Heinrich Heine
Universität
Düsseldorf

INSTITUT FÜR INFORMATIK
Softwaretechnik und
Programmiersprachen
Universitätsstr. 1 D–40225 Düsseldorf

2D-Sichtbarkeit anhand von Gebäudekonturen aus Openstreetmaps

Bachelorarbeit

vorgelegt von

Oliver Kleisch

02. Juni 2025

im Studiengang Informatik
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

Erstgutachter: Dr. Konrad Völkel
Zweitgutachter: Dr. Peter Arndt

Selbstständigkeitserklärung

Hiermit versichere ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, den 02. Juni 2025



Oliver Kleisch

Zusammenfassung

Diese Bachelorarbeit beschäftigt sich mit der Entwicklung eines Verfahrens zur Sichtfeldberechnung um Gebäude. Der Fokus liegt in der Identifikation von Konsumverbotszonen im Rahmen der Teillegalisierung von Cannabis in Deutschland gemäß dem Konsumcannabisgesetz. Bestehende Ansätze bieten meist nur Vereinfachungen realer Umstände. Ziel ist die Entwicklung eines Verfahrens zur realitätsnäheren Berechnung von Sichtbarkeitszonen.

Zur Umsetzung werden Gebäudekonturen aus OpenStreepMap extrahiert und umliegende Gebäude als Sichthindernisse verwendet. Mittels etablierter Algorithmen zur Sichtbarkeitsberechnung im zweidimensionalen Raum werden approximativ die Sichtfelder berechnet. Die Ergebnisse werden grafisch visualisiert und über eine interaktive Karte zugänglich gemacht.

Die Evaluation zeigt, dass das Verfahren im Vergleich zu einem bestehenden Ansatz eine signifikante Flächenreduktion der Sichtbarkeitsbereiche erzielt. Im Fallbeispiel Düsseldorf beträgt die Abnahme bis zu 30%.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Hintergrund und Motivation 1

1.2 Problemstellung 1

1.3 Zielsetzung 2

2 Stand der Forschung **3**

2.1 Sichtbarkeitsanalysen (Theorie/Praxis) 3

2.2 Überblick existierender Algorithmen 4

2.2.1 Algorithmus von Joe und Simpson 4

2.2.2 Algorithmus von Asano 5

2.2.3 Triangular Expansion 5

2.3 Kritische Betrachtung bisheriger Ansätze 7

3 Methodik **8**

3.1 Grundidee 8

3.2 Technische Architektur 8

3.3 Werkzeugauswahl 9

3.4 Datenextraktion/-aufbereitung aus OSM 9

4 Implementierung **11**

4.1 Codestruktur 11

4.2 Algorithmusentwicklung 12

5 Visualisierung und Evaluation **15**

5.1 Performanceanalyse 15

5.2 Fehleranalyse 19
5.3 Vergleich mit bestehenden Ansätzen 21
5.4 Darstellung auf interaktiver Karte 25

6 Schluss **27**

6.1 Diskussion 27
6.2 Fazit 28
6.3 Ausblick auf weiterführende Forschung 28

7 Source Code Listings **29**

Literatur **30**

1 Einleitung

1.1 Hintergrund und Motivation

Seit dem 01.04.2024 ist der Besitz von Cannabis in Deutschland teilweise legalisiert. Infolge dessen existieren Regelungen, die den Konsum in einer Sichtweite von 100 Metern um bestimmte Einrichtungen untersagen. Unter diesen Einrichtungen zählen unter anderem Schulen, Kinderspielplätze, Sportstätten und Anbauvereinigungen. Die Grundidee basiert auf der Bubatzkarte [bub24]. Diese zeichnet um all jene Gebäude einen Umkreis von 100 Metern, um zu visualisieren, in welchen Zonen das Verbot zutreffen könnte.

1.2 Problemstellung

Das Konsumcannabisgesetz (KCanG) besagt hierbei bezüglich des Konsums wörtlich:

“Der öffentliche Konsum von Cannabis ist verboten:

1. in Schulen und in deren Sichtweite,
2. auf Kinderspielplätzen und in deren Sichtweite,
3. in Kinder- und Jugendeinrichtungen und in deren Sichtweite,
4. in öffentlich zugänglichen Sportstätten und in deren Sichtweite,
5. in Fußgängerzonen zwischen 7 und 20 Uhr und
6. innerhalb des befriedeten Besitztums von Anbauvereinigungen und in deren Sichtweite.

Im Sinne von Satz 1 ist eine Sichtweite bei einem Abstand von mehr als 100 Metern von dem Eingangsbereich der in Satz 1 Nummer 1 bis 4 und 6 genannten Einrichtungen nicht mehr gegeben. ”(§5 Absatz 2 KCanG [Deu24])

Die Herausforderung besteht zunächst darin, die hier genannten Einrichtungen, welche im Folgenden als Verbotsgebäude bezeichnet werden, erstmals zu erkennen. Des Weiteren muss zusätzlich ein Sichtbereich von 100 Metern um die besagten Verbotsgebäude, nachfolgend als Verbotszonen bezeichnet, identifiziert werden. Ohne Hilfsmittel ist eine akkurate Erfassung aller Verbotsgebäude und deren Verbotszonen in der Praxis kaum umsetzbar. Bestehende Ansätze bieten außerdem nur eine grobe Berechnung der Sichtbereiche. Dies führt zu einer exzessiven Überschätzung der Verbotszonen, die oftmals nicht die realen Umstände darstellen.

1.3 Zielsetzung

Diese Arbeit verfolgt das Ziel, für jedes Verbotsgebäude, spezifiziert nach §5 Absatz 2 Satz 1 Nummer 1 bis 4 des KCanG, approximativ das Sichtfeld zu berechnen und zu visualisieren. Der berechnete Sichtbereich soll möglichst präzise sein und im besten Fall eine Überschätzung der Verbotszonen sein, anstatt einer Unterschätzung, um rechtliche Konflikte zu vermeiden. Ähnlich wie bei der Bubatzkarte sollen die berechneten Verbotszonen letztendlich auf einer Karte dargestellt werden und über eine derzeit lokal verfügbare Website aufrufbar sein. Auf dieser Karte soll es auch möglich sein, bestimmte Orte suchen zu können, um die gewünschten Zonen schneller ausfindig zu machen.

2 Stand der Forschung

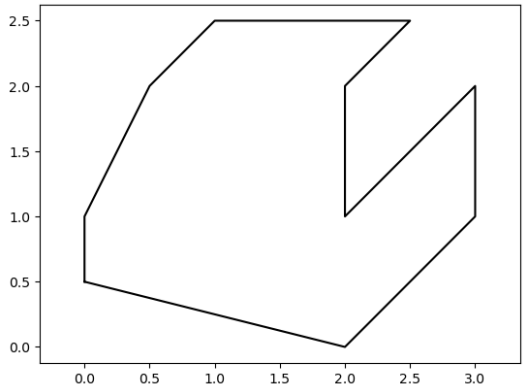
Die Sichtbarkeitsberechnungen in zweidimensionalen Räumen sind ein überaus erforschtes Gebiet und spielen in Bereichen wie der Geoinformatik, Robotik, Stadtplanung und auch Computerspielen eine zentrale Rolle. Hierbei besteht das Hauptziel darin, herauszufinden, welche Bereiche um einen bestimmten Punkt herum sichtbar sind oder auch um kürzeste Wege zwischen zwei Punkten ausfindig zu machen.

2.1 Sichtbarkeitsanalysen (Theorie/Praxis)

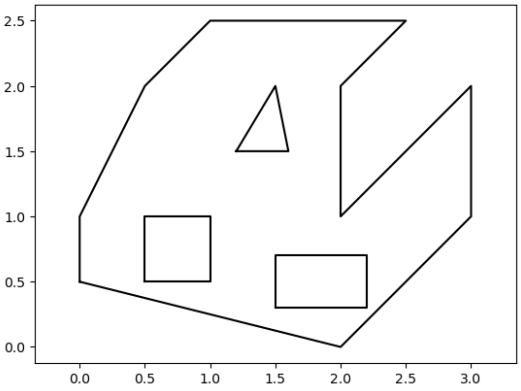
Ein verbreiteter Ansatz, um einen sichtbaren Bereich zu identifizieren, ist die Berechnung von Sichtbarkeitspolygonen (visibility polygons). Nach S.K. Ghosh [Gho07, S.2 ff.] ist ein Polygon eine geschlossene Region innerhalb einer Ebene. Diese Region besteht aus einer endlichen Anzahl von Liniensegmenten, auch Kanten genannt, wobei die Endpunkte jeder Kante auch Ecken genannt werden. Hierbei existiert ein Pfad zwischen jedem Punkt der Region, welcher keine Kante des Polygons schneidet. Außerdem bilden die Kanten des Polygons einen Zyklus, bei dem benachbarte Kanten einen gemeinsamen Eckpunkt besitzen. Diese Art eines Polygons wird auch einfaches Polygon genannt. Die Sichtbarkeitsberechnung für ein solches Polygon nimmt einen Punkt innerhalb des Polygons als Beobachter (observer) und berechnet die Teilfläche des Polygons, welche von dem Beobachter aus als "sichtbar" gilt.

