Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

Martin von Majowski

Bewachen von Türen

11. Juli 2007

Seminararbeit im SS 2007

Zusammenfassung

In einem fensterlosen Raum mit einer Tür sollen Eindringlinge lokalisiert werden. Dazu werden beginnend bei der Tür entweder zwei Wächter losgeschickt, die sich entlang der Wände unter Erhaltung der gegenseitigen Sichtbarkeit in entgegengesetzte Richtungen bewegen und einen Eindringling erkennen, sobald er ihre Sichtlinie kreuzt oder ein Wächter, dem eine Taschenlampe zur Verfügung steht, und der Eindringlinge erkennt, wenn sie von seiner Taschenlampe angeleuchtet werden. Ziel ist es, alle Eindringlinge in dem Raum aufzuspüren, insbesondere soll sichergestellt sein, dass kein Eindringling während der Suche durch die Tür flüchten kann. Der Erfolg solch einer Suche hängt nicht zuletzt davon ab, an welcher Stelle sich die Tür in dem zu durchsuchenden Raum befindet.

Inhaltsverzeichnis

1 Einführung **3**
1.1 Zwei Wächter 4
1.2 Boundary 1-Searcher 4

2 Vorbereitendes **5**
2.1 Allgemeine Notationen 5
2.2 Sichtbarkeit 6
2.3 V-Diagramme 6
2.4 Sprünge (Boundary 1-Searcher) 10

3 Die Suche mit zwei Wächtern **12**
3.1 Durchsuchbarkeit 12
3.2 Test auf Sicherheit einer Tür 14
3.3 Ermittlung aller sicheren Türen 19
3.3.1 Vorbereitendes zum Algorithmus 20
3.3.2 Algorithmus 25

4 Zwei Wächter vs Boundary 1-Searcher **27**

5 Abschließende Bemerkungen **29**

1 Einführung

Die Ausarbeitung wird sich mit der Frage beschäftigen, an welcher Stelle man für einen gegebenen fensterlosen Raum die Tür platzieren sollte, um alle Eindringline innerhalb des Raums ausfindig machen zu können. Zur Lokalisierung der Eindringlinge werden wir zwei mögliche Strategien betrachten. Die eine ist die Suche mit Hilfe von zwei Wächtern, die andere die Suche mit dem so genannten Boundary 1-Searcher.

Der Raum wird dabei durch ein einfaches Polygon P in der Ebene repräsentiert, während die Tür durch einen Punkt d auf dem Rand des Polygons repräsentiert wird. Es wird ein Algorithmus vorgestellt, der alle Punkte auf dem Rand des Polygons ermittelt, die als Türen für die erfolgreiche Lokalisierung von Eindringlingen in Frage kommen.

Einleitend werden zunächst die beiden erwähnten Strategien etwas genauer beschrieben. In Kapitel 2 folgen Definitionen diverser grundlegender Begriffe, die im weiteren Verlauf von Bedeutung sein werden. Kapitel 3 wird sich intensiv der Suche durch zwei Wächter wirdmen. Es wird ein Verfahren vorgestellt, um eine gegebene Tür auf „Sicherheit“ zu testen und daraufhin ein Algorithmus vorgestellt, der all solche Türen bestimmt. Im vierten Kapitel werden die zwei Wächter kurz dem Boundary 1-Searcher gegenübergestellt. In Kapitel 5 werden die Beobachtungen und Ergebnisse noch einmal zusammengefasst.

1.1 Zwei Wächter

Zwei Wächter durchsuchen einen Raum nach Eindringlingen, indem sie sich entlang der Wände beginnend bei der Tür in entgegengesetzte Richtungen bewegen und sich dabei gegenseitig nicht aus den Augen lassen. Die Wächter bilden so gesehen eine mobile Lichtschranke. Es darf nie eine Wand zwischen die Lichtschranke geraten. Außerdem darf sich keiner der beiden Wächter soweit zurückbewegen, dass er auf seinem Weg die Tür überschreitet. Der Raum wird auf diese Weise durch das die Wächter verbindende Liniensegment in zwei Partitionen aufgeteilt. In dem „sauberen" Bereich, in dem sich auch die Tür befindet, kann sich kein Eindringling befinden. In dem noch zu säubernden Bereich können sich dagegen noch potentielle Eindringlinge befinden. Die Suche ist abgeschlossen, wenn die Wächter zusammen einmal komplett um den Raum entlang der Wände gelaufen sind und an einem Punkt zusammentreffen. Ein Eindringling gilt als aufgespürt, wenn er die Lichtschranke, also das die beiden Wächter verbindende Liniensegment, kreuzt. Man könnte sich dieses Liniensegment auch als ein von den Wächtern langgezogenes Gummiband vorstellen. Die Wächter müssten sich dann so bewegen, dass das Gummiband stets eine gerade Linie bildet: Knicke, die durch Berührung von in den Raum hineinragenden Ecken entstehen könnten, wären nicht erlaubt. Eindringlinge würden dann erkannt, wenn sie das Gummiband berührten.

