Brandenburgische Technische Universität Cottbus
Fakultät 1: Mathematik, Naturwissenschaften und Informatik
Lehrstuhl Grafische Systeme

Objekterkennung in Punktwolken auf der Grundlage eines dichtebasierten Clustering-Verfahrens

(Object recognition in point clouds using a density-based clustering algorithm)

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

Eingereicht von:
Jörg Krause
Schillerstraße 8
03046 Cottbus
Matrikel: 2046537

Erstgutachter:
Herr Prof. Dr. Winfried Kurth
Zweitgutachter:
Herr Dipl.-Phys. Thomas Mangoldt

Cottbus, den 25. Mai 2007

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorliegende Arbeit selbständig angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben kann.

Cottbus, den 25. Mai 2007

Danksagung

Mein besonderer Dank gilt Herrn Prof. Dr. Winfried Kurth, für seine ausgezeichnete Betreuung während der Anfertigung dieser Arbeit. Die Gespräche mit ihm und seine Anregungen waren mir stets eine große Hilfe.

Genauso bedanke ich mich bei Alexander Bucksch, der die Idee zu diesem interessanten und spannenden Thema hatte. Er unterstützte mich mit zahlreichen Anregungen und stellte mir die in dieser Arbeit verwendeten Punktwolken zur Verfügung.

Inhaltsverzeichnis

1 Aufgabenstellung und Vorgehensweise

Neue Technologien im Bereich der 3D-Aufnahmetechnik liefern immer bessere Ergebnisse in der Modellierung von Gegenständen, Gebäuden und Landschaften. Seit einigen Jahren sind 3D-Laserscanner in der Lage, in bislang unerreichter Kombination aus Geschwindigkeit und Vollständigkeit, Millionen von Messpunkten in höchster Präzision digital aufzunehmen. Die Messpunkte werden in so genannten Punktwolken zusammengefasst und zur dreidimensionalen Darstellung, Auswertung und Bearbeitung genutzt.

1.1 Aufgabenstellung

Das Ziel dieser Bachelor-Arbeit ist es, ein Verfahren zu implementieren, mit dem Objekte in Punktwolken gefunden und markiert werden. Als Grundlage dient das aus der multivariaten Statistik bekannte dichtebasierte Clustering.

Das Verfahren soll in Anwendungen, bei denen die zu suchenden Objekte durch bestimmte Größenabmessungen charakterisiert und in der Punktwolken-Repräsentation genügend von anderen Objekten getrennt sind, einsatzfähig sein. Es sollen Punktwolken von bis zu 1 Million Punkte bearbeitet werden können.

Das implementierte Verfahren wird anhand mehrerer Beipieldaten getestet und auf seine Anwendbarkeit für Objekterkennungsprobleme untersucht.

1.2 Vorgehensweise

Das folgende 2. Kapitel gibt eine Übersicht über Laserscanning und Objekterkennung im dreidimensionalen Raum.

Eine genaue Beschreibung der dichtebasierten Clusteranalyse enthält Kapitel 3. Neben den Grundlagen wird der in dieser Arbeit verwendete DBSCAN-Algorithmus beschrieben.

Das 4. Kapitel geht auf das verwendete Matching-Verfahren ein und anschließend wird in Kapitel 5 das entwickelte Programm „ORIPUCA“ vorgestellt.

Das Programm und das dadurch realisierte Verfahren der Objekterkennung wird anhand von Beispieldaten auf seine Anwendbarkeit getestet. Kapitel 6 dokumentiert die durchgeführten Testfälle sowie die Ergebnisse.

Das abschließende 7. Kapitel gibt eine Zusammenfassung und einen Ausblick.

2 Laserscanning und Objekterkennung

Das 3D-Laserscanning ist noch eine recht junge Technologie. Mit ihr wachsen die Möglichkeiten in der Vermessung und Erfassung von Objekten im dreidimensionalen Raum. Ebenso steigen die Anforderungen an bestehende Verfahren zur Darstellung, Auswertung und Verarbeitung der erfassten Daten.

Der erste Abschnitt dieses Kapitels gibt eine Übersicht über die Technologie des 3D-Laserscannings. Im zweiten Abschnitt wird ein Framework für ein System zur 3D-Objekterkennung angegeben. Beide Abschnitte geben nur einen Einblick in das jeweilige Thema.

Im Anschluss wird auf die Problematik der Objekterkennung in Punktwolken eingegangen.

2.1 3D-Laserscanning

2.1.1 Verfahren

Als Laserscanning wird die berührungsfreie Oberflächenabtastung von Objekten mittels Laserstrahlen bezeichnet. Die Abtastung kann dabei durch unterschiedliche Verfahren zur Messung der Lichtlaufzeit erfolgen. Die gebräuchlichsten Verfahren sind das Pulslaufzeitverfahren und das Phasendifferenzverfahren.

Der Laserscanner sendet einen Infrarotstrahl auf einen rotierenden Spiegel. Dieser reflektiert den Strahl mit einer vertikalen Bewegung in den zu vermessenden Raum. Trifft der Strahl auf ein Objekt, so wird er in den Scanner zurück reflektiert. Anhand der Differenz der Phasenverschiebung kann die Entfernung des Objekts ermittelt werden. Ein Encoder berechnet anschließend aus der vertikalen Rotation des Spiegels und der horizontalen Stellung des Laserkopfes die x-, y- und z-Koordinaten des reflektierten Punktes.

2.1.2 Einsatzbereiche

Mit einer stetigen Verbesserung der Technologie steigen auch die Anwendungsbereiche für 3D-Laserscanner.

Im Bereich der Denkmalpflege und Bauforschung ermöglicht die Vermessung von Gebäuden eine detaillierte Bestands- und Schadensaufnahme. Mit sehr hohen Abtastraten und Entfernungsauflösungen im Millimeterbereich ist eine Erfassung von feinsten Strukturen und Details möglich. So können zum Beispiel Kathedralen sehr detailliert gespeichert und zu jedem Zeitpunkt Veränderungen in der Bausubstanz festgestellt werden.

Mittels 3D-Messdaten von Gebäuden können Architekten 2D-Pläne, wie Grundrisse und Frontansichten, für spätere Neu- und Umbaumaßnahmen erstellen.

Lasermesstechnik kommt ebenso in der Stadt- und Regionalplanung bei zahlreichen Vermessungen, wie zum Beispiel von Schienen- und Straßennetzen, Städten, Brücken, Bäumen und Wäldern zum Einsatz.

Bestehende Fabrikhallen und Anlagen werden für spätere Umbau- oder Modernisierungsmaßnahmen dreidimensional vermessen. So kann beispielsweise in einer digital erfassten Produktionsanlage ein weiterer Produktionsroboter eingefügt und simuliert werden.

Ein anderes Anwendungsbeispiel ist die Vermessung von Tat- und Unfallorten in der Forensik. Aufgenommene Daten „frieren“ den Ort messtechnisch ein und unterstützen die Bearbeiter bei ihren Ermittlungen.

Weitere Anwendungen von 3D-Laserscannern sind in der Archäologie, der Automobilindustrie, der Geologie, der Filmindustrie und in zahlreichen weiteren Einsatzfeldern zu finden.

2.1.3 Stand der Technik

Moderne 3D-Laserscanner zeichnen sich durch hochpräzise und trotzdem schnelle Aufnahmen sowie eine einfache Bedienung und hohe Flexibilität aus.

Typische moderne Laserscanner, wie der Laser Scanner LS 880 der Firma Faro GmbH & Co. KG, besitzen eine Reichweite von bis zu 70 m. Mit einem entsprechenden Sensormodul sind sogar Reichweiten von bis zu 200 m möglich. Dabei erreicht der Scanner eine Auflösung von unter 1 mm. Der LS 880 erfasst mehr als 120.000 Punkte pro Sekunde und speichert bis zu 700 Millionen Punkte [FAR05a, b].

Diese riesige Menge an Daten wird zumeist auf einer internen Festplatte gespeichert. Zur Darstellung, Bearbeitung oder Weiterverarbeitung können die aufgenommenen Daten auf ein anderes Speichermedium oder System exportiert werden.

2.2 3D-Objekterkennung

2.2.1 Das Prinzip der Objekterkennung

Will ein Mensch ein bestimmtes Objekt aus einer Objektmenge identifizieren, setzt dies gewisse Kenntnisse von ihm über das Objekt voraus. Diese können Informationen über die Größe, die Form, die Lage, die Farbe oder andere Merkmale sein. Liegen Kenntnisse über die Merkmale vor, so kann der Mensch zielgerichtet in einer Objektmenge nach diesem Objekt suchen. Für jedes betrachtete Objekt führt das menschliche Gehirn einen Vergleichsprozess durch. In diesem Prozess werden die Merkmale des betrachteten Objekts mit den Merkmalen des zu identifizierenden Objekts verglichen. Je nach dem Grad der Übereinstimmung wird durch das Gehirn ein Ja, Vielleicht oder Nein als Ergebnis signalisiert.

Wird die Objekterkennung auf eine Maschine übertragen, so verhält es sich bei ihr analog. Dieser müssen ebenso die Informationen über das zu identifizierende Objekt vorliegen. Für jedes Objekt aus der Objektmenge muss ein Vergleichsprozess

durchgeführt werden, der ein Ergebnis liefert. Dieses muss nach bestimmten Kriterien bewertet werden.

Eine Aussage über die Qualität der Objekterkennung kann durch die Erkennungsrate, die Fehlerkennungen und Geschwindigkeit des Vorgangs ausgedrückt werden.

Besl und Jain stellen in [BES85] ein allgemeines Framework für ein System zur Objekterkennung vor. Das in dieser Arbeit entwickelte Verfahren zur Objekterkennung in Punktwolken baut auf diesem Konzept auf.

2.2.2 Allgemeines System zur Objekterkennung

Die Abbildung 2.1 zeigt ein allgemeines System zur Objekterkennung. Es besteht aus den vier Komponenten *„Real World“* (Ausschnitt der realen Welt), *„Sensor Data“* (Sensor-Daten), *„World Model“* (Modell der realen Welt) und *„Symbolic Description“* (symbolische Beschreibung) sowie den beteiligten Prozessen zwischen den Komponenten.

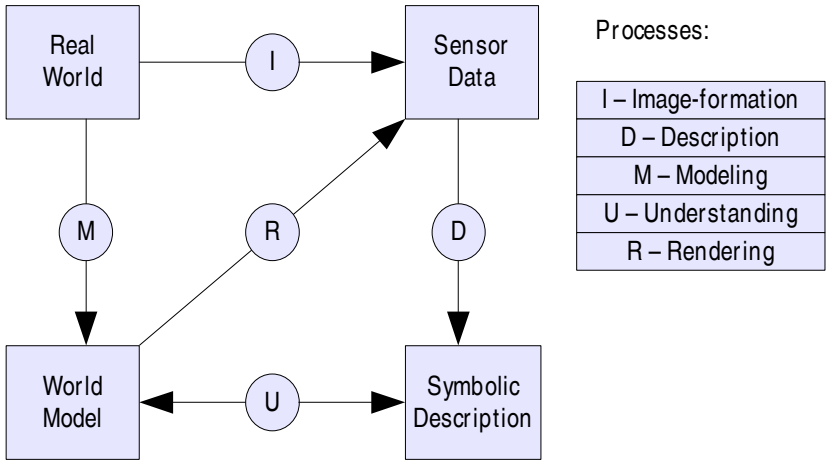


Abb. 2.1 *Framework zur Objekterkennung (Quelle: [BES85])*

Basierend auf einem physikalischen Verfahren, wie zum Beispiel das im vorherigen Abschnitt erläuterte Phasendifferenzverfahren, werden mittels eines Aufnahmegerätes aus einem Ausschnitt aus der realen Welt die Sensor-Daten

erzeugt (Prozess I). Die Sensor-Daten enthalten die Koordinaten der gemessenen Punkte und, abhängig vom Aufnahmegerät, die Intensität.

Ausgehend von dem zu erkennendem Objekt aus der realen Welt wird vom Anwender ein Modell erstellt (Prozess M). Sonka gibt in [SON99] zwei Hauptkategorien von Modellen an. Die Einteilung basiert auf dem Volumen und der Oberfläche eines Objekts. Volumen-Modelle repräsentieren ausdrücklich nur das Innere eines 3D-Objekts, Oberflächen-Modelle nur die Oberfläche eines 3D-Objekts.

Für die Darstellung von Modellen existieren verschiedene Ansätze. Eine Darstellung gilt als vollständig, wenn zwei unterschiedliche Objekte nicht zu demselben Modell gehören können. Eine Darstellung gilt als eindeutig, wenn ein Objekt nicht zu zwei verschiedenen Modellen gehören kann. Als Beispiele zur Modellierung seien hier die Wire-Frame-Darstellung, die CSG-Darstellung, die Voxel-Darstellung und die Octree-Darstellung erwähnt.