Die Berechnung von Sichtbarkeitspolygonen wird oftmals erweitert, sodass Polygone mit Löchern zulässig sind. Polygone mit Löchern erweitern die Definition von einfachen Polygonen, um weitere geschlossene Räume innerhalb des Polygons, welche die Kanten des Polygons nicht schneiden. Diese geschlossenen Räume werden hierbei Löcher genannt. Diese Erweiterung ermöglicht eine komplexere Modellierung der Umgebung.

In anderen Anwendungsgebieten, wie beispielsweise der Computerspielindustrie, wird die Berechnung von Sichtbarkeitspolygonen genutzt, um unter anderem realistische Licht- und Schatteneffekte zu erzeugen. Hierbei wird der Beobachterpunkt als Punkt-



(a) Einfaches Polygon



(b) Polygon mit Löchern

lichtquelle aufgefasst und der sichtbare Bereich wird als erhellt dargestellt, wobei auf den nicht sichtbaren Bereich Schatten geworfen werden. Die Erweiterung, Polygone mit Löchern zuzulassen, ermöglicht in diesem Kontext eine Sichtbarkeitsberechnung mit Hindernissen, um realistische Gegebenheiten zu modellieren.

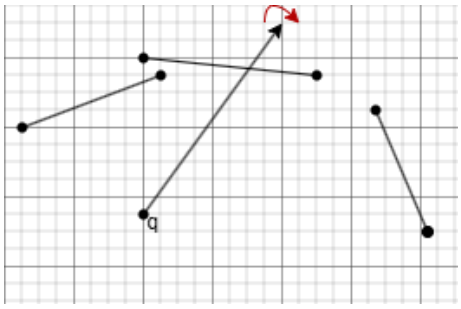
2.2 Überblick existierender Algorithmen

2.2.1 Algorithmus von Joe und Simpson

Der Algorithmus von Joe und Simpson [JS87] ist ein Verfahren zur Berechnung von Sichtbarkeitspolygonen innerhalb einfacher Polygone. Der Algorithmus ist für seine Laufzeitkomplexität in $\mathcal{O}(n)$ bekannt, wobei n die Anzahl der Eckpunkte des Polygons darstellt. Das Grundprinzip basiert auf einem sequentiellen Scan der Polygongrenze, also den Polygonkanten, und benutzt zur Verwaltung der sichtbaren Eckpunkte einen Stack. Während die Kanten sequentiell durchlaufen werden, wird jeweils überprüft, ob sie vom Beobachterpunkt q aus sichtbar sind. Hierbei gibt es drei mögliche Operationen, die durchgeführt werden können; neue sichtbare Punkte werden auf den Stack gelegt, nicht mehr sichtbare Punkte werden vom Stack heruntergenommen oder die nachfolgenden Kanten werden überlesen, falls diese in einem nicht sichtbaren Bereich liegen (dem “hidden-window”). Diese Methode garantiert für einfache Polygone in linearer Laufzeit, dass die sichtbaren Eckpunkte

des gesuchten Sichtbarkeitsbereichs auf dem Stack in richtiger Reihenfolge liegen.

2.2.2 Algorithmus von Asano



Der Algorithmus von Asano [Asa85] erweitert erstmals die Sichtbarkeitsberechnung von einfachen Polygonen auf Polygone mit Löchern. Der Algorithmus zeichnet sich außerdem durch eine Laufzeitkomplexität von $\mathcal{O}(n\ log\ n)$ aus. Die Idee basiert auf einer "Rotationssweep"-Technik, bei der ein Strahl L von dem Beobachterpunkt q ausgeht und um diesen rotiert. Zunächst werden alle Eckpunkte des Polygons nach ihrem Polarwinkel relativ zu q sortiert. Während der Strahl um q rotiert, wird eine balancierte Suchbaumstruktur geführt, welche die Reihenfolge der von L getroffenen Polygonkanten speichert. Jedes Mal, wenn sich die nächstliegende Kante entlang des Strahls ändert, wird ein neuer Punkt des Sichtbarkeitsbereichs bestimmt. Der Algorithmus endet mit einer vollständigen Rotation von 360° um q. Alle Punkte des Sichtbarkeitsbereichs bilden in Reihenfolge die Eckpunkte des Sichtbarkeitspolygons.

2.2.3 Triangular Expansion

Der Algorithmus der "Triangular Expansion" [BHH+14] oder auch der Algorithmus der Dreiecks-Erweiterung ist in der Praxis eine äußerst effiziente Methode zur Berechnung von Sichtbarkeitspolygonen. Theoretisch besitzt dieser eine worstcase Laufzeit von $\mathcal{O}(n^2)$, zeigt jedoch in der Praxis oft (sub-)lineares Verhalten, insbesondere bei der Eingabe von großen und komplexen Strukturen. Aufgrund dessen ist der Algorithmus der Triangular Expansion, auch der gewählte Algorithmus für unser Vorhaben. Der Algorithmus wurde erstmals im zitierten Preprint vorgestellt und unterscheidet sich konzeptionell stark von den bisherigen Algorithmen. Der Algorithmus basiert auf einer einmaligen Triangulierung des Eingabepolygons. Hierbei wird das Polygon in nicht überlappende Dreiecke unterteilt. Anschlie-

ßend wird der Sichtbarkeitsbereich von dem Beobachterpunkt q rekursiv ermittelt. Für die Triangulierung wird im Preprint eine "Constrained-Delaunay-Triangulation" verwendet, welche insbesondere die vorgegebenen Kanten mit in Betracht zieht und im schlimmsten Fall eine Laufzeit von $\mathcal{O}(n^2)$ aufweist. Im nächsten Schritt wird das Dreieck, das q enthält, ermittelt, welches durch eine iterative Suche aller Dreiecke in $\mathcal{O}(n)$ gefunden werden kann. Dieses Dreieck bildet den Ausgangspunkt für die Sichtbarkeitsberechnung und wird als vollständig sichtbar vermerkt. Der letzte Schritt ist die rekursive Ausbreitung des Sichtfeldes. Hierbei wird die Sichtgrenze über jedes Dreieck hinweg ausgedehnt. Für jede Kante des aktuellen Sichtbereiches wird nun einzeln ein rekursiver Aufruf gestartet. Ist die momentan betrachtete Kante eine Polygongrenze, also eine bestehende Kante des Eingabepolygons, so wird der bisherige Bereich zum Sichtfeld hinzugefügt. Andernfalls wird geprüft, ob und wie q in das Nachbardreieck "schauen" kann. Die Sichtkegelgrenzen (links und rechts) werden dabei durch die Geometrie der angrenzenden Kanten sukzessiv eingeschränkt. Die Rekursion läuft weiter, solange Dreiecke erreichbar sind, die innerhalb dieses Sichtkegels liegen. Die letztendlichen Sichtkegelgrenzen der Aufrufe, die eine Polygongrenze erreichen, und der Ausgangspunkt q bilden zusammen jeweils ein Sichtdreieck. Der Zusammenschluss von allen Sichtdreiecken bildet das Sichtbarkeitspolygon.

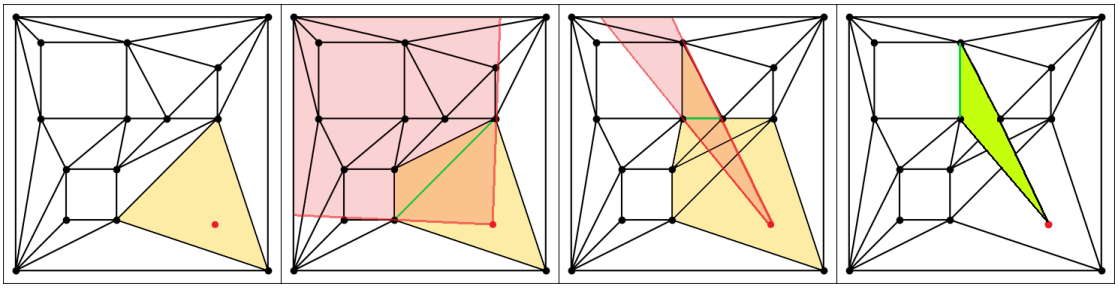


Abbildung 2: Sichtbereich eines rekursiven Durchlaufs

2.3 Kritische Betrachtung bisheriger Ansätze

Auch wenn die genannten Algorithmen für viele Anwendungen effizient sind, existieren dennoch zahlreiche Herausforderungen für die Berechnung von Sichtfeldern, wenn man Gebäudekonturen anstelle von einzelnen Punkten wählt. Zum einen muss man den gewählten Algorithmus so erweitern oder die Umstände so modifizieren, dass die Berechnungen überhaupt erst für ganze Gebäude möglich sind. Des Weiteren sollte das gewählte Verfahren hinreichend effizient sein, um auch größere Städte mit zahlreichen Gebäuden verarbeiten zu können. Außerdem stellt sich noch die Frage, ob und wie die Karte mit Datenupdates umgeht. Zum einen könnte man die Karte für Echtzeitberechnungen nutzen, sodass schnelle Algorithmen streng gefordert werden. Auf der anderen Seite werden Kartendaten auch ständig aktualisiert, falls beispielsweise neue Gebäude oder Strecken gebaut werden. Diese müssen dann gezielt gefiltert werden und die veralteten Daten ersetzen können, möglichst ohne große Performanzprobleme.