1.2 Boundary 1-Searcher

Einen *Boundary 1-Searcher* dagegen kann man sich als einen Wächter in einem dunklen Raum vorstellen, der sich wie seine beiden Kollegen ebenfalls entlang des Randes bewegt, dem aber eine Taschenlampe bzw. eher ein Laserpointer zur Verfügung steht[1]. Ein Laserpointer, da wir nur einen Sichtstrahl und keinen Sichtkegel erlauben wollen, wie ihn eine gewöhnliche Taschenlampe liefern würde. Der Wächter kann mit dem Laserpointer zu jeder Zeit in jede beliebige Richtung zielen und spürt einen Eindringling auf, wenn er ihn mit dem Laserpointer trifft. Ähnlich wie die zwei Wächter, soll der Boundary 1-Searcher bei der Suche stets eine Bedingung aufrechterhalten: Der Bereich zu seiner Linken soll sauber sein, d. h. von ihm aus gesehen sollen sich keine Eindringlinge auf der linken Seite seines Sichtstrahls aufhalten, und damit kein Eindringling durch die Tür flüchten kann, muss sich insbesondere auch die Tür stets zu seiner Linken befinden. Diese Bedingung nennen wir *Links-Invarianz*.

[1] Allgemein wäre ein *Boundary k-Searcher* ein Wächter, der mit k Laserpointern ausgestattet ist.

2 Vorbereitendes

2.1 Allgemeine Notationen

Im weiteren Verlauf der Ausarbeitung soll stets ein einfaches Polygon P in der Ebene (das ist ein Polygon ohne Selbstschnitte und Löcher) gegeben sein. Die n Eckpunkte von P seien im Uhrzeigersinn nummeriert: $0, 1, ..., n-1$, $(n \geq 3)$. Die Kante zwischen den Eckpunkten u und v wird mit (u, v) bezeichnet. Der Rand von P, in Zeichen ∂P, besteht aus den Ecken und Kanten von P. $Pred(v)$ und $Succ(v)$ bezeichnen den Vorgänger und den Nachfolger von v in der Folge der n Eckpunkte. Für zwei Punkte $a, b \in \partial P$ bezeichne $\partial P_{cw}(a, b)$ das offene und $\partial P_{cw}[a, b]$ das abgeschlossene Randstück des Polygons beim Umlauf von a nach b im Uhrzeigersinn entlang des Randes. Entsprechend sollen $\partial P_{ccw}(a, b)$ und $\partial P_{ccw}[a, b]$ die Pendants beim Umlauf gegen den Uhrzeigersinn bezeichnen.

Für die Randpunkte a, b und $d \in \partial P$ sei die Vorgängerrelation $\prec_d \subseteq \partial P \times \partial P$ wie folgt definiert: $a \prec_d b :\Leftrightarrow$ beginnend bei d wird beim Uhrzeigersinnumlauf des Randes der Punkt a vor dem Punkt b angetroffen.

Eine Ecke mit Innenwinkel $> 180°$ nennen wir *reflexe Ecke*. Sei r solch eine Ecke. Wir erhalten den Punkt $B(r) \in \partial P$, indem wir die Kante $(Succ(r), r)$ im Innern von P verlängern und so eine r gegenüberliegende Kante treffen. Analog erhalten wir den Punkt $F(r)$, indem wir die Kante $(Pred(r), r)$ verlängern. Wir nennen die durch $\partial P_{cw}[r, B(r)]$ und durch $\overline{rB(r)}$ begrenzte Fläche die *CW-Komponente* von r. Entsprechend wird die durch $\partial P_{ccw}[r, F(r)]$ und $\overline{rF(r)}$ begrenzte Fläche *CCW-Komponente* von r genannt. $r, B(r)$ und $F(r)$ werden später eine wichtige Rolle einnehmen, wir werden in ihrem Zusammenhang von *kritischen Punkten* sprechen. [2]

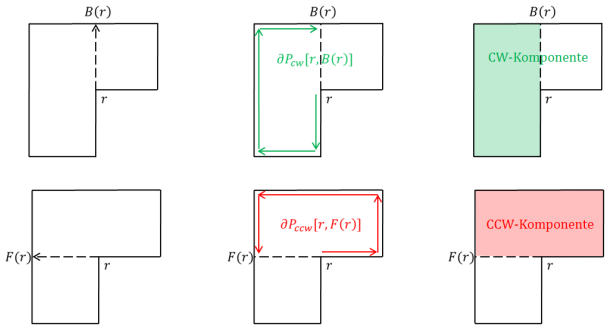

Abbildung 1: Illustration einiger Begriffe.

[2] Die Bezeichnungen *B, F, CW, CCW, Pred* und *Succ* sind Abkürzungen der aus dem Englischen stammenden Begriffe *backward extension point, forward extension point, clockwise, counter-clockwise, predecessor* und *successor*.

