

LV Robotik Projektvorstellung

Wegfindung in 2 nahezu gleichgroßen 2D-Räumen mit Hindernissen



Gliederung

- Aufgabe und Anforderungen
- Konzept
- Umsetzung
 - Interaktion
 - Berechnungen
 - Darstellung
- Demonstration eines Anwendungsfalls
- Fazit



Aufgabenstellung und die daraus entstandenen Anforderungen



Aufgabenstellung

- Pfad zwischen Start- und Endpunkt mit mehreren Wegpunkten
- 2 nahezu gleichgroße Räume mit einer Tür mit statischen Hindernisse
- Statistische Verteilung von Wegpunkten
- Kollisionsprüfung des Weges und interaktive Anpassung der Anzahl von Wegpunkten bei Kollision
- Verwendung von Dijkstra, A* (optional) und Catmull-Rom Splines
- Anzeige der Laufzeiten für verschiedene Abschnitte



Anforderungen

- Verarbeitung von Bilddaten
- Benutzerschnittstelle (GUI oder Konsole)
- Zeichnen und einbetten einer Szene
- Modellierung von Wegpunkten in einem Raum
- Verschiedene Berechnungen zur Problemlösung
- Evtl. Darstellung einzelner Zwischenergebnisse
- Laufzeitmessungen von verschiedenen Programmabschnitten



Konzept, Grobarchitektur



Konzept: GUI

- 2 spaltiges GUI → Einstellungen links, Szene rechts
- Interaktives Ändern der Szene
- Optionen:
 - Wegpunkte hinzufügen/löschen
 - Debugging
 - Wahl von verwendeten Algorithmen
 - Animation eines punktförmigen Roboters
- Status und Hilfstexte
- Laden/Speichern von Projekten und Räumen



Konzept: Szene & Berechnungen

- Farbliche Trennung der dargestellten Objekte
- Initiales Setzen von Wegpunkt in Tür
- Willkürliches Setzen von Start- und Endpunkt zu Beginn
- Berechnung des Pfades bei Setzen oder Löschen von Punkten



