(19) Deutsches Patent- und Markenamt

(10) **DE 10 2021 102 748 A1** 2021.08.12

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2021 102 748.9**
(22) Anmeldetag: **05.02.2021**
(43) Offenlegungstag: **12.08.2021**

(51) Int Cl.: ***G06T 7/70*** (2017.01)
G06T 1/40 (2006.01)
G01C 11/04 (2006.01)
G01C 3/00 (2006.01)

(30) Unionspriorität:
62/975,103 **11.02.2020** **US**
16/897,057 **09.06.2020** **US**

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

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

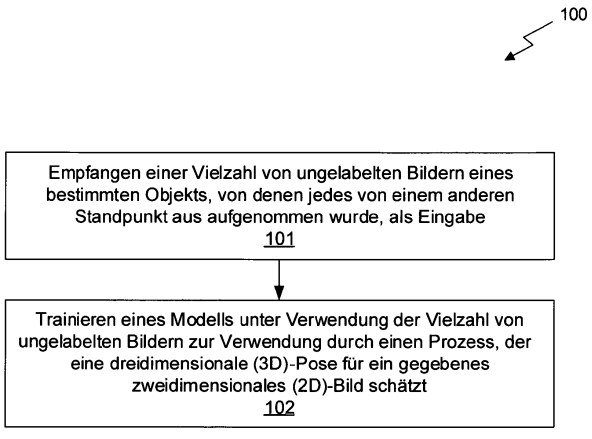
(72) Erfinder:
Iqbal, Umar, Santa Clara, US; Molchanov, Pavlo, Santa Clara, US; Kautz, Jan, Westford, Mass., US

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **3D-POSENSCHÄTZUNG DES MENSCHLICHEN KÖRPERS UNTER VERWENDUNG EINES MODELLS, DAS AUSGEHEND VON UNGELABELTEN MULTI-VIEW-DATEN TRAINIERT WURDE**

(57) Zusammenfassung: Das Erlernen der Schätzung einer 3D-Körperpose und ebenso der Pose eines beliebigen Objekts aus einem einzelnen 2D-Bild ist für viele praktische Grafikanwendungen von großem Interesse und stützt sich im Allgemeinen auf neuronale Netzwerke, die mit Beispieldaten trainiert wurden, die jedes 2D-Beispielbild mit einer bekannten 3D-Pose annotieren (labeln). Das Erfordernis dieser gekennzeichneten Trainingsdaten hat jedoch verschiedene Nachteile, einschließlich z.B. dass traditionell verwendete Trainingsdatensätze nicht vielfältig genug sind und daher das Ausmaß begrenzen, in dem neuronale Netzwerke die 3D-Pose schätzen können. Ein Erweitern dieser Trainingsdatensätze ist ebenfalls schwierig, da es manuell bereitgestellte Annotationen für 2D-Bilder erfordert, welches zeitaufwendig und fehleranfällig ist. Die Erfindung überwindet diese und andere Beschränkungen bestehender Techniken durch Bereitstellen eines Modells, das ausgehend von nicht gelabelten Multi-View-Daten zur Verwendung in der 3D-Posenschätzung trainiert ist.



Beschreibung

TECHNISCHES GEBIET

[0001] Die Erfindung bezieht sich auf neuronale Netzwerke und insbesondere auf neuronale Netzwerke zur Posenschätzung.

HINTERGRUND

[0002] Das Erlernen der Schätzung einer dreidimensionalen (3D) Körperpose und ebenso der Pose eines beliebigen Objekts aus einem einzelnen zweidimensionalen (2D) Bild ist für viele praktische Anwendungen von großem Interesse. Allgemein geht es bei dieser Schätzung darum, eine 3D-Pose zu erzeugen, die mit der räumlichen Position eines Menschen übereinstimmt, die in einem gegebenen 2D-Bild dargestellt ist. Geschätzte 3D-Körperposen können in verschiedenen Computer-Vision-Anwendungen verwendet werden.

[0003] Moderne Verfahren verwenden mit 3D-Posen annotierte Bilder und trainieren tiefe neuronale Netzwerke, um die 3D-Pose direkt aus einem Eingangsbild oder Eingangsbildern zu regressieren. Diese Verfahren weisen jedoch mehrere Nachteile auf. Während sich die Leistung dieser Verfahren deutlich verbessert hat, war ihre Anwendbarkeit auf Umgebungen in freier Wildbahn (engl. in-the-wild) aufgrund des Mangels an Trainingsdaten mit großer Vielfalt begrenzt. Die üblicherweise verwendeten Trainingsdatensätze, wie z.B. Human3.6M, werden in kontrollierten Innenräumen mit hochentwickelten Multikamera-Bewegungserfassungssystemen gesammelt.

[0004] Außerdem ist die Skalierung solcher Systeme auf uneingeschränkte Außenumgebungen unpraktisch. Dies liegt daran, dass manuelle Annotationen schwer zu erhalten und fehleranfällig sind. Daher greifen einige aktuelle Verfahren auf vorhandene Trainingsdaten zurück und versuchen, die Generalisierbarkeit trainierter Modelle zu verbessern, indem sie zusätzliche schwache Überwachung in Form von verschiedenen 2D-Annotationen für „in-the-wild“-Bilder einbeziehen. Während 2D-Annotationen leicht zu erhalten sind, bieten sie keine ausreichenden Informationen über die 3D-Körperhaltung bzw. Körperpose, insbesondere wenn die Körpergelenke verkürzt oder verdeckt sind.

[0005] Es besteht ein Bedarf, diese Probleme und/oder andere Probleme, die mit dem Stand der Technik verbunden sind, zu lösen.

KURZBESCHREIBUNG

[0006] Ein Verfahren, ein computerlesbares Medium und ein System zum Trainieren eines dreidimensionalen (3D) Posenschätzungsmodells ausgehend von ungelabelten Multi-View-Daten werden offenbart. In Verwendung wird eine Vielzahl von ungelabelten Bildern eines bestimmten Objekts als Eingabe empfangen, wobei jedes ungelabelte Bild von einem anderen Standpunkt aus aufgenommen wurde. Ferner wird ein Modell unter Verwendung der mehreren ungelabelten Bilder für die Verwendung durch einen Prozess trainiert, der eine 3D-Pose für ein gegebenes 2-dimensionales (2D) Bild schätzt.

Figurenliste

Fig. 1 zeigt ein Ablaufdiagramm eines Verfahrens zum Trainieren eines Modells ausgehend von ungelabelten Multi-View-Daten zur Verwendung bei der dreidimensionalen (3D) Posenschätzung, gemäß einem Ausführungsbeispiel.

Fig. 2 zeigt Schichten eines Systems zur Schätzung der 3D-Pose aus einem gegebenen 2D-Bild, gemäß einem Ausführungsbeispiel.

Fig. 3A zeigt ein Blockdiagramm des Trainingsprozesses für ein Modell der ersten Schicht des Systems von **Fig. 2**, gemäß einem Ausführungsbeispiel.

Fig. 3B zeigt ein Blockdiagramm des Prozesses einer zweiten Schicht des Systems von **Fig. 2**, gemäß einem Ausführungsbeispiel.

Fig. 4A zeigt eine Inferenz- und/oder Trainingslogik, gemäß mindestens einem Ausführungsbeispiel.

Fig. 4B zeigt die Inferenz- und/oder Trainingslogik, gemäß mindestens einem Ausführungsbeispiel.

Fig. 5 zeigt ein Training und einen Einsatz eines neuronalen Netzwerks, gemäß mindestens einem Ausführungsbeispiel.

Fig. 6 zeigt ein beispielhaftes Rechenzentrumsystem, gemäß mindestens einem Ausführungsbeispiel.

DETAILLIERTE BESCHREIBUNG

[0007] Fig. 1 zeigt ein Ablaufdiagramm eines Verfahrens **100** zum Trainieren eines Modells aus ungelabelten Multi-View-Daten zur Verwendung bei der dreidimensionalen (3D) Posenschätzung, gemäß einem Ausführungsbeispiel. Das Verfahren **100** kann von einer Verarbeitungseinheit, einem Programm, einer kundenspezifischen Schaltungsanordnung oder einer Kombination davon ausgeführt werden.

[0008] Wie in einer Operation **101** gezeigt, wird eine Vielzahl von ungelabelten bzw. nicht gekennzeichneten Bildern eines bestimmten Objekts als Eingabe empfangen, wobei jedes ungelabelte Bild von einem anderen Standpunkt aus aufgenommen wurde. Da die ungelabelten Bilder von verschiedenen Stand-

punkten (z.B. verschiedenen Kamerapositionen) aus aufgenommen sind, können die ungelabelten Bilder hierin auch als Mehrfachansichts- bzw. Multi-View-Bilder bezeichnet werden, wobei jedes ein zweidimensionales (2D) Bild ist, das das Objekt im Wesentlichen in der gleichen Pose oder möglicherweise zum gleichen Zeitpunkt, aber von einem anderen Standpunkt bzw. Blickwinkel aus aufnimmt. Es wird angemerkt, dass die Vielzahl der ungelabelten Bilder mindestens zwei ungelabelte Bilder enthalten kann.

[0009] In verschiedenen hierin beschriebenen Ausführungsbeispielen wird das Objekt als ein Mensch bezeichnet. Es versteht sich jedoch, dass diese Ausführungsbeispiele auch auf andere Kategorien von strukturellen Objekten angewendet werden können, die durch eindeutige Schlüsselpunkte repräsentiert werden können, und daher nicht nur auf menschliche Objekte beschränkt sind. In anderen Ausführungsbeispielen kann das Objekt z.B. ein statisches Objekt (z.B. ein Stuhl), ein sich bewegendes Objekt (z.B. ein Auto) usw. sein.

[0010] Im Kontext der vorliegenden Beschreibung beziehen sich die ungelabelten Bilder auf Bilder ohne 3D-Posenannotation. Daher kann die 3D-Posen-Grundwahrheit bzw. 3D Pose Ground Truth für das Objekt unbekannt sein. In einem weiteren Ausführungsbeispiel können die ungelabelten Bilder ohne Kalibrierung der Kameraposition aufgenommen sein. Mit anderen Worten kann eine Position bzw. ein Ort jeder Kamera in Bezug auf das Objekt unbekannt oder zumindest nicht spezifiziert sein.

[0011] Ferner wird, wie in einer Operation **102** gezeigt, ein Modell unter Verwendung der Vielzahl von ungelabelten Bilder zur Verwendung durch einen Prozess (eines Systems) trainiert, der eine 3D-Pose für ein gegebenes 2D-Bild schätzt. Das gegebene 2D-Bild kann ein beliebiges ungelabeltes Bild eines Objekts sein, wobei das Objekt der gleichen Kategorie angehört wie das Objekt, das durch die in Operation **101** empfangenen ungelabelten Bilder erfasst wurde. Zu diesem Zweck kann das Verfahren **100** optional für verschiedene Objektkategorien wiederholt werden, um das Modell für die Verwendung beim Schätzen der 3D-Pose für gegebene 2D-Bilder dieser verschiedenen Objektkategorien zu trainieren.

[0012] In einem Ausführungsbeispiel, insbesondere wenn das Objekt ein Mensch ist, kann die 3D-Pose durch 3D-Positionen von Gelenken (z.B. Knochengelenken) in Bezug auf eine Kamera (d. h. eine bestimmte Kameraposition in Bezug auf das Objekt) definiert sein. In einem Ausführungsbeispiel kann der Prozess eine erste Schicht beinhalten, die das Modell enthält, das dazu trainiert ist, eine Pose der Dimension 2,5 (2,5D) für das gegebene 2D-Bild vorherzusagen. Die 2,5D-Pose wird durch 2D-Positionskoordinaten und relative Tiefenwerte definiert. In einem weiteren Ausführungsbeispiel kann der Prozess eine zweite Schicht enthalten, die eine 3D-Rekonstruktion der 2,5D-Pose implementiert, um die 3D-Pose für das gegebene 2D-Bild zu schätzen.

[0013] Nachstehend werden verschiedene Ausführungsbeispiele des Trainings und der Verwendung des oben erwähnten Modells dargelegt, einschließlich verschiedener optionaler Architekturen und Funktionen, mit denen das obige Framework je nach den Wünschen des Benutzers implementiert werden kann. Es wird ausdrücklich darauf hingewiesen, dass die folgenden Informationen nur zur Veranschaulichung dienen und in keiner Weise als beschränkend anzusehen sind. Jedes der folgenden Merkmale kann optional mit oder ohne den Ausschluss anderer beschriebener Merkmale vorgesehen sein. Zum Beispiel können sich die unten beschriebenen Ausführungsbeispiele auf ein neuronales Netzwerk beziehen, jedoch wird angemerkt, dass auch andere lernbasierte Modelle in Betracht gezogen werden können.

[0014] **Fig. 2** zeigt ein Blockdiagramm, das Schichten eines Systems **200** zeigt, das eine 3D-Pose aus einem gegebenen 2D-Bild schätzt, gemäß einem Ausführungsbeispiel. Das hierin beschriebene System **200** kann ein Ausführungsbeispiel des Systems sein, das den oben in **Fig. 1** beschriebenen Prozess implementiert.

[0015] Wie gezeigt, enthält das System **200** eine erste Schicht **201** mit einem neuronalen Netz, das die 2,5D-Pose für ein gegebenes 2D-Bild eines Objekts vorhersagt. Die 2,5D-Pose wird durch 2D-Positionskoordinaten und relative Tiefenwerte definiert. So kann die erste Schicht **201** des Systems **200** als Eingabe das gegebene 2D-Bild empfangen und das gegebene 2D-Bild des Objekts verarbeiten, um die 2D-Positionskoordinaten des Objekts sowie die relativen Tiefenwerte für diese 2D-Positionskoordinaten zu schätzen.

[0016] Wie ebenfalls gezeigt, enthält das System **200** eine zweite Schicht **202**, die eine 3D-Rekonstruktion der 2,5D-Pose durchführt, um eine 3D-Pose für das gegebene 2D-Bild des Objekts zu schätzen. Die zweite Schicht **202** des Systems **200** kann als Eingabe die von der ersten Schicht **201** des Systems **200** vorhergesagte 2,5D-Pose empfangen und kann ferner eine 3D-Rekonstruktion der 2,5D-Pose durchführen, um eine 3D-Pose für das gegebene 2D-Bild des Objekts zu schätzen.