2.2 Sichtbarkeit

Zwei Punkte $u, v \in P$ werden als *gegenseitig sichtbar* bezeichnet, wenn das Liniensegment $\overline{uv}$ komplett in P enthalten ist (je nach Konvention abgesehen von u und v selbst). Unter der Annahme $\partial P \subseteq P$ wird so eine *schleifende Sichtbarkeit* definiert: u und v sind auch dann noch gegenseitig sichtbar, wenn das Liniensegment $\overline{uv}$ die reflexe Ecke r berührt.

2.3 V-Diagramme

Seien $p, q \in \partial P$. Sei $|\partial P|$ die Länge des Randes von P. Ein Element aus $\partial P \times \partial P$ wird *Konfiguration* genannt. Für zwei Punkte $p, q \in \partial P$ bezeichne $\langle p, q \rangle$ eine Konfiguration. Eine Konfiguration $\langle p, q \rangle$ wird *sichtbar* genannt, wenn p und q gegenseitig sichtbar sind. Andernfalls nennen wir die Konfiguration *unsichtbar*. In Abbildung 2 ist die Konfiguration $\langle d, B(r) \rangle$ sichtbar, die Konfiguration $\langle d, Pred(r) \rangle$ dagegen unsichtbar.

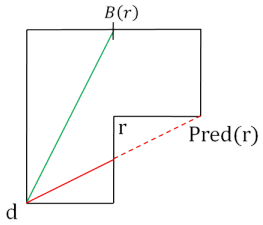

Abbildung 2: Eine sichtbare und eine unsichtbare Konfiguration.

Wir betrachten einen beliebigen Punkt $o \in \partial P$, den Urprung. Umlaufen wir den Rand beginnend bei o im Uhrzeigersinn und messen dabei den zurückgelegten Weg (den wir *Randdistanz* nennen), so können wir mit Hilfe einer reellen Zahl x den Randpunkt repräsentieren, der entlang des Randes die Distanz $x - k \cdot |\partial P|$ zu o hat ($k \in \mathbb{Z}$, so dass $0 \leq x - k \cdot |\partial P| < |\partial P|$).

Für $x, y \in \mathbb{R}$ besteht der *Visibilitäts-* oder kurz *V-Raum* $\mathcal{V}$ der Konfigurationen aus der unendlichen Fläche, die durch die Geraden $S := y = x$ und $G := y = x - |\partial P|$ beschränkt wird. Die Startgerade S ist also die Winkelhalbierende des durch die x- und y-Achse aufgespannten rechten Winkels. Die Zielgerade G verläuft parallel zu S durch den Punkt $(|\partial P|, 0)$. Offensichtlich gilt $(x, y) \in \mathcal{V} \Leftrightarrow x - |\partial P| \leq y \leq x$, wie man sich anhand von Abbildung 3 leicht klarmacht. Wir benötigen noch eine unkomplizierte Notation für die korrekte Beschriftung der Achsen unter Verwendung der Randdistanz eines Punktes zum Ursprung. Dazu sei $\widehat{p}$ die Länge von $\partial P_{cw}[o, p]$, $p \in \partial P$.

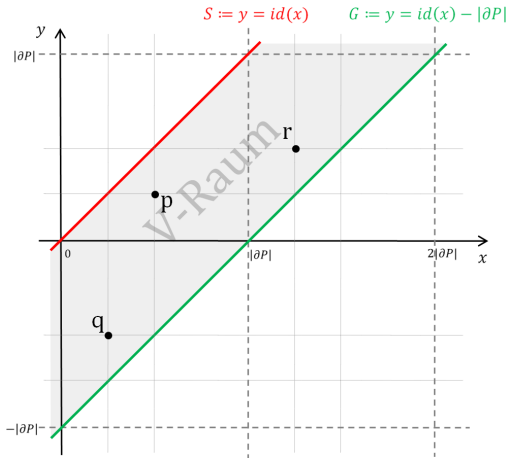

Abbildung 3: V-Raum - Die Punkte p' und p'' sind „symmetrisch“ zu p.

Von Symmetrie im V-Raum kann man in dem Sinne sprechen, dass $\overline{pq} \in P \Leftrightarrow \overline{qp} \in P$, das heißt wenn die Konfiguration $\langle p, q \rangle$ (un)sichtbar ist, dann ist es auch die Konfiguration $\langle q, p \rangle$. Ein so genanntes *V-Diagramm* erhalten wir, indem wir Punkte im anfangs hellgrauen V-Raum wie folgt einfärben:

Punkt $(x, y) \in \mathcal{V}$ ist dunkelgrau $\Leftrightarrow$ die durch $x, y \in \mathbb{R}$ repräsentierten Randpunkte von P sind *nicht* gegenseitig sichtbar.

Nach der beschriebenen Einfärbung stellen also hellgraue Punkte im V-Diagramm genau die sichtbaren Konfigurationen dar. Wir werden die Menge der sichtbaren Konfigurationen mit $\mathcal{V}_v$ und die Menge der unsichtbaren Konfigurationen mit $\mathcal{V}_i$ bezeichnen. Da ein Punkt im V-Diagramm stets entweder hellgrau oder dunkelgrau eingefärbt ist, aber nicht beides zugleich, gilt $\mathcal{V}_v = \mathcal{V} \setminus \mathcal{V}_i$.

