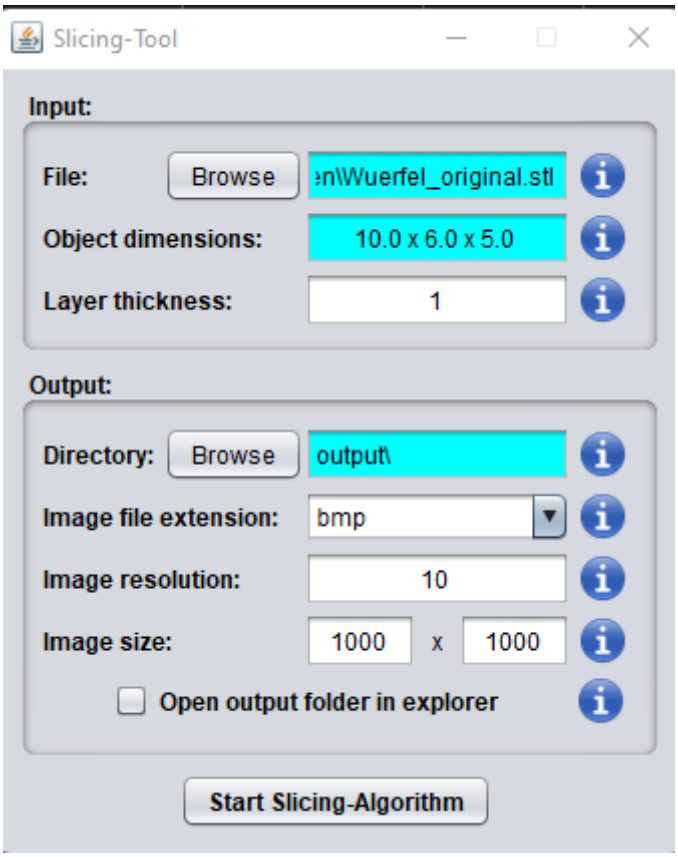
Objekttransformation – API

Ziel ist es die Transformation einer STL-Datei zu bestimmen, welche nach der Verarbeitung/Optimierung durch eine 3D-Druck Software verändert werden kann. Es soll somit die neue Ausrichtung eines Bauteiles bestimmt werden, um dieses in Folgeschritten weiter automatisiert verarbeiten zu können.

Das Slicing-Tool sollte eine Layer thickness von 1 besitzen. Außerdem sollte die Image resolution 10 betragen.



Klasse Main

path:

Variable für die Pfadangabe zu dem Ordnermit den gesliceten Bitmap Dateien/Bildern. (Bilder enthalten die Zieltransformation)

path_stl:

Pfadangabe zu dem Ordner mit der STL-Datei. (Ausgangsdatei welche transformiert **werden soll)**

downsample:

Faktor zur Reduzierung der Punktewolke. Standardmäßig wird der Faktor auf 1 gesetzt. Steht er z.B. auf 3, so wird nur jeder 3te Punkt in der Punktewolke behalten.

Klasse STLReader (Modul Cloud)

STLReader():

Konstruktor erzeugt ein STLReader Objekt, mit dem STL-Dateien verarbeitet werden können.

readMesh(dateiname):

Funktion welche eine STL-Datei einliest und diese in eine Form bringt, welche für die weitere Verarbeitung notwendig ist. Die Funktion gibt eine Liste mit den Dreiecken, eine Liste mit den Normalenvektoren und eine Liste mit allen Punkten zurück.
dateiname: Pfadangabe zu der STL-Datei

Klasse Punktewolke (Modul Cloud)

Punktewolke(path, zAbstand, downsample):

Konstruktor zur Erzeugung eines Punktewolken Objektes. Aufgabe der Klasse ist es, die Bitmap Dateien einzulesen und eine Punktewolke aus den Bildern zu erzeugen.
path: Der Pfad zu den Bitmap-Dateien
zAbstand: Abstand zwischen den Schichten in der Punktewolke. Dieser sollte 1 betragen, da die Layer-Thickness in dem Slicing-Tool auch auf 1mm gestellt wurde.
downsample: siehe downsample in Klasse Main

pointcloud(visualize) :

