Graz University of Technology

Institut für Geodäsie

DI
Barbara **Süsser-Rechberger**
Steyrergasse 30/III
A–8010 Graz

Tel. +43(0)316 873-6359

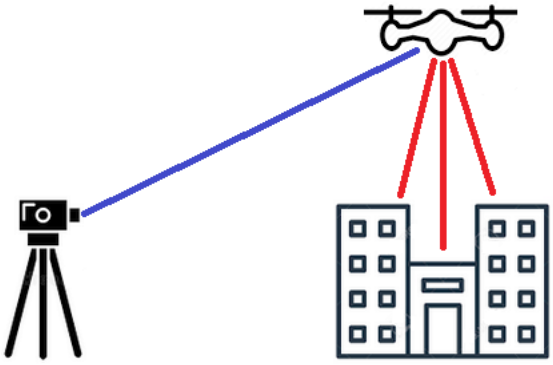
barbara.suesser-rechberger@tugraz.at

www.ifg.tugraz.at

30. März 2023

Übungsprojekt 01: Drohnengetragenes Laserscanning

Zur Erstellung eines digitalen Oberflächenmodells des Gebäudes Steyrergasse 30 wurde ein nach unten gerichteter Laserscanner an eine Drohne montiert. Die Drohne flog eine vorgegebene Trajektorie über dem Projektgebiet ab und nahm dabei Punkte unter sich auf. Um die genaue Trajektorie der Drohne zu bestimmen, wurde diese von einem Tachymeter vollautomatisch getrackt.



Implementierung

Die Tachymeterbeobachtungen sind in der Datei `obsTachy.txt` enthalten. Das Dateiformat enthält folgende Spalten: *Zeit in Sekunden, Distanz in Meter, Zenitwinkel in Radiant, Azimutwinkel in Radiant*. Die Zeit bezieht sich dabei auf den Referenzzeitpunkt 2018-03-13 15:10:00. Die Beobachtungen sind zeilenweise gespeichert.

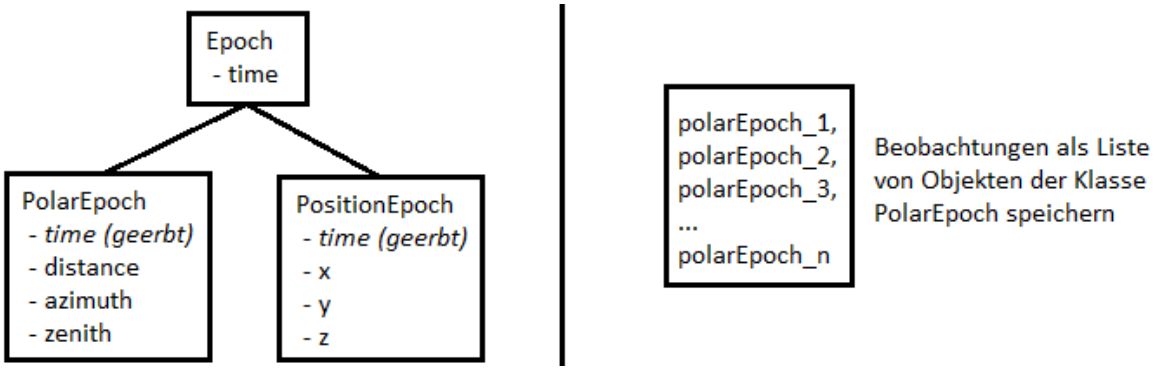
Die Beobachtungen des Laserscanners befinden sich in der Datei `obsDrone.txt`. Das Dateiformat ist gleich wie bei den Tachymeterbeobachtungen. (Hinweis: es handelt sich hier um ein mathematisches Koo-System)

Daten einlesen

Implementieren Sie zur Speicherung/Verarbeitung der Beobachtungsdaten eine Klasse `Epoch` mit einem privaten Attribut `time`. Das Attribut soll als Property implementiert werden, so dass der Zugriff von außen kontrolliert erfolgt. Nutzen Sie dazu entweder Decorators (`@property` und `@time.setter`) oder private Get- und Set-Funktionen zusammen mit der Funktion `property()`. In der Set-Funktion soll überprüft werden, ob der übergebene Wert dem Datentyp `datetime` des Python-Moduls `datetime` entspricht. Dazu kann die Python-Funktion `isinstance()` genutzt werden. Ist dies nicht der Fall, soll eine verständliche Fehlermeldung ausgegeben werden. Beim Anlegen eines Objekts dieser Klasse muss ein Zeitpunkt übergeben werden.

