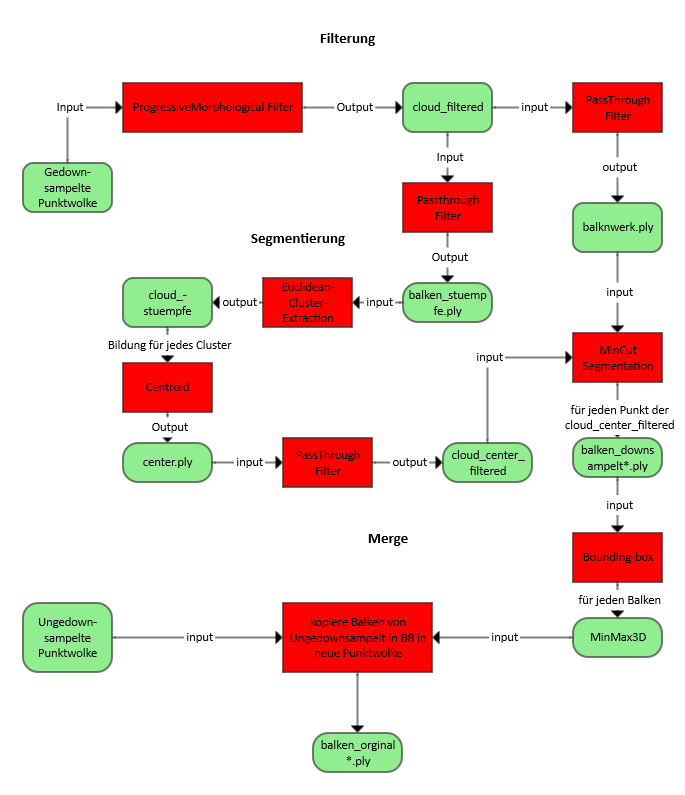
## Segmentierung

* + 1. **Der Workflow**

In der folgenden Abbildung 8 ist der Workflow als Gesamtheit aller Inputs und Outputs, mit den durchgeführten Algorithmen abgebildet. Dieser schließt sich an den ersten Schritt der Datenaufbereitung an und kann in die drei Schritte: Filterung -> Segmentierung -> Merge zusammengefasst werden. In der Abbildung stellen dabei die grünen Felder die Inputs dar und die rot gefärbten Kästen die angewendeten Algorithmen aus der PCL Library.



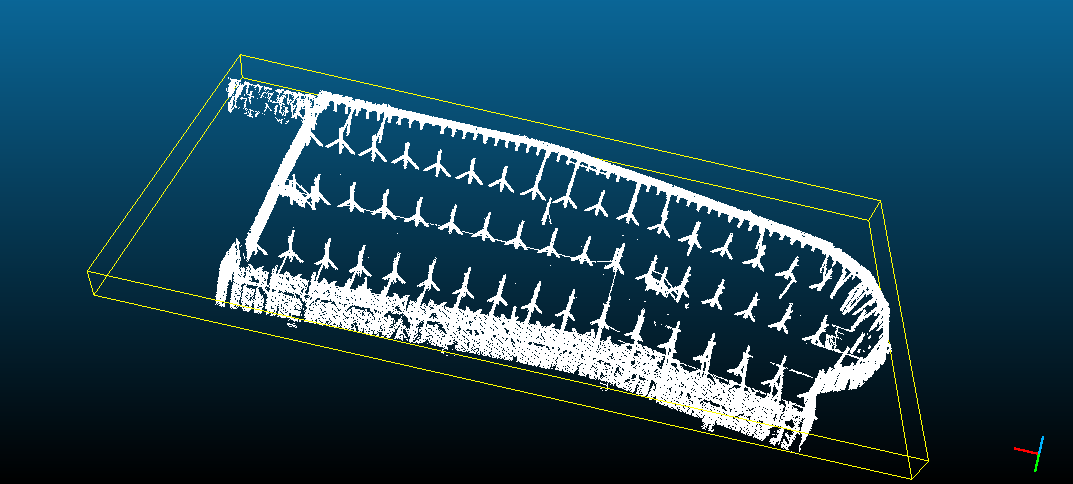
**Abb. 1:** Balkenwerk nach Filterung von Decke und Boden.