Funktion welche aus den Bitmap-Dateien eine Punktewolke erzeugt und diese in einer Liste ablegt. Die Punkte besitzen dabei klassisch drei Koordinaten (x, y, z). Die Funktion gibt die Liste zurück.
visualize: der Parameter vom Typ boolean erwartet True oder False. Es wird True übergeben, wenn die Punktewolke visualisiert werden soll. Andernfalls wird False überegeben.

downsample_list(points, num_points, teiler):

Funktion welche eine Punktewolke in Form einer Liste bekommt und die Anzahl der Punkte in dieser reduziert nach dem Faktor teiler. Die Funktion gibt als ersten Parameter die reduzierte Liste und als zweiten Parameter die Anzahl der neuen Punkte zurück.
points: Punktewolke in Form einer Liste
num_points: Anzahl der Punkte in der Liste
teiler: erwartet eine Ganzzahl, welche angibt wie viele Punkte aus der Punktewolke entfernt werden. Z.B. teiler = 3 bedeutet, dass nur jeder dritte Punkt in der Punktewolke übernommen wird.

Klasse BMP (Modul BMP):

BMP(path, data):

Konstruktor zur Erzeugung eines BMP Objektes. Aufgabe der Klasse ist es, die Bitmap Dateien einzulesen und in ein numpy Array umzuwandeln.
path: Pfadangabe zu den Bitmap-Dateien
data: Dateiname der jeweiligen Bitmap-Datei

bmp_to_array():

Funktion liest die Bitmap Datei ein und wandelt diese in ein numpy Array um. Das numpy Array wird zurückgegeben.

Klasse Shaping (Modul Shaping)

Shaping():

Konstruktor zur Erzeugung eines Shaping Objektes. Aufgabe der Klasse ist es, Die STL-Datei und die Punktewolke zu nehmen und zwischen diesen die Transformation zu berechnen.

objectEstimation(cloud, numberTriangles, downsample, showNormals):

Funktion erzeugt ein einzigartiges Dreieck in der Punktewolke. Hierfür werden zufällig drei Punkte gewählt, bei denen maximal zwei auf der gleichen Ebene liegen dürfen. Für jeden Punkt wird ein Dreieck mit seinen Nachbarn erzeugt. Die Punkte der Dreiecke und ihre Normalenvektoren werden abgespeichert. Auf jedem Dreieck wird ein zufälliger Punkt gewählt und von diesem ausgehend ein großes Dreieck (zwischen den drei zufälligen Punkten) gezogen. Auch hierzu wird der Normalenvektor bestimmt. Zu diesem uniquen Dreieck werden nun die wichtigsten Eigenschaften abgelegt. Die Kantenlängen und die Winkel zwischen den Normalenvektoren der kleinen Dreiecke und dem Normalenvektor des großen aufgespannten Dreiecks.
Die Funktion gibt die plotCloud und den triangle_index zurück, welcher für Visualisierungszwecke benötigt wird. UniqueData und randomTrPoints, werden für den Vergleich mit der STL-Datei benötigt. Sie enthalten die Distanzen und Winkel. Die randomTrPoints sind die zufälligen Punkte auf den kleinen Dreiecksflächen.
cloud: die Punktewolke
numberTriangles: Anzahl der kleinen Dreiecke, welche erezugt werden sollen. (Standardmäßig steht der Wert auf 3)
downsample: Faktor um den die Punktewolke reduziert wurde
showNormals: Boolesche Variable, welche bei dem Wert True die Normalenvektoren in den Nullpunkt schiebt und visualisiert.

objectTransformation(dataName, wolke, uniqueData, randomTrPoints, testCases, numberTriangles):

Funktion steuert das Zusammenspiel bei der Erzeugung von Dreiecken in der STL-Datei bis hin zur Berechnung der Transformation. Mit dem Aufruf dieser Funktion wird somit, bei vorhandener Punktewolke und vorhandere uniquer Daten, der Algorithmus zur Berechnung der Objekttransformation gestartet. Zurückgegeben wird die Transformationsmatrix und der Translationsvektor.
dataName: Pfadangabe zu der STL-Datei
wolke: Die Punktewolke

uniqueData: Die Daten des einzigartigen Dreiecks aus der Punktewolke
randomTrPoints: Zufälligen Punkte in der Punktewolke, welche das unique Dreieck aufspannen
testCases: Anzahl der zufälligen Dreiecke welche in der STL-Datei erzeugt werden
numberTriangles: Anzahl der kleinen Dreiecke, welche für einen Durchlauf benötigt werden

uniqueTriangleData(randomTrPoint, normals, cloud):