Umsetzung

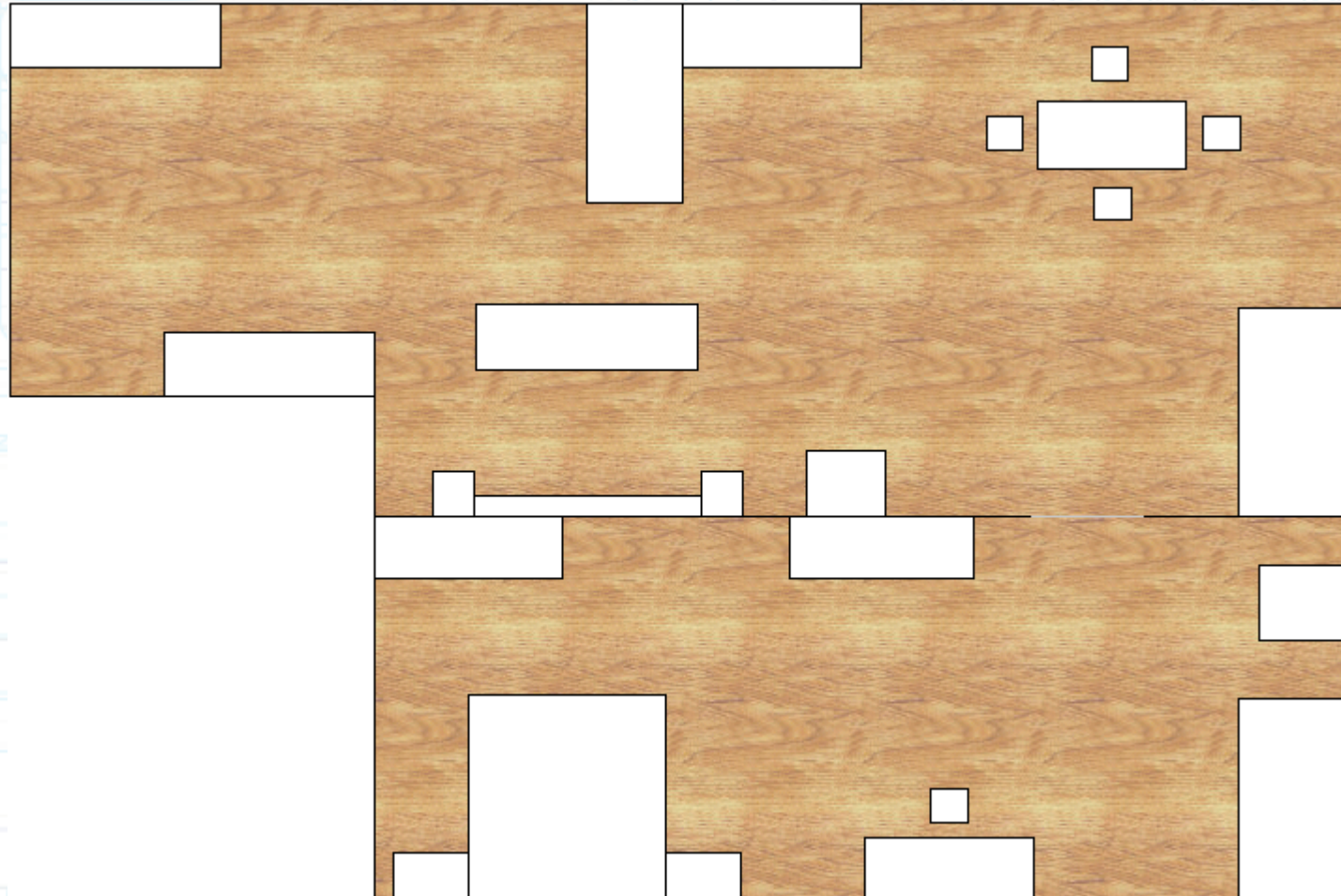


Umsetzung: Bildverarbeitung (1)

- Laden eines Bildes → DevIL-Bibliothek
- Farbanalyse und Kategorisieren von Pixeln
- Umriss, Türen, Objekte → Polygone
- Raumumriss: Umlaufen, Eckpunkte ermitteln
- Objekte: „Innen“-Pixel expandieren
- Tür: Umlaufen der grauen Pixel
- Berücksichtigen des Radius des Roboters



Umsetzung: Bildverarbeitung (2)



Umsetzung: GUI (1)

- Qt 5.4 (Beta): QOpenGLWidget
- Wenige Menüs: Load/Save/Quit
- Optionen als Checkbox/Dropdown
- Hilfetexte bei Mouseover
- Statusnachrichten, falls in der Berechnung Ereignisse auftreten



Umsetzung: GUI (2)

Project Quit

Amount of nodes

☐ Add waypoint

☐ Remove waypoint

☐ Set startpoint

☐ Set endpoint

☐ Show triangulation

☐ Show room triangulation

☐ Show waypoints

☐ Show path

☐ Show neighbours

Animate

Statistics

Algorithms

Dijkstra

Status

Startpoint (716/371) outside domain, can't insert.

Help

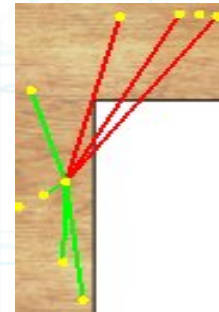
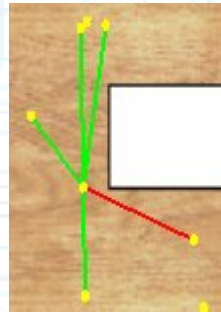


Umsetzung: Wegfindung (1)

- Graphen aus Triangulierung ableiten
- Dijkstra:
 - Gewichtung = euklidischer Abstand
- A*:
 - Gewichtung = Heuristik = euklidischer Abstand
 - Ansonsten gleicher Code (Dijkstra Spezialfall von A* mit $h(x) = 0$)
- Kollisionsprüfung → Nachbarn verwerfen



Umsetzung: Wegfindung (2)



Umsetzung: Triangulierung (1)

- (Constrained) Delaunay Triangulation
- DT:
 - möglichst wenig „kleine“ Dreieckswinkel
 - Kreis der alle Eckpunkte umfasst, enthält keine anderen Punkte
- CDT:
 - Manche Strecken müssen enthalten sein
→ Constraints
 - Nicht immer Delaunay
- Implementierung in CGAL