Zu der Frage, wie die dunkelgrauen Flächen in Abbildung 4 entstehen, sollten wir uns noch einmal die Begriffe aus Abbildung 1 klarmachen. Wählen wir beispielsweise einen Randpunkt p der CW-Komponente und schicken einen Strahl los, der r berührt, dann treffen wir einen Randpunkt p' der CCW-Komponente. Alle Punkte, die auf $\partial P_{ccw}[p', r]$ liegen, sind von p aus sichtbar. Alle anderen sind es nicht, das sind die Punkte auf $\partial P_{cw}(p', r)$. Man sieht, dass gerade die kritischen Punkte $B(r)$, $F(r)$ und r (sowie $Pred(r)$ und $Succ(r)$) für die senkrechten und waagerechten Linien im V-Diagramm verantwortlich sind. Auffällig ist dabei, dass r je eine waagerechte und senkrechte Linie verursacht, die beide ihren Ursprung in $(\widehat{r}, \widehat{r})$ haben und in $(\widehat{r}, \widehat{B(r)})$ sowie $(\widehat{F(r)}, \widehat{r})$ enden. Anzumerken ist, dass die in der Abbildung zu S bzw. G parallel verlaufende diagonale Linie im allgemeinen kein ge-

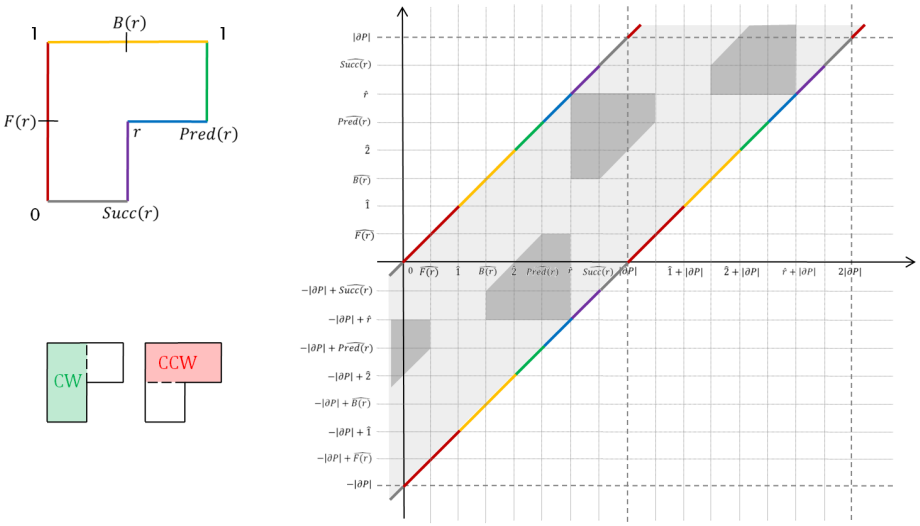

Abbildung 4: Ein Polygon und eine Skizze des zugehörigen V-Diagramms mit Ursprung in Eckpunkt $0 \in \partial P$.

radliniges Segment sein muss, sondern lediglich eine y-monotone Kette. In Abbildung 4 ist sie eine Diagonale, weil das zugehörige Polygon ausschließlich rechte Winkel enthält und symmetrisch ist. Abbildung 5 skizziert, wie der allgemeine Fall aussieht.

Die Autoren von [4] haben gezeigt, dass der Inhalt eines V-Diagramms auf die horizontalen und vertikalen Linien, die S oder G berühren, reduziert werden kann, ohne dass die topologische Information verloren geht. Sie haben dazu ein Lemma über Schnitte dunkelgrauer Regionen aufgestellt, das besagt, dass wenn sich zwei dunkelgraue Regionen überlappen, dann ist einer der Schnittpunkte in jedem Falle einer der durch Schnitt einer horizontalen und einer vertikalen Linie entsteht. Mit anderen Worten, wenn sich zwei dunkelgraue Regionen überlappen, dann überlappt sich insbesondere ein horizontales Liniensegment der einen Region mit einem vertikalen Liniensegment der anderen Region. Es kann Überlappungen dunkelgrauer Bereiche also nur geben, wenn es Schnitte der horizontalen und vertikalen Linien gibt. Gibt es solche Schnitte nicht, dann gibt es auch keine Überlappungen dunkelgrauer Bereiche. Ein solch auf die horizontalen und vertikalen Linien reduziertes Diagramm wird kurz SVD[3] genannt. Die maximalen zusammenhängenden hellgrauen Flächen im SVD werden wir als *Zellen* bezeichnen. Zellen, die S und G miteinander verbinden, das heißt, die sowohl S als auch G berühren, nennen wir *Verbindungszellen*. Pfade, die im V-Diagramm S und G miteinander verbinden (ohne eine horizontale oder vertikale Barriere zu kreuzen)

[3] skeletal V-diagram

werden später beim Begriff der Durchsuchbarkeit eines Polygons eine Rolle spielen. Es wird für unsere Zwecke daher genügen, die SVDs statt der gewöhnlichen V-Diagramme zu betrachten, da aus der Existenz einer Verbindungszelle im SVD unmittelbar die Existenz eines S und G verbindenden Pfades im gewöhnlichen V-Diagramm folgt.

