entlang einer gemeinsamen Kante an der gleichen Position befinden, aber manchmal sind sie versetzt, so dass sie sich nicht an der gleichen Position befinden. Zwei benachbarte Dreiecke sollten eine gemeinsame Kante haben, aber die Vertices dieser Kante haben abweichende Daten. Bei der Verwendung herkömmlicher Netze ist in der Regel gewährleistet, dass die inneren Kanten nicht brechen, aber das ist nicht unbedingt der Fall, wenn Mikronetze mit Verschiebungsabbildung verwendet werden. In einem solchen Kontext könnte jede Kante zwischen benachbarten Mikronetzen potenziell brechen.

[0031] Eine häufig verwendete allgemeine Lösung für Rissbildung besteht darin, sicherzustellen, dass die Hardware oder der Shader dieselben Eingabedaten für gemeinsame Eckpunkte und gemeinsame Kanten verwendet. Es ist jedoch bekannt, dass Verschiebungs-Offsets oder Unterschiede entlang gemeinsamer Kanten leicht unterschiedliche oder variierende Werte erfassen, was zu Rissbildungs-Artefakten führen kann. Dies kann vor allem dann der Fall sein, wenn auf die numerischen Werte der gemeinsam genutzten Vertices und Kanten zugegriffen wird und diese lokal/unabhängig bestimmt werden, z. B. auf einer zufällig angeordneten Basis und nicht gemeinsam und/oder in einer bestimmten Reihenfolge.

[0032] Die beispielhaften, nicht einschränkenden Ausführungsformen hierin bieten trotz solcher Herausforderungen Garantien für Rissfreiheit und Wasserdichtigkeit.

DMM-Kompression

[0033] Bei der Beschreibung sehr detaillierter Geometrien ist es wichtig, dass die Beschreibung so kompakt wie möglich ist. Die Realisierbarkeit detaillierter Geometrien für Echtzeit-Computergrafiken hängt davon ab, dass sie direkt von einer kompakten Darstellung aus gerendert werden können. Die oben erwähnten mitanhängigen gemeinsamen „Mikronetz“-Patentanmeldungen beschreiben die Einbindung von Verschiebungskarten (DMs) in eine μ -Netzdarstellung. Da es sich bei den DMs um hochwertige μ -Netzkomponenten handelt, können sie unter Ausnutzung der ihnen innewohnenden Kohärenz komprimiert werden. DMs können als Repräsentanten von Daten betrachtet werden, die mit Vertices verbunden sind. Diese Datenklasse kann so verstanden werden, dass sie sowohl verlustfreie als auch verlustbehaftete Kompressionsverfahren erfordert. Während ein verlustfreies Schema eine Eingabe exakt darstellen kann, darf ein verlustbehaftetes Schema eine Eingabe bis auf eine bestimmte Toleranz annähern. Die Tatsache, dass ein System verlustbehaftet ist, bedeutet, dass Daten verloren gehen - was höhere Kompressionsraten und kompaktere Datendarstellungen ermöglichen sollte. Wie bereits erwähnt, besteht das Problem jedoch nicht (nur) darin, sicherzustellen, dass der Dekompressor die komprimierten Daten auf deterministische Weise wiederherstellt - es wird zusätzlich durch die Notwendigkeit kompliziert, dieselben (Bit-für-Bit-)Verschiebungswerte wiederherzustellen, wenn die Eckpunkte auf einer gemeinsamen Mosaikkante zwischen zwei verschiedenen Polygonen liegen.

[0034] Verlustbehaftete Verfahren können anzeigen, wo eine ungenaue Codierung stattgefunden hat oder welche Proben nicht verlustfrei codiert werden konnten.

Verschiebungsblockspeicher

[0035] In einer Ausführungsform werden die Netzverschiebungsinformationen in einer Reihe verschiedener komprimierter Formate gespeichert, die es ermöglichen, die Mikrodreiecke mit so wenigen Bits wie möglich zu beschreiben.

[0036] In einer beispielhaften Ausführungsform weist das Mikronetz ein Netz aus Basisdreiecken auf, die an ihren jeweiligen Vertices zusammengenäht oder verbunden sind. Diese Basisdreiecke können als „API“-Basisdreiecke bezeichnet werden, da sie jeweils drei Vertices des Typs definieren, der von einem herkömmlichen Vertex-Shader oder Raytracer verarbeitet werden kann. In einer Ausführungsform wird das Basisdreieck selbst jedoch nicht abgebildet, gerastert oder anderweitig visualisiert und dient stattdessen als Plattform für ein rekursiv unterteilbares verschiebungsabgebildetes Mikronetz. Dieses Mikronetz wird als regelmäßiges Netz von $2^n \times 2^n$ gebildet (wobei n eine beliebige ganze Zahl ungleich Null ist), wobei jede weitere Mosaikstufe jedes Teildreieck der vorherigen Stufe in vier (4) kleinere Teildreiecke gemäß einem baryzentrischen Gitter und einer Raumfüllkurve unterteilt. Siehe Tabelle der **Fig. 4**. In diesem Beispiel sind auf höheren Tessellierungsstufen mehr Teildreiecke innerhalb des Basisdreiecks definiert und bieten somit einen höheren Detaillierungsgrad. Siehe **Fig. 5**.

[0037] In Ausführungsbeispielen wird für jeden Mikrovertex des Mikronetzes ein Verschiebungswert gespeichert. Diese Verschiebungswerte werden in Verschiebungsblöcken gespeichert, wie in **Fig. 6** dargestellt. Die Verschiebungsblöcke, in denen die Verschiebungswerte gespeichert werden, haben eine feste Größe, die vom Speichersubsystem des Grafiksystems und der Größe des Speicherblockverbrauchs der Grafikhardware abhängt. Beispielsweise sind in einer Ausführungsform alle Verschiebungswerte für alle Vertices eines Teildreiecks so konfiguriert, dass sie in eine einzige Cacheline passen (z. B. ist in einem Beispiel eine volle Cacheline 128 Byte und eine halbe Cacheline 64 Byte groß).

[0038] In einer Ausführungsform ist aufgrund der Art und Weise, wie die Verschiebungswerte konfiguriert sind, keine Kompression erforderlich, um Verschiebungswerte für niedrigere Tessellierungsstufen in eine ein-

zige Cacheline einzupassen. Wie **Fig. 6** zeigt, sind die unteren Tesselierungsstufen weniger komprimiert - und ihre Verschiebungsblöcke können den vollen Präzisionswert für jeden Vertex enthalten. Siehe zusammenfassende Tabelle unten:

Basisdreieck U Vertex-Verschiebung	11-Bit UNORM
Basisdreieck V Vertex-Verschiebung	11-Bit UNORM
Basisdreieck W Vertex-Verschiebung	11-Bit UNORM
Stufe 1 Vertex-Verschiebungen	3 zusätzliche 11-Bit-UNORMs
Stufe 2 Vertex-Verschiebungen	9 zusätzliche 11-Bit-UNORMs
Stufe 3 Vertex-Verschiebungen	30 zusätzliche 11-Bit-UNORMs

[0039] Siehe beispielhaften unkomprimierten Verschiebungsblock der **Fig. 6**. Es ist zu beachten, dass dieser Beispiel-Verschiebungsblock alle Verschiebungswerte in der obigen Tabelle für die rekursiven Tesselierungsstufen 0, 1, 2 und 3 im Bereich von $\frac{1}{2}$ Cacheline enthält. Beispielhafte Ausführungsformen speichern Verschiebungswerte für mehrere Tesselierungsstufen 0-3 gleichzeitig, um der Echtzeit-Hardware die Möglichkeit zu geben, Teildreiecke auszuwählen und zwischen verschiedenen Detaillierungsgraden „im Vorübergehen“ zu wählen, ohne dass zusätzlicher Speicherplatz oder Zugriffe erforderlich sind.

Verschiebungswerte mit „voller“ Präzision werden als UNORM11 dargestellt

[0040] **Fig. 7** ist ein beispielhaftes konvexes Prismatoid-Hüllenmodell, das den Eckpunkten des Polygonnetzes Verschiebungswerte zuweist, die zwischen dem maximalen und dem minimalen Dreieck interpoliert werden, wobei ein Bereich von 0 bis 1 verwendet wird (mit Vorspannung und Skalierung, die auf das gesamte Netz des Basisdreiecks angewendet werden, z. B. um das maximale Dreieck zu definieren, wobei das minimale Dreieck in einer Ausführungsform als die ebene Fläche des Basisdreiecks definiert ist, obwohl das Basisdreieck zwischen dem minimalen und dem maximalen Dreieck oder außerhalb davon liegen oder sogar das minimale und/oder maximale Dreieck schneiden könnte, z. B. wenn die Vorspannungen an verschiedenen Vertices unterschiedliche Vorzeichen haben). Beispielsweise kann der Bereich zwischen dem minimalen und dem maximalen Dreieck mit einer angemessenen Auflösung unter Verwendung von 11 Bits definiert werden, wobei 2^{11} oder 2048 inkrementelle Positionen für die lineare Interpolation zur Verfügung stehen und eine sehr kompakte vorzeichenlose normalisierte UNORM11-Zahlendarstellung möglich ist. „UNORM“ bedeutet, dass die Werte vorzeichenlose Ganzzahlen sind, die in Fließkommazahlen umgewandelt werden. Der maximal mögliche darstellbare Wert wird zu 1,0 und der minimal darstellbare Wert wird zu 0,0. Beispielsweise wird der Binärwert 2047 in einer UNORM11 als 1,0 interpretiert. Andere UNORMs oder andere numerische Darstellungen sind ebenfalls möglich.

[0041] So können beispielsweise die Verschiebungsbeträge in einem flachen, unkomprimierten Format gespeichert werden, in dem die UNORM11-Verschiebung für jeden μ -Vertex direkt abgerufen werden kann.

[0042] Mit zunehmender Tesselierungsstufe steigt jedoch auch die Anzahl der Mikrovertices, so dass in einer einzigen Cacheline bald kein Platz mehr ist, um die entsprechenden Verschiebungswerte in UNORM11 zu speichern. Siehe Tabelle in **Fig. 4**. Für höhere Tesselierungsstufen verwenden wir ein komprimiertes Format, das eine Korrektur eines vorhergesagten Wertes codiert und übermittelt, die eine Prädiktorschaltung innerhalb des Decoders basierend auf den ihm bereits vorliegenden Informationen bestimmen kann. Solche Verschiebungsbeträge können also auch in einem Kompressionsformat gespeichert werden, das einen Vorhersage- und Korrekturmechanismus (P&C) verwendet.

Verschiebungskompression mit Vorwärtsdifferenzierung („Vorhersage- und Korrektur“)

[0043] Der P&C-Mechanismus stützt sich in einem Ausführungsbeispiel auf den rekursiven Unterteilungsprozess, der zur Bildung eines μ -Netzes verwendet wird. Für das Basisdreieck wird ein Satz von Basisankerpunkten festgelegt. Auf jeder Unterteilungsstufe werden neue Vertex-Verschiebungswerte gebildet, indem die Verschiebungswerte von zwei benachbarten Vertices in einer höheren Unterteilungsstufe gemittelt werden. Dies ist der Schritt der Vorhersage: Es wird vorhergesagt, dass der Wert der Durchschnitt der beiden benachbarten Vertices ist.

[0044] Im nächsten Schritt wird diese Vorhersage korrigiert, indem sie nach oben oder unten verschoben wird, um die richtige Position zu erreichen. Wenn diese Bewegungen klein sind oder verlustbehaftet gespeichert werden können, kann die Anzahl der Bits, die zur Korrektur der Vorhersage verwendet werden, kleiner sein als die Anzahl der Bits, die zur direkten Codierung benötigt werden. Die Bitbreite der Korrekturfaktoren ist für jede Stufe variabel.

[0045] Genauer gesagt wird für die Vorhersage und Korrektur ein Satz von Basisanker-Verschiebungen für das Basisdreieck festgelegt, wie in **Fig. 6** gezeigt. Bei jedem Unterteilungsschritt zur nächsthöheren Tessellierungsstufe werden die Verschiebungsbeträge für jeden neuen Mikrovertex vorhergesagt, indem die Verschiebungsbeträge der beiden benachbarten (Mikro-)Vertices in der niedrigeren Stufe gemittelt werden. Bei diesem Vorhersageschritt wird der Verschiebungsbetrag als Durchschnitt der beiden (zuvor erhaltenen oder zuvor berechneten) benachbarten Verschiebungsbeträge vorhergesagt:

$$\text{disp_amount_prediction} = (\text{disp_amount_v0} + \text{disp_amount_v1} + 1) / 2$$

[0046] Es sei darauf hingewiesen, dass der Codierer dem Decoder die Verschiebungen des Basisankers mitteilt und dass der Decoder bei der rekursiven Unterteilung des Basisdreiecks in immer tiefere Unterteilungsstufen (die zu immer höheren Tessellierungsstufen führen) bereits die Verschiebungswerte der benachbarten Mikrovertices berechnet hat, die somit für die Berechnung (durch lineare Interpolation) der Verschiebungswerte für neue Zwischen-Mikrovertices zur Verfügung stehen.

[0047] Natürlich ist der tatsächliche Verschiebungswert einer Mikrospitze nicht notwendigerweise derselbe wie der ihrer unmittelbaren Nachbarn - das Mikronetz ist in einer Ausführungsform so konfiguriert, dass jedes Mikrodreieck eine unabhängige Ausrichtung haben kann, was bedeutet, dass seine drei Mikrospitzen unabhängig voneinander definierte Verschiebungswerte haben können. Wie in einem typischen Vorwärtsdifferenzierungssystem berechnet der Codierer auch eine skalare Korrektur der Vorhersage und übermittelt sie an den Decoder. Mit anderen Worten: Der Codierer berechnet die Vorhersage und vergleicht sie dann mit dem tatsächlichen Verschiebungswert des Mikrovertex. Siehe **Fig. 8** und **9**. Aus diesem Vergleich ermittelt der Codierer ein Delta (Differenz) oder eine „Korrektur“, die er an den Decoder weitergibt. Der Decoder (siehe **Fig. 10**) berechnet die Vorhersage selbständig aus den ihm bereits vorliegenden Informationen und wendet dann die vom Codierer erhaltene Korrektur an, um den vorhergesagten Wert anzupassen. In diesem Fall werden die Verschiebungswerte für die Vertices $d(4)$ und $d(7)$ des Mikrodreiecks (siehe **Fig. 10A**) wie folgt berechnet:

$$d(4) = (d(2) + d(1) + 1) / 2 + \text{Korrektur}(4)$$

$$d(7) = (d(5) + d(3) + 1) / 2 + \text{Korrektur}(7).$$

[0048] Der nächste Schritt, der sowohl vom Codierer als auch vom Decoder ausgeführt wird, besteht also darin, den vorhergesagten Verschiebungsbetrag mit einer skalaren Korrektur pro Vertex zu korrigieren, wobei der Verschiebungsbetrag nach oben oder unten verschoben wird, um den endgültigen Verschiebungsbetrag zu erreichen. Wenn diese Bewegungen klein sind oder verlustbehaftet gespeichert werden können, kann die Anzahl der Bits, die zur Korrektur der Vorhersage verwendet werden, kleiner sein als die Anzahl der Bits, die zur direkten Codierung benötigt werden. In der Praxis ist es wahrscheinlich, dass für höhere Unterteilungsstufen aufgrund der Selbstähnlichkeit der Oberfläche kleinere Korrekturen erforderlich sind, so dass die Bitbreiten der Korrekturfaktoren für höhere Ebenen reduziert werden. Siehe **Fig. 11**.

[0049] Die Basisankerverschiebungen sind vorzeichenlos (UNORM11), während die Korrekturen vorzeichenbehaftet sind (Zweierkomplement). In einer Ausführungsform wird auch ein Verschiebungswert eingeführt, damit die Korrekturen mit einer geringeren als der vollen Breite gespeichert werden können. Die Verschiebungswerte werden pro Unterteilungsstufe in vier Varianten gespeichert (ein unterschiedlicher Verschiebungswert für die Mikrovertices jeder der drei Kanten des Teildreiecks und ein vierter Verschiebungswert für die Mikrovertices im Inneren des Dreiecks), damit die Vertices an jeder der Kanten des Teildreiecks unabhängig voneinander (z. B. unter Verwendung einfacher Schieberegister) und von den Vertices innerhalb des Teildreiecks verschoben werden können.

[0050] Genauer gesagt wird die Netzoberfläche bei immer tieferen Tessellierungsstufen immer selbstähnlicher, so dass der Codierer immer weniger Bits zur Codierung der vorzeichenbehafteten Korrektur zwischen der tatsächlichen Oberfläche und der vorhergesagten Oberfläche verwenden kann. Das Codierungsschema in einer Ausführungsform bietet eine Codierung mit variabler Länge für die vorzeichenbehaftete Korrektur.

Für grobe Korrekturen können mehr Codierbits verwendet werden, für feinere Korrekturen werden weniger Codierbits benötigt. In Ausführungsbeispielen ist diese Codierung der Korrekturwerte mit variabler Länge wie folgt an die Tesselierungsstufe gebunden:

Tesselierungsstufe	Anzahl der Korrekturen	Breite der Korrekturen (Bits)
1	3	11
2	9	8
3	30	4
4	108	2
5	408	1

[0051] So kann in einer Ausführungsform, wenn Korrekturen für sehr viele Mikrodreiecke codiert werden, die Anzahl der korrekten Bits pro Mikrodreieck gering sein (z. B. so gering wie ein einziges Bit in einer Ausführungsform).

[0052] In einer Ausführungsform verwendet das Codierungsschema Block-Gleitkomma, was es ermöglicht, sogar ein Bit Präzision an einer beliebigen Stelle im Bereich zu platzieren, wo sie benötigt oder gewünscht wird. Auf diese Weise ermöglichen „Verschiebungsbits“ die Anpassung der Amplitude von Korrekturen, ähnlich wie bei einem gemeinsamen Exponenten. Die Verschiebungen für die oben genannten Tesselierungsstufen können in einer Ausführungsform wie folgt aussehen:

Tesselierungsstufe	Anzahl der Verschiebungswerte	Breite der Verschiebungswerte (Bits)
1	0	
2	4	2
3	4	3
4	4	4
5	4	4

[0053] Der Decoder (und der Codierer bei der Wiederherstellung von zuvor komprimierten Verschiebungswerten) kann einen Hardware-Schiebeschaltkreis wie z. B. ein Schieberegister verwenden, um Korrekturwerte um Beträge und in Richtungen zu verschieben, die durch die Verschiebewerte festgelegt sind. Beispielsweise können die 4-Bit-Verschiebungswerte der Stufe 5 den 1-Bit-Korrekturwert an eine beliebige von 16 verschiedenen Verschiebungspositionen verschieben, um einen relativ großen dynamischen Bereich für den 1-Bit-Korrekturwert bereitzustellen.

[0054] Das Vorsehen unterschiedlicher Verschiebungen für verschiedene Stufen und unterschiedlicher Verschiebungen für jede Kante und die inneren Vertices verhindert „Kettenreaktionen“ oder dominoähnliche Effekte (d. h., wenn das Umstoßen eines Dominosteins dazu führt, dass sich der Impuls auf einen nächsten Dominostein ausbreitet, der ihn wiederum auf einen weiteren Dominostein ausbreitet usw.) und vermeidet die Notwendigkeit einer globalen Optimierung des Netzes. Durch die Entkopplung der Verschiebungswerte, die zur Codierung/Decodierung der inneren Vertices verwendet werden, von den Kanten-Vertices können die Kanten-Vertices mit ihren Gegenstücken auf benachbarten Mikronetzen, die dieselben Kanten teilen, übereinstimmen, ohne dass die Beschränkungen für ihre Werte auf die inneren Vertices übertragen werden. Wenn dies nicht möglich ist, können solche Beschränkungen lokal auftreten und sich über das gesamte Netz ausbreiten und zu globalen Beschränkungen werden. Wie weiter unten erläutert wird, kann die Breite der Verschiebungs- und Korrekturwerte nicht beliebig sein, sondern muss Beschränkungen folgen, um eine Bit-für-Bit-Übereinstimmung zwischen den Kompressionsstufen zu gewährleisten.

[0055] Die Vorhersage-und-Korrektur-Operation in dem folgenden Beispiel Formel 1 unten, in Pseudocode geschrieben:

$\text{disp_amount_prediction} = (\text{disp_amount_v0} + \text{disp_amount_v1} + 1) / 2$

$\text{disp_correction} = \text{signextend}(\text{correction}) \ll \text{shift}[\text{level}][\text{type}]$

$\text{disp_final} = \text{disp_amount_prediction} + \text{disp_correction}$

[0056] Jeder endgültige Verschiebungsbetrag dient dann als Quelle für die Vorhersage der nächsttieferen Stufe. Beachten Sie, dass jede Vorhersage einen zusätzlichen „+ 1“-Term enthält, der das Runden gegenüber dem Abschneiden ermöglicht, da die Division hier die abschneidende Division der Korrektur ist. Dies entspricht $\text{prediction} = \text{round}((v_0 + v_1)/2)$ in exakter Präzisionsarithmetik, wobei Halbzahlen auf die nächste ganze Zahl aufgerundet werden.

[0057] Wie aus der nachstehenden Erörterung hervorgeht, besteht ein primäres Entwurfsziel für diesen Kompressionsalgorithmus darin, die Korrekturbitbreiten so einzuschränken, dass die Menge der mit einem bestimmten μ -Netz-Typ darstellbaren Verschiebungswerte eine strenge Obermenge aller Werte ist, die mit einem stärker komprimierten μ -Netz-Typ darstellbar sind. Die obigen Breiten der Korrektur- und Verschiebungswerte erfüllen diese Bedingung.

[0058] In einer anderen Ausführungsform kann die Verschiebungskarte unter Verwendung der oben beschriebenen Prädiktions- und Kontrolltechnik (P&C) erzeugt und codiert werden, und es wird der Konstant-Zeit-Algorithmus zum Auffinden der nächstliegenden Korrektur verwendet. In einer Ausführungsform werden, wie oben beschrieben, die P&C-Technik und der Algorithmus zum Auffinden der nächstgelegenen Korrektur in Verbindung mit dem schnellen Kompressionsschema verwendet, das darauf abzielt, die Korrekturbitbreiten in Verschiebungscodierungen zu beschränken.

Verschiebungsspeicher

[0059] Verschiebungsbeträge werden in 64B- oder 128B-Granularblöcken gespeichert, die als Verschiebungsblöcke bezeichnet werden. Die Sammlung von Verschiebungsblöcken für ein einzelnes Basisdreieck wird als Verschiebungsblocksatz bezeichnet. Ein Verschiebungsblock codiert Verschiebungsbeträge für entweder 8×8 (64), 16×16 (256) oder 32×32 (1024) μ -Dreiecke.

[0060] In einer bestimmten, nicht einschränkenden Implementierung wird der Verschiebungssatz mit dem größte Speicherbedarf einheitliche, unkomprimierte Verschiebungsblöcke aufweisen, die 8×8 (64) μ -Dreiecke in 64 Byte abdecken. Der kleinste Speicherbedarf würde sich aus einheitlich komprimierten Verschiebungsblöcken ergeben, die 32×32 in 64 Byte abdecken, was $\sim 0,5$ Bit pro μ -Dreieck bedeutet. Der Unterschied zwischen den beiden ist ungefähr ein Faktor 16. Der tatsächlich erreichte Speicherbedarf wird irgendwo in diesem Bereich liegen. Die Größe eines Verschiebungsblocks im Speicher (64B oder 128B) gepaart mit der Anzahl der μ -Dreiecke, die er darstellen kann (64, 256 oder 1024), definiert einen μ -Netztyp. Wir können die μ -Netz-Typen von der höchsten bis zur niedrigsten Kompression ordnen, was eine „Reihenfolge der Kompressionsverhältnisse“ ergibt, die bei der wasserdichten Kompression verwendet wird - siehe **Fig. 5**.

