

Analyse von Pflanzenwachstum auf Basis von 3D-Punktwolken

Masterarbeit
zur Erlangung des Grades *Master of Science*

an der
Hochschule Niederrhein
Fachbereich Elektrotechnik und Informatik
Studiengang *Informatik*

vorgelegt von
Jakob Görner
1003660

Datum: 28. September 2021

Prüfer: Prof. Dr. Regina Pohle-Fröhlich
Zweitprüfer: Prof. Dr. Christoph Dalitz

Eidesstattliche Erklärung

Name: Jakob Görner

Matrikelnr.: 1003660

Titel: Analyse von Pflanzenwachstum auf Basis von 3D-Punktwolken

Ich versichere durch meine Unterschrift, dass die vorliegende Arbeit ausschließlich von mir verfasst wurde. Es wurden keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt.

Die Arbeit besteht aus 28 Seiten.

Mönchengladbach, 28. September 2021

Unterschrift

Zusammenfassung

In dieser Arbeit soll eine Anwendung zum Analysieren von Pflanzen-Wachstum und weiteren Merkmalen des Wachstumsprozesses einer Pflanze erstellt werden. Es soll aus einer Reihe Bilder einer Pflanze, die mit einer gängigen Smartphone-Kamera aufgenommen sind, eine Punktwolke erzeugt werden. Aus der Punktwolke sollen die Punkte die zur Pflanze gehören extrahiert werden. Die extrahierten Punkte werden zur weiteren Analyse in Stamm und Blatt Punkte segmentiert. Des weiteren sollen Werte wie das Wachstum der Pflanze ermittelt werden. Hierzu müssen mehrere Punktwolken über Zeit miteinander verglichen werden. Die Anwendung soll mit möglichst wenig Bildern auskommen um den Datentransfer zu minimieren.

Abstract

Development of an application made to analyse plant growth over time and further characteristics of a plant represented as a point cloud. The point cloud should be generated out of a set of images of the plant, taken with a common smartphone. The point cloud should be splitted in leave and stem points to count leaves and analyse the stem further. Also other characteristics as the hight should be analysed. For this reason multiple point clouds has to be compared over time. The application should use as few pictures as possible to reduce the amount of transferred data.

Inhaltsverzeichnis

1 Motivation	3
2 Stand der Technik	4
2.1 Generierung einer 3D Punktwolke aus Bildern	4
2.2 Segmentierung von 3D-Punktwolken	4
2.3 Registrierung von 3D-Punktwolken	5
3 Realisierung	7
3.1 Architektur	7
3.2 Umsetzung Generierung einer 3D Punktwolke aus Bildern	7
3.3 Umsetzung Registrierung zweier Punktwolken	8
3.4 Umsetzung Segmentierung	9
3.5 Umsetzung Server	11
3.6 Umsetzung Pipelines	13
4 Ergebnisse	16
4.1 Vergleich von Verfahren zur Generierung von 3D-Punktwolken auf Basis von Bildern	16
4.2 Vergleich von Verfahren zur Registrierung 3D-Punktwolken	19
4.3 Vergleich von Verfahren zur Segmentierung von Pflanzen auf 3D-Punktwolken	19
5 Fazit und Ausblick	24
Literatur	26

1 Motivation

Diese Arbeit wird im Rahmen eines Projekts mit den Universitäten Makerere University (Kampala, Uganda) und Central University of Technology (Bloemfontein, South Africa) durchgeführt. Im Rahmen dieser Arbeit soll eine Anwendung entstehen, die in den beiden Ländern für Kleinbauern zum Einsatz kommen soll. Die Anwendung soll mit nicht invasiven Methoden zur Unterstützung bei der selektiven Züchtung eingesetzt werden, um Erträge und somit auch Gewinnmargen zu erhöhen und damit zur Ernährungssicherheit beizutragen. Dies ist nötig, da es im Rahmen der derzeitigen Klimaveränderungen zu höheren Ernteausfällen durch Dürren kommt [1] die durch reinen Einsatz von indigenem Wissen nicht zu kompensieren sind.

Um diesem Ziel näher zu kommen soll in dieser Arbeit eine Software entwickelt werden die es erlaubt Daten über den Wachstums-Prozess von Pflanzen zu sammeln. Diese soll als Grundlage für weiter Analysen wie das frühe Erkennen von Krankheitsbildern, Ertragsschätzungen oder Erkennen unzureichende Wachstumsraten der Pflanzen dienen.

Ziel ist es also eine Software zu schaffen die es ermöglicht aus Bildern von Pflanzen 3D Punktwolken (Punktwolken) zu generieren und diese zu analysieren. Es soll mit der zu entwickelnden Anwendung möglich sein Aussagen über das Wachstum über Zeit, die Entwicklung der Blätter und der Stämme zu treffen. Mit diesen Daten sind Analysen möglich die auf das Wachstum einer Pflanze unter bestimmten Bedingungen schließen lassen. Hierbei gibt es besondere Anforderungen an den Prozess der Datengewinnung und Übertragung. Zum einen muss der Betrieb kostengünstig sein - das betrifft insbesondere den Endverbraucher. Der Endverbraucher sollte möglichst wenig Datenmengen an den Server, der die Rechenkapazität bereit stellt übertragen müssen. Das soll die Kosten auf Seiten der Kleinbauern gering halten. Des weiteren muss die Datengewinnung mit einem Mobiltelefon oder einer anderen Kamera möglich sein. Spezielle Aufnahmegeräte, wie ein LIDAR-Scanner, die den Kleinbauern nicht schon zur Verfügung stehen sollten nicht nötig sein.

2 Stand der Technik

Drei Kernprobleme müssen betrachtet werden die für den Erfolg diese Arbeit gelöst werden müssen. Zuerst muss aus einer Menge an Bildern eine Punktwolke generiert werden. Danach muss die Punktwolke Segmentiert werden um den Hintergrund zu entfernen und die einzelnen Teile der Pflanze wie Blätter und Stiele zu erkennen und weiter zu analysieren. Zuletzt müssen zwei Punktwolken einer Pflanze zu verschiedenen Zeitpunkten miteinander registriert werden um Aussagen über das Wachstum einer Pflanze treffen zu können.

2.1 Generierung einer 3D Punktwolke aus Bildern

Generell kann man zwischen zwei Methoden der Generierung von Punktwolken unterscheiden, die jeweils verschiedene Verfahren beziehungsweise Techniken anwenden.

Die erste Methode nutzt Hardware wie LIDAR-Scanner [2] um Punktwolken direkt zu erzeugen. Es gibt eine Reihe von Sensoren die das ermöglichen. Da der Großteil der Sensoren aus kostengründen nicht in Frage kommt sei hier nur noch der RGB-D erwähnt. Dieser liefert neben den Farbwerten für ein Bild auch noch eine Tiefeninformation für jeden Pixel. In einigen Smartphone-Modellen wird bereits ein solcher Sensor verbaut [3].

Die andere Methode ist Structure from Motion (SfM), bei der die Punktwolke aus Bildern, einer Szene aus verschiedenen Perspektiven, generiert wird. Allgemein werden auf den zugrundeliegenden Bildern nach Merkmalen (Feature) gesucht die robust gegen Translation, Rotation, Skalierung und Beleuchtung sind. Hier können SIFT [4], ORB [5], BRIEF [6] oder SURF [7] eingesetzt werden. Wurden die Feature ermittelt müssen korrespondierende Feature zwischen den Bildern ermittelt werden. Danach werden zwischen den Bildern Übereinstimmungen in den Features mittels RANSAC gesucht. Gleichzeitig werden mittels RANSAC auch die einzelne Feature die keine Übereinstimmungen haben aus dem Verfahren ausgeschlossen. Nun wird die Fundamental-Matrix F für jedes Bildpaar ermittelt. Dazu werden mindestens 8 korrespondierende Features zwischen den beiden Bildpaaren benötigt. Aus der Fundamental-Matrix F kann die Essential-Matrix E unter zuhilfenahme der Kamera-Kalibration-Matrix K ermittelt werden. Mit E kann nun die Kamera-Ausrichtung bestehend aus Kamera- Position und Rotation berechnet werden.

Einige Verfahren benötigen genaue Informationen über die Kamera Position, Ausrichtung und andere Angaben wie die Brennweite der Linse etc. andere wiederum lesen diese Informationen aus den EXIF-Headern der Bildern direkt aus, wenn diese zur Verfügung stehen, oder schätzen diese [9] [10].

Es wurden in den letzten Jahren große Fortschritte im Bereich Deep Learning, zur Gewinnung von 3D-Punktwolken, gemacht [11] [12] [13]. Mit diesen Verfahren ist es möglich aus einer minimalen Datenbasis die auch nur aus einem Bild bestehen kann 3D-Punktwolken zu generieren. Was dem Ziel die Datenübertragung zum Server gering zu halten entgegenkommt. Nachteil der Verfahren ist, dass auf dem Bild nur ein Objekt ohne Hintergrund abgebildet sein darf, oder das zu erkennende Objekt mit einer Binären Maske maskiert werden muss. Eine Rekonstruktion von komplexen Szenen ist so nicht oder nur mit sehr hohem Aufwand möglich [14] und wird daher nicht weiter untersucht. Der in [14] vorgestellte Ansatz geht diese Problematik an und soll daher untersucht werden.

2.2 Segmentierung von 3D-Punktwolken

Unter der Segmentierung von 3D-Punktwolken versteht man, dass jedem Punkt ein Label zu zugewiesen wird, welches Auskunft über eine Eigenschaft des Punktes gibt. Im Falle dieser Arbeit sollen Label wie Stamm, Blatt, Blüte oder Frucht vergeben werden.

