

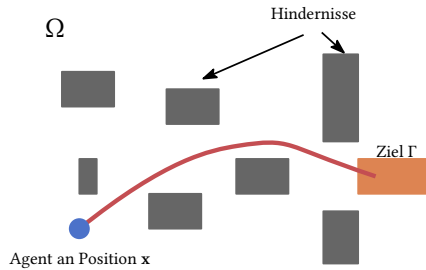
Algorithmen der Navigation

Benedikt Zönnchen



2. Juni 2021

Problemstellung



Problemstellung

Sei

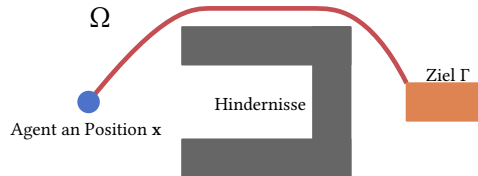
$$d_{\Gamma}(\mathbf{x}) = \min_{\mathbf{y} \in \Gamma} \|\mathbf{x} - \mathbf{y}\|$$

der Euklidische Abstand zwischen \mathbf{x} und dem Ziel Γ .

Falls es keine Hindernisse in dem Gebiet gibt können wir einfach entlang von $-\nabla d_{\Gamma}$ laufen:



(a) Folge $-\nabla d_{\Gamma}$



(b) Folge ?

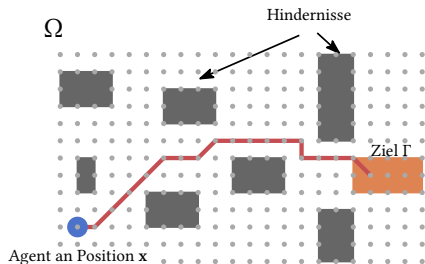
- (1) Dijkstra's Algorithmus
- (2) A^*
- (3) Die Fast Marching Method

Navigation auf dem Gitter (Graphen)

Navigation auf dem Gitter

Sei $\Omega \subset \mathbb{Z}^2$ ein zusammenhängende räumlicher Bereich, $\Gamma \subset \mathbb{Z}^2$ ein Zielgebiet und $\mathbf{u} \in \Omega$ eine bestimmte Position, dann suchen wir nach einer Sequenz von Positionen $\mathbf{u}_0, \dots, \mathbf{u}_m \in \Omega$, sodass

$$\mathbf{u}_0 = \mathbf{u} \text{ und } \mathbf{u}_m \in \Gamma.$$

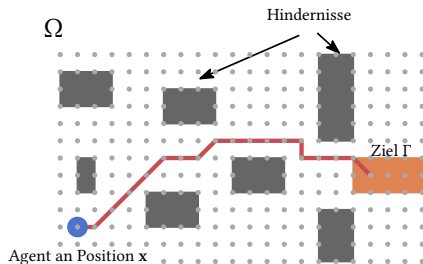


Zudem muss für alle $i > 0$, \mathbf{u}_i zu einer Nachbarschaft von \mathbf{u}_{i-1} gehört (Nebenbedingung).

Navigation auf dem Gitter

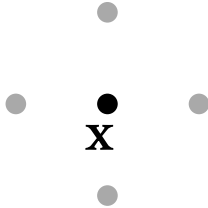
Sei $\Omega \subset \mathbb{Z}^2$ ein zusammenhängende räumlicher Bereich, $\Gamma \subset \mathbb{Z}^2$ ein Zielgebiet und $\mathbf{u} \in \Omega$ eine bestimmte Position, dann suchen wir nach einer Sequenz von Positionen $\mathbf{u}_0, \dots, \mathbf{u}_m \in \Omega$, sodass

$$\mathbf{u}_0 = \mathbf{u} \text{ und } \mathbf{u}_m \in \Gamma.$$

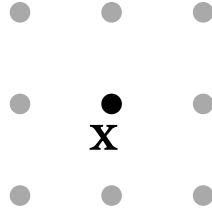


Zudem muss für alle $i > 0$, \mathbf{u}_i zu einer Nachbarschaft von \mathbf{u}_{i-1} gehört (Nebenbedingung).

Navigation auf dem Gitter: Nachbarschaft



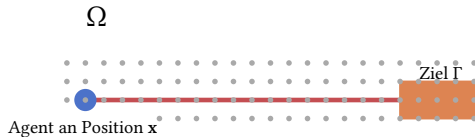
(a) Von Neumann



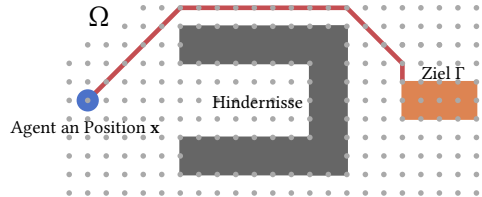
(b) Moore

Navigation auf dem Gitter

Sei $d_T(\mathbf{u})$ die Länge des kürzesten Wegs von \mathbf{u} zum Ziel Γ auf dem Gitter.



(a) Folge dem kürzesten Weg



(b) Folge dem kürzesten Weg

Navigation auf dem Gitter (Graphen): Dijkstra's Algorithmus

Navigation auf dem Gitter: Dijkstra's Algorithmus

Strategie: Berechne kürzeste Wege von Γ zu jedem Gitterpunkt $\mathbf{u} \in \Omega$ durch den DIJKSTRA [3].

Beobachtung: Ist $\mathbf{u}_0, \dots, \mathbf{u}_k, \dots, \mathbf{u}_m$ der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_m , dann ist

$$\mathbf{u}_0, \dots, \mathbf{u}_k$$

der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_k .

Navigation auf dem Gitter: Dijkstra's Algorithmus

Strategie: Berechne kürzeste Wege von Γ zu jedem Gitterpunkt $\mathbf{u} \in \Omega$ durch den DIJKSTRA [3].

Beobachtung: Ist $\mathbf{u}_0, \dots, \mathbf{u}_k, \dots, \mathbf{u}_m$ der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_m , dann ist

$$\mathbf{u}_0, \dots, \mathbf{u}_k$$

der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_k .

Navigation auf dem Gitter: Dijkstra's Algorithmus

Definitionen:

- (i) \mathbf{u}, \mathbf{v} , Knoten des Gitters
- (ii) $d_{\Gamma}(\mathbf{u})$, gesuchte Länge des kürzesten Wegs von \mathbf{u} nach Γ
- (iii) $d(\mathbf{u}, \mathbf{v})$, Länge/Gewicht der Kante (\mathbf{u}, \mathbf{v})
- (iv) Q , eine PRIORITYQUEUE (z.B. FIBONACCIHEAP)

Navigation auf dem Gitter: Dijkstra's Algorithmus

Navigation auf dem Gitter: Dijkstra's Algorithmus

Input : u^* start position

```
1  $d_T(u) \leftarrow 0$  for all  $u \in \Gamma$ ;  
2  $d_T(u) \leftarrow \infty$  for all  $u \notin \Gamma$ ;  
3  $Q \leftarrow \{(u, d_T(u)) \in \Gamma\}$ ;  
4 while  $Q \neq \emptyset$  do  
5      $(u, d_T(u)) \leftarrow Q.POP()$ ;  
6     if  $u = u^*$  then  
7         return  $d_T$ ;  
8     foreach neighbour  $v$  of  $u$  do  
9          $uv \leftarrow d_T(u) + d(u, v)$ ;  
10        if  $uv < d_T(v)$  then  
11             $d_T(v) \leftarrow uv$ ;  
12            if  $(v, d_T(v)) \in Q$  then  
13                 $Q.DECREASE(v, d_T(v))$ ;  
14            else  
15                 $Q.PUSH(v, d_T(v))$ ;  
16 return  $d_T$ ;
```

Q ist eine nach $d_T(v)$ sortierte PRIORITYQUEUE:

- $Q.POP()$, wirft das kleinste Element heraus,
- $Q.DECREASE(v, d_T(v))$ ändert ein Element ab,
- und $Q.PUSH(v, d_T(v))$ fügt ein Element ein.

Komplexität: (einfacher Graph mit positiven Kosten und n Knoten)

- Zeit: $O(n \log(n))$
- Speicher: $O(n)$

Navigation auf dem Gitter: Dijkstra's Algorithmus

Input : u^* start position

```
1  $d_T(u) \leftarrow 0$  for all  $u \in \Gamma$ ;  
2  $d_T(u) \leftarrow \infty$  for all  $u \notin \Gamma$ ;  
3  $Q \leftarrow \{(u, d_T(u)) \in \Gamma\}$ ;  
4 while  $Q \neq \emptyset$  do  
5      $(u, d_T(u)) \leftarrow Q.POP()$ ;  
6     if  $u = u^*$  then  
7         return  $d_T$ ;  
8     foreach neighbour  $v$  of  $u$  do  
9          $uv \leftarrow d_T(u) + d(u, v)$ ;  
10        if  $uv < d_T(v)$  then  
11             $d_T(v) \leftarrow uv$ ;  
12            if  $(v, d_T(v)) \in Q$  then  
13                 $Q.DECREASE(v, d_T(v))$ ;  
14            else  
15                 $Q.PUSH(v, d_T(v))$ ;  
16 return  $d_T$ ;
```

Q ist eine nach $d_T(v)$ sortierte PRIORITYQUEUE:

- $Q.POP()$, wirft das kleinste Element heraus,
- $Q.DECREASE(v, d_T(v))$ ändert ein Element ab,
- und $Q.PUSH(v, d_T(v))$ fügt ein Element ein.