Funktion, welche die Winkel zwischen den Normalenvektoren der Dreiecke und dem Normalenvektor des uniquen aufgespannten Dreiecks berechnet. Zusätzlich werden die Kantenlängen des einzigartigen Dreiecks berechnet und abgelegt. Die Funktion gibt die Kantenlängen über uniqueData und die berechneten Winkel über normalUniqueVektor zurück.
randomTrPoint: zufälligen Punkte auf den kleinen Dreiecken
normals: Normalenvektoren der kleinen Dreiecke
cloud: Die Punktewolke

angleDegrees(normals, uniqueNormal, uniqueData):

Diese Funktion enthält die Formel, welche die Winkel zwsichen Vektoren berechnet. Funktion erweitert die uniqueData Liste um die Winkelinformationen.
normals: enthält die Normalenvektoren der kleinen Dreiecke
uniqueNormal: Normalenvektor des einzigartigen Dreiecks
uniqueData: Enthält bereits die Kantenlängen des uniquen Dreiecks

random_triangle_point(triangleKoord):

Funktion berechnet einen zufälligen Punkt, welcher innerhalb eines Dreiecks liegt. Zurückgegeben wird hier eine Liste mit den Koordinaten des Punktes.
triangleKoord: Koordinaten des Dreiecks

randomPoints(cloud, numberTriangles):

Diese Funktion sucht drei zufällige Punkte aus der Punktewolke heraus. Die Bedingung muss erfüllt werden, dass mindestens ein Punkt auf einer anderen Ebene liegt. Es wird somit vermieden eine Dreieck zu erzeugen, welches auf einer Fläche Außenfläche der Punktewolke liegt. Es werden die drei Punkte in einer Liste zurückgegeben.
cloud: Die Punktewolke
numberTriangles: Anzahl der Dreiecke, welche erzeugt werden sollen.

randomShape2d(cloud, punkt, downsample):

Funktion berechnet zu einem Zufallspunkt seinen nächsten Nachbarn auf der gleichen Ebene z (Ebene z ist die aktuelle Schicht/Höhe in der Punktewolke, auf welcher der zufällige Startpunkt für ein Dreieck bestimmt wurde). Zurückgegeben werden die gefundenen Kanten in einer Form einer Liste.
cloud: Die Punktewolke

punkt: zufälliger Startpunkt
downsample: Faktor zur Reduzierung der Punktewolke

randomTriangulation(cloud, downsample):

Funktion die zufällige Dreiecke auf der Punktewolke erzeugt, welche nicht alle auf einer Ebene liegen sollen. Zurückgegeben werden die Dreiecke und deren Normalenvektoren. Zusätzlich wird der Index der Punkte in der Punktewolke zurückgegeben. Dieser wird für die Visualisierung benötigt.
cloud: Die Punktewolke
downsample: Faktor um den die Punktewolke reduziert wurde

randomTriangleNormals(cloud, p1, p2, pAbove1, triangles, triangle_index, normal):

Funktion speichert die Punkte eines Dreiecks in einer Liste, berechnet den Normalenvektor und legt die Indizes der Dreieckspunkte ab. Zurückgegeben wird somit die Dreiecksliste, eine Liste mit den Indizes der Punkte und eine Liste mit dem Normalenvektor.
cloud: Die Punktewolke
p1: Erster Punkt
p2: Zweiter Punkt
pAbove1: Dritter Punkt
triangles: Liste mit Dreiecken
triangle_index: Inizes der Dreiecke in einer Liste
normal: Liste mit den Normalenvektoren

nextLineOut(area, nextX, size):

Funktion sucht den Index der nächsten Zeile, auf der ein möglicher Nachbar liegt. Es werden die Punkte der nächsten Zeile in einer Liste und der Index der darauffolgenden Zeile zurückgegeben.
area: Alle Punkte der Punktewolke auf einer Ebene z
nextX: Index der nächsten Zeile, die betrachtet wird
size: Länge der area

nearest_above(p1, p2, nextArea):

Funktion berechnet den dritten Punkt eines Dreiecks, welcher auf der übergeordneten Ebene z liegt. Die Funktion gibt diesen Punkt zurück.
p1: Erster Punkt
p2: Zweiter Punkt
nextArea: Liste aller Punkte auf nächster Ebene z

plot_edges(points, lines, normal):