D-Posen-Darstellung

[0017] Im Stand der Technik ist die 2,5D-Posen-Darstellung $P^{2.4D} = \left\{ P_j^{2.5D} = \left(x_j, y_j, Z_j^r \right) \right\}_{j \in J}$, worin x_j und y_j die 2D-Projektion des Körpergelenks j auf ei-

ne Kameraebene sind und $Z_j^r = Z_{root} - Z_j$ seine metrische Tiefe in Bezug auf das Root- bzw. Wurzelgelenk repräsentiert. Diese Zerlegung der 3D-Gelenkpositionen in ihre 2D-Projektion und relative Tiefe hat den Vorteil, dass eine zusätzliche Überwachung ausgehend von „in-the-wild"-Bildern mit nur 2D-Posenannotationen für eine bessere Generalisierung der trainierten Modelle verwendet werden kann. Allerdings berücksichtigt diese Darstellung nicht die in dem Bild vorhandene Maßstabsmehrdeutigkeit, welche zu Mehrdeutigkeiten in den Vorhersagen führen kann.

[0018] Die in den vorliegenden Ausführungsbeispielen verwendete 2,5D-Darstellung unterscheidet sich jedoch in Bezug auf die Maßstabsnormierung bzw. Maßstabsvereinheitlichung von 3D-Posen von den übrigen. Ein Ausführungsbeispiel der 2,5D-Darstellung ist in der US-Patentanmeldung Nr. 16/290,643 offenbart, die am 3.1.2019 eingereicht wurde und den Titel „THREE-DIMENSIONAL (3D) POSE ESTIMATION FROM A MONOCULAR CAMERA" trägt, und deren Einzelheiten hierin durch Bezugnahme aufgenommen werden. Insbesondere skalieren sie die 3D-Pose P so, dass ein bestimmtes Paar von Körpergelenken einen festen Abstand C hat:

$$\hat{\mathrm{P}} = \frac{C}{s} \cdot \mathrm{P}, \qquad (1)$$

worin s = $||P_k - P_l||_2$ für jede Pose unabhängig geschätzt wird. Die resultierende maßstabsnormierte bzw. maßstabsvereinheitlichte 2,5D-Posendarstellung $\hat{P}_j^{2.5D} = \left(x_j, y_j, \hat{Z}_j^r\right)$ ist agnostisch gegenüber dem Maßstab des Menschen, welches nicht nur die Schätzung aus beschnittenen RGB-Bildern erleichtert, sondern auch die Rekonstruktion der absoluten 3D-Pose des Menschen bis zu einem Maßstabs- bzw. Skalierungsfaktor auf vollständig differenzierbare Weise ermöglicht, wie unten beschrieben wird.

Differenzierbare 3D-Rekonstruktion

[0019] Angesichts der 2,5D-Pose $P^{2.5D}$ besteht das Ziel darin, die Tiefe Z_{root} des Root- bzw. Wurzelgelenks zu finden, um die maßstabsvereinheitlichte 3D-Positionen ^ von Körpergelenken unter Verwendung einer perspektivischen Projektion zu rekonstruieren:

$$\hat{P}_j = \hat{Z}_j K^{-1} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix} = \left(\hat{Z}_{root} + \hat{Z}_j^r\right) K^{-1} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix}. \qquad (2)$$

[0020] Der Wert von Z_{root} kann über die Maßstabsvereinheitlichungsbedingung berechnet werden:

$$\left(\hat{X}_k - \hat{X}_l\right)^2 + \left(\hat{Y}_k - \hat{Y}_l\right)^2 + \left(\hat{Z}_k - \hat{Z}_l\right)^2 = C^2, \qquad (3)$$

$$a = \left(\bar{x}_k - \bar{x}_l\right)^2 + \left(\bar{y}_k - \bar{y}_l\right)^2$$

$$b = 2\left(\hat{Z}_k^r \left(\bar{x}_k^2 + \bar{y}_k^2 - \bar{x}_k \bar{x}_l - \bar{y}_k \bar{y}_l\right) + \hat{Z}_l^r \left(\bar{x}_l^2 + \bar{y}_l^2 - \bar{x}_k \bar{x}_l - \bar{y}_k \bar{y}_l\right)\right) \qquad (4)$$

$$c = \left(\bar{x}_k \hat{Z}_k^r - \bar{x}_l \hat{Z}_l^r\right)^2 + \left(\bar{y}_k \hat{Z}_k^r - \bar{y}_l \hat{Z}_l^r\right)^2 + \left(\hat{Z}_k^r - \hat{Z}_l^r\right)^2 - C^2, \qquad (5)$$

d.h., $\hat{Z}_{root} = 0.5\left(-b + \sqrt{b^2 - 4ac}\right) / a$ Hier ist $\left[\bar{x} \quad \bar{y} \quad 1\right] = K^{-1}\left[x \quad y \quad 1\right]^{\mathrm{T}}$, und entspricht das Paar (k, l) den Indizes der Körpergelenke, die für die Maßstabsvereinheitlichung in Gleichung (1) oben verwendet werden. Da alle Operationen für die 3D-Rekonstruktion differenzierbar sind, können Verlustfunktionen entwickelt werden, die direkt auf den rekonstruierten 3D-Posen operieren.

[0021] In den folgenden Beschreibungen wird die maßstabsvereinheitlichte 2,5D-Darstellung verwendet. Es wird C = 1 verwendet, sowie der Abstand zwischen den Hals- und Beckengelenken zur Berechnung des Skalierungsfaktors s. Natürlich versteht sich, dass C ein beliebiger anderer Wert sein kann. Im Gegensatz zu den hierin beschriebenen Ausführungsbeispielen verwenden bekannte Verfahren entweder die Ground-Truth-Werte für Z_{root} oder setzen optimierungsbasierte Verfahren ein.

D-Posenregression welches zu einer quadratischen Gleichung mit den folgenden Koeffizienten führt:

[0022] Da die 3D-Pose analytisch aus der 2,5D-Pose rekonstruiert werden kann, wird das Netzwerk so trainiert, dass es die 2,5D-Pose vorhersagt und die 3D-Rekonstruktion als eine zusätzliche Schicht implementiert, wie in dem neuronalen Netzwerk **200** in **Fig. 2** gezeigt. Zu diesem Zweck wird ein 2,5D-Heatmap- bzw. Wärmekarten-Regressionsansatz verwendet. Konkret erzeugt das neuronale Netzwerk **200** bei einem RGB-Bild als Eingabe 2J Kanäle als Ausgabe, wobei J Kanäle für 2D-Wärmekarten (H^{2D}) stehen, während die verbleibenden J Kanäle als latente Tiefenkarten H^{Zr} betrachtet werden. Die 2D-Wärmekarten werden durch zunächst vereinheitlichen bzw. normalisieren derselben unter Verwendung von Spatial Softmax:

$$\bar{H}_j^{2D}(x,y) = \frac{\exp\left(\lambda H_j^{2D}(x,y)\right)}{\sum_{x',y' \in \chi} \exp\left(\lambda H_j^{2D}(x',y')\right)}, \tag{6}$$

und dann Verwendung der Operation softargmax:

$$x_j, y_j = \sum_{x,y \in \chi} \bar{H}_j^{2D}(x,y) \cdot (x,y), \tag{7}$$

in einen Vektor von 2D-Positionskoordinaten umgewandelt, wobei X ein 2D-Gitter ist, das entsprechend der effektiven Schrittweite des Netzwerks **200** abgetastet wird, und A eine Konstante ist, die die Temperatur der vereinheitlichten Wärmekarten steuert.

[0023] Der relative maßstabsvereinheitlichte Tiefenwert $\hat{Z}_j^r$ für jedes Körpergelenk kann dann als die Summe der elementweisen Multiplikation von $\bar{H}_j^{2D}$ und latenten Tiefenkarten

[0024] H^{Zr} erhalten werden:

$$\hat{Z}_j^r = \sum_{x,y \in \chi} \bar{H}_j^{2D}(x,y) \odot H^{\hat{Z}^r}(x,y). \tag{8}$$

[0025] Angesichts der 2D-Posenkoordinaten $\{(x_j, y_j)\}_{j \in J}$, der relativen Tiefen Z^r und intrinsischer Kameraparameter K kann die 3D-Pose unter Verwendung der zweiten Schicht des neuronalen Netzwerks **202** rekonstruiert werden, wie oben erklärt wurde.

[0026] In der vollständig überwachten (FS; fully supervised) Einstellung kann das Netzwerk unter Verwendung der folgenden Verlustfunktion trainiert werden:

$$\mathcal{L}_{FS} = \mathcal{L}_H\left(H^{2D}, H_{gt}^{2D}\right) + \psi \mathcal{L}_{\hat{Z}}\left(\hat{Z}^r, \hat{Z}_{gt}^r\right), \tag{9}$$

worin H_{gt}^{2D} und $\hat{Z}_{gt}^r$ die Ground Truth-2D-Wärmekarten bzw. die Ground Truthmaßstabsvereinheitlichten relativen Tiefenwerte sind. Als die Verlustfunktionen $\mathcal{L}_{\hat{Z}}(\cdot)$ und $\mathcal{L}_H(\cdot)$ wird der mittlere quadratische Fehler verwendet. In anderen möglichen Ausführungsbeispielen können auch andere Verlustfunktionen verwendet werden, aber in jedem Fall minimieren diese Verlustfunktionen die Differenz zwischen den Vorhersagen des neuronalen Netzwerks und den Ground Truth-2D-Annotationen.

[0027] Zu diesem Zweck muss das Netzwerk **200** nicht unbedingt 2D-Wärmekarten auf latente Weise lernen. Stattdessen können die 2D-Wärmekarten-Vorhersagen über Ground-Truth-Wärmekarten mit Gauß-Verteilungen an den wahren Gelenkpositionen überwacht werden. Die Konfidenzwerte werden herangezogen, um einen schwach überwachten Verlust zu entwickeln, der robust gegenüber Unsicherheiten in 2D-Posenschätzungen ist, wie unten beschrieben. Es können jedoch auch andere Ausführungsbeispiele verwendet werden, bei denen die Wärmekarten auf latente Weise gelernt werden.

[0028] Das Ziel der **Fig. 3A-B** ist es, das durch Gewichte θ parametrisierte neuronale Faltungsnetzwerk F(I,θ) zu trainieren, das bei einem gegebenen RGB-Bild I als Eingabe die 3D-Körperpose $P = \{P_j\}_{j \in J}$ schätzt, die aus 3D-Positionen $P_j = (X_j, Y_j, Z_j) \in \mathbb{R}^3$ von J Körpergelenken in Bezug auf die Kamera besteht.

[0029] Der Trainingsprozess geht nicht von irgendwelchen Trainingsdaten mit gepaarten 2D-3D-Annotationen aus, sondern lernt die Parameter θ des Netzwerks auf schwach überwachte Weise unter Verwendung von nur Mehrfachansichts- bzw. Multi-View-Bildern. Das Netzwerk wird trainiert, um die 2,5D-Posendarstellung für das gegebene 2D-Bild eines Objekts zu bestimmen, wie oben beschrieben. Diese 2,5D-Posendarstellung hat mehrere Schlüsselmerkmale, die die Ausnutzung von Multi-View-Informationen ermöglichen, wie oben erwähnt, und es werden Verlustfunktionen für schwach überwachtes Training entwickelt.

[0030] **Fig. 3A** veranschaulicht ein Blockdiagramm des Trainingsprozesses **300** für das Modell der ersten Schicht des Systems **200** von **Fig. 2**, gemäß einem Ausführungsbeispiel. **Fig. 3B** veranschaulicht ein Blockdiagramm des Prozesses **350** für eine zweite Schicht des Systems **200** von **Fig. 2** gemäß einem Ausführungsbeispiel. Während die Prozesse **300**, **350** hierin unter Bezugnahme auf ein menschliches Objekt beschrieben werden, wird angemerkt, dass die Prozesse **300**, **350** in ähnlicher Weise für jede beliebige andere Kategorie von Objekten verwendet werden können.

[0031] Wie in **Fig. 3A** dargestellt, wird ein Trainingsdatensatz in ein neuronales Faltungsnetzwerk (CNN; Convolutional Neural Network) eingegeben. Das CNN lernt anhand des Trainingsdatensatzes, eine 2,5D-Pose aus einem gegebenen Eingangsbild abzuleiten. Wie in **Fig. 3B** gezeigt, wird die von dem CNN ausgegebene 2,5D-Pose unter Verwendung vordefinierter mathematischer Formeln verarbeitet, um die endgültige Ausgabe zu erhalten, welche die 3D-Rekonstruktion (d. h. die maßstabsvereinheitlichte 3D-Pose) ist. Eine beispielhafte Implementierung folgt.

[0032] Hierin wird ein Ausführungsbeispiel zum Trainieren des Regressionsnetzwerks in einer schwach

überwachten Weise ohne jegliche 3D-Annotationen beschrieben. Für das Training wird ein Satz von $M = \left\{ \left\{ I_v^n \right\}_{v \in V_n} \right\}_{n \in N_N}$ Stichproben angenommen, wobei die n-te Stichprobe aus V_n Ansichten eines Menschen in gleicher Körperhaltung besteht. Die Bilder mit mehreren Ansichten bzw. Multi-View-Bilder können gleichzeitig unter Verwendung mehrerer Kameras oder unter der Annahme einer über der Zeit statischen Körperpose unter Verwendung einer einzelnen Kamera aufgenommen werden. Keine Kenntnis extrinsischer Kameraparameter. Darüber hinaus kann in einem Ausführungsbeispiel ein unabhängiger Satz von Bildern, die nur mit 2D-Posen annotiert sind, verwendet werden (nicht gezeigt), der reichlich vorhanden ist oder von Menschen annotiert werden kann, auch für Daten in freier Wildbahn. Für das Training wird die folgende schwach überwachte (WS; weakly supervised) Verlustfunktion optimiert:

$$\mathcal{L}_{WS} = \mathcal{L}_H \left(H^{2D}, H_{gt}^{2D} \right) + \alpha \mathcal{L}_{MC}(M) + \beta \mathcal{L}_B \left(\hat{L}, \hat{\mu}^L \right), \tag{10}$$

worin $\mathcal{L}_H$ 2D-Heatmap-Verlust, L_{MC} der Multi-View-Konsistenzverlust und $\mathcal{L}_B$ der Gliedmaßenlängenverlust ist.

[0033] Es sei daran erinnert, dass es bei einem gegebenen RGB-Bild das Ziel ist, die maßstabsvereinheitlichte 2,5D-Pose $\hat{P}^{2.5D} = \left\{ P_j^{2.5D} = \left(x_j, y_j, \hat{z}_j^r \right) \right\}_{j \in J}$ zu schätzen, aus der die maßstabsvereinheitlichte 3D-Pose $\hat{P}$ rekonstruiert wird, wie oben erklärt. Während die $\mathcal{L}_H$ eine Überwachung für 2D-Posenschätzung bereitstellt, ist der Verlust L_{MC} für das Lernen der Schätzung der relativen Tiefen-Komponente $\left(\hat{Z}_j^r \right)$ verantwortlich. Der Gliedmaßenlängenverlust $\mathcal{L}_B$ gewährleistet ferner, dass die rekonstruierte 3D-Pose $\hat{P}$ plausible Gliedmaßenlängen hat. Diese Verlustfunktionen werden im Folgenden näher erklärt.