Komplexität: (einfacher Graph mit positiven Kosten und n Knoten)

- Zeit: $O(n \log(n))$
- Speicher: $O(n)$

Navigation auf dem Gitter: Dijkstra's Algorithmus

Strategie: Berechne kürzeste Wege von Γ zu jedem Gitterpunkt $\mathbf{u} \in \Omega$ durch den DIJKSTRA [3].

Beobachtung: Ist $\mathbf{u}_0, \dots, \mathbf{u}_k, \dots, \mathbf{u}_m$ der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_m , dann ist

$$\mathbf{u}_0, \dots, \mathbf{u}_k$$

der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_k .

Invarianz: Für alle $\mathbf{u} \in \Omega$, die aus Q gelöscht werden, ist $d_\Gamma(\mathbf{u})$ die Länge des kürzesten Wegs von \mathbf{u} nach Γ . (Beweis über Induktion)

Navigation auf dem Gitter: Dijkstra's Algorithmus

Strategie: Berechne kürzeste Wege von Γ zu jedem Gitterpunkt $\mathbf{u} \in \Omega$ durch den DIJKSTRA [3].

Beobachtung: Ist $\mathbf{u}_0, \dots, \mathbf{u}_k, \dots, \mathbf{u}_m$ der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_m , dann ist

$$\mathbf{u}_0, \dots, \mathbf{u}_k$$

der kürzeste Weg von \mathbf{u}_0 nach \mathbf{u}_k .

Invarianz: Für alle $\mathbf{u} \in \Omega$, die aus Q gelöscht werden, ist $d_\Gamma(\mathbf{u})$ die Länge des kürzesten Wegs von \mathbf{u} nach Γ . (Beweis über Induktion)

Navigation auf dem Gitter (Graphen): Der A* Algorithmus

Navigation auf dem Gitter: A* Algorithmus

Navigation auf dem Gitter: A* Algorithmus

Strategie: Berechne gerichtet/informiert die kürzesten Wege von Γ zu Gitterpunkten $\mathbf{u} \in \Omega$. Spare dir unnötige Berechnungen (A* [5])

Beobachtung (1): Kein Weg $\mathbf{u}_0, \dots, \mathbf{u}_m$ ist nicht kürzer als der euklidische Abstand zwischen \mathbf{u}_0 und \mathbf{u}_m , d.h.

$$\|\mathbf{u}_0 - \mathbf{u}_m\| \leq \sum_{i=0}^{m-1} d(\mathbf{u}_i, \mathbf{u}_{i+1}),$$

Beobachtung (2): Aus

$$d_{\Gamma}(\mathbf{v}) + \|\mathbf{u} - \mathbf{v}\| > d_{\Gamma}(\mathbf{u})$$

folgt, dass der kürzeste Weg vom Ziel zu \mathbf{u} nicht über \mathbf{v} geht (selbst wenn $d_{\Gamma}(\mathbf{v}) \leq d_{\Gamma}(\mathbf{u})$).

Navigation auf dem Gitter: A* Algorithmus

Strategie: Berechne gerichtet/informiert die kürzesten Wege von Γ zu Gitterpunkten $\mathbf{u} \in \Omega$. Spare dir unnötige Berechnungen (A* [5])

Beobachtung (1): Kein Weg $\mathbf{u}_0, \dots, \mathbf{u}_m$ ist nicht kürzer als der euklidische Abstand zwischen \mathbf{u}_0 und \mathbf{u}_m , d.h.

$$\|\mathbf{u}_0 - \mathbf{u}_m\| \leq \sum_{i=0}^{m-1} d(\mathbf{u}_i, \mathbf{u}_{i+1}),$$

Beobachtung (2): Aus

$$d_{\Gamma}(\mathbf{v}) + \|\mathbf{u} - \mathbf{v}\| > d_{\Gamma}(\mathbf{u})$$

folgt, dass der kürzeste Weg vom Ziel zu \mathbf{u} nicht über \mathbf{v} geht (selbst wenn $d_{\Gamma}(\mathbf{v}) \leq d_{\Gamma}(\mathbf{u})$).

Navigation auf dem Gitter: A* Algorithmus

Strategie: Berechne gerichtet/informiert die kürzesten Wege von Γ zu Gitterpunkten $\mathbf{u} \in \Omega$. Spare dir unnötige Berechnungen (A* [5])

Beobachtung (1): Kein Weg $\mathbf{u}_0, \dots, \mathbf{u}_m$ ist nicht kürzer als der euklidische Abstand zwischen \mathbf{u}_0 und \mathbf{u}_m , d.h.

$$\|\mathbf{u}_0 - \mathbf{u}_m\| \leq \sum_{i=0}^{m-1} d(\mathbf{u}_i, \mathbf{u}_{i+1}),$$

Beobachtung (2): Aus

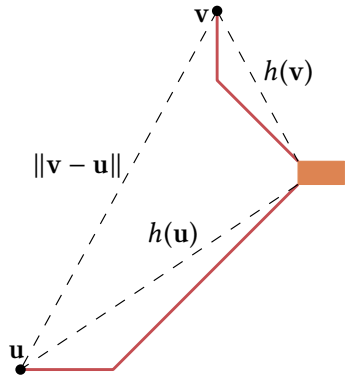
$$d_{\Gamma}(\mathbf{v}) + \|\mathbf{u} - \mathbf{v}\| > d_{\Gamma}(\mathbf{u})$$

folgt, dass der kürzeste Weg vom Ziel zu \mathbf{u} nicht über \mathbf{v} geht (selbst wenn $d_{\Gamma}(\mathbf{v}) \leq d_{\Gamma}(\mathbf{u})$).

Navigation auf dem Gitter: A* Algorithmus

Navigation auf dem Gitter: A* Algorithmus

Navigation auf dem Gitter: A* Algorithmus



Sei

$$h(\mathbf{u}) = \min_{\mathbf{v} \in \Gamma} \|\mathbf{u} - \mathbf{v}\|,$$

dann gilt

$$h(\mathbf{u}) - h(\mathbf{v}) \leq \|\mathbf{v} - \mathbf{u}\|.$$

Beobachtung (2*): Aus

$$d_{\Gamma}(\mathbf{v}) + h(\mathbf{u}) - h(\mathbf{v}) > d_{\Gamma}(\mathbf{u})$$

folgt, dass der kürzeste Weg vom Ziel zu \mathbf{u} nicht über \mathbf{v} geht (selbst wenn $d_{\Gamma}(\mathbf{v}) \leq d_{\Gamma}(\mathbf{u})$).

Navigation auf dem Gitter: A* Algorithmus

Input : u^* start position, h heuristik

```
1  $d_\Gamma(u) \leftarrow 0, \forall u \in \Gamma;$ 
2  $d_\Gamma(u) \leftarrow \infty, \forall u \notin \Gamma;$ 
  /* sortiert nach  $d_\Gamma(u) + h(u)$  */
3  $Q \leftarrow \{(u, d_\Gamma(u)) \in \Gamma\};$ 
4 while  $Q \neq \emptyset$  do
5    $(u, d_\Gamma(u)) \leftarrow Q.POP();$ 
6   if  $u = u^*$  then
7     return  $d_\Gamma;$ 
8   foreach neighbour  $v$  of  $u$  do
9      $uv \leftarrow d_\Gamma(u) + d(u, v);$ 
10    if  $uv < d_\Gamma(v)$  then
11       $d_\Gamma(v) \leftarrow uv;$ 
12      if  $(v, d_\Gamma(v)) \in Q$  then
13         $Q.DECREASE(v, d_\Gamma(v));$ 
14      else
15         $Q.PUSH(v, d_\Gamma(v));$ 
16 return  $d_\Gamma;$ 
```

Q ist eine nach $d_\Gamma(v) + h(v)$ sortierte PRIORITYQUEUE:

- $Q.POP()$, wirft das kleinste Element heraus,
- $Q.DECREASE(v, d_\Gamma(v))$ ändert ein Element ab,
- und $Q.PUSH(v, d_\Gamma(v))$ fügt ein Element ein.

Komplexität: (einfacher Graph mit positiven Kosten und n Knoten)

- Zeit: $O(n \log(n))$
- Speicher: $O(n)$

Navigation auf dem Gitter: A* Algorithmus

Input : u^* start position, h heuristik

```
1  $d_\Gamma(u) \leftarrow 0, \forall u \in \Gamma;$ 
2  $d_\Gamma(u) \leftarrow \infty, \forall u \notin \Gamma;$ 
   /* sortiert nach  $d_\Gamma(u) + h(u)$  */
3  $Q \leftarrow \{(u, d_\Gamma(u)) \in \Gamma\};$ 
4 while  $Q \neq \emptyset$  do
5      $(u, d_\Gamma(u)) \leftarrow Q.POP();$ 
6     if  $u = u^*$  then
7         return  $d_\Gamma;$ 
8     foreach neighbour  $v$  of  $u$  do
9          $uv \leftarrow d_\Gamma(u) + d(u, v);$ 
10        if  $uv < d_\Gamma(v)$  then
11             $d_\Gamma(v) \leftarrow uv;$ 
12            if  $(v, d_\Gamma(v)) \in Q$  then
13                 $Q.DECREASE(v, d_\Gamma(v));$ 
14            else
15                 $Q.PUSH(v, d_\Gamma(v));$ 
16 return  $d_\Gamma;$ 
```

Q ist eine nach $d_\Gamma(v) + h(v)$ sortierte PRIORITYQUEUE:

- $Q.POP()$, wirft das kleinste Element heraus,
- $Q.DECREASE(v, d_\Gamma(v))$ ändert ein Element ab,
- und $Q.PUSH(v, d_\Gamma(v))$ fügt ein Element ein.