Die Sensor-Daten wurden aus einer bestimmten Betrachtungsperspektive gewonnen und stellen somit nicht vollständig das Objekt aus der realen Welt mit allen Merkmalen dar. Aus diesem Grund ist es von Vorteil, mit einer symbolischen Beschreibung zu arbeiten, anhand derer das Matching von Objekt und Modell durchgeführt wird (Prozess U). Dieser Prozess kann aus verschiedenen Unterprozessen bestehen, in denen Merkmale aus der symbolischen Beschreibung mit vorgegebenen Modellen verglichen werden.

Für manche Systeme der Objekterkennung kann es zweckmäßig sein, ein Modell in Sensor-Daten zu transformieren (Prozess R). Der Prozess R bietet eine Art Rückkopplung für ein autonom arbeitendes System, um ein hypothetisches Modell, das für die Erkennung in Frage kommt, zu verifizieren. Dazu wird die symbolische Darstellung des Modells mit der symbolischen Darstellung der, aus dem Prozess R erzeugten, Sensor-Daten verglichen und die Abweichung bestimmt. Ist diese geringer als ein gegebener Schwellenwert, so ist das hypothetische Modell verifiziert und

kann zur Objekterkennung vom System verwendet werden. Andernfalls muss nach einem geeigneteren Modell gesucht werden.

2.2.3 Objekterkennung in Punktwolken

Aufbauend auf dem im vorherigen Abschnitt beschriebenen Konzept wurde folgendes System zur Objekterkennung in Punktwolken entworfen.

Eine gegebene Punktwolke wird mittels des dichtebasierten Clustering-Verfahrens in Cluster unterteilt. Es wird davon ausgegangen, dass jeder Cluster ein Objekt der realen Welt repräsentiert. Jeder Cluster wird durch seine minimale Bounding Box sowie seine Anzahl an Punkten beschrieben. Die Werte der Seitenlängen der Box, gegeben durch die längste, die mittlere und die kürzeste Seite, sowie die Anzahl der Punkte bilden die Merkmale eines Clusters.

Ein vom Nutzer erstelltes Modell stellt ein oder mehrere Objekte aus der realen Welt dar. Es repräsentiert einen unteren und einen oberen Schwellenwert für die minimale Bounding Box sowie einen unteren Schwellenwert für die Anzahl an Punkten.

Mittels eines Matching-Verfahrens werden die Merkmale des Modells mit den Merkmalen der Cluster verglichen.

Liegt eine Übereinstimmung vor, so wird der entsprechende Cluster markiert.

3 Dichtebasiertes Clustering

3.1 Die Idee

Die Grundidee des dichtebasierten Clusterings besteht darin, ähnliche oder zusammengehörige Objekte als zusammenhängende räumliche Gebiete im n-dimensionalen Raum anzusehen und als Cluster zusammenzufassen. Bedingung für ein Cluster ist, dass die lokale Punktdichte aller enthaltenen Objekte einen vorgegebenen Schwellenwert überschreitet. Diese lokale Punktdichte wird dabei durch die Anzahl der Nachbarn in einer festgelegten Umgebung um das jeweilige Objekt beschrieben. Der Schwellenwert wird durch die zwei Eingabeparameter ε (Radius um einen Punkt) und *MinPts (*minimale Anzahl der Punkte innerhalb der ε-Kugel) bestimmt.

3.2 Begriffe

Zur formalen Beschreibung des dichtebasierten Clusterings ist es notwendig, zunächst einige Begriffe zu definieren.

In allen Definitionen sei vorausgesetzt, dass $\epsilon \in \mathbb{R}^+$ und $MinPts \in \mathbb{N}$. Die Menge aller Punkte wird mit *D* bezeichnet.

Der Abstand zwischen zwei Punkten *p* und *q* wird durch folgende euklidische Distanzfunktion ausgedrückt:

$$dist(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2} \qquad (1)$$

Definition: ε-Nachbarschaft

Die ε-Nachbarschaft eines Punktes $p \in D$ ist definiert als

$$N_\epsilon(p) = \{q \in D \mid dist(p, q) \leq \epsilon\} \quad (2)$$

Ein Punkt p soll demnach Teil eines Clusters sein, wenn seine ε-Nachbarschaft zumindest eine gegebene Anzahl an Punkten enthält.

Definition: Kernobjekt

Ein Punkt $p \in D$ ist ein *Kernobjekt* bezüglich ε und *MinPts,* wenn gilt:

$$|N_\epsilon(p)| \geq MinPts \quad (3)$$

Ein Punkt p ist ein *Kernobjekt,* wenn in seiner ε-Umgebung mindestens *MinPts* an Nachbarn liegen.

Definition: Randobjekt

Der Begriff des Randes ist hier anders definiert als in der Topologie.

Ein Punkt $q \in D$ ist ein *Randobjekt*, wenn er kein Kernobjekt ist, sich aber in der ε-Umgebung eines Kernobjekts befindet. Ein *Randobjekt* kann in der Nachbarschaft mehrerer Kernobjekte liegen und somit auch verschiedenen Clustern zugeordnet werden.

Objekte, die weder den Kernobjekten noch den *Randobjekten* zugeordnet werden können, sind so genannte *Rauschobjekte.* Die Definition des *Rauschens* wird weiter unten gegeben.

Abbildung 3.1 veranschaulicht grafisch die Konzepte von Kern-, Rand- und Rauschobjekten für den Parameter MinPts ≤ 7 und dem eingezeichneten Radius ε.

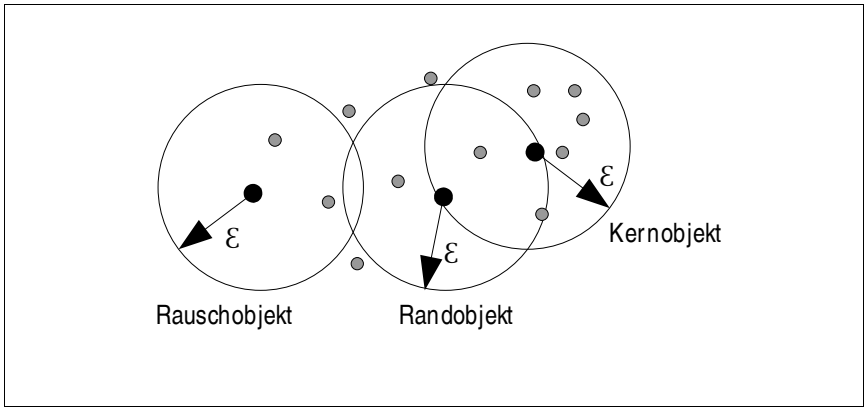


Abb. 3.1: *Rausch-, Rand- und Kernobjekt*

Definition: Direkte Dichte-Erreichbarkeit

Ein Punkt $p \in D$ ist *direkt dichte-erreichbar* von einem Punkt $q \in D$ bezüglich ε und *MinPts* aus, wenn gilt:

$$\begin{aligned} &1.\ p \in |\, N_\epsilon(q)|\,, \\ &2.\ q \textit{ ist ein Kernobjekt in } D. \end{aligned} \tag{4}$$

Alle Punkte, die in der ε-Umgebung eines Kernobjekts *p* liegen, sind *direkt dichte-erreichbar*.

Definition: Dichte-Erreichbarkeit

Ein Punkt $p \in D$ ist *dichte-erreichbar* von einem Punkt $q \in D$ bezüglich ε und *MinPts* aus, wenn es eine Folge von Punkten $p_1, \ldots, p_n$ in D gibt, so dass $p_1 = q$, $p_n = p$ ist, und folgendes gilt: p_{i+1} ist *direkt dichte-erreichbar* von p_i bezüglich ε und *MinPts* aus für $1 \le i \le n$.

Zwei Randobjekte eines Clusters sind unter Umständen nicht dichte-erreichbar zueinander. Jedoch muss es ein Kernobjekt innerhalb des Clusters geben, von dem aus beide Randobjekte dichte-erreichbar sind. Diese Relation von Randobjekten wird durch die *Dichte-Verbundenheit* beschrieben.

Definition: Dichte-Verbundenheit

Ein Punkt $p \in D$ ist *dichte-verbunden* mit einem Punkt $q \in D$ bezüglich ε und *MinPts*, wenn $o \in D$, so dass sowohl *p* als auch *q* dichte-erreichbar bezüglich ε und *MinPts* von *o* aus sind.

Mit Hilfe der Begriffe Dichte-Erreichbarkeit und Dichte-Verbundenheit lassen sich einzelne Cluster, ein komplettes Clustering und das Rauschen in den Daten formal beschreiben.

Definition: Dichtebasiertes Cluster

Ein Cluster *C* bezüglich ε und *MinPts* ist eine nicht-leere Teilmenge von *D*, für die die folgenden Bedingungen erfüllt sein müssen:

1. *Maximalität*: $\forall p, q \in D$: wenn $p \in C$ und *q* dichte-erreichbar von *p* bezüglich ε und *MinPts* aus ist, dann ist auch $q \in C$
2. *Verbundenheit*: $\forall p, q \in C$: *p* ist dichte-verbunden mit *q* bezüglich ε und *MinPts*

Somit ist ein Cluster *C* eine Menge von Punkten, die alle miteinander dichte-verbunden sind, und alle Punkte, die von einem Kernobjekt aus dichte-erreichbar sind, gehören zum Cluster.

Definition: Rauschen

Wenn $CL = \{C_1, ..., C_k\}$ die Menge aller dichtebasierten Cluster der Menge *D* bezüglich ε und *MinPts* ist, dann ist die Menge $Noise_{CL} = D \setminus (C_1 \cup ... \cup C_k)$ definiert als die Menge aller Punkte aus *D*, die nicht zu einem dichtebasiertem Cluster gehören.

Beim Rauschen handelt es sich um Punkte, die weder Kern- noch Randobjekt sind. Sie werden als so genannte Rauschobjekte bezeichnet.

Anhand dieser Definitionen wurde ein einfacher Algorithmus zur Bestimmung aller dichtebasierten Cluster in einer Menge von Objekten *D* entwickelt, der auf folgender wichtigen Eigenschaft beruht: Ein Cluster *C* wird dadurch gefunden, dass man ausgehend von einem beliebigen Kernobjekt aus *C* alle dichte-erreichbaren Punkte zusammenfasst. Die Menge dieser Punkte ist gleich dem Cluster *C*.

Dieser Algorithmus mit dem Namen DBSCAN ("Density-Based Clustering of Applications with Noise") wurde 1996 auf der 2. Internationalen Konferenz für Knowledge Discovery and Data Mining von Ester et al. vorgestellt und in [EST96] beschrieben.

3.3 Das DBSCAN-Verfahren

Der Algorithmus DBSCAN ("Density-Based Clustering of Applications with Noise") findet bezüglich der Eingabeparameter ε und *MinPts* die Cluster und das Rauschen aus einer Datenmenge.

```
DBSCAN (SetOfPoints D, Real Eps, Integer MinPts)
  // D is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO D.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClusterId = UNCLASSIFIED THEN
      IF ExpandCluster(D, Point, ClusterId, Eps, MinPts)
        THEN ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

Abb. 3.2 *Der DBSCAN-Algorithmus [EST96]*

Vor der Ausführung des Algorithmus sind alle Objekte der Datenmenge *D* unklassifiziert. In der Funktion DBSCAN wird jedes Element aus *D* durchlaufen. *Eps* und *MinPts* sind die Dichteparameter, die entweder manuell vorgegeben oder anhand bestimmter Heuristiken, wie im nachfolgenden Abschnitt beschrieben, berechnet werden. Ausgehend von jedem noch unklassifizierten Punkt aus *D* wird versucht, ein Cluster zu finden. Dies geschieht durch den Aufruf der Funktion *ExpandCluster*, die im Folgenden erläutert wird.

Die Methode *regionQuery* gibt die Nachbarn der ε-Umgebung des übergebenen Objekts als eine Liste von Punkten zurück. Bereichsanfragen können durch die Verwendung von geeigneten räumlichen Indexstrukturen effizienter gestaltet werden. Im Abschnitt 3.3.4 werden Möglichkeiten zur Leistungssteigerung näher erläutert.

