Analyse von Pflanzenwachstum auf Basis von 3D-Punkwolken

Masterarbeit
zur Erlangung des Grades *Master of Science*

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

vorgelegt von
Jakob Görner
1003660

Datum: 15. 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-Punkwolken

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 16 Seiten.

Mönchengladbach, 15. 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 Punkwolke erzeugt werden. Aus der Punktwolke sollen die Punkte die zur Pflanze gehören extrahiert werden um diese weiter zu analysieren. Die extrahierten Punkte werden zur weiteren Analyse in Stamm und Blatt Punkte segmentiert, die dann weiter analysiert werden können. 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 analysed over time. The application should need as few pictures as possible to reduce the amount of transfered data.

Inhaltsverzeichnis

1 Motivation

Diese Arbeit wird im Rahmen eines Projekts mit der Universität xx durchgeführt. Im Rahmen dieser Arbeit soll eine Anwendung entstehen, die in Uganda zum Einsatz kommen soll. Die Anwendung soll mit nicht invasiven Methoden zur Unterstützung bei der selektiven Züchtung eingesetzt werden. Ziel ist es 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 sind besondere Anforderungen an den Prozess der Datengewinnung und Übertragung, dass zum einem der Betrieb kostengünstig ist - das betrifft insbesondere den Endverbraucher, der möglichst wenig Datenmengen an den Server, der die Rechenkapazität bereit stellt übertragen muss. Desweiteren muss die Datengewinnung mit einem Mobiltelefon oder einer anderen Kamera möglich sein.Spezielle Aufnahmegeräte wie ein LIDAR-Scanner 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 Bilder eine Punkwolke generiert werden. Danach muss die Punktwolke Segmentiert werden um den Hintergrund zu entfernen und die einzelnen Teile der Pflanze 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 Punkwolke 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 [1] um Punkwolken direkt zu erzeugen. Es gibt eine Reihe von Sensoren die das ermöglichen. Da der Großteil der Sensoren aus kostengrünen 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 [2].

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 [3], ORB [4], BRIEF [5] oder SURF [6] eingesetzt werden. Wurden die Feature ermittelt müssen korrespondierende Feature zwischen den Bilder ermittelt werden. Das Bild-Paar mit den meisten übereinstimmungen wird als Basis für die Rekonstruktion der 3D-Punkwolke genutzt. Wurde aus den beiden Bildern eine initiale Punktwolke erstellt, wird diese mit den verbleibenden Bildern angereichert [7].

Einige Verfahren benötigen genaue Informationen über die Kamera Position, Ausrichtung und andere Angaben wie die Brennweite der Linse etc. andere wiederrum lesen diese Informationen aus den EXIF Header der Bildern direkt aus oder schätzen diese [8] [9].

Es wurden in den letzten Jahren große Fortschritte im Bereich Deep Learning auf 3D-Punkwolken gemacht, zur Gewinnung von 3D-Punkwolken [10] [11] [12]. Mit diesen Verfahren ist es möglich aus einer minimalen Datenbasis die auch nur aus einem Bild bestehen kann 3D-Punkwolken zu generieren. 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 [13] und wird daher nicht weiter untersucht. Der in [13] vorgestellte Ansatz geht die Problematik an und soll daher untersucht werden.

2.2 Segmentierung von 3D-Punkwolken

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 [14] [15] die das segmentieren von 3D Punktwolken von Pflanzen in Stiele und Blätter behandeln, viele dieser Ansätze haben aber das Problem, dass sie nur unter bestimmten Bedingungen gute Ergebnisse liefern. 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 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 Deep Learning auf 3D-Punkwolken 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[16]/PointNet++[17] oder DGCNN [18] auf Punktwolken von Pflanzen trainiert werden.

2.3 Registrierung von 3D-Punkwolken

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.

Die am weitesten verbreitete Method zur Registrierung von 3D Punktwolken ist Iterative Closest Points (ICP) [19]. ICP basiert auf dem Ansatz das zwei Punktwolken iterativ aneinander angenähert werden. Dabei wird bei jeder Iteration für jeden Punkt p_i aus der Quellpunkwolke 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 SVD die Rotation der R ermittelt und auf S_p angewandt. Der Vorgang wird wiederholt bis es zu einer Konvergenz kommt, also keine oder nur noch minimale Änderungen an der Punkwolke S_p vorkommen.

Ein Problem das ICP hat ist, das es auch für lokale Minima konvergiert. Das heißt, das die initiale Transformation von S_p so gewählt sein muss, dass der ICP-Durchlauf bei einem globalen Minimum konvergiert. Ansätze wie global ICP, 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 [20].

Ein weiteres Problem ist, dass die meisten ICP-Ansätze die Skalierung nicht beachten. Es gibt einige wenige Ansätze für dieses Problem [21].