Komplexität: (einfacher Graph mit positiven Kosten und n Knoten)

- Zeit: $O(n \log(n))$
- Speicher: $O(n)$

Navigation auf dem Gitter: A* Algorithmus

Heuristik: Es muss nicht der euklidische Abstand als Heuristik genommen werden.
Doch falls

$$h(\mathbf{u}) \leq d_{\Gamma}(\mathbf{u}) \quad (\text{zulässig})$$

$$h(\mathbf{u}) \leq d(\mathbf{u}, \mathbf{v}) + h(\mathbf{v}) \quad (\text{monoton})$$

für jede Knoten \mathbf{u} , bzw. jede Kante (\mathbf{u}, \mathbf{v}) gilt (**Konsistenz**), so findet der A* Algorithmus garantiert den optimalen Pfad ohne Knoten mehrfach zu besuchen.

Verlorener Vorteil: Müssen Sie ohnehin für jeden Knoten den kürzesten Pfad zum Ziel berechnen, bringt der A* keinen Vorteil gegenüber dem DIJKSTRA.

Navigation auf dem Gitter: A* Algorithmus

Heuristik: Es muss nicht der euklidische Abstand als Heuristik genommen werden.
Doch falls

$$h(\mathbf{u}) \leq d_{\Gamma}(\mathbf{u}) \quad (\text{zulässig})$$

$$h(\mathbf{u}) \leq d(\mathbf{u}, \mathbf{v}) + h(\mathbf{v}) \quad (\text{monoton})$$

für jeden Knoten \mathbf{u} , bzw. jede Kante (\mathbf{u}, \mathbf{v}) gilt (**Konsistenz**), so findet der A* Algorithmus garantiert den optimalen Pfad ohne Knoten mehrfach zu besuchen.

Verlorener Vorteil: Müssen Sie ohnehin für jeden Knoten den kürzesten Pfad zum Ziel berechnen, bringt der A* keinen Vorteil gegenüber dem DIJKSTRA.

Navigation auf dem Gitter: A* Algorithmus

Heuristik: Es muss nicht der euklidische Abstand als Heuristik genommen werden.
Doch falls

$$h(\mathbf{u}) \leq d_{\Gamma}(\mathbf{u}) \quad (\text{zulässig})$$

$$h(\mathbf{u}) \leq d(\mathbf{u}, \mathbf{v}) + h(\mathbf{v}) \quad (\text{monoton})$$

für jeden Knoten \mathbf{u} , bzw. jede Kante (\mathbf{u}, \mathbf{v}) gilt (**Konsistenz**), so findet der A* Algorithmus garantiert den optimalen Pfad ohne Knoten mehrfach zu besuchen.

Verlorener Vorteil: Müssen Sie ohnehin für jeden Knoten den kürzesten Pfad zum Ziel berechnen, bringt der A* keinen Vorteil gegenüber dem DIJKSTRA.

Navigation auf dem Gitter: A* Algorithmus

Heuristik: Es muss nicht der euklidische Abstand als Heuristik genommen werden.
Doch falls

$$h(\mathbf{u}) \leq d_{\Gamma}(\mathbf{u}) \quad (\text{zulässig})$$

$$h(\mathbf{u}) \leq d(\mathbf{u}, \mathbf{v}) + h(\mathbf{v}) \quad (\text{monoton})$$

für jeden Knoten \mathbf{u} , bzw. jede Kante (\mathbf{u}, \mathbf{v}) gilt (**Konsistenz**), so findet der A* Algorithmus garantiert den optimalen Pfad ohne Knoten mehrfach zu besuchen.

Verlorener Vorteil: Müssen Sie ohnehin für jeden Knoten den kürzesten Pfad zum Ziel berechnen, bringt der A* keinen Vorteil gegenüber dem DIJKSTRA.

Navigation auf dem Gitter: A* Algorithmus

Heuristik: Es muss nicht der euklidische Abstand als Heuristik genommen werden.
Doch falls

$$h(\mathbf{u}) \leq d_{\Gamma}(\mathbf{u}) \quad (\text{zulässig})$$

$$h(\mathbf{u}) \leq d(\mathbf{u}, \mathbf{v}) + h(\mathbf{v}) \quad (\text{monoton})$$

für jeden Knoten \mathbf{u} , bzw. jede Kante (\mathbf{u}, \mathbf{v}) gilt (**Konsistenz**), so findet der A* Algorithmus garantiert den optimalen Pfad ohne Knoten mehrfach zu besuchen.

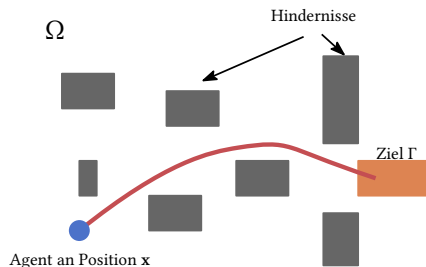
Verlorener Vorteil: Müssen Sie ohnehin für jeden Knoten den kürzesten Pfad zum Ziel berechnen, bringt der A* keinen Vorteil gegenüber dem DIJKSTRA.

Navigation im \mathbb{R}^2

Navigation im \mathbb{R}^2

Sei $\Omega \subset \mathbb{R}^2$ ein zusammenhängender räumlicher Bereich, $\Gamma \subset \mathbb{R}^2$ ein Zielgebiet und $\mathbf{x} \in \Omega$ eine bestimmte Position, dann suchen wir nach einer Sequenz von Positionen $\mathbf{x}_0, \dots, \mathbf{x}_m \in \Omega$, sodass

$$\mathbf{x}_0 = \mathbf{x} \text{ und } \mathbf{x}_m \in \Gamma.$$



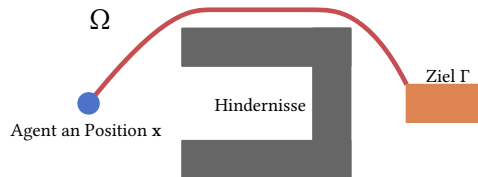
Navigation im \mathbb{R}^2

Sei $d_\Gamma(\mathbf{x}) = \min_{\mathbf{y} \in \Gamma} \|\mathbf{x} - \mathbf{y}\|$ die Euklidische Distanz zwischen \mathbf{x} und dem Ziel Γ .

Falls es keine Hindernisse in dem Gebiet gibt können wir einfach entlang von $-\nabla d_\Gamma$ laufen:



(a) Folge $-\nabla d_\Gamma$



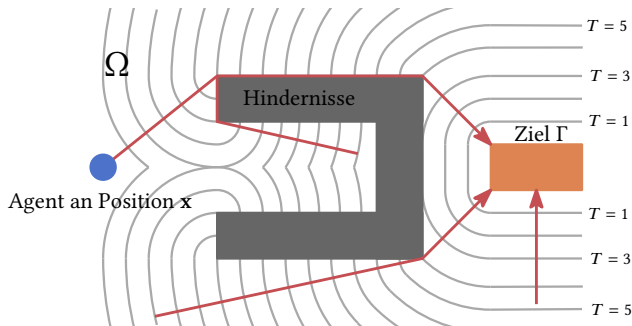
(b) Folge ?

Navigation im \mathbb{R}^2 : Die Eikonalgleichung

Navigation im \mathbb{R}^2

Wir stellen uns eine **Wellenfront** vor, die sich mit der **Reisegeschwindigkeit** $f(\mathbf{x}) = 1$ vom Ziel Γ über das Gebiet Ω ausbreitet.

$T(\mathbf{x})$ ist die **Reisezeit** oder Ankunftszeit der **Wellenfront** am Ort \mathbf{x} .

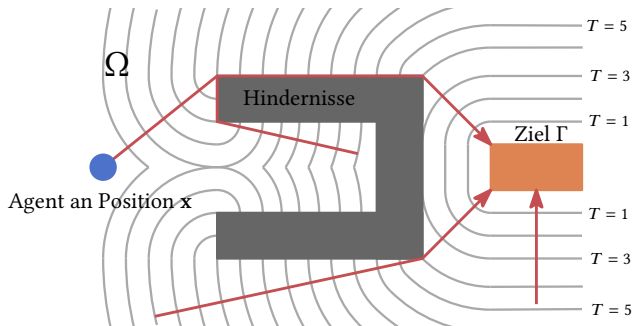


Die Änderung der **Reisezeit** T (über den Ort) ist gleich $1/f$.

