(19) Deutsches Patent- und Markenamt

(10) **DE 10 2020 208 211 A1** 2022.01.05

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2020 208 211.1**
(22) Anmeldetag: **01.07.2020**
(43) Offenlegungstag: **05.01.2022**

(51) Int Cl.: ***G06V 30/20*** (2022.01)

(71) Anmelder:
Robert Bosch Gesellschaft mit beschränkter Haftung, 70469 Stuttgart, DE

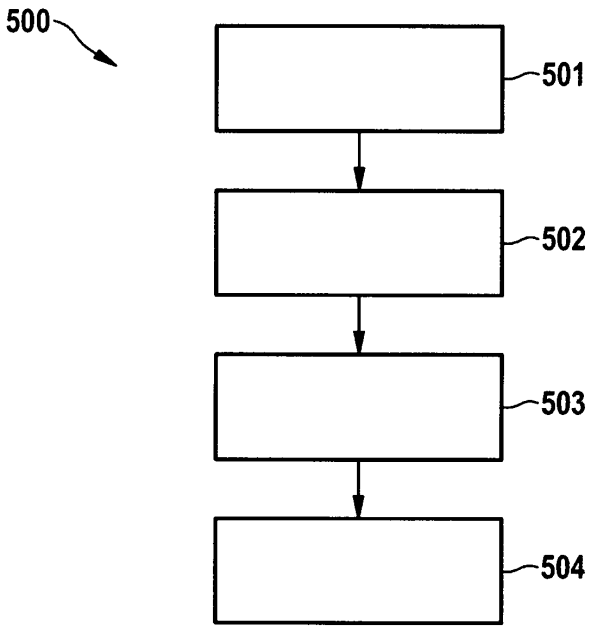
(72) Erfinder:
Spies, Markus, 76137 Karlsruhe, DE; Kupcsik, Andras Gabor, 71034 Böblingen, DE

Rechercheantrag gemäß § 43 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Vorrichtungen und Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts**

(57) Zusammenfassung: Gemäß verschiedenen Ausführungsformen ist ein Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts beschrieben, das umfasst: Erhalten eines 3D-Modells des Objekts, wobei das 3D-Modell ein Gitter aus Vertices umfasst, die durch Kanten verbunden sind, wobei jede Kante ein Gewicht aufweist, das die Nähe von zwei durch die Kante verbundenen Vertices im Objekt spezifiziert, Bestimmen eines Deskriptors für jeden Vertex des Gitters durch Suchen von Deskriptoren für die Vertices, die die Summe, über Paare verbundener Vertices, von Distanzen zwischen den Deskriptoren des Paares von Vertices minimieren, gewichtet durch das Gewicht der Kante zwischen dem Paar von Vertices, Erzeugen von Trainingsdatenbildpaaren, wobei jedes Trainingsdatenbildpaar ein das Objekt zeigendes Trainingseingabebild und ein Zielbild umfasst, und wobei das Erzeugen des Zielbildes Bestimmen der Vertexposition von Vertices des Objektmodells des Objekts, die die Vertices in dem Trainingseingabebild aufweisen, und Zuweisen, für jede bestimmte Vertexposition in dem Trainingseingabebild, des für den Vertex an der Vertexposition bestimmten Deskriptors zu der Position in dem Zielbild und Trainieren des maschinellen Lernmodells durch überwachtes Lernen unter Verwendung des Trainingsdatenbildpaares als Trainingsdaten umfasst.



Beschreibung

[0001] Die vorliegende Offenbarung betrifft Vorrichtungen und Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts.

[0002] Um eine flexible Herstellung oder Bearbeitung von Objekten durch einen Roboter zu ermöglichen, ist es wünschenswert, dass der Roboter in der Lage ist, ein Objekt ungeachtet der Stellung, mit der das Objekt in den Arbeitsraum des Roboters platziert wird, zu handhaben. Daher sollte der Roboter fähig sein, zu erkennen, welche Teil des Objektes sich an welchen Positionen befinden, sodass er zum Beispiel das Objekt an der korrekten Stelle greifen kann, um es z. B. an einem anderen Objekt zu befestigen, oder das Objekt am aktuellen Ort zu schweißen. Dies bedeutet, dass der Roboter fähig sein sollte, die Topologie (Oberfläche) des Objekts zum Beispiel aus einem oder mehreren Bildern, die durch eine am Roboter befestigte Kamera aufgenommen werden, zu erkennen. Ein Ansatz zum Erzielen davon besteht darin, Deskriptoren, d. h. Punkte (Vektoren) in einem vordefinierten Deskriptorraum, für Teile des Objekts (d. h. in einer Bildebene repräsentierte Pixel des Objekts) zu bestimmen, wobei der Roboter trainiert ist, unabhängig von einer aktuellen Stellung des Objekts den gleichen Teilen eines Objekts die gleichen Deskriptoren zuzuweisen.

[0003] In der Veröffentlichung „Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation“ von Peter Florence et al., im Folgenden als Referenz 1 bezeichnet, werden dichte Objektnetze beschrieben, die Modelle für selbstüberwachtes dichtes Deskriptorlernen sind.

[0004] Die Wirksamkeit des Ansatzes der Referenz 1 ist jedoch stark auf den gesammelten Trainingsdaten und die beteiligten Sensoren angewiesen, und den aus dem selbstüberwachten Lernen bestimmten Deskriptoren mangelt es häufig an Erklärbarkeit. Dementsprechend sind Ansätze wünschenswert, die in dieser Hinsicht das Training zum Erkennen von Objekttopologie verbessern.

[0005] Gemäß verschiedenen Ausführungsformen ist ein Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts bereitgestellt, das umfasst: Erhalten eines 3D-Modells des Objekts, wobei das 3D-Modell ein Gitter aus Vertices umfasst, die durch Kanten verbunden sind, wobei jede Kante ein Gewicht aufweist, das die Nähe von zwei durch die Kante verbundenen Vertices im Objekt spezifiziert, Bestimmen eines Deskriptors für jeden Vertex des Gitters durch Suchen von Deskriptoren für die Vertices, die die Summe, über Paare verbundener Vertices, von Distanzen zwischen den Deskriptoren des Paares von Vertices minimieren, gewichtet durch das Gewicht der Kante zwischen dem Paar von Vertices, Erzeugen von Trainingsdatenbildpaaren, wobei jedes Trainingsdatenbildpaar ein das Objekt zeigendes Trainingseingabebild und ein Zielbild umfasst, und wobei das Erzeugen des Zielbildes Bestimmen der Vertexposition von Vertices des Objektmodells des Objekts, die die Vertices in dem Trainingseingabebild aufweisen, und Zuweisen, für jede bestimmte Vertexposition in dem Trainingseingabebild, des für den Vertex an der Vertexposition bestimmten Deskriptors zu der Position in dem Zielbild und Trainieren des maschinellen Lernmodells durch überwachtes Lernen unter Verwendung des Trainingsdatenbildpaares als Trainingsdaten umfasst.

[0006] Das oben beschriebene Trainingsverfahren ermöglicht das Trainieren eines maschinellen Lernmodells (wie etwa neuronalen Netzes), um eine genauere Vorhersage (d. h. Deskriptorbestimmung) im Vergleich zur Verwendung von selbstüberwachtem Lernen durchzuführen (d. h. ermöglicht eine größere Vielfalt von Anwendungen des Netzes). Ferner stellt es größere Flexibilität für das Anpassen des maschinellen Lernmodells bereit, sodass es bei verschiedenen Problemen angewendet werden kann, reduziert die Trainingsdatenanforderungen (z. B. die Menge an erforderlichen Trainingsdaten) und ruft ein erläuterbares maschinelles Lernhilfsmittel hervor.

[0007] Im Folgenden sind verschiedene Beispiele angegeben.

[0008] Beispiel 1 ist das Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts, wie oben beschrieben.

[0009] Beispiel 2 ist das Verfahren des Beispiels 1, wobei das Erzeugen der Trainingsdatenbildpaare Erhalten mehrerer Bilder des Objekts mit unterschiedlichen Stellungen und Erzeugen eines Trainingsdatenbildpaares von jedem erhaltenen Bild durch Erzeugen eines Zielbildes für das erhaltene Bild umfasst.

[0010] Dies ermöglicht das Trainieren des maschinellen Lernmodells (z. B. eines Roboters mit einer Robotersteuerung, die das maschinelle Lernmodell implementiert), um die Topologie eines Objekts ungeachtet der Stellung des Objekts z. B. im Arbeitsraum des Roboters zu erkennen.

[0011] Beispiel 3 ist das Verfahren des Beispiels 1 oder 2, umfassend Bestimmen der Vertexpositionen von Vertices des Objektmodells des Objekts, die die Vertices in den Trainingseingabebildern aufweisen, aus den jeweiligen Stellungen, die das Objekt in den Trainingseingabebildern aufweist (z. B. in dem Kamerakoordinatensystem).

[0012] Dies ermöglicht eine genaue Bestimmung der Vertexpositionen, was im Gegenzug genaue Zielbilder für überwachtes Training ermöglicht.

[0013] Beispiel 4 ist das Verfahren eines der Beispiele 1 bis 3, wobei das Suchen der Deskriptoren Bestimmen von Eigenvektoren der Laplace-Matrix des durch die Vertices und die Kanten des 3D-Modells gebildeten Graphen und Annehmen von Komponenten der Eigenvektoren als Komponenten der Deskriptoren umfasst.

[0014] Dies ermöglicht eine effiziente Bestimmung von (fast) optimalen Deskriptoren. Beispielsweise ist jeder Deskriptor ein Vektor mit der Dimension d und weist somit d Komponenten auf (z. B. ist (3, 4) ein Vektor mit Dimension 2 und weist Komponenten 3 und 4 auf).

