Verständliche Erklärung zentraler Konzepte des RRT-Algorithmus

Wilfried Ornowski

June 1, 2025

Einführung: Was ist RRT (Rapidly-exploring Random Tree)?

Stell dir vor, du bist in einem dunklen Labyrinth. Du kennst deinen Startpunkt und dein Ziel, aber du weißt nicht, wie die Gänge verlaufen. Also gehst du mit einer Taschenlampe los, probierst zufällig Wege aus und markierst die Abschnitte, die frei sind. Nach und nach entsteht ein Baum von Wegen, die alle von deinem Startpunkt ausgehen – und irgendwann findest du den Pfad, der dich zum Ziel bringt.

Genau das macht der RRT-Algorithmus:

- Er baut schrittweise einen *Baum von Pfaden* auf.
- Jeder neue Ast (Verbindung) geht zu einem *zufälligen Punkt*, aber nur dann, wenn der Weg dorthin *nicht blockiert* ist.
- Wenn das Ziel erreicht wird, kann man den Pfad vom Ziel zurück zum Startpunkt rekonstruieren.

Warum ist das nützlich? Weil RRT auch in sehr komplexen oder unbekannten Umgebungen funktioniert – sogar in hochdimensionalen Räumen (z. B. für Roboterarme mit mehreren Gelenken).

Wie funktioniert RRT Schritt für Schritt?

1. **Start:** Der Baum beginnt beim Startpunkt.
2. **Sampling:** Ein neuer Punkt im Raum wird ausgewählt – meist zufällig.
3. **Nächster Punkt:** Der existierende Baumknoten, der am nächsten zum neuen Punkt liegt, wird gesucht.
4. **Steuerung:** Der Algorithmus bewegt sich mit einer festen Schrittweite in Richtung des neuen Punktes.
5. **Kollisionsprüfung:** Die Verbindung zum neuen Punkt wird auf Hindernisse überprüft (siehe unten).
6. **Zielprüfung:** Ist der neue Punkt nahe genug am Ziel, wird der Pfad zurückverfolgt.

1. Wie funktioniert die Kollisionsprüfung bei RRT genau?

Die **Kollisionsprüfung** (engl. *collision checking*) beantwortet die Frage:

Verläuft die geplante Bewegung zwischen zwei Punkten durch ein Hindernis?

Vorgehensweise

Angenommen, man möchte von einem existierenden Punkt P_{start} zu einem neuen Punkt P_{new} wachsen. Dann wird wie folgt geprüft, ob auf dem Weg dorthin ein Hindernis liegt:

1. **Interpolation:** Die Strecke zwischen den beiden Punkten wird in n Zwischenpunkte aufgeteilt, z. B. für $u = 0, 0.1, 0.2, \ldots, 1.0$:
$$x(u) = (1-u) \cdot x_{\text{start}} + u \cdot x_{\text{new}} \quad y(u) = (1-u) \cdot y_{\text{start}} + u \cdot y_{\text{new}}$$
2. **Prüfung:** Jeder Zwischenpunkt $(x(u), y(u))$ wird daraufhin überprüft, ob er innerhalb eines Hindernisses liegt. Bei rechteckigen Hindernissen (Achsen-parallel) gilt:
$$\text{if } x_{\min} \leq x(u) \leq x_{\max} \text{ und } y_{\min} \leq y(u) \leq y_{\max}$$
3. **Entscheidung:**
   - Wenn ein Punkt innerhalb eines Hindernisses liegt: `Kollision erkannt`.
   - Wenn kein Punkt im Hindernis liegt: Die Verbindung ist `kollisionsfrei`.

Fazit: Nur kollisionsfreie Kanten werden dem RRT-Baum hinzugefügt. Das garantiert, dass der Algorithmus keine unzulässigen Wege plant.

2. Was bedeutet `Goal Bias` im RRT-Algorithmus?

Goal Bias ist ein gezielter Mechanismus, mit dem der RRT-Algorithmus gelegentlich den Zielpunkt direkt auswählt, statt rein zufällig zu samplen.

Ziel und Wirkung

- Normalerweise wählt RRT zufällige Punkte im Raum.
- Mit einer bestimmten Wahrscheinlichkeit (z. B. 5–10 %) wird stattdessen direkt der Zielpunkt als *Sample* verwendet.

Beispiel im Pseudocode:

```
if random() < goal_sample_rate:
    sample = goal
else:
    sample = random_point()
```

Vorteil: Der Baum wächst häufiger in Richtung Ziel – die Wahrscheinlichkeit, das Ziel früher zu erreichen, steigt.
Nachteil: Zu viel Ziel-Bias kann dazu führen, dass der Baum stecken bleibt oder keine alternativen Routen erforscht.

Typische Werte

- `goal_sample_rate` ≈ 0.05 bedeutet: In 5 % der Fälle wird das Ziel als Sample gewählt.
- Der Rest sind echte Zufallspunkte, damit der Baum weiter explorativ bleibt.

Zusammenfassung

Konzept	Bedeutung und Zweck
RRT allgemein	Baut einen Baum von Pfaden durch zufällige Stichproben im Raum auf. Findet gültige Wege ohne die gesamte Umgebung vorher zu kennen.
Kollisionsprüfung	Sorgt dafür, dass nur gültige Pfade (keine Hindernisdurchquerungen) in den Baum aufgenommen werden.
Goal Bias	Erhöht die Chance, das Ziel schneller zu erreichen, indem gelegentlich das Ziel direkt als Sample gewählt wird.

Diese drei Konzepte machen den RRT-Algorithmus in vielen praktischen Anwendungen so effektiv – besonders in dynamischen oder unvollständig bekannten Umgebungen.