[0061] Wie die Tabelle in **Fig. 3** zeigt, können für Tesselierungsstufen oberhalb von Stufe 3 mehrere unkomprimierte 8×8 (64B) Verschiebungsblöcke pro Basisdreieck (oder alternativ die maximal mögliche Anzahl von Verschiebungsblöcken für eine bestimmte Tesselierungsstufe) wie folgt verwendet werden:

Tesselierungsstufe	Anzahl der Verschiebungsblöcke
4	4
5	16
6	64
7	256
8	1024
9	4096
10	16384
11	65536
12	262144
13	1048576

[0062] Während die Anzahl der Verschiebungsblöcke in der obigen Tabelle mit einer größeren Anzahl von Dreiecken geometrisch ansteigt, wird die Selbstauslese auf der Seite des Decoders/der Grafikerzeugung oft oder normalerweise (z. B. beim Raytracing) sicherstellen, dass nur ein oder eine kleine Anzahl der Verschiebungsblöcke tatsächlich aus dem Speicher abgerufen wird.

[0063] Fig. 12 und 13 zeigen beispielhafte detaillierte komprimierte Verschiebungsblockformate, die der Codierer verwendet, um dem Decoder komprimierte Verschiebungswerte zu übermitteln. Wie bereits erwähnt, können komprimierte Verschiebungsblöcke in einer Ausführungsform entweder 64B oder 128B groß sein und werden für 16x16 oder 32x32 Teildreiecke verwendet. Diese Blöcke spezifizieren die Ankerverschiebungen in UNORM11, Korrekturen pro Mikrovertex für jede Unterteilungsstufe im Zweierkomplement und vier vorzeichenlose Verschiebungsvarianten pro Stufe über der Unterteilungsstufe 1. Es ist zu beachten, dass die Bitbreiten sowohl für die Korrekturen als auch für die Verschiebungen von der Auflösung des Teildreiecks sowie von der Unterteilungsstufe abhängen. Darüber hinaus können in einer Ausführungsform die Mikrovertex-Verschiebungsinformationen für dieselbe Unterteilungsstufe in mehr oder weniger komprimierten Formaten codiert werden (vergleichen Sie beispielsweise in Fig. 13 die Bitbreiten der 16x16-Korrekturen auf 256-Mikrodreieckebene für volle Cacheline-Blöcke mit 128B gegenüber 64B-Verschiebungsblöcken mit halber Cacheline).

[0064] In einigen Ausführungsformen sind die Basisankerpunkte vorzeichenlos (UNORM11), während die Korrekturen vorzeichenbehaftet sind (Zweierkomplement). Ein Verschiebungswert ermöglicht die Speicherung von Korrekturen mit weniger als der vollen Breite. Die Verschiebungswerte werden pro Stufe in vier Varianten gespeichert, damit die Vertices an jeder der Netzkanten des Teildreiecks unabhängig voneinander und von den Vertices innerhalb des Teildreiecks verschoben werden können. Jeder decodierte Wert wird zu einer Quelle der Vorhersage für die nächsttiefere Stufe.

Kompressor - Teildreieck-Codierer

[0065] Bei einigen Ausführungsformen wird ein 2-Pass-Ansatz verwendet, um ein Teildreieck mit einem bestimmten μ -Netztyp zu codieren. Siehe Fig. 14.

[0066] Im ersten Durchgang wird das oben beschriebene P&C-Schema verwendet, um verlustfreie Korrekturen für eine Unterteilungsstufe zu berechnen, wobei der gesamte Wertebereich der Korrekturen im Auge behalten wird. Anschließend wird der optimale Verschiebungswert bestimmt, der für jede Kante und für die internen Vertices verwendet werden kann (insgesamt 4 Verschiebungswerte in einer Ausführungsform), um die gesamten Bereiche mit der Anzahl der verfügbaren Korrekturbits abzudecken. Dieser Prozess wird unabhängig für die Eckpunkte auf den drei Kanten des Teildreiecks und für die inneren Eckpunkte des Teildreiecks durchgeführt, so dass insgesamt 4 Verschiebungswerte pro Unterteilungsstufe zur Verfügung stehen. Die Unabhängigkeit dieses Prozesses für jede Kante ist erforderlich, um die Bedingungen für eine rissfreie Kompression zu erfüllen.

[0067] Der zweite Durchgang codiert das Teildreieck erneut mit dem P&C-Schema, aber diesmal mit verlustbehafteten Korrekturen und Verschiebungswerten, die im ersten Durchgang berechnet wurden. Der zweite Durchlauf verwendet die Ergebnisse des ersten Durchlaufs (und insbesondere den maximalen Korrekturbereich und die Anzahl der für die Korrektur verfügbaren Bits), um die verlustbehafteten Korrektur- und Verschiebungswerte zu strukturieren - letztere ermöglichen es, größere Zahlen darzustellen, als dies ohne Verschiebung möglich wäre. Das Ergebnis dieser beiden Durchgänge kann unverändert verwendet werden oder als Ausgangspunkt für Optimierungsalgorithmen dienen, mit denen die Qualität und/oder das Kompressionsverhältnis weiter verbessert werden kann.

[0068] Eine Hardware-Implementierung des P&C-Schemas kann im Falle eines (ganzzahligen) Über- oder Unterlaufs ein „wrapping around“-Verhalten aufweisen. Diese Eigenschaft kann im 2. Durchgang ausgenutzt werden, um Korrekturwerte durch „wrapping around“ darzustellen, die angesichts der begrenzten Anzahl verfügbarer Bits sonst nicht erreichbar wären. Dies bedeutet auch, dass die Berechnung von Verschiebungswerten, die auf dem Korrekturbereich basieren, das Wrapping ausnutzen kann, um qualitativ hochwertigere Ergebnisse zu erhalten (siehe „Verbesserung der Berechnung von Verschiebungswerten durch Nutzung des Wrappings“ unten).

[0069] Es sei darauf hingewiesen, dass das Codierungsverfahren an sich nie scheitern kann und dass für einen bestimmten μ -Netztyp immer ein Teildreieck codiert werden kann. Dennoch kann der Kompressor das Ergebnis dieses Kompressionsschritts analysieren und unter Verwendung einer Reihe von Metriken und/oder

Heuristiken entscheiden, dass die resultierende Qualität nicht ausreicht. (Siehe „Verwendung der Längen der Verschiebungsrichtung in der Erfolgsmetrik der Codierung“ weiter unten).

[0070] In diesem Fall kann der Kompressor versuchen, das Teildreieck mit weniger komprimierten μ -Netztypen zu codieren, bis die erwartete Qualität erreicht ist. Dieser iterative Prozess kann dazu führen, dass versucht wird, ein Teildreieck mit einem μ -Netz-Typ zu codieren, der nicht alle seine μ -Dreiecke darstellen kann. In diesem Fall wird das Teildreieck rekursiv in vier Teildreiecke aufgeteilt, bis es codiert werden kann. In einer Ausführungsform wird der anfängliche Teilungsschritt nur dann durchgeführt, wenn das aktuelle Teildreieck mehr Dreiecke enthält, als mit dem aktuellen Mikronetztyp codiert werden können (daher die Notwendigkeit einer rekursiven Teilung, bis die Anzahl der Mikrodreiecke im Teildreieck der Anzahl der Dreiecke entspricht, die mit dem aktuellen Mikronetztyp codiert werden können).

Ausnutzung der Mod 2048 Arithmetik

[0071] In den obigen Ausdrücken zur Berechnung der Vorhersage versucht der Kompressor, die Korrektur basierend auf der Vorhersage, der Verschiebung und dem unkomprimierten Wert zu berechnen. In einer Ausführungsform kann diese Korrekturberechnung jedoch etwas knifflig sein, wenn die Berechnung unter Verwendung der Wrapping-Arithmetik (z. B. 0, 1, 2, ..., 2046, 2047, 0, 1, 2 ...) für die Mod-2048-Arithmetik durchgeführt wird - was in einer Ausführungsform von der Decoder-Hardware verwendet wird, wenn die Vorhersage zur Korrektur basierend auf vorzeichenlosen UNORM11-Werten hinzugefügt wird. Während es sich bei der Mittelwertbildung um eine typische Mittelwertbildung handelt, wird die decodierte Position nach den Regeln der vorzeichenlosen Arithmetik umgewandelt, wenn die Korrektur zur Vorhersage addiert wird. Die Fehlermetrik basiert in einer Ausführungsform nicht auf der Umschlagarithmetik. Daher ist es Sache des Software-Codierers, entweder einen auf gespeicherten Werten basierenden Umbruch zu vermeiden oder das Ergebnis des Umbruchs sinnvoll zu gestalten. Nachfolgend wird ein Algorithmus beschrieben, mit dem der Codierer dieses Wrapping nutzen und zur Qualitätsverbesserung ausschöpfen kann. Eine alternative Ausführungsform könnte die zusätzlichen Ergebnisse festhalten und den Umbruch verhindern (wodurch effektiv Informationen verworfen würden), würde dann aber die Möglichkeit verlieren, die Kompressionsergebnisse durch Ausnutzung des Umbruchverhaltens zu verbessern. In einer Ausführungsform kann die Ausnutzung des Wrap-Around-Verhaltens den Fehler um den Faktor 3 verringern.

Verschiebungskompression - Ein robuster Konstantzeit-Algorithmus zur Ermittlung der nächstliegenden Korrektur

[0072] Wie oben beschrieben, sind Korrekturen von der Unterteilungsstufe n bis zur Unterteilungsstufe $n+1$ vorzeichenbehaftete Ganzzahlen mit einer festen Anzahl von Bits b (gegeben durch das Format des Teildreiecks und die Unterteilungsstufe) und werden gemäß der obigen Formel angewendet. Obwohl ein Codierer Korrekturen auf verschiedene Arten berechnen kann, besteht ein allgemeines Problem für einen Codierer darin, den b -Bit-Wert von c (Korrektur) zu finden, der die absolute Differenz zwischen dem d (decodierten) und einem Referenzwert r (unkomprimiert) in der Formel in **Fig. 15** minimiert, wobei p (Vorhersage) und s (shift[level][type]) gegeben sind.

[0073] Dies wird dadurch erschwert, dass die ganzzahlige Arithmetik umläuft (sie entspricht der Gruppenoperation in der abelschen Gruppe $\mathbb{Z}/2^{11} \mathbb{Z}$), aber die Fehlermetrik wird ohne Umlaufen berechnet (es ist nicht die euklidische Metrik in $\mathbb{Z}/2^{11} \mathbb{Z}$). Anhand eines Beispiels soll gezeigt werden, dass es sich hierbei um ein nicht triviales Problem handelt.

[0074] Betrachten wir den Fall $p=100$, $r=1900$, $s=0$ und $b=7$, wie in **Fig. 15** dargestellt. Die hervorgehobene vertikale Linie p nahe der linken Seite des Diagramms zeigt den vorhergesagten Verschiebungswert, und die vertikale Linie r zeigt den Referenzwert der Verschiebung, dem der decodierte Wert nahe kommen sollte. Beachten Sie, dass die beiden Linien nahe an den entgegengesetzten Extremen des dargestellten 11-Bit-Bereichs liegen. Dies kann relativ häufig vorkommen, wenn zur Definition der Verschiebungswerte eine prismatische konvexe Dreieckshülle mit Maximum und Minimum verwendet wird.

[0075] Dargestellt ist die Zahlenreihe aller UNORM11-Werte von 0 bis 2047, die Lage des vorhergesagten Werts p in der dicken Linie und des Referenzwerts r in einer punktgestrichelten Linie, und in der helleren Schattierung um die dicke Linie von p alle möglichen Werte von d für alle möglichen Korrekturen (da $b=7$, sind die möglichen Korrekturen die vorzeichenbehafteten ganzen Zahlen von $-2^6 = -64$ bis einschließlich $2^6 - 1 = 63$).

[0076] In diesem Beispiel gibt es eine Verschiebung von 0 und einen möglichen Korrekturbereich von -64 bis +63, wie die vertikalen Linien links und rechts der mit p bezeichneten Vorhersagelinie zeigen. Dies scheint die äußerste rechte vertikale Linie bei +63 zu sein. Wendet man jedoch die Wrap-Around-Arithmetik an, so ist die Linie, die der Referenzlinie r am nächsten liegt, nicht die äußerste rechte Linie, sondern die äußerste linke Linie bei -64, da diese äußerste linke Linie bei Anwendung der Wrap-Around-Arithmetik den geringsten Abstand von der Referenzlinie r hat.

[0077] In diesem Fall besteht die Lösung darin, die Korrektur $c=63$ zu wählen, was einen decodierten Wert von $d=163$ und einen Fehler von $\text{abs}(r-d) = 1737$ ergibt. Wäre die Abstandsmetrik $\mathbb{Z}/2^{11}\mathbb{Z}$, wäre die Lösung stattdessen $c=-64$, was einen decodierten Wert von $d=36$ und einen Fehler von 183 ergeben würde (Wrapping Around). Obwohl die Verwendung der Fehlermetrik $\mathbb{Z}/2^{11}\mathbb{Z}$ einfacher zu berechnen ist, führt sie zu einer Korrektur mit dem entgegengesetzten Vorzeichen der korrekten Lösung, was zu unangenehmen visuellen Artefakten wie Pockennarben führt.

[0078] Betrachten wir als nächstes den Fall $p=100$, $r=1900$, $s=6$ und $b=3$, der in **Fig. 16** dargestellt ist. Hier sind weniger Bits und eine Verschiebung ungleich Null zu sehen. Die Linien um p und r haben einen Abstand von $2^s = 32$ und umschließen die Enden des Bereichs. Die Verschiebung ist mit 6 angegeben, und es stehen nur drei Bits für die Korrektur zur Verfügung, so dass die Korrekturwerte einen Abstand von 64 haben. Die möglichen Korrekturen sind die ganzen Zahlen von -8 bis einschließlich 7, wie durch die vertikalen Linien angezeigt.

[0079] In diesem Fall besteht die Lösung darin, die Korrektur von $c=-4$ zu wählen, was zu einem decodierten Wert von $d=1892$ und einem Fehler von $\text{abs}(r-d) = 8$ führt. Das Wrap-Around-Verhalten kann hier ausgenutzt werden, um ein gutes Ergebnis zu erzielen, aber dabei zeigt sich, dass eine Verschiebung ungleich Null einen geringeren Fehler als im vorherigen Fall ergeben kann, sogar mit weniger Bits.

[0080] Andere Szenarien sind möglich. Das vorherige Szenario betraf einen arithmetischen Unterlauf; Fälle, die einen arithmetischen Überlauf erfordern, sind ebenso möglich wie Fälle, in denen kein Über- oder Unterlauf vorliegt, und Fälle, in denen eine Korrektur einen Nullfehler ergibt.

[0081] Im Folgenden wird ein Pseudocode für einen Algorithmus vorgestellt, der bei vorzeichenlosen ganzen Zahlen $0 \leq p < 2048$, $0 \leq r < 2048$, einer vorzeichenlosen ganzzahligen Verschiebung $0 \leq s < 11$ und einer vorzeichenlosen ganzzahligen Bitbreite $0 \leq b \leq 11$ innerhalb einer endlichen Anzahl von Operationen (unabhängig von der Anzahl der b -Bit-Möglichkeiten für c) immer den bestmöglichen ganzzahligen Wert von c (zwischen -2^b und $(2^b)-1$ einschließlich, wenn $b > 0$, oder gleich 0, wenn $b = 0$) zurückgibt. Im unten abgebildeten Pseudocode für die Schritte 1-8 des sequentiellen Algorithmus stellen nichtmathematische kursive Texte in Klammern Kommentare dar, und Modulo-Operationen (mod) werden als positive Werte angesehen.

(Frühe Prüfung für den Null-Bit-Fall) Wenn b gleich 0 ist, wird 0 zurückgegeben.

(Der Bereich der darstellbaren Werte um 0 mit angewandter Verschiebung ist $-nR \dots pR-1$) Set $nR = 2^{b-1+s}$, $pR = nR - 2^s$.

(Differenz in $\mathbb{Z}$) Setze vorzeichenbehaftete ganze Zahl $d = r - p$.

(Liegt der Referenzwert zwischen den beiden extremen Korrekturen?)

If $(d \bmod 2048) > pR$ und $2048 - (d \bmod 2048) > nR$:

- a. *(Setze iLo und iHi auf extreme Korrekturwerte)* Set $iLo = -2^{(b-1)}$, $iHi = 2^{(b-1)} - 1$.
- b. Fahren Sie mit dem übernächsten Schritt „Fehler berechnen ...“ fort.

Andernfalls: (Der Referenzwert liegt zwischen zwei darstellbaren Werten; finde diese in $\mathbb{Z}/2^{11}\mathbb{Z}$; dann muss die ideale Korrektur einer der beiden Werte sein).

- Setze uD auf $d - 2048$ wenn $d > pR$, $d + 2048$ wenn $d < -nR$, und d sonst.
- Setze iLo auf $\text{floor}\left(\frac{uD}{2^s}\right)$ ein, wobei für die Division Fließkomma-Arithmetik verwendet wird.
- Setze iHi auf $iLo + 1$.

(Berechnung des Fehlers für iLo) Setze eLo auf die absolute Differenz von r und das Ergebnis der Substitution $\text{correction} = iLo$ in die obige Formel 1 ein.

(Fehler für iHi berechnen) Setze eHi auf die absolute Differenz von r , und das Ergebnis der Substitution $\text{correction} = iHi$ in Formel 1 einsetzen.

(Wähle die Option mit dem geringeren Fehler) If $eLo \leq eHi$ ist, gib iLo zurück. Andernfalls wird iHi zurückgegeben.

[0082] Grundsätzlich erkennt der Pseudocode-Algorithmus, dass die Bezugslinie r immer zwischen zwei Korrekturwertlinien innerhalb des darstellbaren Bereichs liegen oder genau mit einer Korrekturwertlinie innerhalb des Bereichs übereinstimmen muss. Der Algorithmus wechselt zwischen zwei verschiedenen Fällen (der Referenzwert liegt zwischen den beiden extremen Korrekturen oder der Referenzwert liegt zwischen zwei darstellbaren Werten) und wählt den Fall mit dem geringeren Fehler. Grundsätzlich bietet der Wrap-Around-Fall eine „Abkürzung“ für Situationen, in denen die vorhergesagten und die Referenzwerte in einer Ausführungsform nahe an den entgegengesetzten Enden des bitbegrenzten Verschiebungsbereichs liegen.

Kompressor - Verbesserung der Berechnung von Verschiebungswerten durch Verwendung von Wrapping

[0083] Eine Minimierung der Größe der Verschiebung auf jeder Stufe für jeden Vertextyp kann die Kompressionsqualität verbessern. Der Abstand zwischen den darstellbaren Korrekturen (siehe die in den **Fig. 17** und **18** dargestellten möglichen decodierten Werte) ist proportional zu 2^{hoch} der Verschiebung für diese Stufe und diesen Vertextyp. Eine Verringerung der Verschiebung um 1 verdoppelt die Dichte der darstellbaren Werte, halbiert aber auch die Länge der Spanne, die durch die minimalen und maximalen Korrekturen dargestellt wird. Da die Algorithmen zur Berechnung der Korrekturen das Wrap-Around-Verhalten nutzen können, kann die Berücksichtigung des Wrap-Around-Verhaltens bei der Berechnung der Mindestverschiebung, die erforderlich ist, um alle Korrekturen für eine Stufe und einen Vertextyp abzudecken, die Qualität verbessern.

[0084] Betrachten wir beispielsweise eine Korrektorebene und einen VertexTyp, bei denen die Differenzen $\text{mod } 2048$ zwischen jedem Referenz- und vorhergesagten Wert wie in **Fig. 17** verteilt sind. Im ersten Beispiel gibt es große Verschiebungen und eine endliche Anzahl von Bits (z. B. 4 Bits in diesem Beispiel - was 16 mögliche verschobene Korrekturwerte ergibt), die großen Verschiebungen führen zu einer großen Differenz zwischen den möglichen verschobenen Korrekturwerten, was bedeutet, dass es nicht viel Präzision zwischen den verschobenen möglichen Korrekturwerten gibt. Wird der Betrag der Verschiebung verringert, wird der Abstand zwischen den möglichen verschobenen Korrekturwerten kleiner und die Genauigkeit nimmt zu (siehe **Fig. 18**). Durch eine gute Wahl der Verschiebungswerte wird also die Kompression verbessert, da die verschobenen Korrekturwerte näher am Referenzwert liegen.

[0085] In **Fig. 17** sind die verlustfreien Korrekturen als d_0, d_1, d_2 (in diesem Beispiel $+50, +100$ bzw. $+1900$ (-148)) dargestellt. Basierend auf diesen Werten scheint es, dass Verschiebungswerte erforderlich sind, die den gesamten Bereich zwischen $+100$ und -148 abdecken, was auf große (aber ungenaue) Verschiebungswerte hindeutet, die zu höheren Fehlern aufgrund der Quantisierung führen werden. Daher könnte ein Algorithmus, der das Wrapping nicht berücksichtigt, zu dem Schluss kommen, dass er die maximal mögliche Verschiebung benötigt, um alle diese Unterschiede zu überbrücken. Siehe **Fig. 18**. Da die Korrekturen jedoch auch negativ sein können und sich umdrehen können, kann eine kleinere Verschiebung zu qualitativ besseren Ergebnissen führen.

[0086] Ein möglicher Algorithmus könnte folgendermaßen aussehen. Subtrahieren Sie 2048 von (Differenzen $\text{mod } 2048$), die größer als 1024 sind, so dass alle umhüllten Differenzen w_i innerhalb des Bereichs der

ganzen Zahlen - 1024...1023 einschließlich liegen. Siehe **Fig. 18**. Auf diese Weise werden alle Werte innerhalb einer Teilmenge des ursprünglichen Bereichs platziert - und Werte, die vorher weit voneinander entfernt waren, werden so umgewandelt, dass sie nun nahe beieinander liegen. Die sich daraus ergebenden wesentlich kleineren Verschiebungen kommen dem Referenzwert sehr viel näher.

[0087] Berechnen Sie dann die Verschiebung s bei der Pegel-Bitbreite b als die minimale Zahl s , bei der

$$2^s (2^b - 1) \geq \max (w_i)$$

und

$$-2^s (2^b) \leq \min (w_i).$$

[0088] Beispielsweise kann diese Transformation als Teil von „Durchgang eins“ eines Codierers zur Berechnung verlustfreier Korrekturen einbezogen werden (siehe **Fig. 14**). In Durchgang eins wird also der Verlust für jeden Vertex und Vertextyp verfolgt, die verlustfreien Korrekturen werden berechnet, die Transformation in eine Teilmenge des Bereichs durchgeführt und die minimalen und maximalen verlustfreien Korrekturen über diese Teilmenge des Bereichs verfolgt. Der optimale Verschiebungswert wird basierend auf den minimalen und maximalen verlustfreien Korrekturen berechnet. Im zweiten Durchgang werden die verlustbehafteten Korrekturen aus den vorhergesagten Werten, den Verschiebungswerten und den verlustfreien Korrekturen berechnet. Diese verlustbehafteten Korrekturen und die Verschiebungen werden zusammengepackt und in den komprimierten Block geschrieben.

Kompressor - Verwendung von Verschiebungsbereichen in der Codierungserfolgsmetrik

[0089] Es wird ein Verfahren zur Interpretation von Skalierungsinformationen als ein Wichtigkeitssignal pro Vertex und ein Verfahren zur Verwendung der Wichtigkeit pro Vertex zur Änderung der Fehlermetrik des Verschiebungs-Codierers beschrieben. Dadurch wird die Qualität dort verbessert, wo sie benötigt wird, und die Größe dort reduziert, wo die Qualität nicht so wichtig ist.