3 Methodik

Die Methodik zielt darauf ab, auf Basis von öffentlich zugänglichen Geodaten aus OpenStreetMap die algorithmische Sichtfeldberechnung umzusetzen. Hierfür werden aus zweidimensionalen Gebäudekonturen die Sichtfelder gewisser Gebäude berechnet, insbesondere in Bezug auf das bestehende Cannabisgesetz. Die Umsetzung soll dabei komplett in Python erfolgen, mit Hilfe von speziellen Bibliotheken für die Extraktion von Geodaten und für die benötigten geometrischen Berechnungen.

3.1 Grundidee

Die grundlegende Idee dieser Arbeit verfolgt das Ziel, eine Sichtbarkeitsanalyse in einem zweidimensionalen Raum durchzuführen, insbesondere im Bezug auf einen realen, städtischen Kontext. Hierbei wird ein Sichtbarkeitspolygon berechnet, welches das Sichtfeld von einem bestimmten Standpunkt darstellt. Umliegende Gebäude dienen hierbei als undurchsichtige Hindernisse, die das Sichtfeld des Beobachters einschränken. Die verschiedenen Beobachter setzen sich aus den Verbotsgebäuden zusammen, wie sie in §5 des KCanG definiert werden. Die Konturen jener Verbotsgebäude werden hierbei in einzelne Punkte unterteilt, von welchen das Sichtfeld separat berechnet wird. Die Vereinigung aller einzelnen Sichtfelder ergibt approximativ das Sichtfeld des jeweiligen Gebäudes. Das resultierende Sichtfeld kann schließlich auf einer Karte abgebildet werden.

3.2 Technische Architektur

Die konkrete Umsetzung gliedert sich hierbei in vier zentrale Abschnitte. Zunächst werden die notwendigen Gebäudedaten aus der OpenStreetMap-Datenbank erhoben und abgespeichert. Die Gebäudedaten der Verbotsgebäude werden mittels der Filterungskriterien Quellcode 1 ermittelt, wie sie auf der Seite der Bubatzkarte zu finden sind. Für jenes Gebäude werden zusätzlich alle umliegenden Gebäude ermit-

telt, welche dann für die spätere Verarbeitung notwendig sind. Im zweiten Schritt werden alle abgerufenen Daten gezielt gefiltert und aufbereitet, sodass diese für die Sichtfeldberechnung verwendbar sind. Anschließend können die Sichtfelder mit einem ausgewählten Algorithmus berechnet werden. Um große Daten im Allgemeinen leichter handhaben zu können, wird sich für den Algorithmus der "Triangular Expansion" entschieden, da dieser im Durchschnitt bei der Laufzeit eine gute Performanz mit sich bringt. Zuletzt werden die Ergebnisse evaluiert und grafisch dargestellt, wie etwa mittels einer interaktiven Karte. Dieses Vorgehen erleichtert es, die einzelnen Arbeitsschritte sauber voneinander zu trennen und bei Bedarf anzupassen, insbesondere im Hinblick auf die Optimierung des Algorithmus und die Ergebnisauswertung.

3.3 Werkzeugauswahl

- Die Bibliothek OSMnx [Boe25] dient zur Extraktion von Gebäudekonturen aus OpenStreetMap. Sie ermöglicht einen einfachen und effizienten Abruf aller relevanten Daten, zur weiteren Verarbeitung.
- Shapely [G+te] dient der geometrischen Verarbeitung der extrahierten Gebäudedaten. Diese Bibliothek ist außerdem die zugrundeliegende Bibliothek, der gespeicherten Gebäudekonturen innerhalb von OSMnx. Insbesondere können damit Schnittpunkte und Polygone der Sichtfelder berechnet werden.
- Für die Visualisierung kommen Folium und Matplotlib zum Einsatz. Beide Bibliotheken bieten eine robuste Auswahl, zur Erstellung von Grafiken und Karten. Hiermit wird die Kartenerstellung und das Einfügen von Sichtbarkeitsbereichen ermöglicht.

3.4 Datenextraktion/-aufbereitung aus OSM

Die benötigten Geodaten stammen aus der Datenbank von OpenStreetMap. OpenStreetMap ist eine frei zugängliche Sammlung kartografischer Informationen. Mithilfe der Bibliothek OSMnx, welche eine Schnittstelle in Python zum Zugang und zur

Abfrage von Geodaten darstellt, werden gezielt alle relevanten Gebäudepolygone eines vorab definierten Bereichs abgerufen. Des Weiteren, zur Identifikation der Verbotsgebäude, wird eine Reihe spezifischer Filterungskriterien (OpenStreetMap-Tags) definiert. Die abgefragten Daten werden anschließend aufbereitet, indem sie in das jeweilige Zeitzonen-Koordinatensystem konvertiert werden. Diese Konvertierung erleichtert es, einen akkurate Radius für die Sichtberechnungen zu bestimmen. Außerdem werden nicht relevante Gebäude und Umgebungsdaten verworfen und alle Konturen werden in Shapely-Polygone konvertiert. Gleichzeitig erfolgte eine weitere Filterung, um unvollständige Daten verarbeiten zu können. Ein häufig angetroffenes Problem ist, dass bei der Abfrage der Geodaten, nicht das vollständige Gebäude, sondern nur ein repräsentativer Punkt innerhalb oder neben des gewünschten Gebäudes, zurückgegeben wird. Um solche Fälle einzubinden, wird entschieden, einen Radius um diesen Punkt zu ziehen, ähnlich wie bei bestehenden Ansätzen. Für jedes betrachtete Gebäude wurde zudem ein Umkreis von 100 Metern definiert. Innerhalb dieses Bereichs werden alle Gebäude als potenzielle Sichtbarriere gespeichert und mit in die Berechnungen einbezogen.

4 Implementierung

4.1 Codestruktur

Basierend auf den beschriebenen Hauptschritten gibt es auch voneinander getrennte Klassen, die die einzelnen Arbeitsschritte durchführen.

Die Klasse Extraktor (Extractor) ermöglicht, alle spezifizierten Gebäude in einer Umgebung abzurufen und extrahiert die Gebäudekonturen für die Weiterverarbeitung. Des Weiteren filtert diese für einzelne weitergereichte Gebäude auch alle Gebäude in einer Umgebung von 100 Metern und leitet auch diese Gebäudekonturen weiter.

Der Präprozessor (Preprocessor) hilft bei der unmittelbaren Berechnung des Sichtfeldes. Er stellt für die Grenze von 100 Metern alle Hindernisse in der Umgebung und das Beobachterpolygon die Kanten bereit, da mit diesen im Hauptalgorithmus gearbeitet wird. Außerdem wird im Falle von einem Beobachterpolygon und nicht einem Beobachterpunkt dieses auch in mehrere Punkte auf den Kanten unterteilt. Für diese Punkte werden anschließend die Berechnungen separat ausgeführt. Unter anderem werden einzelne Filterungen vorgenommen, wie beispielsweise einzelne Gebäude oder Kanten außerhalb des Radius von 100 Metern zu filtern.

Die Klasse Calc (kurz für Calculator/ Rechner) berechnet nun für einen einzelnen gegebenen Punkt und alle umliegenden Kanten den sichtbaren Bereich und gibt diesen zurück.

Der Mapper nimmt nun alle errechneten Sichtbereiche entgegen und projiziert diese auf eine interaktive Karte. Diese Karte wird mit einer Suchfunktion im Browser bereitgestellt , um einzelne Gebäude und Orte ausfindig machen zu können.