Bei der Analyse von Pflanzen in Form von 3D Punktwolken gibt es einige nicht gelernte Lösungen [15] [?] die das segmentieren von 3D Punktwolken von Pflanzen in Stiele und Blätter behandeln, diese Ansätze haben aber das Problem, dass sie nur unter bestimmten Bedingungen gute Ergebnisse liefern.

[15] verfolgt den Ansatz eine Punktwolke in Stamm und Blätter anhand der Hauptkrümmung um jeden Punkt zu segmentieren. Hierbei wird die Hauptkrümmung für einen Punkt so interpretiert, dass der Punkt als Stamm interpretiert wird wenn die Krümmung hoch ist. Es wird also davon ausgegangen, dass die Stämme eine höhere Krümmung aufweisen als die Blätter.

In [?] wird ein Ansatz verfolgt, der sich die zylindrische Form von Stielen zunutze macht um diese von anderen Teilen der Pflanze wie Blätter zu unterscheiden. Es wird das Modell eines Zylinders in der Punktwolke gesucht und wird dieses gefunden werden Punkte die in das Modell fallen als Stamm klassifiziert.

Insbesondere die hohe Qualität der Punktwolken die meist mit einem LIDAR-Scanner oder einem vergleichbaren Gerät erzielt wird, kann mit bildbasierten Methoden wie SfM nicht oder nur mit sehr großen Datenmengen und dem damit verbundenen Rechenaufwand erreicht werden. Ein weiteres Problem das viele Lösungen haben ist, dass der Hintergrund, also alles was nicht zur Pflanze gehört, manuell entfernt wird, oder der Hintergrund so vorbereitet wird das dieser beim erstellen der Punktwolke ignoriert wird und erst auf der freigestellten Pflanze der eigentliche Ansatz ausgeführt wird. Um diese Lösungen dennoch in einer voll automatischen Pipeline nutzen zu können muss das freistellen der Pflanzen erst automatisiert werden.

Auch im Bereich Segmentierung wurden große Fortschritte im Bereich tiefer Neuronaler Netze (Deep-Learning) [17] auf 3D-Punktwolken gemacht, die mit einer angelernten Netzarchitektur bestimmte Objekte erkennen und Teile davon segmentieren. Diese Ansätze können auch auf Pflanzen angewandt werden. Dazu müssen Architekturen wie PointNet[18]/PointNet++[19] oder DGCNN [20] auf Punktwolken von Pflanzen trainiert werden, da für diese Klasse von Objekten noch keine Gewichtete Trainiert wurden.

2.3 Registrierung von 3D-Punktwolken

Bei der Registrierung von 3D-Punktwolken probiert man eine Transformationen T zu finden die eine Quell-Punktwolke S_p so transformiert, dass der Abstand $d(S_p, T_p)$ zu einer Ziel-Punktwolke T_p minimiert wird. Das Abstandsmaß $d(S_p, T_p)$ kann verschieden formuliert werden. Eine Möglichkeit besteht darin die Summe über die Abstände eines jeden Punktes in S_p zum nächsten Punkt in T_p zu bilden und dies als Maß $d(S_p, T_p)$ zu nutzen.

Die am weitesten verbreitete Method zur Registrierung von 3D Punktwolken ist Iterative Closest Points (ICP) [21]. ICP basiert auf dem Ansatz das zwei Punktwolken iterativ aneinander angenähert werden. Dabei wird bei jeder Iteration für jeden Punkt p_i aus der Quellpunktwolke S_p der Punkt aus der Zielpunktwolke gesucht der p_i am nächsten ist. In einem zweiten Schritt wird der Abstand zwischen den korrespondierenden Punkten minimiert. Zuerst wird für jede

Punkwolke das Massezentrum c_s und c_t berechnet und S_p um die Differenz $c_t - c_s$ verschoben. Danach wird mittels Singulärwertzerlegung die Rotation R ermittelt und auf S_p angewandt. Der Vorgang wird wiederholt bis es zu einer Konvergenz kommt, also T nahezu der Einheitsmatrix I entspricht.

Ein Problem das ICP hat ist, das es auch für lokale Minima konvergiert. Das heißt, das eine initiale Transformation für S_p so gewählt werden muss, so dass der ICP-Durchlauf bei einem globalen Minimum konvergiert. Ansätze wie Go-ICP [22], die mit verschiedenen initialen Transformationen starten existieren sind aber sehr Rechenaufwendig und damit recht langsam.

Des weiteren ist ICP anfällig für Ausreißer. Für diese Problem gibt es allerdings existierende Lösungen [23]. Hierbei kommt in der Regel RANSAC [24] zum Einsatz um Ausreißer bei der Registrierung auszuschließen. RANSAC basiert auf der Idee ein Model in einer Punktwolke zu suchen und möglichst viele Punkte zu finden die in das Model passen. Initial werden zufällig so viele Punkte wie nötig gezogen um das Model minimal zu repräsentieren. So werden zum Beispiel für ein Linien-Model zwei zufällige Punkte aus der Punktwolke gezogen. Nun werden die Parameter für das Model mit den gezogenen Punkten ermittelt und alle Punkte die nahe genug an dem so gebildeten Model sind mit aufgenommen. Wenn der Anteil der Punkte die in das Model fallen über einem bestimmten Schwellwert liegen, terminiert der Algorithmus. Ansonsten wird der Vorgang solange wiederholt bis eine Parametrisierung gefunden wurde die terminiert oder eine Obergrenze an Iterationen erreicht wurde. Das Ergebnis des Algorithmus ist auf der einen Seite die gefundene Parametrisierung und auf der anderen Seite eine Unterteilung der Punkte in der Punktwolke in Punkte die in das Model mit gegebener Parametrisierung fallen und in die die nicht in das Model fallen. Letzteres kann genutzt werden um Punkte bei der Registrierung auszuschließen.

Ein weiteres Problem ist, dass die meisten ICP-Ansätze keine Werte für die Skalierung schätzen. Es gibt einige wenige Ansätze für dieses Problem die auch eine Schätzung für die Skalierung ermitteln [25].

Auch für das Registrierung-Problem gibt es Lösungen aus dem Deep-Learning-Bereich, aber auch hier gibt es das Problem, dass es nur wenige Ansätze gibt die mit Skalierung umgehen können [26]. Bekannt Ansätze sind DCP [27], PointNetLK [28], RPM-Net [29], DeepGMR [30], PRNet [31] und PCRNet [32]. Einige dieser Ansätze sollen in dieser Arbeit untersucht werden.

3 Realisierung

Es müssen vier Teilprobleme berücksichtigt werden. Zu Beginn muss aus einer Reihe Bilder eine Punktwolke generiert werden. Hauptziel hierbei ist es möglichst wenig Bilder zu benötigen. Trotzdem müssen Aspekte wie die Qualität der Punktwolken berücksichtigt werden. Diese sollte die aufgenommene Szene klar darstellen und wenig Rauschen und andere Störungen enthalten.

Ein zweites Problem ist die Registrierung zweier Punktwolken einer Pflanze zu verschiedenen Zeitpunkten um diese vergleichen zu können. Hierbei gilt es die ideale Transformation T bestehend aus Rotation, Skalierung und Translation zu finden um die beiden Punktwolken so realitätsnah wie möglich aneinander aus zu richten. Es werden drei Ansätze überprüft dieses Problem zu lösen. Der erste Ansatz basiert auf der Idee, beim Beginn einer neuen Zeitserie zur Analyse eines Wachstumsprozesses, eine Punktwolke des Hintergrunds zu erstellen und die Punktwolken der einzelnen Zeitpunkte mit diesem Hintergrund zu registrieren. So wird ein Verhältnis geschaffen das dem der Realität entspricht. Der zweite Ansatz soll untersuchen ob DCP so angepasst werden kann, dass statt Rotation und Translation separat, die komplette Transformations-Matrix mit Skalierung geschätzt werden kann. In einem letzten Ansatz soll die Skalierung durch iteratives probieren gefunden werden mit einer anschließenden Registrierung ohne Schätzung der Skalierung.

Das dritte Problem ist die Segmentierung der Punktwolke in Stamm, Blätter und Hintergrund. Hier gibt es viele Ansätze dieses Problem zu lösen. Allerdings ist es schwer eine allgemein gültigen Lösung zu finden. Ziel ist es daher eine Lösung zu finden die auf möglichst vielen Varianten von Pflanzen funktioniert. Das Problem der Segmentierung ist essentiell für die weitere Analyse einer Zeitserien. Ohne die Information welche Punkte zu Stamm und Blättern gehören kann nicht auf die Entwicklung von Blättern und Stielen geschlossen werden.

Zuletzt müssen aus den bisherigen Problemen in geeignete Pipelines zusammengefasst werden und durch einen Server angesteuert werden. Hier muss die Lastverteilung und Datenhaltung beachtet werden.

3.1 Architektur

Die Anwendung soll über eine REST-API angesteuert werden können. Es soll möglich sein eine neue Messreihe anzulegen. Dazu müssen die Bilder für die initiale Punktwolke auf den Server übertragen werden. Es soll möglich sein weitere Messpunkte zu einer Messreihe hinzuzufügen. Zu einer Messreihe sollen Auswertungen zur Verfügung gestellt werden. Zu jeder Messreihe soll ein Bearbeitungs-Status abgerufen werden können, da die einzelnen Pipelines asynchrone im Hintergrund ausgeführt werden sollen. Die einzelnen Abläufe sind in Abbildung 1 zu sehen.

3.2 Umsetzung Generierung einer 3D Punktwolke aus Bildern

Die aus der Analyse der verschiedenen Verfahren (siehe Evaluation) ausgewählte Anwendung ODM wird als Docker bereit gestellt. Um den Docker auszuführen muss eine Ordner-Struktur bereit gestellt werden. In dieser Ordner-Struktur muss ein Ordner „images“ enthalten sein, der die Bilder des Datensatzes, aus dem eine Punktwolke generiert werden soll, enthält.