In den folgenden Kapiteln soll auf die drei Schritte näher eingegangen werden. Dabei werden die verwendeten Algorithmen kurz vorgestellt und am Schluss das Ergebnis zusammengefasst.

### Filterung des Bodens und des Dachs

Um im zweiten Schritt eine Extraktion der einzelnen Balken durchzuführen, mussten zu Beginn das Dach und der Boden entfernt werden. Für den Boden hat sich der morphologische Filter als der geeignetste Algorithmus herausgestellt. Dieser Filter trennt Boden-(Gelände-)Punkte von den ”Nicht-Boden“-Punkten, indem ein Strukturelement erstellt wird, welches die Geländehöhenunterschiede in Abhängigkeit von der Entfernung beschreibt. Je größer die horizontale Entfernung, desto größer kann der Höhenunterschied zwischen den Bodenpunkten sein.

Für die Filterung wird dieses Strukturelement vertikal beginnend unterhalb der Punktmenge nach oben verschoben. Dazu wird es planimetrisch auf jedem Punkt zentriert und nach oben geschoben, bis es an einen Punkt innerhalb des Umkreises angepasst wird und die Verschiebung gestoppt wird. Der untersuchte Punkt ist kein Geländepunkt, wenn er nicht das Zentrum ist, auf den das Strukturelement zentriert wurde. Auf diese Weise wird die gesamte Punktwolke klassifiziert (Sithole & Vosselman, 2003). Der Algorithmus ist in PCL als *ProgressiveMorphologicalFilter* (Pointclouds.org, 2017) implementiert.



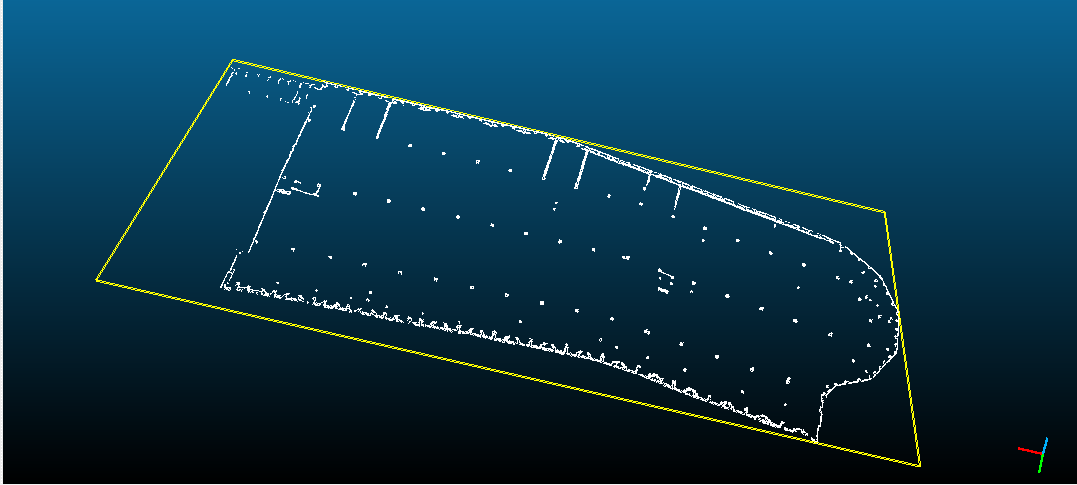
**Abb. 2:** Balkenwerk nach Filterung von Decke und Boden.

Da der Algorithmus sehr rechenintensiv ist, kam als Input nur die downgesampelte Punktwolke („merged.pcd“) mit einer Blattgröße von 5cm aus dem vorangegangenen Datenaufbereitungsschritt, in Frage. Die Anwendung auf die originale Punktwolke wurde nach drei Tagen Rechenzeit abgebrochen. Um jedoch eine größtmögliche Punktdichte zu erreichen, wurde im Anschluss eine Zusammenführung (Merge) mit der Originalen Punktwolke durchgeführt (Kapitel 3.2.4.).