Implementieren Sie zwei Unterklassen `PositionEpoch` und `PolarEpoch`, die von der Klasse `Epoch` erben und somit die Zeitfunktionalität übernehmen. Die Klasse `PositionEpoch` enthält zusätzlich drei Attribute für die 3D-Koordinaten `x, y, z`. Für die Klasse `PolarEpoch` kommen die Attribute `distance, zenith, azimuth` hinzu. Überlegen Sie, ob und nach welchen Kriterien die übergebenen Werte in beiden Klassen überprüft werden können. Etwaige Überprüfungen sollen ebenfalls in Form von Properties erfolgen. Beim Anlegen von Objekten der jeweiligen Klassen sollen neben dem (notwendigen) Zeitpunkt optional Werte für die Attribute übergeben werden können.

Die Beobachtungen des Tachymeters und des Laserscanners sollen jeweils als Liste von Objekten der Klasse `PolarEpoch` eingelesen werden.



Daten ordnen

Aufgrund eines Softwarefehlers im Tachymeter sind die Daten nicht zeitlich sortiert, sondern in zufälliger Reihenfolge ausgegeben worden. Implementieren Sie eine Funktion `bubbleSort`, welche eine Liste von beliebigen Objekten „in place“(direkt die übergebene Liste, keine Kopie als Rückgabe) mittels Bubble-Sort-Algorithmus (siehe VO) sortiert. Implementieren Sie in der Klasse `Epoch` einen entsprechenden Operator, so dass eine Liste von Epoch-Objekten zeitlich aufsteigend sortiert werden kann.

Implementieren Sie eine einfache Klasse `Timer`, die zusammen mit dem `with`-Keyword zur Zeitmessung verwendet werden kann. Der aktuelle Zeitpunkt kann z.B. über die Funktion

`time.process_time()` aus dem Modul `timeit` ermittelt werden.

```
with Timer():
    pass # do something
# when finished, required time is automatically printed
```

Listing 1: Code-Stub Timer Klasse

Sortieren Sie die Beobachtungen des Tachymeters mit der implementierten BubbleSort-Funktion und messen Sie die dafür benötigte Zeit mit der selbst implementierten Timer-Klasse. Zur Veranschaulichung soll eine Kopie der Liste mit unsortierten Tachymeterbeobachtungen mit der Python-internen Sortierfunktion (`someList.sort()`) sortiert werden. Die dafür benötigte Zeit soll ebenfalls via Timer gemessen und mit der von BubbleSort verglichen werden.

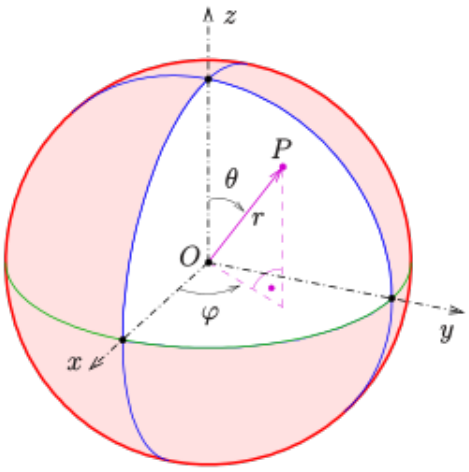
Drohnentrajektorie berechnen

Berechnen Sie die Trajektorie der Drohne basierend auf den Tachymeterbeobachtungen. Die berechneten Koordinaten sollen zeitlich sortiert als Liste von Objekten der Klasse `PositionEpoch` gespeichert werden. Die Formeln zur Berechnung von 3D-Koordinaten aus Tachymeterbeobachtungen (Distanz r, Zenitwinkel θ, Azimutwinkel ϕ) lauten:

$$x = r \cdot sin(\theta) \cdot cos(\phi)$$

$$y = r \cdot sin(\theta) \cdot sin\phi)$$

$$z = r \cdot cos(\theta)$$



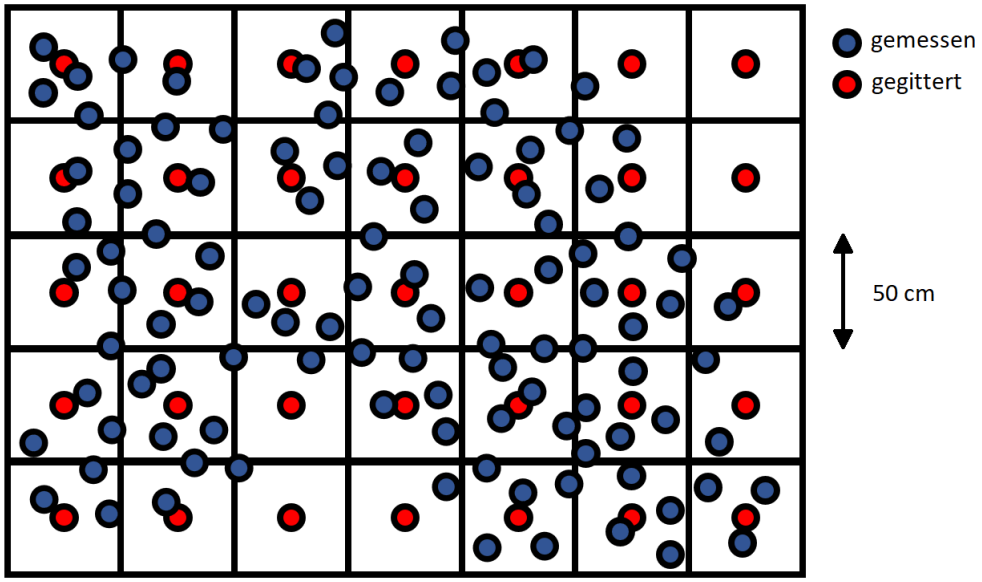
Die Koordinaten des Tachymeters sind zeitlich konstant und bekannt: `x = -51.280 m, y = -4.373 m, z = 1.340 m`. Etwaige Exzentrizitäten sind bereits an die Beobachtungen angebracht und müssen nicht berücksichtigt werden.

Bodenpunkte berechnen und gittern

Berechnen Sie die mittels Laserscanner gemessenen Bodenpunkte ausgehend von den Positionen der Drohne zur jeweiligen Epoche (Messzeitpunkt). Da es sich um die gleiche Art von Beobachtungen wie beim Tachymeter handelt, können erneut die oben angeführten Formeln verwendet werden. Die Anzahl der gemessenen Bodenpunkte unterscheidet sich je nach Epoche. Es können auch Epochen ohne Laserscannerbeobachtungen vorkommen. Daher ist es notwendig, jeder Epoche (Position der Drohne zu einem bestimmten Zeitpunkt) die entsprechenden Laserscannerbeobachtungen zuzuordnen.
Die absoluten Koordinaten eines Bodenpunkts berechnen sich zusammengefasst wie folgt:

$$\vec{r}_{Bodenpunkt} = \vec{r}_{Tachymeter} + \Delta\vec{r}_{Tachymetermessung} + \Delta\vec{r}_{Laserscannermessung}$$

Die berechneten Bodenpunkte sollen in ein regelmäßiges Raster mit einer Auflösung von 50 cm in X- und Y-Richtung gegittert werden. Kommen in einer Gitterzelle mehrere Bodenpunkte vor, so ist der Mittelwert der Z-Werte aller Punkte zu berechnen und der Gitterzelle zuzuordnen. Für einen effizienten Algorithmus zum Gittern der Bodenpunkte ist es empfehlenswert, *numpy*-Arrays und Funktionen wie `numpy.where()` zu verwenden.



Visualisierung

Die Ergebnisse sollen auf ansprechende Weise in verschiedenen Varianten mittels `matplotlib` grafisch dargestellt werden. Interessant sind unter anderem die Trajektorie der Drohne (z.B. Horizontal- und Vertikalkomponenten), das Oberflächenmodell (z.B. 2D mit farbkodierter Höheninformation, 3D, original vs. gegittert, Aufnahmezeitpunkt farbcodiert, etc.) oder auch die Beobachtungsgrößen. Kombinierte Darstellungen (Oberfläche, Trajektorie, ...) sind ebenfalls wünschenswert.