[0034] Wie in **Fig. 3A** gezeigt, wird die erste Schicht **201** dazu trainiert, 2D-Wärmekarten H^{2D} für jedes ungelabelte Bild der Vielzahl von ungelabelten Bildern $I_{1\text{-}v}$ zu generieren und um latente Tiefenkarten $H^{\hat{z}r}$ für jedes ungelabelte Bild aus der Vielzahl von ungelabelten Bildern zu generieren.

[0035] In einem Ausführungsbeispiel kann auch mindestens ein gelabeltes Bild empfangen werden, wobei jedes gelabelte Bild von einem entsprechenden Objekt ist, das sich von dem Objekt unterscheidet, das durch ungelabelte Bilder $I_{1\text{-}v}$ erfasst wurde, und mit einer 2D-Pose des entsprechenden Objekts annotiert ist. In diesem Ausführungsbeispiel wird das gelabelte Bild zur Bestimmung des Wärmekartenverlusts $\mathcal{L}_H$ während des Trainings verwendet, welches weiter unten im Einzelnen beschrieben wird.

[0036] Die erste Schicht **201** vereinheitlicht bzw. normalisiert die 2D-Wärmekarten weiter, um vereinheitlichte bzw. normalisierte 2D-Wärmekarten zu erzeugen. Darüber hinaus konvertiert die erste Schicht **201** die vereinheitlichten 2D-Wärmekarten in 2D-Posenkoordinaten. In diesem Ausführungsbeispiel erhält die erste Schicht **201** relative Tiefenwerte aus den latenten Tiefenkarten und den vereinheitlichten 2D-Wärmekarten, wobei die relativen Tiefenwerte und die 2D-Posenkoordinaten die 2,5D-Pose definieren.

[0037] Wie in **Fig. 3B** gezeigt, bestimmt die zweite Schicht **202** eine Tiefe eines Root- bzw. Wurzelgelenks und verwendet die Tiefe des Wurzelgelenks, um maßstabsvereinheitlichte 3D-Positionen von Gelenken unter Verwendung einer perspektivischen Projektion zu rekonstruieren. In diesem Ausführungsbeispiel werden vordefinierte Durchschnittslängen für einen oder mehrere Aspekte des Objekts verwendet, um den Längenverlust (z.B. den Knochenverlust $\mathcal{L}_B$), der den maßstabsvereinheitlichten 3D-Positionen der Gelenke während des Trainings zugeordnet ist, zu bestimmen, wie in einem nachstehenden Ausführungsbeispiel beschrieben. Die zweite Schicht **202** führt eine starre Ausrichtung für die maßstabsvereinheitlichten 3D-Positionen der Gelenke durch, um die 3D-Pose zu erzeugen. Diese starre Ausrichtung richtet die 3D-Posen ausgehend von zwei verschiedenen Ansichten durch Entfernen der Rotation, der Translation und der Skalierung zwischen ihnen aus. In diesem Ausführungsbeispiel wird während des Trainings ein Multi-View-Konsistenzverlust L_{MC} für die 3D-Pose bestimmt, wobei ein Multi-View-Konsistenzverlust erzwingt, dass aus verschiedenen Ansichten erzeugte 3D-Posen bis zu hin zu einer starren Transformation gleich sein sollten.

Verlust

[0038] Ein Wärmekartenverlust $\left(\mathcal{L}_H \right)$ misst die Differenz bzw. den Unterschied zwischen vorhergesagten 2D-Wärmekarten H^{2D} und Ground-Truth-Wärmekarten H_{gt}^{2D} mit einer Gauß-Verteilung an der wahren Gelenkposition. Er wirkt nur auf Bilder, die mit 2D-Posen annotiert sind, und wird für alle anderen Bilder als Null angenommen.

[0039] Ein Multi-View-Konsistenzverlust $\left(\mathcal{L}_{MC} \right)$ erzwingt, dass die aus verschiedenen Ansichten erhaltenen 3D-Posenschätzungen bis hin zu einer starren

Transformation identisch sind. Formal gesehen ist bei einer gegebenen Multi-View-Trainingsstichprobe mit V Ansichten $\{I_v\}_{v\in V}$ der Multi-View-Konsistenzverlust definiert als die gewichtete Summe der Differenz zwischen den 3D-Gelenkpositionen über verschiedene Ansichten nach einer starren Ausrichtung:

$$\mathcal{L}_{MC} = \sum_{\substack{r,r'\in V \\ r\neq r'}} \sum_{j\in J} \phi_{j,v}\phi_{j,v'} \cdot d\left(\hat{P}_{j,n}, R_v^{v'}\hat{P}_{j,v'}\right). \tag{11}$$

worin

$$\phi_{j,v} = H_{j,v}^{2D}\left(x_{j,v}, y_{j,v}\right) \text{ and } \phi_{j,v'} = H_{j,v'}^{2D}\left(x_{j,v'}, y_{j,v'}\right)$$

die Konfidenzwerte des j-ten Gelenks in Standpunkt I_v bzw. $I_{v'}$ sind. Die $P_{j,v}$ and $P_{j,v'}$ sind die maßstabsvereinheitlichten 3D-Koordinaten des j-ten Gelenks, die ausgehend von dem Standpunkt I_v bzw. $I_{v'}$ geschätzt werden. $R_v^{v'} \in \mathbb{R}^{3\times 4}$ ist eine starre Transformationsmatrix, die die beiden 3D-Posen am besten ausrichtet, und d ist die Abstandsmetrik, die verwendet wird, um den Unterschied zwischen den ausgerichteten Posen zu messen. In einem Ausführungsbeispiel wird L1-norm als die Abstandsmetrik d verwendet. Um den Beitrag von $\mathcal{L}_{MC}$ klarer zu verstehen, kann der Abstandsterm in (11) in Bezug auf die 2,5D-Posendarstellung unter Verwendung von (2) neu geschrieben werden, d. h.:

$$d\left(\hat{P}_{j,v}, R_v^{v'}\hat{P}_{j,v'}\right) = d\left(\left(\hat{Z}_{root,v} + \hat{Z}_{j,v}^{r}\right)K_v^{-1}\begin{bmatrix} x_{j,v} \\ y_{j,v} \\ 1 \end{bmatrix}, R_v^{v'}\left(\hat{Z}_{root,v'} + \hat{Z}_{j,v'}^{r}\right)K_{v'}^{-1}\begin{bmatrix} x_{j,v'} \\ y_{j,v'} \\ 1 \end{bmatrix}\right). \tag{12}$$

[0040] Es sei angenommen, dass die 2D-Koordinaten ($x_{j,v}$, $y_{j,v}$) und ($x_{j,v'}$, $y_{j,v'}$) aufgrund des Verlusts $\mathcal{L}_H$ genau vorhergesagt werden und die Kamera-Intrinsik K_v and $K_{v'}$ bekannt ist. Der Einfachheit sei auch angenommen, dass die Ground-Truth-Transformation $R_v^{v'}$ zwischen den beiden Ansichten bzw. Standpunkten bekannt ist. Dann besteht die einzige Möglichkeit für das Netzwerk, die Differenz d(.,.) zu minimieren, darin, die korrekten Werte für die relativen Tiefen $\hat{Z}_{j,v}^{r}$ and $\hat{Z}_{j,v'}^{r}$ vorherzusagen. Daher ermöglicht die gemeinsame Optimierung der Verluste L_{MC} und $\mathcal{L}_H$ das Erlernen korrekter 3D-Posen unter Verwendung nur schwacher Überwachung in Form von Multi-View-Bildern und 2D-Posenannotationen. Ohne den Verlust $\mathcal{L}_H$ kann das Modell zu degenerierten Lösungen führen, obwohl auch andere Ausführungsbeispielen in Betracht gezogen werden, bei denen der Verlust $\mathcal{L}_H$ während des Trainings nicht genutzt wird.

[0041] Während in vielen praktischen Szenarien die Transformationsmatrix $R_v^{v'}$ über eine extrinsische Kalibrierung a priori bekannt sein kann, wird in den vorliegenden Ausführungsbeispielen davon ausgegangen, dass sie nicht verfügbar ist, und wird sie unter Verwendung vorhergesagter 3D-Posen und der Procrustes-Analyse wie folgt geschätzt:

$$R_v^{v'} = \underset{R}{\operatorname{argmin}} \sum_{j\in J} \phi_{j,v}\phi_{j,v'} \left\| \hat{P}_{j,v} - R\hat{P}_{j,v'} \right\|_2^2. \tag{13}$$

[0042] Während des Trainings darf ein Ausführungsbeispiel nicht über die Optimierung der Transformationsmatrix (13) rückpropagieren, da dies zu numerischen Instabilitäten führt, die aufgrund einer Singulärwertzerlegung entstehen. Die Gradienten aus $\mathcal{L}_{MC}$ beeinflussen nicht nur die Tiefenschätzungen, sondern beeinträchtigen auch die Wärmekarten-Vorhersagen aufgrund der Berechnung von Z_{root} in (4). Daher kann L_{MC} die Fehler in 2D-Posenschätzungen beheben.

[0043] Ein Gliedmaßenlängenverlust $(\mathcal{L}_B)$ misst die Abweichung der Gliedmaßenlängen einer vorhergesagten 3D-Pose von den mittleren Knochenlängen:

$$\mathcal{L}_B = \sum_{j,j'\in\varepsilon} \phi_j \left(\left\| \hat{P}_j - \hat{P}_{j'} \right\| - \hat{\mu}_{j,j'}^{L} \right)^2, \tag{14}$$

worin ε der verwendeten kinematischen Struktur des menschlichen Körpers entspricht und $\hat{\mu}_{j,j'}^{L}$ die maßstabsvereinheitlichte mittlere Gliedmaßenlänge für ein Gelenkpaar (j, j') ist. Da die Gliedmaßenlängen aller Menschen nach der Maßstabsvereinheitlichung (1) ungefähr gleich sind, stellt dieser Verlust sicher, dass die vorhergesagten Posen plausible Gliedmaßenlängen haben. Während des Trainings führt ein Gliedmaßenlängenverlust zu einer schnelleren Konvergenz.

Zusätzliche Regularisierung

[0044] Falls eine große Anzahl von Abtastwerten in Multi-View-Daten einen konstanten Hintergrund haben, kann das Netzwerk **200** lernen, diese Bilder zu erkennen und sagt für diese Bilder die gleiche 2D-Pose und die gleichen relativen Tiefen voraus. Um dies zu verhindern, kann für solche Abtastwer-

te ein zusätzlicher Regularisierungsverlust eingebaut werden. Insbesondere kann ein vortrainiertes 2D-Posenschätzungsmodell ausgeführt werden, um Pseudo-Ground-Truths zu erzeugen, indem gemeinsame Schätzungen mit einem Konfidenzwert größer als ein Schwellenwert τ = 0.5 ausgewählt werden. Diese Pseudo-Ground-Truths werden dann verwendet, um den 2D-Wärmekartenverlust $\mathcal{L}_{\text{H}}$ zu erzwingen, welcher verhindert, dass das Modell degenerierte Lösungen vorhersagt. Die Pseudo-Ground-Truths können einmal zu Beginn des Trainings erzeugt und während des gesamten Trainings beibehalten werden. Konkret kann der Regularisierungsverlust für Bilder von Human3.6M und MPII-INF-3DHP verwendet werden, die beide in kontrollierten Innenraumkonfigurationen aufgenommen werden.

Beispielhafte Implementierung

[0045] Wir verwenden HRNet-w32 als das Grundgerüst der Netzwerkarchitektur. Wir trainieren das Modell für die 2D-Posenschätzung vor, bevor wir schwach überwachte Verluste einführen. Dadurch wird sichergestellt, dass die 2D-Posenschätzungen ausreichend gut sind, um eine Multi-View-Konsistenz bzw. eine Konsistent zwischen mehreren Ansichten $\mathcal{L}_{\text{MC}}$ zu erzwingen. Wir verwenden maximal vier Ansichten V_n =4 zur Berechnung von $\mathcal{L}_{\text{MC}}$. Falls ein Abtastwert mehr als vier Ansichten enthält, nehmen wir daraus in jeder Epoche zufällig vier Ansichten heraus. Wir trainieren das Modell mit einer Stapelgröße von 256, wobei jeder Stapel aus 128 Bildern mit 2D-Posenannotationen und 32 ungelabelten Multi-View-Abtastwerten besteht (32 x 4=128 Bilder). Wir verwenden eine Eingabebildauflösung von 256x256. Die Trainingsdaten werden durch zufällige Skalierung (+- 20%) und Rotation (+- 30% Grad) erweitert. Wir haben festgestellt, dass das Training nach 60k Iterationen konvergiert. Die Lernrate wird auf 5e-4 gesetzt, die bei 50k Iterationen nach dem Adam-Optimierungsalgorithmus auf 5e-5 sinkt. Wir verwenden λ=50 in (6). Da die Trainingsziele (9) und (10) aus mehreren Verlusttermen bestehen, gleichen wir ihre Beiträge durch empirische Wahl von ψ=5. α=10. und β=100 aus. Da unser Modell für die Posenschätzung absolute 3D-Posen bis hin zu einem Maßstabsfaktor schätzt, approximieren wir während der Inferenz den Maßstab unter Verwendung mittlerer Knochenlängen aus den Trainingsdaten:

$$\hat{s} = \underset{s}{\operatorname{argmin}} \sum_{j,j' \in \varepsilon} \left(s \cdot \left\| \hat{P}_j - \hat{P}_{j'} \right\| - \mu^{L}_{j,j'} \right)^2, \tag{15}$$

worin $\mu^{L}_{j,j'}$ die mittlere Länge der durch das Gelenkpaar (j, j') gebildeten Extremität ist.

[0046] Das oben beschriebene Ausführungsbeispiel stellt einen schwach überwachten Ansatz für 3D-Posenschätzung von Menschen in freier Wildbahn dar. Der vorgeschlagene Ansatz erfordert keine 3D-Annotationen und kann lernen, 3D-Posen aus ungelabelten Multi-View-Daten zu schätzen. Ermöglicht wird dies durch ein neuartiges Ende-zu-Ende-Lern-Framework und eine neuartige Zielfunktion, die so optimiert ist, dass sie konsistente 3D-Posen über verschiedene Kameraansichten bzw. Kamerastandpunkte hinweg vorhersagt. Da das Sammeln ungelabelter Multi-View-Daten sehr einfach ist und ähnliche Daten im Internet reichlich vorhanden sind, profitiert der Ansatz, wenn zusätzliche ungelabelte Daten zur Verfügung gestellt werden. Dies macht den Ansatz sehr praktisch, da zusätzliche Daten sehr einfach eingebunden werden können, um die Generalisierbarkeit der trainierten Modelle auf zuvor nicht gesehene Umgebungen zu verbessern.