Ist das Verzeichnis angelegt kann der Docker gestartet werden. Hierbei muss allerdings beachtet

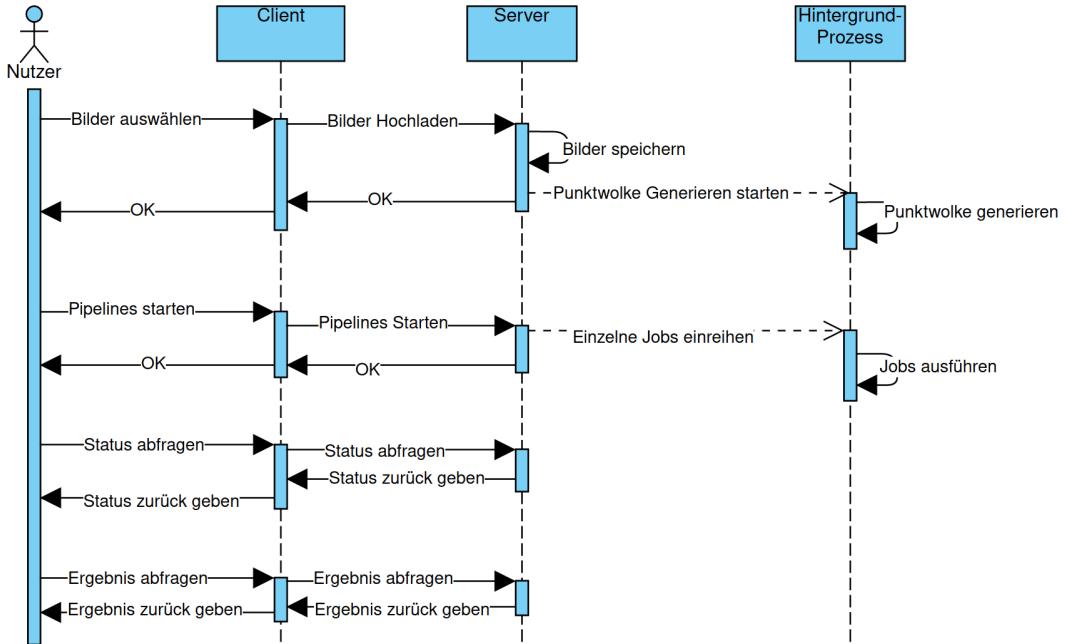


Abbildung 1: Arbeitsabläufe zwischen Nutzer, Client, Server und Hintergrundprozess.

werden, dass der Docker mit den selben Rechten wie der User arbeitet. Ansonsten werden per Standardeinstellung Root-Rechte genutzt. Das führt dazu das auf die erstellten Daten nur noch lesend zugegriffen werden kann, was das Aufräumen erschwert.

3.3 Umsetzung Registrierung zweier Punktwellen

Bei der Umsetzung der Registrierung muss die Besonderheit beachtet werden, dass die Skalierung der Punktwellen nicht bekannt ist und sich unterscheiden. Das heißt die Punktwellen liegen in unterschiedlichen Maasstäben vor. Um ein gutes Verfahren zu finden was mit dieser Besonderheit umgehen kann wurden mehrere Ansätze verglichen.

In einem ersten Ansatz wurde versucht ein Pipeline zu erstellen die eine gute Initiale Lösung findet ohne den ganzen Suchraum $SO(3)$ zu durchsuchen. Die Pipeline sucht zuerst für die Quell- und Ziel-Punktwellen S und T nach der größten Ebene in der Punktwolke und richtet die Punktwolke so aus, dass die gefundene Ebene auf der Ebene liegt, die die x und y Achse bilden. Zudem wurden die Punktwellen so skaliert, dass die begrenzenden Boxen gleich groß sind. Des weiteren werden aus beiden Punktwellen nur ein Teilmenge P_c der Punkte in einem Radius r um dem Punkt c der das Zentrum der Masse der Punktwellen repräsentiert entnommen. Damit soll Rauschen an den Rändern verhindert werden und ein Teil des Hintergrunds soll ausgeblendet werden. Aus diese Teilmenge werden Subsample gezogen um den Rechenaufwand zu minimieren. In diesem Zustand haben Quell- und Ziel-Punktwolke eine gute initiale Transformation. Für die zu registrierende Punktwolke wird zunächst mittels SIFT 3D [33] eine Verbesserung der initiale Transformation gesucht. Danach wird ein ICP-Durchlauf gestartet der auch die Skalierung der Ziel-Punktwolke schätzt. Um das Ergebnis nach der Schätzung der Skalierung weiter zu verbessern wird noch ein ICP-Durchlauf gestartet - diesmal ohne Schätzung der Skalierung.

Ein weiterer Ansatz ist es DCP so abzuwandeln das statt Translations-Vektor und Rotations-Matrix direkt die ganze Transformations-Matrix geschätzt wird. Dazu muss der SVD-Head so angepasst werden das er statt einer 3×3 Rotations-Matrix R und dem Translations-Vektor \vec{t} die Translations-Matrix T zurück liefert wird. Um das zu Erreichen muss die Eingabe der Größe $N \times 3$ auf eine Eingabe der Größe $N \times 4$ erweitert werden. Hier kann die Eingabe um einen Vektor mit Einsen erweitert werden. Das führt dazu das die Singulärwert-Zerlegung von H im SVD-Head nun eine 4×4 Matrix ist. Die Annahme die wir treffen ist das H als Transformations-Matrix interpretiert werden kann.

In einem letzten Ansatz wird die Skalierung geschätzt in dem ein vorgegebener Bereich an Werten für die Skalierung in einer vorgegebenen Schrittgröße s durchsucht wird. Wieder werden die Quell- und Ziel-Punktwolke an der XY-Achse ausgerichtet und eine Teilmenge P_c um das Zentrum c entnommen wie im ersten Ansatz. Auch werden wieder Subsample gezogen. In jedem Skalierungs-Schritt wird die Punktwolke mit der aktuellen Skalierung skaliert und danach mit einem Registrierungs-Verfahren registriert. Um die Ergebnisse der einzelnen Registrierungen zu messen wird der Abstand zwischen der transformierten Quell-Punktwolke und der Ziel-Punktwolke berechnet. Der Abstand $d(S, T)$ wird so berechnet, dass für jeden Punkt p aus Punktwolke T der Abstand zum nächsten Punkt in der Punktwolke S berechnet wird.

Man könnte auch den Abstand von allen Punkten in S messen, aber das kann zu Problemen bei kleinen Skalierungen führen, da wenn im extrem Fall die Punktwolke S sehr klein ist alle Punkte in S nahezu keinen Abstand mehr zueinander haben. Man könnte S also auch durch einen Punkt p_S abstrahieren. Bei der Registrierung muss S nur sehr nah an einen Punkt p_t in T sein sodass gilt $p_t \approx p_S$ und der Abstand zwischen S und T geht gegen 0. Dadurch werden sehr kleine Skalierung durch dieses Maß bevorzugt. Nimmt man statt aller Punkte in S alle Punkte in T wird der Fehler in diesem Fall größer als 0 sein, es sei den T ist auch sehr groß.

Mit diesem robustem Maß kann man nun die Güte der einzelnen Skalierungs-Iterationen bewerten und die beste Skalierung mit zugehörigen Transformation für Rotation und Translation finden. Da die Wahl des Registrierungs-Verfahren offen bleibt wurden hier mehrere Verfahren mit einander verglichen. Es wurden ICP, DCP, PointNetLK und RPM-Net miteinander verglichen.

3.4 Umsetzung Segmentierung

Zur Segmentierung der Pflanzen-Punktwolke in Stamm und Blätter wurde ein Ansatz verfolgt der an den in [15] angelehnt ist, der mittels der Hauptkrümmung eines Punktes ermitteln soll ob es sich um ein Blatt oder einen Stiel handelt. Hierbei wird eine einfache Entscheidungsregel $f(x)$ mittels eines Schwellwerts T_k genutzt: Ist die Stärke der Krümmung $k(x)$ des Punktes höher als der Schwellwert handelt es sich um einen Stiel da diese einen größeren Krümmung haben als Blätter. Die Schwierigkeit ist es einen guten Schwellwert und eine gute Kennzahl für die Stärke der Krümmung zu finden.

Die Hauptkrümmung eines Punktes p_i kann über die Normalen des Punktes und der k nächsten Nachbarpunkte P_i berechnet werden.

$$\vec{m}_j = (I - \vec{n}_i \otimes \vec{n}_i) \cdot \vec{n}_j \quad (1)$$

Aus den Abbildungen \vec{m}_j (Siehe Gleichung 1) für alle P_i kann die Kovarianzmatrix C_i berechnet werden. Die Berechnung von C_i ist in Gleichung 2 zu sehen.

$$C_i = \frac{1}{k} \sum_{j=1}^k (\vec{m}_j - \vec{m}) \otimes (\vec{m}_j - \vec{m}) \quad (2)$$

Die Hauptkrümmung kann aus den Eigenwerten $0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3$ von C_i bestimmt werden. λ_3 ist die stärkste Krümmung und λ_2 die schwächste Krümmung.