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. Bekannt Ansätze sind DCP [22], PointNetLK [23], RPM-Net [24], DeepGMR [25], PRNet [26] und PCRNet [27].

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 Punkwolken berücksichtigt werden.

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 Hintegrund zu registrieren. So wird ein Verhältnis geschaffen das dem der realität entspricht. Der zweite Ansatz basiert darauf das die Registrierung direkt zwischen zwei Zeitpunkten erfolgt. Hierbei kann es zu Abweichungen von der Realität kommen, wenn die Pflanzen skaliert werden müssen für die Registrierung. Der letzte Ansatz soll überprüfen ob es mittels eines platziertem Objekts möglich ist die skalierung der Punktwolke zu ermitteln und so das Problem beim direkten Vergleich zweier Szenen lösen soll.

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 die 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 Punkwolke 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.

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 aus zu führen muss eine Ordner-Struktur erstellt werden. In dieser Ordner-Struktur muss ein Ordner ïmagesënthalten 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 werden, dass der Docker mit den selben Rechten wie der User arbeitet. Ansonsten werden

per Standarteinstellung Root-Rechte genutzt. Das führt dazu das auf die erstellten Daten kein Zugriff mehr existiert, was das Aufräumen erschwert.

3.3 Umsetzung Registrierung zweier Punktwolken

Bei der Umsetzung der Registrierung muss die Besonderheit beachtet werden, dass die Skalierung der Punktwolken nicht bekannt ist und sich unterscheiden. Das heißt die Punkwolken 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-Punktwolke S und T nach der größten Ebene in der Punkwolke und richtet die Punkwolke so aus, dass die gefundene Ebene auf der Ebene liegt, die die x und y Achse bilden. Des weiteren werden aus beiden Punkwolken nur ein Teilmenge P_c der Punkte in einem Radius r um dem Punkt c der das Zentrum der Masse der Punktwolken 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 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 war 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 $3X3$ Rotations-Matrix R und dem Translations-Vektor $\vec{(t)}$ die Translations-Matrix T zurück geliefert wird. Um das zu Erreichen muss die Eingabe der Größe $NX3$ auf eine Eingabe der Größe $NX4$ 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 $4X4$ 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 dursucht wird. Wieder werden die Quell- und Ziel-Punktwolke an der XY-Achse ausgerichtet und eine Teilmenge P_c um das Zentrum c entnommen wie in dem 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 könnt 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 sodas 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 [14] 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 größere 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.

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

Aus den Abbildungen $\vec{m}_j$ für alle P_i kann die Kovarianzmatrix C_i berechnet werden.

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

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.

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

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

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

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 Punkwolke mit einbezogen, können auch Krankheitsbilder wie vertrocknende Blätter in die Repräsentation eingeschlossen werden. Da die Ergebnisse für PointNet nicht gut genug waren wurde eine verbesserte Version PointNet++ auf dem Datensatz trainiert. Diese wurde mit und ohne Normalen, mit 2 Labelen ohne Hintergrund und mit 3 Labeln mit Hintergrund trainiert.

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

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

3.5 Umsetzung Pipelines und 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 wird der aktuelle Stand der Datenhaltung eingelesen und der Status des Servers aufgebaut. Bei der Datenhaltung wurde auf eine Datenbank verzichtet, da die meisten Daten als Blob vorliegen und daher nicht geeignet 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:

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 die Job-Queue hinzugefügt.

PUT /data/{Messreihe}/{Zeitstempel}

Dient dazu Pipelines für einen Datensatz zu starten. Hierzu wird im Payload des Request an den Server eine JSON-Datei übermittelt welches eine Liste der Pipelines enthält (Beispiel siehe Listing 1). Die spezifizierten Pipelines werden, in der angegebene Reihenfolge, der Job-Queue hinzugefügt.

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

Listing 1: Beispiel Payload zum starten einer Pipeline

Es stehen folgende Pipelines zur Verfügung:

- Punkwolke generieren
- In Shapnet-Format überführen
- Segmentierung Hintergrund
- Segmentierung Pflanze
- Blatt/Stamm Trennung
- Blätter zählen
- In Registrierungs-Format überführen
- Hintergrund-Registrierung
- Größen berechnen

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.

	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.

4 Ergebnisse

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

Um eine geeignete Software zur Generierung von 3D-Punktwolken auszuwählen muss erst entschieden werden nach welchen Gütekriterien verschiedene Anwendungen 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 das wichtigste Kriterium. Um dieses Kriterium zu messen 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 zu lassen. Unter den erstellten Punktwolken kann die ausgesucht werden die die Szene am besten repräsentiert. Das trifft auf die Punkwolke 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. Weitere Maße um die Güte des Goldstandarts zu bewerten ist die Anzahl der Punkte in der Wolke oder die Dichte der Punkte in der normalisierten Punktwolke.