Maschinelles Lernen

[0047] Tiefe neuronale Netzwerke (DNNs, Deep Neural Networks), die hierin auch als neuronale Netzwerke bezeichnet werden und Modelle für tiefes Lernen bzw. Deep-Learning-Modelle umfassen, welche auf Prozessoren entwickelt wurden, sind für verschiedenste Anwendungsfälle eingesetzt worden, von selbstfahrenden Autos bis zu schnellerer Medikamentenentwicklung, von automatischer Bildbeschriftung in Online-Bilddatenbanken bis zu intelligenter Echtzeit-Sprachübersetzung in Video-Chat-Anwendungen. Deep Learning ist eine Technik, die den Prozess des neuronalen Lernens des menschlichen Gehirns modelliert, kontinuierlich lernt, immer schlauer wird und mit der Zeit immer genauere Ergebnisse liefert. Ein Kind lernt anfangs von einem Erwachsenen, verschiedene Formen richtig zu identifizieren und zu klassifizieren, und ist schließlich in der Lage, Formen ohne Nachhilfe zu erkennen. In ähnlicher Weise muss ein System für Deep Learning oder neuronales Lernen in der Objekterkennung und -klassifizierung trainiert werden, damit es schlauer und effizienter bei der Identifizierung von Basisobjekten, verdeckten Objekten usw. wird und gleichzeitig den Objekten einen Kontext zuordnen kann.

[0048] Auf der einfachsten Ebene betrachten Neuronen im menschlichen Gehirn verschiedene Inputs bzw. Eingaben, die empfangen werden, werden jedem dieser Eingaben Wichtigkeitsstufen zugewiesen, und wird eine Ausgabe an andere Neuronen weitergeleitet, die darauf reagieren. Ein künstliches Neuron oder Perzeptron ist das einfachste Modell eines neuronalen Netzwerks. In einem Beispiel kann ein Perzeptron eine oder mehrere Eingaben empfangen, die verschiedene Merkmale eines Objekts repräsentieren, für dessen Erkennung und Klassifizierung das Perzeptron trainiert wird, und wird jedem dieser Merkmale eine bestimmte Gewichtung zugewiesen,

die auf der Wichtigkeit dieses Merkmals bei der Definition der Form eines Objekts basiert.

[0049] Ein Modell eines tiefen neuronalen Netzwerks (DNN) beinhaltet mehrere Schichten vieler verbundenen Knoten (z.B. Perceptrons, Boltzmann-Maschinen, Radialbasisfunktionen, Faltungsschichten usw.), die mit enormen Mengen an Eingabedaten trainiert werden können, um komplexe Probleme schnell und mit hoher Genauigkeit zu lösen. In einem Beispiel zerlegt eine erste Schicht des DNN-Modells ein Eingabebild eines Automobils in verschiedene Abschnitte und sucht nach grundlegenden Mustern wie beispielsweise Linien und Winkeln. Die zweite Schicht setzt die Linien zusammen, um nach Mustern einer höheren Ebene wie beispielsweise Rädern, Windschutzscheiben und Spiegeln zu suchen. Die nächste Schicht identifiziert den Fahrzeugtyp, und die letzten paar Schichten generieren ein Label für das Eingabebild, das das Modell einer bestimmten Automarke identifiziert.

[0050] Sobald das DNN trainiert ist, kann das DNN eingesetzt und zur Identifizierung und Klassifizierung von Objekten oder Mustern in einem als Inferenz bekannten Prozess verwendet werden. Beispiele für Inferenz (der Prozess, durch den ein DNN nützliche Informationen aus einer gegebenen Eingabe extrahiert) sind die Identifizierung handgeschriebener Zahlen auf Schecks, die in Geldautomaten hinterlegt wurden, die Identifizierung von Bildern von Freunden auf Fotos, die Bereitstellung von Filmempfehlungen für über fünfzig Millionen Benutzer, die Identifizierung und Klassifizierung verschiedener Arten von Autos, Fußgängern und Straßengefahren in fahrerlosen Autos oder die Übersetzung menschlicher Sprache in Echtzeit.

[0051] Während des Trainings fließen die Daten in einer Vorwärtspropagationsphase durch das DNN, bis eine Vorhersage erzeugt wird, die ein der Eingabe entsprechendes Label angibt. Falls das neuronale Netzwerk die Eingabe nicht korrekt labelt bzw. kennzeichnet, werden Fehler zwischen dem korrekten Label bzw. der korrekten Beschriftung und der vorhergesagten Beschriftung analysiert und werden die Gewichte für jedes Merkmal in einer Rückwärtspropagierungsphase angepasst, bis das DNN die Eingabe und andere Eingaben in einem Trainingsdatensatz korrekt labelt. Das Training komplexer neuronaler Netzwerke erfordert enorme Mengen an paralleler Rechenleistung, einschließlich Fließkommamultiplikationen und - additionen. Die Inferenzierung ist weniger rechenintensiv als das Training, da sie ein latenzempfindlicher Prozess ist, bei dem ein trainiertes neuronales Netzwerk auf neue Eingaben angewendet wird, die es zuvor noch nicht gesehen hat, um Bilder zu klassifizieren, Sprache zu übersetzen und allgemein neue Informationen abzuleiten.

INFERENZ- UND TRAININGSLOGIK

[0052] Wie oben erwähnt, muss ein System für Deep-Learning oder neuronales Lernen trainiert werden, um Inferenzen bzw. Schlussfolgerungen aus Eingabedaten zu generieren. Einzelheiten zur Inferenz- und/oder Trainingslogik **415** für ein System für Deep Learning oder neuronales Lernen werden nachstehend in Verbindung mit den **Fig. 4A** und/oder 4B bereitgestellt.

[0053] In mindestens einem Ausführungsbeispiel kann die Inferenz- und/oder Trainingslogik **415**, ohne darauf beschränkt zu sein, einen Datenspeicher **401** beinhalten, um Vorwärts- und/oder Ausgangs-Gewichtungen und/oder Eingabe-/Ausgabe-Daten zu speichern, die Neuronen oder Schichten eines neuronalen Netzwerks entsprechen, das in Aspekten eines oder mehrerer Ausführungsbeispiele trainiert und/oder zur Inferenzierung verwendet wird. In mindestens einem Ausführungsbeispiel speichert der Datenspeicher **401** Gewichtungsparameter und/oder Eingabe-/Ausgabe-Daten jeder Schicht eines neuronalen Netzwerks, das während der Vorwärtspropagierung von Eingabe-/Ausgabedaten und/oder Gewichtungsparametern während des Trainings und/oder der Inferenzierung unter Verwendung von Aspekten eines oder mehrerer Ausführungsbeispiele trainiert oder in Verbindung mit einem oder mehreren Ausführungsbeispielen verwendet wird. In mindestens einem Ausführungsbeispiel kann ein beliebiger Teil des Datenspeichers **401** in einen anderen On-Chip- oder Off-Chip-Datenspeicher, einschließlich des L1-, L2- oder L3-Cachespeichers oder des Systemspeichers eines Prozessors, enthalten sein.

[0054] In mindestens einem Ausführungsbeispiel kann jeder beliebige Teil des Datenspeichers **401** intern oder extern zu einem oder mehreren Prozessoren oder anderen Hardware-Logikgeräten oder -Schaltungen sein. In mindestens einem Ausführungsbeispiel kann der Datenspeicher **401** ein Cachespeicher, ein dynamischer Direktzugriffsspeicher („DRAM“), ein statischer Direktzugriffsspeicher („SRAM“), ein nichtflüchtiger Speicher (z.B. Flash-Speicher) oder ein anderer Speicher sein. In mindestens einem Ausführungsbeispiel kann die Wahl, ob der Datenspeicher **401** beispielsweise intern oder extern zu einem Prozessor ist oder aus DRAM, SRAM, Flash oder einem anderen Speichertyp besteht, von dem verfügbaren Speicher auf dem Chip gegenüber dem verfügbaren Speicher außerhalb des Chips, den Latenzanforderungen der durchgeführten Trainings- und/oder Inferenzfunktionen, der Stapelgröße von Daten, die bei der Inferenzierung und/oder dem Training eines neuronalen Netzwerks verwendet werden, oder einer Kombination dieser Faktoren abhängen.

[0055] In mindestens einem Ausführungsbeispiel kann die Inferenz- und/oder Trainingslogik **415**, oh-

ne darauf beschränkt zu sein, einen Datenspeicher **405** enthalten, um Rückwärts- und/oder Ausgangsgewichtungen und/oder Eingabe-/Ausgabedaten zu speichern, die Neuronen oder Schichten eines neuronalen Netzwerks entsprechen, das in Aspekten eines oder mehrerer Ausführungsbeispiele trainiert und/oder zur Inferenzierung verwendet wird. In mindestens einem Ausführungsbeispiel speichert der Datenspeicher **405** Gewichtungsparameter und/oder Eingabe-/Ausgabedaten jeder Schicht eines neuronalen Netzwerks, das während der Rückwärtspropagation von Eingabe-/Ausgabedaten und/oder Gewichtungsparametern während des Trainings und/oder der Inferenzierung unter Verwendung von Aspekten eines oder mehrerer Ausführungsbeispiele trainiert oder in Verbindung mit einem oder mehreren Ausführungsbeispielen verwendet wird. In mindestens einem Ausführungsbeispiel kann ein beliebiger Teil des Datenspeichers **405** in einem anderen On-Chip- oder Off-Chip-Datenspeicher, einschließlich des L1-, L2- oder L3-Cachespeichers oder Systemspeichers eines Prozessors, enthalten sein. In mindestens einem Ausführungsbeispiel kann ein beliebiger Teil des Datenspeichers **405** intern oder extern zu einem oder mehreren Prozessoren oder anderen Hardware-Logikvorrichtungen oder -Schaltungen sein. In mindestens einem Ausführungsbeispiel kann der Datenspeicher **405** ein Cache-Speicher, DRAM, SRAM, nichtflüchtiger Speicher (z.B. Flash-Speicher) oder ein anderer Speicher sein. In mindestens einem Ausführungsbeispiel kann die Wahl, ob der Datenspeicher **405** z.B. intern oder extern zu einem Prozessor ist oder aus DRAM, SRAM, Flash oder einem anderen Speichertyp besteht, von dem verfügbaren Speicher auf dem Chip gegenüber dem verfügbaren Speicher außerhalb des Chips, den Latenzanforderungen der durchgeführten Trainings- und/oder Inferenzfunktionen, der Stapelgröße der bei der Inferenzierung und/oder dem Training eines neuronalen Netzwerks verwendeten Daten oder einer Kombination dieser Faktoren abhängen.

[0056] In mindestens einem Ausführungsbeispiel können der Datenspeicher **401** und der Datenspeicher **405** separate Speicherstrukturen sein. In mindestens einem Ausführungsbeispiel können der Datenspeicher **401** und der Datenspeicher **405** dieselbe Speicherstruktur sein. In mindestens einem Ausführungsbeispiel können der Datenspeicher **401** und der Datenspeicher **405** teilweise dieselbe Speicherstruktur und teilweise separate Speicherstrukturen sein. In mindestens einem Ausführungsbeispiel kann ein beliebiger Teil des Datenspeichers **401** und des Datenspeichers **405** in einen anderen On-Chip- oder Off-Chip-Datenspeicher, einschließlich des L1-, L2- oder L3-Cache oder Systemspeichers eines Prozessors, enthalten sein.

[0057] In mindestens einem Ausführungsbeispiel kann die Inferenz- und/oder Trainingslogik **415**, ohne darauf beschränkt zu sein, eine oder mehrere Arithmetik-Logik-Einheit(en) („ALU(s)“) **410** enthalten, um logische und/oder mathematische Operationen durchzuführen, die zumindest teilweise auf einem Trainings- und/oder Inferenzcode basieren oder durch diesen angezeigt werden, deren Ergebnis in Aktivierungen (z.B. Ausgangswerten von Schichten oder Neuronen innerhalb eines neuronalen Netzwerks) resultieren kann, die in einem Aktivierungsspeicher **420** gespeichert sind und Funktionen von Eingabe-/Ausgabe- und/oder Gewichtungsparameterdaten sind, die in dem Datenspeicher **401** und/oder in dem Datenspeicher **405** gespeichert sind. In mindestens einem Ausführungsbeispiel werden die in dem Aktivierungsspeicher **420** gespeicherten Aktivierungen gemäß linearer algebraischer und/oder matrixbasierter Mathematik erzeugt, die von der/den ALU(s) **410** als Reaktion auf Ausführungsbefehle oder anderen Code ausgeführt wird, wobei in dem Datenspeicher **405** und/oder in dem Datenspeicher **401** gespeicherte Gewichtungswerte als Operanden zusammen mit anderen Werten, wie z.B. Bias-Werten, Gradienteninformationen, Impulswerten oder anderen Parametern oder Hyperparametern, verwendet werden, die alle oder teilweise in dem Datenspeicher **405** oder in dem Datenspeicher **401** oder in einem anderen Speicher auf oder außerhalb des Chips gespeichert sein können. In mindestens einem Ausführungsbeispiel sind die ALU(s) **410** in einem oder mehreren Prozessoren oder anderen Hardware-Logikgeräten oder -Schaltungen enthalten, während in einem anderen Ausführungsbeispiel die ALU(s) **410** extern zu einem Prozessor oder einem anderen Hardware-Logikgerät oder einer Schaltung sein können, die sie verwenden (z.B. ein Co-Prozessor). In mindestens einem Ausführungsbeispiel können die ALUs **410** in den Ausführungseinheiten eines Prozessors oder anderweitig in einer Bank von ALUs enthalten sein, auf die die Ausführungseinheiten eines Prozessors zugreifen können, entweder innerhalb desselben Prozessors oder verteilt auf verschiedene Prozessoren unterschiedlichen Typs (z.B. zentrale Verarbeitungseinheiten, Grafikverarbeitungseinheiten, Festfunktionseinheiten usw.). In mindestens einem Ausführungsbeispiel können sich der Datenspeicher **401**, der Datenspeicher **405** und der Aktivierungsspeicher **420** auf demselben Prozessor oder einer anderen Hardware-Logikvorrichtung oder -Schaltung befinden, während sie sich in einem anderen Ausführungsbeispiel in verschiedenen Prozessoren oder anderen Hardware-Logikvorrichtungen oder -schaltungen oder in einer Kombination aus gleichen und verschiedenen Prozessoren oder anderen Hardware-Logikvorrichtungen oder - schaltungen befinden können. In mindestens einem Ausführungsbeispiel kann ein beliebiger Teil eines Aktivierungsspeichers **620** in anderen On-Chip- oder Off-Chip-Datenspeichern, einschließlich des L1-, L2- oder L3-Cache oder Systemspeichers eines Prozessors, enthalten sein. Darüber hinaus kann der Inferenz- und/oder Trainings-

code zusammen mit anderem Code gespeichert sein, auf den ein Prozessor oder eine andere Hardware-Logik oder -Schaltung zugreifen kann und der mithilfe der Abruf-, Dekodier-, Planungs-, Ausführungs-, Ausscheidungs- und/oder anderer logischer Schaltungen eines Prozessors abgerufen und/oder verarbeitet wird.