Navigation im \mathbb{R}^2

Wir stellen uns eine **Wellenfront** vor, die sich mit der **Reisegeschwindigkeit** $f(\mathbf{x}) = 1$ vom Ziel Γ über das Gebiet Ω ausbreitet.

$T(\mathbf{x})$ ist die **Reisezeit** oder Ankunftszeit der **Wellenfront** am Ort \mathbf{x} .

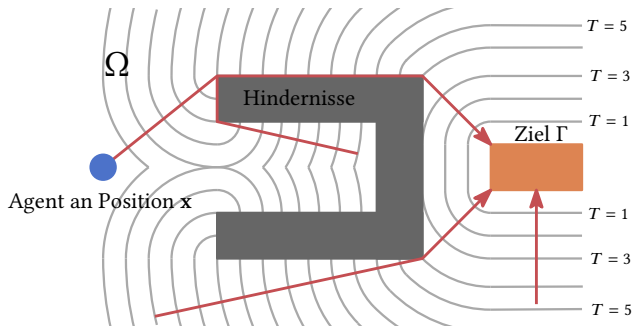


Die Änderung der **Reisezeit** T (über den Ort) ist gleich $1/f$.

Navigation im \mathbb{R}^2

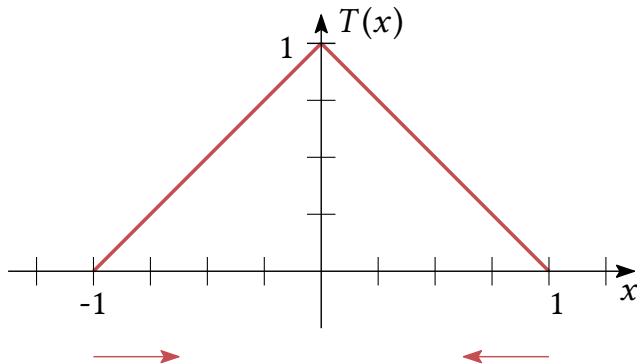
Wir stellen uns eine **Wellenfront** vor, die sich mit der **Reisegeschwindigkeit** $f(\mathbf{x}) = 1$ vom Ziel Γ über das Gebiet Ω ausbreitet.

$T(\mathbf{x})$ ist die **Reisezeit** oder Ankunftszeit der **Wellenfront** am Ort \mathbf{x} .



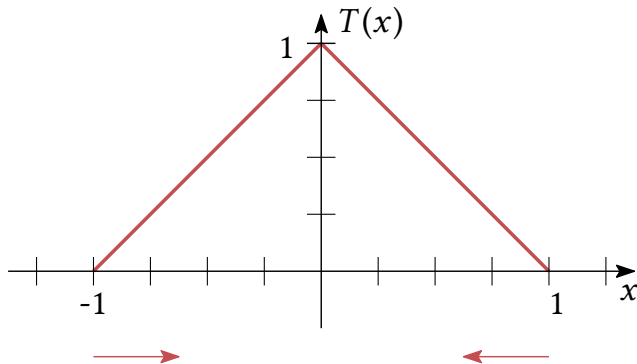
Die Änderung der **Reisezeit** T (über den Ort) ist gleich $1/f$.

Ein eindimensionaler Fall: Sei $\Omega = [-1; 1], \Gamma = \{-1, 1\}$.



$\Rightarrow T(x) = 1 - |x|$, ist die Viskositätslösung der sog. **Eikonalgleichung!**

Ein eindimensionaler Fall: Sei $\Omega = [-1; 1], \Gamma = \{-1, 1\}$.

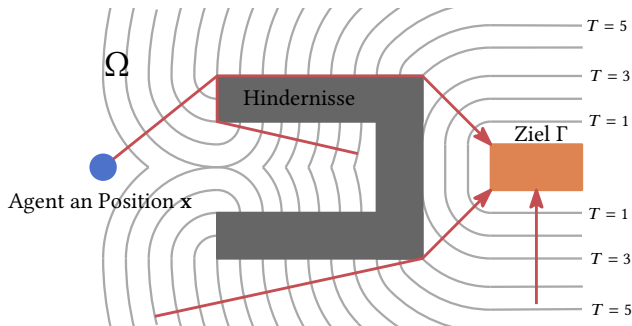


$\Rightarrow T(x) = 1 - |x|$, ist die Viskositätslösung der sog. **Eikonalgleichung!**

Navigation im \mathbb{R}^2

Wir stellen uns eine **Wellenfront** vor, die sich mit der **Reisegeschwindigkeit** $f(\mathbf{x}) = 1$ vom Ziel Γ über das Gebiet Ω ausbreitet.

$T(\mathbf{x})$ ist die **Reisezeit** oder Ankunftszeit der **Wellenfront** am Ort \mathbf{x} .



Die Änderung der **Reisezeit** T (über den Ort) ist gleich $1/f$.

Navigation im \mathbb{R}^2

Die **Wellenfront**, die sich mit der **Reisegeschwindigkeit** $f(\mathbf{x}) = 1$ vom Ziel Γ über das Gebiet Ω ausbreitet, wird von der **Eikonalgleichung** beschrieben:

$$\begin{aligned}\|\nabla T(\mathbf{x})\| \cdot f(\mathbf{x}) &= 1, \mathbf{x} \in \Omega \\ T(\mathbf{x}) &= 0, \mathbf{x} \in \Gamma \\ f(\mathbf{x}) &\geq 0, \mathbf{x} \in \Omega.\end{aligned}\tag{1}$$

Bemerkungen:

- (i) Hyperbolische partielle Differenzialgleichung
- (ii) Randwertproblem ($T = 0$ auf Γ)
- (iii) Für die Viskositätslösung muss T nicht überall differenzierbar sein
- (iv) Gilt $f = 1$, so ist T die **geodätische Distanz**.

Navigation im \mathbb{R}^2

Die **Wellenfront**, die sich mit der **Reisegeschwindigkeit** $f(\mathbf{x}) = 1$ vom Ziel Γ über das Gebiet Ω ausbreitet, wird von der **Eikonalgleichung** beschrieben:

$$\begin{aligned}\|\nabla T(\mathbf{x})\| \cdot f(\mathbf{x}) &= 1, \mathbf{x} \in \Omega \\ T(\mathbf{x}) &= 0, \mathbf{x} \in \Gamma \\ f(\mathbf{x}) &\geq 0, \mathbf{x} \in \Omega.\end{aligned}\tag{1}$$

Bemerkungen:

- (i) Hyperbolische partielle Differenzialgleichung
- (ii) Randwertproblem ($T = 0$ auf Γ)
- (iii) Für die Viskositätslösung muss T nicht überall differenzierbar sein
- (iv) Gilt $f = 1$, so ist T die **geodätische Distanz**.

Die **Wellenfront**, die sich mit der **Reisegeschwindigkeit** $f(\mathbf{x}) = 1$ vom Ziel Γ über das Gebiet Ω ausbreitet, wird von der **Eikonalgleichung** beschrieben:

$$\begin{aligned}\|\nabla T(\mathbf{x})\| \cdot f(\mathbf{x}) &= 1, \mathbf{x} \in \Omega \\ T(\mathbf{x}) &= 0, \mathbf{x} \in \Gamma \\ f(\mathbf{x}) &\geq 0, \mathbf{x} \in \Omega.\end{aligned}\tag{1}$$

Bemerkungen:

- (i) Hyperbolische partielle Differenzialgleichung
- (ii) Randwertproblem ($T = 0$ auf Γ)
- (iii) Für die Viskositätslösung muss T nicht überall differenzierbar sein
- (iv) Gilt $f = 1$, so ist T die **geodätische Distanz**.

Navigation im \mathbb{R}^2 : Die Fast Marching Method (FMM)

Navigation im \mathbb{R}^2 : FMM

Die FASTMARCHINGMETHOD [9, 10] berechnet T auf einer Diskretisierung (hier Gitter) wobei der Algorithmus die **Wellenfront** 'nachahmt'.

Die Methode arbeitet die Punkte in der gleichen Reihenfolge wie der DIJKSTRA ab, jedoch sind die Kosten/Distanz die **Reisezeit** $T(\mathbf{u})$.

Navigation im \mathbb{R}^2 : FMM

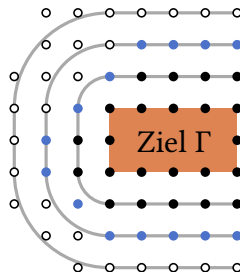
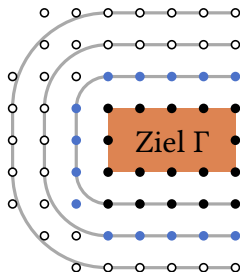
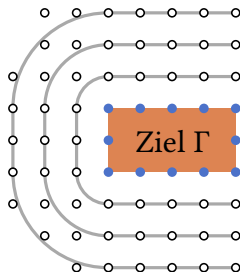
Die FASTMARCHINGMETHOD [9, 10] berechnet T auf einer Diskretisierung (hier Gitter) wobei der Algorithmus die **Wellenfront** 'nachahmt'.

Die Methode arbeitet die Punkte in der gleichen Reihenfolge wie der DIJKSTRA ab, jedoch sind die Kosten/Distanz die **Reisezeit** $T(\mathbf{u})$.