Einige untersuchte Ansätze für eine Funktion $k(p_i)$ sind in 3 zu finden.

$$\begin{aligned} k_1(p_i) &= \lambda_3 \\ k_2(p_i) &= \lambda_2 \\ k_3(p_i) &= (\lambda_3 + \lambda_2)/2 \end{aligned} \quad (3)$$

Je höher der Wert der Funktion $k(p_i)$ aus den Gleichungen in 3 ist desto Wahrscheinlicher gehört ein Punkt zum Stiel einer Pflanze. Die Entscheidungsregel dafür ist in Gleichung 4 zu sehen. Ein guter Wert für den Schwellwert T_k muss in Experimenten für die Funktionen k_1 , k_2 und k_3 gefunden werden.

$$f(p_i) = \begin{cases} 1 & k(p_i) \geq T_k \\ 0 & \text{sonst} \end{cases} \quad (4)$$

In einem weiteren Ansatz wurde eine Implementierung von PointNet auf einem Datensatz von Pflanzen-Punktwolken trainiert um so einen geeigneten Classifier zu trainieren. Der Classifier soll für jeden Punkt einer Punktwolke bestehend aus Position und Normalen eine Schätzung liefern was der Punkt repräsentiert. Mögliche Repräsentationen können Stamm, Blatt oder Hintergrund sein. Weitere denkbare Repräsentationen können die Früchte und Blüten der Pflanzen sein. Wird die Farbe der Punktwolke mit einbezogen, können auch Krankheitsbilder wie vertrocknende Blätter in die Repräsentation eingeschlossen werden. Da die Ergebnisse mit PointNet nicht gut genug waren wurde eine verbesserte Version PointNet++ auf dem Datensatz trainiert. Diese wurde mit und ohne Normalen, mit 2 Labels ohne Hintergrund und mit 3 Labels mit Hintergrund trainiert.

Nach der Segmentierung wird das Ergebnis noch einmal überarbeitet. Für jedem Punkt p_i werden die Schätzungen N_i der k ($k = 10$) nächsten Nachbarn bestimmt und aus deren Repräsentations-Schätzungen ein Histogramm H_i erzeugt. Die Schätzung mit dem höchsten Histogramm-Wert wird als neue Schätzung s_i für den Punkt p_i genutzt. Die Berechnungs-Vorschrift ist in Gleichung 5 zu finden. Dieser Vorgang wird für alle Punkte solange wiederholt bis es bei den Punkten zu keinen Änderungen mehr kommt oder eine maximale Iterations-Obergrenze erreicht wird.

$$\begin{aligned}
L &= \{0, 1, 2\} \\
w(x) &= \begin{cases} 0,5 & x = 2 \\ 1 & \text{sonst} \end{cases} \\
H_i(x) &= \sum_{j=0}^k (w(x)|N_{ij} = x) \\
x &\in L \\
s_i &= \operatorname{argmax}_x(H_i(x))
\end{aligned} \tag{5}$$

3.5 Umsetzung Server

Der Server ist mit dem Python-Framework Flask erstellt. Flask bietet eine Schlanke API um neue REST-Endpunkte zu erstellen und Daten vom Client anzunehmen.

Ein Hintergrund-Prozess führt die einzelnen Jobs aus und sorgt für den Lastausgleich. Die Anfragen an den Server werden asynchron verarbeitet. Jede Anfrage wird in die Job-Queue eingeordnet. Diese wird vom Hintergrund-Prozess abgearbeitet. Einzige Ausnahme bildet hier das Speichern der Bilder. Die Bilder müssen synchrone zum Request auf der Platte persistiert werden, da Flask die Datei-Streams nach dem Lebenszyklus eines Requests schließt.

Beim Start des Servers wird neben dem Hintergrund-Prozess der aktuelle Stand der Datenhaltung eingelesen und der Status des Servers aufgebaut. Bei der Datenhaltung wurde auf eine Datenbank verzichtet, da die Anwendung Datenhaltungs-Technisch simpel ist und die meisten Daten als BLOB [34] vorliegen und daher nicht ideal für ein relationales Datenbank-Modell sind. Die Daten werden direkt auf der Festplatte des Host-Systems abgelegt. Des weiteren werden zwei Instanzen von PointNet++ gestartet. Eine für die Segmentierung des Hintergrundes und einer für die Segmentierung der Pflanze.

Der Server bietet folgende Schnittstellen die den Betrieb der Anwendung ermöglichen:

GET /listings/{Messreihe}

Dient dazu zu einer Messreihe den aktuellen Status abzuholen. Der Status der Messreihe setzt sich aus den einzelnen Pipeline-Status zusammen.

PUT /results/{Messreihe}/{Zeitstempel}

Dient dazu zu einem Zeitpunkt einer Messreihe die ermittelten Werte wie Anzahl Blätter oder Volumen abzuholen.

```
{
    "LeaveCount": 7,
    "Height": 0.276259834557858,
    "Volume": 0.025656320729475497,
    "GrothSinceLastSnapshot": 1.1,
    "BackgroundRegistration": {
        "Transformation": [
            [0.9995419979095459, 0.0299260001629591, -0.0045190
             70032984018, -0.030479200184345245],
            [-0.029895899817347527, 0.9995309710502625, 0.00658
             34098495543, 0.025272000581026077],
            ...
        ]
    }
}
```

```

        [0.004713969770818949, -0.006445290055125952, 0.999
         9679923057556, 0.0013244400033727288],
        [0.0, 0.0, 0.0, 1.0]
    ],
    "Scale": 1.3
}

```

Listing 1: Beispiel Ergebnisse eines Zeitstempels

POST /data/{Messreihe}/{Zeitstempel}

Dient dazu neue Datensätze hinzu zu fügen. Es muss eine Sammlung von Bildern einer Pflanze mit geliefert werden. Wird der Endpunkt angesprochen werden die Bilder gespeichert und die Pipeline zum generieren der Punktwolke in der Job-Queue hinzugefügt. Wird als Zeitstempel „background“ angegeben wird dieser Datensatz als Hintergrund für die derzeitige Messreihe interpretiert.

PUT /data/{Messreihe}/{Zeitstempel}

Dient dazu Pipelines für einen Zeitstempel einer Messreihe zu starten. Hierzu wird im Payload des Request an den Server eine JSON-Datei übermittelt welche eine Liste der Pipelines enthält (Beispiel siehe Listing 2). Die spezifizierten Pipelines werden, in der angegebene Reihenfolge, der Job-Queue hinzugefügt.

```

{
  "jobs" : [
    {
      "jobName" : "SegmentBackground",
      "jobParameter" : {}
    }
  ]
}

```

Listing 2: Beispiel Payload zum starten einer Pipeline

Es stehen folgende Pipelines zur Verfügung:

- Punktwolke generieren
- In Shapnet-Format überführen
- Entfernung des Hintergrundes
- Segmentierung Pflanze
- Blatt/Stamm Trennung
- Blätter zählen
- In Registrierungs-Format überführen

- Hintergrund-Registrierung
- Größen berechnen

3.6 Umsetzung Pipelines

Punktwolke generieren

Die Punktwolke wird auf Basis der hoch geladenen Bilder zu einem Zeitstempel generiert. Beim Anlegen eines neuen Zeitstempels wird die Pipeline automatisch gestartet, kann aber auch manuell neu gestartet werden sollte es während der Generierung zu einem Server-Ausfall kommen.

In Shapnet-Format überführen

Diese Pipeline überführt die Punktwolke in das Shapenet-Format. Die Punktwolke muss in diesem Format vorliegen um die beiden Segmentierungs-Pipelines nutzen zu können. Bevor die Punktwolke in das Shapenet-Format überführt wird können zwei optionale Schritte durchgeführt werden. Bei der Hintegrund-Segmentierung wird die Punktwolke mittels der größten in der Punktwolke zu findenden Ebene an der XY-Ebene ausgerichtet. Bei der Pflanzen-Segmentierung werden alle als Hintergrund gelabelten Punkte entfernt. Danach beginnt die eigentliche Überführung in das Shapenet-Format. Zuerst wird die Punktwolke so verschoben, dass die Bounding-Box im Koordinaten-Ursprung beginnt und sich auf den positiven Bereich der Achsen erstreckt. Danach wird die Punktwolke in der Form normiert, dass sich alle Punkte der Punktwolke zwischen den Punkten $(0, 0, 0)$ und $(1, 1, 1)$ befinden. Zuletzt werden $n = 16384$ Punkte aus der Punktwolke zufällig gezogen, wobei die Punkte nicht zurück gelegt werden.

Entfernung des Hintergrundes

Die im Shapenet-Format vorliegende Punktwolke wird mit der Instanz von PointNet++ für die Hintegrund-Segmentierung segmentiert. Das Ergebnis wird im Speicher des Host-Systems abgelegt. Da die Segmentierung etwas verrauscht ist wird das Ergebniss mit einem nächsten Nachbarn Vote verbessert. Im Anschluss wir das Ergebnis auf die original Punktwolke übertragen, da sonst zu wenig Punkte für die Segmenetierung der Pflanze übrig bleiben. Auch wird der Hintegrund entfernt und aus den verbleibenden Punkten, die nun nur noch Pflanzen-Punkte enthalten wieder eine Punktwolke im Shapenet-Format erstellt.

TODO: Erleutern wie das Segmentierungs-Ergebnis auf die originale Punktwolke übertragen wird. Auch erwähnen das eine Punktwolke die nur den Hintergrund enthält für die Registrierung gespeichert wird.

Segmentierung Pflanze

Auf dem Ergebniss der Hintergrund-Entfernung wird PointNet++ zur Segmentierung der Pflanze gestartet. Auch hier wird das Ergebnis im Speicher des Host-Systems abgegelegt. Wieder wird das Segmentierungs-Ergebniss wie bei der Hintergrundentfernung verbessert. Im Falle der Pflanzen-Segmentierung ist dieser Schritt nicht zwangsläufig nötig, da die Ergebnisse gut genug sind.

Blatt/Stamm Trennung

Die Punktwolke die die segmentierte Pflanze enthält wird in zwei Punktwolken aufgeteilt. Wobei die eine Punktwolke nur noch Punkte enthält die Blätter repräsentieren und eine die entsprechend nur noch Punkte enthält die Stiele repräsentieren.