4.2 Algorithmusentwicklung

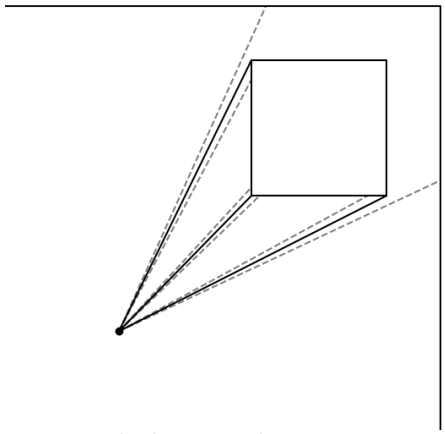


Abbildung 3: Beispiel Linien naiver Algorithmus

Die Implementierung ist in zwei Entwürfe unterteilt. In einem ersten Versuch ist ein naiver Algorithmus, basierend auf der Idee von Joe und Simpson, implementiert. In dieser Version wird von dem Beobachterpunkt erstmals versucht, alle sichtbaren Eckpunkte der Hindernisse zu finden. Hierfür wird von dem Beobachter eine Linie l zu jedem Eckpunkt gezogen. Falls l kein anderes Hindernis kreuzt, gilt der Eckpunkt als sichtbar. Der Unterschied zu dem Algorithmus von Joe und Simpson besteht darin, dass für l zwei weitere Linien hinzugefügt werden und derselbe Kollisionstest durchgeführt wird. Diese Linien haben zu l einen Winkelunterschied von ± 0.001 Grad. Schließlich werden all diese Linien bezüglich des Winkels zum Beobachter sortiert und die Endpunkte der Linien bilden in sortierter Reihenfolge den sichtbaren Bereich. Führen wir nun diesen Prozess für jeden Beobachterpunkt unseres Beobachterpolygons, welchen wir durch den Präprozessor erhalten, durch, dann bekommen wir den sichtbaren Bereich für das ganze Polygon.

Algorithmus 1 naiver Algorithmus

```
1:  function NAIVE(Punkt: Beobachter, Liste: Kanten, Liste: Ecken)
2:      sichtlinien ← Liste
3:      for all v ∈ Ecken do
4:          l ← Linie von beobachter zu v
5:          if l schneidet keine Kanten then
6:              sichtlinien ← sichtlinien ∪ l
7:              sichtlinien ← sichtlinien∪ versetzte Linien
8:          end if
9:      end for
10:     sichtlinien ← SortiereNachWinkel(sichtlinien)
11:     return sichtlinien
12: end function
```

Trotz einiger Verbesserungen des naiven Algorithmus stößt dieser, bezüglich der Laufzeit, schnell an seine Grenzen. Insbesondere durch das Hinzufügen der umliegenden Linien wird erfordert, dass jede neue Linie auf Schnittpunkte mit allen gegebenen Kanten überprüft werden muss. Mit steigender Komplexität der zu überprüfenden Hindernisse erweist sich der naive Ansatz als nicht praktikabel. Aus diesem Grund wird der Algorithmus mit dem Algorithmus der Triangular Expansion im weiteren Verlauf ersetzt. Neben der vorhandenen Version aus der Bibliothek SKGeom, welche viele robuste, unterliegende Datenstrukturen zur maximalen Optimierung verwendet, wird der Algorithmus nichtsdestotrotz auch selbst implementiert und ausgetestet. Die zuvor beschriebene allgemeine Vorgehensweise liefert ein mehrstufiges Verfahren und wird nach dem Algorithmus 2 implementiert.

In der ersten Phase, ausgehend von dem gewählten Beobachtergebäude, werden mehrere Beobachterpunkte entlang dessen Konturen erzeugt und für umliegende Gebäude, mittels deren Kanten, eine eingeschränkte Delaunay-Triangulierung erstellt. Im nächsten Schritt werden für alle Beobachterpunkte, mittels Triangular Expansion, eigene Sichtfeldberechnungen durchgeführt. Zuletzt werden dann alle berechneten Sichtbereiche zusammengeführt und in einer interaktiven Karte für die Benutzung freigegeben. Basierend auf den vorher beschriebenen Klassen sieht der Ablauf dann wie folgt aus:

Algorithmus 3 Ablauf

1: $gebaeude \leftarrow$ EXTRAKTOR.VERBOTSGEBÄUDE
2: $sichtbereich \leftarrow \emptyset$
3: **for all** $g \in gebaeude$ **do**
4: $\quad hindernisse \leftarrow$ EXTRAKTOR.UMGEBUNGFUER(g)
5: $\quad beobachter, kanten \leftarrow$ PRÄPROZESSOR.VERARBEITE($hindernisse, g$)
6: $\quad sichtbereich \leftarrow sichtbereich \cup$ CALC.BERECHNE($beobachter, kanten$)
7: **end for**
8: MAPPER.MAP($sichtbereich$)

Algorithmus 2 Triangular Expansion

Require: Dreiecke T, Abfragepunkt q

Ensure: Sichtbarkeitspolygon $V(q)$

1: Lokalisiere Dreieck $D \in T$, das q enthält
2: Setze $V(q) \leftarrow \emptyset$
3: **for** alle Kanten $(a, b) = e \in D$ **do**
4: Rufe EXPAND$(q, a, b, D, e, V(q))$ auf
5: **end for**
6: **return** $V(q)$

7: **function** EXPAND$(q, x, y, D, e, V(q))$
8: **if** Kante $e = (a, b)$ ist Polygongrenze **then**
9: Füge (x, y, e) zu $V(q)$ hinzu
10: **return** $V(q)$
11: **end if**
12: Bestimme benachbartes Dreieck $D' = (a, b, c)$ von D, das e teilt
13: Berechne die Winkel θ_x, θ_y und θ_c von q zu x,y und c
14: **if** $\theta_x < \theta_c$ und $\theta_y < \theta_c$ **then**
15: $e' = (a, c)$
16: EXPAND$(q, x, y, D', e', V(q))$
17: **else if** $\theta_x > \theta_c$ und $\theta_y > \theta_c$ **then**
18: $e' = (b, c)$
19: EXPAND$(q, x, y, D', e', V(q))$
20: **else**
21: $e'_1 = (a, c)$
22: EXPAND$(q, x, c, D', e'_1, V(q))$
23: $e'_2 = (b, c)$
24: EXPAND$(q, c, y, D', e'_2, V(q))$
25: **end if**
26: **end function**

5 Visualisierung und Evaluation

Die Visualisierung der berechneten Sichtbarkeitsbereiche stellt einen zentralen Bestandteil dieser Arbeit dar. Sie ermöglicht es, die berechneten Ergebnisse in einen geografischen Kontext zu setzen und deren praktische Bedeutung nachvollziehbar zu machen. Gleichzeitig dient die visuelle Darstellung auch der Überprüfung der Berechnungen, da sich Abweichungen, Fehler und Auffälligkeiten teilweise besser über eine visuelle Darstellung erkennen lassen als durch rein numerische Auswertungen.

5.1 Performanceanalyse

Für die Vergleichstests des eingebauten Algorithmus von SKGeom und der selbst implementierten Variante werden zwei verschiedene Szenarien durchlaufen und gegenübergestellt. Der erste Testfall verwendet die Ländergrenze von Kirgisistan Abbildung 4, welches ein einfaches Polygon mit insgesamt 30814 Kanten ist. Als Beobachterpunkt wird der geografische Landesmittelpunkt gewählt. Danach wird eine kleine Schule aus der Stadt Hilden ausgewählt. Hierfür werden außerdem alle umliegenden Gebäude in einem 100 Meter Radius mitgefiltert und nach der vorgestellten Methodik verarbeitet. Es entsteht ein Polygon mit Löchern, welches insgesamt 468 Kanten und 42 Löcher besitzt.