[0090] Wie oben beschrieben, hat jeder Vertex einen Bereich, über den er verschoben werden kann und der durch die Spezifikation der Verschiebungskarte gegeben ist. Bei der Prisma-Darstellung beispielsweise skaliert die Länge dieses Bereichs mit der Länge des interpolierten Richtungsvektors und der interpolierten Skala. Die decodierte Eingabe und Ausgabe des codierten Formats hat dagegen einen festen Bereich und eine feste Genauigkeit (UNORM11-Werte), wie oben beschrieben. Dies bedeutet, dass die Minimal- und Maximalwerte zu unterschiedlichen absoluten Verschiebungen in verschiedenen Bereichen eines Netzes führen können - und daher kann ein UNORM11-Fehler einer bestimmten Größe für einen Teil eines Netzes zu einer mehr oder weniger starken visuellen Beeinträchtigung im Vergleich zu einem anderen führen.

[0091] In einer Ausführungsform kann dem Codierer eine Wichtigkeit pro Netz-Vertex (z. B. eine „Salienz“) zur Verfügung gestellt werden, z. B. über die Fehlermetrik. Eine Möglichkeit ist, dass es sich dabei um den möglichen Verschiebungsbereich jedes Vertex im Objektraum handelt (z. B. Abstand x Skala in der Prisma-Darstellung - was ein Maß für die Unterschiede und somit für den berechneten Fehler im Objektraum ist); dies könnte jedoch auch die Ausgabe eines anderen Prozesses sein oder von einem Benutzer gesteuert werden. Beispielsweise könnte ein Künstler angeben, welche Vertices eine höhere „Wichtigkeit“ haben, um bessere Abbildungsergebnisse zu erzielen, z. B. um das Gesicht und die Hände einer Figur herum eine höhere Qualität zu erzielen als um ihre Kleidung.

[0092] Die Wichtigkeit der Netz-Vertices wird linear interpoliert, um einen „Wichtigkeitsgrad“ für jeden μ -Netz-Vertex zu erhalten. Dann wird innerhalb der Fehlermetrik der komprimierte gegenüber dem unkomprimierten Fehler für jedes Element der Fehlermetrik mit einer „Wichtigkeit“ der Fehlermetrik gewichtet, die sich aus dem „Wichtigkeitsgrad“ der μ -Netz-Vertices des Elements ergibt. Diese werden dann akkumuliert und der daraus resultierende akkumulierte Fehler - der nun basierend auf der „Wichtigkeit“ gewichtet ist - wird mit der/den Fehlerbedingung(en) verglichen. Auf diese Weise wählt der Kompressor häufig stärker komprimierte Formate für Regionen des Netzes mit geringerer „Wichtigkeit“ und weniger komprimierte Formate für Regionen des Netzes mit höherer „Wichtigkeit“.

[0093] Die obigen Ausführungen erklären, wie ein Kompressor ein durch ein Basisdreieck definiertes Mikronetz komprimieren kann. Durch die Anordnung der μ -Netz-Typen vom am stärksten komprimierten bis zum am wenigsten komprimierten, wie in **Fig. 5** dargestellt, können die Ausführungsformen mit der direkten Codierung von Teildreiecken in der „Reihenfolge des Kompressionsverhältnisses“ unter Verwendung des oben beschriebenen P&C-Schemas fortfahren, beginnend mit dem am stärksten komprimierten μ -Netz-Typ, bis ein gewünschtes Qualitätsniveau erreicht ist. Dieses Schema ermöglicht eine parallele Codierung bei maximaler Kompression und ohne die Einführung nicht übereinstimmender Verschiebungswerte entlang der von den Teildreiecken gemeinsam genutzten Kanten.

[0094] **Fig. 20A** veranschaulicht den Fall, dass zwei Teildreiecke eine gemeinsame Kante haben. Beide Teildreiecke werden mit der gleichen Rate tesseliert, aber mit unterschiedlichen μ -Netz-Typen codiert. In der **Fig.** dient der Abstand zwischen den beiden Dreiecken nur der besseren Veranschaulichung.

[0095] Im gezeigten Beispiel werden die Mikrospitzen mit einem Bezeichner wie „S1“ versehen. Dabei steht der Buchstabe „S“ für „Subdivision“ und die nachfolgende Zahl für die Nummer der Subdivision. Man sieht also, dass die „S0“-Vertices oben und unten an der gemeinsamen Kante für jedes Teildreieck auf der Unterteilungsstufe Null gespeichert werden - und zwar im unkomprimierten Format. Eine erste Unterteilung erzeugt die „S1“-Vertices auf Unterteilungsstufe 1, und eine zweite Unterteilung erzeugt die „S2“-Vertices auf Unterteilungsstufe 2.

[0096] Um Risse entlang der gemeinsamen Kante zu vermeiden, müssen die decodierten Verschiebungswerte der beiden Dreiecke übereinstimmen. S0-Vertices stimmen überein, da sie immer unkomprimiert codiert werden. S1- und S2-Vertices stimmen nur dann überein, wenn (1) das Teildreieck in der Reihenfolge des Kompressionsverhältnisses codiert ist und (2) die mit einem stärker komprimierten μ -Netz-Typ codierten Verschiebungswerte immer durch weniger komprimierte μ -Netz-Typen dargestellt werden können. Die zweite Bedingung besagt, dass für eine bestimmte Unterteilungsstufe ein weniger komprimierter μ -Netz-Typ niemals weniger Bits verwenden sollte als ein stärker komprimierter μ -Netz-Typ. Wenn beispielsweise das rechte Teildreieck einen kompakteren μ -Netz-Typ verwendet als das linke Teildreieck, wird das rechte Teildreieck zuerst codiert. Außerdem werden die Verschiebungswerte der Kante des rechten Teildreiecks (d. h. die Kante, die mit dem rechten Teildreieck gemeinsam genutzt wird) nach der Codierung kopiert, um die Verschiebungswerte des linken Teildreiecks zu ersetzen. Eigenschaft (2) stellt sicher, dass die Verschiebungswerte entlang der Kante des linken Teildreiecks nach der Kompression verlustfrei codiert werden, wodurch eine perfekte Übereinstimmung entlang der gemeinsamen Kante entsteht.

[0097] In diesem Beispiel werden diese beiden Teildreiecke mit unterschiedlichen Mikronetztypen codiert (angenommen beispielsweise, das linke Teildreieck ist stärker komprimiert als das rechte). Wie oben beschrieben, arbeitet der Kompressor in einer Ausführungsform von stärker komprimierten zu weniger komprimierten Formaten, so dass in diesem Fall die Verschiebungen für das linke Teildreieck zuerst codiert werden. Nehmen wir also an, dass die Verschiebungen für das linke Teildreieck bereits erfolgreich codiert wurden und ein Prozessor nun versucht, die Verschiebungen für das rechte Teildreieck zu codieren - und insbesondere die Verschiebungen für die Mikroecken des rechten Dreiecks, die auf der gemeinsamen Kante der beiden Dreiecke liegen. Die Verschiebungswerte, die für die Mikrovertices der gemeinsamen Kante des rechtsseitigen Teildreiecks codiert werden sollen, müssen Bit für Bit mit den bereits codierten Verschiebungswerten für die Vertices der gemeinsamen Kante des linksseitigen Teildreiecks übereinstimmen. Wenn sie nicht genau übereinstimmen, kann es zu Rissen kommen.

[0098] Wenn die Vertices der gemeinsamen Kante des rechten Teildreiecks Bit für Bit mit den Vertices der gemeinsamen Kante des linken Teildreiecks übereinstimmen sollen, muss die Anzahl der Bits, die zur Darstellung der Verschiebung für das rechte Teildreieck verwendet werden, gleich oder größer sein als die Anzahl der Bits, die zur Darstellung der Verschiebung für das linke Teildreieck verwendet werden. Aus diesem Grund haben die Vertices, die sich auf der gemeinsamen Kante des linken und rechten Teildreiecks gegenüberliegen, die gleiche Unterteilungsstufe - beispielsweise passt ein linker S0-Vertex zu einem rechten S0-Vertex, ein linker S1-Vertex zu einem rechten S1-Vertex, ein linker S2-Vertex zu einem rechten S2-Vertex und so weiter. Daher kann ein weniger komprimiertes Verschiebungsformat bei Kanten, die von Teildreiecken gemeinsam genutzt werden, niemals weniger Bits für eine bestimmte Unterteilungsstufe verwenden als ein gegenüberliegendes, stärker komprimiertes Verschiebungsformat. Stellt man sich beispielsweise die Aufzeichnung auf einer horizontalen Linie wie in einer Tabellenkalkulation vor, so bildet die Anzahl der Bits, die zur Darstellung der Vertices für eine bestimmte Unterteilungsstufe zugewiesen werden, für alle verschiede-

nen Mikronetztypen, sortiert von stärker komprimiert bis weniger komprimiert, eine monotone Folge, die zunimmt oder sich nicht ändert und nicht abnehmen kann. Mit anderen Worten: Es kann niemals weniger Bits für eine bestimmte Unterteilungsstufe im weniger komprimierten Typ geben als im stärker komprimierten Typ. In Beispielen wird diese Einschränkung dem Codierungsschema auferlegt, um die Wasserdichtigkeit zu garantieren, wobei davon ausgegangen wird, dass der Codierungsalgorithmus deterministisch ist (er enthält keine stochastischen Komponenten).

[0099] Fig. 20B ist etwas komplizierter, da die Tesselierungsraten der Teildreiecke links und rechts nun unterschiedlich sind. **Fig. 20B** veranschaulicht insbesondere den Fall einer gemeinsamen Kante zwischen Dreiecken mit unterschiedlichen Tessellierungsraten (2x Unterschied), die aber mit demselben μ -Netztyp codiert sind. Um sicherzustellen, dass die decodierten Verschiebungen auf beiden Seiten der gemeinsamen Kante übereinstimmen, müssen die auf einer bestimmten Stufe codierten Werte auch auf der nächsten Unterteilungsstufe darstellbar sein (siehe z. B. S1-S2 und S0-S1 Vertex-Paare). Es gibt zwar viele Möglichkeiten, dies zu erreichen, aber in einer bestimmten Ausführungsform kann dies nur erreicht werden, wenn (1) Teildreiecke mit niedrigerer Tessellierungsrate vor Teildreiecken mit höherer Tessellierungsrate codiert werden und (2) für einen bestimmten μ -Netztyp die Korrekturbitbreite für die Unterteilungsstufe N gleich oder kleiner ist als für die Stufe N-1. Mit anderen Worten, die letztgenannte Eigenschaft diktiert, dass für einen μ -Netztyp die Anzahl der Bits, sortiert nach Unterteilungsstufe, eine monoton abnehmende Folge bilden sollte. So wird beispielsweise das linke Dreieck in **Fig. 20B** zuerst codiert, und seine Verschiebungswerte werden nach der Decodierung in die Vertices kopiert, die von den drei Dreiecken auf der rechten Seite gemeinsam genutzt werden, bevor mit ihrer Codierung fortgefahren wird.

[0100] In diesem Beispiel sehen wir also 2x mehr Vertices auf der rechten Seite als auf der linken Seite. Einige Kanten-Vertices, die von den Teildreiecken auf der linken und rechten Seite gemeinsam genutzt werden, gehören nicht zur selben Unterteilungsstufe. Beispielsweise stehen „S2“-Vertices auf dem linken Teildreieck S1-Vertices auf dem rechten Teildreieck gegenüber, und S1-Vertices auf dem linken Teildreieck stehen S0-Vertices auf dem rechten Teildreieck gegenüber. Daher ist die Anzahl der Bits, die zur Codierung derselben gemeinsamen Vertices für das linke und das rechte Teildreieck zugewiesen werden, nicht unbedingt dieselbe.

[0101] Insbesondere werden in einer Ausführungsform den höheren Unterteilungsstufen (Tessellierungsrate) weniger Bits pro Vertex für die Verschiebungscodierung zugewiesen, so dass die Anzahl der für die Codierung von beispielsweise S1 verfügbaren Bits wahrscheinlich höher ist als die Anzahl der für die Codierung von beispielsweise S2 verfügbaren Bits. Wie bereits erwähnt, ist es jedoch bei der Verarbeitung von Teildreiecken mit unterschiedlichen Tessellierungsraten in einigen Ausführungsformen vorzuziehen, Teildreiecke mit niedrigerer Tessellierungsrate zu codieren, bevor angrenzende Dreiecke mit höherer Tessellierungsrate codiert werden, um zu gewährleisten, dass die mit dem angrenzenden Teildreieck verbundenen Informationen Bit für Bit übereinstimmen können. Da für die Codierung des rechten Teildreiecks mit höherer Tessellierungsrate weniger Bits zur Verfügung stehen, kann nicht garantiert werden, dass die Vertex-Codierung des rechten Teildreiecks mit höherer Tessellierungsrate mit der des linken Teildreiecks mit niedrigerer Tessellierungsrate vergleichbar ist. Die erste Codierung des Teildreiecks mit der niedrigeren Tessellierungsrate auf der linken Seite stellt sicher, dass das Teildreieck mit der höheren Tessellierungsrate auf der rechten Seite dieselben Vertex-Informationen darstellen kann, solange innerhalb eines Mikronetztyps die Anzahl der Bits zur Codierung der Verschiebung für immer tiefere/rekursive Unterteilungsstufen nicht zunimmt:

Bits für Unterteilungsstufe $k \leq$ Bits für Unterteilungsstufe j wobei j eine beliebige Stufe ist, die weniger unterteilt ist (geringeres Mosaikverhältnis) als k .

[0102] Zusammenfassend lässt sich sagen, dass bei der Codierung eines Dreiecksnetzes gemäß einigen hochleistungsfähigen Ausführungsformen die folgenden Ordnungsbeschränkungen angenommen werden, um Risse im Netz zu vermeiden:

- Teildreiecke werden in aufsteigender Reihenfolge der Tessellierungsrate codiert (angrenzende Teildreiecke mit der niedrigeren Tessellierungsrate zuerst codieren); und
- Teildreiecke mit der gleichen Tessellierungsrate werden in absteigender Reihenfolge der Kompressionsrate codiert (beginnend mit der höchsten gewünschten Kompressionsrate).

[0103] Daher gelten in einigen Ausführungsformen die folgenden Einschränkungen für die Konfigurationen der Korrekturbitbreiten:

- Bei einem gegebenen μ -Netz-Typ verwendet eine Unterteilungsstufe nie weniger Bits als die nächste (komprimiertere) Stufe; und
- Bei einem gegebenen Unterteilungsstufe verwendet ein μ -Netz-Typ niemals weniger Bits als ein komprimierterer Typ.

[0104] Die obige Regel berücksichtigt Netztypen, die die gleiche Anzahl von Mikrodreiecken (d. h. die gleiche Anzahl von Unterteilungen) darstellen, aber unterschiedliche Speicheranforderungen haben (z. B. 1024 Mikrodreiecke in 128B oder 64B).

[0105] In einer Ausführungsform wird die effektive Anzahl der Bits, die zur Darstellung eines Verschiebungswertes verwendet werden, durch die Summe der Korrektur- und Verschiebungsbreiten bestimmt. Im Beispiel von **Fig. 20B** wird den Vertices auf einer Kante eines Teildreiecks, die mit einem anderen Teildreieck des Netzes geteilt wird, eine Nullkorrektur zugewiesen - ihre Verschiebungswerte sind das reine Ergebnis der Vorhersage, d. h. der Interpolation oder des Durchschnitts der Verschiebungswerte ihrer benachbarten Vertices auf der Kante. Darüber hinaus kann in einer Ausführungsform eine Technik, die wir „Dezimierung“ nennen (bei der die Hardware bei der Erstellung von 3D-Darstellungen von Mikrodreiecken für Strahlenschnittprüfungen Vertices löscht), verwendet werden, um die Topologie von Teildreiecken mit angrenzenden Kanten zu ändern, um T-Kreuzungen zu vermeiden.

[0106] **Fig. 20C** zeigt eine weitere Beispielsituation, in der zwei benachbarte Teildreiecke unterschiedliche Unterteilungs-Tessellierungsraten haben und auch mit unterschiedlichen Mikronetztypen codiert wurden. Gemäß den obigen Beispielbedingungen wird das linke Teildreieck vor dem rechten Teildreieck codiert, da es eine geringere Auflösung und einen komprimierteren Mikronetztyp hat. Die codierten Werte des linken Teildreiecks entlang der gemeinsamen Kante werden dann auf das rechte Teildreieck kopiert, um das rechte Teildreieck zu codieren. Es wird sich jedoch zeigen, dass das rechte Teildreieck mehr Vertices aufweist als das linke Teildreieck. In diesem speziellen Fall, in dem die Mikronetztypen der beiden Teildreiecke nicht identisch sind, setzen Ausführungsbeispiele eine Markierung an der rechten Dreieckskante, die den Codierer auffordert, die codierten Vertices des rechten Teildreiecks zu inspizieren und zu überprüfen, um sicherzustellen, dass sie ohne Fehler codiert wurden. Zur Verdeutlichung: Die Vertices im rechten Dreieck, die fehlerfrei codiert werden müssen, sind diejenigen, die auch im linken Dreieck vorhanden sind (übereinstimmen), d. h. die bei 2 und 2 und 1 und 1. Wird ein Verlust festgestellt, markiert der Codierer das Teildreieck als nicht erfolgreich codiert, und der Codierer versucht es erneut mit einem weniger komprimierten Mikronetztyp, wie im oben genannten Beispiel. Es ist anzumerken, dass der Codierer in einem Beispiel den Codierungsprozess mit einem Format wiederholen könnte, das mehr Bits pro Vertex-Verschiebung bereitstellt (z. B. ein volles Cacheline-Format im Gegensatz zu einem halben Cacheline-Format). Die Anzahl der Unterteilungen konstant zu halten, während die Anzahl der Bits/Speicher geändert wird, ist gleichbedeutend mit einer Änderung des Mikronetztyps, d. h. in einer Ausführungsform wird ein Mikronetztyp durch die Anzahl der Unterteilungsstufen UND den zugehörigen Speicherplatz bestimmt. Um sicherzustellen, dass die Ausgabe des Codierers konform und kompatibel mit Hardware-Decodern ist, die nur mit vorgegebenen Codierformaten arbeiten, kann dies den Codierer in einigen Fällen dazu zwingen, einen anderen Mikronetztyp für das Teildreieck auf der rechten Seite zu wählen, so dass es den gleichen Mikronetztyp wie das Teildreieck auf der linken Seite hat.

[0107] Diese Beispielbeschränkungen ermöglichen es, dass verschiedene Teildreiecke im Netz unabhängig voneinander (sowohl bei der Codierung als auch bei der anschließenden Decodierung) durch eine hochleistungsfähige, asynchrone Parallelverarbeitung verarbeitet werden können, während gleichzeitig sichergestellt wird, dass diese Prozesse unabhängig voneinander dieselben Verschiebungswerte für Vertices ableiten, die zwischen benachbarten Teildreiecken bei der Codierung des Netzes gemeinsam genutzt werden, und dass Situationen vermieden werden, in denen eine Datenrepräsentation mit höherer Präzision in eine geringere Anzahl von Bits gequetscht wird, was zu einem Verlust an numerischer Auflösung führen würde und somit eine Bit-für-Bit-Übereinstimmung der Verschiebungswerte an den Schnittpunkten von Vertices verschiedener Teildreiecke unmöglich machen würde. Es ist ein bisschen so, als würde man verschiedene Augenzeugen eines wichtigen Ereignisses unabhängig voneinander in verschiedenen Räumen befragen, ohne sie miteinander sprechen zu lassen, und jeder Zeuge würde genau denselben Ablauf der Ereignisse bestätigen.

Kompressor - Netz-Codierer (einheitlich)

[0108] Der nachstehende, in **Fig. 21, 21A, 21B** dargestellte Pseudocode veranschaulicht, wie die Codierung eines gleichmäßig tesselierten Netzes gemäß einigen Ausführungsformen funktioniert:

```

für jede Mikronetztyp (von der höchsten zur niedrigsten Kompression):
  foreach nicht verschlüsseltes Teildreieck:
    codiere Teildreieck
      if erfolgreich, then markiere Teildreieck als codiert
        foreach teilweise codierte Kante
          aktualisiere Referenzverschiebungen in noch nicht
codierten Teildreiecken.

```

[0109] Zu beachten ist, dass jedes Teildreieck einen Satz von Referenzverschiebungswerten enthält, die die Zielwerte für die Kompression sind. Eine Kante, die von einem codierten Teildreieck und einem oder mehreren noch nicht codierten Teildreiecken geteilt wird, gilt als „teilweise codiert“. Um eine rissfreie Kompression zu gewährleisten, werden die dekomprimierten Verschiebungswerte an die noch nicht codierten Teildreiecke weitergegeben, wo sie deren Referenzwerte ersetzen.

[0110] Das Flussdiagramm in **Fig. 22** und die Flipchart-Animationssequenz der Dreiecks-Tessellierungsstufen in den **Fig. 23A-23F** zeigen eine Beispielimplementierung des obigen Pseudocodes. Ein Beispielalgorithmus beginnt mit der am stärksten komprimierten Implementierung für den gewünschten Detaillierungsgrad - in diesem Fall ein Dreieck der Stufe 6, das so tesseliert ist, dass es 4096 Mikrodreiecke hat. Wie **Fig. 22** zeigt, verwendet der Builder die obigen Algorithmen zur Erstellung von Verschiebungsblöcken und prüft dann, ob die Qualität akzeptabel ist oder nicht (dieser Test kann auf der Grundlage einer Reihe verschiedener Heuristiken, Metriken künstlicher Intelligenz, tiefer neuronaler Netze oder anderer Tests durchgeführt werden). Wenn die Qualität akzeptabel ist, schreibt der Builder die Verschiebungsblöcke aus und ist damit fertig. Ist die Qualität inakzeptabel, verringert der Builder das Kompressionsverhältnis und versucht es erneut. Eine solche Verringerung der Kompression kann bedeuten, dass mehr unterteilt wird oder ein anderer Speicher für die gleiche Anzahl von Mikrodreiecken/Unterteilungen verwendet wird (siehe **Fig. 23B**).

[0111] In diesem Fall hat der Builder das Teildreieck in **Fig. 23A** in vier Dreiecke der Stufe 5 unterteilt, die jeweils 1024 Mikrodreiecke definieren. Wie **Fig. 22** zeigt, wird der Vorgang wiederholt, um die Informationen des Verschiebungsblocks zu erstellen und zu testen. Nehmen wir nun an, dass die drei unteren Teildreiecke eine akzeptable Qualität auf der Tessellierungsstufe 5 liefern, das obere Dreieck jedoch nicht. Das bedeutet, dass der Builder das obere Dreieck auf Tessellierungsstufe 4 unterteilen muss (siehe **Fig. 23B**). In der Situation von **Fig. 23C** unterscheidet sich jedoch der Kompressionsverhältnis des oberen Dreiecks vom Kompressionsverhältnis der unteren Dreiecke.

[0112] Hier macht sich der Algorithmus die Einschränkung zunutze, dass die weniger komprimierten Vertex-Formate der oberen Teildreiecke in der Lage sein müssen, die stärker komprimierten Vertex-Formate der unteren Teildreiecke darzustellen. Dies mag wie eine überflüssige Anforderung klingen - wird ein weniger komprimiertes Format nicht immer in der Lage sein, die Werte eines stärker komprimierten Formats darzustellen? Nicht unbedingt - wenn beide Formate verlustbehaftete Kompression verwenden, besteht die Möglichkeit, dass ein weniger komprimiertes Format bestimmte Werte nicht darstellen kann, die ein stärker komprimiertes Format darstellen kann. Wenn eine solche Situation eintreten würde, wären Risse im Netz die Folge. In Ausführungsbeispielen wird daher eine Beschränkung eingeführt, um dies zu verhindern - nämlich, dass ein weniger komprimierter Typ immer alle Werte eines stärker komprimierten Typs darstellen kann.