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 Punkwolken mit dem Goldstandart registriert ist.

Ein weiterer Punkt bei der Bewertung des Verfahrens zur Generierung von 3D Punktwolken ist die Zeit die benötigt wird die Punkwolke zu generieren.

4.2 Vergleich von Verfahren zur Registrierung 3D-Punkwolken

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 in 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 learning3dïmplementiert und nutzen die selbe Datenbasis ModelNet40 für das Training. Für DCP und RPM-Net wurden das vortrainierte Model 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

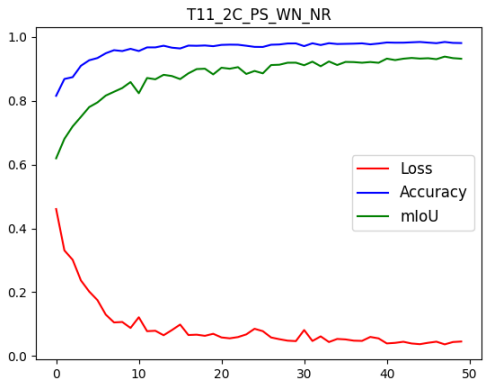

Abbildung 1: PointNet++ Trainings-Ergebnisse pro Epoche mit 2 Klassen und Normalisierung ohne zufällige Rotationen

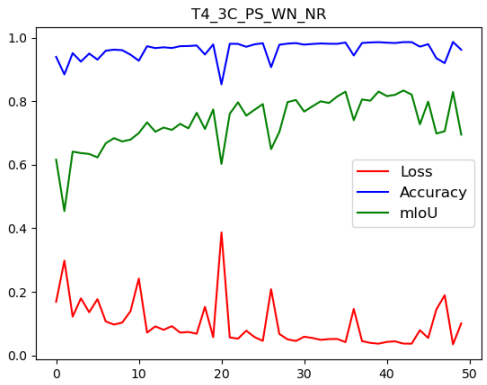

Abbildung 2: PointNet++ Trainings-Ergebnisse pro Epoche mit 3 Klassen und Normalisierung ohne zufällige Rotationen

getestet.

Vergleich DCP / PointNetLK

Verleich NN / ICP

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

Als Datenbasis wurden 144 handgelabelte Punkwolken 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 weiteren sind die Daten so aufbereitet das der Boden der Punktwolke an der XY-Ebene ausgerichtet ist und der Stiel bei geradem Wachstum richtung z-Achse zeigt. Des weiteren sind die Punkte in der Punktwolke auf 1 normiert. Das heißt sie liegen in dem quadratischen 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/ohne Hintergrund Mit Hintergrund -¿ lohnt es sich nur das Zentrum zu betrachten?

Vergleich Training mit/ohne Normalen

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

Da die Punkwolken für die Segmentierung auf eine bestimmte Größe reduziert werden, muss die Schätzung für jeden Punkt aus dem Ergebnis der Segmentierung zurück auf die vollständige Punktwolke übertragen werden um die Informations-Reduktion zu minimieren. Dafür muss reduzierte Punktwolke ihre Position nicht ändern. Um das zu gewährleisten muss die Normali-

	t4 mit normalisiert Daten	t4 mit nicht normalisiert Daten	t11 mit normalisiert Daten	t11 mit nicht normalisiert Daten
Loss	$0,038$	$0,175$	–	–
Genauigkeit	$0,986$	$0,946$	–	–
IoU	$0,84$	$0,684$	–	–

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

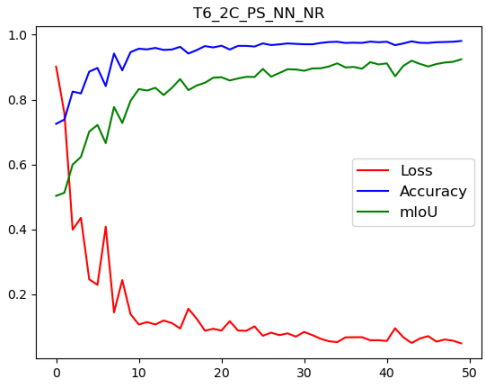

Abbildung 3: PointNet++ Trainings-Ergebnisse pro Epoche mit 2 Klassen, ohne Normalisierung und ohne zufällige Rotationen

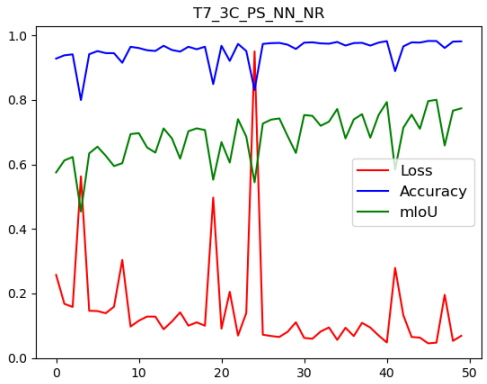

Abbildung 4: PointNet++ Trainings-Ergebnisse pro Epoche mit 3 Klassen, ohne Normalisierung und ohne zufällige Rotationen

sierung die PointNet++ ausführt verhindert oder umgekehrt werden. Wird die Normalisierung verhindert, hat das Auswirkungen auf die Güte der Ergebnisse der Segmentierung.