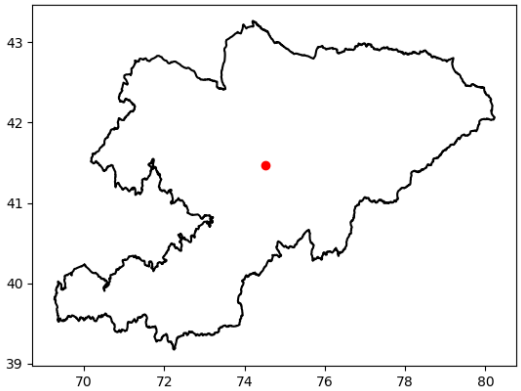
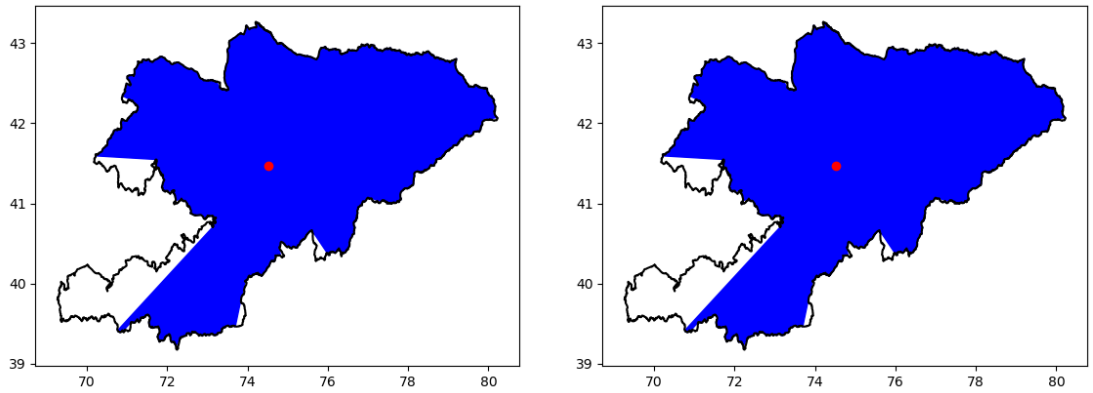


Abbildung 4: Kirgisistan mit Mittelpunkt in rot

Laufzeit für einfaches Polygon			
Algorithmus	Vorverarbeitung	Suche	Insgesamt
Triangular Expansion SKGeom	$36.73s$	$78,22ms$	$36.81s$
Triangular Expansion selbst implementiert	$28.59s$	$4.18s$	$33.17s$

Für den Fall eines einfachen Polygons lässt sich erkennen, dass die Zeit für die Vorverarbeitung im selbst implementierten Algorithmus schneller ist, beide Algorithmen aber in der gleichen Größenordnung liegen. Für die eigentliche Suche ist der selbst implementierte Algorithmus um fast zwei Größenordnungen langsamer.

Die folgende Abbildung veranschaulicht die Ergebnisse der Sichtbarkeitsberechnung beider Algorithmen. Hierbei wird gezeigt, dass beide Algorithmen übereinstimmende Ergebnisse liefern.



(a) Eigene Reproduktion Kirgisistans **(b)** SKGeom Sichtbereich Kirgisistans

Für den Vergleich eines Polygons mit Löchern wird eine zufällige Schule aus der Stadt Hilden benutzt. Das Polygon umschließt einen 100 Meter Umkreis um den Schulbereich. Umliegende Gebäude im Umkreis, die nicht die 100 Meter Grenze schneiden und völlig innerhalb des Bereichs liegen, gelten hierbei als potenzielle Löcher des Polygons. Falls diese Gebäude aneinander angrenzen sollten, beispielsweise wenn

eine Kante zweier Gebäudekonturen geteilt wird, werden sie zu einem Loch zusammengeschlossen. Zur weiteren Vereinfachung wird der Mittelpunkt des Schulbereichs als einzelner Beobachterpunkt zur Sichtfeldberechnung ausgewählt. In Abbildung 6 werden alle möglichen sichtverhindernden Gebäude mit einer Umrandung markiert und der Beobachterpunkt als Marker dargestellt.

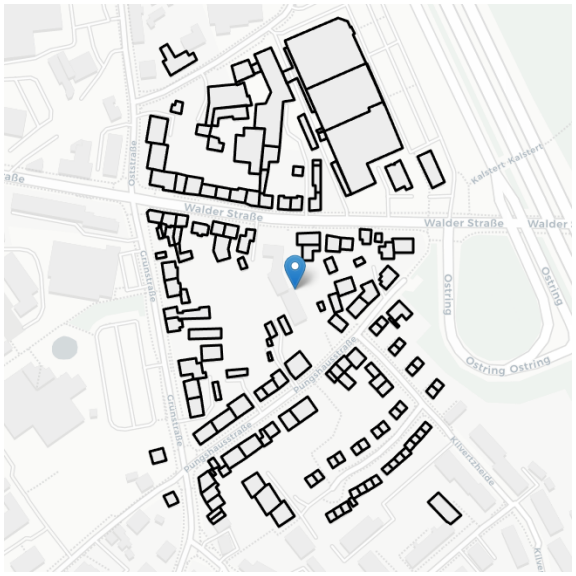
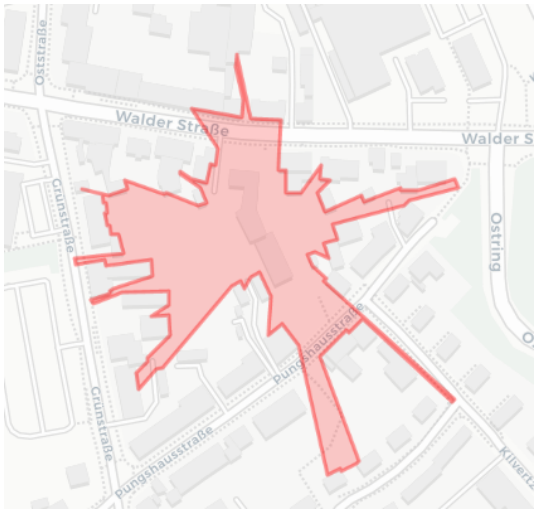


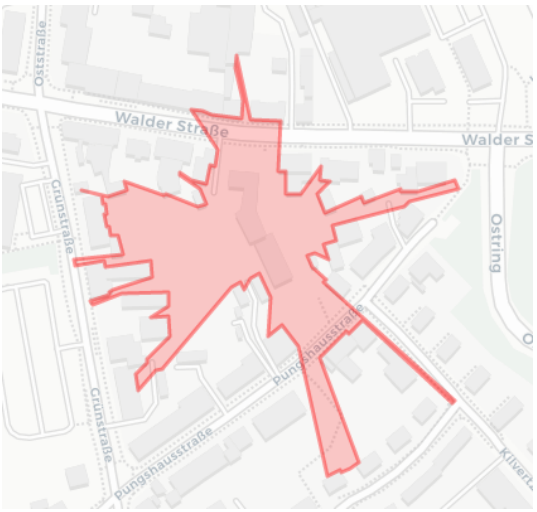
Abbildung 6: Schulgebiet mit markierten umliegenden Gebäuden und Beobachterpunkt

In der folgenden Abbildung wird der Sichtbereich dargestellt, wobei die erste Abbildung die Berechnung des selbst implementierten Algorithmus darstellt und die zweite Abbildung die Berechnung, mittels des eingebauten Algorithmus aus SKGeom, gegenüberstellt. Ähnlich wie bei der Berechnung für Kirgisistan stimmt auch für den ausgewählten Schulbereich die Berechnung mit dem selbst implementierten Algorithmus vollständig mit dem Ergebnis aus SKGeom überein.

Die Tabelle zeigt, dass die Vorverarbeitungszeit für beide Algorithmen in der gleichen Größenordnung liegt, die Suche des eigenen Algorithmus aber um eine Größenordnung langsamer ist. Ähnlich wie bei dem vorherigen Vergleichsbeispiel ist hier auch die Suche mittels SKGeom schneller.



(a) Eigene Reproduktion des Schulgebiets



(b) SKGeom Output des Schulgebiets

Laufzeit für Polygon mit Löchern			
Algorithmus	Preprocessing	Suche	Insgesamt
Triangular Expansion SKGeom	$6,36ms$	$1.01ms$	$7.37ms$
Triangular Expansion LPP	$18,91ms$	$32.30ms$	$51.21ms$

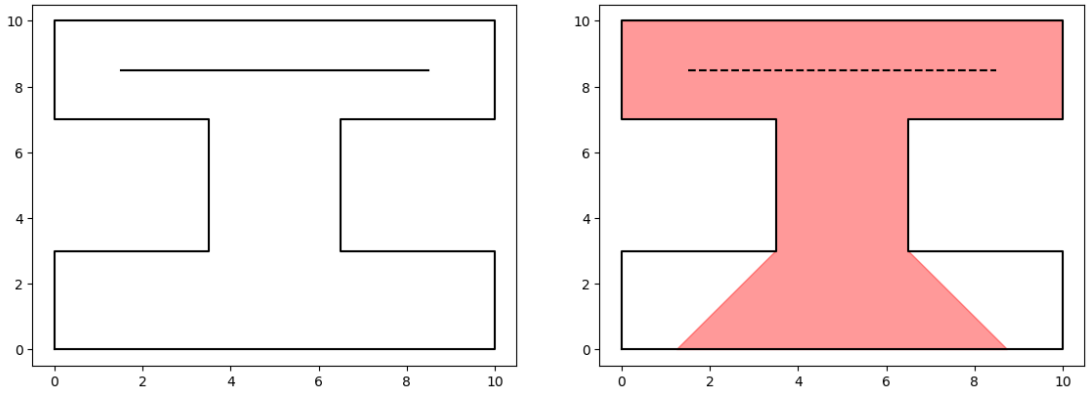
Zieht man die Ergebnisse beider Vergleiche in Betracht, so erkennt man, dass die Anzahl der Löcher wahrscheinlich keinen oder geringen Einfluss auf die unterschiedliche Suchzeit hat. Die Anzahl der Kanten sorgt höchstwahrscheinlich für die Differenz der Suchzeiten. Es liegt nahe, dass der selbst implementierte Algorithmus eine höhere Laufzeitkomplexität besitzt, als der Algorithmus aus SKGeom. Somit wurde nichtsdestotrotz gezeigt, dass die Algorithmen ein korrektes Ergebnis liefern. SKGeom läuft nach Erfahrung schneller, aufgrund der zugrunde liegenden robusten Datenstrukturen und tieferen Vorverarbeitung der Daten, welche eine schnelle Verarbeitung der Operationen während der Suche gewähren.