```
ExpandCluster(SetOfPoints D, Object Point,
  Integer ClId, Real Eps, Integer MinPts) : Boolean;

  seeds := SetOfPoints.regionQuery(Point, Eps);
  IF seeds.size<MinPts THEN // no core point
    SetOfPoints.changeClId(Point, NOISE);
    RETURN False;
  ELSE     // all points in seeds are density-
           // reachable from Point
    SetOfPoints.changeClIds(seeds, ClId);
    seeds.delete(Point);
    WHILE seeds <> Empty DO
      currentP := seeds.first();
      result := SetOfPoints.regionQuery(currentP, Eps);
      IF result.size >= MinPts THEN
        FOR i FROM 1 TO result.size DO
          resultP := result.get(i);
          IF resultP.ClId IN {UNCLASSIFIED, NOISE} THEN
            IF resultP.ClId = UNCLASSIFIED THEN
              seeds.append(resultP);
            END IF;
          SetOfPoints.changeClId(resultP, ClId);
          END IF; // UNCLASSIFIED or NOISE
        END FOR;
      END IF; // result.size >= MinPts
    seeds.delete(currentP);
    END WHILE; // seeds <> Empty
    RETURN True;
  END IF
END; // ExpandCluster
```

Abb. 3.3 *Die Funktion ExpandCluster [EST96]*

Anhand der Anzahl der Nachbarn wird für das aktuelle Objekt geprüft, ob es sich um ein Kernobjekt handelt. Ist dies nicht der Fall, so wird es zunächst dem Rauschen zugeordnet. Handelt es sich jedoch um ein Kernobjekt, so gilt es als Ausgangspunkt für die Suche eines neuen Clusters in der Datenmenge. Von ihm aus wird ein neues Cluster mit allen dichte-erreichbaren Objekten gebildet.

Alle Nachbarn in der ε-Umgebung gehören laut der Definition der Direkten Dichte-Erreichbarkeit auf jeden Fall zum Cluster. Die von den Nachbarn direkt dichte-

erreichbaren Punkte gehören ebenso zum Cluster, da sie gemäß der Definition der Dichte-Erreichbarkeit transitiv dichte-erreichbar sind.

In der Funktion *ExpandCluster* wird durch iterative Berechnung der direkten Dichte-Erreichbarkeit von einem initialen Kernobjekt aus der gesamte Cluster gefunden und alle Elemente dieses Clusters werden mit derselben ID markiert.

3.3.1 Parameterbestimmung für ε und MinPts

Mit dem DBSCAN-Verfahren werden die Cluster bestimmt, deren Punktdichte größer ist als der durch die Parameter ε und *MinPts* spezifizierte Dichteschwellenwert.

Der Ansatz zur Bestimmung der Parameter ε und *MinPts* besteht darin, die Werte, die den am wenigsten dichten Cluster charakterisieren, als Parameter zu übernehmen.

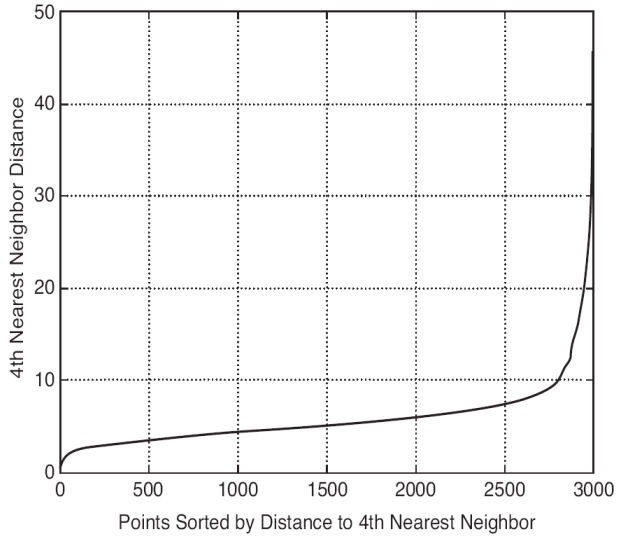


Abb. 3.4 *Das k-dist Plot für ein Beispiel [TAN05, S. 530]*

Dazu wird für ein gegebenes $k \geq 1$ die Funktion *k-dist* definiert, die jedem Objekt der Datenmenge die Distanz zu seinem *k*-nächsten Nachbarn zuordnet. Die aufsteigende Sortierung der Abstände ergibt die in Abbildung 3.4 in Abhängigkeit von *k* (hier 4) dargestellte Funktion.

Für diese Funktion gibt es ein Tal, dem alle Kernobjekte der Datenmenge angehören. Ab einem bestimmten Grenzpunkt, der noch zu einem Cluster gehört, kommen nur noch Objekte mit einer großen k-Distanz vor. Diese Punkte sind keinem Cluster mehr zuordenbar und gehören somit zum Rauschen.

Zur Bestimmung der Parameter im *d*-dimensionalen Raum wird in [EST00, S. 75] folgende allgemeine Heuristik angegeben:

1. Der Benutzer definiert einen Wert *k* (Default ist $k = 2*d-1$), $MinPts := k + 1$.
2. Das System berechnet das *k*-Distanz-Diagramm für eine Stichprobe der Datenmenge und zeigt das Diagramm an.
3. Der Benutzer wählt ein Objekt *o* im *k*-Distanz-Diagramm als Grenzobjekt aus, mit $\varepsilon := k\text{-Distanz}(o)$.

3.3.2 Eigenschaften des Verfahrens

Der Algorithmus berechnet die Menge aller dichtebasierten Cluster *CL* und eine Menge *Noise* gemäß den oben angeführten Definitionen. Bis auf die Zuordnung von Randobjekten, die zu mehreren Clustern gehören können, ist das Ergebnis unabhängig von der Reihenfolge der Daten. Das bedeutet, dass die Clusteranalyse unabhängig von einem Startobjekt durchgeführt werden kann.

Im Gegensatz zu anderen partitionierenden Verfahren ist es mit DBSCAN möglich, Cluster beliebiger Formen zu finden. Ebenso bietet es den Vorteil, die Analyse ohne vorherige Kenntnis der konkreten Anzahl der Cluster durchzuführen.

Für das Clustering durchläuft der Algorithmus einmal komplett die Datenmenge. Somit lässt sich die Laufzeitkomplexität des Algorithmus mit $O(n$ * Aufwand zur Bestimmung der ε-Nachbarschaft) abschätzen, wobei *n* die Anzahl der Objekte *o* ist. Wird für jeden Punkt *p* einzeln die Bedingung $dist(p, o) \leq \varepsilon$ geprüft, so beträgt der Aufwand $O(n^2)$. Dieser Aufwand lässt sich durch effizientere Bereichsanfragen verringern.

Das Verfahren besitzt einige Schwachstellen.

Hierarchische Cluster, die ineinander verschachtelt sind, können nicht erkannt werden.

Es ist nicht möglich, alle Cluster in einer Datenmenge mit stark unterschiedlichen Dichten zu erkennen. In diesem Fall kann der Algorithmus auf keine Parameter zur Bestimmung der Cluster zurückgreifen. Abhängig davon, wie ε in diesem Fall gewählt wird, werden unterschiedliche Teilmengen der Cluster als Ergebnis ausgegeben. Ein kleinerer Dichteschwellenwert erkennt zwar dünnere Cluster, weist aber dichtere Cluster als zusammengehörig aus.

Problematisch ist die Berechnung für hoch-dimensionierte Daten, da hier die Dichte schwieriger definierbar und die Bestimmung der k-nächsten Nachbarschaft sehr rechenaufwendig wird.

Da die von Laserscannern generierten Punktwolken nicht verschachtelt und diese niedrig-dimensional sind, haben die erst- und letztgenannte Schwachstelle für das Clustering von Punktwolken keinen Einfluss.

3.3.3 Bereichsanfragen für dichtebasiertes Clustering

In der Funktion *regionQuery* des DBSCAN-Algorithmus wird die ε-Nachbarschaft für jeden Punkt o der Punktwolke D berechnet. Die einfachste Möglichkeit der Bestimmung ist, wie schon im oberen Abschnitt erwähnt, der direkte Vergleich der Distanz $dist(p, o) \leq \varepsilon$ für jeden Punkt p. Der Aufwand beträgt dann $O(n^2)$.

Bereichsanfragen mit einem Zentrum o und einem Radius ε werden durch räumliche Indexstrukturen unterstützt. In niedrig-dimensionalen Räumen kann dann eine Bereichsanfrage in logarithmischer Laufzeit beantwortet werden.

Ist ein direkter Zugriff auf die ε-Nachbarschaft der Objekte möglich, so erfolgt der Zugriff sogar in konstanter Zeit. Die Tabelle 3.1 zeigt die sich ergebenden

Laufzeitkomplexitäten für den DBSCAN-Algorithmus mit unterschiedlichen Bereichsanfragen.

Laufzeitkomplexität	- einer einzelnen Bereichsanfrage	- DBSCAN-Algorithmus
ohne Index	$O(n)$	$O(n^2)$
mit räumlichem Index	$O(\log n)$	$O(n * \log n)$
mit direktem Zugriff	$O(1)$	$O(n)$

Tab. 3.1 *Laufzeitkomplexität des DBSCAN-Algorithmus für unterschiedliche Bereichsanfragen [EST00, S. 89]*

Da ein quadratischer Aufwand für das Clustering bei großen Datenmengen von der Laufzeit her nicht akzeptabel ist, wird die Verwendung einer indexbasierten Datenstruktur zur Bestimmung der ε-Nachbarschaften zwingend notwendig. Dazu werden die Punkte einem Octree zugeordnet. Diese Datenstruktur wird im nächsten Abschnitt näher erläutert.

3.3.4 Die Octree-Datenstruktur

Der Octree ist eine hierarchische Datenstruktur. Er basiert auf der rekursiven Unterteilung eines Knotens in acht Söhne. Jeder Knoten repräsentiert einen Kubus im dreidimensionalen Raum. Die Wurzel des Octrees stellt zugleich die achsenparallele Bounding Box der Punktwolke dar. Anschaulich sind der Octree in Form eines Kubus' mit acht Unterkuben und der äquivalente Baum in Abbildung 3.5 dargestellt.

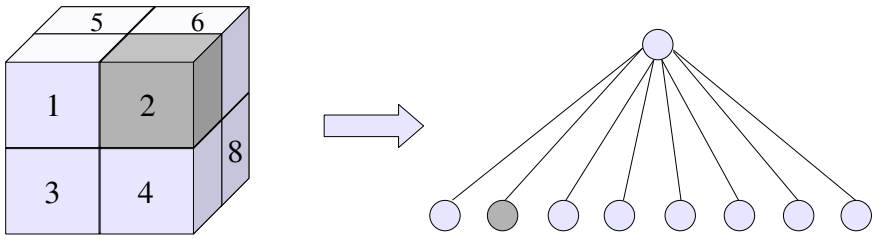


Abb. 3.5 *Die Octree-Datenstruktur [TOE94]*

Samet gibt in [SAM90] eine Vereinbarung für den so genannten Point Region Quadtree (PR Quadtree) an. Der Quadtree ist das zweidimensionale Äquivalent zum Octree. In einem PR Quadtree wird jeder Punkt entsprechend seiner Lage im Koordinatensystem einem Quadranten zugeordnet. Alle Quadranten besitzen dieselbe Kapazität. Sie gibt die Anzahl an Punkten an, die einem Quadranten zugeordnet werden kann. Der Wert für die Kapazität wurde mittels einer Laufzeituntersuchung bestimmt. Diese Untersuchung ist am Ende dieses Abschnittes beschrieben.

Die Vereinbarungen für den PR Quadtree werden für den in dieser Arbeit verwendeten Octree übernommen.

Jeder Knoten des Octree enthält als Schlüssel eine Information, ob der durch ihn repräsentierte Unterkubus die Eigenschaft gefüllt (schwarz), überfüllt (grau) oder leer (weiß) besitzt. Knoten mit der Eigenschaft gefüllt oder leer sind Blätter des Octree. Sind einem Blatt des Octree keine Punkte zugeordnet, so gilt es als leer. Ein Blatt gilt als gefüllt, wenn es mindestens einen Punkt, aber höchstens so viele, wie die Kapazität angibt, enthält. Werden einem Blatt mehr Punkte zugeordnet, als durch die Kapazität vorgegeben wird, so gilt es als überfüllt. Ein überfülltes Blatt wird zu einem Knoten mit acht Söhnen. Sie repräsentieren eine Unterteilung des durch den Knoten repräsentierten Kubus' in acht gleich große, sich nicht überlappende Unterkuben. Diese haben die halbe Seitenlänge des Vaterkubus.

Die rekursive Erzeugung des Octrees erfolgt durch die Zuordnung der Punkte der Punktwolke. Begonnen wird mit dem leerem Wurzelelement. Nach der Zuordnung des ersten Punkts wird der Schlüssel der Wurzel auf schwarz gesetzt. Solange die Anzahl der zugeordneten Punkte nicht die durch die Kapazität vorgegebene Schwelle überschreitet, werden ihr weitere Punkte zugeordnet. Wird die Schwelle erreicht, wird die Wurzel in ihre acht Söhne unterteilt und jeder ihr bisher zugeordnete Punkt wird dem entsprechendem Sohn zugeordnet. Der Schlüssel der Wurzel wird auf grau gesetzt. Ihr sind nach diesem Vorgang keine Punkte mehr zugeordnet. Die Söhne der Wurzel enthalten, je nach ihrer Eigenschaft, den Schlüssel weiß oder schwarz. Die weitere Zuordnung der Punkte erfolgt nach diesem beschriebenen Prinzip.

Eine Schwachstelle des Octree, wie er in dieser Arbeit verwendet wird, ist die zuvor notwendige Bestimmung der räumlichen Ausdehnung des durch die Wurzel repräsentierten Kubus'. Erst im Anschluss kann der Octree erzeugt werden.

Ein weiteres Manko stellt die potentielle Unbalance des Octrees dar. Diese tritt ein, sobald die Punkte innerhalb der achsenparallelen Bounding Box nicht gleichverteilt sind.