[0015] Beispiel 5 ist das Verfahren des Beispiels 4, umfassend Assoziieren jedes Vertex, der mit einer Komponentenposition assoziiert ist, in den Eigenvektoren, und Annehmen, für jeden Vertex, der Komponenten in den Eigenvektoren an der mit dem Vertex assoziierten Komponentenposition als Komponenten des Deskriptors für den Vertex.

[0016] Insbesondere kann die Deskriptorraumdimension flexibel durch eine Auswahl einer Anzahl von Eigenvektorkomponenten für einen Deskriptor entsprechend einer gewünschten Deskriptorraumdimension gewählt werden.

[0017] Beispiel 6 ist das Verfahren des Beispiels 5, wobei das Bestimmen der Deskriptoren ferner Komprimieren der Deskriptoren durch Kombinieren von Komponenten von Eigenvektoren, deren Eigenwerte sich um weniger als eine vorbestimmte Schwelle unterscheiden, zu einer einzelnen Komponente umfasst.

[0018] Dies ermöglicht eine Reduktion der Deskriptorraumdimension, wenn das Objekt symmetrisch ist. Mit anderen Worten kann eine unnötige Unterscheidung zwischen symmetrischen Teilen des Objekts vermieden werden.

[0019] Beispiel 7 ist das Verfahren eines der Beispiele 1 bis 6, wobei das Erhalten des 3D-Modells des Objekts umfasst: Erhalten eines 3D-Gitters von Vertices und Kanten, die das Objekt modellieren, und Zuweisen, als Gewichte für die Kanten, der Kantengewichte eines Laplace-Beltrami-Operators, der an dem Gitter angewendet wird.

[0020] Die Verwendung dieses Ansatzes ermöglicht die Berücksichtigung der Geometrie des Modells, wenn die Nähe zweier Vertices bestimmt wird, indem die geodätische Distanz auf dem Objekt anstatt die euklidische Metrik in einem umliegenden Raum genommen wird, die ungenau sein kann, wenn das Objekt gekrümmt ist.

[0021] Beispiel 8 ist ein Verfahren zum Steuern eines Roboters, das umfasst: Trainieren eines maschinellen Lernmodells gemäß einem der Beispiele 1 bis 7, Erhalten eines das Objekt zeigenden Bildes, Einspeisen des Bildes in das maschinelle Lernmodell, Bestimmen einer Stellung des Objekts aus der Ausgabe des maschinellen Lernmodells und Steuern des Roboters in Abhängigkeit von der bestimmten Stellung des Objekts.

[0022] Beispiel 9 ist das Verfahren des Beispiels 8, wobei das Bestimmen der Objektstellung Bestimmen der Position eines gewissen Teils des Objekts umfasst, und wobei das Steuern des Roboters in Abhängigkeit von der bestimmten Stellung des Objekts Steuern eines Endeffektors des Roboters, sich zu der Position des Teils des Objekts zu bewegen und mit dem Teil des Objekts zu interagieren, umfasst.

[0023] Beispiel 10 ist ein Software- oder Hardware-Agent, insbesondere ein Roboter, der eine Kamera, die zum Bereitstellen von Bilddaten eines Objekts ausgelegt ist, eine Steuerung, die zum Implementieren eines

maschinellen Lernmodells ausgelegt ist, und eine Trainingsvorrichtung, die zum Trainieren des maschinellen Lernmodells durch das Verfahren eines der Beispiele 1 bis 7 ausgelegt ist, umfasst.

[0024] Beispiel 11 ist eine Software- oder Hardware-Agent nach Beispiel 10, der mindestens einen Aktor umfasst, wobei die Steuerung zum Steuern des mindestens einen Aktors unter Verwendung einer Ausgabe aus dem maschinellen Lernmodell ausgelegt ist.

[0025] Beispiel 12 ist ein Computerprogramm, das Anweisungen umfasst, die bei Ausführung durch einen Prozessor veranlassen, dass der Prozessor ein Verfahren nach einem der Beispiele 1 bis 9 durchführt.

[0026] Beispiel 13 ist ein computerlesbares Medium, das Anweisungen speichert, die bei Ausführung durch einen Prozessor veranlassen, dass der Prozessor ein Verfahren nach einem der Beispiele 1 bis 9 durchführt.

[0027] In den Zeichnungen beziehen sich im Allgemeinen überall in den verschiedenen Ansichten ähnliche Bezugszeichen auf die gleichen Teile. Die Zeichnungen sind nicht notwendigerweise maßstabsgetreu, stattdessen wird der Schwerpunkt allgemein auf die Veranschaulichung der Prinzipien der Erfindung gelegt. In der folgenden Beschreibung werden verschiedene Aspekte unter Bezugnahme auf die folgenden Zeichnungen beschrieben, wobei gilt:

Fig. 1 zeigt einen Roboter.

Fig. 2 veranschaulicht das Trainieren eines neuronalen Netzes gemäß einer Ausführungsform.

Fig. 3 zeigt eine beispielhafte Einbettung eines 4-Knoten-Graphen zur Deskriptorbestimmung.

Fig. 4 veranschaulicht die Definition von Winkeln an Vertices eines 3D-Modells zur Kantengewichtsbestimmung gemäß dem Laplace-Beltrami-Operator.

Fig. 5 zeigt ein Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts gemäß einer Ausführungsform.

[0028] Die folgende ausführliche Beschreibung bezieht sich auf die begleitenden Zeichnungen, die veranschaulichend spezifische Einzelheiten und Aspekte dieser Offenbarung zeigen, in der die Erfindung umgesetzt werden kann. Andere Aspekte können genutzt werden und strukturelle, logische und elektrische Änderungen können vorgenommen werden, ohne vom Schutzumfang der Erfindung abzuweichen. Die verschiedenen Aspekte dieser Offenbarung schließen sich nicht notwendigerweise gegenseitig aus, da manche Aspekte dieser Offenbarung mit einem oder mehreren anderen Aspekten dieser Offenbarung kombiniert werden können, um neue Aspekte zu bilden.

[0029] Im Folgenden werden verschiedene Beispiele ausführlicher beschrieben.

[0030] **Fig. 1** zeigt einen Roboter 100.

[0031] Der Roboter 100 beinhaltet einen Roboterarm 101, zum Beispiel einen Industrieroboterarm zum Handhaben oder Montieren eines Arbeitsstücks (oder eines oder mehrerer anderer Objekte). Der Roboterarm 101 beinhaltet Manipulatoren 102, 103, 104 und eine Basis (oder Stütze) 105, mittels der die Manipulatoren 102, 103, 104 gestützt werden. Der Ausdruck „Manipulator“ bezieht sich auf die bewegbaren Bauteile des Roboterarms 101, deren Betätigung eine physische Interaktion mit der Umgebung ermöglicht, um z. B. eine Aufgabe auszuführen. Zur Steuerung beinhaltet der Roboter 100 eine (Roboter-) Steuerung 106, die zum Implementieren der Interaktion mit der Umgebung gemäß einem Steuerprogramm ausgelegt ist. Das letzte Bauteil 104 (das am weitesten von der Stütze 105 entfernt ist) der Manipulatoren 102, 103, 104 wird auch als der Endeffektor 104 bezeichnet und kann ein oder mehrere Werkzeuge beinhalten, wie etwa einen Schweißbrenner, ein Greifinstrument, ein Lackiergerät oder dergleichen.

[0032] Die anderen Manipulatoren 102, 103 (die sich näher an der Stütze 105 befinden) können eine Positionierungsvorrichtung bilden, sodass, zusammen mit dem Endeffektor 104, der Roboterarm 101 mit dem Endeffektor 104 an seinem Ende bereitgestellt ist. Der Roboterarm 101 ist ein mechanischer Arm, der ähnliche Funktionen wie ein menschlicher Arm bereitstellen kann (möglicherweise mit einem Werkzeug an seinem Ende).

[0033] Der Roboterarm 101 kann Gelenkelemente 107, 108, 109 beinhalten, die die Manipulatoren 102, 103, 104 miteinander und mit der Stütze 105 verbinden. Ein Gelenkelement 107, 108, 109 kann ein oder mehrere Gelenke aufweisen, die jeweils eine drehbare Bewegung (d. h. Drehbewegung) und/oder translatorische

Bewegung (d. h. Verlagerung) für assoziierte Manipulatoren relativ zueinander bereitstellen können. Die Bewegung der Manipulatoren 102, 103, 104 kann mittels Aktoren initiiert werden, die durch die Steuerung 106 gesteuert werden.

[0034] Der Ausdruck „Aktor“ kann als eine Komponente verstanden werden, die als Reaktion auf ihren Antrieb zum Bewirken eines Mechanismus oder Prozesses ausgebildet ist. Der Aktor kann durch die Steuerung 106 erstellte Anweisungen (die sogenannte Aktivierung) in mechanische Bewegungen implementieren. Der Aktor, z. B. ein elektromechanischer Wandler, kann dazu ausgelegt sein, als Reaktion auf seinen Antrieb elektrische Energie in mechanische Energie umzuwandeln.

[0035] Der Ausdruck „Steuerung“ kann als ein beliebiger Typ von logikimplementierender Entität verstanden werden, die zum Beispiel eine Schaltung und/oder einen Prozessor beinhalten kann, die/der in der Lage ist, in einem Speicherungsmedium gespeicherte Software, Firmware oder eine Kombination davon auszuführen, und die/der Anweisungen, z. B. zu einem Aktor im vorliegenden Beispiel, ausstellen kann. Die Steuerung kann zum Beispiel durch Programmcode (z. B. Software) konfiguriert werden, um den Betrieb eines Systems, eines Roboters im vorliegenden Beispiel, zu steuern.