Blätter zählen

Die Punktwolke P die die Blätter enthält wird nun mit einer 3D Interpretation des Flood-Fill-Algoritmus untersucht. Hierbei wird ein zufälliger Punkt in der Punktwolke gewählt und in die Menge Q aufgenommen. Nun wird solange bis Q keine Punkte mehr enthält ein Punkt p aus Q entnommen und mittels nächster Nachbarn-Suche im Radius r nach weiteren Punkten in der Nähe gesucht. Gefundene Punkte werden in Q aufgenommen, wenn sie nicht in B enthalten sind. Der Punkt p wird danach in die Menge B aufgenommen. Ist Q leer wird die Menge B aus P entfernt und der Blatt-Zählstand um eins erhöht. Der Vorgang wird so lange wiederholt, bis keine Punkte mehr in P enthalten sind.

In Registrierungs-Format überführen

Zuerst wird die Punktwolke wieder anhand der größten Ebene die in der Punktwolke zu finden ist an der XY-Ebene ausgerichtet. Danach wird um das Zentrum der Punktwolke ein Ausschnitt entnommen. Im folgenden wird die Punktwolke wie beim Shapenet Format normiert und es werden zufällig 1024 Punkte aus der Punktwolke gezogen.

Hintergrund-Registrierung

Um einen Zeitstempel t einer Messung mit dem Hintergrund der Messreihe zu registrieren muss sowohl die Punktwolke P_t für den Zeitstempel t als auch die Punktwolke für den Hintergrund im Registrierungs-Format vorliegen. Ist das gegeben kann die Registrierung gestartet werden. Hierbei wird ein Skalierungs-Faktor s und eine Transformation T ermittelt. Das Registrierungs-Ergebnis der Punktwolke P_t kann dann mit der Formel $P_{Registration} = (P_t \cdot s) \cdot T$ ermittelt werden. Wobei beachtet werden muss, dass erst die Skalierung und dann die Transformation angewandt wird.

Größen berechnen

Ist die Punktwolke P_t zum Zeitstempel t mit dem Hintergrund registriert, können die Größen Höhe und Volumen nun leicht berechnet werden. Die Höhe kann über den höchsten z-Wert der Punkte in P_t ermittelt werden. Um das Volumen zu berechnen, muss für P_t die Länge l , Breite b und Höhe h ermittelt werden (siehe Gleichungen 6 - 8). Das Volumen v lässt sich dann folgendermaßen berechnen: $v = h \cdot b \cdot l$.

$$l = \begin{cases} |minX| + maxX & minX \leq 0 \\ maxX - minX & \text{sonst} \end{cases} \quad (6)$$

$$b = \begin{cases} |minY| + maxY & minY \leq 0 \\ maxY - minY & \text{sonst} \end{cases} \quad (7)$$

$$h = \begin{cases} |minZ| + maxZ & minZ \leq 0 \\ maxZ - minZ & \text{sonst} \end{cases} \quad (8)$$

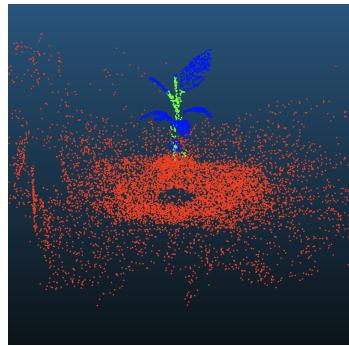


Abbildung 2: Hintegrund-Segmentierung

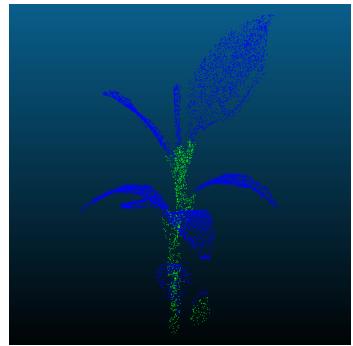


Abbildung 3: Planzen-Segmentierung

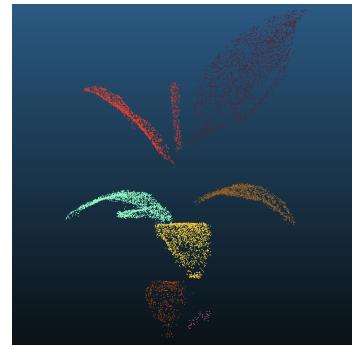


Abbildung 4: Blatt-Segmentierung

Ist die Höhe für den vorherigen Zeitstempel $t - 1$ auch bekannt, kann das relative Wachstum seit dem letzten Zeitpunkt ermittelt werden. Dieser Schritt kann nicht für den ersten Zeitstempel ausgeführt werden.

4 Ergebnisse

4.1 Vergleich von Verfahren zur Generierung von 3D-Punktwolken auf Basis von Bildern

Um eine geeignete Anwendung zur Generierung von 3D-Punktwolken auszuwählen muss erst entschieden werden nach welchen Gütekriterien verschiedene Verfahren betrachtet werden sollen und wie wichtig diese sind. Bei dieser Arbeit ist die nötige Anzahl der Bilder die zur Generierung einer Punktwolke mit genug Information nötig sind wie die Dauer die Punktwolke zu erstellen sind die wichtigsten Kriterien.

Um zu messen wie viele Bilder benötigt werden um eine Szene gut zu repräsentieren muss ein Goldstandart von der fotografierten Szene als 3D-Punktwolke erstellt werden, mit dem Verglichen werden kann. Dieser Goldstandart kann erstellt werden, indem alle Fotografien genutzt werden die zur Verfügung stehen und damit alle Verfahren eine Punktwolke erstellen lassen. Unter den erstellten Punktwolken kann die ausgesucht werden die die Szene am besten repräsentiert. Das trifft auf die Punktwolke zu die den höchsten Detail-Grad hat und möglichst wenig Fehler, wie fehlende Teile der Szene oder Rauschen enthält. Die so erstellte Szene sollte nachbearbeitet werden um etwaige Fehler wie Rauschen manuell zu entfernen.

Nun kann mit weniger Bildern aus dem selben Datensatz ein weitere Punktwolke generiert werden und die Distanz der Punkte der Goldstandart-Punktwolke zum jeweils nächsten Punkt der erstellten Punktwolke gemessen werden. Die Summe dieser Abstände kann als Fehlermaß genutzt werden. Bei der Berechnung der Abstände muss beachtet werden, dass die Punktwolken mit dem Goldstandart registriert ist.

Nach einem manuellen Auswahlverfahren wurden ODM und Colmap als geeignete Anwendung zur Generierung von Punktwolken gewählt, da diese die einzigen untersuchten Anwendungen sind die gute Ergebnisse bei wenig Bildern erzielen die die Szene erkennen lassen. Um die beiden Anwendungen weiter zu vergleichen wurde aus 73 Bildern sowohl mit Colmap als auch mit ODM eine Punktwolke generiert, da ODM die Szene besser repräsentiert und weniger Rauschen beinhaltet wurde die mit ODM erstellet Punktwolke als Goldstandart gewählt.

Im Folgenden wurden aus Teilmengen der Bilder jeweils für ODM und Colmap Punktwolken generiert. Die Teilmengen bestehen aus 5 bis 70 Bildern wobei immer 5 Bilder mehr pro Schritt genutzt wurden. Bei der Wahl der Bilder pro Schritt wurde beachtet, dass die Bilder ähnliche Ausschnitte enthalten, da diese zur Generierung von Punktwolken nötig sind.

In zwei Vergleichen wurden zum einen die Durchschnittliche Distanz für jeden Punkt zum nächsten Punkt in der Goldstandart-Punktwolke verglichen und zum anderen die Anzahl generierter Punkte pro Schritt verglichen. Die Ergebnisse sind in Abbildung 5 und 6 zu sehen. Es ist zu erkennen, dass Colmap zwar mehr Punkte generiert als ODM, aber ab einer bestimmten Anzahl Bilder überwiegt das Rauschen in den mit Colmap generierten Punktwolken so stark, dass der Abstand zur Goldstandart-Punktwolke größer als der mit ODM erreichte Abstand wird.

Generell lässt sich in den generierten Punktwolken beobachten, dass Colmap mehr Rauschen enthält als ODM. Exemplarisch in den Abbildungen 7 und 8 zu sehen. Ein Nachteil von ODM ist, dass es um die Kannten herum sehr viele Hintergrundpunkte mit aufgenommen werden die eine Art Halo um die erkannten Strukturen bilden. Dadurch können feine Strukturen in einander

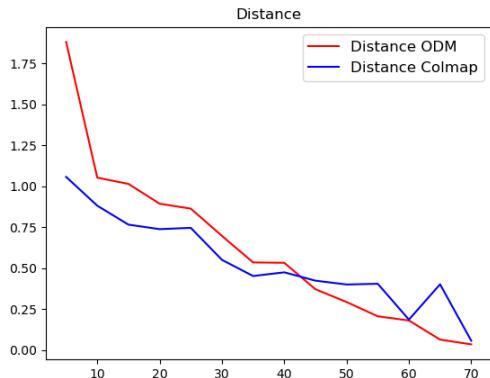


Abbildung 5: Distanz der Punktfolge mit bestimmter Anzahl Bilder zur Goldstandart-Punktfolge.

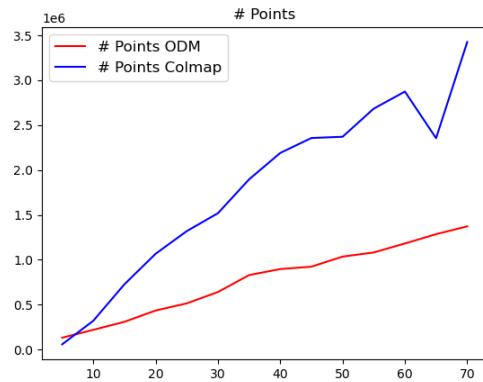


Abbildung 6: Anzahl Punkte die mit einer bestimmten Anzahl Bilder generiert werden kann.