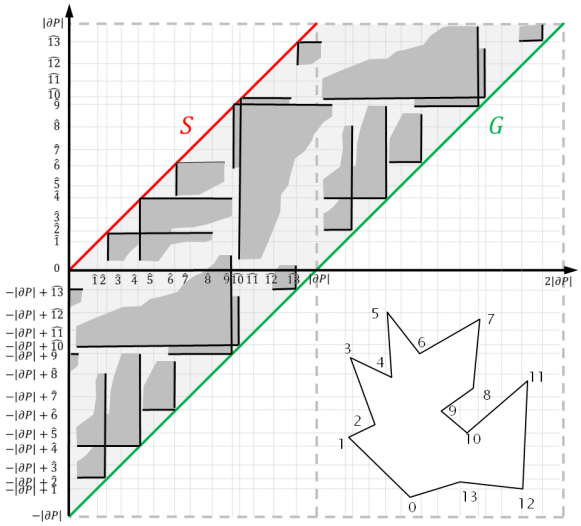

Abbildung 5: Ein Polygon und sein V-Diagramm ($o = 0$).

In Abbildung 5 ist ein etwas komlexeres Polygon und sein zugehöriges V-Diagramm dargestellt. Abbildung 6 zeigt das zugehörige skelettale V-Diagramm.

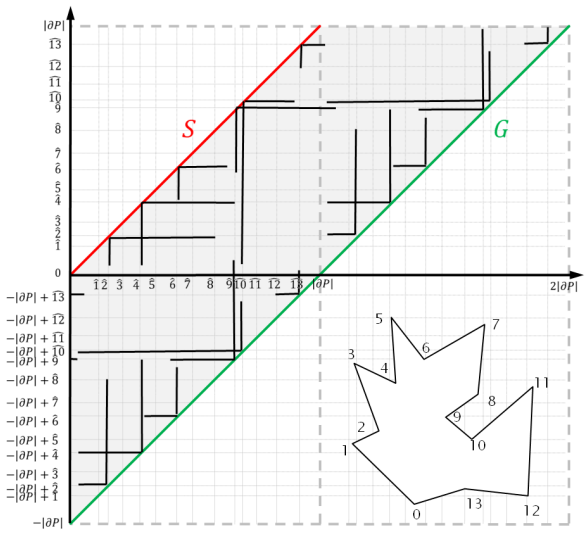

Abbildung 6: Ein Polygon und sein SVD ($o = 0$).

2.4 Sprünge (Boundary 1-Searcher)

Wenn der Boundary 1-Searcher seinen Laserpointer über eine reflexe Ecke r hinaus bewegt, kann es abhängig von der Position des Boundary 1-Searchers passieren, dass der Laserpunkt von der Kante $(Pred(r), r)$ bzw. $(r, Succ(r))$ auf eine Kante (p, q) wechselt mit $p \neq r$ und $q \neq r$, der Laserpunkt also von einer Kante auf eine an sie nicht angrenzende andere Kante springt. Dabei wird zwischen *legalen* und *illegalen* Sprüngen unterschieden. Ein legaler Sprung ist einer, bei dem der Boundary 1-Searcher seinen Laserstrahl gegen den Uhrzeigersinn über eine reflexe Ecke hinwegbewegt, von der saubere Randpunkte verdeckt werden. Bei solch einem Sprung verkleinert sich der gesäuberte Bereich, kann aber durch eine geschickte Wahl von Folgekonfigurationen wiederhergestellt werden. Mit einem illegalen Sprung haben wir es dagegen zu tun, wenn der Boundary 1-Searcher seinen Laserstrahl über eine reflexe Ecke bewegt, die einen „kontaminierten“ Bereich verdeckt. Tritt solch ein Sprung auf, muss die Suche neugestartet werden, da danach das gesamte Polygon kontaminiert ist. Wir werden illegale Sprünge daher verbieten. Sprünge des Boundary 1-Searchers selbst, sind ebenfalls nicht erlaubt.

In Abbildung 7 sehen wir die geschilderte Problematik. Weiße Flächen sollen gesäuberte Flächen repräsentieren, graue Flächen sind noch kontanimiert. Die Tür sei in Ecke 1. Der Boundary 1-Searcher rotiert seinen Laserstrahl nach links. Der Laserpunkt springt dabei von der Kante $(r, Succ(r))$ auf die Kante $(2, Pred(r))$. Ein in der grauen Fläche aus den beiden ersten Teilabbildungen gefangener Eindringling könnte nun „hinter“ die reflexe Ecke eilen und sich in dem größeren grauen Bereich verstecken, der zuvor

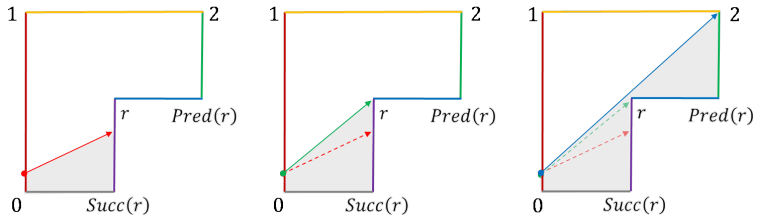

Abbildung 7: Illustration eines legalen Sprungs von (a) über (b) nach (c).

noch als sauber gegolten hat. Würde der Boundary 1-Searcher den Laserstrahl nun unüberlegt in die Ausgangskonfiguration zurückrotieren, würde er dem Eindringling gar die Flucht durch die Tür ermöglichen.