Navigation im \mathbb{R}^2 : FMM

Jeder Gitterpunkt gehört zu genau einer der folgenden Mengen:

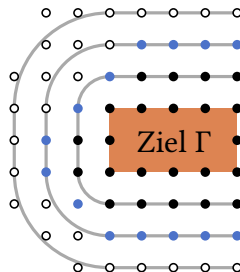
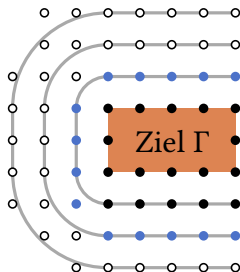
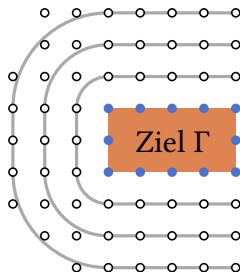
- (i) *unerreicht*: 'Die Wellenfront ist noch nicht angekommen'
- (ii) *erreicht*: 'Die Wellenfront ist gerade hier'
- (iii) *verlassen*: 'Die Wellenfront hat den Punkt verlassen'



Navigation im \mathbb{R}^2 : FMM

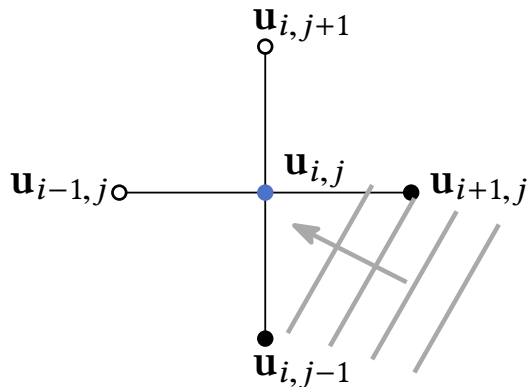
Jeder Gitterpunkt gehört zu genau einer der folgenden Mengen:

- (i) *unerreicht*: 'Die Wellenfront ist noch nicht angekommen'
- (ii) *erreicht*: 'Die Wellenfront ist gerade hier'
- (iii) *verlassen*: 'Die Wellenfront hat den Punkt verlassen'



Navigation im \mathbb{R}^2 : Lokale Lösung

Die **Wellenfront** erreicht jeden Gitterpunkt $\mathbf{u}_{i,j}$ von einer bestimmten Richtung:

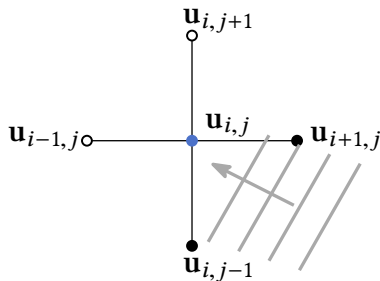


Auf einem Gitter berechnen wir die **Reisezeit** $T(\mathbf{u}_{i,j})$ anhand des Stencils.

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir approximieren ∇T durch finite Differenzen (Taylor-Expansion):

$$\begin{aligned}\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} &\approx D_{i,j}^{\pm x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} \\ \frac{\partial T(\mathbf{u}_{i,j})}{\partial y} &\approx D_{i,j}^{\pm y} \mathbf{u} = \frac{T(\mathbf{u}_{i,j\pm 1}) - T(\mathbf{u}_{i,j})}{\pm \Delta y}.\end{aligned}$$



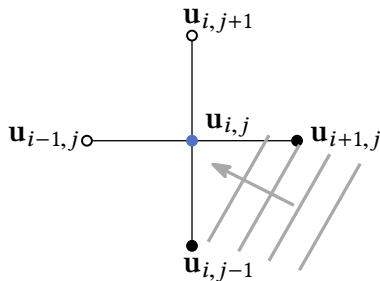
Wüssten wir, dass die Welle von unten rechts kommt bräuchten wir nur

$$D_{i,j}^{+x} \mathbf{u} = \frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \text{ und } D_{i,j}^{-y} \mathbf{u} = \frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y}.$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir approximieren ∇T durch finite Differenzen (Taylor-Expansion):

$$\begin{aligned}\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} &\approx D_{i,j}^{\pm x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} \\ \frac{\partial T(\mathbf{u}_{i,j})}{\partial y} &\approx D_{i,j}^{\pm y} \mathbf{u} = \frac{T(\mathbf{u}_{i,j\pm 1}) - T(\mathbf{u}_{i,j})}{\pm \Delta y}.\end{aligned}$$



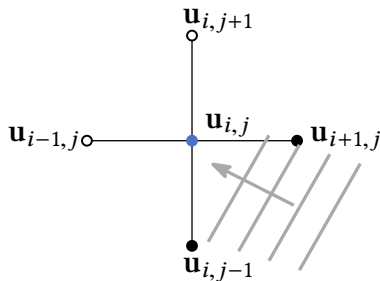
Wüssten wir, dass die Welle von unten rechts kommt bräuchten wir nur

$$D_{i,j}^{+x} \mathbf{u} = \frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \text{ und } D_{i,j}^{-y} \mathbf{u} = \frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y}.$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir approximieren ∇T durch finite Differenzen (Taylor-Expansion):

$$\begin{aligned}\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} &\approx D_{i,j}^{\pm x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} \\ \frac{\partial T(\mathbf{u}_{i,j})}{\partial y} &\approx D_{i,j}^{\pm y} \mathbf{u} = \frac{T(\mathbf{u}_{i,j\pm 1}) - T(\mathbf{u}_{i,j})}{\pm \Delta y}.\end{aligned}$$



Wüssten wir, dass die Welle von unten rechts kommt bräuchten wir nur

$$D_{i,j}^{+x} \mathbf{u} = \frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \text{ und } D_{i,j}^{-y} \mathbf{u} = \frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y}.$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Wie bestimmen wir nun $T(\mathbf{u}_{i,j})$?

$$\|\nabla T(\mathbf{x})\| \cdot f(\mathbf{x}) = 1 \quad (2)$$

wird zu

$$\|\nabla T(\mathbf{x})\|^2 = \frac{1}{f(\mathbf{x})^2}, \quad (3)$$

wird in unserem Beispiel approximiert durch

$$(D_{i,j}^{+x} \mathbf{u})^2 + (D_{i,j}^{-y} \mathbf{u})^2 = f(\mathbf{x}_{i,j})^{-2}. \quad (4)$$

Das heißt wir lösen die quadratische Gleichung

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2} \quad (5)$$

nach $T(\mathbf{u}_{i,j})$.

Navigation im \mathbb{R}^2 : Lokale Lösung

Wie bestimmen wir nun $T(\mathbf{u}_{i,j})$?

$$\|\nabla T(\mathbf{x})\| \cdot f(\mathbf{x}) = 1 \quad (2)$$

wird zu

$$\|\nabla T(\mathbf{x})\|^2 = \frac{1}{f(\mathbf{x})^2}, \quad (3)$$

wird in unserem Beispiel approximiert durch

$$(D_{i,j}^{+x} \mathbf{u})^2 + (D_{i,j}^{-y} \mathbf{u})^2 = f(\mathbf{x}_{i,j})^{-2}. \quad (4)$$

Das heißt wir lösen die quadratische Gleichung

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2} \quad (5)$$

nach $T(\mathbf{u}_{i,j})$.

Navigation im \mathbb{R}^2 : Lokale Lösung

Wie bestimmen wir nun $T(\mathbf{u}_{i,j})$?

$$\|\nabla T(\mathbf{x})\| \cdot f(\mathbf{x}) = 1 \quad (2)$$

wird zu

$$\|\nabla T(\mathbf{x})\|^2 = \frac{1}{f(\mathbf{x})^2}, \quad (3)$$

wird in unserem Beispiel approximiert durch

$$(D_{i,j}^{+x} \mathbf{u})^2 + (D_{i,j}^{-y} \mathbf{u})^2 = f(\mathbf{x}_{i,j})^{-2}. \quad (4)$$

Das heißt wir lösen die quadratische Gleichung

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2} \quad (5)$$

nach $T(\mathbf{u}_{i,j})$.

Navigation im \mathbb{R}^2 : Lokale Lösung

Wie bestimmen wir nun $T(\mathbf{u}_{i,j})$?

$$\|\nabla T(\mathbf{x})\| \cdot f(\mathbf{x}) = 1 \quad (2)$$

wird zu

$$\|\nabla T(\mathbf{x})\|^2 = \frac{1}{f(\mathbf{x})^2}, \quad (3)$$

wird in unserem Beispiel approximiert durch

$$(D_{i,j}^{+x}\mathbf{u})^2 + (D_{i,j}^{-y}\mathbf{u})^2 = f(\mathbf{x}_{i,j})^{-2}. \quad (4)$$

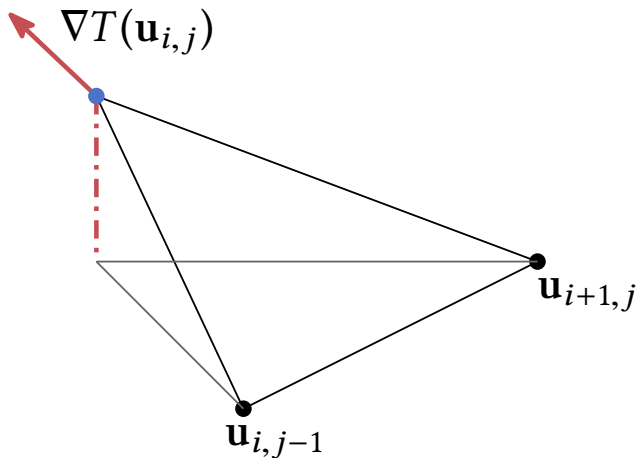
Das heißt wir lösen die quadratische Gleichung

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2} \quad (5)$$

nach $T(\mathbf{u}_{i,j})$.