Die Bestimmung einer geeigneten Kapazität mittels Laufzeitmessung wurde anhand von vier Punktwolken unternommen. Punktwolke 1 und 2 sind Aufnahmen von Zweigen eines Apfelbaumes und repräsentieren eine ungleichmäßige Verteilung der Punkte innerhalb der achsenparallelen Bounding Box. Punktwolke 3 und 4 sind künstlich erzeugte Punktwolken in der Form eines Quaders. Sie repräsentieren eine gleichmäßige Verteilung.

Punktwolke 1 (23.743 Punkte) und 3 (24.389 Punkte) sowie Punktwolke 2 (149.212 Punkte) und 4 (148.877) enthalten jeweils ähnlich viele Punkte.

Die Tabelle 3.2 zeigt die durchschnittlichen Ergebnisse der Laufzeitmessungen aus jeweils drei Durchläufen. Gemessen wurde die Ausführungszeit für den DBSCAN-Algorithmus in Millisekunden.

Kapazität	Laufzeit [ms]			
	Beispiel 1	Beispiel 2	Beispiel 3	Beispiel 4
10	812	7.359	390	2.485
25	797	6.016	375	2.472
35	782	6.594	375	2.485
50	1.500	10.531	682	4.562

Tab. 3.2 *Laufzeitmessungen der Clusteranalyse für verschiedene Kapazitäten für die Blätter des Octree*

Da sich die Laufzeit ab einer Kapazität von 50 Punkten je Blatt deutlich verschlechtert, wurde hier die Messung beendet. Ein akzeptabler Wert für die Kapazität der Blätter des Octree liegt im Bereich von 10 bis 35 Punkten je Blatt. Als Kapazität wurde in dieser Arbeit ein Wert von 25 Punkten je Blatt gewählt.

4 Objekterkennung

Wie bereits in Kapitel 2.2 erläutert, wird das Matching zwischen dem Modell und den Merkmalen der symbolischen Beschreibung des betrachteten Objekts durchgeführt. Die Merkmale sind durch die Seitenlängen der minimalen Bounding Box sowie der Anzahl an Punkten des Objekts gegeben.

4.1 Bestimmung der Parameter zur Objektbeschreibung

Ein Modell wird vom Nutzer der Anwendung durch die Parameter zur Objektbeschreibung vorgegeben. Die Parameter definieren eine untere und obere Schwelle für die Seitenlängen der minimalen Bounding Box sowie eine untere Schwelle für die Anzahl an Punkten eines oder mehrerer Objekt aus der realen Welt. Die Seitenlängen der Box werden durch die Werte für die längste, die mittlere und die kürzeste Seite angegeben.

Sind die Parameter zur Objektbeschreibung dem Nutzer bereits bekannt, können diese manuell eingegeben werden.

Eine andere Möglichkeit bietet das Bestimmen der Parameter durch die Eingabe einer Menge von Trainingsobjekten. Diese müssen als Punktwolke im xyz- oder txt-Dateiformat vorliegen. Für jedes eingegebene Trainingsobjekt wird der Wert für die längste, die mittlere und die kürzeste Seite der minimalen Bounding Box sowie die Anzahl der Punkte bestimmt und wie folgt in Tabelle 4.1 ausgegeben.

Dabei gilt folgende Definition: P sei die Menge der Anzahl der Punkte aller Trainingsobjekte. L sei die Menge der längsten Seiten, M die Menge der mittleren Seiten und K die Menge der kürzesten Seiten aller Trainingsobjekte.

Objekt	P	L	M	K
1	p_1	l_1	m_1	k_1
...	...	...	...	...
n	p_n	l_n	m_n	k_n
Minimum	$\min(p_1, ..., p_n)$	$\min(l_1, ..., l_n)$	$\min(m_1, ..., m_n)$	$\min(k_1, ..., k_n)$
Maximum		$\max(l_1, ..., l_n)$	$\max(m_1, ..., m_n)$	$\max(k_1, ..., k_n)$

Tab. 4.1 *Bestimmung der Parameter zur Objekterkennung mittels Trainingsobjekten (P sei die Menge der Anzahl der Punkte, L sei die Menge der längsten Seiten, M sei die Menge der mittleren Seiten und K sei die Menge der kürzesten Seiten).*

4.2 Bestimmung der minimalen Bounding Box

Zur Bestimmung der minimalen Bounding Box wird die Hauptkomponentenanalyse (HKA), ein Verfahren aus der multivariaten Statistik, verwendet. Die HKA dient dem Auffinden von Abhängigkeiten zwischen den Dimensionen von Daten. Sie findet eine breite Anwendung in der Gesichtserkennung und der Kompression von Bilddaten.

Die HKA erhält als Eingabe die Kovarianzen einer Punktwolke und berechnet anhand dieser die drei orthogonalen Eigenvektoren. Diese bilden die Hauptachsen eines neuen Koordinatensystems. Mathematisch gesehen wird demnach mittels der HKA eine Hauptachsentransformation durchgeführt. In diesem neuen Koordinatensystem stellt die achsenparallele Bounding Box zugleich eine approximative minimale Bounding Box dar. Im Folgenden werden die Schritte der Hauptkomponentenanalyse näher erläutert:

Schritt 1: Berechnung der Kovarianzmatrix *C* der Punktwolke:

$$\boldsymbol{C} = \begin{pmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{pmatrix}$$

mit den neun Kovarianzen

$$c_{xx} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{(n-1)}$$

$$c_{yy} = \frac{\sum_{i=1}^{n} (y_i - \bar{y})^2}{(n-1)}$$

$$c_{zz} = \frac{\sum_{i=1}^{n} (z_i - \bar{z})^2}{(n-1)}$$

$$c_{xy} = c_{yx} = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y})}{(n-1)}$$

$$c_{xz} = c_{zx} = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) * (z_i - \bar{z})}{(n-1)}$$

$$c_{yz} = c_{zy} = \frac{\sum_{i=1}^{n} (y_i - \bar{y}) * (z_i - \bar{z})}{(n-1)}$$

Schritt 2: Die Berechnung der Eigenwerte und der Eigenvektoren **u**, **v** und **w** mittels Singulärwertzerlegung. Zur Erläuterung dieses Verfahrens wird an dieser Stelle auf die entsprechende Fachliteratur verwiesen. Der verwendete Quellcode für die Singulärwertzerlegung stammt von Press et al. [PRE02].

Schritt 3: Die drei orthogonalen Eigenvektoren **u**, **v** und **w** definieren ein neues Koordinatensystem. Die erste Achse des neuen Koordinatensystems wird dabei so gelegt, dass die Varianz der Daten in diese Richtung maximiert wird. Die zweite Achse steht auf dieser senkrecht. In ihrer Richtung ist die Varianz am zweitgrößten. Die Dritte steht wiederum auf dieser senkrecht und die Varianz ist hier am geringsten. Die neuen Koordinaten x'_i, y'_i und z'_i der Punkte sind durch die lineare Transformation der alten Koordinaten x_i, y_i und z_i mit den drei Eigenvektoren gegeben:

$$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

In diesem neuen Koordinatensystem stellt die achsenparallele Bounding Box eine approximative minimale Bounding Box dar.

Der Aufwand für die Berechnung der approximativen minimalen Bounding Box wird von Wu [WU92] als linear angegeben.

5. Das Programm „ORIPUCA“

In diesem Kapitel wird das für diese Arbeit entwickelte Programm „ORIPUCA“ (**O**bject **R**ecognition **I**n **P**oint Clouds **U**sing **C**luster **A**nalysis) vorgestellt. In den folgenden Abschnitten wird auf den Entwurf, die Implementierung, die Installation und die Anwendung eingegangen. Anhang B enthält ein vereinfachtes Klassendiagramm des Programms.

5.1 Entwurf

Entworfen wurde das Programm als klassische Desktop-Anwendung, das auf einer Zwei-Schichten-Architektur basiert. Die Fachkonzept-Schicht ist von der GUI-Schicht getrennt.

Das Fachkonzept besteht aus den folgenden Klassen:

- *PointCloud*: Die Klasse PointCloud beschreibt das Konzept einer Punktwolke. Eine Punktwolke kann aus mehreren Clustern bestehen. Die Klasse verwaltet diese Cluster.
- *Octree*: Die Klasse Octree beschreibt das Konzept einer Octree-Datenstruktur aus Abschnitt 3.3.4 Sie enthält die Funktion zur Unterstützung der Bereichsanfrage bei der Clusteranalyse.
- *ClusterAnalysis*: Die Klasse ClusterAnalysis setzt den DBSCAN-Algorithmus aus Abschnitt 3.3 um. Sie stellt die Funktion zur Berechnung der knn-Distanz zur Verfügung.
- *ObjectRecognition*: Die Klasse ObjectRecognition realisiert das im 4. Kapitel beschriebene Verfahren zur Berechnung der minimalen Bounding Box und führt das Matching durch.

– *SVD*: Diese Klasse wird von ObjectRecognition zur Bestimmung der benötigten Eigenwerte und Eigenvektoren benutzt. Der Quellcode dieser Klasse stammt von Press et al. [PRE05].

Die Benutzeroberfläche von „ORIPUCA“ ist auf der folgenden Seite in Abbildung 5.1 dargestellt und soll im Folgenden näher erläutert werden. Sie besteht aus einem Bereich zur Eingabe der Dichte- und Abmessungsparameter, der links angeordnet ist. Für die Eingabeparameter der Clusteranalyse steht die Fenster-Box „Cluster Analysis Setting“ zur Verfügung. Sie integriert die grafische Darstellung der knn-Distanzfunktion. Über einen Schieberegler kann ein entsprechender Wert für den Parameter ε interaktiv bestimmt werden.

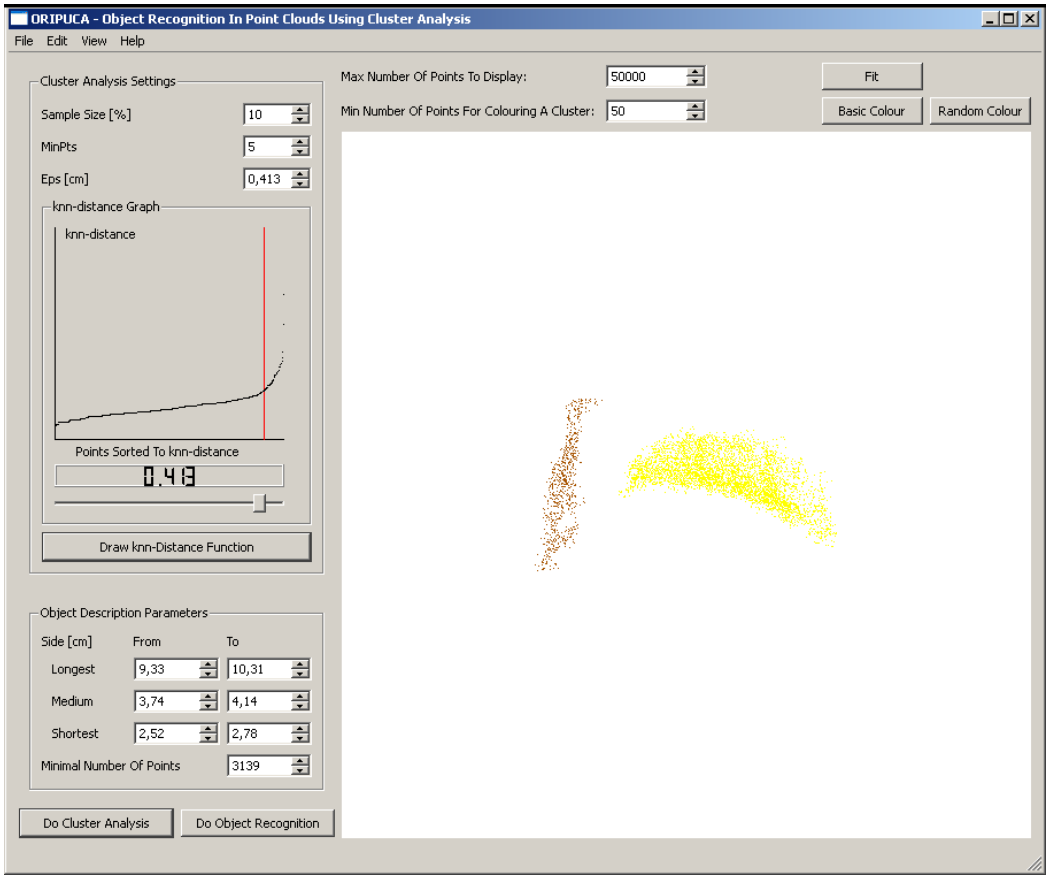


Abb. 5.1 *Benutzeroberfläche des Programms „ORIPUCA“*

Für die Eingabe der Parameter zur Objekterkennung dient die Fenster-Box „Object Description Parameters“, in die die Werte direkt eingegeben werden können. Alternativ ist eine Bestimmung der Parameter mittels des Trainingseditors möglich, der in der Abbildung 5.2 dargestellt ist. Der Trainingseditor setzt die Tabelle 4.1 grafisch um.

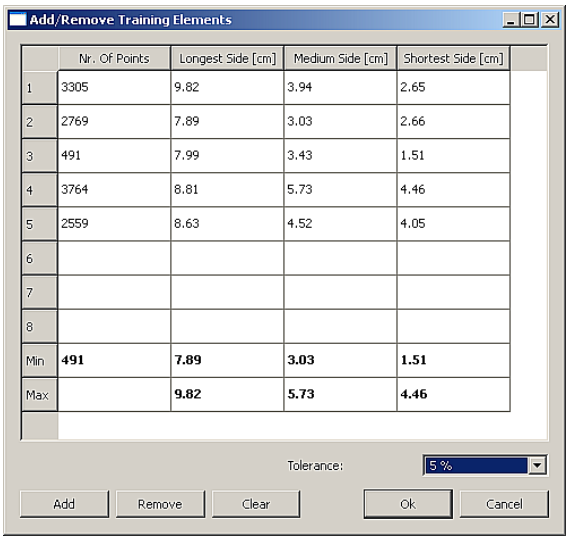


Abb. 5.2 *Fenster zur Bestimmung der Parameter zur Objekterkennung mittels Trainingsobjekten*

Der Hauptteil der Oberfläche wird durch die grafische Anzeige der Punktwolke und der Cluster eingenommen. Zur besseren Darstellung wurde in Abbildung 5.1 die Hintergrundfarbe der Anzeige von schwarz auf weiß geändert.

Oberhalb der Anzeige befindet sich die Scroll-Box „Max Number Of Points To Display“, die den Wert der maximal anzeigbaren Punkte angibt. Die Scroll-Box „Min Number Of Points For Colouring A Cluster“ gibt einen Wert für die minimale Anzahl an Punkten an, die ein Cluster enthalten muss, um nicht grau dargestellt zu werden. Durch das Betätigen des Buttons „Fit“ wird eine automatische Anpassung des Abbildungsmaßstabs an die maximale Abmessung der Punktwolke vorgenommen. Durch das Betätigen des Buttons „Random Colour“ wird jedem nicht-grau markierten Cluster eine zufällige Farbe der 13 verwendeten Farben

zugeordnet. Mit dem Betätigen des Buttons „Basic Colour“ wird jeder nicht-grau markierte Cluster gemäß einer Farbverwaltung markiert. Die genannten Elemente werden im Folgenden näher beschrieben.

Da eine Darstellung aller Punkte gerade für große Punktwolken eine rechentechnische Herausforderung für die Grafikkarte des Computers ist, kann der Anwender des Programms eine obere Grenze für die Anzahl der dargestellten Punkte vordefinieren. Der Defaultwert beträgt 50.000 Punkte. Ist die Anzahl der Punkte größer, so wird nur eine systematische Stichprobe der Punktwolke grafisch angezeigt. Begonnen mit dem ersten Punkt aus der Menge der Punktwolke wird jeder i-te Punkt gezeichnet. Der fixe Abstand *i* ist dabei der Quotient aus der Anzahl der Punkte der Punktwolke und der Anzahl der zu zeichnenden Punkten.