Abgabe

Die Abgabe besteht aus zwei Teilen, dem **Quellcode** und dem **technischen Bericht**. Der Quellcode (aber nicht die gegebenen Beobachtungsdateien) und der technische Bericht (als `pro01DroneLaserScanning.pdf`) müssen in einem Archiv `pro01DroneLaserScanning.zip` bis 05. Mai 2023, 23:59 Uhr im TeachCenter abgegeben werden. Die Abgabegespräche finden am 17.05.2023 statt (einzeln). Die Anmeldung zu den Abgabegesprächen erfolgt im TeachCenter.

Hinweis: Quellcode

Die Formatierung und Präsentation des Quellcodes soll dem Verständnis förderlich sein, achten Sie deshalb auf folgende Gesichtspunkte:

- Dateikopf mit Namen der Ersteller, der Gruppe und des Projekts in jeder Datei.
- Verwendung aussagekräftiger Variablennamen, z. B. `tachymeterEpoch` anstatt `te`.
- Erklärende Kommentare. Wieso ist dieser Quellcode so aufgebaut wie er ist? Welche Formel wird hier implementiert? Welches Ergebnis steht am Ende des Quellcodeabschnittes?

Weiters soll der Quellcode auch inhaltlich einer erkennbaren Struktur folgen:

- Sinnvoller Einsatz von Funktionen z. B. zur Vermeidung von Wiederholungen. Klar definierte Schnittstellen auch und speziell für Klassen (Mit Beschreibung von Parametern und zu erwartenden Rückgabewerten).
- Strukturierte Abfolge der Anweisungen, z. B. zuerst Abarbeitung eines Aufgabenteils gefolgt von Anweisungen zum nächsten Aufgabenteil.
- Variablen so lokal wie möglich deklarieren, im Idealfall erst bei der ersten Verwendung.

Hinweis: Technischer Bericht-Inhalt

Der Inhalt des technischen Berichts soll zum einen die Problemstellung klar wiedergeben, als auch die während der Bearbeitung durchgeführten Aufgaben erläutern und illustrieren. Der Bericht ist in verschiedene Abschnitte unterteilt, die eine klare Struktur verleihen und das Verständnis fördern sollen.

1. Der Bericht beginnt mit der Aufgabenstellung, formuliert in eigenen Worten. Diese umfasst eine kurze Beschreibung des zu lösenden Problems (Was ist zu tun?), den Mitteln, die bei der Implementierung eingesetzt werden sollen (Wie ist das Problem zu lösen?), sowie einem Ausblick auf die zu erwartenden Ergebnisse (Was soll das Ergebnis des Projekts sein?).
2. Es folgt ein Kapitel, welches den Programmablauf beschreibt. Es soll einen allgemeinen Einblick in den inneren Aufbau des Programms verschaffen. Der Programmablauf sowie die zum Ergebnis führende Logik sollen auf abstrakter Ebene (ohne Implementierungsdetails wie Variablennamen oder Formeln) erläutert werden. Dies geschieht am

besten in einem Fließtext mit unterstützenden Hilfsmitteln wie z.B. Diagrammen oder Pseudocode-Ausschnitten.

3. Das nun folgende Kapitel zur Implementierung vertieft die wichtigsten Punkte der zuvor beschriebenen Struktur. Es soll darauf eingegangen werden, wie die beschriebenen Algorithmen implementiert sind. Hierbei dürfen auch die tatsächlich verwendeten Formeln nicht fehlen. Kurze Quellcode-Ausschnitte (wenige Zeilen) können besonders interessante Code-Passagen illustrieren. Hinweise auf aufgetretene Probleme und deren Lösungen finden hier ebenfalls Ihren Platz.

4. Das letzte Kapitel des Berichts sind die Ergebnisse. Es soll demonstriert werden, dass das Programm die gestellten Anforderungen erfüllt. Dies beinhaltet sinnvolle Tests der Funktionalität und eine Diskussion der Ergebnisse. Anschauliche grafische Darstellungen der Ergebnisse sind hierbei besonders wichtig. Dieses Kapitel schließt mit einem Absatz, der das Projekt zusammenfasst.