[0036] Im vorliegenden Beispiel beinhaltet die Steuerung 106 einen oder mehrere Prozessoren 110 und einen Speicher 111, der Code und Daten speichert, basierend auf denen der Prozessor 110 den Roboterarm 101 steuert. Gemäß verschiedenen Ausführungsformen steuert die Steuerung 106 den Roboterarm 101 auf Basis eines maschinellen Lernmodells 112, das im Speicher 111 gespeichert ist.

[0037] Gemäß verschiedenen Ausführungsformen ist das maschinelle Lernmodell 112 dazu ausgelegt und trainiert, dem Roboter 100 zu ermöglichen, ein Objekt 113 zu erkennen, das zum Beispiel in einen Arbeitsraum des Roboterarms 101 platziert wird. Beispielsweise kann der Roboter entscheiden, was mit dem Objekt 113 durchzuführen ist, in Abhängigkeit davon, was für ein Objekt es ist (z. B. der Objekttyp), oder kann auch erkennen und entscheiden, welches Teil des Objekts 113 durch den Endeffektor 109 gegriffen werden sollte. Der Roboter 100 kann zum Beispiel mit einer oder mehreren Kameras 114 ausgestattet sein, die es ihm ermöglichen, Bilder seines Arbeitsraums aufzunehmen. Die Kamera 114 ist zum Beispiel an dem Roboterarm 101 befestigt, sodass der Roboter Bilder des Objekts 113 von verschiedenen Perspektiven aus aufnehmen kann, indem er den Roboterarm 101 herumbewegt.

[0038] Ein Beispiel für ein maschinelles Lernmodell 112 zur Objekterkennung ist ein dichtes Objektnetz. Ein dichtes Objektnetz bildet ein Bild (z. B. ein durch die Kamera 114 bereitgestelltes RGB-Bild) auf ein beliebiges dimensionales (Dimension D) Deskriptorraumbild ab, wie in Referenz 1 beschrieben.

[0039] Das dichte Objektnetz ist ein neuronales Netz, das trainiert ist, ein Deskriptorraumbild für ein Eingabebild eines Bildes unter Verwendung von selbstüberwachtem Lernen zu geben. Die Wirksamkeit dieses Ansatzes ist jedoch stark auf die gesammelten Trainingsdaten und die beteiligten Sensoren (z. B. die Kameras 114) angewiesen Ferner kann die Interpretation der Netzwerkvorhersage schwierig sein.

[0040] Gemäß verschiedenen Ausführungsformen ist ein Ansatz zum Erkennen eines Objekts und seiner Stellung bereitgestellt, unter der Annahme, dass ein 3D-Modell (z. B. ein CAD(Computer Aided Design)-Modell) des Objekts bekannt ist, was typischerweise für Industriemontage- oder -bearbeitungsaufgaben der Fall ist. Gemäß verschiedenen Ausführungsformen werden nichtlineare Dimensionalitätsreduktionstechniken verwendet, um optimale Zielbilder zum Trainieren von Eingabebildern für ein neuronales Netz zu berechnen. Anstatt somit selbstüberwachtes Training eines neuronalen Netzes zu verwenden, wird überwachtes Training eines neuronalen Netzes verwendet.

[0041] Gemäß einer Ausführungsform wird zum Erzeugen von Trainingsdaten zum Trainieren des maschinellen Lernmodells 112 zunächst eine Datensammlung durchgeführt. Insbesondere werden zum Beispiel registrierte RGB(Rot-Grün-Blau)-Bilder gesammelt. Ein registriertes Bild bedeutet hier ein RGB-Bild mit bekannten intrinsischen und extrinsischen Kamerawerten. In einem Szenario in der realen Welt wird eine an einem Roboter befestigte Kamera 114 (z. B. eine an einem Roboterhandgelenk befestigte Kamera) zum Beispiel verwendet, um ein Objekt zu scannen, während sich der Roboter (z. B. Roboterarm 101) herumbewegt. Andere extrinsische Schätztechniken, z. B. ChArUco-Marker, können verwendet werden, d. h. ein Objekt kann an verschiedene Positionen und Stellungen relativ zu einem ChArUco-Board platziert werden und Bilder der Anordnung (von ChArUco-Board und Objekt) werden aufgenommen. In einem simulierten Szenario werden fotorealistisch erzeugte RGB-Bilder unter Verwendung bekannter Objektstellungen verwendet.

[0042] Nach dem Sammeln der RGB-Bilder werden für überwachtes Training eines neuronalen Netzes Zielbilder für die RGB-Bilder gerendert.

[0043] Es wird angenommen, dass die Stellung jedes Objekts in Weltkoordinaten in jedem gesammelten RGB-Bild bekannt ist. Dies ist für ein simuliertes Szenario unkompliziert, erfordert aber eine manuelle Abstimmung für ein Szenario in der realen Welt, z. B. Platzieren des Objekts an vordefinierten Positionen.

[0044] Mit diesen Informationen und unter Verwendung einer Vertexdeskriptorberechnungstechnik, wie zum Beispiel nachstehend beschrieben, für jedes RGB-Bild (d. h. Trainingseingabebild) wird ein Deskriptorbild (d. h. Trainingsausgabebild, auch als Zielbild oder Ground-Truth-Bild bezeichnet) gerendert.

[0045] Wenn für jedes RGB-Bild ein Zielbild erzeugt wurde, d. h. Paare von RGB-Bildern und Zielbildern gebildet wurden, können diese Paare von Trainingseingabebild und assoziiertem Zielbild als Trainingsdaten zum Trainieren eines neuronalen Netzes verwendet werden, wie in **Fig. 2** veranschaulicht.

[0046] **Fig. 2** veranschaulicht das Trainieren eines neuronalen Netzes 200 gemäß einer Ausführungsform.

[0047] Das neuronale Netz 200 ist ein voll faltendes Netz, das einen h × w × 3-Tensor (Eingabebild) auf einen h × w × D-Tensor (Ausgabebild) abbildet.

[0048] Es umfasst mehrere Stufen 204 von Faltungsschichten, gefolgt von einer Pooling-Schicht, Upsampling-Schichten 205 und Skip-Verbindungen 206, um die Ausgaben verschiedener Schichten zu kombinieren.

[0049] Für das Training empfängt das neuronale Netz 200 ein Trainingseingabebild 201 und gibt ein Ausgabebild 202 mit Pixelwerten im Deskriptorraum (z. B. Farbkomponenten gemäß Deskriptorvektorkomponenten) aus. Ein Trainingsverlust wird zwischen dem Ausgabebild 202 und dem mit dem Trainingseingabebild assoziierten Zielbild 203 berechnet. Dies kann für einen Stapel von Trainingseingabebildern stattfinden und der Trainingsverlust kann über die Trainingseingabebilder gemittelt werden und die Gewichte des neuronalen Netzes 200 werden unter Verwendung stochastischen Gradientenabstiegs unter Verwendung des Trainingsverlustes trainiert. Der zwischen dem Ausgabebild 202 und dem Zielbild 203 berechnete Trainingsverlust ist zum Beispiel eine L2-Verlustfunktion (um einen pixelweisen Least-Square-Fehler zwischen dem Zielbild 203 und dem Ausgabebild 202 zu minimieren).

[0050] Das Trainingseingabebild 201 zeigt ein Objekt und das Zielbild sowie das Ausgabebild beinhalten Vektoren im Deskriptorraum. Die Vektoren im Deskriptorraum können auf Farben abgebildet werden, sodass das Ausgabebild 202 (sowie das Zielbild 203) eine Heatmap des Objekts ähneln.

[0051] Die Vektoren im Deskriptorraum (auch als (dichte) Deskriptoren bezeichnet) sind d-dimensionale Vektoren (z. B. beträgt d 1, 2 oder 3), die jedem Pixel im jeweiligen Bild (z. B. jedem Pixel des Eingabebildes 201, unter der Annahme, dass das Eingabebild 201 und das Ausgabebild 202 die gleiche Dimension aufweisen) zugewiesen sind. Die dichten Deskriptoren codieren implizit die Oberflächentopologie des im Eingabebild 201 gezeigten Objekts, invariant gegenüber seiner Stellung oder der Kameraposition.

[0052] Wenn ein 3D-Modell des Objekts gegeben ist, ist es möglich, einen optimalen (im riemannschen Sinne) und eindeutigen Deskriptorvektor für jeden Vertex des 3D-Modells des Objekts analytisch zu bestimmen. Gemäß verschiedenen Ausführungsformen werden unter Verwendung dieser optimalen Deskriptoren (oder Schätzungen dieser Deskriptoren, die durch eine Optimierung bestimmt werden) Zielbilder für registrierte RGB-Bilder erzeugt, was zu einen voll überwachten Training des neuronalen Netzes 200 führt. Zusätzlich wird der Deskriptorraum ungeachtet der gewählten Deskriptordimension d erläuterbar und optimal.

[0053] Im Folgenden werden die 3D-Modelle als riemannsche Mannigfaltigkeiten $\mathcal{M}$ eingebettet in $\mathbb{R}^3$ angesehen, was das Berechnen von Geodäten (kürzester Pfad zwischen Vertices) hervorruft, während die Geometrie beachtet wird. Eine optimale Oberflächentopologie kann explizit codiert werden, indem das 3D-Modell in einen d-dimensionalen euklidischen Deskriptorraum eingebettet wird, sodass geodätische Distanzen zwischen benachbarten Vertices so gut wie möglich bewahrt werden. Dieser euklidische Raum wird als der Deskriptorraum angesehen, und eine Suche nach der optimalen Abbildung $f : \mathcal{M} \rightarrow \mathbb{R}^d$ wird durchgeführt. Gemäß einer Ausführungsform werden die Laplace-Operator-Berechnung für Gitter und ihre Eigenwertzerlegung zum Bestimmen (oder zumindest Schätzen) der optimalen Einbettung von Vertices im Deskriptorraum verwendet. Anstatt somit Geodätenberechnung und Abbildungsoptimierung zu trennen, werden die Deskriptoren in einem einzigen Framework durch das Berechnen des Laplace-Operators des 3D-Modells extrahiert.