```
für jedes Cluster C aus AllClusters {
      finde zu C die 13 nächsten Nachbarn aus
            ColouredClusters;
      zähle die Farben der Nachbarn;
      if not alle Farben vergeben
            Colour = eine nicht vergebene Farbe;
      else Colour = die Farbe des am weitesten entfernten
            Clusters;
      C.setzeFarbe(Colour);
      ColouredClusters.add(C);
}
```

Abb. 5.3 *Pseudocode der Farbverwaltung*

Für eine sinnvolle Darstellung der Cluster wurde eine einfache Farbverwaltung entworfen, die dazu dient, dass benachbarte Cluster farblich unterschiedlich markiert werden und somit für den Nutzer eindeutig unterscheidbar sind. Der Pseudocode für den Algorithmus zur Farbverwaltung ist in Abbildung 5.3 dargestellt. Dabei sei *AllClusters* als die Menge aller Cluster, die durch die Clusteranalyse gefunden

wurden, sowie *ColouredClusters* als die Menge aller bereits gezeichneten Cluster definiert.

Cluster, die weniger als eine bestimmte Anzahl an Punkten enthalten, werden grau gezeichnet, damit sie keine Farbressourcen beanspruchen. Als Defaultwert ist eine Grenze von 50 Punkten, die ein Cluster enthalten muss, gesetzt. Der Anwender kann diesen Wert anpassen.

Das Programm ermöglicht eine Speicherung der gefundenen Cluster im xyz- oder im xml-Format. Der strukturelle Aufbau der xml-Datei ist in Abbildung 5.4 dargestellt.