Funktion zur Visualisierung der Punktewolke und des erzeugten uniquen Dreiecks. Zusätzlich können in einem zusätzlichen Plot die Normalenvektoren visualisiert werden.
points: zu visualisierenden Punkte

lines: Indizes der Punkte zwischen denen eine Linie gezogen werden soll (Punkte der Dreiecke)
normal: Liste der Normalenvektoren

mittelpunkt(cloud):

Funktion berechnet den Mittelpunkt einer Liste von gegebenen dreidimensionalen Punkten.
Der Mittelpunkt wird zurückgegeben.
cloud: Die Punktewolke

update_normals(cloud, normals, triangles):

Funktion richtet die Normalenvektoren der erzeugten kleinen Dreiecke so aus, dass sie aus dem Bauteil heraus zeigen. Die aktualisierten Normalenvektoren werden zurückgegeben.
cloud: Die Punktewolke
normals: Die Normalenvektoren
triangles: Die Dreiecksdaten

norm(vektor):

Funktion welche einen Vektor normalisiert. Der Vektor wird als Liste oder als numpy Array übergeben. Es wird der normalisierte Vektor als numpy Array zurückgegeben.
vektor: zu normalisierende Vektor

randomTrianglePoints(triangles, normals, numberTriangles):

Die Funktion bestimmt zufällig drei Dreiecke in der STL Datei. Zusätzlich wird noch ein zufälliger Punkt auf den Dreiecken bestimmt. Zurückgegeben werden die zufälligen Punkte auf den Dreiecken, die Normalenvektoren der Dreiecke und die zufälligen Koordinaten der Dreiecke.
triangles: Dreiecke der STL-Datei
normals: Normalenvektoren der STL-Datei
numberTriangles: Anzahl der Dreiecke, welche aus der STL-Datei verwendet werden sollen

rankingSolutions(wolke, stlPoints, randomTrPoints, possibleSolution, diffMatrix):

Die Funkiton filtert aus allen möglichen Lösungen die beste heraus. Hierzu wird die Differenz zwischen den transformierten Punkten der STL-Datei und der Punktewolke berechnet. Der Lösungskandidat mit der geringsten Differenz wird hierbei präfereiert. Die Funktion gibt die finalen Punkte zurück. Die Punkte in der STL-Datei, mit denen die Rotation und Translation berechnet werden kann.
wolke: Die Punktewolke
stlPoints: STL-Datei Koordinaten
randomTrPoints: Punkte des uniquen Dreiecks in der Punktewolke
possibleSolution: Liste aller möglichen Lösungskandidaten
diffMatrix: Differenz zwischen jedem Lösungskandidaten und den uniquen Daten aus der Punktewolke

rotationMatching(originalPoints, rotatedPoints, visualize):

Diese Funktion berechnet die Rotation und Translation zwischen zwei Punktemengen. Es werden hier zwei Mengen vorausgesetzt, welche aus jeweils 3 Punkten bestehen. Zurückgegeben wird die Rotationsmatrix und der Translationsvektor.
originalPoints: Die Punkte, welche rotiert werden
rotatedPoints: Die Punkte, welche die Zielposition darstellen
visualize: eine boolesche Variable, welche die Transformation visualisiert, wenn visualize auf True gesetzt wird

Klasse Directory (Modul Directory)

Directory(path):

Der Konstruktor der Klasse Directory erzeugt ein Directory Objekt. Die Klasse ermöglicht es Dateinamen in einem Ordner in eine Liste abzuspeichern.
path: Die Pfadangabe in das gewünschte Verzeichnis

data_name_list():

Funktion erhält eine Pfadangabe und gibt alle Dateinamen von oben nach unten und die Verzeichnisangabe zurück.

Klasse TimeMeasurement (Modul TimeMeasure)

TimeMeasurement():

Konstruktor erzeugt ein Objekt der Klasse TimeMeasurement, welches den Zugriff auf Funktionen ermöglicht, mit denen die Ausführungszeit von Codeabschnitten gemessen wird. Konstruktor öffnet eine Datei Namens „TimeLogger.txt“, in welche die Zeiten geschrieben werden.

start():

Starte die Zeiterfassung.

end(description):

Ende der Zeiterfassung.
description: String der den Codeabschnitt beschreibt (für Anzeige in TimeLogger.txt)

closeData():

Schließt die TimeLogger Datei.