5.2 Fehleranalyse

Trotz der Präzision der Triangular Expansion weist die methodische Herangehensweise einen wesentlichen Fehler auf. Dieser ergibt sich aus der diskreten Punktwahl bei der Berechnung. Die Beobachterpunkte werden entlang der Gebäudekonturen gewählt und die Berechnung auf jedem dieser Punkte einzeln aufgerufen. Sollte der Abstand zwischen den Punkten zu groß gewählt sein, führt dies zu potenziellen Lücken im berechneten Sichtbereich. Insbesondere, falls Bereiche zwischen den Hindernissen schmal sein sollten, können tatsächliche Sichtbereiche "übersehen" werden.

In Abbildung 8 ist ein Polygon dargestellt, welches eine Verengung darstellt. Die gestrichelte Linie stellt hierbei den Beobachter dar. Der numerisch berechnete Wert für den Sichtbereich beträgt genau 57.75 Flächeneinheiten.

Abbildung 8



(a) Polygon mit Verengung und Sichtlinie

(b) Exakter Sichtbereich der Linie

Die Linie wird dabei in gleich große Liniensegmente "geschnitten". Die Beobachterpunkte ergeben sich aus den Endpunkten der Liniensegmente.

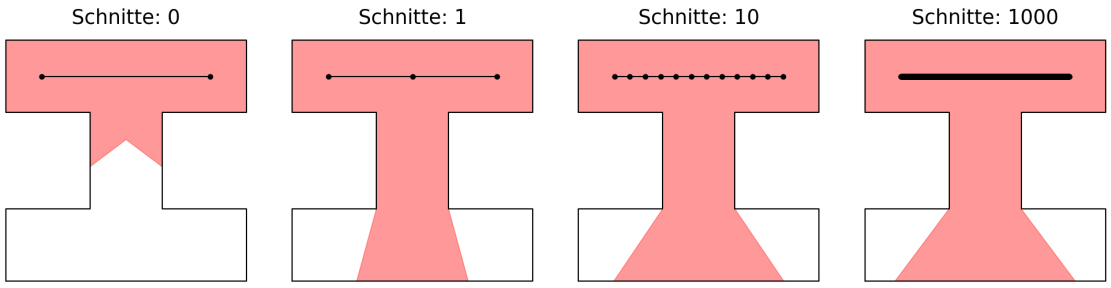


Abbildung 9: Sichtbereichvergleich im Polygon mit Verengung

Sichtbereich des Polygons für verschiedene Anzahl an Schnitten		
Schnitte	Flächeneinheiten	Fläche in Prozent
0	35.06	60.71
1	53.45	92.56
2	55.36	95.87
5	57.27	99.17
8	56.64	98.07
10	57.10	98.87
20	57.55	99.65
50	57.61	99.76
100	57.71	99.93
1000	57.74	99.98
10000	57.75	100.00

Der Tabelle kann man entnehmen, dass die Fläche des berechneten Sichtbereichs sich dem tatsächlichen Sichtbereich, mit steigender Anzahl an Beobachterpunkten, im Allgemeinen annähert. Eine geschickte Auswahl an Beobachterpunkten kann auch einen Unterschied machen. So beträgt für fünf Schnitte die prozentuale Fläche bereits 99.17%, liegt jedoch für acht Schnitte nur bei 98.07%. Nichtsdestotrotz zeigt dieses Ergebnis, dass dieser Ansatz im Allgemeinen eine Unterschätzung für den tatsächlichen Sichtbereich liefert. Um eine Überschätzung zu erhalten, kann der berechnete Sichtbereich in alle Richtungen vergrößert werden. Zur weiteren Präzision wird der Bereich außerhalb des möglichen Sichtbereichs entfernt.

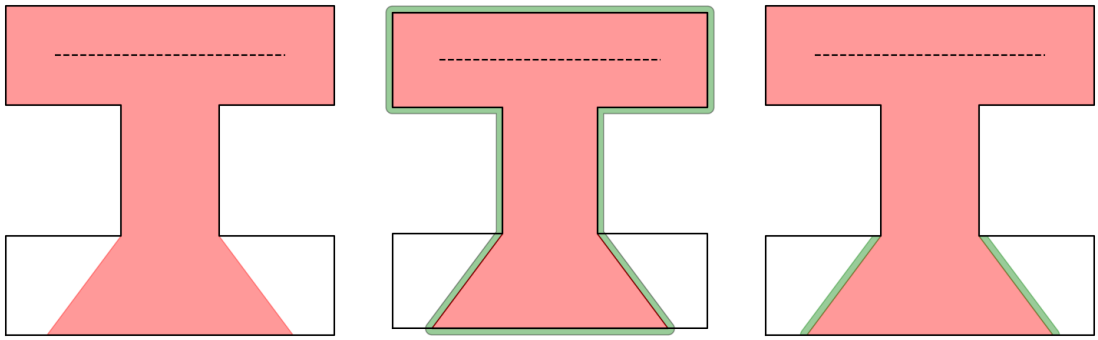


Abbildung 10: Vergrößerung des Sichtbereichs im Polygon

5.3 Vergleich mit bestehenden Ansätzen

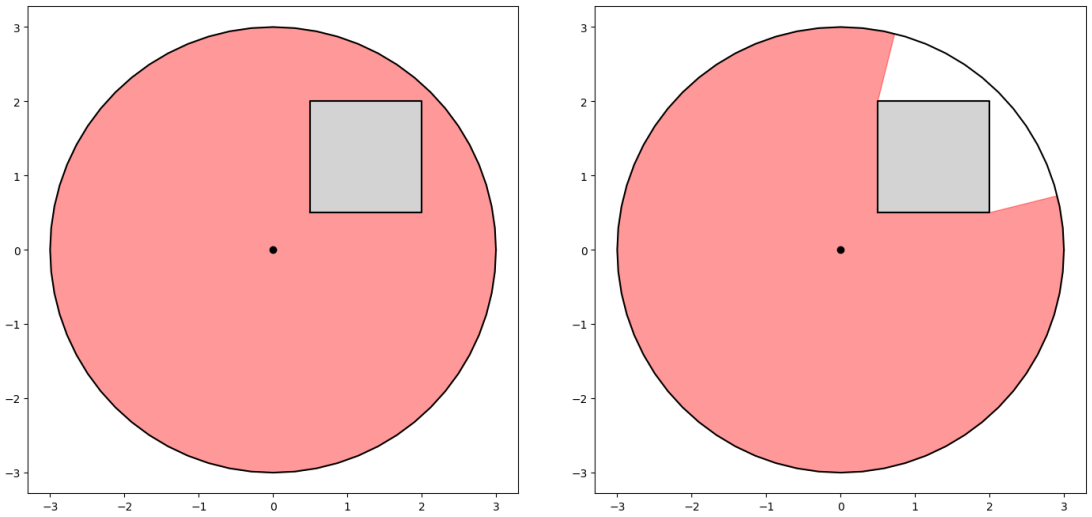
Ein wesentlicher Bestandteil der Evaluation besteht darin, tatsächlich zu messen, wie hoch der Anteil der "gewonnenen" Fläche der gewählten Methodik ist, beispielsweise im Vergleich zu einem bestehenden Ansatz. Für den Vergleich stellen wir die Vorgehensweise der Bubatzkarte mit der Berechnung eines Sichtbarkeitspolygons gegenüber. Der Ansatz der Bubatzkarte vergrößert die Grenzen der jeweiligen Verbotsgebäude um 100 Meter in alle Richtungen. Dies ergibt, wie in Abbildung 12a erkennbar ist, eine Pufferzone. Diese Pufferzone stellt das Sichtfeld ohne Hindernisse dar und wird aus diesem Grund maximaler Sichtbereich genannt.