```
<?xml version="1.0"?>
      <pointcloud>
            <cluster>
                  <point>
                        <x></x>
                        <y></y>
                        <z></z>
                  </point>
            </cluster>
      </pointcloud>
```

Abb. 5.4 *Struktureller Aufbau der xml-Datei*

5.2 Implementierung

Das Programm ist in der Programmiersprache C++ geschrieben. Für die Implementierung der grafischen Benutzeroberfläche wurde die Klassenbibliothek Qt in der Version 4.2.2 Open Source Edition verwendet. Als Entwicklungsumgebung diente Microsoft Visual Studio 2005 Professional Edition unter Windows XP Service Pack 2.

5.3 Installation

Für die Installation des Programms und der benötigten Bibliotheken wird ein Installer für Microsoft Windows zur Verfügung gestellt. Der Speicherplatzbedarf beträgt circa 8 Mbyte.

5.4 Anwendung

Über das Menü „Datei“ kann eine Punktwolke entweder im Datenformat xyz oder txt geöffnet werden. Die Punkte der Punktwolke werden in dem Anzeigefenster gezeichnet. Mit Hilfe der Maus kann der Nutzer die Betrachtungsperspektive der Punktwolke ändern. Bei gedrückter linker Maustaste und gleichzeitigem Bewegen der Maus lässt sich die Punktwolke um ihre Achsen rotieren. Bei gedrückter rechter Maustaste und einem Bewegen der Maus lässt sich die Punktwolke im Raum verschieben. Mit dem Mausrad lässt sich die Punktwolke heran- oder wegzoomen. Der Button „Fit“ bietet jederzeit die Möglichkeit, zur ursprünglichen Betrachtung zurückzukehren.

Im Fenster „Cluster Analysis Settings“ wird durch das Betätigen der „Draw knn-Distance Function“ der entsprechende Funktionsverlauf im Unterfenster „knn-Distance Graph“ gezeichnet. Mithilfe des Schiebereglers ist es möglich, einen Wert für den Dichteparameter ε zu bestimmen. Alternativ kann ein Wert auch direkt ohne das Zeichnen der Funktion eingegeben werden.

Nachdem der Wert für ε festgelegt wurde, wird durch das Betätigen des Buttons „Do Cluster Analysis“ die Clusteranalyse durchgeführt. Im Anzeigefenster werden die gefundenen Cluster unterschiedlich farbig dargestellt. Sind Cluster nicht zufriedenstellend unterteilt worden, so kann der Nutzer diese Cluster selektieren. Dazu muss im Menü „Edit“ der Menüpunkt „Select Clusters“ gewählt werden. Das Programm befindet sich nun im Selektiermodus. Selektierte Cluster werden mit groben weißen Punkten dargestellt. Aus diesem Selektiermodus kann über die

Funktion „Edit“ und den Eintrag „Edit Selected Clusters...“ in den Editiermodus gewechselt werden. Über die Funktionen „Edit“ und „View Clusters“ kehrt man wieder aus dem Editiermodus zurück.

Im Editiermodus werden die selektierten Cluster als neue Punktwolke dargestellt. Das Clustering kann nun, wie bereits beschrieben, erneut mit anderen Dichteparametern durchgeführt werden. Ist das Ergebnis des Clusterings zufriedenstellend, so werden durch das Betätigen des „Ok“ Buttons im Anzeigefenster die neu gefundenen Cluster in die vorherige Darstellung übernommen. Dabei kann es aufgrund von Farbneuzuordnungen zu einer anderen Markierung der Cluster kommen.

Liegt eine Punktwolke in Form ihrer Cluster vor, so kann die Objekterkennung durchgeführt werden. Sind die Parameter für die Objektbeschreibung nicht bekannt, so kann der Trainingseditor genutzt werden. Zu finden ist dieser im Menü „Edit“ unter dem Menüpunkt „Trainingseditor“. Mit dem „Add“ Button können Trainingsobjekte, die als Punktwolke im Dateiformat xyz oder txt vorliegen, geladen werden. Für jede hinzugefügte Punktwolke werden die Abmessungsparameter und die Anzahl der Punkte als Merkmale extrahiert und in der Tabelle des Trainingseditors angezeigt. Einzelne Trainingsobjekte können mittels des „Remove“ Buttons entfernt werden. Die Funktion „Clear“ entfernt alle Trainingsobjekte der Liste.

Für die Menge aller Trainingsobjekte werden die Minima und Maxima der Abmessungsparameter längste Seite, mittlere Seite und kürzeste Seite sowie die Anzahl der Punkte bestimmt und in dem entsprechenden Tabellenfeld dargestellt. Mit der Auswahl „Tolerance“ kann ein prozentualer Toleranzwert für die Abmessung und die Anzahl der Punkte angegeben werden. Durch Drücken des „Ok“ Buttons werden die Parameter als solche in die Objektbeschreibung übernommen. Mit Betätigen des „Do Object Recognition“ Buttons im Hauptfenster wird die

Objekterkennung ausgeführt und es werden alle Cluster markiert, die der Objektbeschreibung entsprechen.

Unter dem Menüpunkt „View“ kann der Nutzer zwischen den folgenden Ansichten der Punktwolke wählen:

- **Point Cloud** zeigt die Punktwolke bzw. eine vordefinierte Stichprobe an Punkten der Punktwolke
- **All Clusters** zeigt alle durch das Clustering gefundenen Cluster an
- **Matched Clusters** zeigt alle Cluster an, die durch das Matching gefunden wurden, das heißt die Objektbeschreibungsparametern entsprechen
- **Non-matched Clusters** zeigt alle Cluster an, die nicht den Objektbeschreibungsparametern entsprechen
- **Octree** zeigt die Octree-Datenstruktur und die Punktwolke an.

Das Programm bietet die Möglichkeit, die Cluster der Punktwolke als xyz- oder xml-Datei zu speichern. Dabei besteht für die Cluster eine Auswahlmöglichkeit zwischen All Clusters, Matched Clusters und Non-matched Clusters.

6. Test und Evaluierung

In diesem Kapitel wird das entwickelte Programm „ORIPUCA“ und das damit umgesetzte Verfahren der Objekterkennung in Punktwolken auf der Grundlage des DBSCAN-Algorithmus in seiner Anwendung getestet und evaluiert. Der Schwerpunkt liegt in der Untersuchung der korrekten Objekterkennung anhand der vorgegebenen Abmessungsparameter von Beispieldaten. Da von der Korrektheit des DBSCAN-Algorithmus ausgegangen wird, reicht ein Testdurchlauf, um die richtige Implementierung zu überprüfen. Am Beispiel einer Punktwolke, für die unterschiedliche Punktdichten generiert werden, soll die Laufzeit und der Speicherverbrauch ermittelt werden.

Im Folgenden werden die durchgeführten Tests erläutert und die dafür verwendeten Beispieldaten vorgestellt. Für den Test liegen die Beispieldaten in zwei Datensätzen vor. Die Punktwolken aus Datensatz 1 sind alle künstlich generiert. Die Punktwolken aus Datensatz 2 wurden von einem Laserscanner generiert. In Kapitel 6.1 werden die Testergebnisse mit den künstlich generierten Beispieldaten aus Datensatz 1 und in Kapitel 6.2 anhand der Aufnahmen des Laserscanners dokumentiert.

6.1 Test am Beispiel künstlich generierter Punktwolken

In einem ersten Testfall soll die korrekte Implementierung des DBSCAN-Algorithmus getestet werden. Da von einer korrekten Funktionsweise des Algorithmus ausgegangen wird, wird hierfür nur ein Test durchgeführt.

In drei weiteren Testfällen soll das Verfahren der Objekterkennung untersucht werden.

Testfall 1 besteht aus zwei unterschiedlich großen Quadern, die die gleiche Punktdichte aufweisen. Im ersten betrachteten Fall ist der Abstand d zwischen den

zwei Quadern geringfügig kleiner als die Punktdichte. Im zweiten Fall wird der Abstand *d* so gewählt, dass er geringfügig größer ist als die Punktdichte.

Für den ersten Fall wird erwartet, dass durch den DBSCAN-Algorithmus ein Cluster gefunden wird, der beiden Quadern entspricht. Im zweiten Fall sollen zwei Cluster ermittelt werden, die jeweils einem Quader entsprechen.

Testfall 2 besteht aus zwei gleichen Quadern mit den jeweiligen Maßen 10 cm x 6,25 cm x 4,25 cm. Der Abstand beider Quader zueinander ist größer als die Grenzdichte, so dass die Clusteranalyse zwei Cluster ergeben sollte. Als Eingabeparameter für die Objekterkennung dienen die Abmessungsparameter der Quader sowie ein Toleranzwert von 1 %.

Testfall 3 besteht aus drei unterschiedlichen Quadern. Sie sind willkürlich im Raum angeordnet und haben folgende räumliche Abmessungen:

- Quader 1: 10,4 cm x 6,0 cm x 4,0 cm
- Quader 2: 10,0 cm x 6,0 cm x 4,0 cm
- Quader 3: 9,5 cm x 6,0 cm x 4,0 cm

Als Eingabeparameter dienen die Abmessungsparameter der Quader. Es wird erwartet, dass für die entsprechende Eingabe der jeweilige Cluster gefunden wird.

Testfall 4 besteht aus einem Quader und einem Objekt, das aus einem Schnitt entlang einer Ebene quer durch den Quader hervorgegangen ist (siehe Abbildung 6.4). Das Objekt besitzt dieselben Abmessungsparameter in Bezug auf die längste, mittlere und kürzeste Seite.

Es wird erwartet, dass die minimalen Bounding Boxen beider Objekte die gleichen Abmessungen besitzen und somit erkannt werden.

Nachdem die Testfälle anhand der künstlich generierten Punktwolken vorgestellt wurden, folgt nun die Auswertung der Tests.