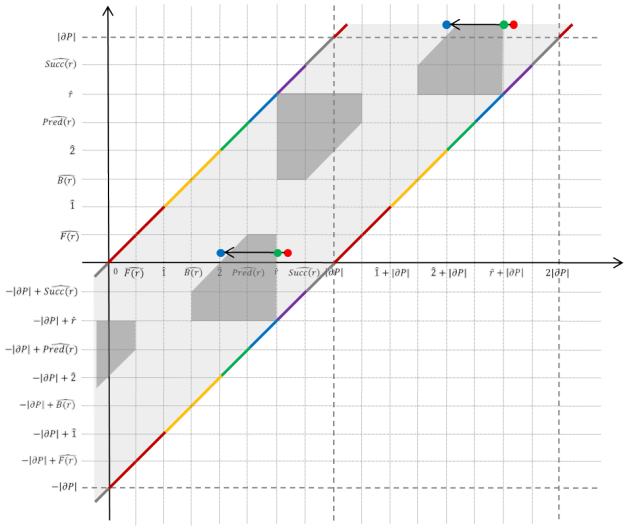

Abbildung 8: Illustration des Sprungs im V-Diagramm.

Abbildung 8 veranschaulicht, was bei einem legalen Sprung wie in Abbildung 7 im zugehörigen V-Diagramm passiert. Die Position des Boundary 1-Searchers wurde hier o.B.d.A. auf der vertikalen Achse abgetragen, die Position des Laserpunktes auf der horizontalen. Die an sich stetige Rotation des Laserstrahls ist in $\mathcal{V}_v$ nicht mehr stetig. Die rote Konfiguration kann durch ein geradliniges Segment nicht mit der blauen verbunden werden, ohne dass eine dunkelgraue Fläche gekreuzt wird. Legale Sprünge sind im SVD solche, die die vertikalen „Barrieren“ von rechts nach links überwinden. Hätten wir die Position des Boundary 1-Searchers auf der horizontalen Achse abgetragen und die des Laserpunktes auf der vertikalen, dann wären Sprünge legal, die horizontale Barrieren von oben nach unten überwinden.

3 Die Suche mit zwei Wächtern

3.1 Durchsuchbarkeit

Wir wollen die in Abschnitt 1.1 beschriebene Strategie nun formalisieren.

Definition 1 Gegeben ein Polygon P, $d \in \partial P$. Der Raum $P(d)$ kann mit zwei Wächtern durchsucht werden, wenn es zwei stetige Funktionen $l : [0,1] \to \partial P$ und $r : [0,1] \to \partial P$ gibt, so dass

- $l(0) = r(0) = d$ und $l(1) = r(1) \in \partial P$
- $l(x)$ und $r(x)$ sind gegenseitig sichtbar $\forall x \in (0,1)$
- $d \in \partial P_{cw}[r(t), l(t)]$ $\forall t \in [0,1]$
- $|\partial P_{cw}[l(0), l(1)] + \partial P_{ccw}[r(0), r(1)]| = |\partial P|$

Wir nennen solch einen Raum kurz *durchsuchbar* und d eine *sichere* Tür, wenn $P(d)$ durchsuchbar ist.

Die Funktionen $l(t)$ und $r(t), t \in [0,1]$, beschreiben die Positionen des linken und rechten Wächters auf ∂P zum Zeitpunkt t. Im ersten Punkt der Definition ist gefordert, dass beide Wächter die Suche bei der Tür d starten und in einem beliebigen Randpunkt beenden. Der zweite Punkt fordert, dass zu jedem Zeitpunkt die gegenseitige Sichtbarkeit der beiden Wächter gewährleistet sein soll. Der zweite Punkt stellt zusammen mit dem dritten Punkt sicher, dass sich die Tür d stets im sauberen Bereich befindet. Punkt vier sorgt dafür, dass die beiden Wächter beim Zusammentreffen in $t = 1$, auch wirklich den ganzen Rand durchsucht haben. Da durch den dritten Punkt sichergestellt ist, dass keiner der Wächter auf seinem Weg die Tür überschreitet, ist jede Konfiguration im V-Raum durch $x \geq \widehat{d}$ und $y \leq \widehat{d}$ beschränkt. Das bedeutet wiederum, dass wir uns im V-Diagramm nur auf diesen Bereich konzentrieren müssen, das ist gerade der Bereich, der durch das Dreieck mit den Eckpunkten $(\widehat{d}, \widehat{d})$, $(\widehat{d}, \widehat{d} - |\partial P|)$ und $(\widehat{d} + |\partial P|, 0)$ berandet wird (vgl. Abbildung 9). Die Startgerade S besteht in solch einem Dreieck aus genau einem Punkt, das ist derjenige Punkt, der die Startkonfiguration $\langle d, d \rangle$ repräsentiert.