(a) Eigene Reproduktion der Bubatzkarte **(b)** Referenzfoto Bubatzkarte

Abbildung 11: Kartenausschnitt Düsseldorfs

Die gewählte Methode, den Flächengewinn zu beschreiben, ist die Berechnung der totalen und prozentualen Abnahme der Fläche gegenüber einem maximalen Sichtbereich. Dies bietet einen direkten Vergleich der Fläche nach Anwendung des Algorithmus in Bezug auf bestehende Ansätze.



(a) Maximaler Sichtbereich **(b)** Sichtbarkeitspolygon

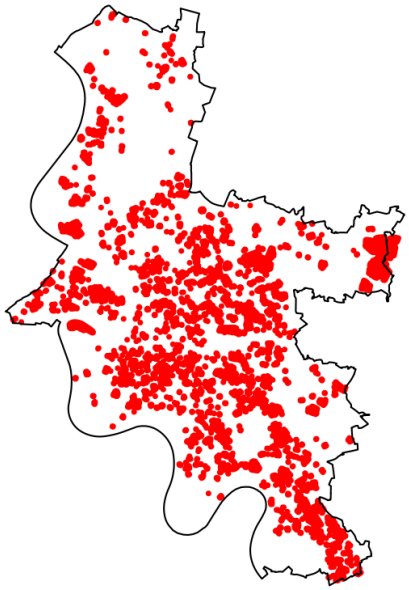
Abbildung 12: Beispiel Abnahmegewinn

Das Beispiel, siehe Abbildung 12, stellt einen Beobachter mit einer Sichtweite von drei Metern dar. Der maximale Sichtbereich ergibt somit eine Fläche von $\pi * 3^2 \approx 28.27m^2$. Das mittels Triangular Expansion berechnete Sichtbarkeitspolygon besitzt eine Fläche von gerundet $24.12m^2$. Dies entspricht einer totalen Abnahme von $4.15m^2$ und einer prozentualen Abnahme von knapp $14,7\%$.

Mit dieser Definition kann nun der tatsächliche Flächengewinn, anhand eines realen Beispiels, quantifiziert werden. Im Folgenden wird der sichtbare Bereich, im Vergleich zu einem maximalen Sichtbereich, anhand der Stadt Düsseldorf berechnet. Mit Hilfe einer eigenen Reproduktion des maximalen Sichtbereichs der Verbotsgebäude, kann nun zuerst die gesamte Fläche berechnet werden, unter der Annahme eines Sichtradius von 100 Metern.

Hierbei ergibt sich ein Sichtbereich von über $57,41km^2$. Bei einer gesamten Fläche von $217,41km^2$ entspricht dies $26,4\%$ der Fläche Düsseldorfs. Ein kleiner Überlauf der Sichtbereiche über die Stadtgrenze folgt daraus, dass einige Orte nahe an der Grenze Düsseldorfs liegen und auch über die Grenze hinweg fortfahren.

Abbildung 13: Düsseldorfer Verbotszonen mit maximalem Sichtbereich



Mittels der Berechnung, mit einer maximalen Schnittlänge von 20 Zentimeter entlang der Gebäudekanten, reduziert sich der Sichtbarkeitsbereich auf eine Fläche von knapp $44,34km^2$. Dies entspricht einer totalen Abnahme von $13,07km^2$ und einer prozentualen Abnahme von knapp $22,7\%$.

Auf Grund von fehlenden Einträgen im Datensatz werden einige Gebäude nur anhand von Punkten, anstelle von Polygonen, repräsentiert und eine konkrete Sichtbarkeitsberechnung ist ohne Weiteres nicht möglich. Für diese Orte wird der maximale Sichtbereich ermittelt und der Abnahmeeffekt geht teilweise verloren. Aus diesem Grund wird zum Vergleich die Sichtbarkeitsberechnung auf dem Teildatensatz, in dem die Gebäudekonturen nicht als Punkte dargestellt werden, separat durchgeführt, um an diesem Beispiel das Potenzial der gewählten Methodik hervorzuheben.

Vergleiche der Sichtbarkeitsbereiche		
	Vollständiger Datensatz	Datensatz ohne Punkte
Maximale Sichtbarkeit	$57,41km^2$	$51,14km^2$
Berechnete Sichtbarkeit	$44,34km^2$	$34,37km^2$
Totale Abnahme	$13,07km^2$	$16,77km^2$
Prozentuale Abnahme	$22,7\%$	$32,8\%$

Wie der Tabelle zu entnehmen ist, liefert der Algorithmus bei dem Datensatz ohne Punkte eine prozentuale Abnahme von $32,8\%$. Somit liegt das Potenzial für die gewählte Stadt Düsseldorf bei ungefähr einem Drittel.

Für ausgewählte Gebäude kann diese Differenz aber auch deutlich ansteigen. Ein Beispiel eines Gebäudes in Düsseldorf ist Abbildung 14 zu entnehmen. Die Abbildung stellt den Bereich auf einer Karte um das ausgewählte Gebäude herum dar. In Rot markiert ist der berechnete Sichtbereich und in Grün die überschüssige Fläche des maximalen Sichtbereichs dargestellt.

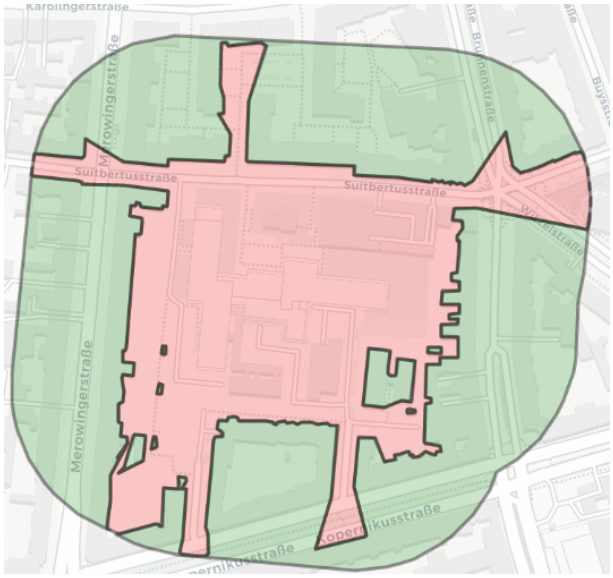


Abbildung 14: Direkter Vergleich: maximaler - berechneter Sichtbereich

Der maximale Sichtbereich um den gewählten Platz beträgt hierbei über $114.200m^2$. Nach der Berechnung sinkt der Sichtbereich auf rund $46.400m^2$, sodass für dieses einzelne Gebäude die prozentuale Abnahme knapp 60% beträgt.

Auf der anderen Seite kann auch der Fall eintreten, dass für ein spezifisches Gebäude keine umliegenden Gebäude als Sichtbehinderung vorliegen. Sollte dieser Fall eintreten, so stimmt die Fläche des maximalen Sichtbereichs komplett überein mit dem berechneten und die Abnahme liegt folglich bei 0%.

5.4 Darstellung auf interaktiver Karte

Für die Zugänglichkeit der Sichtbereiche wird sich entschieden, die ausgewerteten Daten auf einer interaktiven Karte darzustellen. Die Karte wird auch in Python mittels des Pakets Folium dargestellt. Alle berechneten Sichtbereiche werden auf dem jeweiligen Gebiet abgebildet. Die Karte ist lokal im Browser zugänglich. Zur deutlichen Darstellung bekommen die jeweiligen Bereiche schwarze Umrandungen. Abbildung 15 stellt einen beispielhaften Ausschnitt dar.



Abbildung 15: Kartenausschnitt Düsseldorfs

Des Weiteren besitzt die Karte eine Funktion zur Suche von spezifischen Orten und Gebäuden, um Verbotszonen leichter ausfindig zu machen. Bei erfolgreicher Suche wird das Kartenzentrum auf den gefundenen Ort gelegt und mit einer Markierung versehen.