Die Entfernung des Daches aus der Punktwolke, wurde mit einem *PassThrough* Filter (Pointclouds.org, 2013) realisiert, welcher Punkte entfernt die nicht in einem festgelegten Bereich liegen. Die Parameter des *PassThrough* Filters konnten mit Hilfe des Programms *CloudCompare*[[1]](#footnote-1) bestimmt werden. Der gefilterte Bereich wurde für die z-Koordinate auf 0 m und 3 m gesetzt. Abbildung 2 dokumentiert das Ergebnis der Filterung von Decke und Boden.

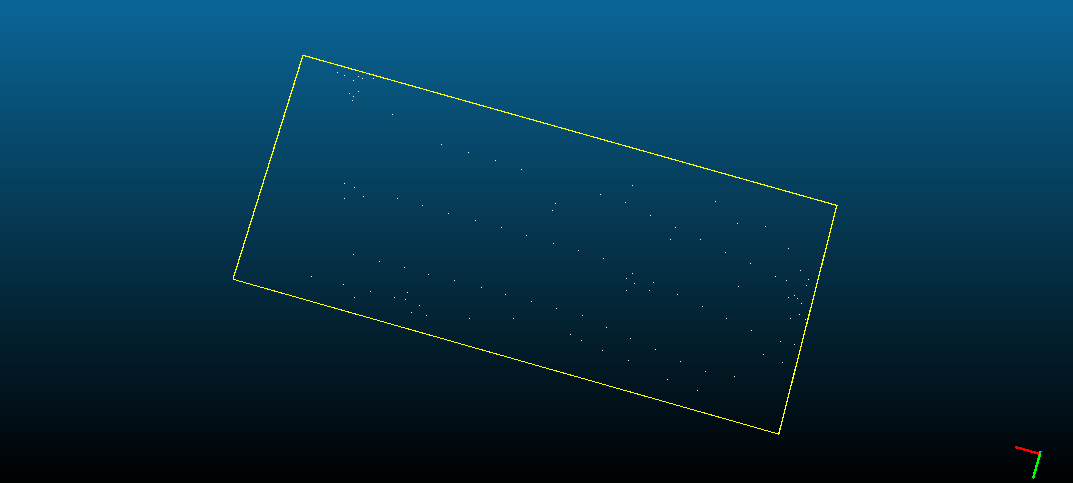
### Clusterung und Segmentierung

Nach dem Filterprozess erfolgte die Extraktion der Balken durch eine *Clusterung* und Segmentierung. Das Ziel der Clusterung bestand darin, möglichst alle Balkengrundrisse zu bekommen, indem die *EuclideanClusterExtraction* (Pointcloud.org, 2003) auf eine vorher gefilterte „Balkenstumpf“-Punktwolke angewandt wurde. Hierfür wurde erneut ein *PassThrough* Filter verwendet. Als Ergebnis wurden alle irrelevanten und störenden Elemente bis auf die Seitenwände entfernt, was mit der Parametrisierung der z-Achse zwischen 1.8 m und 2 m erreicht werden konnte. Abbildung 3 stellt den Output des *PassThrough* Filters dar. Das Ergebnis wurde als PLY-Datei unter dem Namen „balkenstuempfe.ply“ gespeichert und diente als Input für die *Clusterung*.