[0054] Gemäß dem im Folgenden beschriebenen Ansatz wird die Einbettung eines 3D-Objektmodells im euklidischen Raum in den Deskriptorraum bestimmt, um Distanzen (z. B. geodätische Distanzen) zwischen Vertices zu bewahren.

[0055] Zur Dimensionalitätsreduktion über den Laplace-Operator soll ein Satz von Punkten $\{\boldsymbol{x}_i\}_{i=1}^{N}, \boldsymbol{x} \in \mathbb{R}^D$ Knoten in einen ungerichteten Graphen entsprechen. $\omega_{ij} \geqq 0$. soll die Verbindungsstärke oder die Nähe zwischen zwei Knoten ^{x}i und ^{x}j bezeichnen, z. B. $w_{ij} = \exp(- \| x_i - x_j \|_2)$.

[0056] Das Ziel besteht darin, eine d-dimensionale Einbettung $\{\boldsymbol{y}_i\}_{i=1}^{N}, \boldsymbol{y} \in \mathbb{R}^d$ zu finden (typischerweise gilt d < D), sodass, falls ^{x}i und ^{x}j nahe liegen, ihre Einbettung dann auch nahe sein sollte:

$$Y^* = \arg\min_{Y} \frac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{N} w_{ij} \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_2, \qquad (1)$$

wobei $Y = [y_1,...,y_N]_{d \times N}$.

[0057] Das Optimierungsproblem (1) ist äquivalent zu

$$\boldsymbol{Y}^* = \arg\min_{Y} \mathrm{Tr}\left(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^{\boldsymbol{T}}\right), \ \boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}, \qquad (2)$$

wobei $L_{N \times N}$ die positiv-semidefinite Laplace-Matrix ist. A ist die Adjazenzmatrix mit Elementen $A_{ij} = A_{ji} = w_{ij}$ und $D_{ii} = \Sigma_j A_{ij}$. Es sollte angemerkt werden, dass die optimale Lösung $^{Y}*$ eine beliebige Skalierung und Tendenz aufweisen kann. Um diese Zufälligkeit zu entfernen, kann das gewichtete zweite Moment mit $Y^T DY = I$ normalisiert werden, was eine Einheitsvarianz in verschiedene Dimensionen erzwingt. Das resultierende Optimierungsproblem wird dann

$$\begin{aligned} \boldsymbol{Y}^* &= \arg\min_{Y} \mathrm{Tr}\left(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^{\boldsymbol{T}}\right), \\ &s.t.\ \boldsymbol{Y}\boldsymbol{D}\boldsymbol{Y}^{\boldsymbol{T}} = \boldsymbol{I} \end{aligned} \qquad (3)$$

unter Verwendung beschränkter Optimierung mit Lagrange-Parametern $\lambda \in \mathbb{R}^d$

$$\begin{aligned} \mathrm{J}\left(\boldsymbol{Y}, \lambda\right) &= \mathrm{Tr}\left(\boldsymbol{Y}\ \boldsymbol{L}\boldsymbol{Y}^{T}\right) - \mathrm{Tr}\left(\mathrm{diag}(\lambda)\left(\boldsymbol{Y}\ \boldsymbol{D}\boldsymbol{Y}^{T} - \boldsymbol{I}\right)\right), \\ \frac{\partial \mathrm{J}\left(\boldsymbol{Y}, \lambda\right)}{\partial \boldsymbol{Y}} &= 2\boldsymbol{L}\boldsymbol{Y}^{T} - 2\mathrm{diag}(\lambda)\boldsymbol{D}\boldsymbol{Y}^{T}, \\ \boldsymbol{L}\boldsymbol{Y}^{T} &= \mathrm{diag}(\lambda)\boldsymbol{D}\boldsymbol{Y}^{T}, \end{aligned} \qquad (4)$$

was ein generalisiertes Eigenwertproblem ist, das durch standardmäßige Bibliotheken der linearen Algebra gelöst werden kann. Da L und D positiv-(semi)definite Matrizen sind, können die Eigenwerte als $\lambda_0 = 0 \leq \lambda_1 \leq ... \leq \lambda_N$- 1. geschrieben werden.

[0058] Ferner ist der erste Eigenvektor (erste Spalte von Y^T) gleich 1 (Vektor mit nur Einsen), was eine triviale Lösung ist, die jeden Vertex auf einen Punkt abbildet. Zusätzlich sind zwei beliebige Eigenvektoren orthogonal zueinander. Die Lösung zu dem Eigenwertproblem ergibt N Eigenwerte und entsprechende Eigenvektoren mit Dimensionalität N. In der Praxis werden jedoch nur die ersten d Eigenvektoren entsprechend den niedrigsten Eigenwerten (außer der trivialen Lösung) verwendet.

[0059] Somit ist die i-te Spalte von Y die Einbettung des Knotens i in R^d, während jede Zeile die Einbettung jedes Punktes in verschiedenen orthogonalen Dimensionen repräsentiert.

[0060] **Fig. 3** zeigt eine beispielhafte Einbettung eines 4-Knoten-Graphen.

[0061] Die Eigenwerte weisen eine wichtige Bedeutung bezüglich der Optimalität der Einbettung auf. Im Fall einer optimalen Einbettung $^{Y}*$ ist die Beschränkung $Y^*DY^{*T} = I$ erfüllt, und demzufolge gilt

$$\boldsymbol{Y}^*\boldsymbol{L}\boldsymbol{Y}^{*T} = \mathrm{diag}(\lambda), \qquad (5)$$

das heißt, die Eigenwerte entsprechen den Einbettungsfehlern in verschiedenen Dimensionen. Zur Einfachheit halber soll d = 1 sein, in welchem Fall jedes x auf Punkty = 1 abgebildet wird. In diesem Fall wird (5) vereinfacht zu:

$$\hat{\boldsymbol{y}}^{\boldsymbol{T}}\boldsymbol{L}\hat{\boldsymbol{y}} = \lambda_0,\ \hat{y} = \left[y_1,...,y_N\right]^T = 1^T. \qquad (6)$$

[0062] Da $\lambda_0 = 0$, das heißt, dass, falls alle Vertices des Objekts auf einen Punkt abgebildet sind, der Einbettungsfehler 0 beträgt, da die Distanz zwischen allen Punkten y 0 beträgt. Dies ist für praktische Zwecke nicht nützlich, und somit werden der erste Eigenwert und Eigenvektor ausgelassen. Das Verwenden von d= 2 entspricht der Abbildung jedes Punktes x auf eine Linie, und λ1 ist der entsprechende Einbettungsfehler, und so weiter. Da die Eigenvektoren orthogonal zueinander sind, wird das Erhöhen von d neue Dimensionen zu der Einbettung hinzufügen, was darauf abzielt, den Fehler in der neuen, orthogonalen Dimension zu minimieren. Der gleiche Effekt kann in (3) gesehen werden: da $\mathrm{Tr}\left(\boldsymbol{YLY}^T\right) = \sum_{i=1}^{d}\lambda_i$, kann die ursprüngliche Zielsetzung übersetzt werden, um den Einbettungsfehler in jeder Dimension zu minimieren. Ungeachtet des gewählten d ist somit der resultierende Deskriptorvektor optimal.

[0063] Im manchen Fällen sind subsequente Eigenwerte identisch, das heißt, $\lambda_i = \lambda_{i+1} = \lambda_{i+2} = ...$ (siehe das Beispiel von **Fig. 3**, bei dem die Eigenwerte für d=2 und d=3 die gleichen sind). Dies führt einige Informationen bezüglich der Symmetrie, wobei es mehrere orthogonale Dimensionen mit identischem Einbettungsfehler gibt. In der Tat ist in dem 4-Knoten-Graphen-Beispiel von **Fig. 3**, falls der Graph voll verbunden ist, die Einbettung in jede Dimension symmetrisch und sind alle Eigenwerte identisch, mit Ausnahme der trivialen Lösung.

[0064] Der obige Grapheneinbettungsansatz kann direkt bei Gittern, Punktwolken usw. angewendet werden. Beispielsweise kann der K-Nearest-Neighbour(KNN)-Algorithmus verwendet werden, um lokale Verbindungen zwischen Vertices zu bilden und die Adjazenzmatrix zu erstellen. Dieser Ansatz ist ausreichend, um den Graphen-Laplace-Operator zu erstellen und die Einbettung jedes Vertex zu berechnen. Nichtsdestotrotz baut dieser Ansatz inhärent auf euklidischer Distanzmetrik und Heuristiken auf, die nicht notwendigerweise die zugrundeliegende riemannsche Geometrie des 3D-Objektmodells beachten. Beispielsweise können manche Kanten durch das Objekt laufen, oder können nicht benachbarte Vertices des Gitters verbinden. Selbst nur einige wenige unkorrekte Einträge in der Adjazenzmatrix können in schlechter Einbettungsleistungsfähigkeit resultieren. Gemäß einer Ausführungsform, wenn mit Modellen gearbeitet wird, wird daher gewährleistet, dass geodätische Distanzen zwischen zwei beliebigen Vertices korrekt sind oder einen minimalen Approximationsfehler aufweisen.

[0065] Im Allgemeinen können Objektmodelle, wie etwa Gitter, oder Punktwolken als riemannsche Mannigfaltigkeiten eingebettet in $\mathbb{R}^3$ repräsentiert werden. Eine riemannsche Mannigfaltigkeit M mit einer sich gleichmäßig ändernden Metrik g kann als „lokal euklidisch“ angesehen werden, was die Eigenschaft lokaler Glattheit von Objekten der realen Welt erfasst. Die Generalisierung des Laplace-Operators zu riemannschen Mannigfaltigkeiten ist der Laplace-Beltrami(LB)-Operator Δ. Ähnlich dem Laplace-Operator in euklidischen Räumen ist der an einer Funktion angewendete LB-Operator die Divergenz des Gradienten der Funktion. Während der Laplace-Operator für Graphen und in euklidischen Räumen leicht zu berechnen ist (entweder aus Adjazenzinformationen oder finiten Differenzen), baut der LB-Operator in Differentialgeometrie auf Exterior Calculus (äußeres Kalkül) auf, und ist im Allgemeinen für eine Mannigfaltigkeit nicht leicht verfügbar.