Navigation im \mathbb{R}^2 : Lokale Lösung

Der wesentliche Unterschied zum DIJKSTRA ist die Berechnung der **Reisezeit** $T(\mathbf{u})$.



Navigation im \mathbb{R}^2 : Lokale Lösung

Im allgemeinen kennen wir die Richtung aus der die **Wellenfront** kommt nicht! Sie kommt **entweder** von **oben** oder **unter** UND von **entweder links** oder **rechts**.

Unten, rechts:

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Oben, links:

$$\left(\frac{T(\mathbf{u}_{i-1,j}) - T(\mathbf{u}_{i,j})}{-\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j+1}) - T(\mathbf{u}_{i,j})}{+\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Navigation im \mathbb{R}^2 : Lokale Lösung

Im allgemeinen kennen wir die Richtung aus der die **Wellenfront** kommt nicht! Sie kommt **entweder** von **oben** oder **unter** UND von **entweder links** oder **rechts**.

Unten, rechts:

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Oben, links:

$$\left(\frac{T(\mathbf{u}_{i-1,j}) - T(\mathbf{u}_{i,j})}{-\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j+1}) - T(\mathbf{u}_{i,j})}{+\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Navigation im \mathbb{R}^2 : Lokale Lösung

Im allgemeinen kennen wir die Richtung aus der die **Wellenfront** kommt nicht! Sie kommt **entweder** von **oben** oder **unter** UND von **entweder links** oder **rechts**.

Unten, rechts:

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Oben, links:

$$\left(\frac{T(\mathbf{u}_{i-1,j}) - T(\mathbf{u}_{i,j})}{-\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j+1}) - T(\mathbf{u}_{i,j})}{+\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Navigation im \mathbb{R}^2 : Lokale Lösung

Im allgemeinen kennen wir die Richtung aus der die **Wellenfront** kommt nicht! Sie kommt **entweder** von **oben** oder **unter** UND von **entweder links** oder **rechts**.

Unten, rechts:

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Oben, links:

$$\left(\frac{T(\mathbf{u}_{i-1,j}) - T(\mathbf{u}_{i,j})}{-\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j+1}) - T(\mathbf{u}_{i,j})}{+\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Navigation im \mathbb{R}^2 : Lokale Lösung

Im allgemeinen kennen wir die Richtung aus der die **Wellenfront** kommt nicht! Sie kommt **entweder** von **oben** oder **unter** UND von **entweder links** oder **rechts**.

Unten, rechts:

$$\left(\frac{T(\mathbf{u}_{i+1,j}) - T(\mathbf{u}_{i,j})}{+\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j-1}) - T(\mathbf{u}_{i,j})}{-\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Oben, links:

$$\left(\frac{T(\mathbf{u}_{i-1,j}) - T(\mathbf{u}_{i,j})}{-\Delta x} \right)^2 + \left(\frac{T(\mathbf{u}_{i,j+1}) - T(\mathbf{u}_{i,j})}{+\Delta y} \right)^2 = f(\mathbf{x}_{i,j})^{-2}$$

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Das heißt aus

$$(D_{i,j}^{+x}\mathbf{u})^2 + (D_{i,j}^{-y}\mathbf{u})^2 = f(\mathbf{x}_{i,j})^{-2}$$

wird im Allgemeinen

$$\max \left\{ D_{i,j}^{-x}\mathbf{u}, -D_{i,j}^{+x}\mathbf{u} \right\}^2 + \max \left\{ D_{i,j}^{-y}\mathbf{u}, -D_{i,j}^{+y}\mathbf{u} \right\}^2 = f(\mathbf{u}_{i,j})^{-2}. \quad (6)$$

Wir lösen die Gleichung lokal durch Godunov's Schemata [9, 11].

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Das heißt aus

$$(D_{i,j}^{+x}\mathbf{u})^2 + (D_{i,j}^{-y}\mathbf{u})^2 = f(\mathbf{x}_{i,j})^{-2}$$

wird im Allgemeinen

$$\max \left\{ D_{i,j}^{-x}\mathbf{u}, -D_{i,j}^{+x}\mathbf{u} \right\}^2 + \max \left\{ D_{i,j}^{-y}\mathbf{u}, -D_{i,j}^{+y}\mathbf{u} \right\}^2 = f(\mathbf{u}_{i,j})^{-2}. \quad (6)$$

Wir lösen die Gleichung lokal durch Godunov's Schemata [9, 11].

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir gehen davon aus, dass die **Wellenfront** von der Richtung kommt von der sie auch früher bei $\mathbf{u}_{i,j}$ eintrifft.

Das heißt aus

$$(D_{i,j}^{+x}\mathbf{u})^2 + (D_{i,j}^{-y}\mathbf{u})^2 = f(\mathbf{x}_{i,j})^{-2}$$

wird im Allgemeinen

$$\max \left\{ D_{i,j}^{-x}\mathbf{u}, -D_{i,j}^{+x}\mathbf{u} \right\}^2 + \max \left\{ D_{i,j}^{-y}\mathbf{u}, -D_{i,j}^{+y}\mathbf{u} \right\}^2 = f(\mathbf{u}_{i,j})^{-2}. \quad (6)$$

Wir lösen die Gleichung lokal durch Godunov's Schemata [9, 11].

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir können ∇T durch weitere Taylor-Terme besser approximieren. Zur Erinnerung:

$$f(x+h) \approx f(x) + hf'(x) + h^2 \frac{f''(x)}{2} \Rightarrow f'(x) \approx \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) \quad (7)$$

Für die Approximation der Ableitung von

$$D_{i,j}^{\pm x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} \quad (8)$$

ergibt sich

$$(D_{i,j}^{\pm x})' \mathbf{u} \approx \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}. \quad (9)$$

Somit ist

$$\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} \approx D_{i,j}^{\pm 2x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} - \frac{\Delta x}{2} \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir können ∇T durch weitere Taylor-Terme besser approximieren. Zur Erinnerung:

$$f(x+h) \approx f(x) + hf'(x) + h^2 \frac{f''(x)}{2} \Rightarrow f'(x) \approx \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) \quad (7)$$

Für die Approximation der Ableitung von

$$D_{i,j}^{\pm x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} \quad (8)$$

ergibt sich

$$(D_{i,j}^{\pm x})' \mathbf{u} \approx \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}. \quad (9)$$

Somit ist

$$\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} \approx D_{i,j}^{\pm 2x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} - \frac{\Delta x}{2} \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir können ∇T durch weitere Taylor-Terme besser approximieren. Zur Erinnerung:

$$f(x+h) \approx f(x) + hf'(x) + h^2 \frac{f''(x)}{2} \Rightarrow f'(x) \approx \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) \quad (7)$$

Für die Approximation der Ableitung von

$$D_{i,j}^{\pm x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} \quad (8)$$

ergibt sich

$$(D_{i,j}^{\pm x})' \mathbf{u} \approx \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}. \quad (9)$$

Somit ist

$$\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} \approx D_{i,j}^{\pm 2x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} - \frac{\Delta x}{2} \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir können ∇T durch weitere Taylor-Terme besser approximieren. Zur Erinnerung:

$$f(x+h) \approx f(x) + hf'(x) + h^2 \frac{f''(x)}{2} \Rightarrow f'(x) \approx \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x) \quad (7)$$

Für die Approximation der Ableitung von

$$D_{i,j}^{\pm x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} \quad (8)$$

ergibt sich

$$(D_{i,j}^{\pm x})' \mathbf{u} \approx \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}. \quad (9)$$

Somit ist

$$\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} \approx D_{i,j}^{\pm 2x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} - \frac{\Delta x}{2} \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2}$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Somit ist

$$\begin{aligned}\frac{\partial T(\mathbf{u}_{i,j})}{\partial x} &\approx D_{i,j}^{\pm 2x} \mathbf{u} = \frac{T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm \Delta x} - \frac{\Delta x}{2} \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm (\Delta x)^2} \\ &= \frac{2T(\mathbf{u}_{i\pm 1,j}) - T(\mathbf{u}_{i,j})}{\pm 2\Delta x} - \frac{T(\mathbf{u}_{i\pm 2,j}) - 2T(\mathbf{u}_{i\pm 1,j}) + T(\mathbf{u}_{i,j})}{\pm 2\Delta x} \\ &= \frac{-T(\mathbf{u}_{i\pm 2,j}) + 4T(\mathbf{u}_{i\pm 1,j}) - 3T(\mathbf{u}_{i,j})}{\pm 2\Delta x}.\end{aligned}$$

Und in y -Richtung ebenso:

$$\frac{\partial T(\mathbf{u}_{i,j})}{\partial y} \approx D_{i,j}^{\pm 2y} \mathbf{u} = \frac{-T(\mathbf{u}_{i,j\pm 2}) + 4T(\mathbf{u}_{i,j\pm 1}) - 3T(\mathbf{u}_{i,j})}{\pm 2\Delta y}.$$

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir lösen noch immer eine quadratische Gleichung:

$$\max \left\{ D_{i,j}^{-2x} \mathbf{u}, -D_{i,j}^{+2x} \mathbf{u} \right\}^2 + \max \left\{ D_{i,j}^{-2y} \mathbf{u}, -D_{i,j}^{+2y} \mathbf{u} \right\}^2 = f(\mathbf{u}_{i,j})^{-2}. \quad (10)$$

Vorteil: Bessere Konvergenzrate für feiner werdendes Gitter ($\Delta x, \Delta y \rightarrow 0$), denn

$$f(x+h) = f(x) + hf'(x) + \mathcal{O}(h^2) = f(x) + hf'(x) + h^2 \frac{f''(x)}{2} + \mathcal{O}(h^3)$$

Nachteil: Möglicherweise ungewollte Glättung der Singularitäten.