**Abb. 3:** GefilterteBalkenstümpfe für das *EuclideanClusterung*

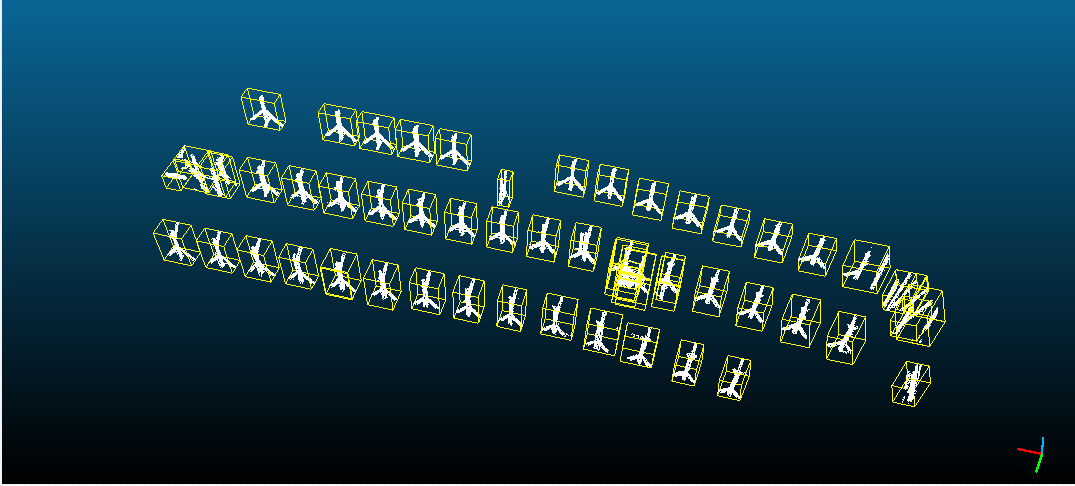
Für die *Clusterung* der Balken wurde das euklidische Clusterverfahren verwendet, welches ebenfalls in der PCL-Library, als *EuclideanClusterExtraction* (Pointcloud.org, 2003) zur Verfügung gestellt wird. Die geclusterten Balkenstümpfe wurden in einer Punktwolke gespeichert. Um eine Segmentierung der Balken anhand der Clusterungsergebnisse durchzuführen, ist es essentiell den Mittelpunkt zu ermitteln, mit welchen im darauffolgenden Schritt der *MinCut* (Pointcloud.org, 2014,2) Segmentierungsalgorithmus angewandt werden konnte. Aus diesem Grund wurde in jedes Cluster ein *Centroid* eingepasst, welcher den jeweiligen Mittelpunkt des Clusters darstellte. Die Menge aller „Centroid-Punkte“, wurde in einer weiteren Punktwolke gespeichert(„center.ply“). Abbildung 4 zeigt diese Punktwolke mit den Mittelpunkten der Cluster.



**Abb. 4:** Punktwolke der Centroide, welche anhand der Cluster gebildet wurden.

Die „Centroid-Punktwolke“ enthielt jedoch auch Punkte für die Seitenwände, welche nicht das Balkenwerk widerspiegelten, das extrahiert werden sollte. Deshalb wurde erneut der *PassThrough* -Filter zur Entfernung dieser Punkte angewandt. Der gefilterte Bereich für die y-Koordinate lag zwischen -7 m und 14 m und konnte mit Hilfe von CloudCompare ermittelt werden.

