

HOCHSCHULE MÜNCHEN
FAKULTÄT FÜR INFORMATIK UND MATHEMATIK

Praktikumsaufgabe 1

in der Vorlesung

Computational Geometry

Bestimmung von Schnittpunkten aus einem
gegebenen Satz von Strecken

Team:	Christopher Hinz, Tobias Gruber
Studiengruppe:	Master Informatik
Studiensemester:	1. Semester
Schwerpunkt:	Embedded Computing

11.04.2022

Sommersemester 2022

1 Einführung

Im Rahmen der Vorlesung Computational Geometry lautet die Aufgabe für das erste Praktikum, die Anzahl der Schnittpunkte aus einem Satz von Strecken zu ermitteln. Dafür werden 3 Dateien zur Verfügung gestellt, die jeweils Datensätze zu 1001, 10.001 und 100.001 Strecken enthalten.

Die Dateien enthalten jeweils 4 Koordinaten pro Zeile, welche jeweils die x- und y-Koordinaten eines Start- bzw. Endpunkts einer Strecke darstellen. Die Dateien sollen jeweils eingelesen und die Anzahl der sich schneidenden Strecken durch paarweises Testen ermittelt werden. Zusätzlich wird die aufgewendete Zeit zur Ermittlung der Anzahl der Schnittpunkte für jede Datei gemessen.

Im folgenden Abschnitt wird darauf eingegangen, wie die Grundstruktur der Problemlösung aufgebaut ist und wie die mathematische Strategie zur Lösung des Problems in C++ formuliert werden kann. Anschließend werden die Ergebnisse der Datenauswertung vorgestellt. Es ist zu begründen, warum die Anzahl der vom Programm jeweils gefundenen Schnittpunkte korrekt ist.

2 Abschätzung der zu erwartenden Größenordnung des Ergebnisses

Um einschätzen zu können, inwiefern die Ergebnisse sinnvoll sind, sollen an dieser Stelle einige Vorüberlegungen getroffen werden.

Welche Anzahl an Schnitten sind zu erwarten bzw. in welchem Wertebereich sollten sich die Ergebnisse bewegen?

Die minimal zu erwartende Anzahl an Schnitten wäre für 1.000, 10.000 und 100.000 Strecken jeweils null. Dies ist somit die untere Grenze des zu erwartenden Wertebereichs. Die maximale Anzahl an Schnitten würden in dem Fall auftreten, dass sich alle Strecken gegenseitig schneiden. Dann wäre nämlich für jede Strecke ein Schnittpunkt mit allen anderen Strecken gegeben. Das Problem ist in der Mathematik auch als Sektklasproblem bekannt. Die Berechnung erfolgt in Anlehnung an die Formel für die Dreieckszahlen. Für eine Anzahl von n Strecken ergeben sich somit k Schnittpunkte wie folgt:

$$k = \frac{(n-1) \cdot n}{2} \quad (1)$$

Damit wären für 1.000 Strecken maximal 499.500 (für 10.000 maximal 49.995.000, für 100.000 maximal 4.999.950.000) zu erwarten. Dies ist somit die obere Grenze des zu erwartenden Wertebereichs.

3 Ergebnisse der Schnittstellenberechnung und der Zeitmessung

Das Programm verwendet die Funktion `read_dat()` um die Daten aus den Dateien einzulesen und über `pack_koords()` die Datensätze zu Strecken zusammenzufassen. Jede Strecke wird anschließend gegen alle anderen Strecken geprüft, ob sie diese schneidet, oder nicht. Zur Feststellung, ob sich die zwei Strecken schneiden wird ausgewertet, ob die Punkte p_1, p_2 der ersten Strecke zu den Punkten p_1, p_2 der zweiten Strecke im oder gegen den Uhrzeigersinn angeordnet sind. Dafür wird folgende Formel verwendet, die im Ergebnis den Betrag des Kreuzproduktes aus den Richtungsvektoren der drei Punkte enthält:

$$ccw(p, q, r) := \begin{vmatrix} p_1 & p_2 & 1 \\ q_1 & q_2 & 1 \\ r_1 & r_2 & 1 \end{vmatrix} = (p_1q_2 - p_2q_1) + (q_1r_2 - q_2r_1) + (p_2r_1 - p_1r_2) \quad (2)$$

Ist das Produkt der `ccw`-Funktion der ersten Strecke mit den Endpunkten der zweiten Strecke, sowie das Produkt der `ccw`-Funktion der zweiten Strecke mit den Endpunkten der ersten Strecke kleiner oder gleich 0, so kann unter der Annahme, dass die Strecken nicht kollinear zueinander sind davon ausgegangen werden, dass sich die Strecken schneiden.