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir lösen noch immer eine quadratische Gleichung:

$$\max \left\{ D_{i,j}^{-2x} \mathbf{u}, -D_{i,j}^{+2x} \mathbf{u} \right\}^2 + \max \left\{ D_{i,j}^{-2y} \mathbf{u}, -D_{i,j}^{+2y} \mathbf{u} \right\}^2 = f(\mathbf{u}_{i,j})^{-2}. \quad (10)$$

Vorteil: Bessere Konvergenzrate für feiner werdendes Gitter ($\Delta x, \Delta y \rightarrow 0$), denn

$$f(x+h) = f(x) + hf'(x) + \mathcal{O}(h^2) = f(x) + hf'(x) + h^2 \frac{f''(x)}{2} + \mathcal{O}(h^3)$$

Nachteil: Möglicherweise ungewollte Glättung der Singularitäten.

Navigation im \mathbb{R}^2 : Lokale Lösung

Wir lösen noch immer eine quadratische Gleichung:

$$\max \left\{ D_{i,j}^{-2x} \mathbf{u}, -D_{i,j}^{+2x} \mathbf{u} \right\}^2 + \max \left\{ D_{i,j}^{-2y} \mathbf{u}, -D_{i,j}^{+2y} \mathbf{u} \right\}^2 = f(\mathbf{u}_{i,j})^{-2}. \quad (10)$$

Vorteil: Bessere Konvergenzrate für feiner werdendes Gitter ($\Delta x, \Delta y \rightarrow 0$), denn

$$f(x+h) = f(x) + hf'(x) + \mathcal{O}(h^2) = f(x) + hf'(x) + h^2 \frac{f''(x)}{2} + \mathcal{O}(h^3)$$

Nachteil: Möglicherweise ungewollte Glättung der Singularitäten.

Navigation im \mathbb{R}^2 : FMM

```
1  $T(\mathbf{u}) \leftarrow 0$  for all  $\mathbf{u} \in \Gamma$ ;  
2  $T(\mathbf{u}) \leftarrow \infty$  for all  $\mathbf{u} \notin \Gamma$ ;  
3  $\mathcal{B} \leftarrow \emptyset$  // von Welle bereits verlassene Punkte  
4  $Q \leftarrow \{(\mathbf{u}, T(\mathbf{u})) \mid \mathbf{u} \in \Gamma\}$  // von der Welle erreichte Punkte  
5 while  $Q \neq \emptyset$  do  
6      $(\mathbf{u}, T(\mathbf{u})) \leftarrow Q.\text{POP}()$ ;  
7     foreach neighbor  $\mathbf{v}$  of  $\mathbf{u}$  with  $\mathbf{v} \notin \mathcal{B}$  do  
8          $T(\mathbf{v}) \leftarrow \text{SOLVEEIKONAL}(\mathbf{v})$ ;  
9         if  $(\mathbf{v}, T(\mathbf{v})) \in Q$  then  
10              $Q.\text{DECREASE}(\mathbf{v}, T(\mathbf{v}))$ ;  
11         else  
12              $Q.\text{PUSH}(\mathbf{v}, T(\mathbf{v}))$ ;  
13      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{u}\}$ ;  
14 return  $T$ ;
```

Navigation im \mathbb{R}^2 : FMM

Q ist eine nach $T(\mathbf{u})$ sortierte PRIORITYQUEUE:

- $Q.\text{POP}()$, wirft das kleinste Element heraus,
- $Q.\text{DECREASE}(\mathbf{u}, T(\mathbf{u}))$ ändert ein Element ab,
- und $Q.\text{PUSH}(\mathbf{u}, T(\mathbf{u}))$ fügt ein Element ein.
- $\text{SOLVEEIKONAL}(\mathbf{u})$ berechnet die lokale Lösung aus Gleichung (10) oder Gleichung (6).

Komplexität: (für n Knoten)

- Zeit: $O(n \log(n))$
- Speicher: $O(n)$

Navigation im \mathbb{R}^2 : FMM

Q ist eine nach $T(\mathbf{u})$ sortierte PRIORITYQUEUE:

- $Q.\text{POP}()$, wirft das kleinste Element heraus,
- $Q.\text{DECREASE}(\mathbf{u}, T(\mathbf{u}))$ ändert ein Element ab,
- und $Q.\text{PUSH}(\mathbf{u}, T(\mathbf{u}))$ fügt ein Element ein.
- $\text{SOLVEEIKONAL}(\mathbf{u})$ berechnet die lokale Lösung aus Gleichung (10) oder Gleichung (6).

Komplexität: (für n Knoten)

- Zeit: $O(n \log(n))$
- Speicher: $O(n)$

Implementierungstipps:

- Sie finden eine sehr gute Beschreibung in [1].
- Um schnell zu prüfen ob ein Gitterpunkt *unerreicht*, *erreicht*, oder *verlassen* ist verwenden Sie ein **flag** (keine Mengen).
- Sie können auch Punkte um Γ herum mit z.B. dem Wert der Euklidischen Distanz initialisieren und in Q einfügen.
- Sie können etwas Performance rausholen, wenn Sie auf DECREASE verzichten und nur PUSH verwenden (d.h. Sie lassen doppelte Einträge zu \Rightarrow Sie müssen den Algorithmus etwas anpassen, siehe [8])

Implementierungstipps:

- Sie finden eine sehr gute Beschreibung in [1].
- Um schnell zu prüfen ob ein Gitterpunkt *unerreicht*, *erreicht*, oder *verlassen* ist verwenden Sie ein **flag** (keine Mengen).
- Sie können auch Punkte um Γ herum mit z.B. dem Wert der Euklidischen Distanz initialisieren und in Q einfügen.
- Sie können etwas Performance rausholen, wenn Sie auf DECREASE verzichten und nur PUSH verwenden (d.h. Sie lassen doppelte Einträge zu \Rightarrow Sie müssen den Algorithmus etwas anpassen, siehe [8])

Implementierungstipps:

- Sie finden eine sehr gute Beschreibung in [1].
- Um schnell zu prüfen ob ein Gitterpunkt *unerreicht*, *erreicht*, oder *verlassen* ist verwenden Sie ein **flag** (keine Mengen).
- Sie können auch Punkte um Γ herum mit z.B. dem Wert der Euklidischen Distanz initialisieren und in Q einfügen.
- Sie können etwas Performance rausholen, wenn Sie auf DECREASE verzichten und nur PUSH verwenden (d.h. Sie lassen doppelte Einträge zu \Rightarrow Sie müssen den Algorithmus etwas anpassen, siehe [8])

Implementierungstipps:

- Sie finden eine sehr gute Beschreibung in [1].
- Um schnell zu prüfen ob ein Gitterpunkt *unerreicht*, *erreicht*, oder *verlassen* ist verwenden Sie ein **flag** (keine Mengen).
- Sie können auch Punkte um Γ herum mit z.B. dem Wert der Euklidischen Distanz initialisieren und in Q einfügen.
- Sie können etwas Performance rausholen, wenn Sie auf DECREASE verzichten und nur PUSH verwenden (d.h. Sie lassen doppelte Einträge zu \Rightarrow Sie müssen den Algorithmus etwas anpassen, siehe [8])

Implementierungstipps:

- Sie finden eine sehr gute Beschreibung in [1].
- Um schnell zu prüfen ob ein Gitterpunkt *unerreicht*, *erreicht*, oder *verlassen* ist verwenden Sie ein **flag** (keine Mengen).
- Sie können auch Punkte um Γ herum mit z.B. dem Wert der Euklidischen Distanz initialisieren und in Q einfügen.
- Sie können etwas Performance rausholen, wenn Sie auf `DECREASE` verzichten und nur `PUSH` verwenden (d.h. Sie lassen doppelte Einträge zu \Rightarrow Sie müssen den Algorithmus etwas anpassen, siehe [8])

Navigation im \mathbb{R}^2 : FMM

Eigenschaften:

- + Für alle arten von Wellen ein sehr schneller sequenzieller Algorithmus
- + Besonders schnell für 'schmale' Wellen
- Schwer zu parallelisieren, Versuche [6, 12]
- Benötigt komplizierte/unstrukturierte PRIORITYQUEUE
- Nutzt die Parallelität der Wellenfrontausbreitung nicht aus

Alternative Methoden:

- FASTSWEEPINGMETHOD [13], nur für sehr 'einfache' Wellen geeignet
- FASTITERATIVEMETHOD [7], besonders für 'breite' Wellen geeignet
- INFORMEDFASTITERATIVEMETHOD (meine Dissertation), für wiederholte Berechnungen sich leicht ändernder Wellen geeignet.