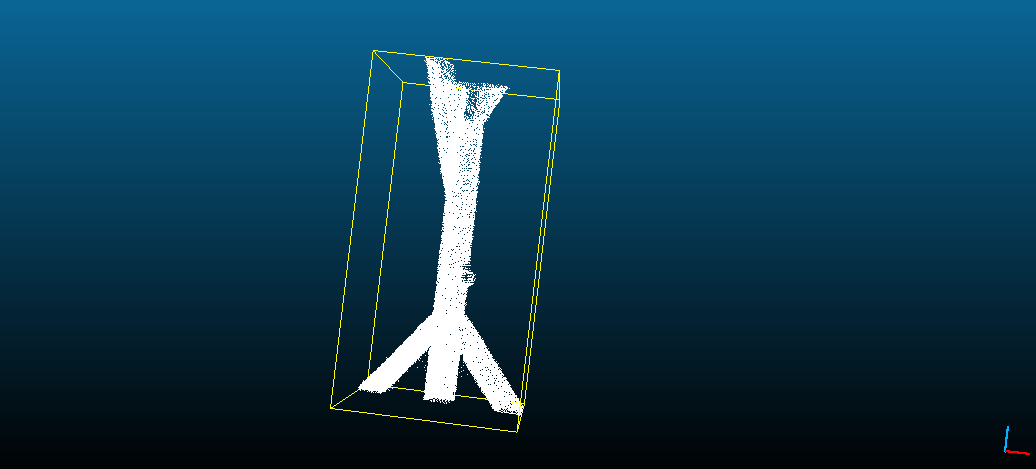
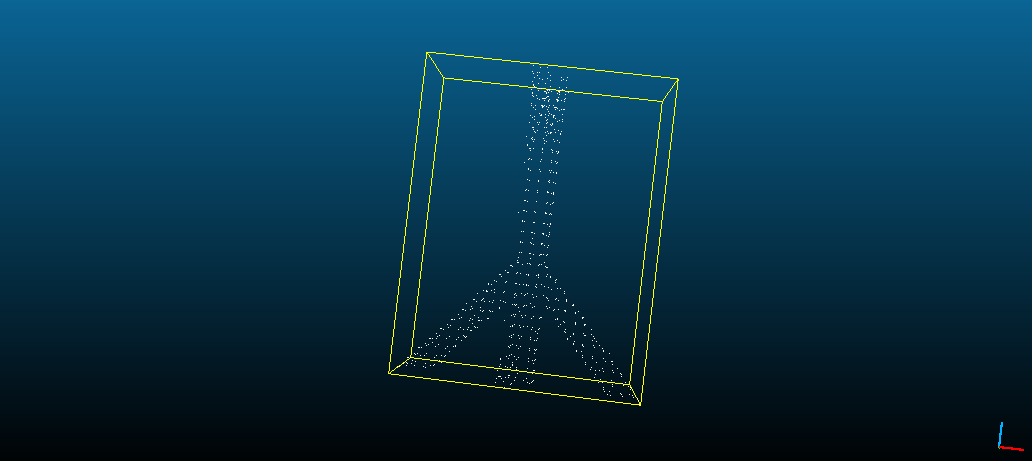
Als nächstes erfolgt die Segmentierung der Balkenkomplexe, aus der downgesampleten Gesamt-Punktwolke mittels der *MinCut*-Segmentierung. Dieser Algorithmus führt eine binäre Zerlegung der Eingangsdaten anhand eines Mittelpunktes und eines Radius, in Vorder- und Hintergrundpunkte (Punkte, die zum Objekt gehören und welche, die nicht dazu gehören), durch. Für die Segmentierung wurde die „Centroid-Punktwolke“ iteriert und jeder iterierte Punkt als Mittelpunkt gesetzt, der Radius betrug 1,5m. Jeder so gefilterte Balkenkomplex wurde in eine separate Ausgabedatei gespeichert („balken\_downsampelt\*.ply“). Als Erweiterung der Iteration diente ein Array, welches die Anzahl der extrahierten Balkenkomplexe speicherte. Dafür wurde jeder extrahierte Balken gezählt und der Wert im Array gespeichert. Der Maximalwert des Arrays stellte die Anzahl der gefilterten Balken dar.



**Abb. 4:** Ergebnis der *MinCut*-Segmentierung mit den einzelnen Balkenkomplexe

### Zusammenführen der Gedownsampelten Balkenkomplexe mit der Originalen Punktwolke

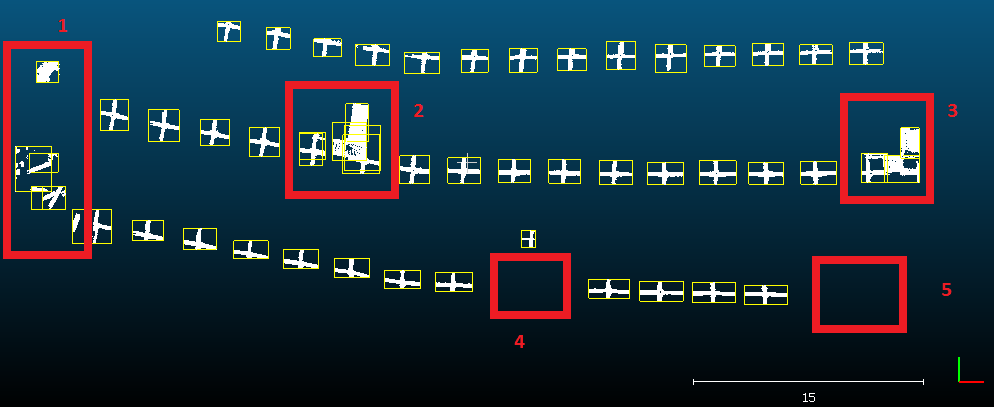
Die extrahierten Balkenkomplexe wiesen nur eine sehr geringe Punktdichte auf, was auf die Verwendung der downgesampelten Punktwolke zurückzuführen ist. Für ein besseres Ergebnis musste die originale Punktwolke einbezogen werden, welche die höchste Punktdichte besaß. Für diesen Schritt wurde über alle extrahierten Balkenkomplexe iteriert und die Bounding Box berechnet. Darauffolgend wurde geprüft, welche Punkte aus der originalen Punktwolke innerhalb der Bounding Box lagen. Alle Punkte innerhalb der Bounding Box, welche zur Originalen Punktwolke gehörten, wurden in eine neue Punktwolke kopiert und in einer Ausgabedatei gespeichert („balken\_orginal\*.ply“). Die Anzahl der Punkte je Balkenkomplex konnte dadurch von durchschnittlich 700 auf 50.000 erhöht werden.