[0066] Für bekannte diskrete Mannigfaltigkeiten, wie etwa Gitter, kann der LB-Operator jedoch approximiert werden. Dies bietet einen hocheffizienten und einfachen Berechnungsrahmen, wenn mit Gittern, Punktwolken usw. gearbeitet wird. Da das riemannsche Äquivalent des Laplace-Operators der Laplace-Beltrami ist, kann der oben beschriebene Einbettungsansatz direkt bei Δ angewendet werden. Die Eigenvektoren Y von Δ werden die optimale d-dimensionale euklidische Einbettung der Vertices des Gitters repräsentieren.

[0067] Δ kann für Gitter effizient wie folgt berechnet werden. Unter der Annahme, dass ein Gitter mit N Vertices V, Flächen Fund Kanten E gegeben ist. In diesem Fall ist die Größe von Δ N × N. Die i-te Zeile von Δ beschreibt die Adjazenzinformationen des i-ten Vertex zu seinen verbindenden Vertices. Es soll φ eine beliebige Funktion an dem Gitter sein. Dann wird das Anwenden des diskreten LB-Operators bei dieser Funktion auf Δ φ abgebildet. Das i-te Element dieser Funktion kann durch Folgendes beschrieben werden:

$$\left(\Delta\phi\right)_i = \frac{1}{2}\sum_{j\in E}\left(\cot\left(\theta_k^{i\,j}\right) + \cot\left(\theta_l^{j\,i}\right)\right)\left(\phi_j - \phi_i\right), \qquad (7)$$

Fig. 4 veranschaulicht die Definition von Winkeln $\theta_k^{i\,j}$ und $\theta_l^{i\,j}$.

[0068] Die Summe der Cotangens-Ausdrücke $\cot\left(\theta_k^{i\,j}\right) = \cot\left(\theta_l^{j\,i}\right)$ agiert als das Verbindungsgewicht wij. Diese in (7) auftretenden Gewichte, d. h. die Gewichte des LB-Operators, wenn bei dem Gitter angewendet, werden gemäß einer Ausführungsform als die Gewichte zum Bestimmen von D und A der Gleichung (2) verwendet.

[0069] Es sollte angemerkt werden, dass, da $\alpha > \pi/2 \Rightarrow \cot(\alpha) < 0$, negative Verbindungsgewichte wij auftreten können, besonders dann, falls ein Winkel erheblich größer ist als die anderen (nicht gute Flächen). Um dieses Problem zu überwinden, kann das Verbindungsgewicht durch Kantenumlegung (Edge Flipping) approximiert werden.

[0070] Der oben beschriebene Deskriptorerzeugungsansatz behandelt jeden Vertex eindeutig. Das heißt, jeder Vertex wird einem eindeutigen Deskriptor zugewiesen. Objekte können jedoch symmetrisch sein, und somit wird das Zuweisen eindeutiger Deskriptoren zu scheinbar identischen Vertices zu unsymmetrischen Einbettungen führen.

[0071] Um dieses Problem anzusprechen, werden gemäß verschiedenen Ausführungsformen intrinsische Symmetrien von Formen detektiert und symmetrische Einbettungen werden komprimiert, sodass symmetrische Vertices auf denselben Deskriptor abgebildet werden. Es kann gezeigt werden, dass eine Form intrinsische Symmetrie aufweist, falls die Eigenfunktionen des Laplace-Beltrami-Operators im euklidischen Raum symmetrisch erscheinen. Mit anderen Worten werden symmetrische Merkmale eines Gitters, einer Punktwolke usw. detektiert, wenn ihre geodätenbewahrende euklidische Einbettung (Deskriptorraum) euklidische Symmetrie zeigt. Eine kompakte Mannigfaltigkeit weist intrinsische Symmetrie auf, falls ein Homöomorphismus T besteht, der geodätische Distanzen zwischen jedem Vertex der Mannigfaltigkeit bewahrt.

[0072] Für die Komprimierung der symmetrischen Deskriptoren können sogenannte globale intrinsische symmetrieinvariante Funktionen (GISIFs) verwendet werden. Unter der Annahme eines globalen intrinsischen symmetrischen Homöomorphismus $T: \mathcal{M} \to \mathcal{M}$, s. t. $g(p, q) = g(T(p), T(q)) \; \forall p, q \in \mathcal{M}$, wobei g geodätische Distanz repräsentiert, und einer Funktion an der Mannigfaltigkeit ƒ, falls für jeden Punkt p auf der Mannigfaltigkeit gilt:

$$f \circ T(\boldsymbol{p}) = f\left(T(\boldsymbol{p})\right) = f(\boldsymbol{p}) \qquad (8)$$

dann ist ƒ eine GISIF. Beispielsweise soll auf einem Torus dieser Homöomorphismus eine beliebige Drehung um die z-Achse sein. Dies bedeutet, dass, falls f eine GISIF ist, sie dann gegenüber dieser Drehung invariant sein muss.

[0073] Ferner kann gezeigt werden, dass im Fall identischer Eigenwerte $\lambda_i = \lambda_{i+1} = \ldots = \lambda_{i+N}$, eine derartige GISIF die quadrierte Summe der Eigenvektorkomponenten des Punktes ist, d. h.

$$f(\boldsymbol{p}) = \Sigma_i^{i+N} \phi_i(\boldsymbol{p})^2 .$$

[0074] Dies steht im Einklang mit der obigen Analyse identischer Eigenwerte, was die notwendige Bedingung symmetrischer Einbettung ist. Da es in der Praxis aufgrund numerischer Beschränkungen selten identische Eigenwerte gibt, kann dort eine Heuristik verwendet werden, wo Eigenwerte als identisch angesehen werden, falls sie innerhalb derselben ε-Kugel liegen (mit einem kleinen ε), d. h. wenn sie sich um weniger als eine vorbestimmte Schwelle, z. B. 0,1 % oder 0,01 %, unterscheiden. Da symmetrische Dimensionen nur einmal für ein gegebenes Objekt gefunden werden müssen, kann dies manuell durchgeführt werden.

[0075] Es sollen zum Beispiel die ersten 7 Eigenwerte einer Eigenwertzerlegung für einen Torus Folgendes sein:

$$\lambda_{0-6} = [0.0,\ 0.0175,\ 0.0175,\ 0.069,\ 0.069,\ 0.1525,\ 0.1525]$$

[0076] Eine GISIF-Einbettung in $\mathbb{R}^3$ - ohne Berücksichtigung der trivialen Lösung - ist dann gegeben durch

$$f(\boldsymbol{p}) = \left[\phi_1(\boldsymbol{p})^2 + \phi_2(\boldsymbol{p})^2, \phi_3(\boldsymbol{p})^2 + \phi_4(\boldsymbol{p})^2, \phi_5(\boldsymbol{p})^2 + \phi_6(\boldsymbol{p})^2\right].$$

[0077] Im Fall mehrerer Objekte kann dies als mehrere getrennte verbundene Graphen repräsentiert werden. Die Adjazenzmatrix ist in diesem Fall blockdiagonal. Der symmetrische positiv-definite Laplace-Operator wird erneut orthogonale Eigenvektoren aufweisen. Es gibt zwei Unterschiede im Ergebnis der Eigenwertzerlegung im Vergleich zu den Fall mit einzelner Grapheneinbettung: erstens werden die nicht abnehmenden Eigenwerte die Einbettungsfehler aller Objekte ohne Ordnung sein. Zweitens werden die Eigenvektoren Null-Einträge aufweisen, da die entsprechenden Eigenwerte orthogonal bleiben. Dies bedeutet, dass jede Dimension des Deskriptorraums nur einer Objekteinbettung entsprechen wird. Ferner sind die Dimensionen mit Bezug auf den Einbettungsfehler des entsprechenden Objekts geordnet. Falls somit eine 3-dimensionale Einbettung von zwei Objekten erzeugt werden sollte, wird d=8 verwendet, da es zwei triviale Lösungen entsprechend $\lambda=0$ gibt.

[0078] Dieser unkomplizierte Ansatz handhabt mehrere Objekte unabhängig, während es suboptimale Verfahren geben kann, die dennoch recht gute Einbettungen mit niedrigerem d bereitstellen, die die Korrelation zwischen Objekten ausnutzen.

[0079] Angesichts der Stellungen von Objekten können die Zielbilder durch Projizieren der Deskriptoren auf die Bildebene erzeugt werden. Als Nichtobjekt (Hintergrund) kann Deskriptorraum-Bildzufallsrauschen oder ein einzelner Deskriptor, der auf den am weitesten entfernten Punkt im Deskriptorraum abgebildet wird, verwendet werden.

[0080] Um die Robustheit des trainierten Netzes 200 zu verbessern, können Bildaugmentationsverfahren wie etwa Domänenrandomisierung oder Störungen wie Gaußscher Weichzeichner, Cropping oder Dropout angewendet werden.

[0081] Zusammengefasst wird gemäß verschiedenen Ausführungsformen ein Verfahren wie in **Fig. 5** veranschaulicht bereitgestellt.

[0082] Fig. 5 zeigt ein Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts gemäß einer Ausführungsform.