[0113] Aber selbst diese Einschränkung reicht nicht aus, um zu garantieren, dass keine Risse entstehen. Denn die Verschiebungswerte, die der Dekompressor von den untersten Teildreiecken an der gemeinsamen Kante mit dem obersten Teildreieck zurückgewinnt, sind nicht die ursprünglichen Verschiebungen des Netzes, sondern haben einen verlustbehafteten Kompressionsprozess durchlaufen. Dementsprechend stellen wir in einer Ausführungsform die Bit-für-Bit-Anpassung über die Präzision und übertragen die erfolgreich komprimierten und dann wiederhergestellten Werte von den Vertices der unteren Teildreiecke auf die gemeinsame Kante mit dem obersten Teildreieck, wobei die übertragenen Werte die eigenen Vertex-Verschiebungen des obersten Teildreiecks ersetzen. Durch die Weitergabe dieser Verschiebungswerte, die aus der Dekompression der Vertices des unteren Teildreiecks zurückgewonnen wurden, an das weniger komprimierte oberste Teildreieck - und mit der Einschränkung, dass das weniger komprimierte Format des obersten Teildreiecks genau diese weitergegebenen Werte aus einem stärker komprimierten Format darstellen kann - kann nun garantiert werden, dass die Vertex-Verschiebungen, die der Decoder für das oberste Teildreieck zurückgewinnt, Bit für Bit identisch mit den entsprechenden Vertex-Verschiebungen sind, die der Decoder für die untersten Teildreiecke entlang der gemeinsamen Kante zurückgewinnt - ohne dass der Decoder beide gleichzeitig decodieren muss oder weiß, dass es eine gemeinsame Kante gibt.

[0114] Der Algorithmus versucht dann, die vier unterteilten oberen Teildreiecke erneut zu komprimieren, wie in **Fig. 23D** gezeigt, wobei die oben beschriebenen propagierten Werte verwendet werden. Nehmen wir nun an, dass, wie in **Fig. 23E** gezeigt, alle Dreiecke bis auf das mittlere Dreieck eine akzeptable Qualität aufweisen, dass aber das mittlere Dreieck mit einer noch niedrigeren Tesselierungsrate neu komprimiert werden muss.

[0115] Wie **Fig. 23E** zeigt, werden alle drei Kanten des mittleren Dreiecks mit anderen Teildreiecken geteilt. In diesem Fall werden die wiederhergestellten Verschiebungen für alle Vertices des mittleren Teildreiecks von den bereits komprimierten umliegenden Teildreiecken übernommen, um eine Bit-für-Bit-Übereinstimmung mit den Vertices an gemeinsamen Kanten sicherzustellen. **Fig. 23F** zeigt, dass das mittlere Dreieck weiter in Teildreiecke der Stufe 3 unterteilt ist, die möglicherweise überhaupt nicht komprimiert werden, sondern einfach die dekomprimierten Verschiebungswerte von den gemeinsamen Kanten in unkomprimierter Form wiedergeben.

Kompressor - Netz-Codierer (adaptiv)

[0116] Wie unten gezeigt, wird bei der Codierung von adaptiv tesselierten Netzen eine zusätzliche äußere Schleife verwendet, um die Teildreiecke in aufsteigender Reihenfolge der Tesselierungsrate zu verarbeiten:

```
foreach Auflösung des Basisdreiecks (von niedriger zu höherer
Auflösung):
    foreach Netztyp (von der höchsten zur niedrigsten
Kompression):
        foreach nicht codiertes Dreieck:
            codiere Teildreieck
                if erfolgreich, then markiere Teildreieck als codiert
        foreach teilweise codierte Kante:
            aktualisiere Referenzverschiebungen in noch nicht codierten
            Teildreiecken.
```

[0117] Die hier beschriebene Kompressionstechnik geht nicht davon aus, ob das zu komprimierende Netz vielfältig ist oder nicht, und daher können wir auch nicht-vielfältige Netze problemlos komprimieren. Diese Eigenschaft kann sehr wichtig sein (oft sind Assets aus Spielen nicht mannigfaltig) und macht die Beispielverkörperung robuster.

[0118] Man beachte, dass bei der Aktualisierung der Referenzverschiebungen für Kanten, die mit Teildreiecken geteilt werden, die eine 2x höhere Tesselierungsrate verwenden, nur jeder zweite Vertex betroffen ist (siehe **Fig. 20B**), während in einer Ausführungsform die verbleibenden Vertices gezwungen sind, Nullkorrekturen zu verwenden, um die Verschiebungsneigung an der gemeinsamen Kante des Teildreiecks mit niedrigerer Auflösung anzupassen. Außerdem ist nicht gewährleistet, dass Teildreiecke mit höherer Auflösung, die aktualisierte Verschiebungswerte von Teildreiecken mit niedrigerer Auflösung „empfangen“, diese Werte auch darstellen können. Diese Fälle sind zwar eher selten, aber um Risse zu vermeiden, können die aktualisierten Referenzwerte verlustfrei codiert werden, damit sie immer mit ihrem Gegenstück an der Kante des Teildreiecks mit geringerer Auflösung übereinstimmen. Ist eine solche verlustfreie Codierung nicht möglich, scheitert die Codierung des Teildreiecks und es wird ein neuer Versuch mit einem weniger komprimierten μ -Netz-Typ unternommen.

Beispielhafte Implementierungen

[0119] **Fig. 24** zeigt ein Beispielsystem, das die hier beschriebene, nicht beschränkende Technologie implementiert. In dem gezeigten Beispiel wird die Druckvorlage 100 in geeigneter Form von mindestens einem Prozessor empfangen, der die Algorithmen des Builders ausführt, wie in den **Fig. 9, 14** und **22** gezeigt (Block 102). Der Builder codiert/komprimiert die Vorlage in ein Mikronetz, wie oben beschrieben, und speichert das codierte Mikronetz in einem nicht-übertragbaren Speicher als Beschleunigungsdatenstruktur, die eine Begrenzungsvolumenhierarchie 104 aufweist, die ein Datenformat wie in den **Fig. 6, 12** und **13** zeigt. Das codierte Mikronetz 104 wird (z. B. über ein Netzwerk, auf einem Speichermedium usw.) an einen Decoder/Dekompressor 106 übermittelt. Der Decoder/Dekompressor 106 kann Hardwareschaltungen und/oder mindestens einen Prozessor aufweisen, der die oben im Zusammenhang mit **Fig. 10** erörterten Algorithmen

ausführt, um die komprimierten Verschiebungswerte wiederherzustellen und sie einer GPU mit einer Grafikkipeline 108 für die Darstellung von Bildern auf einem Display 110 zur Verfügung zu stellen.

[0120] Fig. 25A zeigt, dass die Grafikkipeline Vertex-Shader 204 und Textur-Mapper 205 aufweist, die Vertex- und Verschiebungsdatenblöcke 202 in Cacheline-Größe aus einem Cache-Speicher über eine Speicherschnittstellenschaltung empfangen und Informationen an Rasterizer 206 weitergeben, die ihrerseits Fragmente unter Verwendung von Fragment-Shadern 208 erzeugen, die zur Bilddarstellung gemischt werden.

[0121] Fig. 25B zeigt eine alternative Grafikkipeline, bei der die Vertexdaten 202 einem Raytracing-Shader 202 und auch der Raytracing-Hardware 214 zugeführt werden, die die Strahlen- und Pfadverfolgung zur Erzeugung der Anzeige 110 verwenden. **Fig. 26** zeigt ein beispielhaftes Blockdiagramm eines Teils der Raytracing-Hardware 214, die den Dekompressor 106 enthält, der Verschiebungsblöcke aus dem Speichersystem empfängt und dekomprimierte Verschiebungswerte an eine Schnittpunkt-Testschaltung zum Testen gegen Strahlen liefert.

[0122] Fig. 25C zeigt eine kombinierte Grafikkipeline, die eine Mischung aus verschobenen mikronetz-basierten Ausgaben verwendet, die von Vertex-Shadern 204, Texture Mappern 205 und Ray Tracer 214 erzeugt werden, um Bilder zu erzeugen.

[0123] Bilder, die unter Anwendung einer oder mehrerer der hier offenbarten Techniken erzeugt wurden, können auf einem Monitor oder einem anderen Anzeigegerät angezeigt werden. In einigen Ausführungsformen kann das Anzeigegerät direkt mit dem System oder Prozessor gekoppelt sein, das die Bilder erzeugt oder wiedergibt. In anderen Ausführungsformen kann die Anzeigevorrichtung indirekt mit dem System oder Prozessor gekoppelt sein, beispielsweise über ein Netzwerk. Beispiele für solche Netzwerke sind das Internet, mobile Telekommunikationsnetzwerke, ein WIFI-Netzwerk sowie jedes andere drahtgebundene und/oder drahtlose Netzwerksystem. Wenn das Anzeigegerät indirekt gekoppelt ist, können die vom System oder Prozessor erzeugten Bilder über das Netzwerk an das Anzeigegerät gestreamt werden. Ein solches Streaming ermöglicht es beispielsweise, dass Videospiele oder andere Anwendungen, die Bilder rendern, auf einem Server oder in einem Datenzentrum ausgeführt werden und die gerenderten Bilder auf ein oder mehrere Benutzergeräte (wie einen Computer, eine Videospielekonsole, ein Smartphone, ein anderes mobiles Gerät usw.) übertragen und angezeigt werden, die physisch von dem Server oder Datenzentrum getrennt sind. Daher können die hier offenbart Techniken angewendet werden, um die gestreamten Bilder zu verbessern und um Dienste zu verbessern, die Bilder streamen, wie NVIDIA GeForce Now (GFN), Google Stadia und dergleichen.

[0124] Darüber hinaus können Bilder, die unter Anwendung einer oder mehrerer der hier offenbarten Techniken erzeugt wurden, zum Trainieren, Testen oder Zertifizieren von tiefen neuronalen Netzen (DNNs) verwendet werden, die zur Erkennung von Objekten und Umgebungen in der realen Welt eingesetzt werden. Solche Bilder können Szenen von Straßen, Fabriken, Gebäuden, städtischen Umgebungen, ländlichen Umgebungen, Menschen, Tieren und anderen physischen Objekten oder realen Umgebungen umfassen. Solche Bilder können zum Trainieren, Testen oder Zertifizieren von DNNs verwendet werden, die in Maschinen oder Robotern eingesetzt werden, um physische Objekte in der realen Welt zu manipulieren, zu handhaben oder zu verändern. Darüber hinaus können solche Bilder verwendet werden, um DNNs zu trainieren, zu testen oder zu zertifizieren, die in autonomen Fahrzeugen eingesetzt werden, um die Fahrzeuge durch die reale Welt zu navigieren und zu bewegen. Darüber hinaus können Bilder, die unter Anwendung einer oder mehrerer der hier offengelegten Techniken erzeugt wurden, verwendet werden, um den Benutzern solcher Maschinen, Roboter und Fahrzeuge Informationen zu vermitteln.

[0125] Darüber hinaus können Bilder, die unter Anwendung einer oder mehrerer der hier offengelegten Techniken erzeugt wurden, verwendet werden, um Informationen über eine virtuelle Umgebung wie das Metaverse, das Omniverse oder einen digitalen Zwilling einer realen Umgebung anzuzeigen oder zu vermitteln. Darüber hinaus können Bilder, die unter Anwendung einer oder mehrerer der hier offengelegten Techniken erzeugt wurden, dazu verwendet werden, Informationen auf einer Vielzahl von Geräten anzuzeigen oder zu übermitteln, einschließlich eines PCs (z. B. eines Laptops), eines Internet of Things (IoT)-Geräts, eines Handheld-Geräts (z. B. eines Smartphones), eines Fahrzeugs, eines Roboters oder eines beliebigen Geräts, das ein Display enthält.

[0126] Alle hierin zitierten Patente, Patentanmeldungen und Veröffentlichungen sind für alle Zwecke so einbezogen, als ob sie ausdrücklich aufgeführt wären.

[0127] Alle oben zitierten Patente und Veröffentlichungen werden durch Bezugnahme einbezogen, als ob sie ausdrücklich aufgeführt wären. Während die Erfindung im Zusammenhang mit dem, was derzeit als die meisten praktischen und bevorzugten Ausführungsformen beschrieben wurde, ist es zu verstehen, dass die Erfindung nicht auf die offenbart Ausführungsformen beschränkt werden, sondern im Gegenteil, soll verschiedene Modifikationen und gleichwertige Anordnungen innerhalb des Geistes und Umfangs der beigefügten Ansprüche enthalten.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

Diese Liste der vom Anmelder aufgeführten Dokumente wurde automatisiert erzeugt und ist ausschließlich zur besseren Information des Lesers aufgenommen. Die Liste ist nicht Bestandteil der deutschen Patent- bzw. Gebrauchsmusteranmeldung. Das DPMA übernimmt keinerlei Haftung für etwaige Fehler oder Auslassungen.

Zitierte Patentliteratur

- US 63245155 [0001]
- US 17946235 [0003]
- US 17946221 [0004]
- US 10825230 [0009]
- US 9437042 [0009]
- US 8860742 [0009]
- US 8698802 [0009]
- US 8570322 [0009]
- US 8558833 [0009]
- US 8471852 [0009]
- US 20110085736 [0009]
- US 7324105 [0009]
- US 7196703 [0009]
- US 6597356 [0009]
- US 6738062 [0009]
- US 6504537 [0009]

Zitierte Nicht-Patentliteratur

- Lee et al, „Displaced subdivision surfaces“, SIGGRAPH ,00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques July 2000 Pages 85-94 //doi.org/10.1145/344779.344829 [0009]
- Cantlay, „DirectX 11 Terrain Tessellation“, Nvidia (Januar 2011);
khronos.org/opengl/wiki/Tessellation#Tessellation_control_shader; Moreton et al, (2001);
Moreton, Tessellation and Geometry Shaders: Trends CMU 15-869 [0009]
- Dudash, „My Tessellation Has Cracks!“, Game Developer's Conference (2012); Sfarti et al, „New 3D Graphics Rendering Engine Architecture for Direct Tessellation of Spline Surfaces“, V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 224-231 (2005) [0009]
- N. Pietroni et al, „Almost Isometric Mesh Parameterization through Abstract Domains,“ IEEE Transactions on Visualization and Computer Graphics, vol. 16, no. 4, pp. 621-635, July-Aug. 2010, doi: 10.1109/TVCG.2009.96 [0009]
- Thonat et al, Tessellation-free displacement mapping for ray tracing, pp 1-16 ACM Transactions on Graphics Volume 40 Issue 6 No.: 282 (Dezember 2021) doi.org/10.1145/3478513.3480535 [0017]
- Wang et al, View-dependent displacement mapping, ACM Transactions on Graphics Volume 22 Issue 3, Juli 2003 pp 334-339, doi.org/10.1145/882262.882272 [0017]
- Lier et al, „A high-resolution compression scheme for ray tracing subdivision surfaces with displacement“, Proceedings of the ACM on Computer Graphics and Interactive Techniques Volume 1 Issue 2 August 2018 Article No.: 33 pp 1-17, doi.org/10.1145/3233308 [0017]
- Chun et al, „Multiple layer displacement mapping with lossless image compression“, International Conference on Technologies for E-Learning and Digital Entertainment Edutainment 2010: Entertainment for Education. Digital Techniques and Systems pp 518-528 [0017]
- Szirmay-Kalos et al, Displacement Mapping on the GPU-State of the Art, Computer Graphics Forum Volume 27, Issue 6 September 2008 Pages 1567-1592 [0017]

Patentansprüche

1. Vertex-Verschiebungs-Codierungsverfahren, das aufweist:
 - (a) Codieren von Teildreiecken in aufsteigender Reihenfolge der Tesselierungsrate;
 - (b) Codieren von Teildreiecken mit der gleichen Tesselierungsrate in absteigender Reihenfolge der Kompressionsrate; und
 - (c) für einen bestimmten Mikronetztyp, Verwenden mindestens derselben Anzahl von Bits für eine Unterteilungsstufe wie für die nächste, stärker unterteilte Stufe.
2. Nichtflüchtiger Speicher, der konfiguriert ist, einen Vertex-Verschiebungsdatenblock zu speichern, der aufweist:
 - einen ersten UNORM-Verschiebungswert für einen ersten Basisdreieck-Vertex;
 - einen zweiten UNORM-Verschiebungswert für einen zweiten Basisdreieck-Vertex;
 - einen dritten UNORM-Verschiebungswert für einen dritten Basisdreieck-Vertex; und
 - Korrekturwerte, die konfiguriert sind, Vertex-Verschiebungswerte zu korrigieren, die zumindest teilweise auf den ersten, zweiten und dritten UNORM-Verschiebungswerten basieren.
3. Speicher nach Anspruch 2, wobei der Vertex-Verschiebungsdatenblock ferner Verschiebungswerte aufweist, die konfiguriert sind, die Korrekturwerte anzupassen.
4. Kompressor, der aufweist:
 - ein Prädiktor, der eine Mittelwertbildung verwendet, um Vertex-Verschiebungswerte vorzusagen;
 - einen Komparator, der die vorhergesagten Vertex-Verschiebungswerte mit vorgegebenen Vertex-Verschiebungswerten vergleicht und Korrekturen bestimmt; und
 - einen Qualitätstester, der die durch die Qualitätskorrekturen korrigierten vorausgesagten Vertex-Verschiebungswerte prüft.
5. Kompressor nach Anspruch 4, bei dem der Komparator die Wrap-Around-Arithmetik nutzt, um die Qualität zu erhöhen.
6. Kompressor nach Anspruch 4, wobei der Komparator Verschiebungswerte zur Anwendung auf die bestimmten Korrekturen berechnet und wiederhergestellte dekomprimierte Vertex-Verschiebungswerte anstelle von Mikrovertex-Eingangsverschiebungswerten zur Verwendung bei der Vorhersage von Vertex-Verschiebungswerten für Polygonkanten ersetzt, die von Teildreiecken mit unterschiedlichen Tesselierungsraten gemeinsam genutzt werden.
7. Kompressor nach Anspruch 4, wobei der Kompressor konfiguriert ist, Teildreiecke in aufsteigender Reihenfolge der Tesselierungsrate zu codieren, Teildreiecke mit der gleichen Tesselierungsrate in absteigender Reihenfolge der Kompressionsrate zu codieren und für einen gegebenen Mikronetztyp mindestens die gleiche Anzahl von Bits für eine Unterteilungsstufe zu verwenden wie für die nächste, stärker unterteilte Stufe.
8. Dekompressor, der aufweist:
 - einen Prädiktor, der basierend auf zuvor empfangenen und/oder berechneten Mikrovertex-Verschiebungswerten Mikrovertex-Verschiebungswerte für ein Mikronetz vorhersagt;
 - einen Korrektor, der die vorhergesagten Mikrovertex-Verschiebungswerte basierend auf den empfangenen Korrekturen korrigiert.
9. Dekompressor nach Anspruch 8, der ferner ein Schieberegister aufweist, das die empfangenen Korrekturen um einen Verschiebungsbetrag verschiebt, der durch einen empfangenen Verschiebungswert spezifiziert wird.
10. Dekompressor nach Anspruch 8, der ferner einen rekursiven Unterteiler aufweist, der das Mikronetz rekursiv unterteilt und aussortiert.
11. Grafiksystem, das aufweist:
 - eine Speicherschnittstellenschaltung, die Mikrovertex-Verschiebungsblöcke in Cacheline-Größe aus einem Cache-Speicher empfängt;
 - einen Dekompressor, der Verschiebungswerte zumindest teilweise basierend auf dem Inhalt der Mikrovertex-Verschiebungsblöcke vorhersagt und die vorhergesagten Verschiebungswerte korrigiert; und

eine Grafikpipeline, die ein Bild zumindest teilweise basierend auf den korrigierten, vorhergesagten Verschiebungswerten rendert.

12. Grafiksystem nach Anspruch 11, wobei die Verschiebungsblöcke Teil einer Beschleunigungsstruktur sind und die Grafikpipeline einen Raytracer aufweist.

13. Graphiksystem nach Anspruch 12, wobei der Raytracer eine Strahlenschnitttestschaltung aufweist, die die korrigierten vorhergesagten Verschiebungswerte vom Dekompressor empfängt.

14. Grafiksystem nach Anspruch 13, wobei der Dekompressor ein Schieberegister aufweist, das die empfangenen Korrekturwerte basierend auf Verschiebungsbeträgen verschiebt, die die Mikrovertex-Verschiebungsblöcke spezifizieren.

15. Grafiksystem nach Anspruch 11, wobei die Grafikpipeline einen Dreieckstexturmapper enthält.

16. Graphiksystem nach Anspruch 11, wobei der Dekompressor Bit-für-Bit-Übereinstimmungen der korrigierten Verschiebungswerte für Vertices unterschiedlicher Teildreiecke bereitstellt, die zu unterschiedlichen Zeiten dekomprimiert wurden.

17. Graphikverarbeitungsverfahren, das aufweist:
Empfangen von Mikrovertex-Verschiebungsblöcken aus einem Cache-Speicher, wobei die Blöcke so bemessen sind, dass sie in eine Cacheline passen;
Vorhersagen von Mikrovertex-Verschiebungen, zumindest teilweise auf dem Inhalt der Mikrovertex-Verschiebungsblöcke basierend;
Korrigieren der vorhergesagten Mikrovertex-Verschiebungen basierend auf Korrekturfaktoren, die von den Blöcken spezifiziert werden; und
Rendern eines Bildes, zumindest teilweise auf den korrigierten, vorhergesagten Mikrovertex-Verschiebungen basierend.

18. Graphikverarbeitungsverfahren nach Anspruch 17, wobei das Rendern das Rendern eines rissfreien Bildes aufweist.

19. Graphikverarbeitungsverfahren nach Anspruch 17, wobei das Rendern das Testen eines Strahls auf einen Schnitt mit einer Geometrie aufweist, die durch die korrigierten, vorhergesagten Mikrovertex-Verschiebungen spezifiziert ist.

20. Graphikverarbeitungsverfahren nach Anspruch 17, das ferner das Verschieben der Korrekturfaktoren als Reaktion auf Verschiebungswerte aufweist, die die Blöcken spezifizieren.

21. Graphikverarbeitungsverfahren nach Anspruch 17, das ferner das Komprimieren eines Netzes aufweist, ohne dass eine Annahme darüber getroffen wird, ob das Netz vielfältig ist oder nicht.