Die Abbildung 6.1 zeigt das Ergebnis für Testfall 1. Im hinteren Teil der Abbildung ist zu erkennen, dass die beiden Quader als ein Cluster erkannt wurden. Rechts sind zwei unterschiedlich markierte Cluster abgebildet.

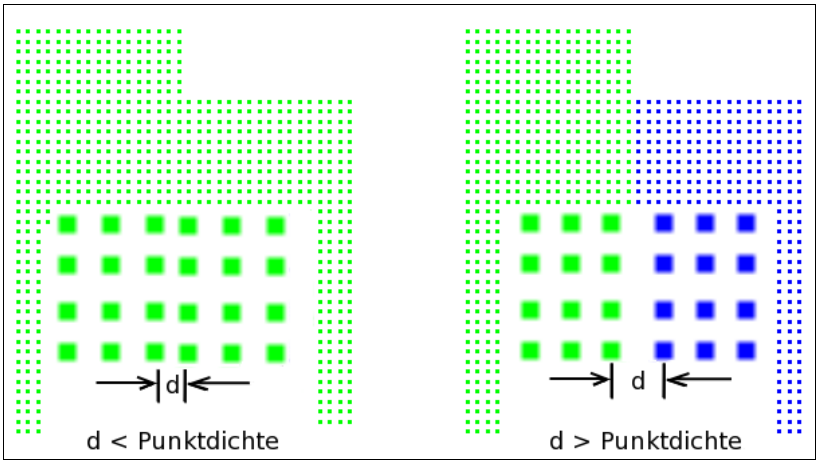


Abb. 6.1 *Testfall 1:*
Links: DBSCAN findet für zwei Quader ein Cluster, da der Abstand d der Cluster zueinander kleiner als die Grenzdichte ist
Rechts: DBSCAN findet zwei Cluster, da der Abstand d hier größer als die Grenzdichte ist

Das Ergebnis des Testfalls 1 bestätigt die korrekte Implementierung des DBSCAN-Algorithmus.

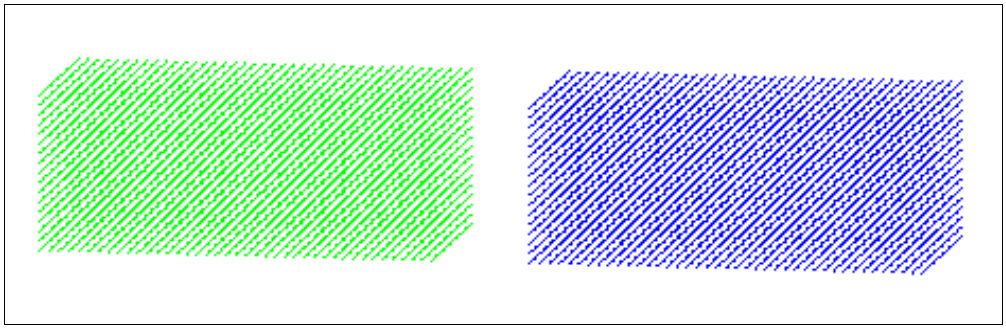


Abb. 6.2 *Testfall 2: Zwei Objekte mit den gleichen Abmessungen wurden korrekt gefunden*

In Testfall 2 (vgl. Abbildung 6.2) ergab die Clusteranalyse die zwei erwarteten Cluster. Der grün dargestellte Cluster wurde als Trainingsobjekt gewählt und dessen Abmessungsparameter mit 1 % Toleranz als Parameter für die Objektbeschreibung

übernommen. Die Objekterkennung hat die zwei Cluster anhand der Abmessungen richtig erkannt.

In Testfall 3 (vgl. Abbildung 6.3) ergab die Clusteranalyse die drei erwarteten Cluster. Für jeden Cluster wurde das entsprechende Trainingsobjekt geladen und die Werte der Abmessungen mit 1 % Toleranz als Parameter für die Objektbeschreibung übernommen.

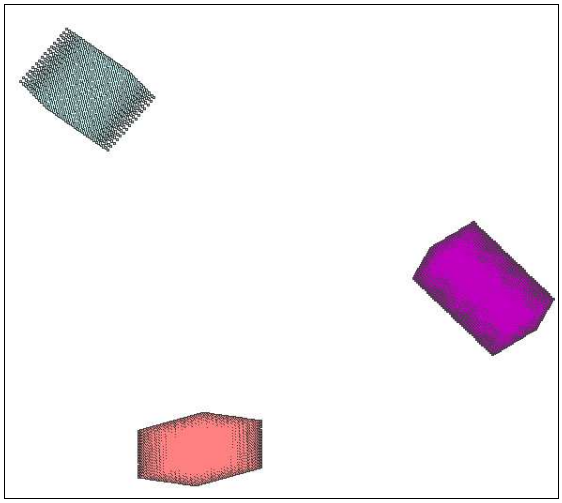


Abb. 6.3 *Testfall 3: Die drei Cluster unterscheiden sich in der Länge der längsten Seite. Diese beträgt bei Quader 1 (blau) 10,4 cm, bei Quader 2 (lila) 10,0 cm und bei Quader 3 (rosa) 9,5 cm.*

Die jeweils durchgeführte Objekterkennung fand jeweils das richtige Objekt.

In Testfall 4 (vgl. Abbildung 6.4) fand die Clusteranalyse die erwarteten zwei Cluster.

Abb. 6.4 *Testfall 4: Links ein Quader (grün) und rechts ein abgeschrägter Quader (blau) mit den gleichen Abmessungen wie der linke Quader.*

Die Parameter für die Objektbeschreibung wurden über die Abmessungsparameter des Quaders ermittelt. Jedoch wird durch das Matching nur der Quader erkannt. Die Ursache liegt darin, dass für das abgeschrägte Objekt nicht die optimale minimale Bounding Box bestimmt wurde, sondern eine Approximation, wie sie in der Abbildung 6.5 dargestellt ist.

Bei einer Abmessung von 10,0 cm x 6,0 cm x 4,0 cm wird eine angenäherte, minimale Bounding Box von 10,67 cm x 6,0 cm x 4,37 cm bestimmt.

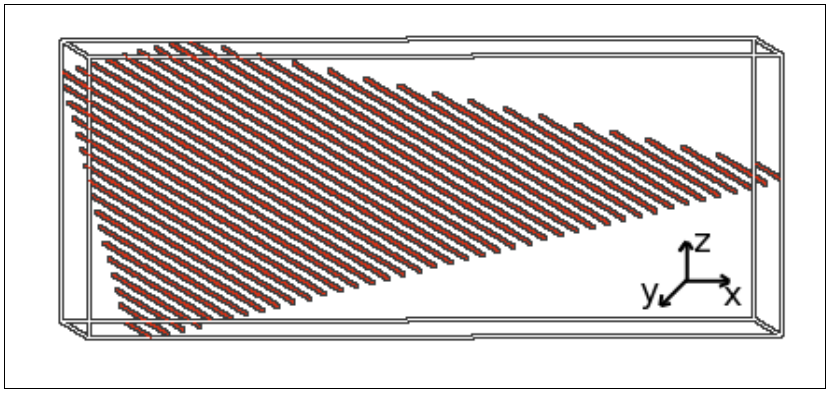


Abb. 6.5 *Testfall 4: Für den abgeschrägten Quader wird eine approximative, nicht optimale minimale Bounding Box bestimmt*

6.2 Test anhand von Beispieldaten aus der realen Welt

Datensatz 2 besteht aus mehreren Ausschnitten einiger Apfelbäume, die auf einer Baumplantage in den Niederlanden mit einem 3D-Laserscanner aufgenommen wurden. Der Lehrstuhl Optical and Laser Remote Sensing der TU Delft, Niederlande, hat sie für diese Arbeit zur Verfügung gestellt. Der Datensatz besteht aus insgesamt fünf unterschiedlich großen Punktwolken, die sich jeweils in der Anzahl der durch sie repräsentierten Objekte unterscheiden. Die Anzahl der Objekte wird durch eine Menge an Blättern und Zweigen gebildet. Diese treten in unterschiedlichster Form, Größe, Lage, Anzahl an Punkten und der Qualität der Aufnahme auf.

Der Test der vorliegenden Beispieldaten soll das in dieser Arbeit beschriebene Verfahren auf die Anwendbarkeit auf Punktwolken, die von einem Laserscanner generierte Ausschnitte aus der realen Welt darstellen, untersuchen. Dazu wird zunächst überprüft, wie gut sich das dichtebasierte Clustering zur Segmentierung dieser Punktwolken eignet. Anschließend wird das Matching getestet.

6.2.1 Test der Anwendbarkeit der dichtebasierten Clusteranalyse

Die Punktwolke 1 aus Datensatz 2 besteht aus 5.255 Punkten. Sie enthält als Objekte ein Blatt und das Teilstück eines Zweiges. Das Clustering wird für verschiedene Dichteparameter MinPts und ε durchgeführt. Betrachtet werden die Anzahl der gefundenen Cluster und die Anzahl der Punkte, die dem Rauschen zugeordnet werden.

Ziel dieses Tests ist es, eine Aussage darüber treffen zu können, welchen Einfluss die Dichteparameter auf das Ergebnis der Clusteranalyse haben. Aus Tabelle 6.1 geht hervor, dass das Ergebnis des Clusterings von der durch die Parameter MinPts und ε spezifizierten Grenzdichte abhängig ist. Mit steigender Grenzdichte werden mehr Punkte einem Cluster zugeordnet und die Anzahl der gefundenen Cluster nimmt ab.

Das Ergebnis lässt sich in drei Bereiche einteilen. Im Bereich 1 wird ein Objekt der Punktwolke in mehrere kleinere Cluster unterteilt. Im Bereich 2 repräsentiert jedes gefundene Cluster der Punktwolke genau ein Objekt. Im Bereich 3 werden mehrere Objekte zu einem Cluster zusammengefasst.

MinPts	ε [cm]	Anzahl gefundener Cluster	Anzahl der Rauschpunkte
1	0.164	358	505
1	0,192	180	284
1	0,229	53	92
1	0,224	43	73
1	0,304	17	31
5	0.30	5	130
5	0,44	2	14
5	0,58	2	1
5	0,72	2	0
5	0,86	1	0
10	0.30	5	323
10	0,44	2	25
10	0,58	2	7
10	0,72	2	0
10	0,86	2	0
15	0.30	3	322
15	0,44	2	47
15	0,58	2	15
15	0,72	2	3
15	0,86	2	0

Tab. 6.1 *Anzahl der gefundenen Cluster und Anzahl der Rauschpunkte für unterschiedliche Werte von MinPts und* ε

Die Abbildung 6.6 zeigt die Punktwolke 2. Ein zufriedenstellendes Clustering für MinPts gleich 5 und ε gleich 0,44 cm ist in Abbildung 6.7 dargestellt.

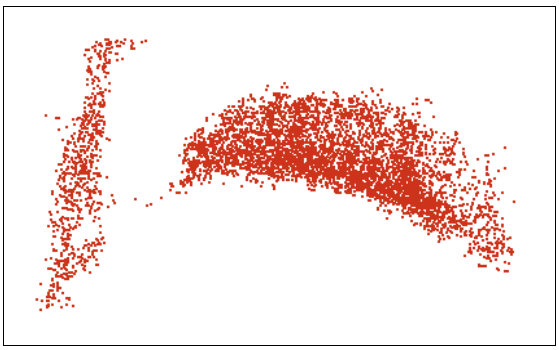


Abb. 6.6 *Punktwolke 1*

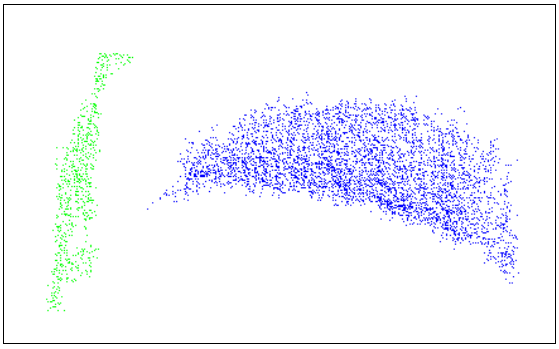


Abb. 6.7 *Die gefundenen Cluster der Punktwolke 1 bei einer Clusteranalyse mit MinPts = 5 und ε = 0,44 cm*

Die Abbildung 6.8 zeigt die Punktwolke 2 aus dem Datensatz 1. Sie enthält 23.743 Punkte. Als Objekte liegen mehrere Blätter und einige kleine Äste vor. Auffällig ist, dass mehrere Objekte nur unvollständig in der Punktwolke dargestellt werden.

Das Clustering wurde für mehrere unterschiedliche Dichteparameter durchgeführt. Das Ergebnis der Clusteranalyse für MinPts gleich 5 und ε gleich 0,603 cm zeigt Abbildung 6.9. Es werden 18 Cluster gefunden und insgesamt 39 Punkte dem Rauschen zugeordnet. Außer dem gelb und dem rosa dargestellten Cluster repräsentieren alle Cluster, die räumlich ausreichend voneinander getrennt sind, ein Objekt. Das gelbe Cluster beinhaltet zwei Blätter, deren Punkte für das durchgeführte Clustering zu nah beieinander liegen, so dass diese für ε gleich 0,603 cm nicht getrennt werden konnten. Analog verhält es sich bei dem rosa dargestellten Cluster.

Für diese beiden Cluster ist ein Clustering mit geringeren Dichteparametern erforderlich. Das Programm „ORIPUCA“ bietet diese Möglichkeit.

Abb. 6.8 *Punktwolke 2*

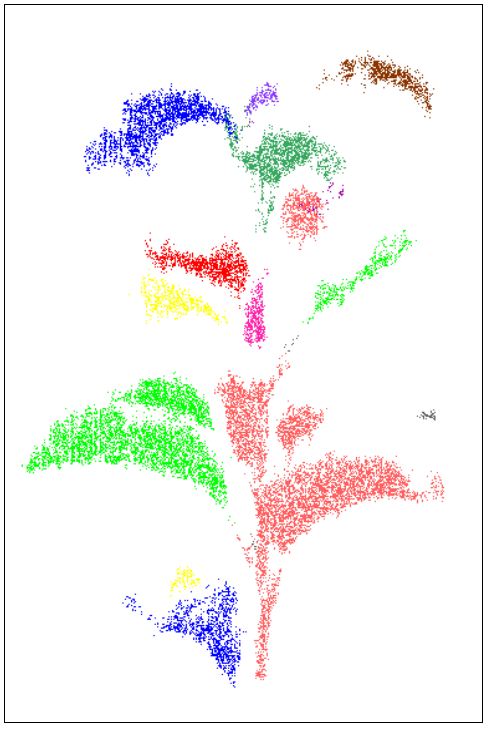


Abb. 6.9 *Die gefundenen Cluster der Punktwolke 2 bei einer Clusteranalyse mit MinPts = 5 und ε = 0,603 cm.*

Für den gelben Cluster ergibt sich bei MinPts gleich 5 ein Wert für ε von 0,233 cm, bis die zwei Blätter getrennt werden. Neben diesen beiden Clustern werden noch drei weitere Cluster gefunden, deren Punktanzahlen jedoch sehr gering sind. Von 5.271 Punkten, die zu dem gelben Cluster gehören, werden 198 Punkte dem Rauschen zugeordnet.

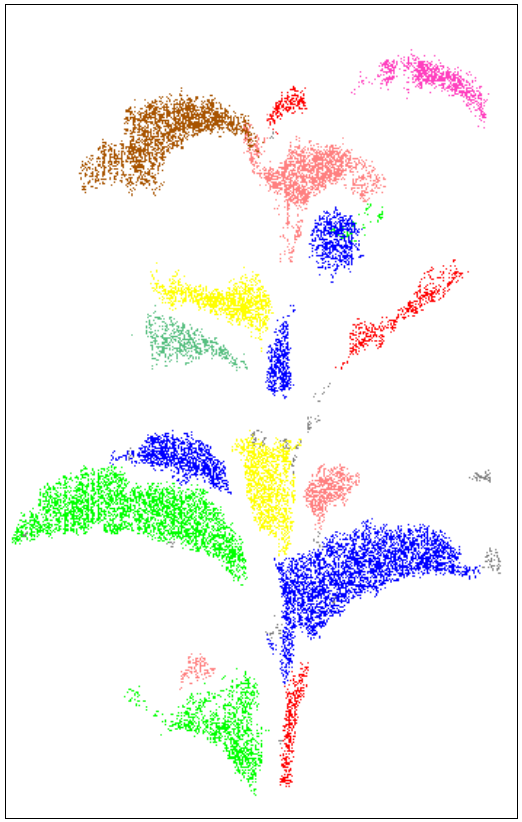
Für den rosa dargestellten Cluster wird ein zufriedenstellendes Clustering bei ε gleich 0,246 cm erreicht. Es werden 17 Cluster, davon 14 mit einer geringen Punktzahl von unter 50 Punkten, gefunden. Von 7.226 Punkten werden 321 Punkte dem Rauschen zugeordnet. Abbildung 6.11 zeigt alle gefundenen Cluster der Punktwolke 2 nach dem erneuten individuellen Clustering des gelben und rosa Clusters.

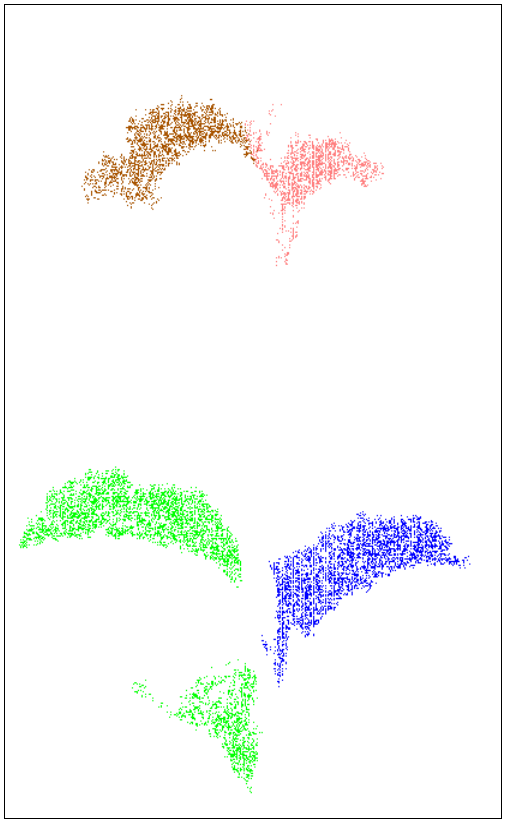
Dieser Test zeigt, dass die Segmentierung einer Punktwolke sehr stark von dem Abstand der Punkte innerhalb eines Objekts und dem Abstand der Objekte zueinander abhängig ist. Das heißt, eine Punktwolke muss die Voraussetzung erfüllen, dass die durch sie repräsentierten Objekte genügend voneinander getrennt sind. Erst dadurch wird eine geeignete Segmentierung ermöglicht. Dies ist wiederum die Voraussetzung für das anschließende Matching auf der Grundlage der Abmessungsparameter.

6.2.2 Test der Anwendbarkeit des Matching-Verfahrens

Der Test ist so aufgebaut, dass anhand mehrerer Trainingsobjekte die Abmessungsparameter für die zu suchenden Cluster ermittelt werden. Diese dienen als Eingabe für die Objekterkennung. Durch das Matching sollen alle Cluster gefunden werden, die diese Abmessungsparameter erfüllen.