Abbildung 7: ODM mit 40 Bildern - Ansicht von der Seite



Abbildung 8: Colmap mit 40 Bildern - Ansicht von der Seite

über fließen. Bei Colmap hat man hingegen das Problem, dass es zu größeren Lücken in erkannten Oberflächen kommt. Beide Probleme sind in Abbildung 9 und 10 zu sehen. Des weiteren kann es vorkommen, dass Colmap zwei Szenen in den Bildern erkennt was insbesondere bei Nutzung von vielen Bildern vorkommt (Abbildung 11).

Ein weiterer Punkt bei der Bewertung des Verfahrens zur Generierung von 3D Punktfolgen ist die Zeit die benötigt wird die Punktfolge zu generieren. Hier dominiert ODM klar. Obwohl mangels der GEO-Position der Bilder, ODM nur in der Lage ist die CPU parallel zu nutzen und nicht die GPU werden die Punktfolgen wesentlich schneller generiert als mit Colmap, welches die GPU nutzt und diese auch gut auslastet. ODM benötigt zur Generierung einer Punktfolge mit 73 Bildern 566 Sekunden während Colmap für die gleiche Datenbasis 4148 Sekunden benötigt. Sollten GEO-Positionen für die Bilder bekannt sein kann ODM die GPU nutzen und somit die Zeit die benötigt wird die Punktfolgen zu generieren weiter verbessern.

Da ODM wesentlich weniger Rauschen beinhaltet, keine Probleme mit doppelt erkannten Sze-



Abbildung 9: ODM mit 40 Bildern - Ansicht von Oben



Abbildung 10: Colmap mit 40 Bildern - Ansicht von Oben



Abbildung 11: Colmap mit 55 Bildern und doppelt erkannter Szene.

	DCP	RPM-Net	PointNetLK
Distanz	0, 0292	0, 0663	0, 026
Loss	0, 922	0, 061	0, 111

Tabelle 1: Vergleich von Registrierungs-Verfahren auf ModelNet40. Distanz und Loss sind jeweils Mittelwerte über alle Datensätze in der Iteration.

nen macht und weniger Zeit zur Generierung der Punktwolken benötigt wurde in dieser Arbeit ODM zur Generierung von Punktwolken genutzt.

4.2 Vergleich von Verfahren zur Registrierung 3D-Punktwolken

In einem ersten Schritt wurden verschiedene Lösungen aus dem Deep-Learning Bereich miteinander verglichen. Es wird die Distanz zwischen transformierter Quell-Punktwolke und Ziel-Punktwolke wie im Abschnitt 3.3 beschrieben gemessen. Neben der Distanz berichten wir über den bei der Evaluation gemessenen Loss. Die einzelnen Verfahren sind alle in dem Python-Paket „learning3d“ implementiert und nutzen die selbe Datenbasis ModelNet40 für das Training und sind somit gut vergleichbar. Für DCP und RPM-Net wurden das vorgebildete Modell vom Author des Pakets genutzt. PointNetLK musste trainiert werden und wurde mit den empfohlenen Angaben des Authors trainiert. Die Evaluation wurde drei mal mit zufälligen Daten wiederholt und aus den drei Durchläufen die Mittelwerte für die ermittelten Werten gebildet. Die Ergebnisse sind in Tabelle 2 zu finden.

In einem zweiten Schritt wurden die Verfahren auf konkreten Beispielen der Problemstellung getestet.

TODO: Tabelle mit Vergleich auf echten Daten für DCP RPM-Net PointNetLK ICP und RICP.

TODO: Erkenntnis aus dem Vergleich. Ggf Bilder von den Ergebnissen.

4.3 Vergleich von Verfahren zur Segmentierung von Pflanzen auf 3D-Punktwolken

In Experimenten mit dem handgeschriebenen Klassifier konnte keine gute Parametrisierung gefunden werden, die eine gute Unterteilung in Blätter und Stämme erreichen konnte. Dies liegt in erster Linie an der Qualität der Punktwolken. Diese haben das Problem, dass die oft sehr feinen Stiele der Pflanze nicht als Zylinder sondern als Ebene in den Punktwolken repräsentiert werden. Das ist in Abbildung 12 zu sehen. Ein weiteres Problem ist eine allgemein gültige Parametrisierung zu finden. Ein Beispiel dafür ist in Abbildung 12 zu sehen. Hier wurde mit dem selben Schwellwert klassifiziert.

Datenbasis für das Training von PointNet++

Als Datenbasis für das Training von PointNet++ wurden 144 handgelabelte Punktwolken von verschiedenen Pflanzen gewählt. Darunter sind Tomaten, Mais, Paprika, Avocado, Bananen und weitere nicht bekannte Sorten. Da diese Datenbasis für das Training eines Neuronalen Netz recht klein ist wurde aus jeder einzelnen Punktwolke bis zu 20 Subsample erstellt. Diese wurden so erstellt, dass zufällig n Punkte aus einer Punktwolke gezogen und entfernt wurden. Des

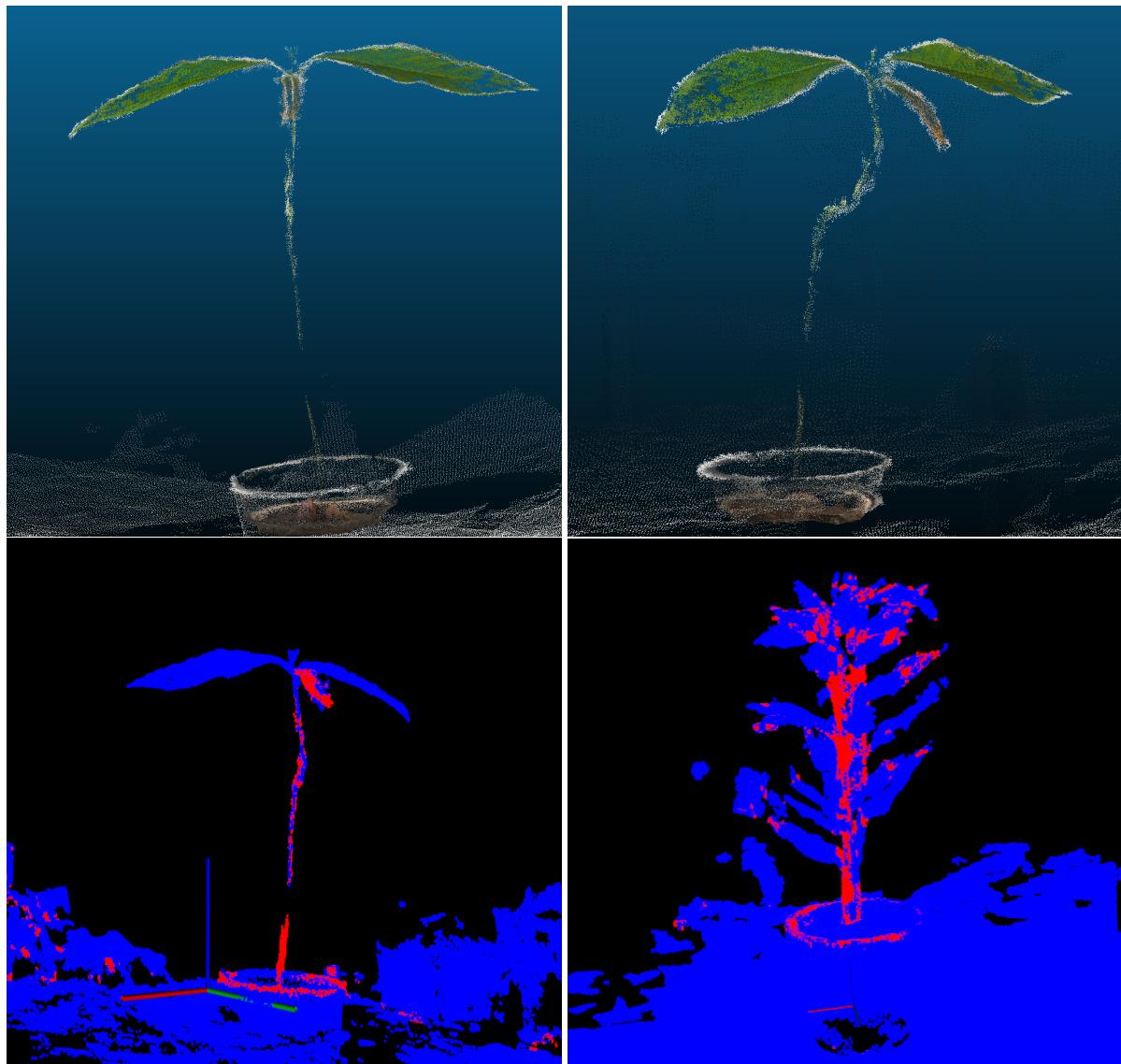


Abbildung 12: Oben: ODM Seitenansicht die als Ebene erkannten Stiel zeigt.
Unten: Segmentierungs-Ergebniss für zwei verschiedene Pflanzen mit dem selben Schwellert.