Es folgen 36 Seiten Zeichnungen

Anhängende Zeichnungen

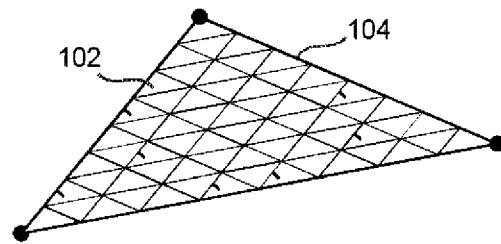


FIG. 1A

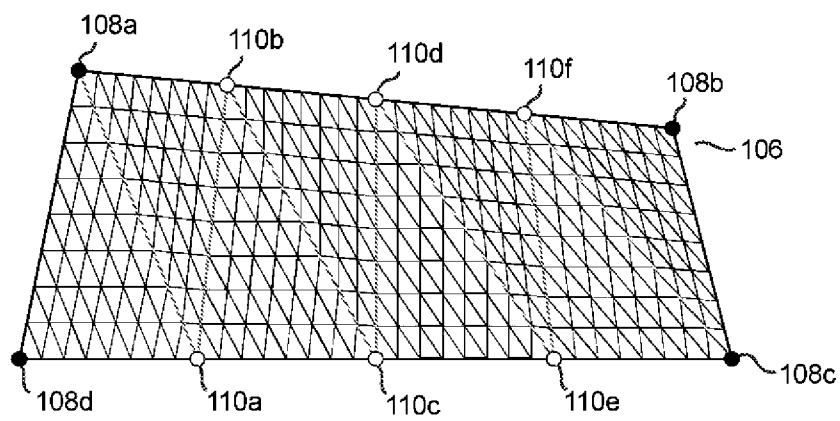


FIG. 1B

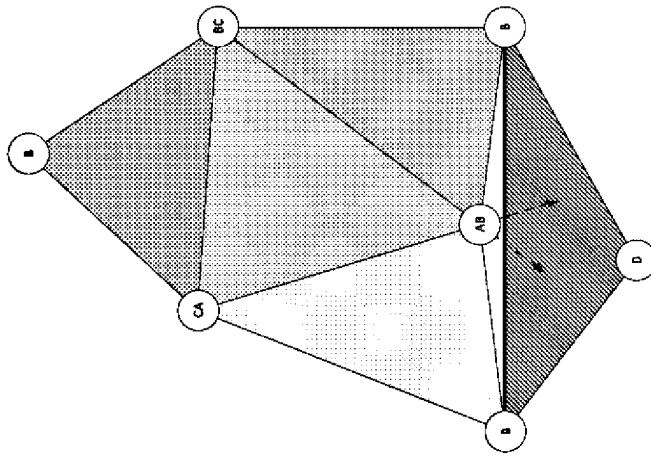


FIG. 2B

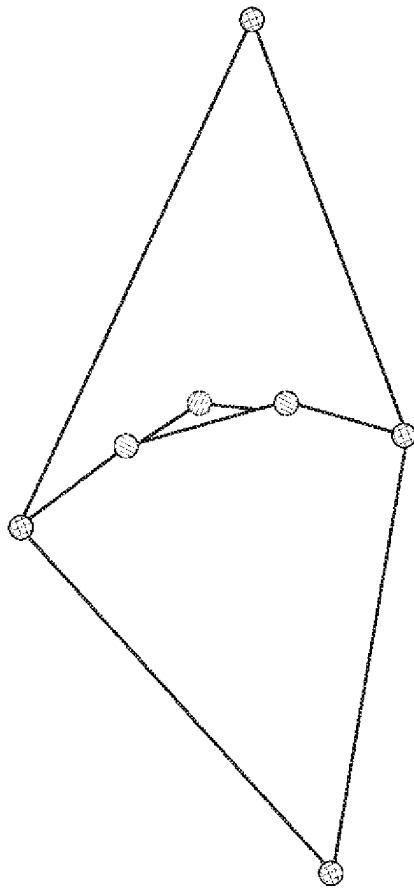


FIG. 2A

Basisdreieck			Mit maximalen Verschiebungsblöcken: Unkomprimiertes 8x8 64B Format				Mit minimalen Verschiebungsblöcken: Komprimiertes 32x32 64B Format				Explizite Dreiecksblöcke (Peak-Builder: 12 Dreiecke/Block)			
Stufe	Auflösung	Mikrodreiecke	Verschiebungsblöcke	Dreiecksblöcke	Speicherbedarf (Verschiebungssatz + Dreiecksblöcke)	Bits/Mikrodreiecke	Verschiebungsblöcke	Dreiecksblöcke	Speicherbedarf (Verschiebungssatz + Dreiecksblöcke)	Bits/Mikrodreiecke	Dreiecksblöcke	Speicherbedarf	Bits/Dreieck	
0	1 x 1	1	1	1	64 B + 128 B	1,536.0	1	1	64 B + 128 B	1,536.0	1	128 B	1024	
1	2 x 2	4	1	1	64 B + 128 B	384.0	1	1	64 B + 128 B	384.0	1	128 B	256	
2	4 x 4	16	1	1	64 B + 128 B	96.0	1	1	64 B + 128 B	96.0	2	256 B	128	
3	8 x 8	64	1	1	64 B + 128 B	24.0	1	1	64 B + 128 B	24.0	6	768 B	96	
4	16 x 16	256	4	1	256 B + 128 B	12.0	1	1	64 B + 128 B	6.0	22	3 KB	88	
5	32 x 32	1,024	16	1	1 KB + 128 B	9.0	1	1	64 B + 128 B	1.5	86	11 KB	86	
6	64 x 64	4,096	64	1	4 KB + 128 B	8.3	4	1	256 B + 128 B	0.8	342	43 KB	85	
7	128 x 128	16,384	256	4	16 KB + 512 B	8.3	16	1	1 KB + 128 B	0.6	1,366	171 KB	85	
8	256 x 256	65,536	1,024	16	64 KB + 2 KB	8.3	64	1	4 KB + 128 B	0.5	5,462	683 KB	85	
9	512 x 512	262,144	4,096	64	256 KB + 8 KB	8.3	256	4	16 KB + 512 B	0.5	21,846	3 MB	85	
10	1024 x 1024	1,048,576	16,384	256	1 MB + 32 KB	8.3	1,024	16	64 KB + 2 KB	0.5	87,382	11 MB	85	
11	2048 x 2048	4,194,304	65,536	1,024	4 MB + 128 KB	8.3	4,096	64	256 KB + 8 KB	0.5	349,526	43 MB	85	
12	4096 x 4096	16,777,216	262,144	4,096	16 MB + 512 KB	8.3	16,384	256	1 MB + 32 KB	0.5	1,398,102	171 MB	85	
13	8192 x 8192	67,108,864	1,048,576	16,384	64 MB + 2 MB	8.3	65,536	1,024	4 MB + 128 KB	0.5	5,592,406	683 MB	85	

FIG. 3

Stufe: n	Auflösung: $2^n \times 2^n$	Mikrodreiecke: 4^n	μ -Dreieck- Indexbits $\log_2(4^n)$	Stufen-Codierungsbits: $\log_2(n+1)$
0	1 x 1	1	0	0
1	2 x 2	4	2	1
2	4 x 4	16	4	2
3	8 x 8	64	6	2
4	16 x 16	256	8	3
5	32 x 32	1,024	10	3
6	64 x 64	4,096	12	3
7	128 x 128	16,384	14	3
8	256 x 256	65,536	16	4
9	512 x 512	262,144	18	4
10	1024 x 1024	1,048,576	20	4
11	2048 x 2048	4,194,304	22	4
12	4096 x 4096	16,777,216	24	4
13	8192 x 8192	67,108,864	26	4
14	16384 x 16384	268,435,456	28	4
15	32768 x 32768	1,073,741,824	30	4
16	65536 x 65536	4,294,967,296	32	5

FIG. 4

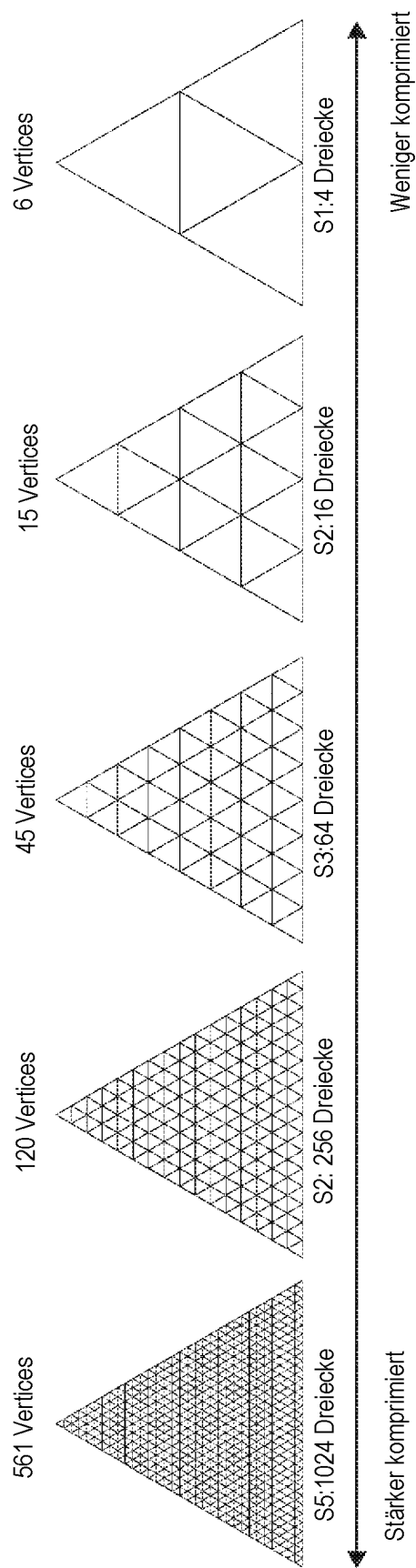


FIG. 5

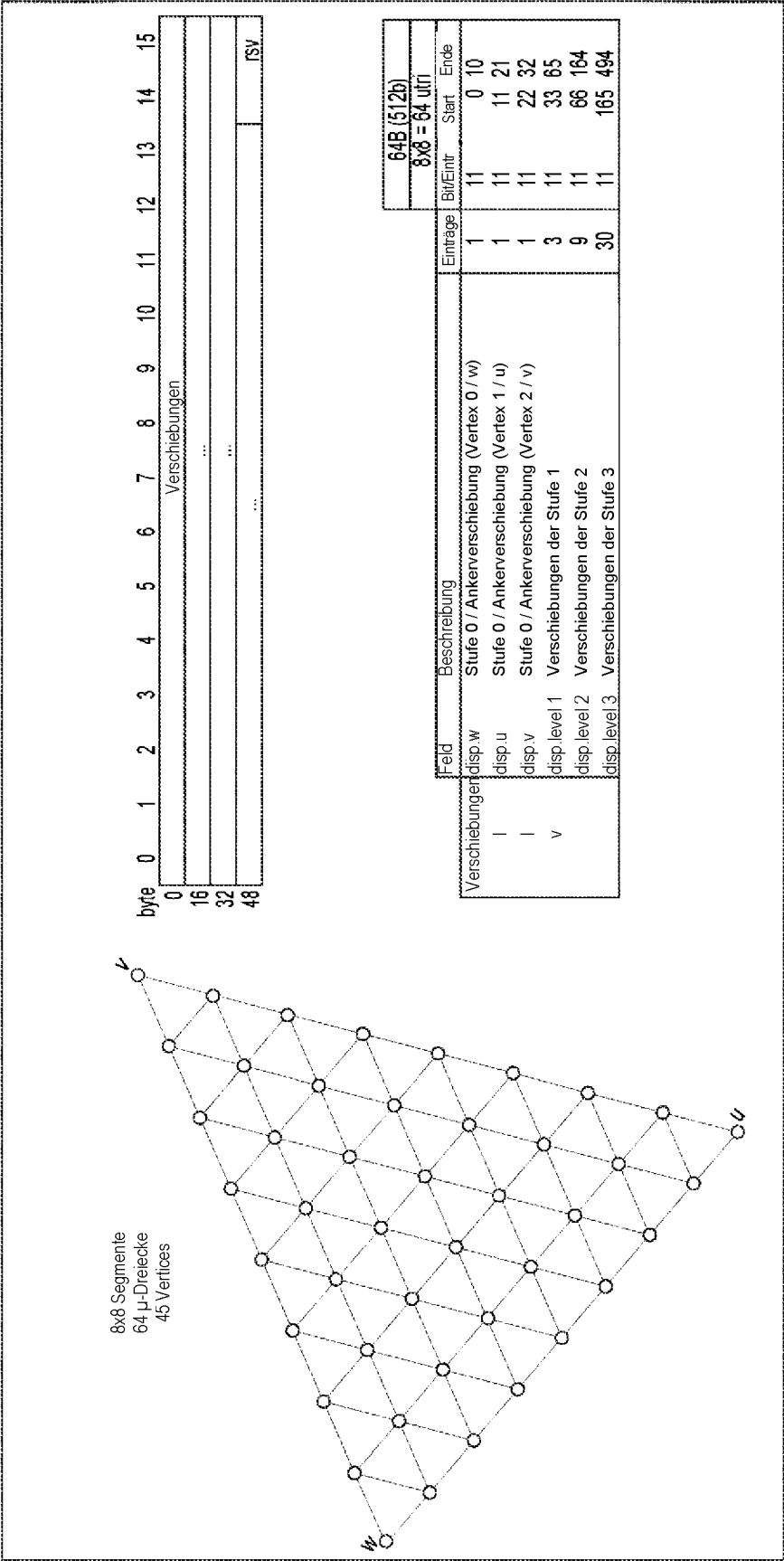


FIG. 6
Unkomprimierter Block – Flache Codierung - (64B)