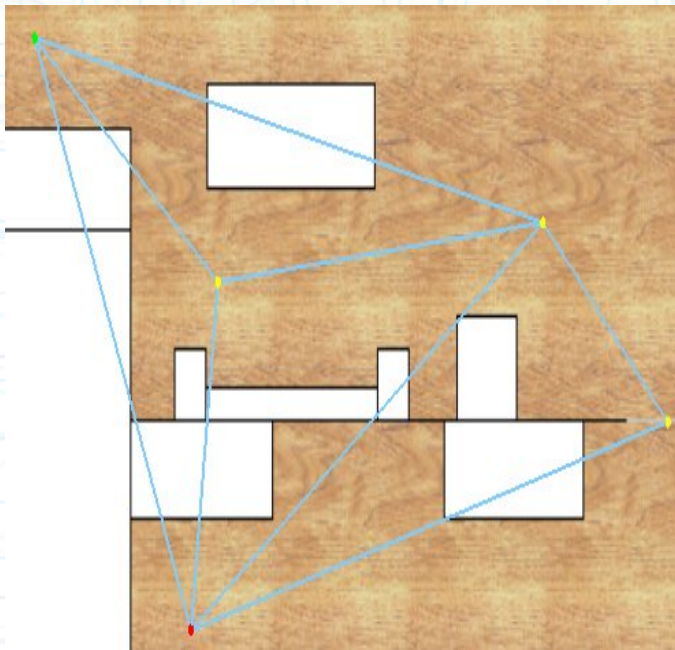
Umsetzung: Triangulierung (2)

- DT für Triangulierung der Wegpunkte
 - Basis eines gerichteten Graphen
- CDT für Triangulierung des Raumes
 - einfache Überprüfung, ob Punkte im Raum liegen
 - Kollisionsprüfung mit Grenzen

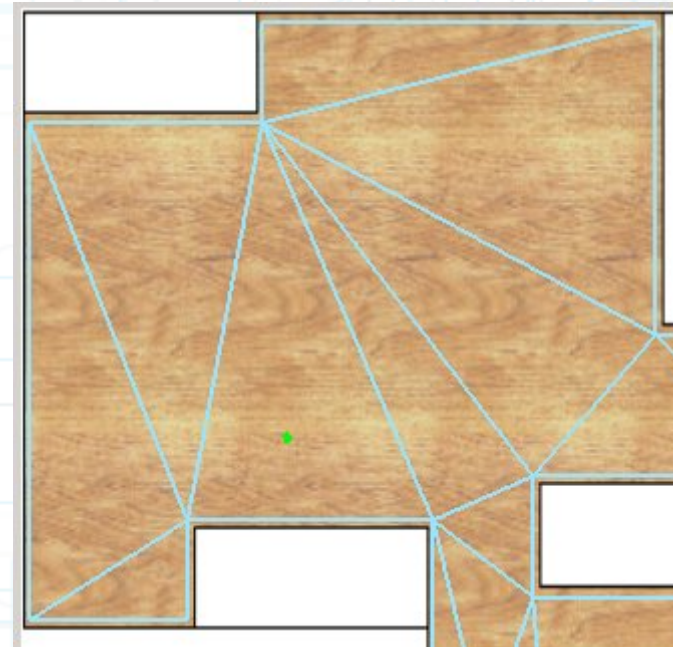


Umsetzung: Triangulierung (3)

Delaunay Triangulation



Constrained Delaunay Triangulation



Umsetzung: Kurvendarstellung des Wegs (1)

- Catmull-Rom Splines
- Kubischer Spline
 - 4 Stützpunkte (die mittleren 2 werden durchlaufen)
 - Dadurch besondere Betrachtung von Start- und Endsegment
 - Start-/Endpunkt verdoppeln
- Granularität der Segmente
- Grundstein der Animation