In [2, 4] finden Sie Vergleiche verschiedener Methoden.

Navigation im \mathbb{R}^2 : FMM

Eigenschaften:

- + Für alle arten von Wellen ein sehr schneller sequenzieller Algorithmus
- + Besonders schnell für 'schmale' Wellen
- Schwer zu parallelisieren, Versuche [6, 12]
- Benötigt komplizierte/unstrukturierte PRIORITYQUEUE
- Nutzt die Parallelität der Wellenfrontausbreitung nicht aus

Alternative Methoden:

- FASTSWEEPINGMETHOD [13], nur für sehr 'einfache' Wellen geeignet
- FASTITERATIVEMETHOD [7], besonders für 'breite' Wellen geeignet
- INFORMEDFASTITERATIVEMETHOD (meine Dissertation), für wiederholte Berechnungen sich leicht ändernder Wellen geeignet.

In [2, 4] finden Sie Vergleiche verschiedener Methoden.

Eigenschaften:

- + Für alle arten von Wellen ein sehr schneller sequenzieller Algorithmus
- + Besonders schnell für 'schmale' Wellen
- Schwer zu parallelisieren, Versuche [6, 12]
- Benötigt komplizierte/unstrukturierte PRIORITYQUEUE
- Nutzt die Parallelität der Wellenfrontausbreitung nicht aus

Alternative Methoden:

- FASTSWEEPINGMETHOD [13], nur für sehr 'einfache' Wellen geeignet
- FASTITERATIVEMETHOD [7], besonders für 'breite' Wellen geeignet
- INFORMEDFASTITERATIVEMETHOD (meine Dissertation), für wiederholte Berechnungen sich leicht ändernder Wellen geeignet.

In [2, 4] finden Sie Vergleiche verschiedener Methoden.

Eigenschaften:

- + Für alle arten von Wellen ein sehr schneller sequenzieller Algorithmus
- + Besonders schnell für 'schmale' Wellen
- Schwer zu parallelisieren, Versuche [6, 12]
- Benötigt komplizierte/unstrukturierte PRIORITYQUEUE
- Nutzt die Parallelität der Wellenfrontausbreitung nicht aus

Alternative Methoden:

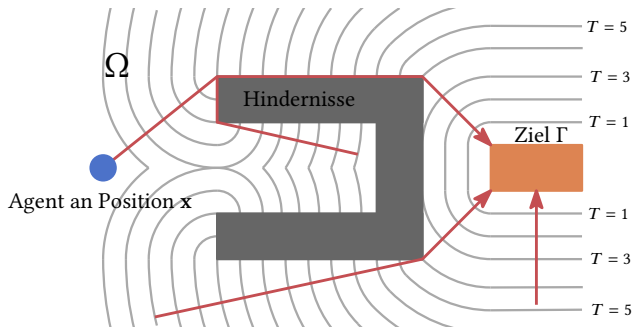
- FASTSWEEPINGMETHOD [13], nur für sehr 'einfache' Wellen geeignet
- FASTITERATIVEMETHOD [7], besonders für 'breite' Wellen geeignet
- INFORMEDFASTITERATIVEMETHOD (meine Dissertation), für wiederholte Berechnungen sich leicht ändernder Wellen geeignet.

In [2, 4] finden Sie Vergleiche verschiedener Methoden.

Navigation im \mathbb{R}^2 : Modellierung

Navigation im \mathbb{R}^2 : Modellierung

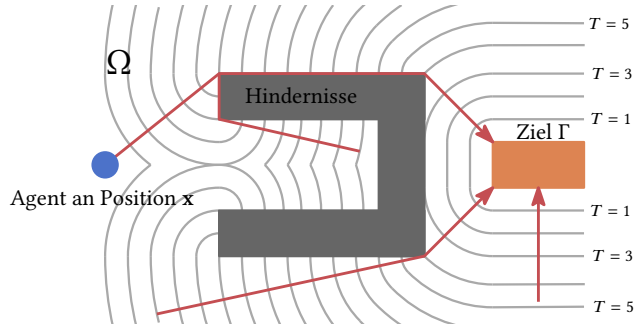
Wie erreichen wir, dass Agenten nicht direkt an den Wänden entlang laufen?



Tipp: Verringern Sie die Reisegeschwindigkeit der Welle f in der Nähe von Hindernissen!

Navigation im \mathbb{R}^2 : Modellierung

Wie erreichen wir, dass Agenten nicht direkt an den Wänden entlang laufen?



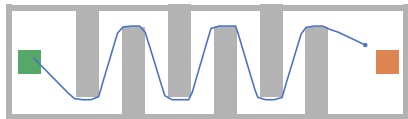
Tipp: Verringern Sie die Reisegeschwindigkeit der Welle f in der Nähe von Hindernissen!

Navigation im \mathbb{R}^2 : Modellierung

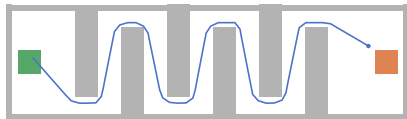
Zum Beispiel, sei $d_W(\mathbf{x})$ die Euklidische Distanz zum nächstliegenden Hindernis/Wand, dann könnte sich

$$f(\mathbf{x}) = \begin{cases} 1/(2 - (d_W(\mathbf{x})/\delta_W)) & \text{falls } d_W(\mathbf{x}) < \delta_W \\ 1 & \text{sonst.} \end{cases}$$

eignen.



(a) $\delta_W = 0.2$ Meter

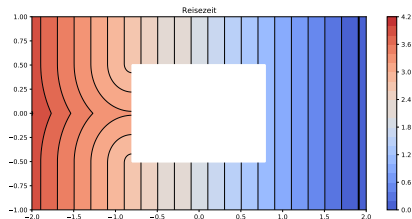


(b) $\delta_W = 0.5$ Meter

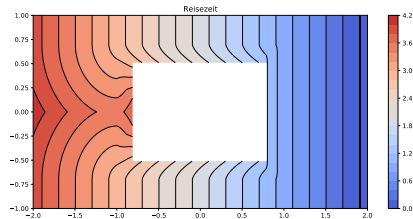


(c) $\delta_W = 1.0$ Meter

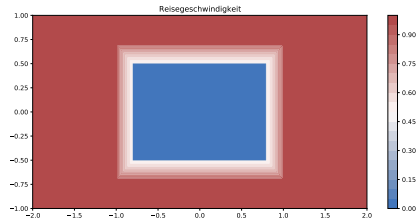
Navigation im \mathbb{R}^2 : Modellierung



(a) $T(\mathbf{x})$ für $f(\mathbf{x}) = 1$



(b) $T(\mathbf{x})$ für $f(\mathbf{x}) \leq 1$



(c) $f(\mathbf{x}) \leq 1$

References I

- [1] J. Andreas Bærentzen. On the implementation of fast marching methods for 3d lattices. Technical report, Technical University of Denmark, 2001.
- [2] A. Capozzoli, C. Curcio, A. Liseno, and S. Savarese. A comparison of fast marching, fast sweeping and fast iterative methods for the solution of the eikonal equation. In *2013 21st Telecommunications Forum Telfor (TELFOR)*, pages 685–688, 2013.
- [3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. doi: 10.1007/BF01386390.
- [4] Javier V. Gómez, David Álvarez, Santiago Garrido, and Luis Moreno. Fast methods for eikonal equations: an experimental survey. *CoRR*, abs/1506.03771, 2015.
- [5] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.
- [6] M. Herrmann. A domain decomposition parallelization of the fast marching method. Technical report, Stanford Univ. Center for Turbulence Research; Stanford, CA, United States, 2003.
- [7] Won-Ki Jeong and Ross T. Whitaker. A fast iterative method for eikonal equations. *SIAM Journal of Scientific Computing*, 30(5):2512–2534, 2008. doi: 10.1137/060670298.
- [8] M. W. Jones, J. A. Baerentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599, July 2006. ISSN 1077-2626. doi: 10.1109/TVCG.2006.56.

References II

- [9] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences of the United States of America*, 95(15):8431–8435, 1998. doi: 10.1073/pnas.95.15.8431.
- [10] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, Cambridge, 1999.
- [11] J. A. Sethian and A. Vladimirsky. Fast methods for the eikonal and related Hamilton-Jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences*, 97(11):5699–5703, 2000. doi: 10.1073/pnas.090060097.
- [12] J. Yang and F. Stern. A highly scalable massively parallel fast marching method for the eikonal equation. *ArXiv e-prints*, feb 2015.
- [13] Hongkai Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74(250):603–627, 2005. doi: 10.1090/S0025-5718-04-01678-3.