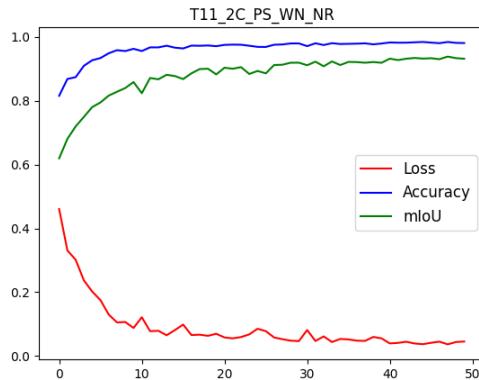


Abbildung 13: PointNet++ Trainings-Ergebnisse pro Epoche mit 2 Klassen und Normalisierung ohne zufällige Rotationen

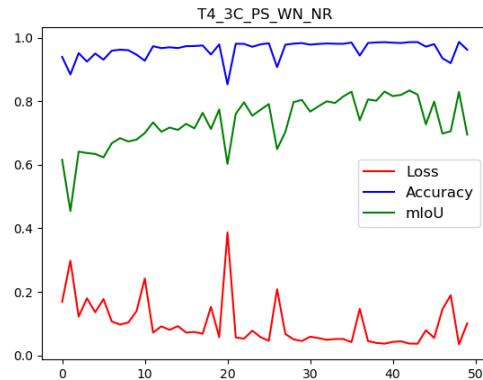


Abbildung 14: PointNet++ Trainings-Ergebnisse pro Epoche mit 3 Klassen und Normalisierung ohne zufällige Rotationen

weiteren sind die Daten so aufbereitet das der Boden der Punktfolge an der XY-Ebene ausgerichtet ist und der Stiel bei geradem Wachstum Richtung z-Achse zeigt. Des weiteren sind die Punkte in der Punktfolge auf 1 normiert. Das heißt sie liegen in dem Raum den die beiden Punkte (0, 0, 0) und (1, 1, 1) aufspannen. Die Daten werden in einem Verhältnis 80 zu 10 zu 10 in Trainings-, Test- und Evaluations-Datensatz aufgeteilt.

Vergleich Training mit und ohne Hintergrund

In einem ersten Vergleich wird die Güte der Segmentierung zwischen dem Training mit und ohne Hintergrund verglichen. Die Ergebnisse des Trainings sind in Abbildung 13 und 14 zu sehen. Die Ergebnisse der Evaluation sind in Tabelle ?? zu sehen. Sowohl die Ergebnisse des Trainings, als auch die Ergebnisse der Evaluation zeigen deutlich, dass die Segmentierung ohne Hintergrund die besseren Ergebnisse liefert. Das wird vermutlich daran liegen, dass der Anteil der Punkte die zum Hintergrund gehören dominiert. Da die Segmentierung mit Hintergrund aber zur Entfernung des Hintergrundes benötigt wird, wird in einem zweiten Vergleich die Hintergrund-Segmentierung neu trainiert. Diesmal wird aber ein Großteil der Hintergrund-Punkte entfernt, indem nur das Zentrum der Punktfolge beim Training betrachtet wird. Die Ergebnisse sind in Abbildung 15 zu sehen. Wie in Abbildung 15 zu sehen ist verbessern sich die Ergebnisse für das Training mit weniger Hintergrund deutlich.

Vergleich Training mit und ohne Normalen

Um zu überprüfen ob sich der Einsatz von Normalen lohnt wird PointNet++ ohne Normalen trainiert und die Ergebnisse mit den Ergebnissen des Trainings mit Normalen (Abbildung 13) verglichen. Die Ergebnisse des Trainings ohne Normalen sind in Abbildung 16 zu sehen. Es ist ein kleiner Unterschied zum Training mit Normalen zu erkennen. Mit Normalen liefern die ersten Epochen zwar schlechtere Ergebnisse, aber nach einigen Epochen gleicht sich die Güte an und es kommt zudem zu weniger Ausreißern. Der Einsatz von Normalen scheint sich also in geringem Maß zu lohnen.

Vergleich Training mit und ohne Normalisierung (Einsatz von nicht normalisierten Punkten trotz Normalisierung im Training möglich?)

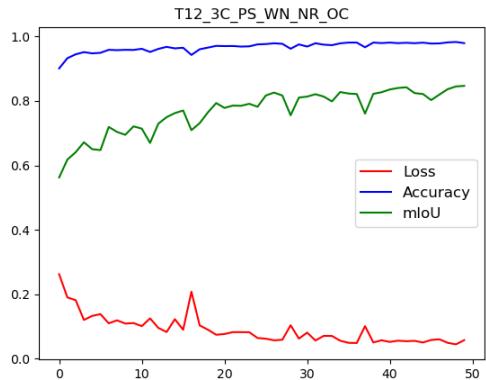


Abbildung 15: PointNet++ Trainings-Ergebnisse pro Epoche mit 3 Klassen und Normalisierung ohne zufällige Rotationen. Mit Ausschnitt um das Zentrum.

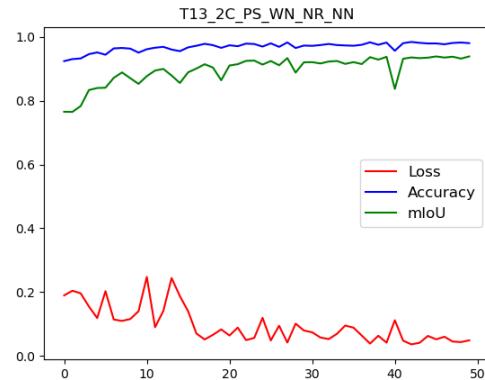


Abbildung 16: PointNet++ Trainings-Ergebnisse pro Epoche mit 2 Klassen und Normalisierung ohne zufällige Rotationen. Ohne Normalen

	t4 mit normalisiert Daten	t4 mit nicht normalisiert Daten	t11 mit normalisiert Daten	t11 mit nicht normalisiert Daten
Loss	0, 038	0, 175	0.044	0.044
Genauigkeit	0, 986	0, 946	0.98	0.98
IoU	0, 84	0, 684	0.926	0.925

Tabelle 2: Vergleich von Evaluations-Ergebnissen von den mit normalisierten Daten trainierten Netzen t4 (mit Hintergrund) und t11(ohne Hintergrund) mit jeweils normalisierten und nicht normalisierten Daten.

Es soll Überprüft werden ob PointNet++ mit oder ohne Normalisierung bessere Ergebnisse liefert. Des weiteren soll geprüft werden, ob auch nicht normalisierte Eingabe-Daten gute Segmentierungs-Ergebnisse liefern wenn PointNet++ mit normalisierten Daten trainiert wurde. Dieser Vergleich wird aufgestellt, da die Punktwolken für die Segmentierung auf eine bestimmte Anzahl Punkte reduziert werden und das Ergebniss - zumindest bei der Hintergrund-Segmentierung - zurück auf die vollständige Punktwolke übertragen werden muss. Muss die Punktwolke nicht normalisiert werden, kann das Ergebniss direkt auf die vollständige Punktwolke übertragen werden. Im Falle einer Normalisierung müsste diese erst umgekehrt werden, aber diese Information wird von PointNet++ nicht zur Verfügung gestellt und müsste ermittelt werden. Alternativ dazu könnten auch die Ergebnisse von der normalisierten Punktwolke auf die nicht normalisierte Punktwolke übertragen werden, was aber mit mehr Rechenaufwand verbunden ist.

Wird die Normalisierung komplett verhindert, kann das Auswirkungen auf die Güte der Ergebnisse der Segmentierung haben.

In Tabelle 2 werden die Ergebnisse der Evaluation von PointNet++ auf nicht normalisierten Daten nach dem Training mit normalisierten Daten mit normalisierten Daten verglichen. Da die Ergebnisse sich für die Segmentierung mit Hintergrund stark verschlechtern, wenn die Daten nicht normalisiert werden wird PointNet++ ohne Normalisierung erneut trainiert. Die Ergebnis-

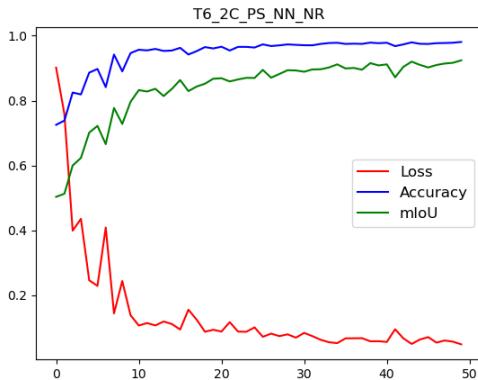


Abbildung 17: PointNet++ Trainingsergebnisse pro Epoche mit 2 Klassen, ohne Normalisierung und ohne zufällige Rotationen

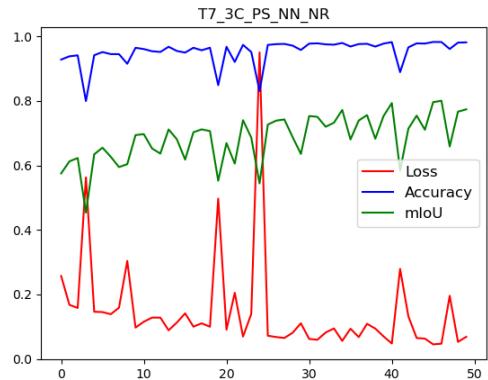


Abbildung 18: PointNet++ Trainingsergebnisse pro Epoche mit 3 Klassen, ohne Normalisierung und ohne zufällige Rotationen

se für das erneute Training sind in Abbildung 17 für zwei Klassen und in Abbildung 18 für drei Klassen zu sehen.

Gegenüber dem Training mit Normalisierung ist das Training ohne Normalisierung etwas schlechter. Insbesondere für das Netz mit drei Klassen haben die Ausreißer für Loss-, Accuracy- und IoU-Werte zugenommen. Das führt zu dem Schluss, dass PointNet++ mit Normalisierung genutzt werden sollt und der zusätzliche Rechenaufwand zum übertragen der Ergebnisse auf die vollständige Punktwolke in Kauf genommen werden sollt.