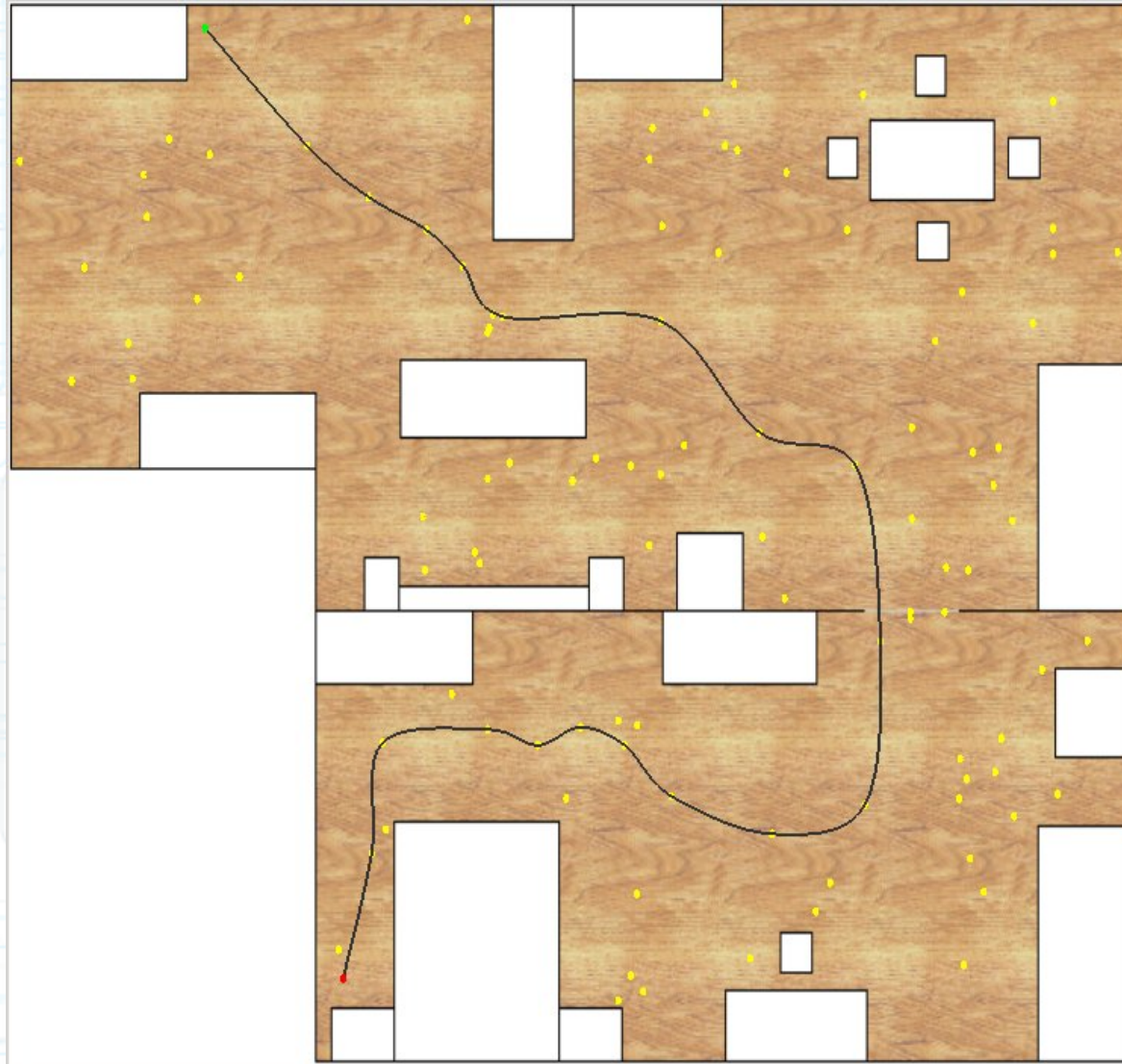


Umsetzung: Kurvendarstellung des Wegs (2)

- Geringe Auflösung führt zu „pixeligen“ Kurven
- Kollisionsprüfung mit CDT



Umsetzung: Kurvendarstellung des Wegs (3)



Demonstration eines Anwendungsfalls

- Statistisches und Interaktives Setzen von Stützpunkten
- Triangulierungen zeigen
- Kollisionsprüfung verdeutlichen
- Statistiken anzeigen



Fazit

- Projektaufgabe erfüllt und Wissen vertieft
- Laufzeit angemessen
- Geringer Anteil Algorithmen bzgl. Robotik
- Optimierung in der Berechnung
 - Geometriekentnisse
 - Was tun bei Implementierung auf μC ?
- Aufwand bei 4k LOC recht hoch
- Evtl. andere/mehr Bibliotheken nutzen



Vielen Dank fürs Zuhören!



Eure Fragen/Meinungen