[0058] In mindestens einem Ausführungsbeispiel kann ein Aktivierungsspeicher **420** Cachespeicher, DRAM, SRAM, nichtflüchtiger Speicher (z.B. Flash-Speicher) oder ein anderer Speicher sein. In mindestens einem Ausführungsbeispiel kann sich der Aktivierungsspeicher **420** vollständig oder teilweise innerhalb oder außerhalb eines oder mehrerer Prozessoren oder anderer logischer Schaltungen befinden. In mindestens einem Ausführungsbeispiel kann die Wahl, ob der Aktivierungsspeicher **420** beispielsweise intern oder extern zu einem Prozessor ist oder aus DRAM, SRAM, Flash oder einem anderen Speichertyp besteht, von dem verfügbaren Speicher auf dem Chip gegenüber dem verfügbaren Speicher außerhalb des Chips, den Latenzanforderungen der ausgeführten Trainings- und/oder Inferenzfunktionen, der Stapelgröße der bei der Inferenz und/oder dem Training eines neuronalen Netzwerks verwendeten Daten oder einer Kombination dieser Faktoren abhängen. In mindestens einem Ausführungsbeispiel kann die in **Fig. 4A** dargestellte Inferenz- und/oder Trainingslogik **415** in Verbindung mit einem anwendungsspezifischen integrierten Schaltkreis („ASIC“) verwendet werden, wie z.B. der Tensorflow® Processing Unit von Google, einer Inferenzverarbeitungseinheit (IPU) von Graphcore™ oder einem Nervana® (z.B. „Lake Crest“) Prozessor von Intel Corp. In mindestens einem Ausführungsbeispiel kann die in **Fig. 4A** dargestellte Inferenz- und/oder Trainingslogik **615** in Verbindung mit Hardware der Zentraleinheit („CPU“), der Grafikverarbeitungseinheit („GPU“) oder anderer Hardware, wie z.B. Field Programmable Gate Arrays („FPGAs“), verwendet werden.

[0059] **Fig. 4B** veranschaulicht die Inferenz- und/oder Trainingslogik **415**, gemäß mindestens einem Ausführungsbeispiel. In mindestens einem Ausführungsbeispiel kann die Inferenz- und/oder Trainingslogik **415**, ohne darauf beschränkt zu sein, eine Hardware-Logik umfassen, in der Rechenressourcen dediziert oder anderweitig ausschließlich in Verbindung mit Gewichtungswerten oder anderen Informationen verwendet werden, die einer oder mehreren Schichten von Neuronen innerhalb eines neuronalen Netzwerks entsprechen. In mindestens einem Ausführungsbeispiel kann die in **Fig. 4B** dargestellte Inferenz- und/oder Trainingslogik **415** in Verbindung mit einer anwendungsspezifischen integrierten Schaltung (ASIC) verwendet werden, wie z.B. der Tensorflow® Processing Unit von Google, einer Inferenzverarbeitungseinheit (IPU) von Graphcore™ oder einem Nervana® (z.B. „Lake Crest“)-Prozessor von Intel Corp. In mindestens einem Ausführungsbeispiel kann die in **Fig. 6B** dargestellte Inferenz- und/oder Trainingslogik **415** in Verbindung mit Hardware der Zentraleinheit (CPU), der Grafikverarbeitungseinheit (GPU) oder anderer Hardware, wie z.B. FPGAs (Field Programmable Gate Arrays), verwendet werden. In mindestens einem Ausführungsbeispiel beinhaltet die Inferenz- und/oder Trainingslogik **415**, ohne darauf beschränkt zu sein, den Datenspeicher **401** und den Datenspeicher **405**, die zum Speichern von Gewichtungswerten und/oder anderen Informationen, einschließlich Bias-Werten, Gradienteninformationen, Impulswerten und/oder anderen Parameter- oder Hyperparameterinformationen, verwendet werden können. In mindestens einem in **Fig. 4B** dargestellten Ausführungsbeispiel ist jeder Datenspeicher **401** und jeder Datenspeicher **405** mit einer dedizierten Rechenressource verbunden, wie z.B. Rechenhardware **402** bzw. Rechenhardware **406**. In mindestens einem Ausführungsbeispiel beinhaltet jede Rechenhardware **406** eine oder mehrere ALUs, die mathematische Funktionen, wie z.B. lineare algebraische Funktionen, nur auf den in dem Datenspeicher **401** bzw. in dem Datenspeicher **405** gespeicherten Informationen ausführen, deren Ergebnis in dem Aktivierungsspeicher **420** gespeichert wird.

[0060] In mindestens einem Ausführungsbeispiel entsprechen jeder der Datenspeicher **401** und **405** und die entsprechende Rechenhardware **402** bzw. **406** verschiedenen Schichten eines neuronalen Netzwerks, so dass die resultierende Aktivierung von einem „Speicher-/Rechenpaar 401/402“ des Datenspeichers **401** und der Rechenhardware **402** als Eingabe für das nächste „Speicher-/Rechenpaar 405/406“ des Datenspeichers **405** und der Rechenhardware **406** bereitgestellt wird, um die konzeptionelle Organisation eines neuronalen Netzwerks zu spiegeln. In mindestens einem Ausführungsbeispiel kann jedes der Speicher-/Rechenpaare **401/402** und **405/406** mehr als einer Schicht des neuronalen Netzwerks entsprechen. In mindestens einem Ausführungsbeispiel können zusätzliche Speicher-/Berechnungs-Paare (nicht dargestellt) im Anschluss an oder parallel zu den Speicher-/Berechnungs-Paaren **401/402** und **405/406** in der Inferenz- und/oder Trainingslogik **415** enthalten sein.

TRAINING UND EINSATZ VON NEURONALEN NETZWERKEN

[0061] **Fig. 5** zeigt ein weiteres Ausführungsbeispiel für das Training und den Einsatz eines tiefen neuronalen Netzwerks. In mindestens einem Ausführungsbeispiel wird das untrainierte neuronale Netzwerk **506** mit einem Trainingsdatensatz **502** trainiert. In mindestens einem Ausführungsbeispiel ist das Trainingsframework **504** ein PyTorch-Framework, während in anderen Ausführungsbeispielen das Trainingsframework **504** ein Tensorflow-, Boost-, Caf-

fe-, Microsoft Cognitive Toolkit/CNTK-, MXNet-, Chainer-, Keras-, Deeplearning4j- oder ein anderes Trainingsframework ist. In mindestens einem Ausführungsbeispiel trainiert das Trainingsframework **504** ein untrainiertes neuronales Netzwerk **506** und ermöglicht dessen Training unter Verwendung der hierin beschriebenen Verarbeitungsressourcen, um ein trainiertes neuronales Netzwerk **508** zu erzeugen. In mindestens einem Ausführungsbeispiel können die Gewichtungen zufällig oder durch Vortraining mit einem Deep Belief Network ausgewählt werden. In mindestens einem Ausführungsbeispiel kann das Training entweder auf überwachte, teilweise überwachte oder nicht überwachte Weise durchgeführt werden.

[0062] In mindestens einem Ausführungsbeispiel wird das untrainierte neuronale Netzwerk **506** unter Verwendung von überwachtem Lernen trainiert, wobei der Trainingsdatensatz **502** eine Eingabe beinhaltet, die mit einer gewünschten Ausgabe für eine Eingabe gepaart ist, oder wobei der Trainingsdatensatz **502** eine Eingabe mit bekannter Ausgabe beinhaltet und die Ausgabe des neuronalen Netzwerks manuell bewertet wird. In mindestens einem Ausführungsbeispiel wird das untrainierte neuronale Netzwerk **506** auf überwachte Weise trainiert, verarbeitet Eingaben aus dem Trainingsdatensatz **502** und vergleicht die resultierenden Ausgaben mit einem Satz von erwarteten oder gewünschten Ausgaben. In mindestens einem Ausführungsbeispiel werden Fehler dann durch das untrainierte neuronale Netzwerk **506** zurückpropagiert. In mindestens einem Ausführungsbeispiel passt das Trainingsframework **504** die Gewichtungen an, die das untrainierte neuronale Netzwerk **506** steuern. In mindestens einem Ausführungsbeispiel enthält das Trainings-Framework **504** Werkzeuge, um zu überwachen, wie gut das untrainierte neuronale Netzwerk **506** zu einem Modell konvergiert, wie z.B. dem trainierten neuronalen Netzwerk **508**, das geeignet ist, korrekte Antworten zu erzeugen, wie z.B. im Ergebnis **514**, basierend auf bekannten Eingabedaten, wie z.B. neuen Daten **512**. In mindestens einem Ausführungsbeispiel trainiert das Trainingsframework **704** das untrainierte neuronale Netzwerk **506** wiederholt, während es die Gewichtungen anpasst, um eine Ausgabe des untrainierten neuronalen Netzwerks **506** unter Verwendung einer Verlustfunktion und eines Anpassungsalgorithmus, wie z.B. stochastischem Gradientenabstieg, zu verfeinern. In mindestens einem Ausführungsbeispiel trainiert das Trainingsframework **504** das untrainierte neuronale Netzwerk **506**, bis das untrainierte neuronale Netzwerk **506** eine gewünschte Genauigkeit erreicht. In mindestens einem Ausführungsbeispiel kann das trainierte neuronale Netzwerk **508** dann eingesetzt werden, um eine beliebige Anzahl von maschinellen Lernoperationen zu implementieren.

[0063] In mindestens einem Ausführungsbeispiel wird das untrainierte neuronale Netzwerk **506** durch unüberwachtes Lernen trainiert, wobei das untrainierte neuronale Netzwerk **506** versucht, sich selbst mit ungelabelten Daten zu trainieren. In mindestens einem Ausführungsbeispiel enthält der Trainingsdatensatz **502** des unüberwachten Lernens Eingabedaten ohne zugehörige Ausgabedaten oder „Ground Truth-Daten“. In mindestens einem Ausführungsbeispiel kann das untrainierte neuronale Netzwerk **506** Gruppierungen innerhalb des Trainingsdatensatzes **502** lernen und kann bestimmen, wie einzelne Eingaben mit dem untrainierten Datensatz **502** in Beziehung stehen. In mindestens einem Ausführungsbeispiel kann unüberwachtes Training verwendet werden, um eine selbstorganisierende Karte zu erzeugen, die eine Art von trainiertem neuronalem Netzwerk **508** ist, das in der Lage ist, Operationen durchzuführen, die bei der Reduzierung der Dimensionalität neuer Daten **512** nützlich sind. In mindestens einem Ausführungsbeispiel kann unüberwachtes Training auch dazu verwendet werden, eine Anomalieerkennung durchzuführen, die die Identifizierung von Datenpunkten in einem neuen Datensatz **512**, die von normalen Mustern des neuen Datensatzes **512** abweichen, ermöglicht.

[0064] In mindestens einem Ausführungsbeispiel kann halbüberwachtes Lernen verwendet werden, welches eine Technik ist, in welcher der Trainingsdatensatz **502** eine Mischung aus gelabelten und ungelabelten Daten enthält. In mindestens einem Ausführungsbeispiel kann das Trainingsframework **504** verwendet werden, um inkrementelles Lernen durchzuführen, wie beispielsweise durch übertragene Lerntechniken. In mindestens einem Ausführungsbeispiel ermöglicht das inkrementelle Lernen dem trainierten neuronalen Netzwerk **508**, sich an neue Daten **512** anzupassen, ohne Wissen zu vergessen, das dem Netzwerk während des anfänglichen Trainings beigebracht wurde.

RECHENZENTRUM

[0065] **Fig. 6** zeigt ein Beispiel für ein Rechenzentrum **600**, in dem mindestens ein Ausführungsbeispiel verwendet werden kann. In mindestens einem Ausführungsbeispiel beinhaltet das Rechenzentrum **600** eine Rechenzentrumsinfrastrukturschicht **610**, eine Frameworkschicht **620**, eine Softwareschicht **630** und eine Anwendungsschicht **640**.

[0066] In mindestens einem Ausführungsbeispiel, wie in **Fig. 6** gezeigt, kann die Rechenzentrumsinfrastrukturschicht **610** einen Ressourcen-Orchestrator **612**, gruppierte Rechenressourcen **614** und Knoten-Rechenressourcen („Knoten-C.R.s“) 616(1)-616(N) enthalten, wobei „N“ eine beliebige ganze, positive Zahl darstellt. In mindestens einem Ausführungsbeispiel können die Knoten-C.R.s 616(1)-616(N) eine beliebige Anzahl von Zentralverarbeitungseinheiten („CPUs“) oder anderen Prozessoren

(einschließlich Beschleunigern, feldprogrammierbaren Gate-Arrays (FPGAs), Grafikprozessoren usw.), Speichergeräten (z.B. dynamischer Festwertspeicher), Speichergeräte (z.B. Solid-State- oder Festplattenlaufwerke), Netzwerk-Eingabe-/Ausgabe-Geräte („NW-E/A“), Netzwerk-Switches, virtuelle Maschinen („VMs“), Leistungsmodule und Kühlmodule usw. enthalten. In mindestens einem Ausführungsbeispiel können ein oder mehrere Knoten-C.R.s unter den Knoten-C.R.s 616(1)-616(N) ein Server mit einer oder mehreren der oben genannten Rechenressourcen sein.

[0067] In mindestens einem Ausführungsbeispiel können die gruppierten Rechenressourcen **614** separate Gruppierungen von Knoten-C.R.s umfassen, die in einem oder mehreren Racks (nicht dargestellt) oder in vielen Racks in Rechenzentren an verschiedenen geografischen Standorten (ebenfalls nicht dargestellt) untergebracht sind. Separate Gruppierungen von Knoten-C.R.s innerhalb der gruppierten Rechenressourcen **614** können gruppierte Rechen-, Netzwerk-, Speicher- oder Speicherressourcen umfassen, die zur Unterstützung einer oder mehrerer Arbeitslasten konfiguriert oder zugewiesen sein können. In mindestens einem Ausführungsbeispiel können mehrere Knoten-C.R.s mit CPUs oder Prozessoren in einem oder mehreren Racks gruppiert sein, um Rechenressourcen zur Unterstützung einer oder mehrerer Arbeitslasten bereitzustellen. In mindestens einem Ausführungsbeispiel können ein oder mehrere Racks auch eine beliebige Anzahl von Stromversorgungsmodulen, Kühlmodulen und Netzwerk-Switches in beliebiger Kombination enthalten.