[0083] In 501 wird ein 3D-Modell des Objekts erhalten, wobei das 3D-Modell ein Gitter von durch Kanten verbundenen Vertices umfasst, wobei jede Kante eine Gewicht aufweist, das die Nähe zweier durch die Kante verbundener Vertices im Objekt spezifiziert. Die Gewichte können zum Beispiel aus einem CAD-Modell (z. B. durch einen Benutzer bereitgestellt) erhalten (d. h. bestimmt) werden.

[0084] In 502 wird ein Deskriptor für jeden Vertex des Gitters durch Suchen von Deskriptoren für die Vertices bestimmt, die die Summe, über Paare verbundener Vertices, von Distanzen zwischen den Deskriptoren des Paares von Deskriptoren minimieren, gewichtet durch das Gewicht der Kante zwischen dem Paar von Vertices.

[0085] In 503 werden Trainingsdatenbildpaare erzeugt, wobei jedes Trainingsdatenbildpaar ein das Objekt zeigendes Trainingseingabebild und ein Zielbild umfasst, und wobei das Erzeugen des Zielbildes Bestimmen der Vertexpositionen von Vertices des Objektmodells des Objekts, die die Vertices in dem Trainingseingabebild aufweisen, und Zuweisen, für jede bestimmte Vertexposition in dem Trainingseingabebild, des für den Vertex an der Vertexposition bestimmten Deskriptors zu der Position in dem Zielbild umfasst.

[0086] In 504 wird das maschinelle Lernmodell durch überwachtes Lernen unter Verwendung der Trainingsdatenbildpaare als Trainingsdaten trainiert.

[0087] Gemäß verschiedenen Ausführungsformen wird mit anderen Worten ein Objekt in einen Deskriptorraum durch lokales Bewahren geometrischer Eigenschaften eingebettet. Dies wird durch Suchen nach Einbettungen durchgeführt, die die Summe, über Paare verbundener Vertices, von Distanzen zwischen den Einbettungen des Paares von Vertices minimieren, gewichtet durch das Gewicht der Kante zwischen dem Paar von Vertices, wie durch Gleichung (1) beschrieben. Während in den obigen Beispielen das maschinelle Lernnetzwerk als ein neuronales Netz beschrieben ist, können andere Arten von Regressor verwendet werden, die einen 3D-Tensor auf einen anderen 3D-Tensor abbilden.

[0088] Gemäß verschiedenen Ausführungsformen weist das maschinelle Lernmodell Pixeln des Objekts (in der Bildebene) Deskriptoren zu. Dies kann als indirektes Codieren der Oberflächentopologie des Objekts angesehen werden. Diese Verbindung zwischen Deskriptoren und der Oberflächentopologie kann durch

Rendern explizit vorgenommen werden, um die Deskriptoren auf die Bildebene abzubilden. Es sollte angemerkt werden, dass Deskriptorwerte an Flächen (d. h. Punkten, die keine Vertices sind) des Objektmodells mittels Interpolation bestimmt werden können. Wenn beispielsweise eine Fläche durch 3 Vertices des Objektmodells mit ihren jeweiligen Deskriptorwerten yl, y2, y3 gegeben ist, dann kann an einem beliebigen Punkt der Fläche der Deskriptorwert y als eine gewichtete Summe dieser Werte
$w_1 \cdot y_1 + w_2 \cdot y_2 + w_3 \cdot y_3$ berechnet werden. Mit anderen Worten werden die Deskriptorwerte an den Vertices interpoliert.

[0089] Zum Erzeugen von Bildpaaren für Trainingsdaten, ein Bild des Objekts (z. B. ein RGB-Bild) einschließlich des Objekts (oder mehrerer Objekte) mit bekanntem 3D(z. B. CAD)-Modell und bekannter Stellung (in einem globalen (d. h. Welt-) Koordinatensystem zu einem (dichten) Deskriptorbild, das in dem Sinne optimal ist, dass es durch eine Suche nach Deskriptoren zum Minimieren der Abweichung geometrischer Eigenschaften (insbesondere der Nähe von Punkten des Objekts) zwischen dem Objektmodell und seiner Repräsentation (Einbettung) im Deskriptorraum erzeugt wird. Im praktischen Gebrauch wird die theoretische optimale Lösung zur Minimierung im Allgemeinen nicht gefunden werden, da die Suche auf einen gewissen Suchraum beschränkt ist. Nichtsdestotrotz wird eine Schätzung des Minimums innerhalb der Beschränkungen einer praktischen Anwendung (verfügbare Berechnungsgenauigkeit, maximale Anzahl von Iterationen usw.) bestimmt.

[0090] Somit werden Deskriptoren für die Vertices gesucht, indem ein Minimierungsprozess der Summe, über Paare verbundener Vertices, von Distanzen zwischen den Deskriptoren des Paares von Vertices durchgeführt wird, gewichtet durch das Gewicht der Kante zwischen dem Paar von Vertices, wobei jeder Deskriptor für einen jeweiligen Vertex der Vertices gesucht wird.

[0091] Jedes Trainingsdatenbildpaar umfasst ein Trainingseingabebild des Objekts und ein Zielbild, wobei das Zielbild durch Projizieren der in dem Trainingseingabebild sichtbaren Deskriptoren der Vertices auf die Trainingseingabebildebene gemäß der Stellung,die das Objekt in dem Trainingseingabebild aufweist, erzeugt wird.

[0092] Die Bilder zusammen mit ihren assoziierten Zielbildern werden zum überwachten Trainieren des maschinellen Lernmodells verwendet.

[0093] Das maschinelle Lernmodell wird somit trainiert, um eindeutige Merkmale eines Objekts (oder mehrerer Objekte) zu erkennen. Diese Informationen können für verschiedene Anwendungen bei der Robotersteuerung mittels Evaluierung des maschinellen Lernmodells in Echtzeit verwendet werden, z. B. Vorhersagen einer Objektgreifstellung zur Montage. Es sollte angemerkt werden, dass der überwachte Trainingsansatz das explizite Codieren von Symmetrieinformationen ermöglicht.

[0094] Das Verfahren von **Fig. 5** kann durch einen oder mehrere Computer durchgeführt werden, die eine oder mehrere Datenverarbeitungseinheiten beinhalten. Der Ausdruck „Datenverarbeitungseinheit“ kann als eine beliebige Art von Entität verstanden werden, die die Verarbeitung von Daten oder Signalen ermöglicht. Beispielsweise können die Daten oder Signale gemäß mindestens einer (d. h. einer oder mehr als einer) spezifischen Funktion bearbeitet werden, die durch die Datenverarbeitungseinheit durchgeführt wird. Eine Datenverarbeitungseinheit kann eine analoge Schaltung, eine digitale Schaltung, eine Mischsignalschaltung, eine Logikschaltung, einen Mikroprozessor, einen Mikrocontroller, eine Zentralverarbeitungseinheit (CPU), eine Grafikverarbeitungseinheit (GPU), einen Digitalsignalprozessor (DSP), ein programmierbares Gate-Array (FPGA), eine integrierte Schaltung oder eine beliebige Kombination davon beinhalten oder daraus gebildet werden. Eine beliebige andere Weise zum Implementieren der jeweiligen Funktionen, die nachstehend ausführlicher beschrieben wird, kann auch als eine Datenverarbeitungseinheit oder Logikschaltungsanordnung verstanden werden. Es versteht sich, dass ein oder mehrere der hierin ausführlich beschriebenen Verfahrensschritte durch eine Datenverarbeitungseinheit über eine oder mehrere spezifische durch die Datenverarbeitungseinheit durchgeführte Funktionen ausgeführt (z. B. implementiert) werden kann.

[0095] Der Ausdruck „Roboter“ kann so verstanden werden, dass er sich auf ein beliebiges physisches System (mit einem mechanischen Teil, dessen Bewegung gesteuert wird) bezieht,wie etwa eine computergesteuerte Maschine, ein Fahrzeug, ein Haushaltsgerät, ein Elektrowerkzeug, ein Personal Assistant oder ein Zugangskontrollsystem.

[0096] Verschiedene Ausführungsformen können Sensorsignale von verschiedenen Sensoren wie etwa eine (z. B. RGB) Kamera, Video, Radar, LiDAR, Ultraschall, Wärmebildtechnik usw. empfangen und verwenden,

um beispielsweise Sensordaten zu erhalten, die ein Objekt zeigen. Ausführungsformen können zum Erzeugen von Trainingsdaten und Trainieren eines maschinellen Lernsystems verwendet werden, z. B. zum autonomen Steuern eines Roboters, z. B. eines Robotermanipulators, um verschiedene Manipulationsaufgaben unter verschiedenen Szenarios zu erzielen. Insbesondere sind Ausführungsformen bei der Steuerung und Überwachung der Ausführung von Manipulationsaufgaben z. B. bei Montagelinien, anwendbar.

[0097] Obwohl spezifische Ausführungsformen hierin veranschaulicht und beschrieben wurden, sollte ein Durchschnittsfachmann erkennen, dass eine Vielfalt alternativer und/oder äquivalenter Implementierungen für die spezifischen gezeigten und beschriebenen Ausführungsformen ersetzt werden können, ohne vom Schutzumfang der vorliegenden Erfindung abzuweichen. Diese Anmeldung soll jegliche Anpassungen oder Variationen der spezifischen hierin besprochenen Ausführungsformen abdecken. Daher wird beabsichtigt, dass diese Erfindung nur durch die Ansprüche und deren Äquivalente beschränkt wird.