Da die Ergebnisse sich Stark verschlechtern, wenn die Daten nicht normalisiert werden wird PointNet++ ohne Normalisierung erneut trainiert. Die Ergebnisse für das erneute Training sind in Abbildung 3 für zwei Klassen und in Abbildung 4 für drei Klassen zu sehen.

Gegenüber dem Training mit der 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.

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 5 zu sehen. Die Ergebnisse für das 3 Klassen-Netz in Abbildung 6. 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. Im Einsatz muss bei der Entfernung des Hintergrundes darauf geachtet werden, dass dieser zumindest grob Ausgerichtet ist.

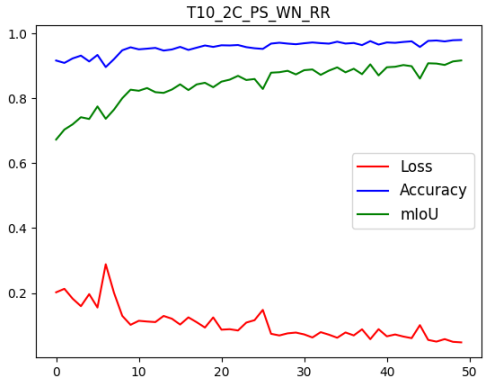

Abbildung 5: PointNet++ Trainings-Ergebnisse pro Epoche mit 2 Klassen, mit Normalisierung und zufälliger Rotationen

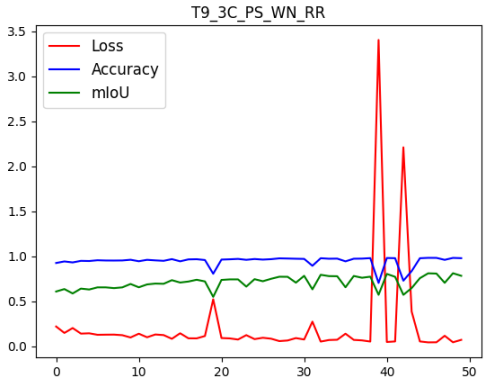

Abbildung 6: PointNet++ Trainings-Ergebnisse pro Epoche mit 3 Klassen, mit Normalisierung und zufälliger Rotationen

5 Fazit und Ausblick

Generierung der Punkwolken

Die Generierung von Punktwolken mit ODM liefert zwar gute Ergebnisse und ist vergleichsweise schnell. Allerdings ist es dennoch die Komponente die am meisten Laufzeit verbraucht. Da keine Geo-Informationen zu den Bilder 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.

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.

Literatur

[1] N. Mehendale and S. Neoge, "Review on lidar technology," *ssrn preprint ssrn:3604309*, 2020.

[2] A. Taneja, "Top 10 smartphones with a dedicated depth sensor camera to capture perfect bokeh shots," 2020.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] 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.

[5] 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.

[6] 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.

[7] A. Alouache and Q. Wu, "Evaluation of an opencv implementation of structure from motion on open source data,"

[8] pierotofy, "Open drone map - a command line toolkit to generate maps, point clouds, 3d models and dems from drone, balloon or kite images.," 2020.

[9] 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.

[10] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3d object reconstruction from a single image," 2016.

[11] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," 2017.

[12] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," 2018.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] J. Zhang, Y. Yao, and B. Deng, "Fast and robust iterative closest point," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[21] T. Zinßer, J. Schmidt, and H. Niemann, "Point set registration with integrated scale estimation," in *Point Set Registration with Integrated Scale Estimation*, 2005.

[22] 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.

[23] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "Pointnetlk: Robust and efficient point cloud registration using pointnet," 2019.

[24] 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.

[25] W. Yuan, B. Eckart, K. Kim, V. Jampani, D. Fox, and J. Kautz, "Deepgmr: Learning latent gaussian mixture models for registration," 2020.

[26] Y. Wang and J. M. Solomon, "Prnet: Self-supervised learning for partial-to-partial registration," 2019.

[27] V. Sarode, X. Li, H. Goforth, Y. Aoki, R. A. Srivatsan, S. Lucey, and H. Choset, "Pcrnet: Point cloud registration network using pointnet encoding," 2019.