[0068] In mindestens einem Ausführungsbeispiel kann der Ressourcen-Orchestrator **622** einen oder mehrere Knoten-CRs 616(1)-616(N) und/oder gruppierte Rechenressourcen **614** konfigurieren oder anderweitig steuern. In mindestens einem Ausführungsbeispiel kann der Ressourcen-Orchestrator **622** eine Software-Design-Infrastruktur („SDI“)-Verwaltungseinheit für das Rechenzentrum **600** enthalten. In mindestens einem Ausführungsbeispiel kann der Ressourcen-Orchestrator Hardware, Software oder eine Kombination davon beinhalten.

[0069] In mindestens einem Ausführungsbeispiel, wie in **Fig. 6** gezeigt, umfasst die Frameworkschicht **620** einen Auftragsplaner bzw. Job-Scheduler **632**, einen Konfigurationsverwalter **634**, einen Ressourcenverwalter **636** und ein verteiltes Dateisystem **638**. In mindestens einem Ausführungsbeispiel kann die Frameworkschicht **620** ein Framework bzw. Rahmenwerk zur Unterstützung der Software **632** der Softwareschicht **630** und/oder einer oder mehrerer Anwendung(en) **642** der Anwendungsschicht **640** enthalten. In mindestens einem Ausführungsbeispiel können die Software **632** oder die Anwendung(en) **642** jeweils webbasierte Dienstsoftware oder Anwendungen umfassen, wie sie beispielsweise von Amazon Web Services, Google Cloud und Microsoft Azure bereitgestellt werden. In mindestens einem Ausführungsbeispiel kann die Frameworkschicht **620** eine Art freies und quelloffenes Software-Webanwendungs-Framework sein, wie z.B. Apache SparkTM (im Folgenden „Spark“), das ein verteiltes Dateisystem **638** für die Verarbeitung großer Datenmengen (z.B. „Big Data“) nutzen kann, ist aber nicht darauf beschränkt. In mindestens einem Ausführungsbeispiel kann der Auftragsplaner **632** einen Spark-Treiber enthalten, um die Planung von Arbeitslasten zu erleichtern, die von verschiedenen Schichten des Rechenzentrums **600** unterstützt werden. In mindestens einem Ausführungsbeispiel kann der Konfigurationsverwalter **634** in der Lage sein, verschiedene Schichten zu konfigurieren, z.B. die Softwareschicht **630** und die Frameworkschicht **620** einschließlich Spark und das verteilte Dateisystem **638** zur Unterstützung der Verarbeitung großer Datenmengen. In mindestens einem Ausführungsbeispiel kann der Ressourcenverwalter **636** in der Lage sein, geclusterte oder gruppierte Rechenressourcen zu verwalten, die zur Unterstützung des verteilten Dateisystems **638** und des Auftragsplaners **632** zugeordnet sind. In mindestens einem Ausführungsbeispiel können geclusterte oder gruppierte Rechenressourcen die gruppierte Rechenressource **614** auf der Rechenzentrumsinfrastrukturebene **610** umfassen. In mindestens einem Ausführungsbeispiel kann der Ressourcenverwalter **636** mit dem Ressourcen-Orchestrator **612** koordiniert werden, um diese zugeordneten oder zugewiesenen Rechenressourcen zu verwalten.

[0070] In mindestens einem Ausführungsbeispiel kann die in der Softwareschicht **630** enthaltene Software **632** Software enthalten, die von mindestens Teilen der Knoten-C.R.s 616(1)-616(N), der gruppierten Rechenressourcen **614** und/oder des verteilten Dateisystems **638** der Frameworkschicht **620** verwendet wird. Eine oder mehrere Arten von Software können Software für die Suche nach Internet-Webseiten, Software zum Abtasten auf E-Mail-Viren, Datenbanksoftware und Software für das Streaming von Videoinhalten enthalten, sind aber nicht darauf beschränkt.

[0071] In mindestens einem Ausführungsbeispiel kann (können) die in der Anwendungsschicht **640** enthaltene(n) Anwendung(en) **642** eine oder mehrere Arten von Anwendungen umfassen, die von mindestens Teilen der Knoten-C.R.s 616(1)-616(N), den gruppierten Rechenressourcen **614** und/oder dem verteilten Dateisystem **638** der Frameworkschicht **620** verwendet werden. Eine oder mehrere Anwendungen können, ohne darauf beschränkt zu sein, eine beliebige Anzahl einer Genetikanwendung, eines kognitiven Rechnens und einer Anwendung maschinellen Lernens beinhalten, einschließlich einer Trainings- oder Inferenzierungs-Software, einer Soft-

ware eines Frameworks für maschinelles Lernen (z.B. PyTorch, TensorFlow, Caffe, usw.) oder andere Anwendungen maschinellen Lernens, die in Verbindung mit einem oder mehreren Ausführungsbeispielen verwendet werden.

[0072] In mindestens einem Ausführungsbeispiel können ein beliebiger des Konfigurationsverwalters **634**, des Ressourcenverwalters **636** und des Ressourcen-Orchestrators **612** eine beliebige Anzahl und Art von selbstmodifizierenden Aktionen implementieren, die auf einer beliebigen Menge und Art von Daten basieren, die auf jede beliebige technisch mögliche Weise erlangt wurden. In mindestens einem Ausführungsbeispiel können selbstmodifizierende Aktionen einen Rechenzentrumsbetreiber eines Rechenzentrums **800** davon entlasten, möglicherweise schlechte Konfigurationsentscheidungen zu treffen, und möglicherweise nicht ausgelastete und/oder schlecht funktionierende Teile eines Rechenzentrums vermeiden.

[0073] In mindestens einem Ausführungsbeispiel kann das Rechenzentrum **600** Werkzeuge, Dienste, Software oder andere Ressourcen enthalten, um ein oder mehrere Modelle maschinellen Lernens zu trainieren oder Informationen unter Verwendung eines oder mehrerer Modelle maschinellen Lernens vorherzusagen oder abzuleiten, gemäß einem oder mehreren hierin beschriebenen Ausführungsbeispielen. Zum Beispiel kann in mindestens einem Ausführungsbeispiel ein Modell maschinellen Lernens durch Berechnen von Gewichtungsparametern gemäß einer neuronalen Netzwerkarchitektur unter Verwendung von Software und Rechenressourcen, die oben in Bezug auf das Rechenzentrum **600** beschrieben wurden, trainiert werden. In mindestens einem Ausführungsbeispiel können trainierte Modelle maschinellen Lernens, die einem oder mehreren neuronalen Netzwerken entsprechen, verwendet werden, um Informationen unter Verwendung der oben in Bezug auf das Rechenzentrum **600** beschriebenen Ressourcen abzuleiten oder vorherzusagen, indem Gewichtungsparameter verwendet werden, die durch eine oder mehrere hierin beschriebene Trainingstechniken berechnet werden.

[0074] In mindestens einem Ausführungsbeispiel kann das Rechenzentrum CPUs, anwendungsspezifische integrierte Schaltkreise (ASICs), GPUs, FPGAs oder andere Hardware verwenden, um das Training und/oder die Inferenzierung mit den oben beschriebenen Ressourcen durchzuführen. Darüber hinaus können eine oder mehrere der oben beschriebenen Software- und/oder Hardware-Ressourcen als Dienst konfiguriert sein, um Benutzern das Training oder die Inferenzierung von Informationen, wie z.B. Bilderkennung, Spracherkennung oder andere Dienste der künstlichen Intelligenz, zu ermöglichen.

[0075] Die Inferenz- und/oder Trainingslogik **415** wird verwendet, um Inferenz- und/oder Trainingsoperationen in Zuordnung mit einem oder mehreren Ausführungsbeispielen durchzuführen. In mindestens einem Ausführungsbeispiel kann die Inferenz- und/oder Trainingslogik **415** in dem System von **Fig. 6** für Inferenz- oder Vorhersageoperationen verwendet werden, die zumindest teilweise auf Gewichtungsparametern basieren, die unter Verwendung von Trainingsoperationen für neuronale Netzwerke, Funktionen und/oder Architekturen neuronaler Netzwerke oder hierin beschriebenen Anwendungsfällen für neuronale Netzwerke berechnet werden.

[0076] Wie hierin beschrieben, werden ein Verfahren, ein computerlesbares Medium und ein System zum Trainieren eines Modells aus ungelabelten Mehrfachansichtsdaten zur Verwendung bei der dreidimensionalen (3D) Posenschätzung offenbart. In Übereinstimmung mit den **Fig. 1-3B** kann ein Ausführungsbeispiel ein trainiertes Modell zum Durchführen von Inferenzoperationen (z.B. Schätzen der 2,5D-Pose) und zum Bereitstellen von inferenzierten Daten (z.B. 2,5D-Pose) bereitstellen, wobei das Modell (teilweise oder vollständig) in einem oder beiden der Datenspeicher **401** und **405** in der Inferenz- und/oder Trainingslogik **415** gespeichert ist, wie in den **Fig. 4A** und **Fig. 4B** dargestellt. Das Training und der Einsatz des Modells können wie in **Fig. 5** dargestellt und hierin beschrieben durchgeführt werden. Die Verteilung des Modells kann unter Verwendung eines oder mehrerer Server in einem Rechenzentrum **600** erfolgen, wie in **Fig. 6** dargestellt und hierin beschrieben.

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 16290643 [0018]

Patentansprüche

1. Verfahren zum Trainieren eines dreidimensionalen, 3D, Posenschätzungsmodells unter Verwendung ungelabelter Bilder, umfassend:
Empfangen einer Vielzahl von ungelabelten Bildern eines bestimmten Objekts, die jeweils von einem anderen Standpunkt aus aufgenommen wurden, als Eingabe; und
Trainieren eines Modells unter Verwendung der Vielzahl von ungelabelten Bildern zur Verwendung durch einen Prozess, der eine dreidimensionale, 3D, Pose für ein gegebenes zweidimensionales, 2D, Bild schätzt.

2. Verfahren nach Anspruch 1, wobei die Vielzahl von ungelabelten Bildern Bilder ohne 3D-Posenannotationen enthält.

3. Verfahren nach Anspruch 1 oder 2, wobei die Vielzahl von ungelabelten Bildern ohne Kalibrierung der Kameraposition aufgenommen wird.

4. Verfahren nach einem der vorangehenden Ansprüche, wobei das bestimmte Objekt ein Mensch ist.

5. Verfahren nach einem der vorangehenden Ansprüche, wobei die 3D-Pose durch 3D-Positionen von Gelenken in Bezug auf eine Kamera definiert ist.

6. Verfahren nach einem der vorangehenden Ansprüche, wobei das Modell von einer ersten Schicht des Prozesses verwendet wird, die eine Pose einer 2,5-Dimension, 2,5D, für das gegebene 2D-Bild vorhersagt.

7. Verfahren nach Anspruch 6, wobei eine zweite Schicht des Prozesses eine 3D-Rekonstruktion der 2,5D-Pose implementiert, um die 3D-Pose für das gegebene 2D-Bild zu schätzen.

8. Verfahren nach Anspruch 6 oder 7, wobei das Modell:
2D-Wärmekarten für jedes ungelabelte Bild der Vielzahl von ungelabelten Bildern erzeugt, und
latente Tiefenkarten für jedes ungelabelte Bild der Vielzahl von ungelabelten Bildern erzeugt.

9. Verfahren nach Anspruch 8, ferner umfassend:
Empfangen mindestens eines gelabelten Bilds, wobei jedes gelabelte Bild des mindestens einen gelabelten Bilds von einem entsprechenden Objekt stammt, das sich von dem bestimmten Objekt unterscheidet und mit einer 2D-Pose des entsprechenden Objekts annotiert ist,
wobei das mindestens eine gelabelte Bild zur Bestimmung des Wärmekartenverlusts während des Trainings verwendet wird.

10. Verfahren nach Anspruch 8 oder 9, wobei die erste Schicht:
die 2D-Wärmekarten vereinheitlicht, um vereinheitlichte 2D-Wärmekarten zu erzeugen.

11. Verfahren nach Anspruch 10, wobei die erste Schicht:
die vereinheitlichten 2D-Wärmekarten in 2D-Positionskoordinaten konvertiert.

12. Verfahren nach Anspruch 11, wobei die erste Schicht:
relative Tiefenwerte aus den latenten Tiefenkarten und den vereinheitlichten 2D-Wärmekarten erhält,
wobei die relativen Tiefenwerte und die 2D-Posenkoordinaten die 2,5D-Pose definieren.

13. Verfahren nach einem der Ansprüche 7 bis 12, wobei die zweite Schicht:
eine Tiefe eines Root Joints bestimmt, und
die Tiefe des Root Joints verwendet, um maßstabsvereinheitlichte 3D-Positionen von Gelenken unter Verwendung einer perspektivischen Projektion zu rekonstruieren.

14. Verfahren nach Anspruch 13, wobei vordefinierte Durchschnittslängen für einen oder mehrere Aspekte des bestimmten Objekts zur Bestimmung des Längenverlusts verwendet werden, der mit den maßstabsvereinheitlichten 3D-Positionen der Gelenke während des Trainings verbunden ist.

15. Verfahren nach Anspruch 13 oder 14, wobei die zweite Schicht:
eine starre Ausrichtung für die maßstabsvereinheitlichten 3D-Positionen der Gelenke durchführt, um die 3D-Pose zu erzeugen.

16. Verfahren nach Anspruch 15, wobei ein Multi-View-Konsistenzverlust für die 3D-Pose während des Trainings bestimmt wird, wobei der Multi-View-Konsistenzverlust erzwingt, dass die ausgehend von verschiedenen Ansichten erzeugten 3D-Posen bis hin zu einer starren Transformation gleich sein sollten.

17. Nicht-transitorisches computerlesbares Medium, das Computeranweisungen speichert, die dann, wenn sie von einem oder mehreren Prozessoren ausgeführt werden, den einen oder die mehreren Prozessoren veranlassen, ein Verfahren durchzuführen, das die Schritte eines Verfahrens nach einem der Ansprüche 1 bis 16 umfasst.