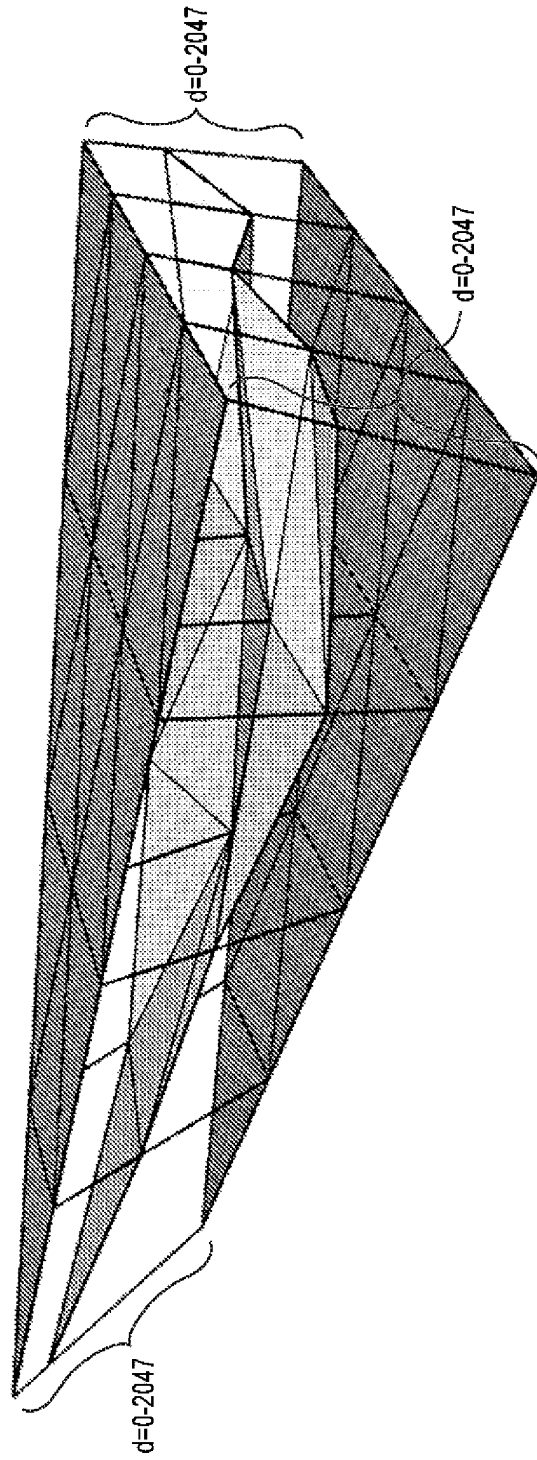


FIG. 7

Konvexes Prismatoid-Hüllenmodell für verschobenes Mikronetz

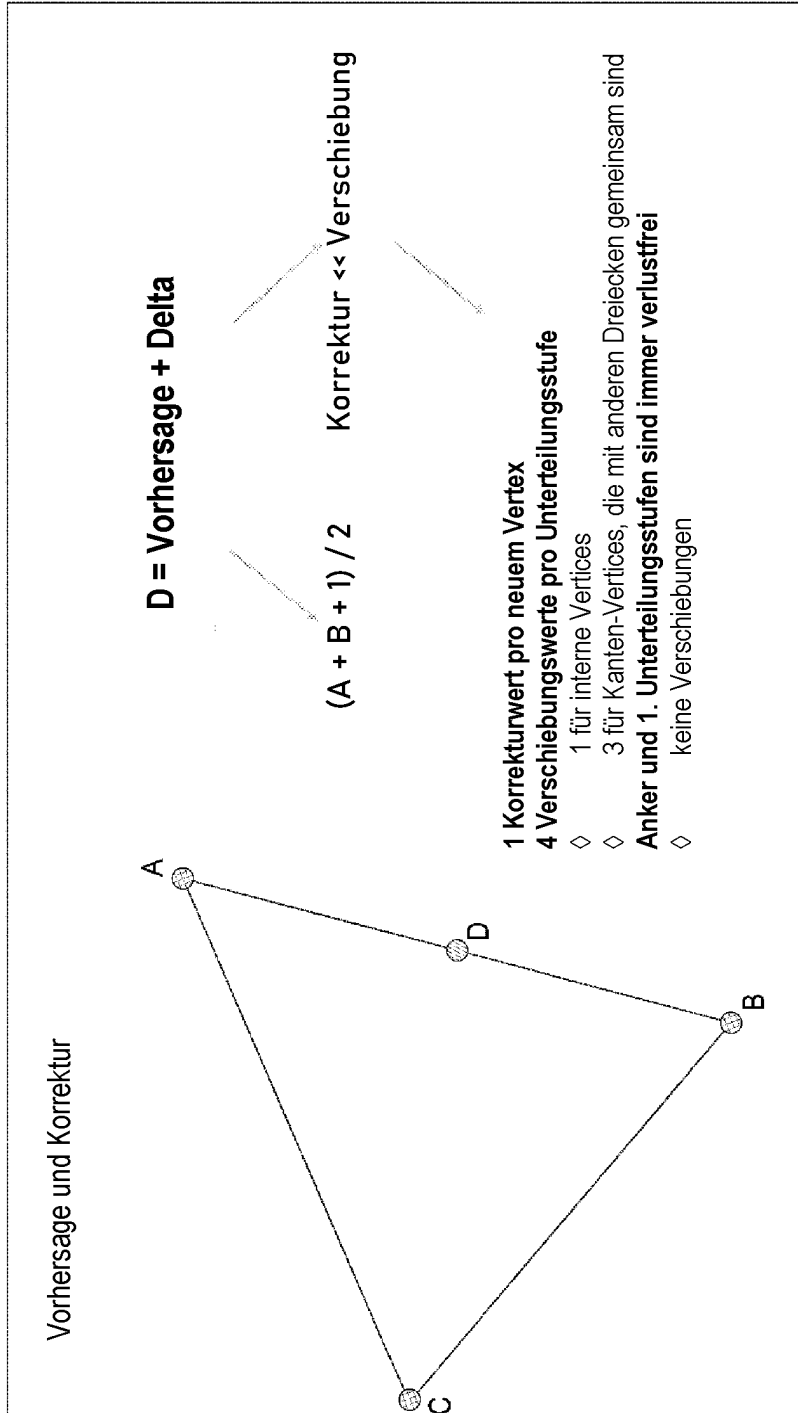


FIG. 8

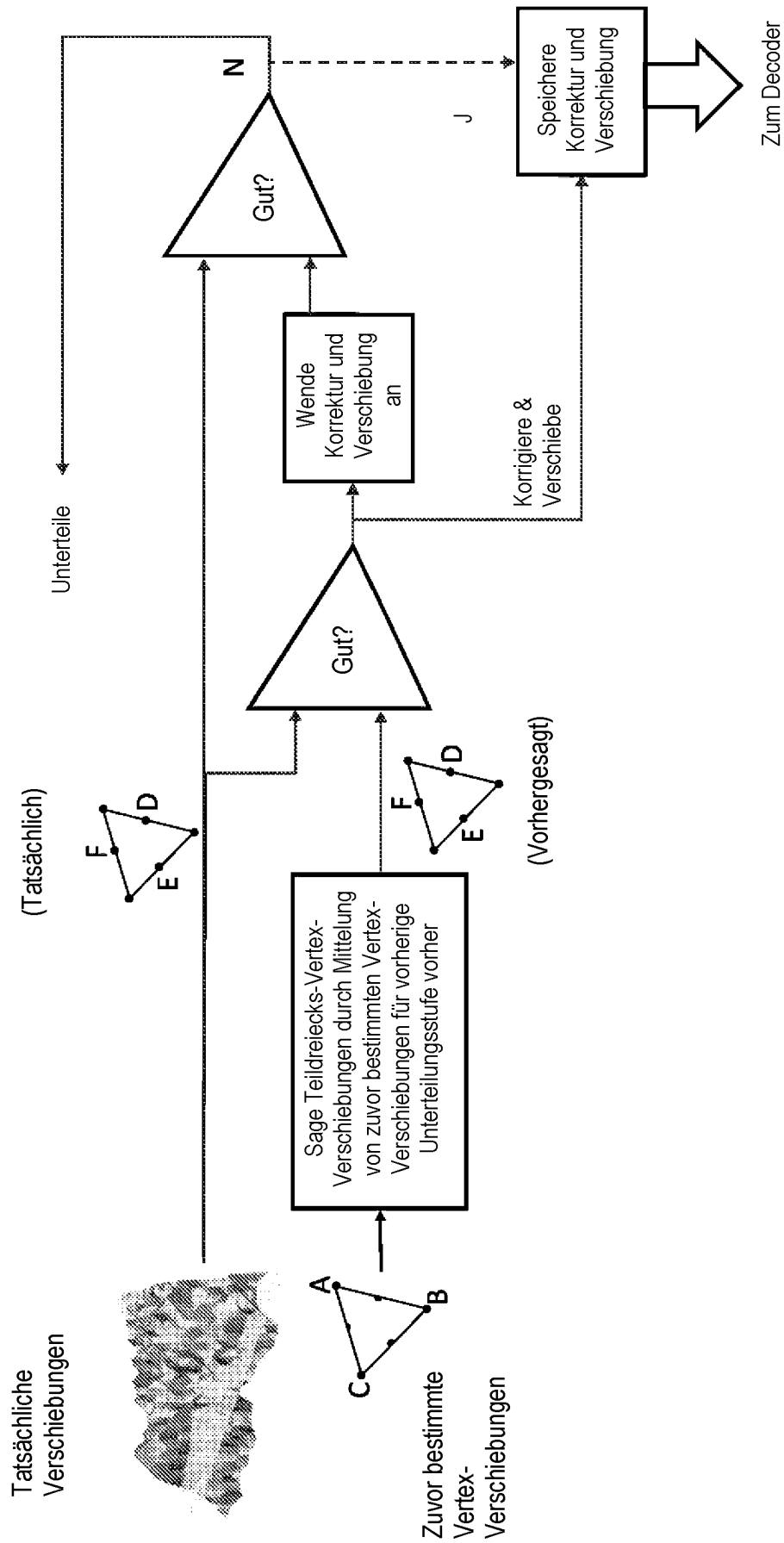


FIG. 9
Codierer

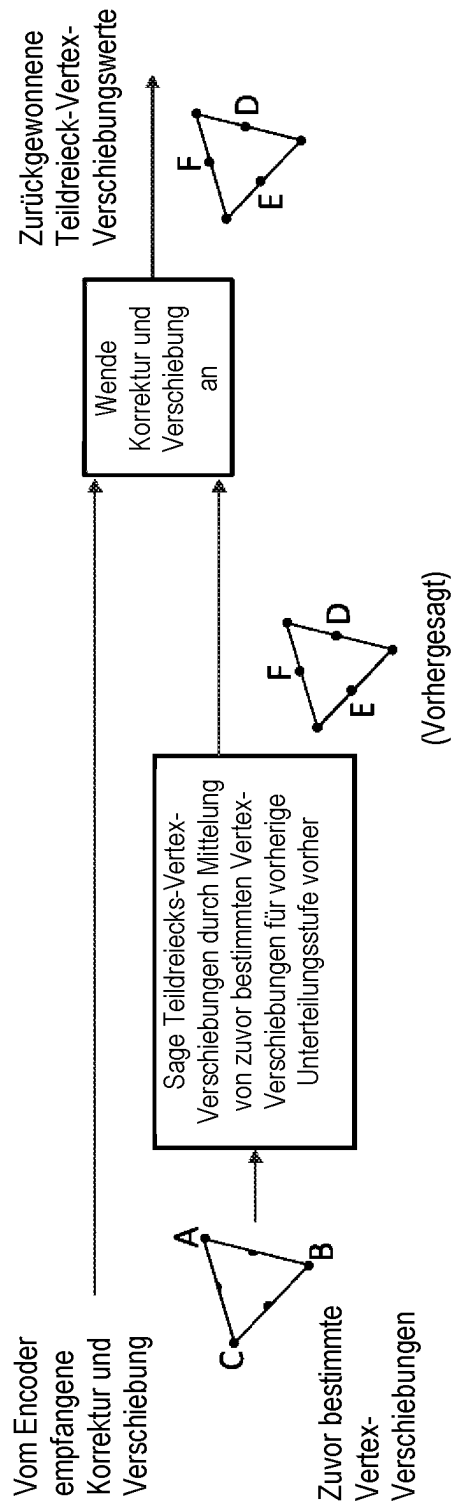


FIG. 10
Decodierungsprozess

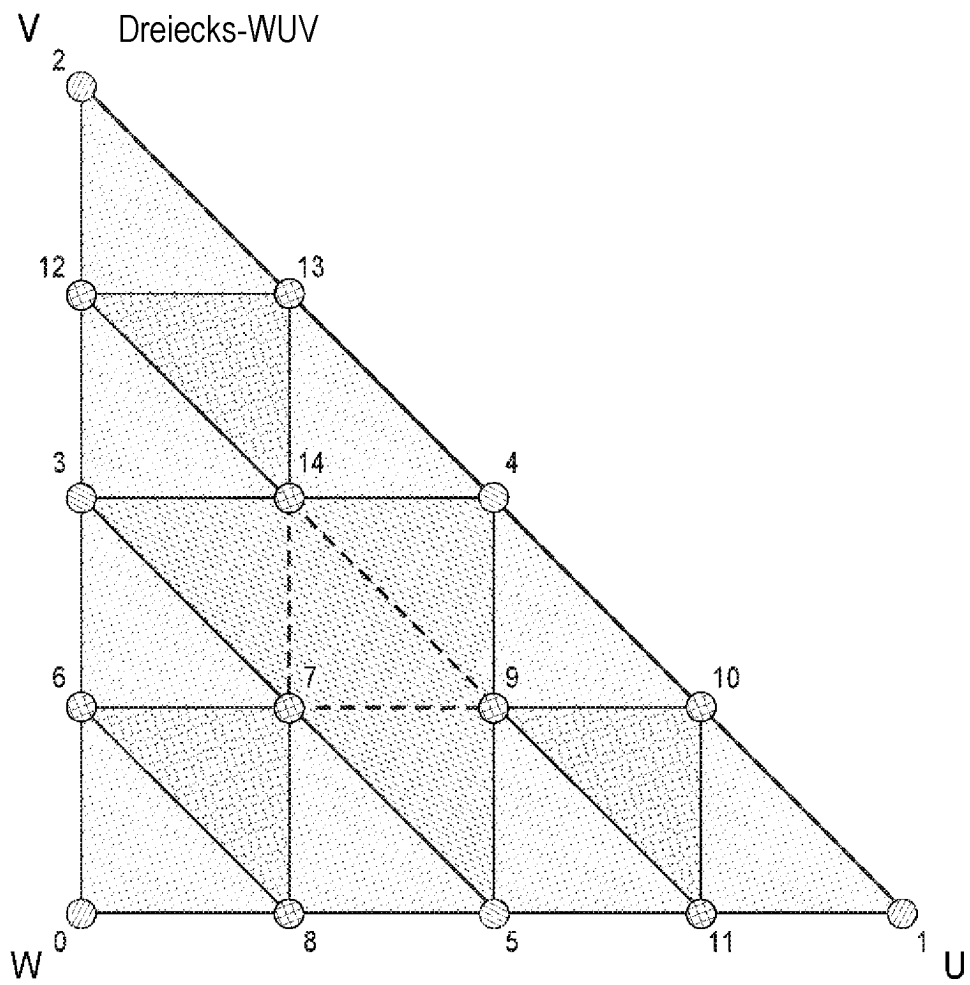


FIG. 10A

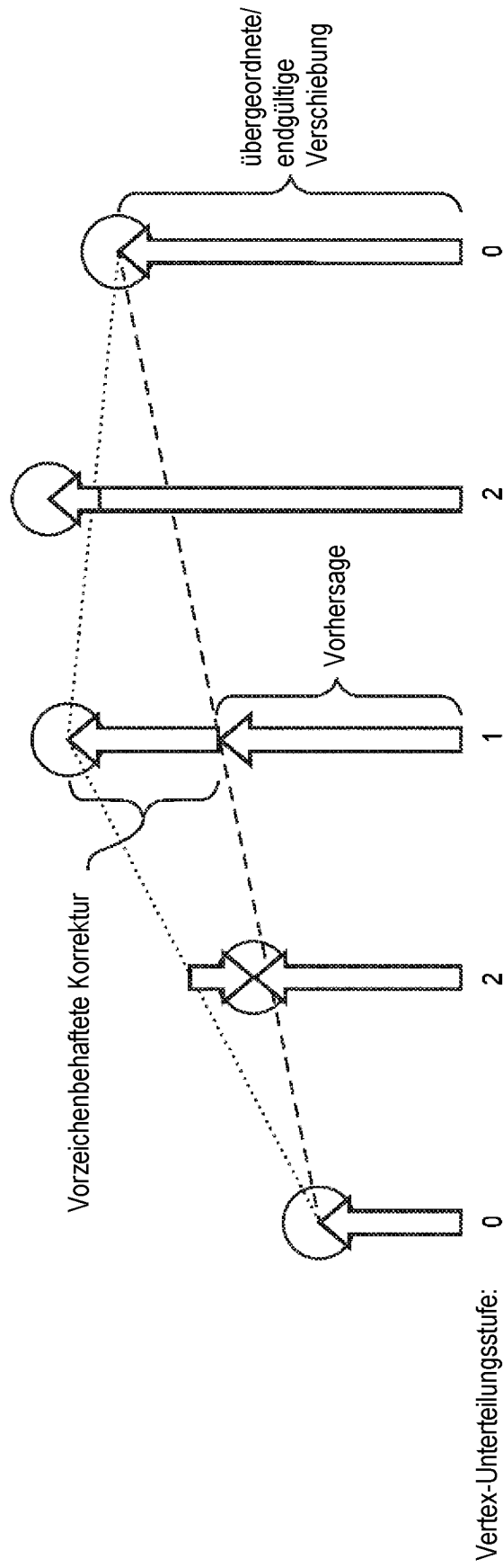


FIG. 11

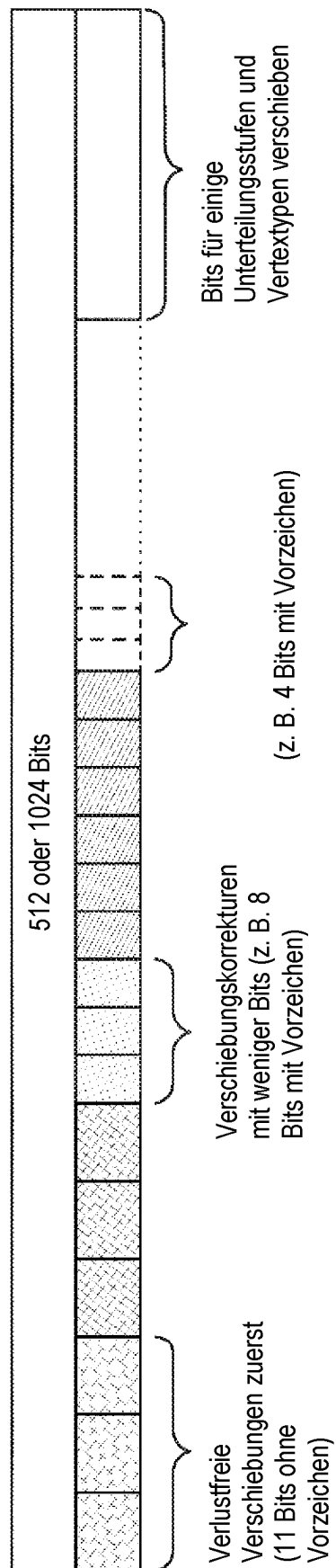


FIG. 12

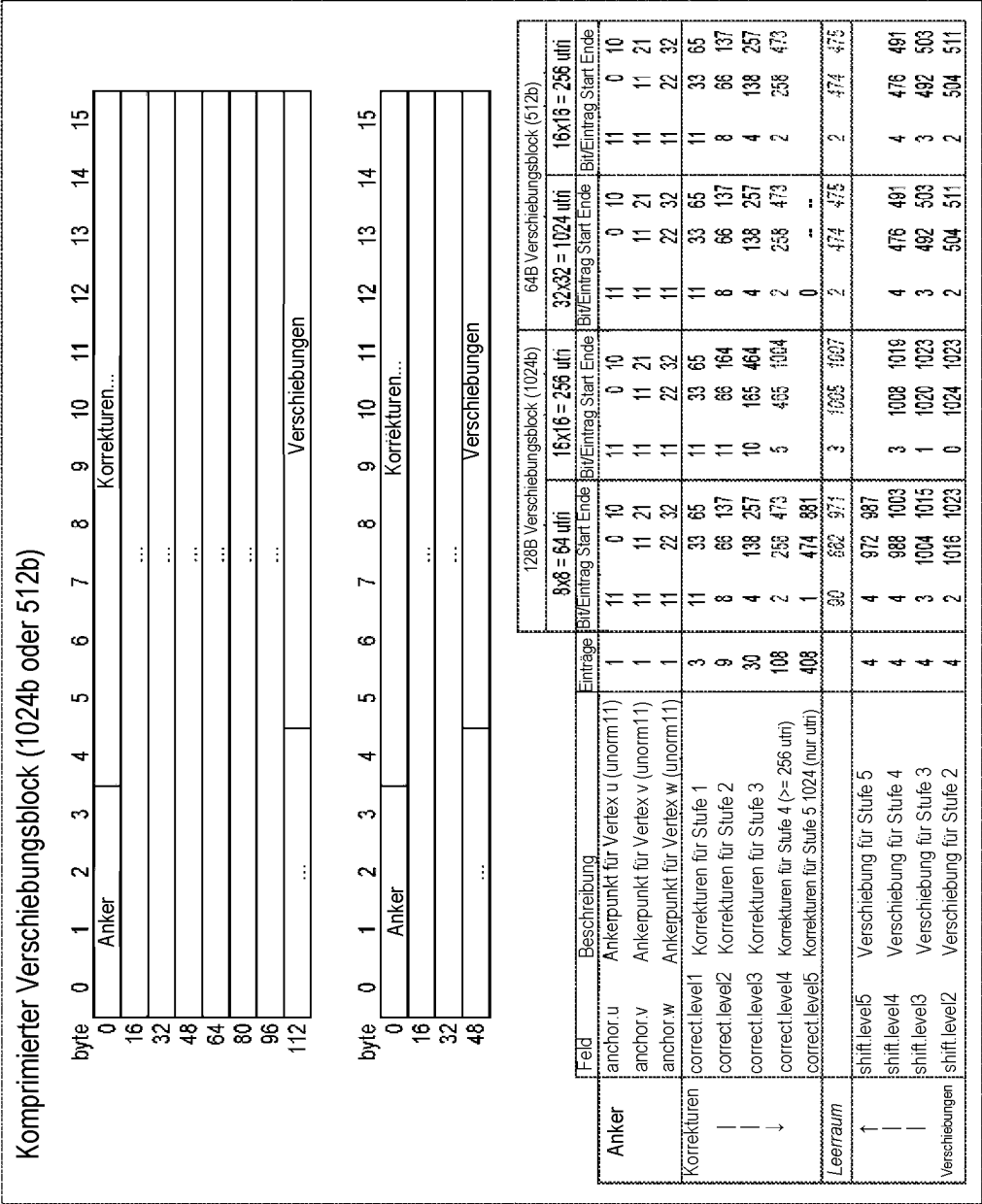


FIG. 13

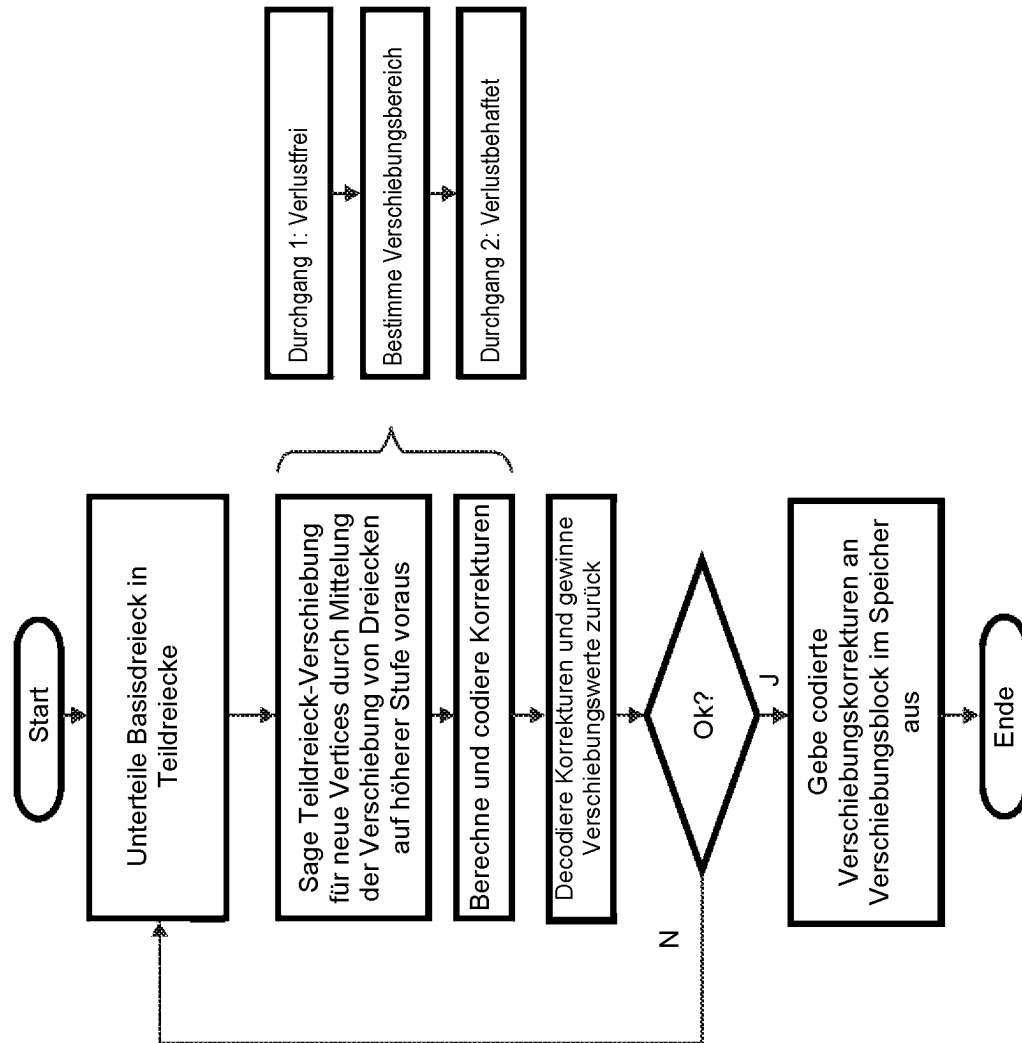


FIG. 14

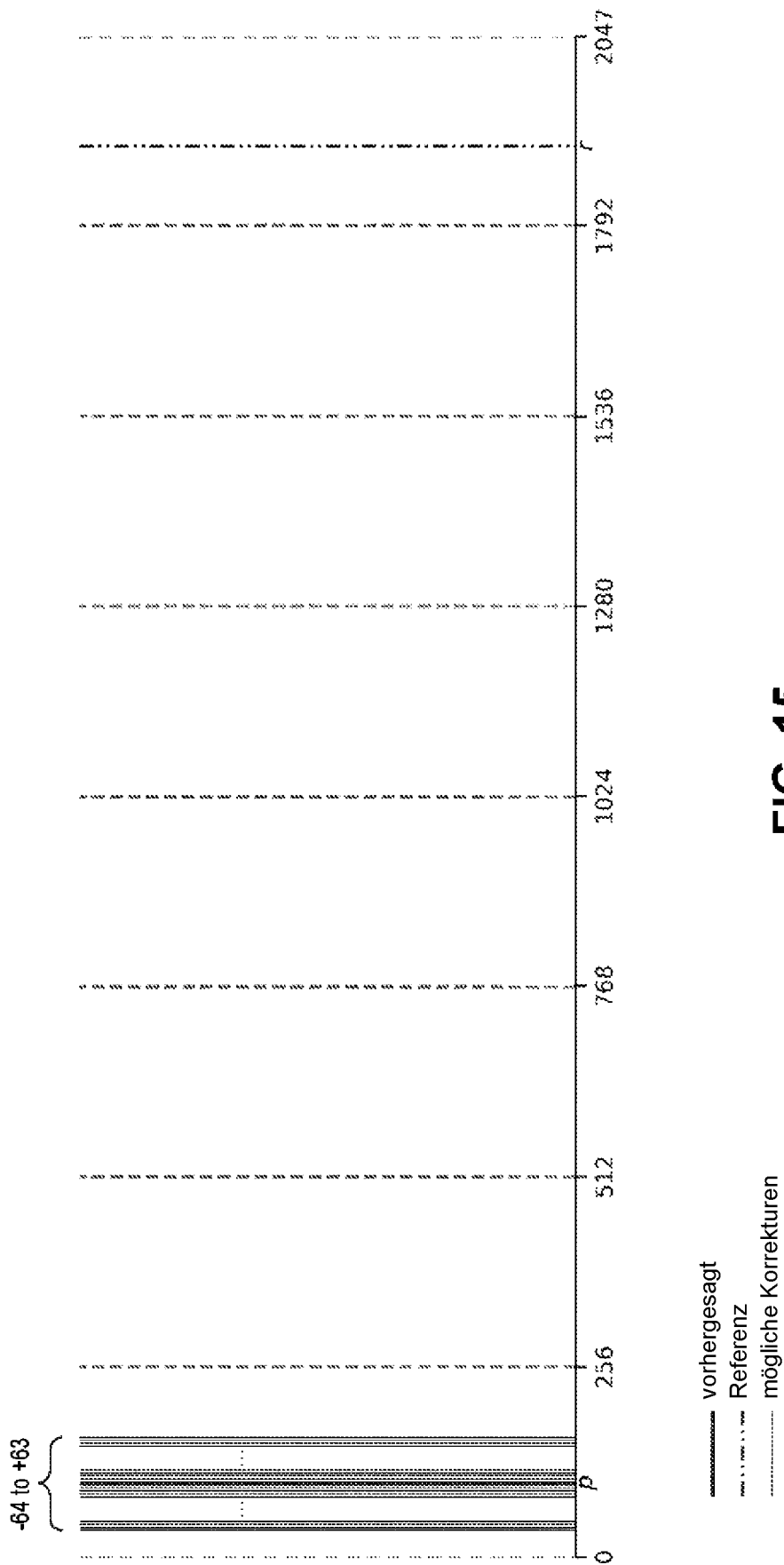
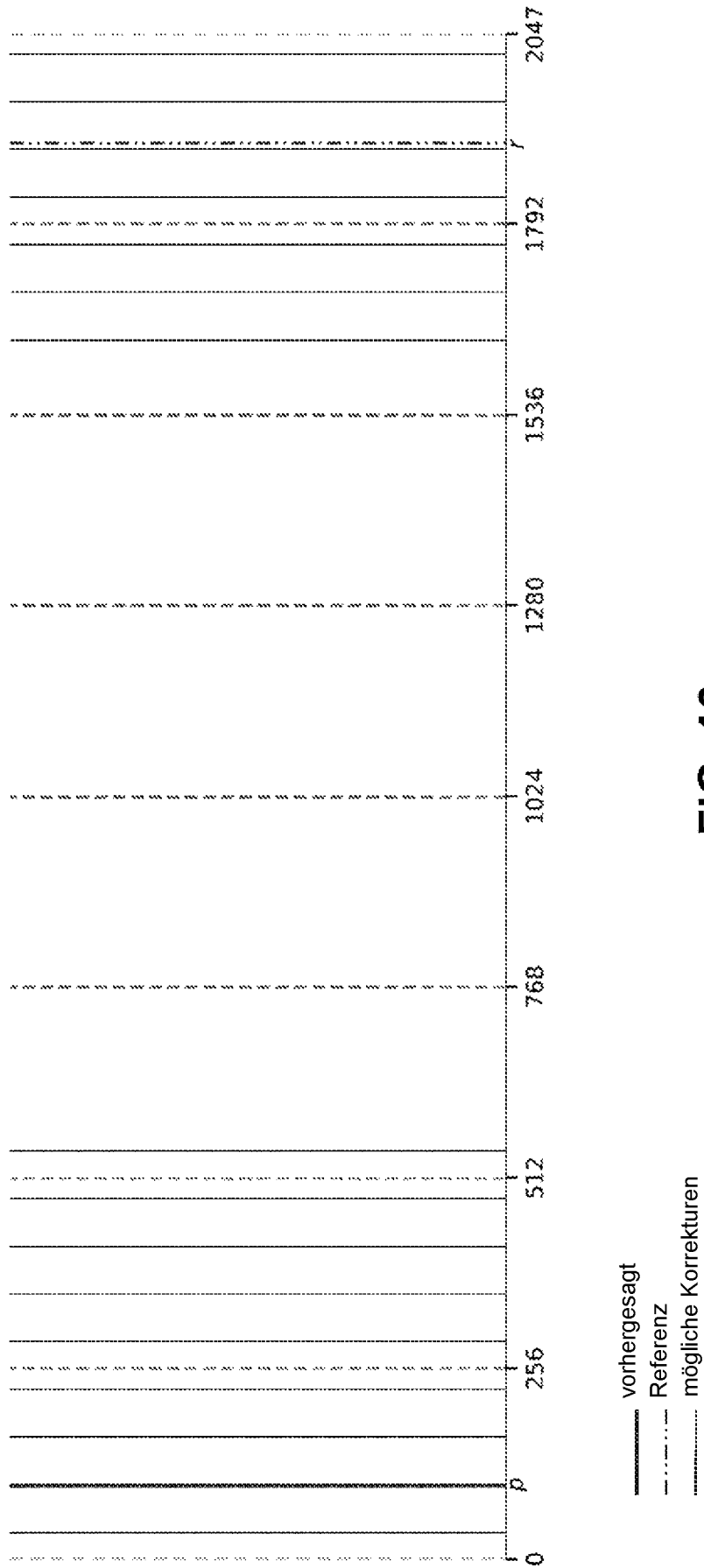


FIG. 15

Zahlenbereich, der den Korrekturproblemfall $p=100$, $r=1900$, $s=0$, $b=7$ darstellt



Zahlenbereich, der den Korrekturproblemfall $p=100$, $r=1900$, $s=6$, $b=3$ darstellt

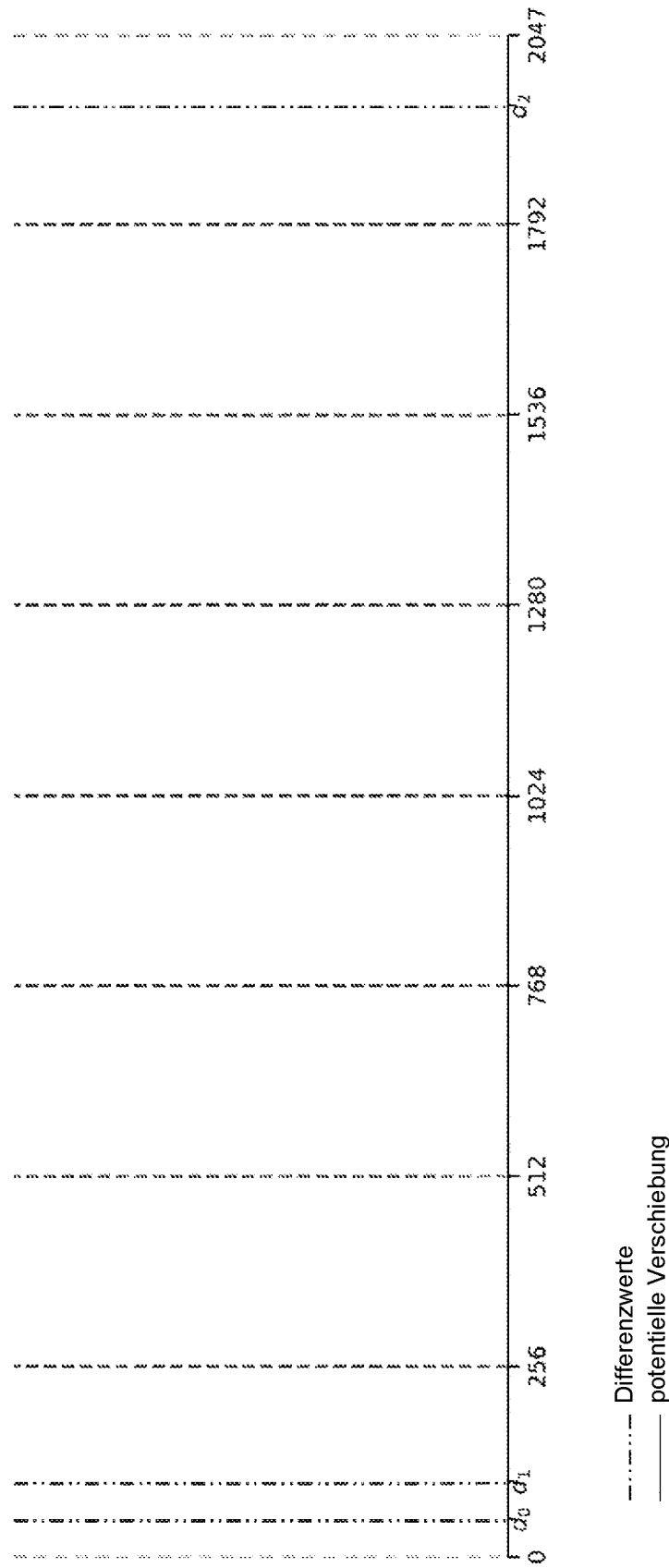


FIG. 17

Eine Beispielveilteilung der Differenzen zwischen Referenz- und vorhergesagten Werten mod 2048 für ein bestimmtes Teildreieck, eine Korrekturstufe und einen Vertextyp. Da sich die Differenzen um 0 und 2047 ansammeln, kann die Verwendung des Wrap-Around-Verhaltens hier eine höhere Präzision ermöglichen.