Patentansprüche

1. Verfahren zum Trainieren eines maschinellen Lernmodells zum Erkennen einer Objekttopologie eines Objekts aus einem Bild des Objekts, das Folgendes umfasst:
Erhalten eines 3D-Modells des Objekts, wobei das 3D-Modell ein Gitter von durch Kanten verbundenen Vertices umfasst, wobei jede Kante eine Gewicht aufweist, das die Nähe zweier durch die Kante verbundener Vertices im Objekt spezifiziert;
Bestimmen eines Deskriptors für jeden Vertex des Gitters durch Suchen von Deskriptoren für die Vertices, die die Summe, über Paare verbundener Vertices,
von Distanzen zwischen den Deskriptoren des Paares von Deskriptoren minimieren, gewichtet durch das Gewicht der Kante zwischen dem Paar von Vertices;
Erzeugen von Trainingsdatenbildpaaren, wobei jedes Trainingsdatenbildpaar ein das Objekt zeigendes Trainingseingabebild und ein Zielbild umfasst, und wobei das Erzeugen des Zielbildes Folgendes umfasst:
Bestimmen der Vertexpositionen von Vertices des Objektmodells des Objekts, die die Vertices in dem Trainingseingabebild aufweisen; und
Zuweisen, für jede bestimmte Vertexposition in dem Trainingseingabebild, des für den Vertex an der Vertexposition bestimmten Deskriptors zu der Position in dem Zielbild; und
Trainieren des maschinellen Lernmodells durch überwachtes Lernen unter Verwendung der Trainingsdatenbildpaare als Trainingsdaten.

2. Verfahren nach Anspruch 1, wobei das Erzeugen der Trainingsdatenbildpaare Erhalten mehrerer Bilder des Objekts in unterschiedlichen Stellungen und Erzeugen eines Trainingsdatenbildpaares aus jedem erhaltenen Bild durch Erzeugen eines Zielbildes für das erhaltene Bild umfasst.

3. Verfahren nach Anspruch 1 oder 2, umfassend Bestimmen der Vertexpositionen von Vertices des Objektmodells des Objekts, die die Vertices in den Trainingseingabebildern aufweisen, aus den jeweiligen Stellungen, die das Objekt in den Trainingseingabebildern aufweist.

4. Verfahren nach einem der Ansprüche 1 bis 3, wobei das Suchen der Deskriptoren Bestimmen von Eigenvektoren der Laplace-Matrix des durch die Vertices und die Kanten des 3D-Modells gebildeten Graphen und Annehmen von Komponenten der Eigenvektoren als Komponenten der Deskriptoren umfasst.

5. Verfahren nach Anspruch 4, umfassend Assoziieren jedes Vertex, der mit einer Komponentenposition assoziiert ist, in den Eigenvektoren, und Annehmen, für jeden Vertex, der Komponenten in den Eigenvektoren an der mit dem Vertex assoziierten Komponentenposition als Komponenten des Deskriptors für den Vertex.

6. Verfahren nach Anspruch 5, wobei das Bestimmen der Deskriptoren ferner Komprimieren der Deskriptoren durch Kombinieren von Komponenten von Eigenvektoren, deren Eigenwerte sich um weniger als eine vorbestimmte Schwelle unterscheiden, zu einer einzelnen Komponente umfasst.

7. Verfahren nach einem der Ansprüche 1 bis 6, wobei das Erhalten des 3D-Modells des Objekts umfasst: Erhalten eines 3D-Gitters von Vertices und Kanten, die das Objekt modellieren, und Zuweisen, als Gewichte für die Kanten, der Kantengewichte eines Laplace-Beltrami-Operators, der an dem Gitter angewendet wird.

8. Verfahren zum Steuern eines Roboters, das Folgendes umfasst:
Trainieren eines maschinellen Lernmodells gemäß einem der Ansprüche 1 bis 7; Erhalten eines das Objekt zeigenden Bildes;
Einspeisen des Bildes in das maschinelle Lernmodell;
Bestimmen einer Stellung des Objekts von der Ausgabe des maschinellen Lernmodells; und
Steuern des Roboters in Abhängigkeit von der bestimmten Stellung des Objekts.

9. Verfahren nach Anspruch 8, wobei das Bestimmen der Objektstellung Bestimmen der Position eines gewissen Teils des Objekts umfasst, und wobei das Steuern des Roboters in Abhängigkeit von der bestimmten Stellung des Objekts Steuern eines Endeffektors des Roboters, sich zu der Position des Teils des Objekts zu bewegen und mit dem Teil des Objekts zu interagieren, umfasst.

10. Software- oder Hardware-Agent, insbesondere Roboter, der Folgendes umfasst:
eine Kamera, die zum Bereitstellen von Bilddaten eines Objekts ausgelegt ist;
eine Steuerung, die zum Implementieren eines maschinellen Lernmodells ausgelegt ist; und
eine Trainingsvorrichtung, die zum Trainieren des maschinellen Lernmodells mittels des Verfahrens nach einem der Ansprüche 1 bis 7 ausgelegt ist.

11. Software- oder Hardware-Agent nach Anspruch 10, der mindestens einen Aktor umfasst, wobei die Steuerung zum Steuern des mindestens einen Aktor unter Verwendung einer Ausgabe von dem maschinellen Lernmodell ausgelegt ist.

12. Computerprogramm, das Anweisungen umfasst, die bei Ausführung durch einen Prozessor veranlassen, dass der Prozessor ein Verfahren nach einem der Ansprüche 1 bis 9 durchführt.

13. Computerlesbares Medium, das Anweisungen speichert, die bei Ausführung durch einen Prozessor veranlassen, dass der Prozessor ein Verfahren nach einem der Ansprüche 1 bis 9 durchführt.