18. System, umfassend:
einen Speicher, der Computeranweisungen speichert; und
einen Prozessor, der die Computeranweisungen ausführt, um ein Verfahren nach einem der Ansprüche 1 bis 16 durchzuführen

19. System nach Anspruch 18, wobei die Vielzahl von ungelabelten Bildern ohne Kalibrierung der Kameraposition aufgenommen wird.

20. System nach Anspruch 18, wobei die Vielzahl von ungelabelten Bildern Bilder ohne 3D-Posenannotationen enthält.

Es folgen 8 Seiten Zeichnungen

Anhängende Zeichnungen

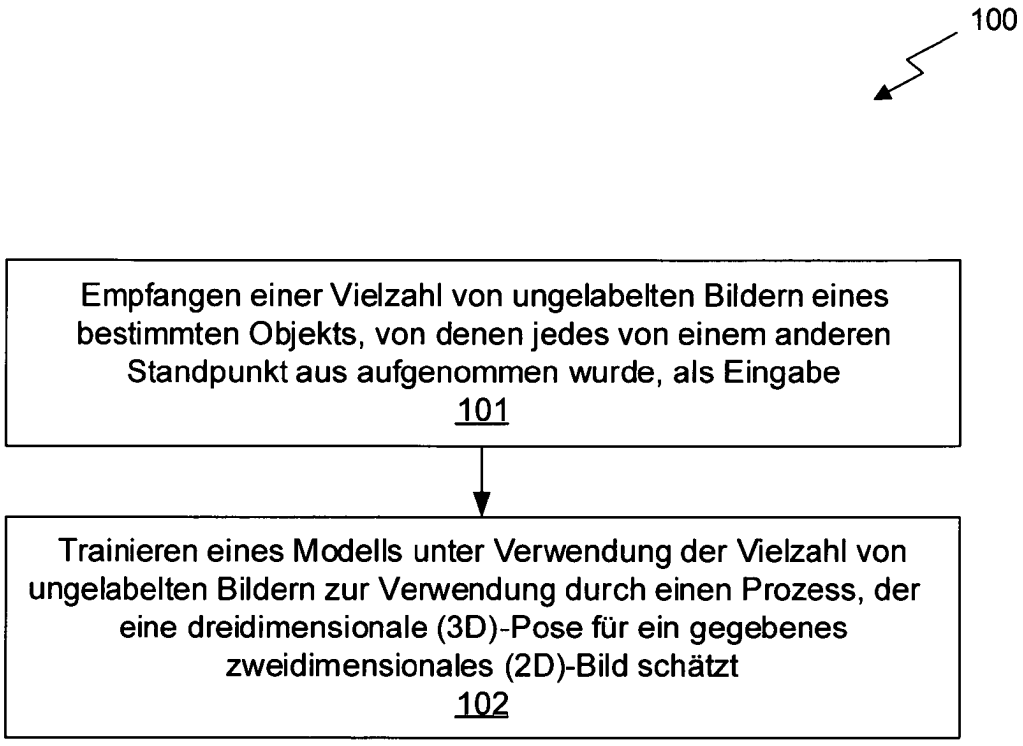


Fig. 1

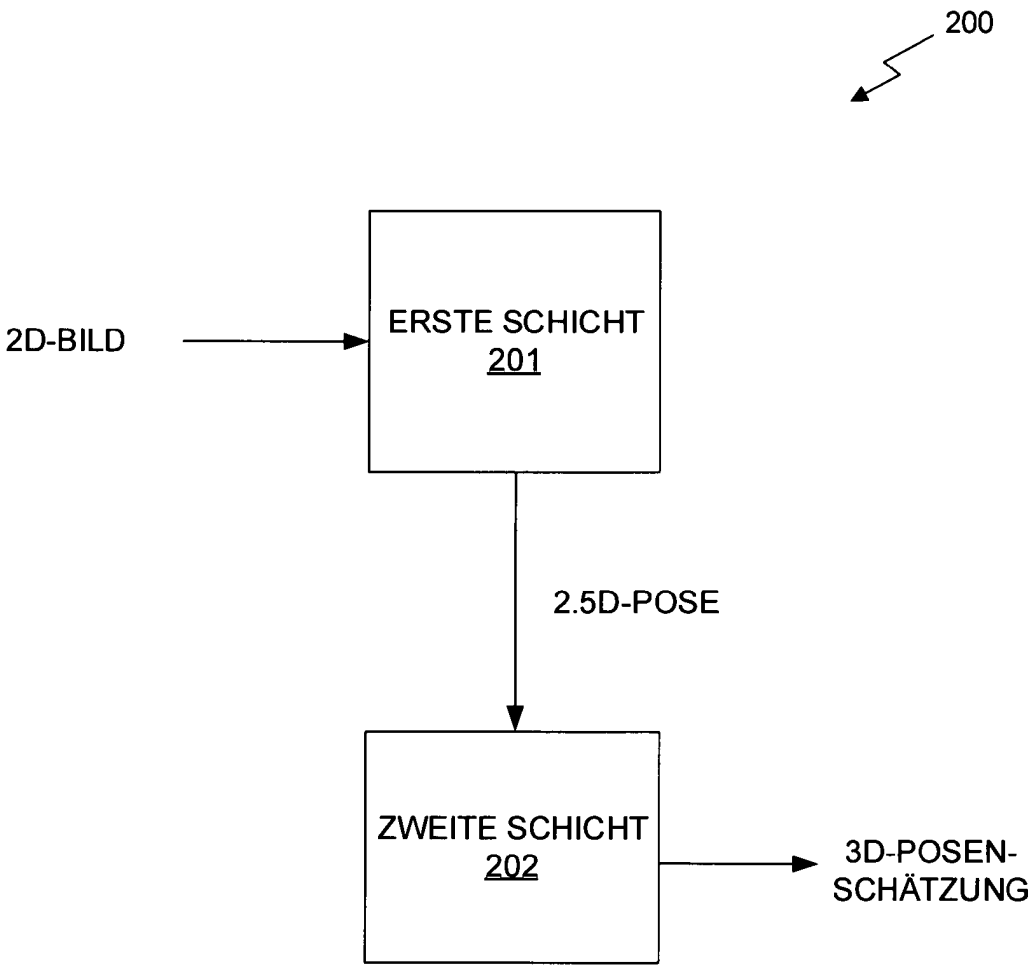

200
2D-BILD
ERSTE SCHICHT
201
2.5D-POSE
ZWEITE SCHICHT
202
3D-POSEN-
SCHÄTZUNG

Fig. 2

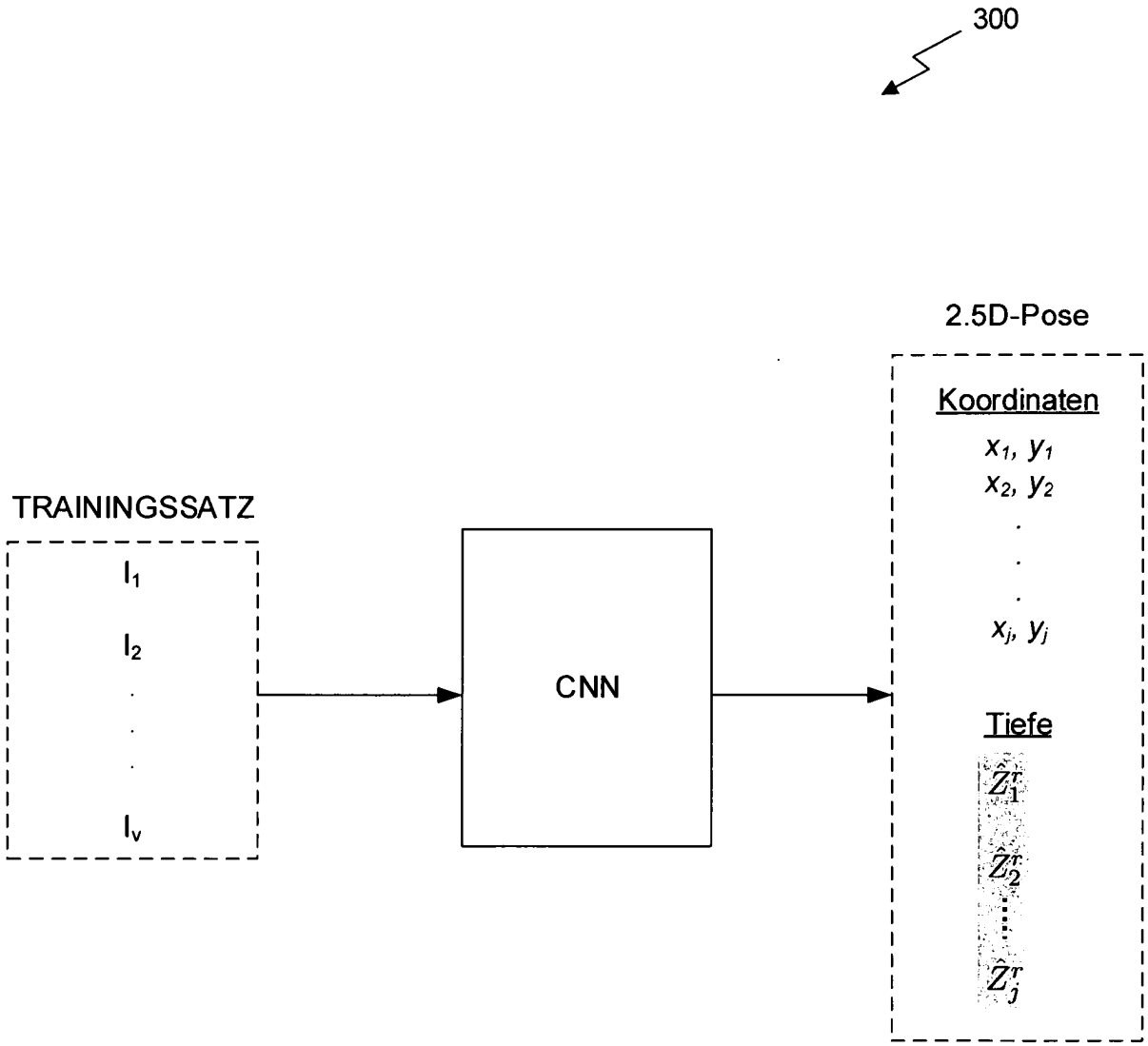

300
2.5D-Pose
Koordinaten
x_1, y_1
x_2, y_2
x_j, y_j
Tiefe
$\hat{Z}_1^r$
$\hat{Z}_2^r$
$\hat{Z}_j^r$
TRAININGSSATZ
I_1
I_2
I_v
CNN

Fig. 3A

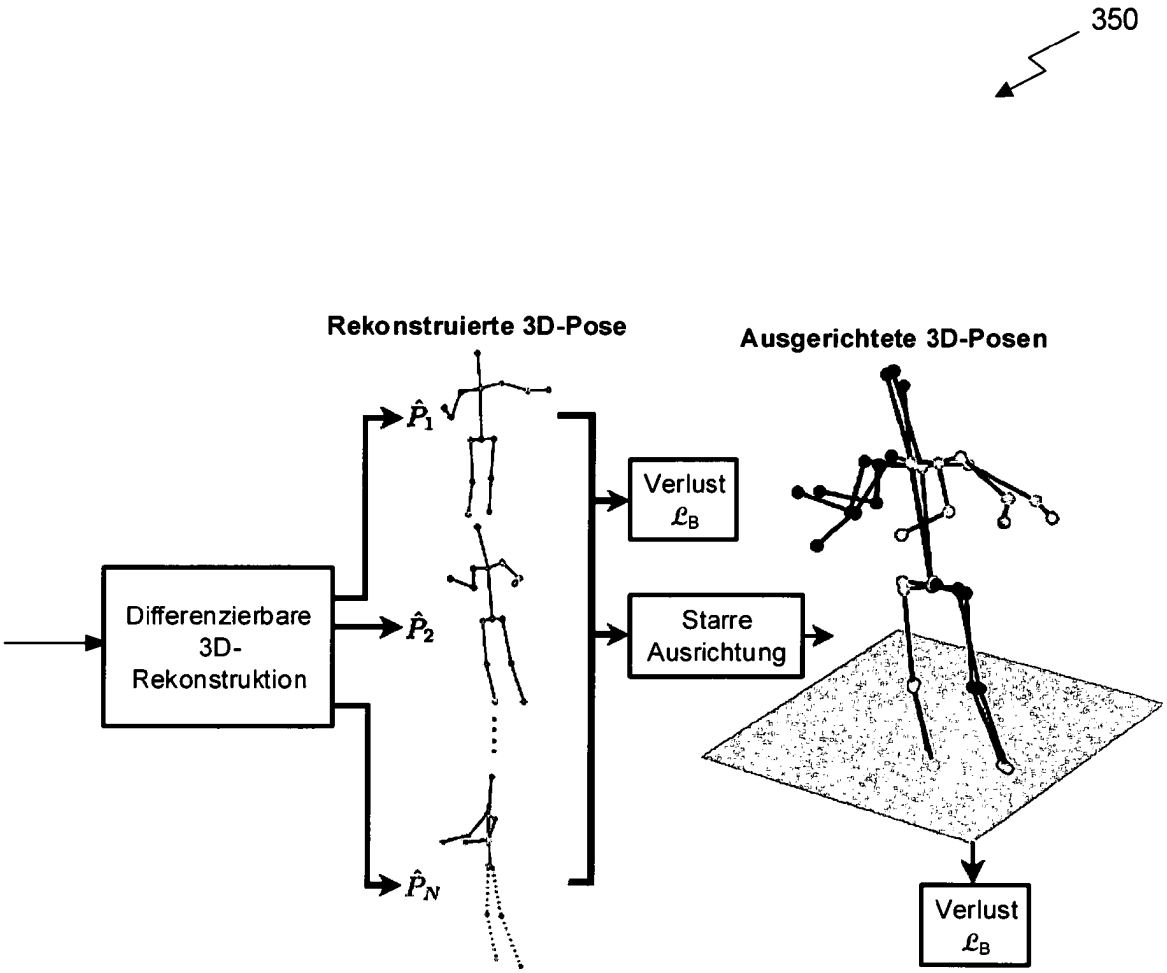

350
Rekonstruierte 3D-Pose
Ausgerichtete 3D-Posen
Differenzierbare 3D-Rekonstruktion
$\hat{P}_1$
$\hat{P}_2$
$\hat{P}_N$
Verlust $\mathcal{L}_B$
Starre Ausrichtung
Verlust $\mathcal{L}_B$