Als Beispiel 1 wurden die gefundenen Cluster der Punktwolke 2 aus dem vorherigen Abschnitt verwendet. Der gelbe und der rosa Cluster wurden, wie oben beschrieben, weiter differenziert. Die Abbildung 6.10 zeigt in der linken Darstellung die Cluster der Punktwolke.





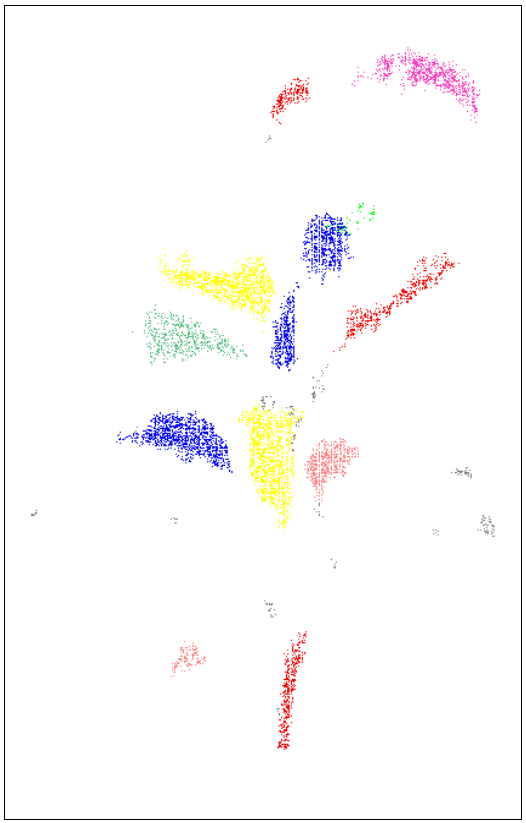


Abb. 6.10 Links: *Die Cluster der Punktwolke 2*
Mitte*: Die durch das Matching erkannten Cluster*
Rechts: *Cluster, die nicht der Objektbeschreibung entsprechen*

Es wurden sieben Trainingsobjekte verwendet, die extrahierte Blätter aus den Punktwolken des Datensatzes 2 repräsentieren. Die Auswahl beschränkte sich auf Blätter, die aufgrund ihrer Größe und Vollständigkeit optisch gut zu erkennen sind. In der Punktwolke sollen im Test alle entsprechenden Cluster markiert werden, die den Abmessungsparametern der Objektbeschreibung entsprechen. Als Parameter ergeben sich bei einer im vorgegebenen Toleranzbereich von 5 % die folgenden Abmessungen für die zu erkennenden Objekte:

- minimale Anzahl an Punkten: 491
- Längste Seite: von 6,82 cm bis 10,35 cm
- Mittlere Seite: von 2,88 cm bis 6,02 cm
- Kürzeste Seite: von 1,43 cm bis 4,68 cm

Der mittlere Teil der Abbildung 6.10 zeigt die Cluster, die durch das Matching erkannt wurden, weil sie den Parametern der Objektbeschreibung entsprechen. Diese repräsentieren Blätter, die auch optisch als große und recht vollständige Blätter zu erkennen sind. Der rechte Teil der Abbildung 6.10 zeigt die Cluster, die nicht der Objektbeschreibung entsprechen. Es ist in der Abbildung gut erkennbar, dass dort keine großen Blätter dargestellt werden.

Mit diesem Test wurde eine Anwendung simuliert, in der in einer vorgegebenen Punktwolke nach bestimmten Objekten gesucht wurde, welche sich aufgrund ihrer Abmessungen von anderen Objekten unterscheiden und markiert werden sollen.

In einem zweiten Test müssen in einer Punktwolke, die ebenfalls einen Zweig eines Baumes repräsentiert, möglichst alle Blätter erkannt werden. Dazu wurden auch kleinere und unvollständige Blätter als Trainingsobjekte extrahiert.

Als Objektbeschreibungsparameter ergeben sich bei einer vorgegebenen Toleranz von 5 % für die zu erkennenden Objekte die folgenden Abmessungen:

- minimale Anzahl an Punkten: 466
- Längste Seite: von 3,26 cm bis 10,35 cm
- Mittlere Seite: von 1,89 cm bis 6,02 cm
- Kürzeste Seite: von 1,43 cm bis 5,20 cm

Das Ergebnis der Objekterkennung ist in Abbildung 6.11 dargestellt. Der linke Teil der Abbildung dokumentiert alle gefundenen Cluster. In der Mitte sind die Cluster abgebildet, die durch das Matching erkannt wurden und rechts die Cluster, die nicht den Parametern der Objektbeschreibung entsprechen.

In diesem Testdurchlauf konnten aus einer Punktwolke alle Blätter extrahiert werden, die einer vordefinierten Objektgröße entsprachen. Sehr kleine und unvollständige Blätter sowie die Stiele des Zweiges konnten entfernt werden.

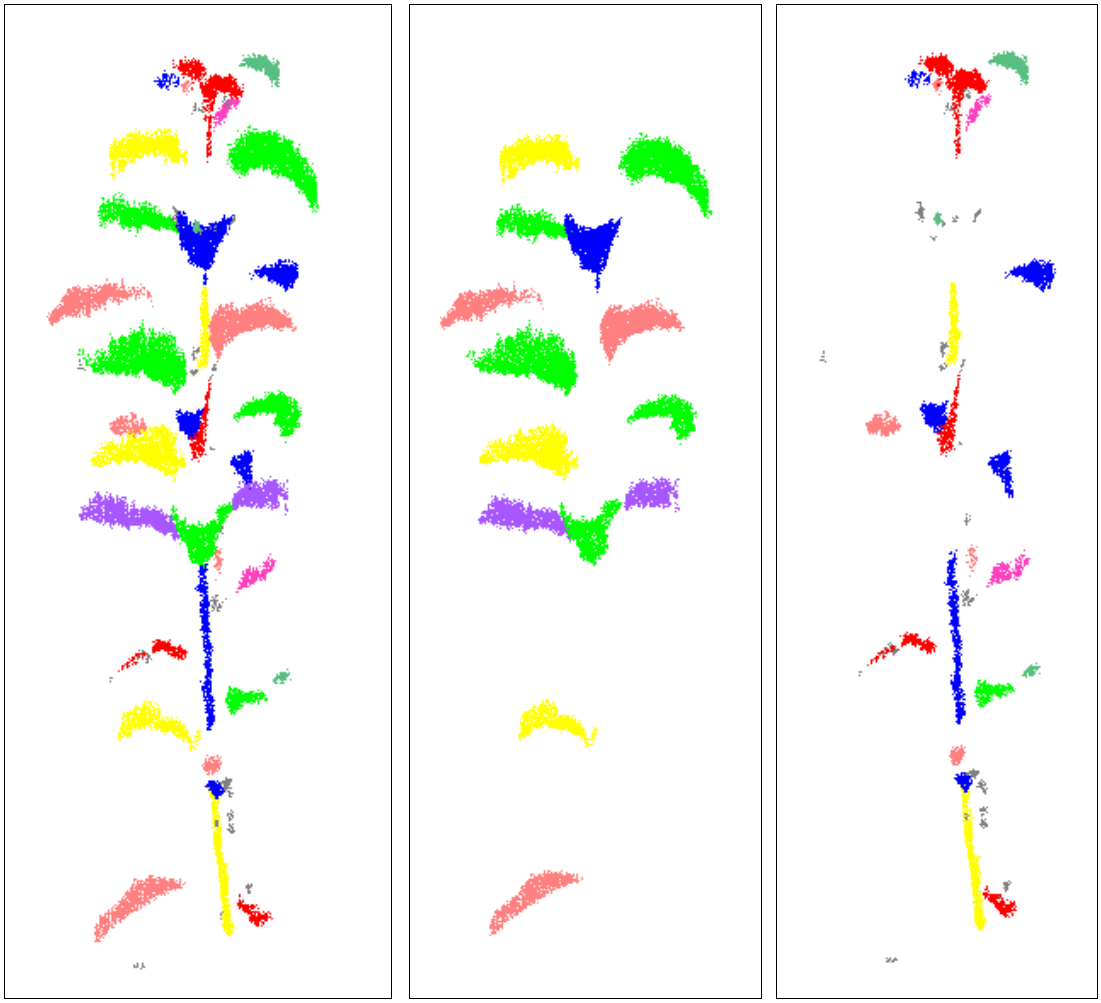


Abb. 6.11 *Links: Alle gefundenen Cluster der Punktwolke*
Mitte: Durch das Matching erkannte Cluster
Rechts: Cluster, die nicht der Objektbeschreibung entsprechen

6.3 Laufzeitverhalten

In einem weiteren Test soll das Laufzeitverhalten am Beispiel eines Quaders mit unterschiedlicher Punktdichte bei konstanter räumlicher Ausdehnung ermittelt werden.

Es wurde ein Quader mit einer räumlichen Ausdehnung von 120 cm x 80 cm x 60 cm generiert, der um 45° in Richtung der x-Achse rotiert wurde. Die Ergebnisse der Laufzeit sind grafisch in der Abbildung 6.12 dargestellt.

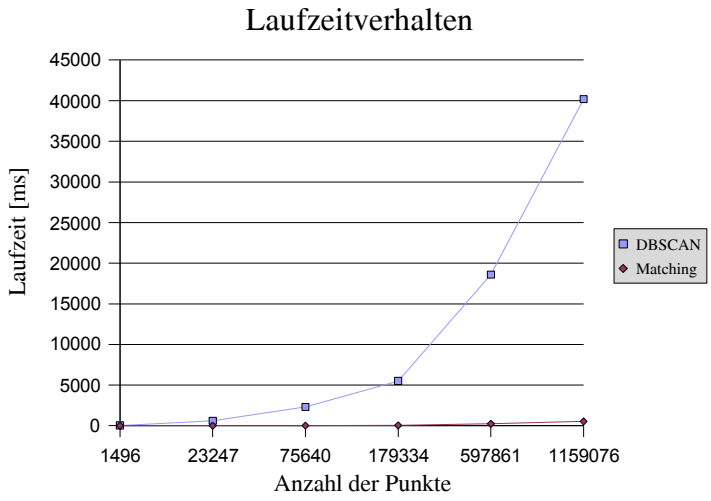


Abb. 6.12 *Laufzeitverhalten für eine Punktwolke mit konstanter räumlicher Ausdehnung und unterschiedlicher Punktdichte*

Der Verlauf der Funktion entspricht der erwarteten O(n * log n)-Funktion.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Der in dieser Arbeit implementierte DBSCAN-Algorithmus findet in einer dreidimensionalen Punktwolke sämtliche Cluster, deren Dichte größer ist, als die durch die Parameter MinPts und ε spezifizierte Schwellenwertdichte. Der Algorithmus ist relativ unempfindlich gegenüber Rauschen. Seine Schwachstelle liegt darin, dass nicht alle Cluster mit unterschiedlichen Punktdichten gefunden werden. Dieses Problem wurde in dieser Arbeit durch das Bearbeiten und das unabhängige Clustering einzelner Cluster gelöst.

Repräsentiert jeder Cluster der Punktwolke ein Objekt der realen Welt, werden alle Objekte gefunden und markiert, die vorgegebene Abmessungsparameter einhalten. Die Bestimmung der Parameter beruht auf der Berechnung einer approximativen minimalen Bounding Box für jeden Cluster.

Für Anwendungen, in denen die Objekte in der Punktwolken-Repräsentation genügend voneinander getrennt sind und in denen die Abmessungen der Objekte bekannt sind oder durch Trainingsobjekte bestimmt werden können, stellt das in dieser Arbeit beschriebene und implementierte Verfahren eine einfache Möglichkeit der Objekterkennung dar.

Eine Schwachstelle des Verfahrens liegt darin, dass Objekte nicht aufgrund ihrer Form erkannt werden können.

Die Berechnung der minimalen Bounding Box ist nur eine Annäherung. Dadurch wird für einige Cluster nicht die optimale minimale Bounding Box berechnet. Bei sehr genauen Angaben der Abmessungsparameter kann es dann zu Fehlern bei der Objekterkennung kommen.

7.2 Ausblick

Da das Matching keine Formen der Objekte berücksichtigt, kann dieses erweitert oder durch ein geeigneteres Verfahren der Objekterkennung ersetzt werden.

Der Großteil der benötigten Rechenzeit wird durch das Clustering beansprucht. Deshalb sind Effizienzsteigerungen vor allem im Bereich der Bereichsanfrage sinnvoll. Wie im Abschnitt 3.3.4 dargelegt, stellt die in dieser Arbeit verwendete Octree-Datenstruktur eine einfach zu implementierende Unterstützung der Bereichsanfrage dar, diese ist jedoch nicht optimal. Wird für die Anfrageunterstützung eine Datenstruktur verwendet, die einen direkten Zugriff auf die ε-Nachbarschaft eines Punktes ermöglicht, so verringert sich die Laufzeitkomplexität des DBSCAN-Algorithmus auf $O(n)$.

Das Verfahren der Hauptkomponentenanalyse bestimmt für eine vorliegende Punktwolke nur eine approximative minimale Bounding Box. Dadurch ist bei der Objekterkennung mittels der Abmessungsparameter mit Fehlern zu rechnen. In dieser Arbeit erfolgte keine genaue Untersuchung, wie sehr sich diese Fehler auf das Gesamtergebnis der Objekterkennung auswirken.

A. Literaturverzeichnis

[BES85] Besl P. J., Jain R. C.: Three-dimensional object recognition, ACM Computing Surveys (CSUR), volume 17 number 1, page 75-145, March 1985.

[EST96] Ester M., Kriegel H.-P., Sander J., Xu X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", Proc. 2nd int. Conf. on Knowledge Discovery and Data Mining (KDD '96), Portland, Oregon, 1996, AAAI Press, 1996.

[EST00] Ester, M.; Sander, J.: Knowledge Discovery in Databases. Techniken und Anwendungen. Springer Verlag, Berlin Heidelberg, 2000.

[FAR05a] Faro: Laser Scanner LS Broschüre: URL: http://www.iqvolution.com/de/Data/Downloads/LS880_DE.pdf (Abfrage: 03.12.2006).

[FAR05b] Faro: FARO LS 880 Technische Spezifikationen: URL: http://www.iqvolution.com/de/Data/Downloads/TS_DE.pdf (Abfrage: 12.11.2006).

[LIN02] Lindsay I. Smith: A tutorial on Principal Components Analysis. 26.02.2002. URL: http://csnet.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf (Abfrage: 12.05.2007).

[PRE02] Press W.-H., Teukolsky S. A., Vetterling W. T., Flannery B. A.: Numerical Recipes in C++. The Art of Scientific Computing. Second Edition. Cambridge University Press. 2002.

[SAM90] Samet H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley Publishing Company Inc., 1990. (Reprinted 1994).

[SON99] Sonka M., Hlavac V., Boyle R.: Image Processing, Analysis, and Machine Vision. 2nd Edition. Brooks/Cole Publishing Company. Pacific Grove, Albany, Bonn, u. a., 1999.

[TAN05] Tan P.-N., Steinbach M., Kumar V.: Introduction to Data Mining. First Edition. Addison Wesley, 8. Kapitel: Cluster Analysis: Basic Concepts and Algorithm, 2005.

[TOE94] Tönnies K. D., Lemke H. U.: 3D-Computergrafische Darstellungen. Handbuch der Informatik, hrsg. v. Endres A., Krallmann H., und Schnupp P., Band 9.2, R. Oldenbourg Verlag, München, Wien, 1994.

[WOE05] Wölfelschneider H., Blug A., Baulig C., und Höfler H.: Schnelle Entfernungsmessung für Laserscanner. Technisches Messen 72 (2005) 7-8. Oldenbourg Verlag, 2005.

[WU92] Wu, X: A linear-time simple bounding volume algorithm. In *Graphics Gems III*, D. Kirk, Ed. Academic Press Graphics Gems Series. Academic Press Professional, San Diego, CA, USA. page 301-306, 1992.

B. Klassendiagramm