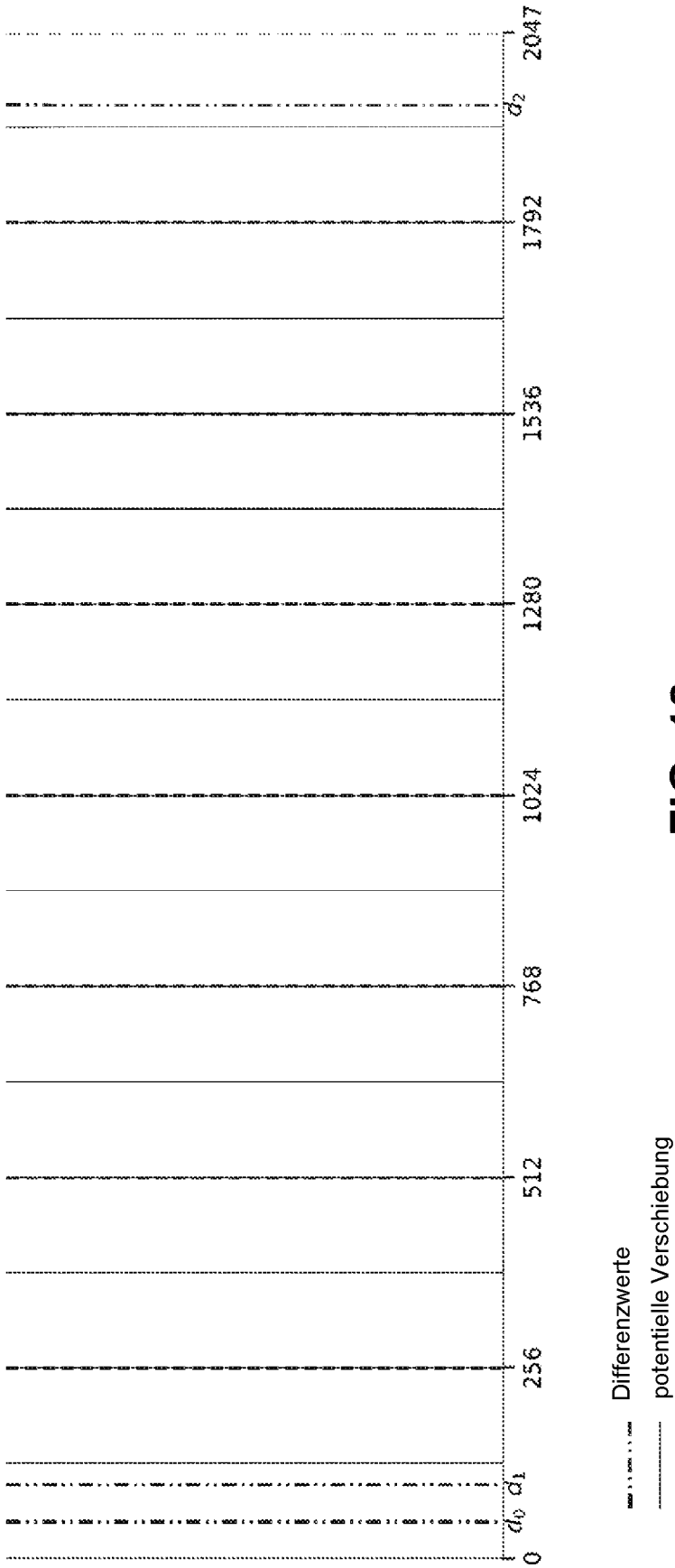


FIG. 18

Auswählen von Verschiebungen, die den gesamten Bereich möglicher Differenzen abdecken, ohne Wrapping zu berücksichtigen, kann zu großen Fehlern führen (Abstand zwischen jeder roten Linie und der ihr am nächsten liegenden grünen Linie).

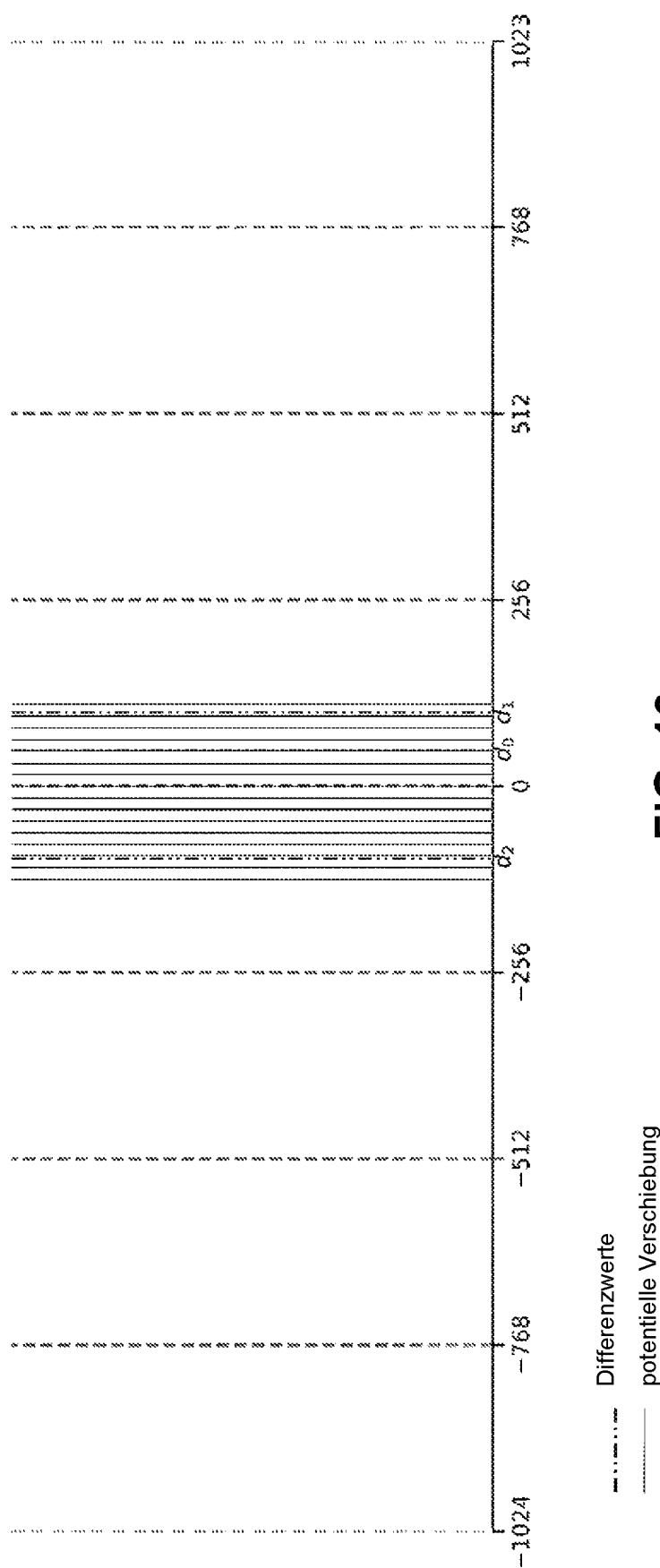


FIG. 19

Auf den Bereich -1024...1023 umgebrochene Verschiebungsdifferenzen. Die umgebrochenen Werte sammeln sich um 0, und ihr Minimum und Maximum ergeben Verschiebungen, so dass die darstellbaren Differenzen aus verschobenen Korrekturen (grün) näher an den Differenzen liegen, wenn man die Abstandsmetrik von $Z/2^{11}Z$ verwendet (wobei zu beachten ist, dass dies nicht die Metrik ist, die letztendlich für die Korrektur verwendet wird), was die Qualität verbessert.

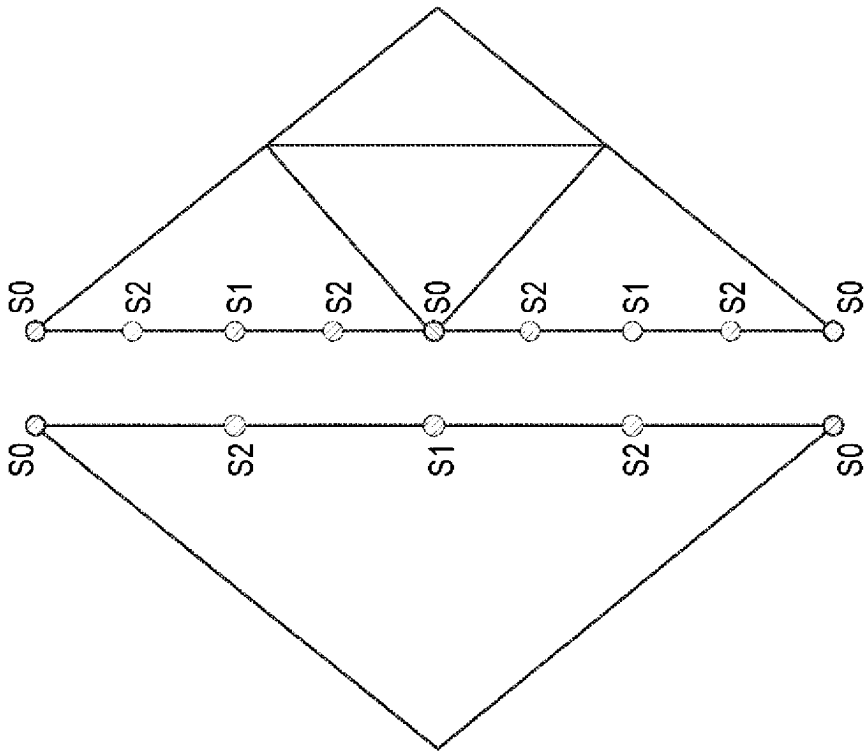


FIG. 20B

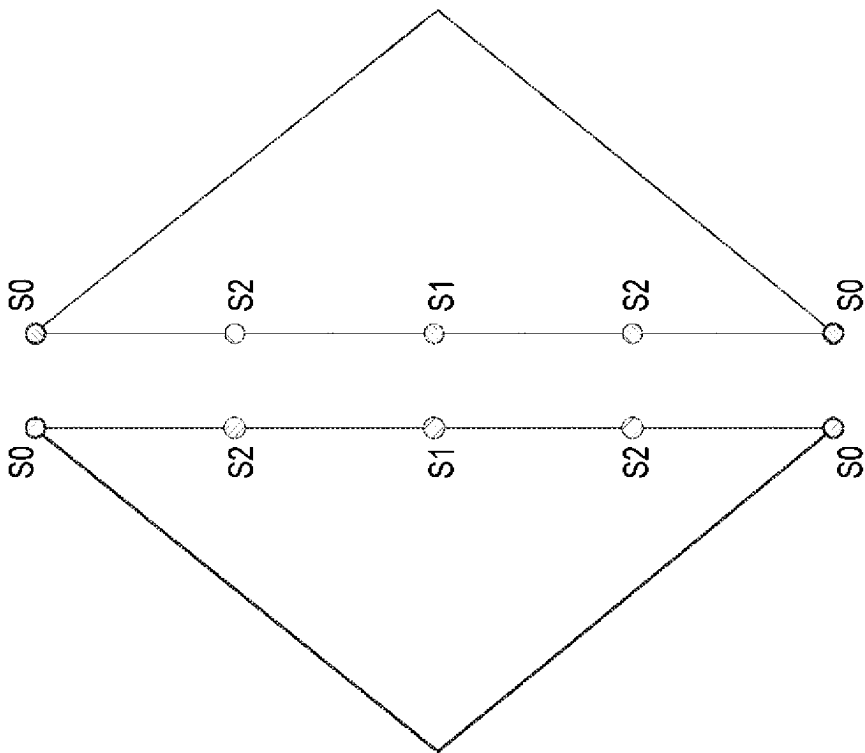


FIG. 20A

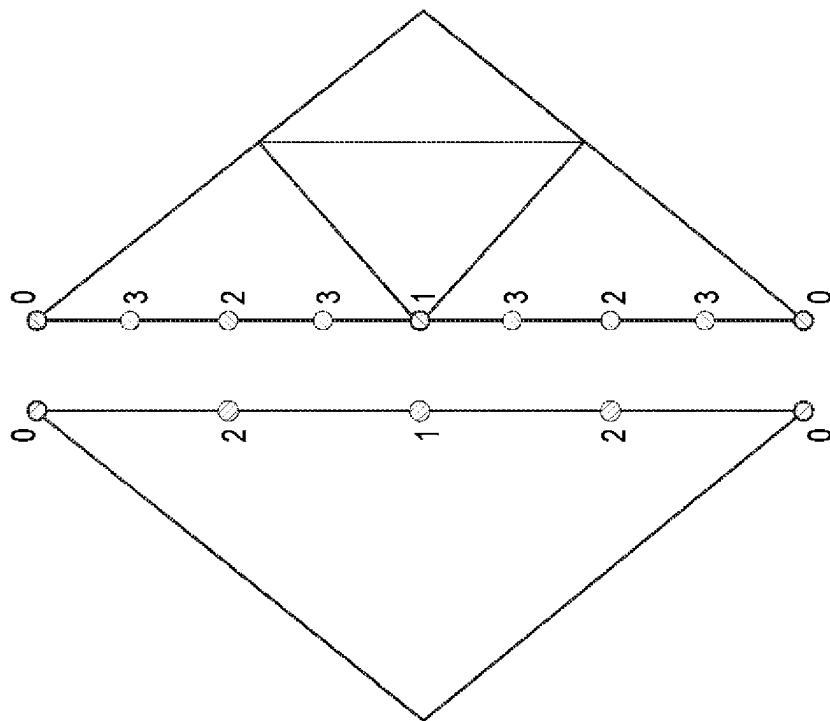


FIG. 20C

foreach Basisdreieck Auflösung (von niedriger zu höherer Auflösung):
foreach Mikronetztyp (von am meisten zu am wenigsten komprimiert):
 foreach nicht codiertes Dreieck:
 codiere Dreieck
 if erfolgreich **then** markiere Dreieck als codiert
 foreach teilweise codierte Kante:
 aktualisiere Referenzverschiebungen

FIG. 21

foreach Mikronetztyp:
foreach nicht codiertes Dreieck:
 codiere Dreieck
 if erfolgreich **then** markiere Dreieck als codiert
foreach teilweise codierte Kante:
 aktualisiere Referenzverschiebungen