Lemma 2 *Ein Raum $P(d)$ ist durchsuchbar genau dann, wenn im durch $x \geq \widehat{d}$ und $y \leq \widehat{d}$ beschränkten SVD ein stetiger Pfad π existiert, der den Startpunkt $(\widehat{d}, \widehat{d}) \in S$ mit der Zielgeraden G verbindet. Insbesondere darf der Pfad keine horizontale oder vertikale Linie kreuzen. Wir nennen π einen Suchpfad.*

Beweis. Im Folgenden werden wir für das wie in Lemma 2 beschränkte SVD kurz LVD(d) schreiben (lokales Visibilitäts-Diagramm) und im Rest dieses Kapitels das LVD(d) meinen, wenn wir von SVD oder V-Diagramm sprechen. Wir müssen zwei Richtungen zeigen:

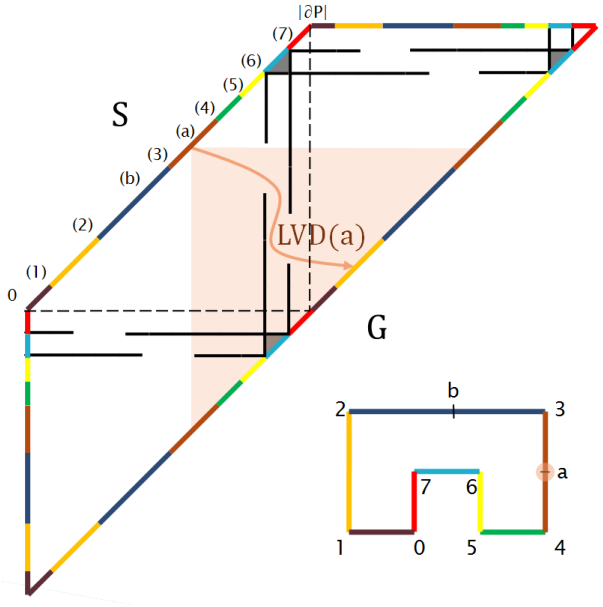

Abbildung 9: Ein Polygon, das GVD und das LVD für die Tür in Punkt a.

1. $\exists \pi \Rightarrow P(d)$ durchsuchbar

Wenn es im SVD einen stetigen Pfad π gibt, der S und G verbindet, ohne eine horizontale oder vertikale Kante zu kreuzen, der also vollständig in $\mathcal{V}_v$ verläuft, dann ist jede Konfiguration entlang des Pfades eine sichtbare. Die beiden Wächter erhalten zu jeder Zeit die gegenseitige Sichtbarkeit. Die Startkonfiguration besteht aus dem einzigen Punkt von S, die Wächter beginnen ihre Suche bei der Tür. Da π stetig ist, sind die Bewegungen der Wächter ebenfalls stetig. Der Pfad verläuft nach Voraussetzung zu jeder Zeit innerhalb des SVDs, damit ist gewährleistet, dass keiner der Wächter auf seinem Weg die Tür überschreitet, sich die Tür also stets im sauberen Bereich befindet. Die Endkonfiguration liegt in einem Punkt auf G, das bedeutet, dass sich die Wächter am Ende ihrer Suche in einem Randpunkt treffen.

2. $P(d)$ durchsuchbar $\Rightarrow \exists \pi$

Wenn $P(d)$ durchsuchbar ist, dann folgt daraus, dass die Punkte aus Definition 1 erfüllt sind. Die Suche beginnt bei der Tür d. Die Startkonfiguration ist daher $\langle d, d \rangle$ und damit ist $(\widehat{d}, \widehat{d}) \in S$ ein Punkt des Pfades π. Mit selbem Argument folgt, dass $(\widehat{g}, \widehat{g}) \in G$, wenn sich die Wächter am Ende ihrer Suche in $g \in \partial P$ treffen. Nun bewegen sich beide Wächter stetig entlang des Randes unter Erhaltung der gegenseitigen Sichtbarkeit, und keiner der beiden kreuzt

auf seinem Weg die Tür. Das bedeutet, dass wir es im V-Diagramm mit einem stetigen Pfad zu tun haben, dass jede von den Wächtern eingenommene Konfiguration vollständig in $\mathcal{V}_v$ verläuft, inbesondere keine horizontale oder vertikale Kante kreuzt, und dass keine Konfiguration das SVD verläßt.