Vergleich Training mit und ohne zufällige Rotationen

Die Tests des Servers haben gezeigt, das es bei der Segmentierung der freigestellten Pflanzen zu starken Fehlern kommt, wenn diese rotiert sind. Das hat zu der Erkenntnis geführt, dass PointNet++ anfällig für die Ausrichtung von Punktwolken ist. Aus diesem Grund wurde das Training für das zwei und drei Klassen-Netz mit zufälligen Rotationen wiederholt um ein Robustheit gegen Rotationen anzutrainieren. Die Ergebnisse für zwei Klassen-Netz ist in Abbildung 19 zu sehen. Die Ergebnisse für das 3 Klassen-Netz in Abbildung 20. Beim Training von dem Netz mit drei Klassen sind starke Ausreißer in den Loss-Werten zu erkennen die sich auch in der Accuracy und dem IoU wiederspiegeln. Diese Beobachtung kann man bei dem anderen Netz nur in wesentlich kleinerem Ausmaß feststellen. Dennoch sind die Ergebnisse wesentlich schlechter für das Training mit zufälligen Rotationen als das Training ohne. Im Einsatz sollt also bei der Entfernung des Hintergrundes darauf geachtet werden, dass dieser zumindest grob an der XY-Ebene Ausgerichtet ist.

TODO: Bilder von fehlgeschlagener Segmentierung einfügen

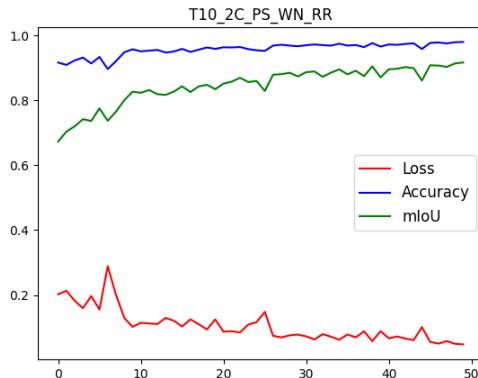


Abbildung 19: PointNet++ Trainings-Ergebnisse pro Epoche mit 2 Klassen, mit Normalisierung und zufälliger Rotationen

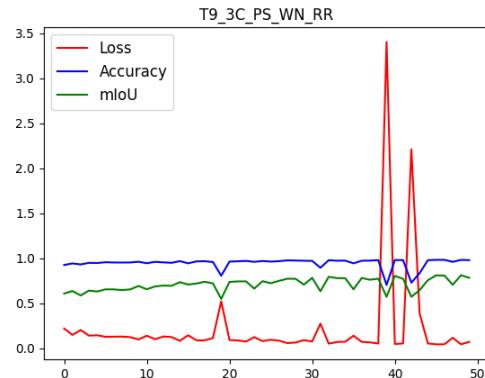


Abbildung 20: PointNet++ Trainings-Ergebnisse pro Epoche mit 3 Klassen, mit Normalisierung und zufälliger Rotationen

5 Fazit und Ausblick

Generierung der Punktwolken

Die Generierung von Punktwolken mit ODM liefert zwar gute Ergebnisse und ist vergleichsweise schnell. Allerdings ist es dennoch die Komponente die am meisten Laufzeit in Anspruch nimmt. Da keine Geo-Informationen zu den Bildern vorliegen, kann die GPU von ODM nicht eingesetzt werden. An der Stelle könnte ODM erweitert werden und die alternative Implementierung die ohne Geo-Informationen auskommt könnte parallelisiert werden.

Sollte sich in Zukunft bei Smartphones eine RGB-D Kamera als Standart etablieren, könnte diese Technologie genutzt werden um eine schnelle Generierung der Punktwolken zu ermöglichen.

Ein weiterer Ansatz der untersucht werden könnte ist die Generierung von Punktwolken mit Neuronalen Netzen. Hier gibt es einige interessante Ansätze die im Rahmen dieser Arbeit nicht untersucht wurden.

Registrierung

Die Ergebnisse der Registrierung sind zwar für den Zweck ausreichend, aber könnten zuverlässiger und genauer sein. Insbesondere die nicht bekannte Skalierung bereitet Probleme.

Bei der Datengewinnung könnte zu den Bildern auch eine GPS-Position ermittelt werden und so Punktwolken mit gleichem Maßstab erstellt werden. Das würde die Registrierung erleichtern, da die Skalierung nicht mehr geschätzt werden müsste.

Ein weiterer Ansatz der in dieser Arbeit nicht verfolgt wurde ist ScaleLK.

Segmentierung

Die Segmentierung ist für die Hintergrundentfernung noch nicht gut genug. Hier könnte probiert werden mit einem größerem Datensatz zu trainieren oder mehr Punkte während des Trainings zu nutzen.

Die Segmentierung der Pflanze kann erweitert werden um zusätzliche Informationen über eine Pflanze zu erhalten. Zum Beispiel können neben Blättern und Stielen auch Blüten und Früchte erkannt werden, was für die Analyse des Ertrags einer Fruchtfolge genutzt werden könnte. Um das zu bewerkstelligen muss für jedes Merkmal ein weiters Label hinzugefügt werden und die Datenbasis, die für das Training genutzt wird, um Punktwolken erweitert werden die diese Merkmale auch beinhalten.

Server

Um die Perfomance des Servers zu steigern könnte für die in c++ Implementierten Funktionalitäten ein Python-Binding erstellt werden. Bisher werden diese Funktionen über System-Kommandos angesteuert. Mit einem Python-Binding könnten diese direkt aus dem Python-Code angesprochen werden. Dadurch müssten die Daten zwischen den einzelnen Pipelines im Hauptspeicher gehalten werden und müssten nur noch zu Persistierungs-Zwecken auf der Festplatte abgelegt werden.

Literatur

- [1] I. Masih, S. Maskey, F. Mussá, and P. Trambauer, “A review of droughts on the african continent: A geospatial and long-term perspective,” *Hydrology and Earth System Sciences*, vol. 18, pp. 3635–3649, 09 2014.
- [2] N. Mehendale and S. Neoge, “Review on lidar technology,” *ssrn preprint ssrn:3604309*, 2020.
- [3] A. Taneja, “Top 10 smartphones with a dedicated depth sensor camera to capture perfect bokeh shots,” 2020.
- [4] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [5] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*, pp. 2564–2571, Ieee, 2011.
- [6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *European conference on computer vision*, pp. 778–792, Springer, 2010.
- [7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.
- [8] A. Alouache and Q. Wu, “Evaluation of an opencv implementation of structure from motion on open source data,”
- [9] pierotofy, “Open drone map - a command line toolkit to generate maps, point clouds, 3d models and dems from drone, balloon or kite images.,” 2020.
- [10] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, “Pixelwise View Selection for Unstructured Multi-View Stereo,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [11] H. Fan, H. Su, and L. Guibas, “A point set generation network for 3d object reconstruction from a single image,” 2016.
- [12] M. Tatarchenko, A. Dosovitskiy, and T. Brox, “Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs,” 2017.
- [13] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, “Pixel2mesh: Generating 3d mesh models from single rgb images,” 2018.
- [14] Y. Xia, C. Wang, Y. Xu, Y. Zang, W. Liu, J. Li, and U. Stilla, “Realpoint3d: Generating 3d point clouds from a single image of complex scenarios,” *Remote Sensing*, vol. 11, no. 22, 2019.
- [15] I. Ziamtsov and S. Navlakha, “Machine Learning Approaches to Improve Three Basic Plant Phenotyping Tasks Using Three-Dimensional Point Clouds1[OPEN],” *Plant Physiology*, vol. 181, pp. 1425–1440, 10 2019.

- [16] D. Li, Y. Cao, G. Shi, X. Cai, Y. Chen, S. Wang, and S. Yan, “An overlapping-free leaf segmentation method for plant point clouds,” *IEEE Access*, vol. 7, pp. 129054–129070, 2019.
- [17] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.
- [19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” *arXiv preprint arXiv:1706.02413*, 2017.
- [20] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” *ACM Transactions on Graphics (TOG)*, 2019.
- [21] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, International Society for Optics and Photonics, 1992.
- [22] J. Yang, H. Li, D. Campbell, and Y. Jia, “Go-icp: A globally optimal solution to 3d icp point-set registration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, p. 2241–2254, Nov 2016.
- [23] J. Zhang, Y. Yao, and B. Deng, “Fast and robust iterative closest point,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [24] K. Derpanis, “Overview of the ransac algorithm,” 2005.
- [25] T. Zinßer, J. Schmidt, and H. Niemann, “Point set registration with integrated scale estimation,” in *Point Set Registration with Integrated Scale Estimation*, 2005.
- [26] S. Xu and Y. Shang, “ScaleLK: Registration of point clouds with different scales using deep learning methods,” *IOP Conference Series: Materials Science and Engineering*, vol. 768, p. 072089, mar 2020.
- [27] Y. Wang and J. M. Solomon, “Deep closest point: Learning representations for point cloud registration,” in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.
- [28] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, “Pointnetlk: Robust and efficient point cloud registration using pointnet,” 2019.
- [29] Z. J. Yew and G. H. Lee, “Rpm-net: Robust point matching using learned features,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.
- [30] W. Yuan, B. Eckart, K. Kim, V. Jampani, D. Fox, and J. Kautz, “Deepgmr: Learning latent gaussian mixture models for registration,” 2020.
- [31] Y. Wang and J. M. Solomon, “Prnet: Self-supervised learning for partial-to-partial registration,” 2019.

- [32] V. Sarode, X. Li, H. Goforth, Y. Aoki, R. A. Srivatsan, S. Lucey, and H. Choset, “Pcnet: Point cloud registration network using pointnet encoding,” 2019.
- [33] R. Hänsch, T. Weber, and O. Hellwich, “Comparison of 3d interest point detectors and descriptors for point cloud fusion,” *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-3, 09 2014.
- [34] R. Sears, C. Van Ingen, and J. Gray, “To blob or not to blob: Large object storage in a database or a filesystem?,” *arXiv preprint cs/0701168*, 2007.