**Abb. 5:** Vergleich Extraktion eines Balkenkomplexes aus downgesampelten und originalen Punktwolke, mit unterschiedlichen Punktdichten

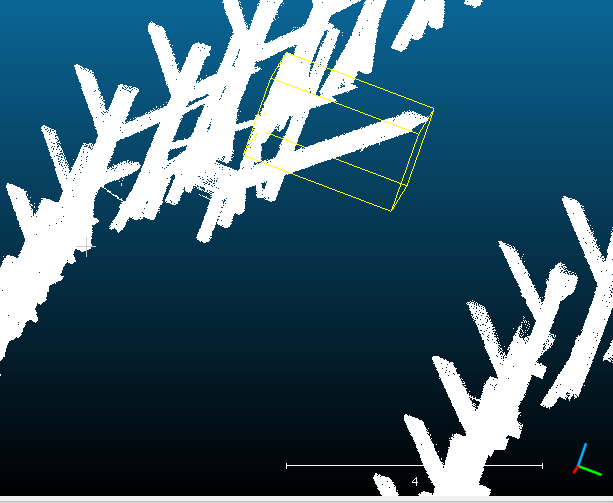
In PCL ist leider keine automatische Berrechnung der Bounding Box vorhanden, weshalb mittels der Methode *pcl::getMinMax3D*(Pointcloud.org, 2014) die Maximalwerte für X,Y und Z ermittelt werden konnte. Anhand dieser Werte konnte die ZUsammenführung durchgeführt werden.

### Ergebnis der Segmentierung

Mit der Segmentierung und Zusammenführung der originalen Punktwolke wurden 56 Balken automatisch extrahiert. Die originale Punktwolke besitzt jedoch nur 51 Balken, womit 5 falsch extrahiert wurden. In Abbildung 7 sind die Problemfälle dokumentiert.

**Abb. 6:** Falsch extrahierte Balken mit den Fällen 1 bis 5

Durch eine fehlerhafte Clusterung wurden einige Balkenkomplexe mehrfach Segmentiert (Fall 2 und 3). Hier liegt das Problem an zwei Leitern, welche als Balkenkomplexe geclustert wurden. Diese Problematik bildet Abbildung 7 ab. Dabei konnte im Laufe dieser Arbeit das Problem nicht behoben werden. Es wurde versucht die *Clusterung* durch einen kleineren Radius fehlerfreier zu gestalten, was aber zu falschen Ergebnissen bei den Balken führte.

**Abb. 7:** Leiter als Grund für falsche Extraktion

In den Fällen 4 und 5 wurden keine Balkenkomplexe erkannt, weil diese innerhalb der *MinCut*-Segmentierung außerhalb der Schwellenwerte lagen und somit nicht segmentiert wurden. In diesem Fall müsste geschaut werden inwieweit es möglich wäre mit erneuten Iterationen über die Ergebnisse der *Clusterung* die Schwellenwerte automatisch anpassen zu können. Bei den verschiedenen Fällen in Punkt 1 handelt es sich um Wände, welche den Balken vorgelagert sind. Diese konnten durch den *PassThrough* Filter und der Setzung von Schwellenwerten bei der Euklidischen *Clusterung* nicht entfernt werden, da ihre Punktmenge ähnlich der Balkenkomplexe war.

1. http://www.danielgm.net/cc/ [↑](#footnote-ref-1)