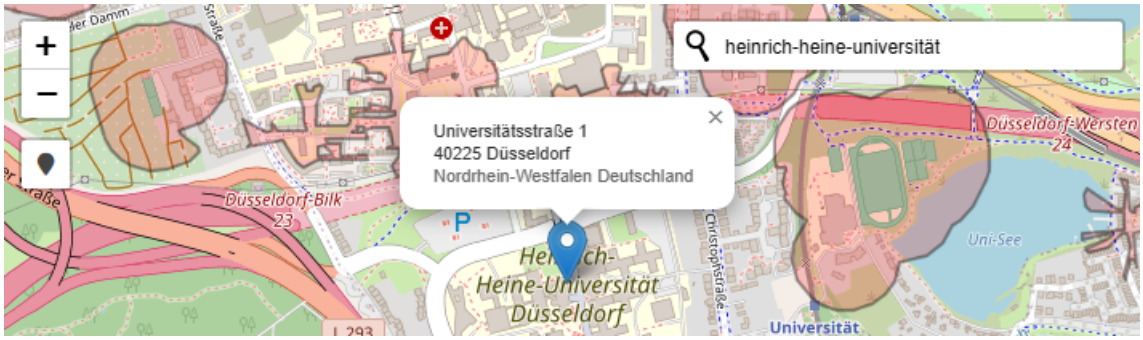
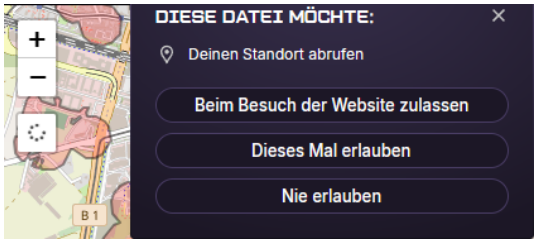
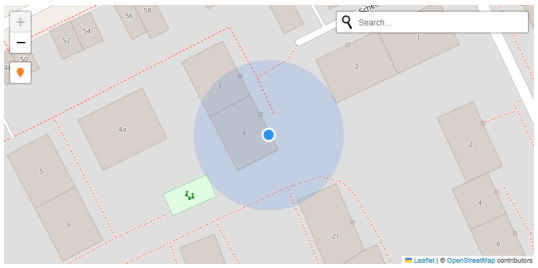


Abbildung 16: Beispiel Standortsuche

Zuletzt existiert die Funktion zur Suche des eigenen Standorts. Diese ermöglicht, den aktuellen Aufenthaltsort auf der Karte anzeigen zu lassen, sowie umliegende Verbotszonen schnell einsehen zu können.





6 Schluss

6.1 Diskussion

Der zuerst gewählte naive Ansatz stellt sich für kleine Mengen als praktikabel und weitestgehend akkurat dar. Dennoch, im Falle der Berechnung von größeren Datenmengen, wie etwa ganzen Städten, erweist die Implementierung deutliche Performanzprobleme. Obwohl die Berechnung in vielen Fällen nahe am tatsächlichen Ergebnis liegt, ist der Fehler eine Unterschätzung des wahren Sichtbarkeitsbereichs. Dies sollte eigentlich vermieden werden, um Komplikationen mit dem Gesetz zu vermeiden. Die Ersetzung des Algorithmus, durch den Algorithmus der Triangular Expansion, gewährt eine deutliche Verbesserung der Performanz. Durch diese Überarbeitung kann somit der Algorithmus für präzisere Berechnungen verwendet werden und mittels der Vergrößerung des Sichtbereichs eine Überschätzung erhalten. Eine Verarbeitung von größeren Datenmengen, wie beispielsweise großen Städten, kann somit ohne Probleme gewährleistet werden.

Abseits davon gibt es außerdem noch einige methodische Herausforderungen.

Unter anderem hängt die Genauigkeit der Sichtbarkeitsberechnung stark von der Genauigkeit und Qualität der Daten aus OpenStreetMap ab. Falls dort fehlende oder verfälschte Daten abgerufen werden, hat dies auch verfälschte Berechnungen zur Folge.

Für Berechnungen und die Evaluation wird außerdem die Annahme getroffen, dass es sich ausschließlich um zweidimensionale Daten handelt. Höheninformationen und andere topologische Faktoren werden nicht berücksichtigt. Im realen Kontext kann dies einen erheblichen Unterschied machen. Eine weitere Vereinfachung ist die Annahme, dass Hindernisse lediglich aus Gebäuden bestehen. Hindernisse wie etwa Bäume, Mauern oder auch Zäune werden nicht in die Berechnungen eingeschlossen.

Trotz dieser Einschränkungen bietet die angewandte Methodik eine solide Annäherung für die Bestimmung von Sichtbarkeitszonen, insbesondere in urbanen Gebieten.

6.2 Fazit

Ziel dieser Arbeit ist es, ein Verfahren zur algorithmischen Berechnung von Sichtbarkeitszonen anhand von Gebäudekonturen zu erstellen. Besonderer Fokus hierbei haben die Gebäude, deren Sichtbereich die Cannabis-Verbotszonen darstellen, gemäß der aktuellen Gesetzeslage.

Das Verfahren zeigt für die Stadt Düsseldorf, dass, im Vergleich zum maximalen Sichtbereich, über ein Fünftel an Fläche außerhalb des berechneten Sichtbarkeitsbereichs liegt. In Bezug auf den Teildatensatz, in welchem die Gebäudekonturen ausschließlich als Polygone enthalten sind, liegt die überschüssige Fläche schon bei nahezu einem Drittel. Dies weist auf eine deutliche Verbesserung hin. Vor allem in städtischen Gebieten liefert das Verfahren einen verkleinerten Sichtbereich, aufgrund vieler möglicher Hindernisse. Vor allem zeigt dies aber auch, dass realitätsnähere Darstellungen umsetzbar sind.

6.3 Ausblick auf weiterführende Forschung

Dieses Verfahren bietet eine grundlegende Idee zur Annäherung an den tatsächlichen Sichtbereich von Gebäuden. Ein naheliegender nächster Schritt besteht in der Erweiterung des Verfahrens auf eine dreidimensionale Ebene. Hierfür werden, zusätzlich zu den Gebäudekonturen, auch die Höheninformationen benötigt. Diese Änderung gewährt eine realitätsnähere Abbildung des Sichtbereichs. Auch die Datenqualität bietet Raum für Verbesserung. OpenStreetMap liefert bereits eine große Datenbank zur Analyse, besitzt aber nichtsdestotrotz vermehrt Lücken und unzureichende Informationen. Mit Zugriff auf genauere Daten lassen sich im Allgemeinen auch genauere Berechnungen durchführen. Darüber hinaus ist es auch möglich, das Verfahren auf das ganze Land anzuwenden. Mittels Webanwendung kann dann eine interaktive Karte für alle Bürger:innen den Zugang zu den konkreten Zonen ermöglichen.

7 Source Code Listings

Quellcode 1: Filter für die Häuserauswahl

```
tags = {
"leisure":["playground", "pitch", "sports_hall", "sports_centre",
           "horse_riding", "stadium", "water_park", "golf_course",
           "indoor_play"],
"amenity":["school", "kindergarten", "childcare"],
"building":"kindergarten",
'name':'*Jugendherberge',
'social_facility':"for=child social_facility:for=juvenile",
"healthcare":"speciality=child_psychiatry",
"community_centre":["youth_centre",
                    "for=child community_centre:for=juvenile",
                    "for=girl community_centre:for=boy"],
"club":"cannabis",
"sport":"swimming",
"smoking":"cannabis=no"
}
```

Literatur

[Asa85] Asano, Tetsuo: Efficient algorithm for finding the visibility polygon for a polygonal region with holes. In: *IEICE Transactions* 68 (1985), Nr. 9, S. 557–559. http://dx.doi.org/10.24517/00062806. – DOI 10.24517/00062806

[BHH+14] Bungiu, Francisc ; Hemmer, Michael ; Hershberger, John ; Huang, Kan ; Kröller, Alexander: *Efficient Computation of Visibility Polygons*. https://arxiv.org/abs/1403.3905. Version: 2014. – arXiv preprint

[Boe25] Boeing, Geoff: Modeling and Analyzing Urban Networks and Amenities with OSMnx. In: *Geographical Analysis* (2025). http://dx.doi.org/10.1111/gean.70009. – DOI 10.1111/gean.70009. – Online ahead of print

[bub24] bubatzkarte.de: *Bubatzkarte – Cannabis-Konsumzonen in Deutschland*. https://bubatzkarte.de. Version: 2024. – Zugriff am 19. März 2025

[Deu24] Deutschland, Bundesrepublik: *§ 5 KCanG – Konsumverbot*. https://www.gesetze-im-internet.de/kcang/__5.html. Version: 2024. – Zugriff am 19. Mai 2025

[G+te] Gillies, Sean u. a.: *Shapely: manipulation and analysis of geometric objects*. https://github.com/Toblerity/Shapely. Version: 2007-Heute

[Gho07] Ghosh, Subir K.: Visibility Algorithms in the Plane. (2007). http://dx.doi.org/10.1017/CBO9780511543340. – DOI 10.1017/CBO9780511543340. ISBN 9780521875745

[JS87] Joe, B. ; Simpson, R. B.: Corrections to Lee's visibility polygon algorithm. In: *BIT Numerical Mathematics* 27 (1987), Nr. 4, S. 458–473. http://dx.doi.org/10.1007/BF01937271. – DOI 10.1007/BF01937271