Fig. 3B

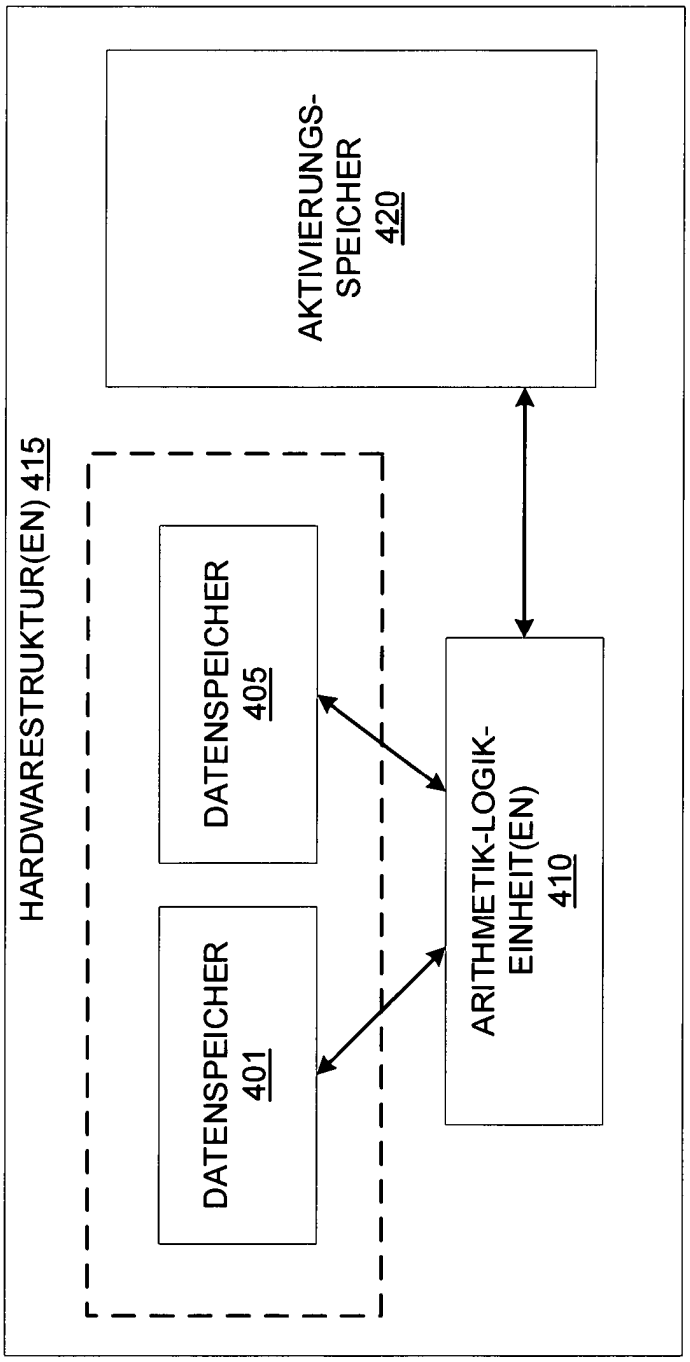

AKTIVIERUNGS-SPEICHER 420
HARDWARESTRUKTUR(EN) 415
DATENSPEICHER 405
DATENSPEICHER 401
ARITHMETIK-LOGIK-EINHEIT(EN) 410

Fig. 4A

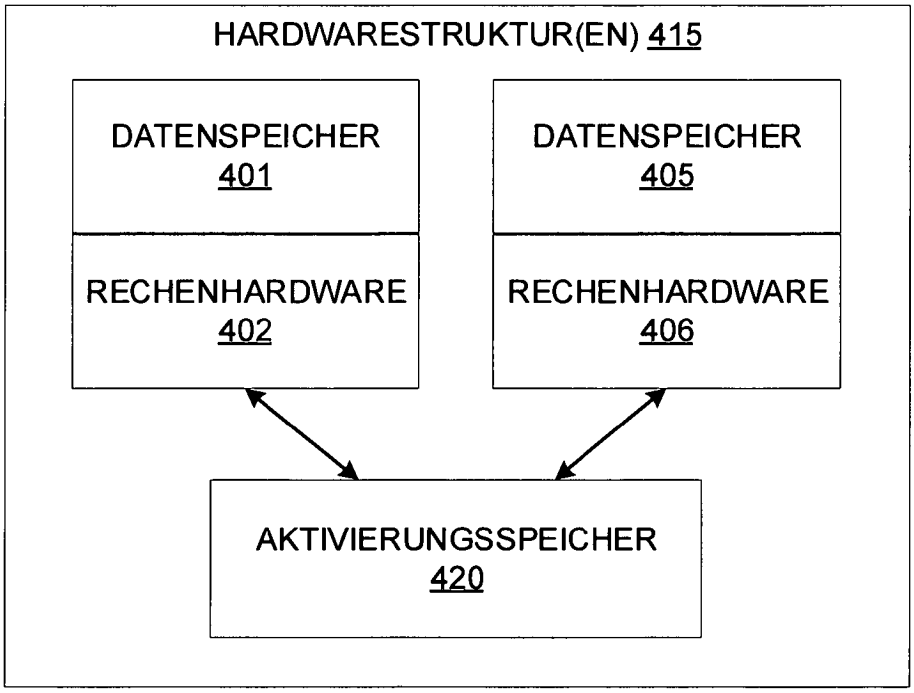

HARDWARESTRUKTUR(EN) 415
DATENSPEICHER 401
DATENSPEICHER 405
RECHENHARDWARE 402
RECHENHARDWARE 406
AKTIVIERUNGSSPEICHER 420

Fig. 4B

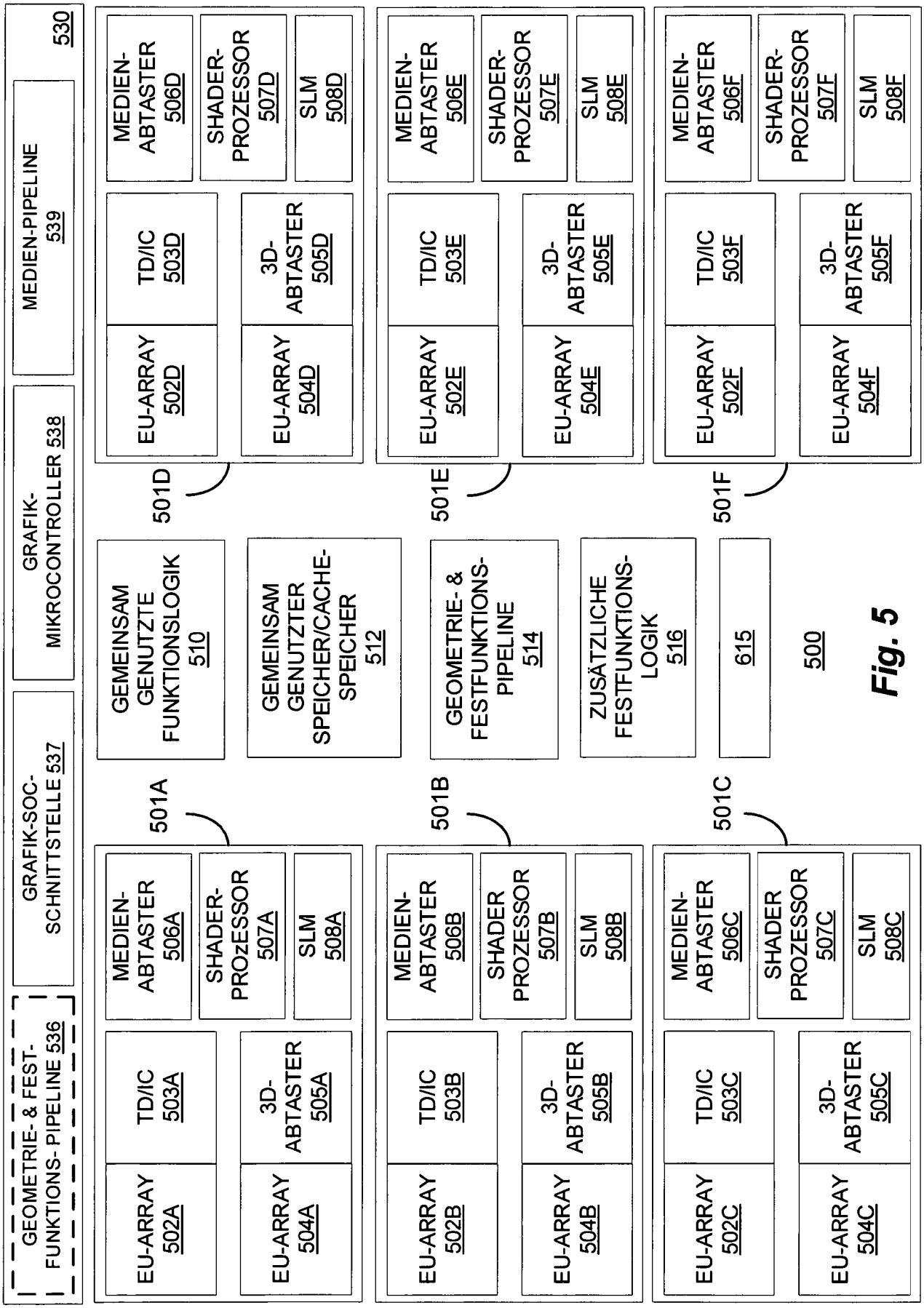

GEOMETRIE- & FEST-FUNKTIONS- PIPELINE 536
GRAFIK-SOC-SCHNITTSTELLE 537
GRAFIK-MIKROCONTROLLER 538
MEDIEN-PIPELINE 539
530
501A
501B
501C
501D
501E
501F
EU-ARRAY 502A
TD/IC 503A
MEDIEN-ABTASTER 506A
SHADER-PROZESSOR 507A
SLM 508A
EU-ARRAY 504A
3D-ABTASTER 505A
EU-ARRAY 502B
TD/IC 503B
MEDIEN-ABTASTER 506B
SHADER PROZESSOR 507B
SLM 508B
EU-ARRAY 504B
3D-ABTASTER 505B
EU-ARRAY 502C
TD/IC 503C
MEDIEN-ABTASTER 506C
SHADER PROZESSOR 507C
SLM 508C
EU-ARRAY 504C
3D-ABTASTER 505C
EU-ARRAY 502D
TD/IC 503D
MEDIEN-ABTASTER 506D
SHADER-PROZESSOR 507D
SLM 508D
EU-ARRAY 504D
3D-ABTASTER 505D
EU-ARRAY 502E
TD/IC 503E
MEDIEN-ABTASTER 506E
SHADER-PROZESSOR 507E
SLM 508E
EU-ARRAY 504E
3D-ABTASTER 505E
EU-ARRAY 502F
TD/IC 503F
MEDIEN-ABTASTER 506F
SHADER-PROZESSOR 507F
SLM 508F
EU-ARRAY 504F
3D-ABTASTER 505F
GEMEINSAM GENUTZTE FUNKTIONSLOGIK 510
GEMEINSAM GENUTZTER SPEICHER/CACHE-SPEICHER 512
GEOMETRIE- & FESTFUNKTIONS-PIPELINE 514
ZUSÄTZLICHE FESTFUNKTIONS-LOGIK 516
615
500

Fig. 5

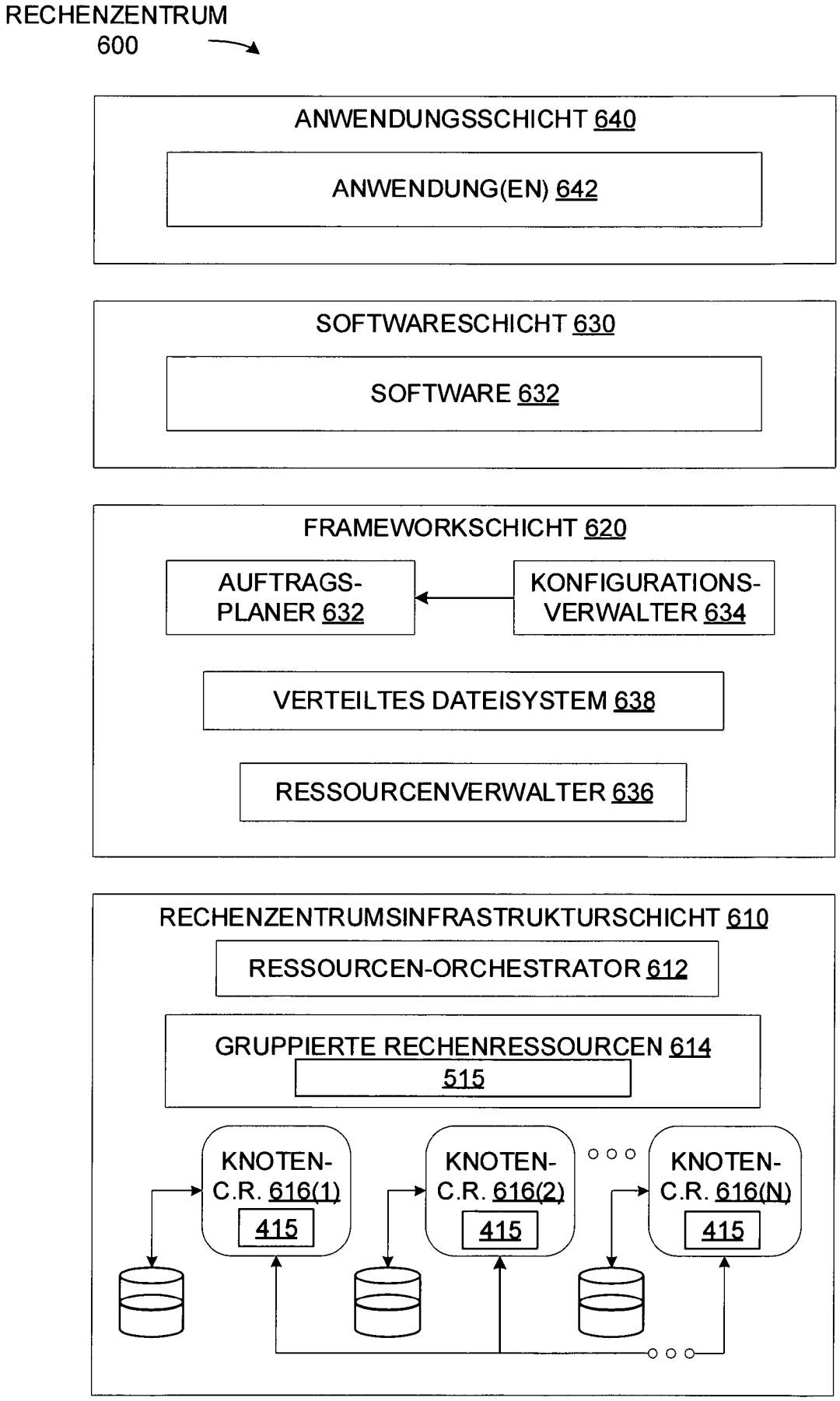


Fig. 6