Es folgen 4 Seiten Zeichnungen

Anhängende Zeichnungen

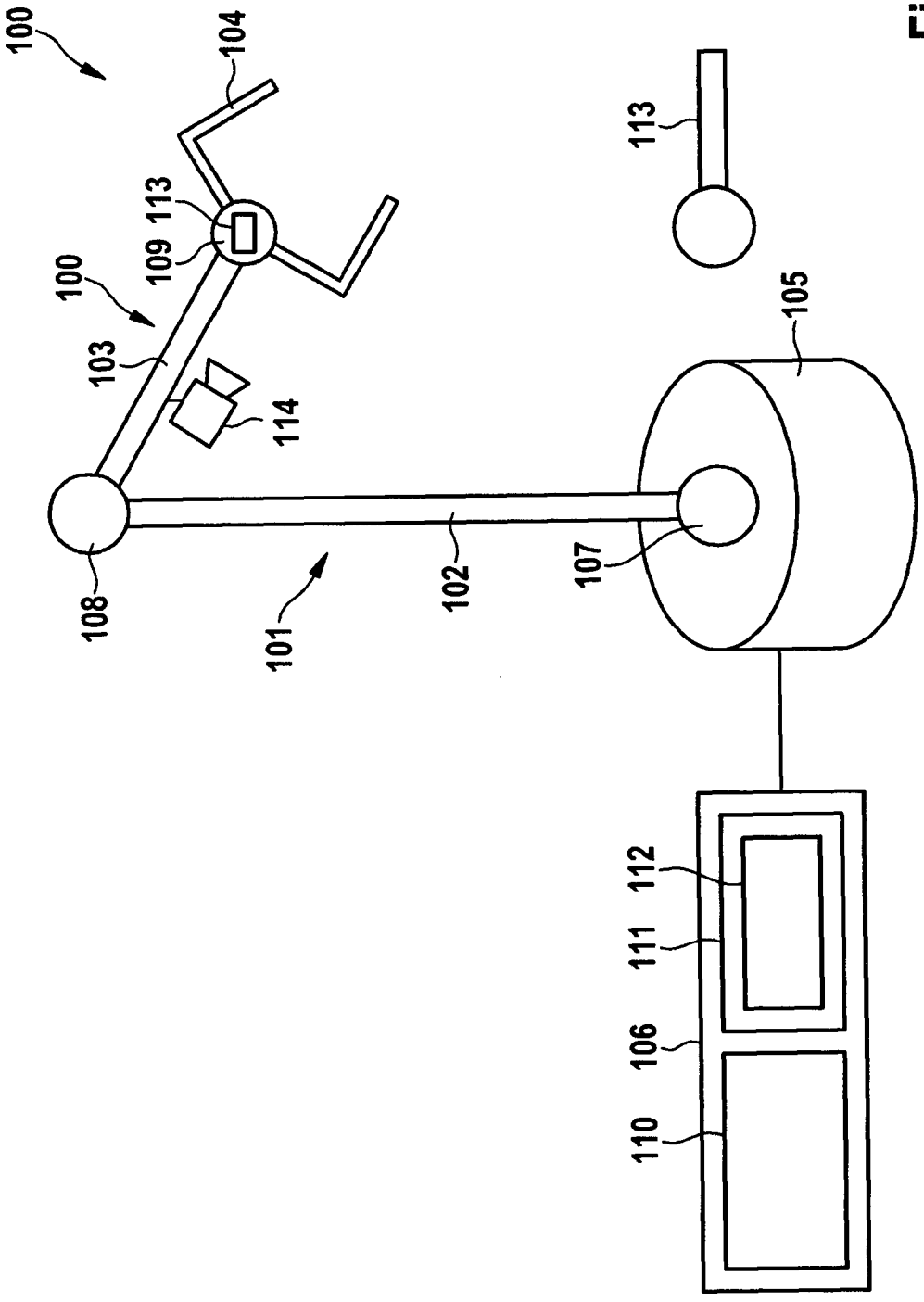


Fig. 1

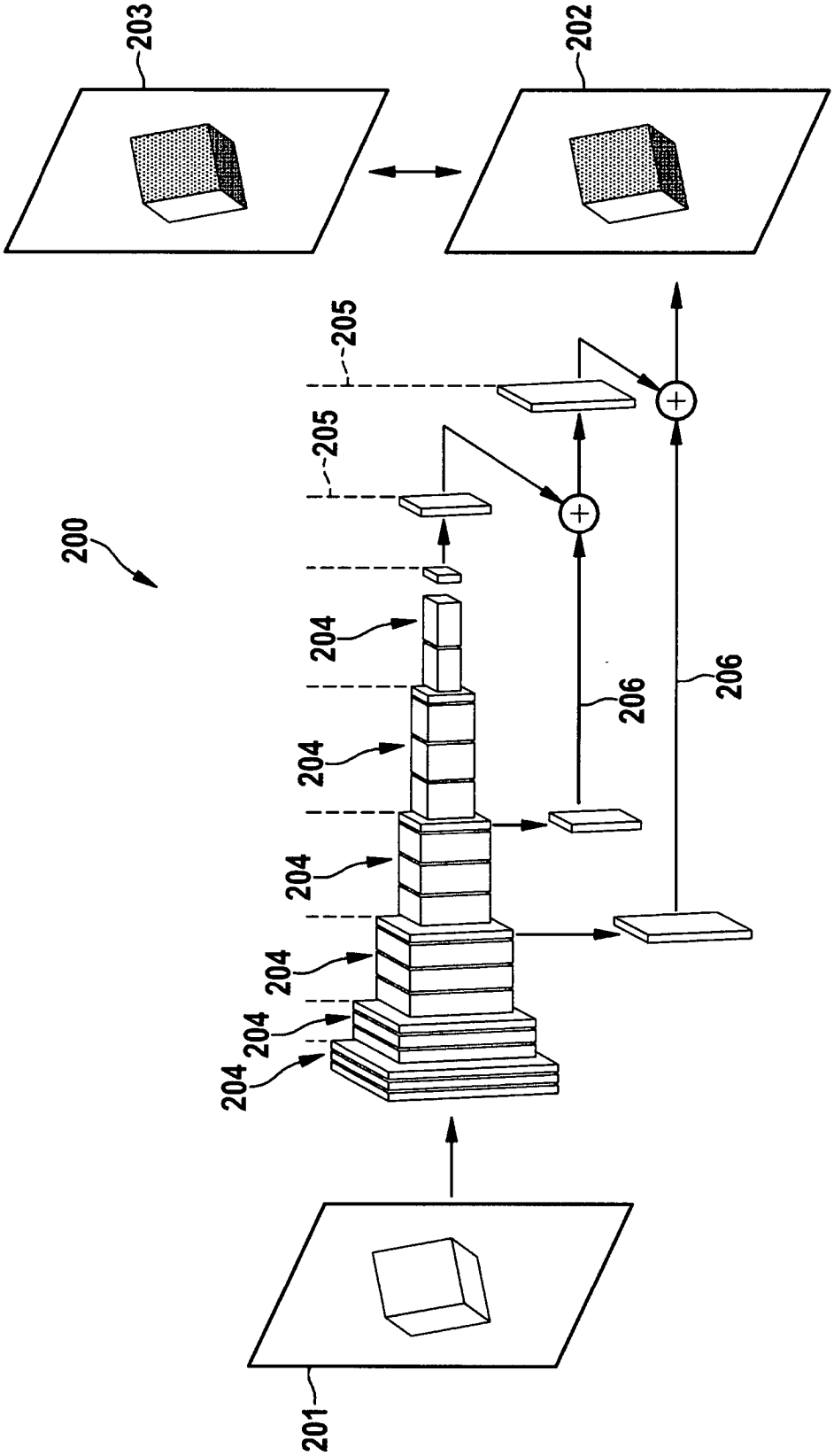


Fig. 2

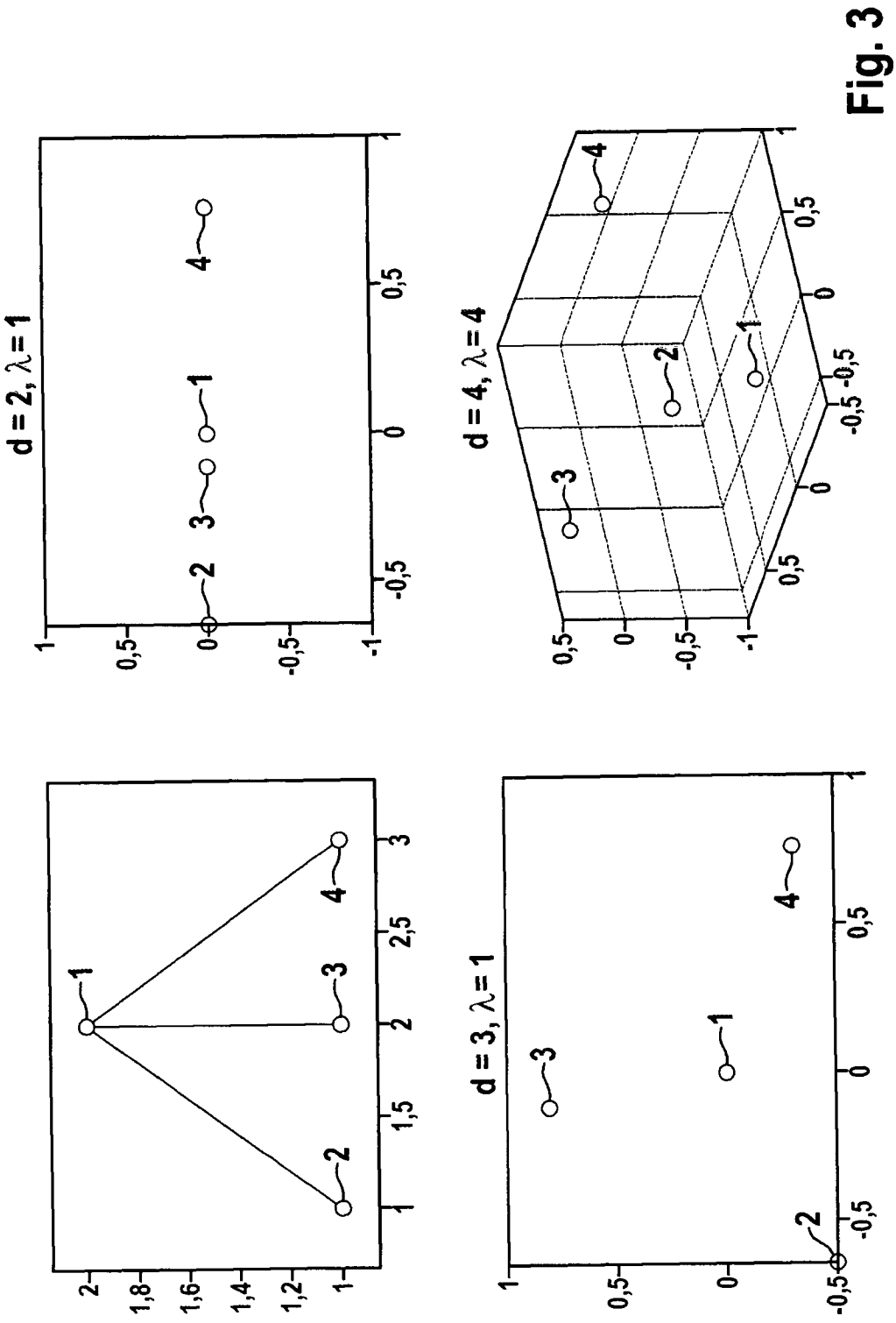


Fig. 3

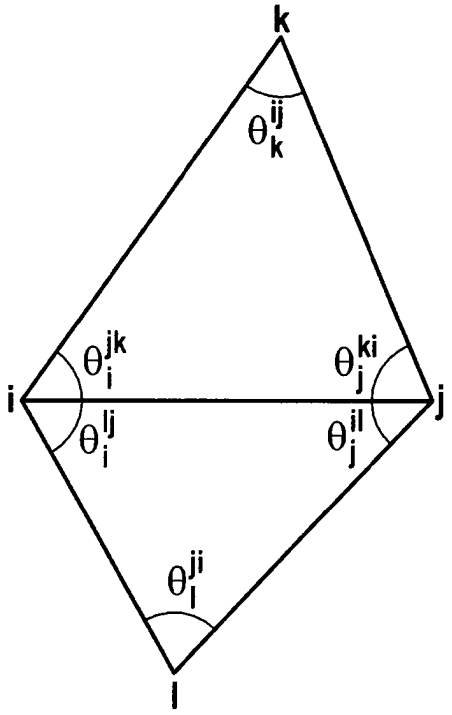


Fig. 4

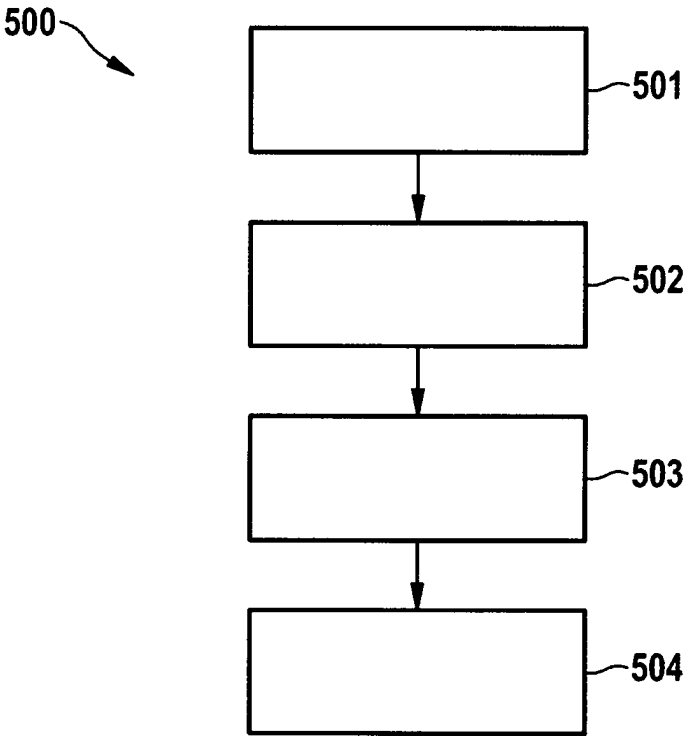


Fig. 5

 Das Dokument wurde durch die Firma Jouve hergestellt.