Sind die Strecken kollinear angeordnet, so ergeben beide Kreuzprodukte 0. In diesem Fall muss noch überprüft werden, ob sich die Strecken überlappen. Bei Überlappung der Strecken lässt sich mindestens einer der Endpunkte der zweiten Strecke über ein Vielfaches zwischen 0 und 1 des ersten Streckenvektors addiert mit einem Endpunkt der ersten Strecke darstellen. Da zuvor bereits festgestellt wurde, dass die Strecken kollinear sind, kann der Skalar zum Richtungsvektor der zweiten Strecke über das Auflösen der parametrischen Darstellung durch die x-Komponenten der Ortsvektoren aufgelöst werden.

$$\lambda = \frac{x_r - x_p}{x_q - x_p} \quad (3)$$

Liegt der Wert von λ zwischen 0 und 1, so überlappen sich die Funktionen. Nimmt λ den Wert 0 oder 1 an, schneiden sich die Strecken in jeweils einem der Endpunkte. In Listing 1 wird die Ausgabe des Programmes zur Schnittpunktezählung aufgeführt. In der Ausgabe ist zu erkennen, dass die Anzahl der Schnitte für alle Fälle (1.000, 10.000, 100.000) im zu erwartenden Wertebereich liegt. Schnittpunkte:

- 1.000 Strecken: $0 \leq 11 \leq 499.500$
- 10.000 Strecken: $0 \leq 732 \leq 49.995.000$
- 100.000 Strecken: $0 \leq 77.126 \leq 4.999.950.000$

```
# s_1000_1.dat
Strecken insgesamt: 1001
```

```

Schnitte zweier Strecken: 11
Runtime: 40.965 ms
#s_10000_1.dat
Strecken insgesamt: 10001
Schnitte zweier Strecken: 732
Runtime: 3984.3 ms
#s_100000_1.dat
Strecken insgesamt: 100001
Schnitte zweier Strecken: 77126
Runtime: 394721 ms

```

Listing 1: strecken.cpp: Ausgabe Konsole

4 Verifikation und Validierung

Um die korrekte Funktionsweise des Programmes zu verifizieren werden verschiedene Testfälle definiert. Die Testfälle decken unterschiedliche Streckenanordnungen ab und sollen dadurch die in der Vorlesung besprochenen Grenzfälle für Streckenschnitte überprüfen. Abbildung 1 zeigt die implementierten Testfälle.