FIG. 21A

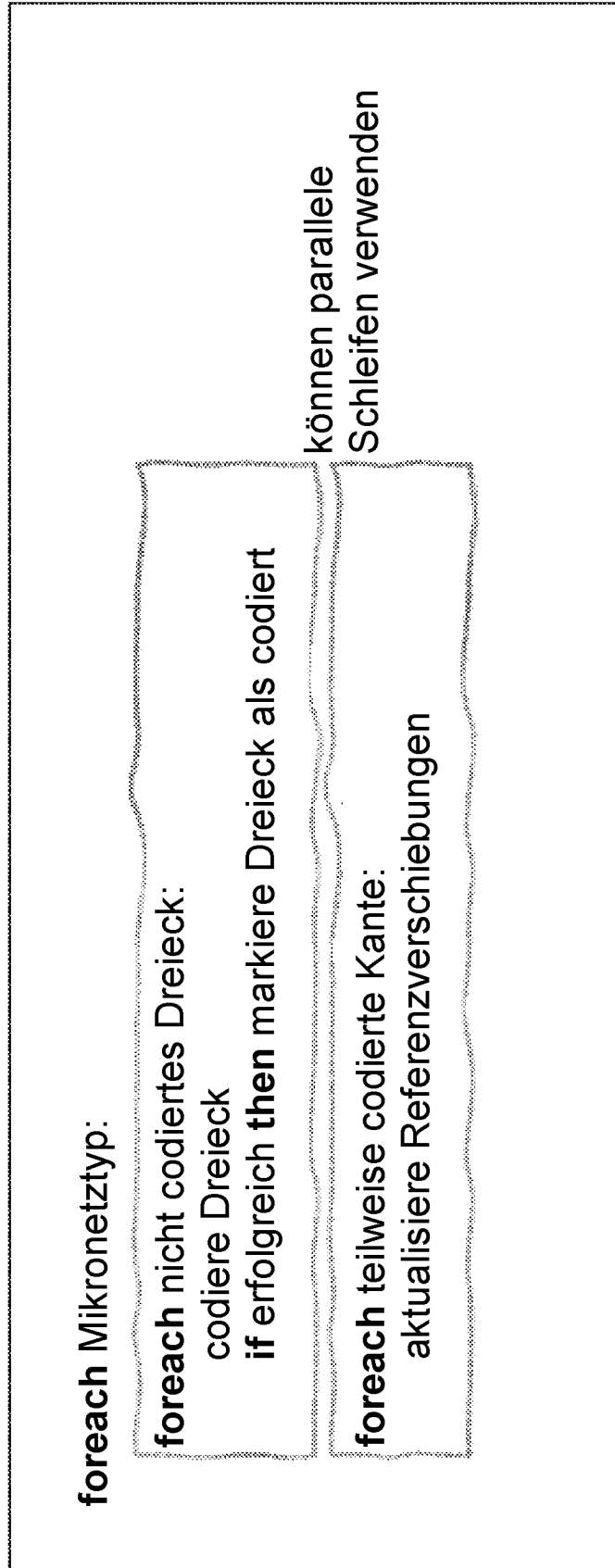


FIG. 21B

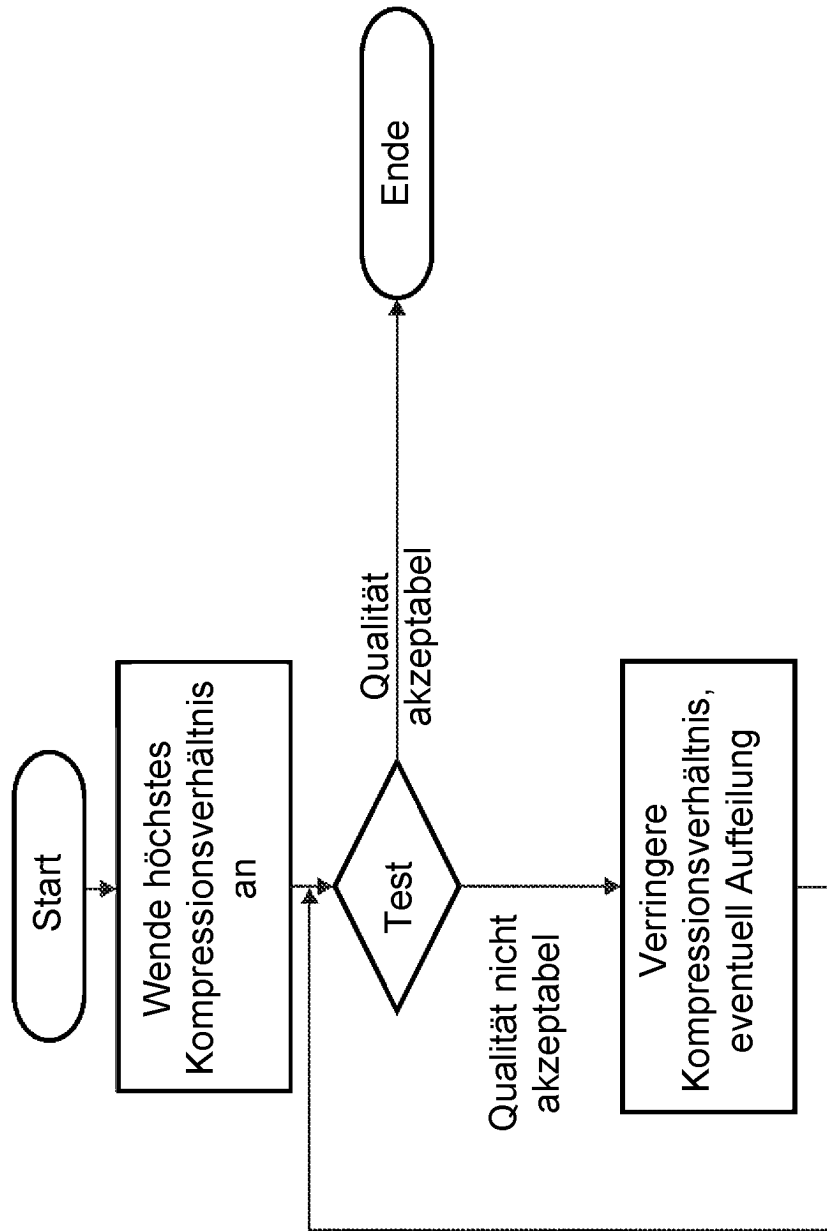


FIG. 22

- 1. Typ: 1024t
- 2. Typ: 256t
- 3. Typ: 64t

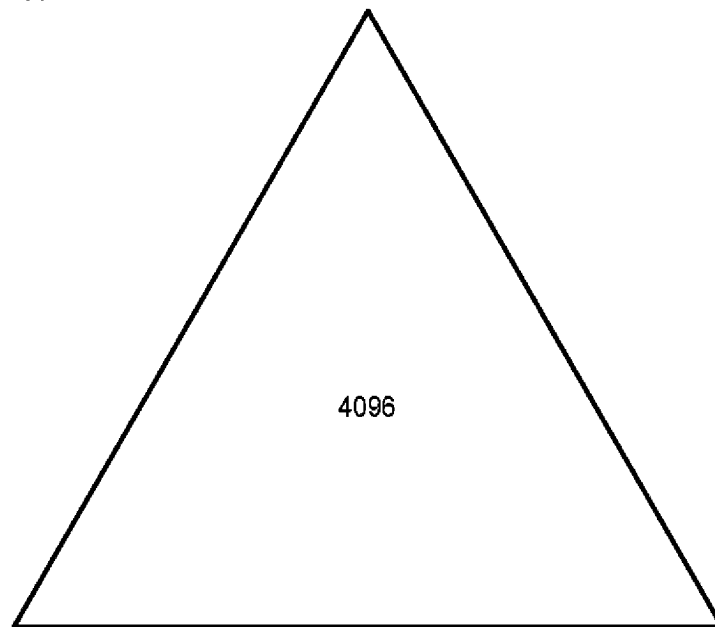


FIG. 23A

1. Typ: 1024t
2. Typ: 256t
3. Typ: 64t

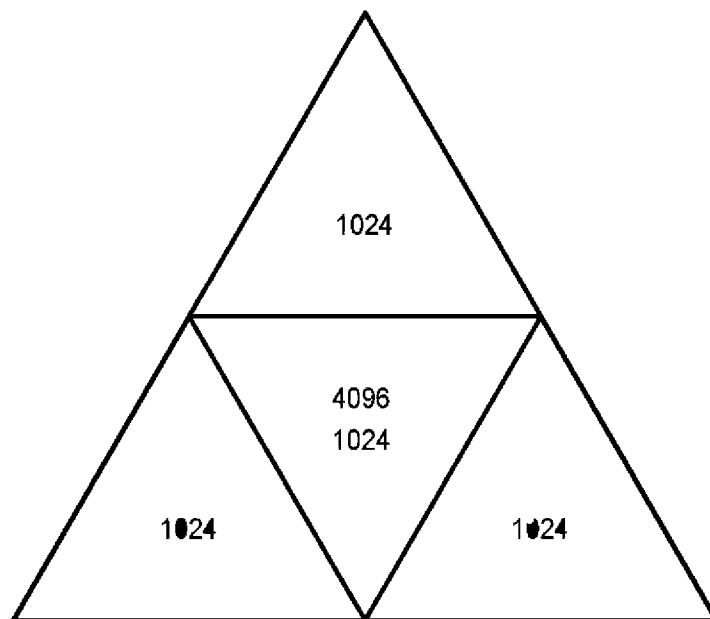


FIG. 23B

- 1. Typ: 1024t
- 2. Typ: 256t
- 3. Typ: 64t

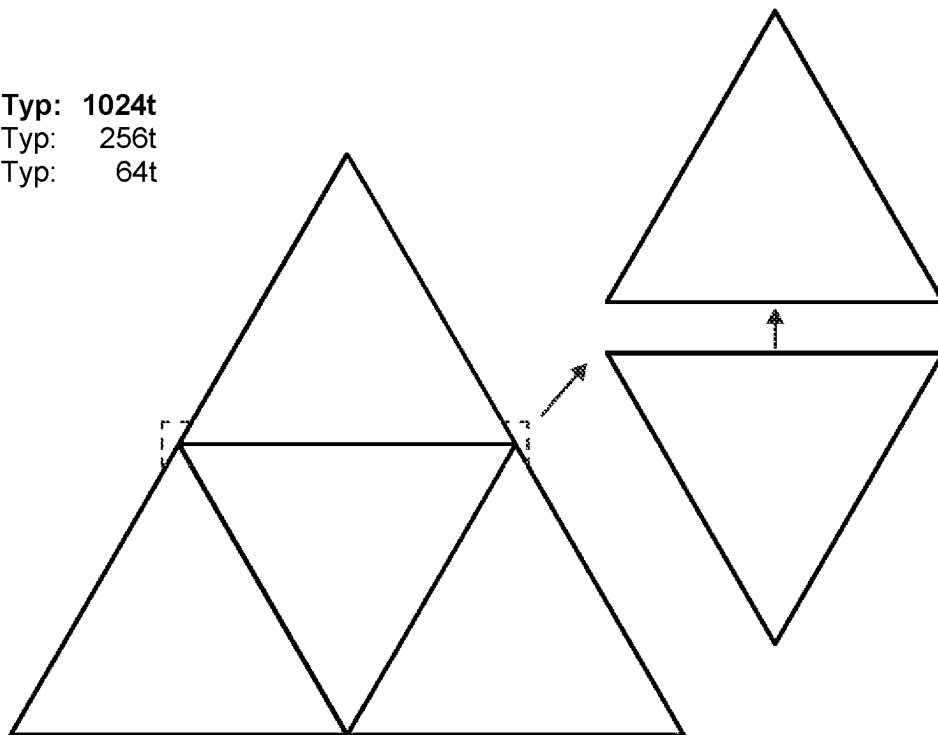


FIG. 23C

- 1. Typ: 1024t
- 2. Typ: **256t**
- 3. Typ: 64t

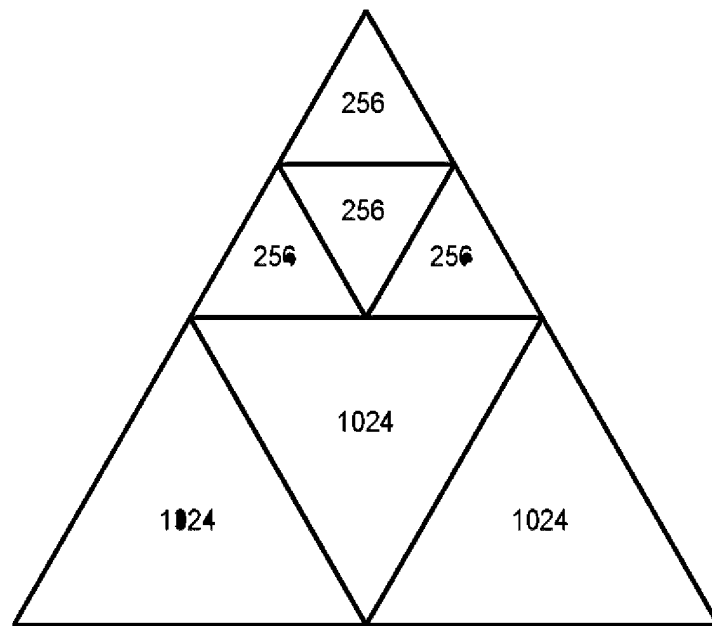


FIG. 23D

1. Typ: 1024t
2. Typ: **256t**
3. Typ: 64t

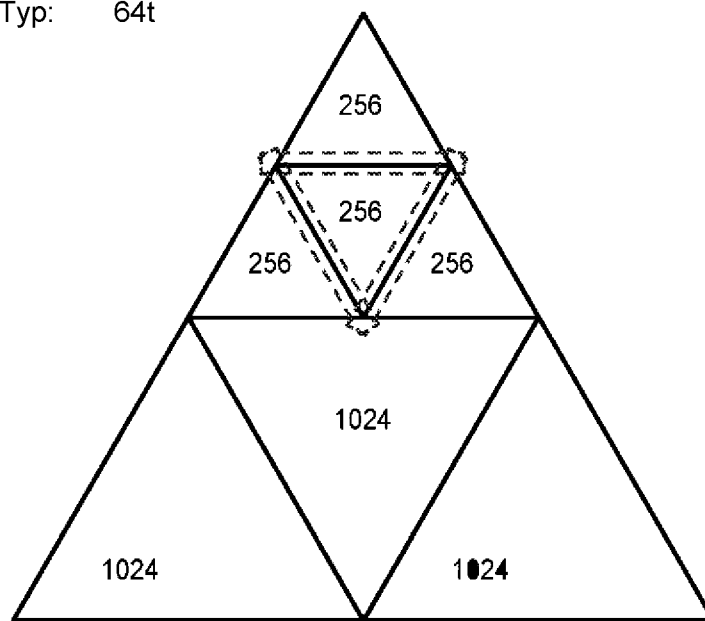


FIG. 23E

1. Typ: 1024t
2. Typ: 256t
3. Typ: 64t

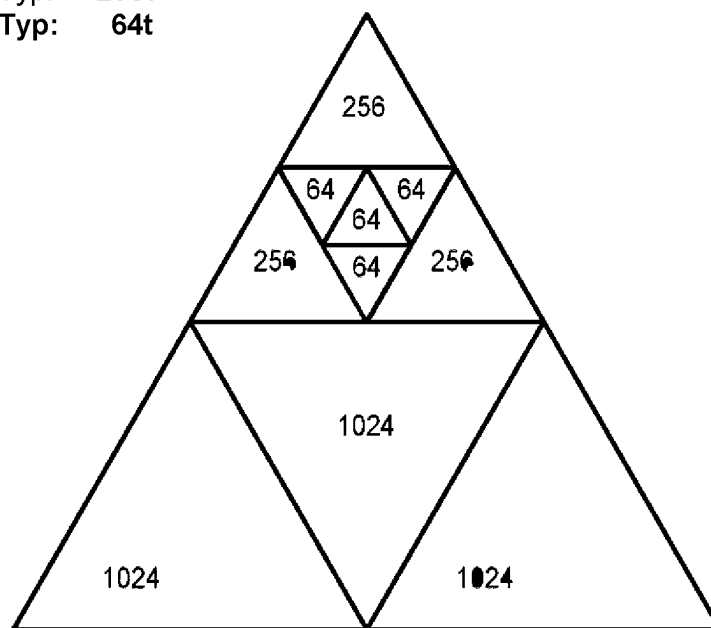


FIG. 23F

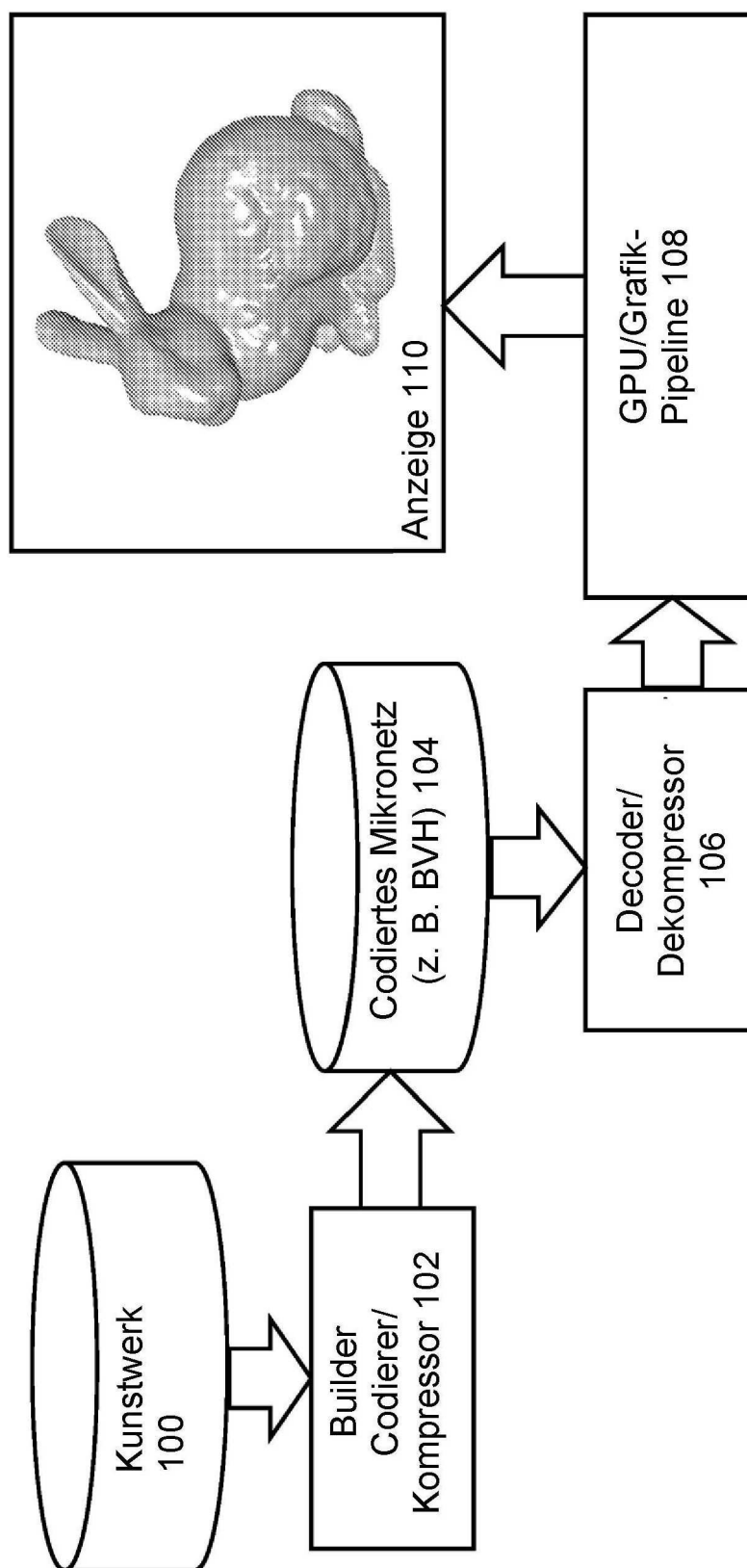


FIG. 24

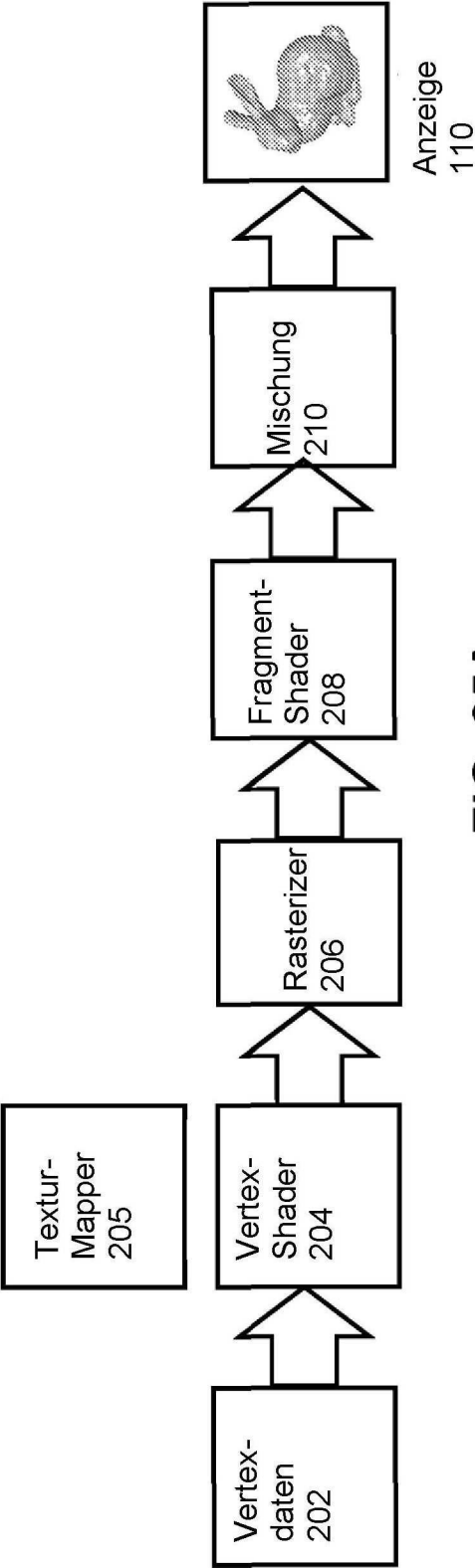


FIG. 25A

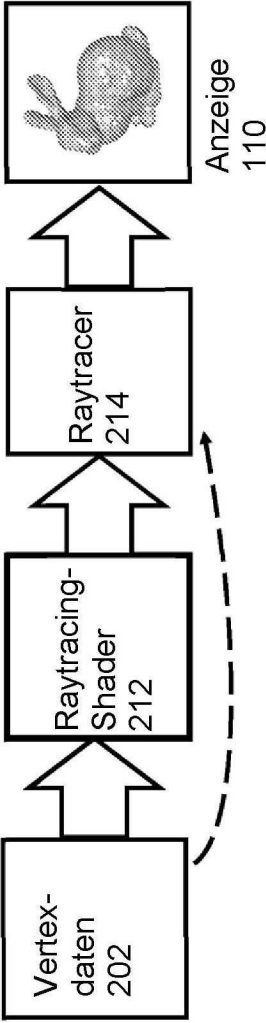


FIG. 25B

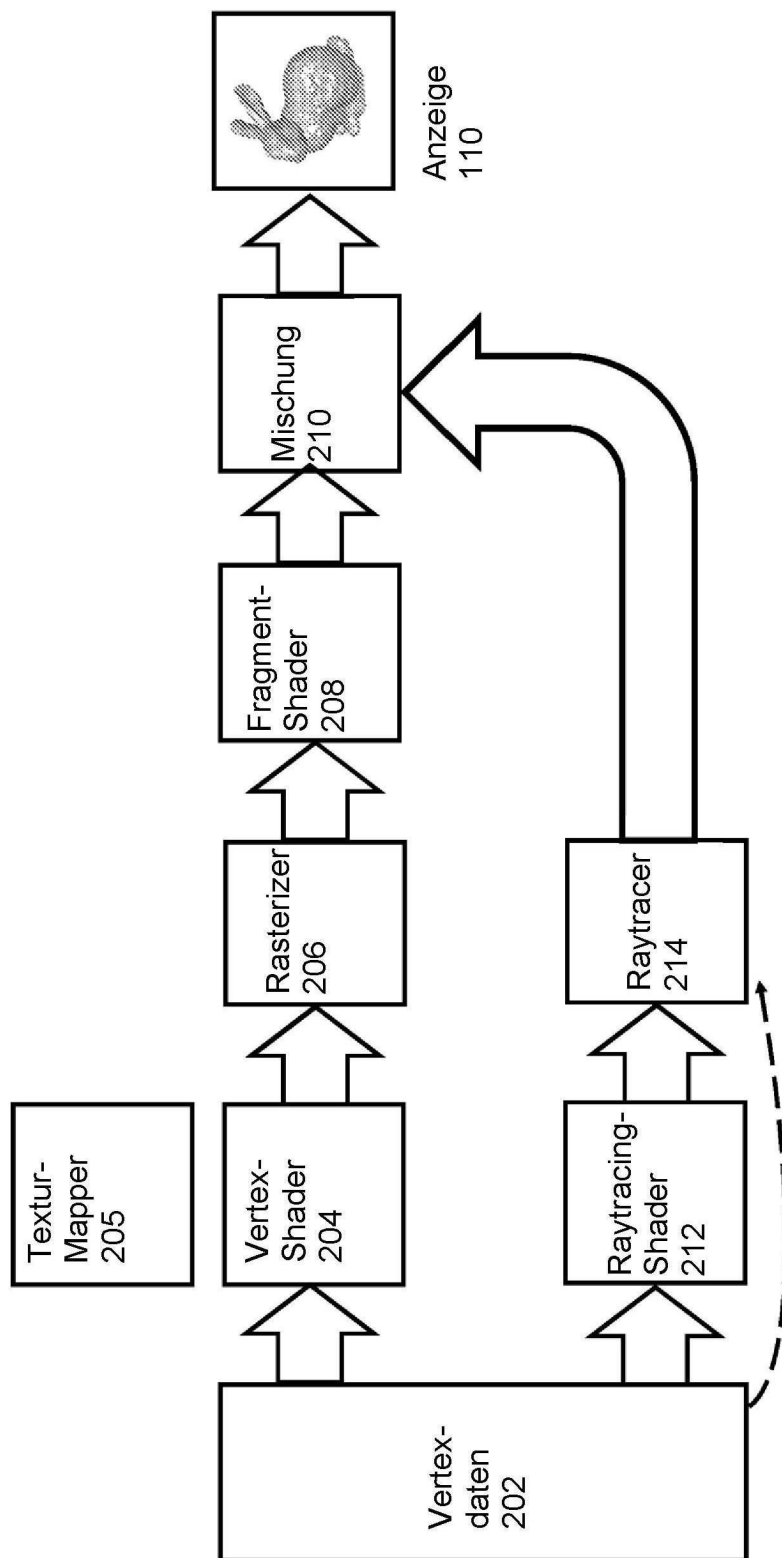


FIG. 25C

214

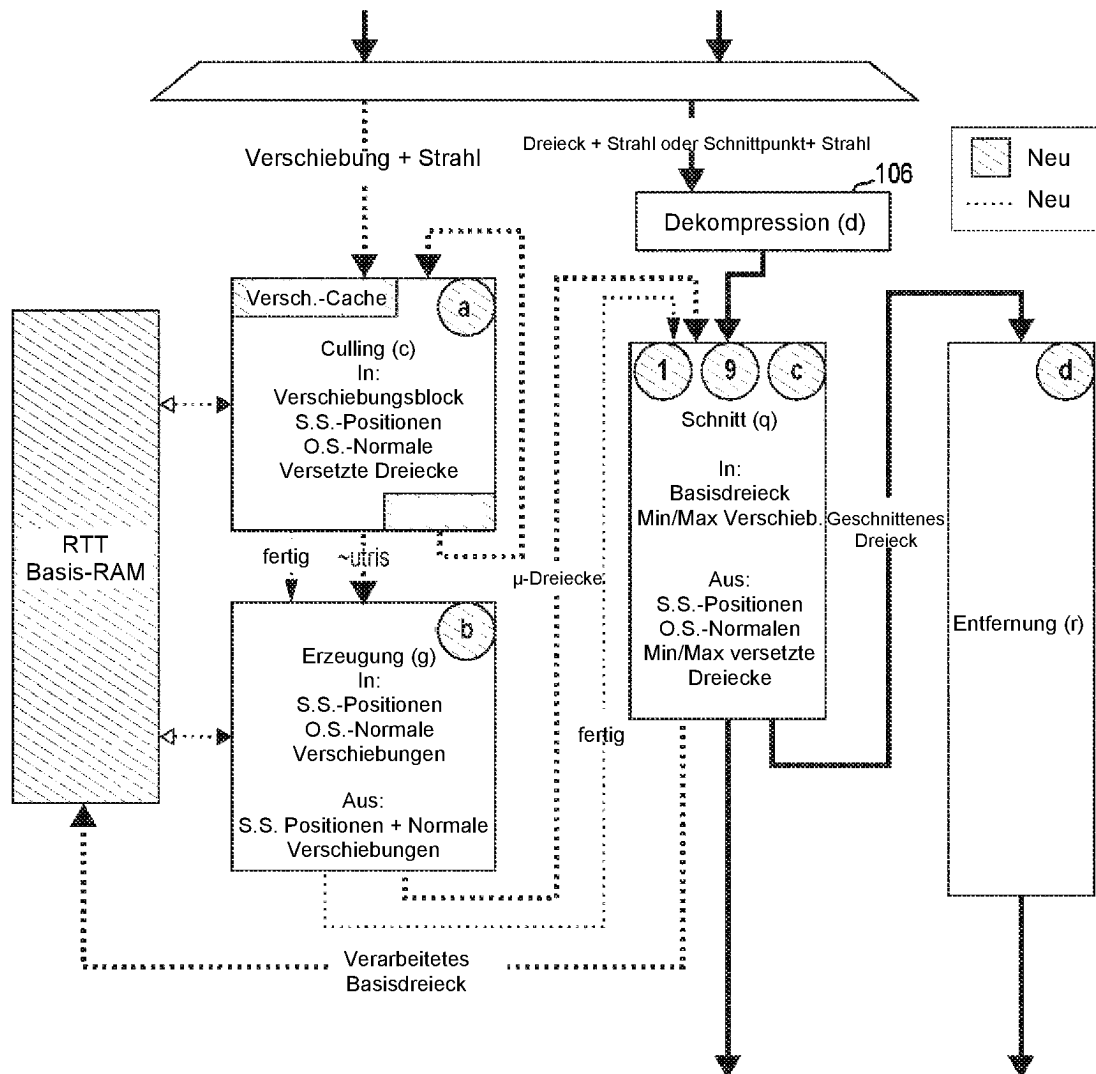


FIG. 26