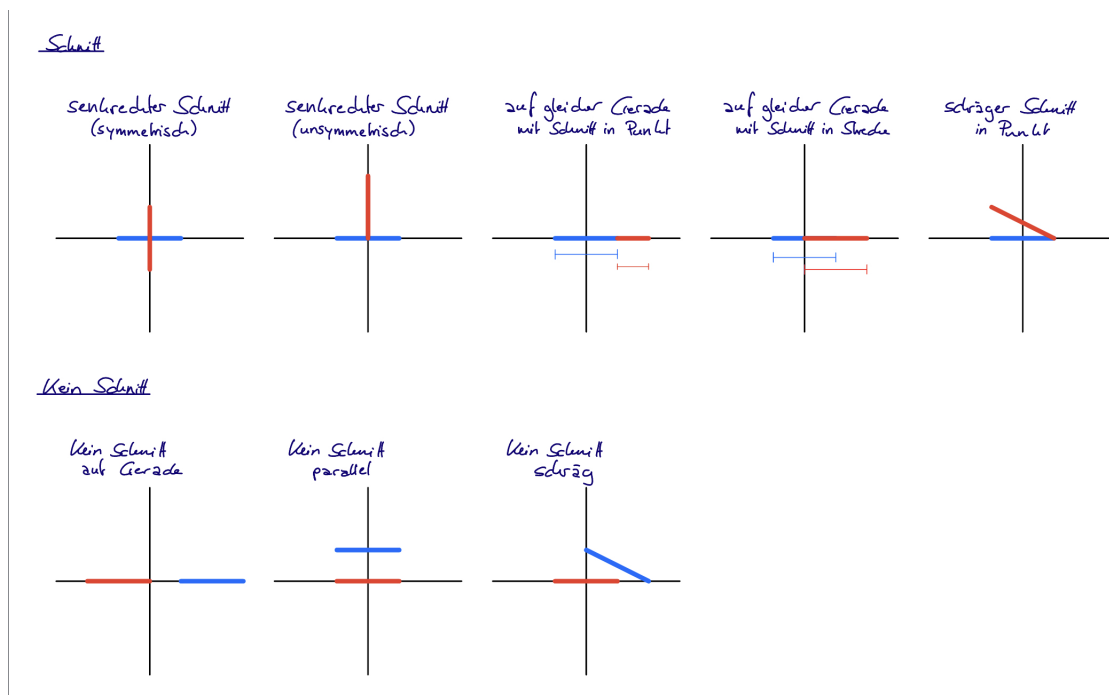


Abbildung 1: Testfälle für Streckenschnitte

Listing 2 zeigt, dass für die Testfälle die erwarteten Ergebnisse erzielt werden. Das bedeutet, dass für alle Streckenpaare, die wie in den Testfällen angeordnet sind, die

Schnittstellenfunktion korrekt berechnet ob sie sich schneiden oder nicht.

```
Soll: true , Ist: true
Soll: true , Ist: true
Soll: true , Ist: true
Soll: true , Ist: true
Soll: true , Ist: true
Soll: false , Ist: false
Soll: false , Ist: false
Soll: false , Ist: false
```

Listing 2: testing.cpp: Ausgabe Konsole

Des weiteren wurde ein Python-Programm geschrieben. Dieses plottet die Strecken mit Schnittpunkten und stellt damit eine weitere Möglichkeit dar die Ergebnisse zu überprüfen. Dies Graphen werden mittels matplotlib erstellt und sehen wie nachfolgend dargestellt aus. Auf Implementierungsdetails soll an dieser Stelle verzichtet werden, da dieses Programm ausschließlich zur Überprüfung dient. Stichprobenartige Überprüfungen der Schnittpunkte mit diesem Programm sind alle positiv ausgefallen.

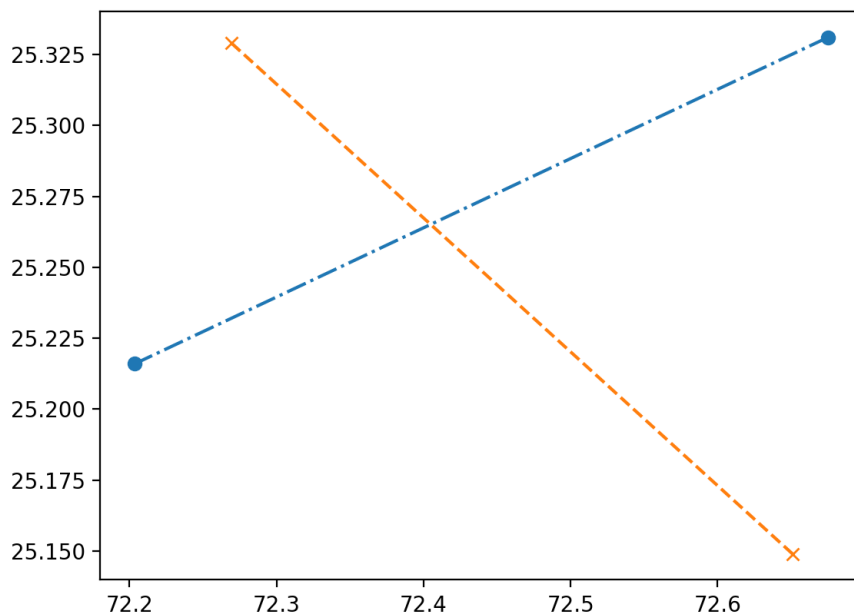


Abbildung 2: Python-Plot mit matplotlib

Zur Validierung der Ergebnisse wurde Rücksprache mit anderen Gruppen gehalten. Diese haben die gleichen Ergebnisse für die Anzahl der Schnittpunkte erhalten. Zusätzlich zu der durchgeführten Verifikation ist dies ein weiteres Indiz, dass die Ergebnisse mit hoher Wahrscheinlichkeit korrekt ausfallen.