Abbildung 9 illustriert Lemma 2. Im LVD(a) gibt es einen Pfad, der a auf S mit der Zielgeraden G verbindet. Im nicht eingezeichneten LVD zur Tür b würde dagegen jeder Pfad durch die horizontalen und vertikalen Barrieren blockiert. □

3.2 Test auf Sicherheit einer Tür

Ein Suchpfad kann nur innerhalb einer Verbindungszelle verlaufen, er darf im SVD keine horizontale oder vertikale Kante kreuzen. Würde er es nämlich tun, dann würde das einer Verletzung der gegenseitigen Sichtbarkeit gleichkommen. Zellen die keine Verbindungszellen sind, werden wir in diesem Zusammenhang als *Sackgassen* bezeichnen. Abbildung 9 illustriert einen Suchpfad für die Tür im Punkt a sowie „drei“ Sackgassen (grau), die Segmente von G unerreichbar machen. Drei in Anführungsstrichen, weil es sich bei dem unsicheren und zugleich unerreichbaren Segment immer um ein und dasselbe Segment handelt, nämlich um das cyan markierte Segment $(6, 7)$. Es ist klar, dass ein Raum nicht durchsucht werden kann, wenn sich die Tür in einer Sackgasse befindet. Genauso ist es klar, dass Punkte auf G, die auf dem Rand einer Sackgasse liegen, von S aus nicht erreicht werden können, d. h. dass sich die beiden Wächter in diesen Punkten zum Abschluss ihrer Suche nicht treffen können. Daraus folgt unmittelbar, dass $P(d)$ auch nicht durchsucht werden kann, wenn alle Punkte von G durch Sackgassen berandet werden. Wir werden uns nun ein paar Beispiele anschauen. Der Einfachheit halber werden wir für die folgende Betrachtung das Polygon als Kreis zeichnen und nur die für unsere Betrachtung relevanten Punkte markieren.

Abbildung 10 listet alle Fälle auf, in denen die Tür in einer Sackgasse gefangen ist, folglich es keinen stetigen Pfad geben kann der S und G verbindet, ohne eine horizontale oder vertikale Kante im SVD zu kreuzen. Man überlegt sich leicht, dass zwei reflexe Ecken benötigt werden, damit eine Tür unsicher wird. Ein Polygon mit einer einzigen reflexen Ecke ist immer durchsuchbar: Man läßt zunächst einen der Wächter zur reflexen Ecke laufen, und anschließend kann der andere Wächter den restlichen Rand absuchen, ohne die gegenseitige Sichtbarkeit zu verletzen, da alle Randpunkte des Polygons von der einzigen reflexen Ecke gesehen werden. Es sind also zwei reflexe Ecken nötig, damit es unsichere Türen geben kann. Anhand der LVDs sieht man schnell ein, dass es nur die abgebildeten Fälle geben kann. Jede der beiden reflexen Ecken verursacht im LVD je eine horizontale und

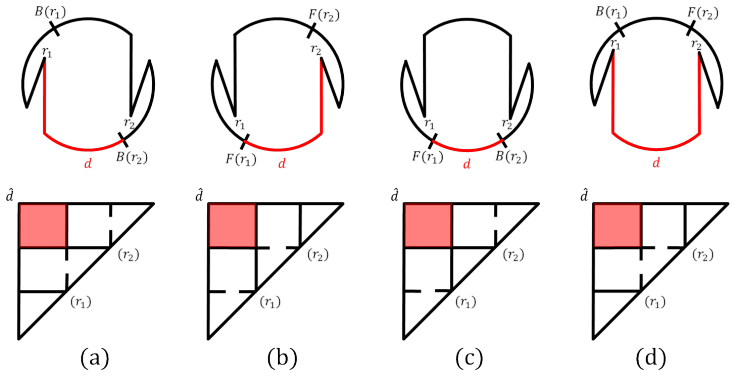

Abbildung 10: Anordnungen reflexer Ecken, die d unsicher machen.

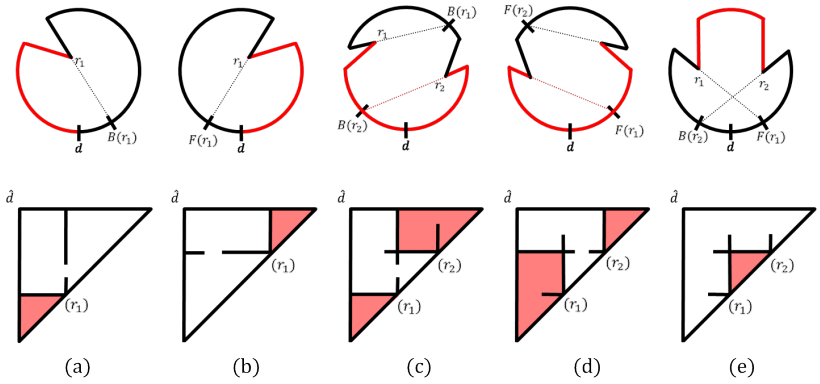

Abbildung 11: Anordnungen reflexer Ecken, die Segmente von G unerreichbar machen.

eine vertikale Linie. Die Länge der zu einer reflexen Ecke gehörenden Linien muss in der Summe immer kleiner als $|\partial P|$ sein. Das bedeutet, es gibt immer eine „Öffnung“, entweder im horizontalen oder im vertikalen Segment einer reflexen Ecke (oder in beiden, aber dann wäre die Tür sicher): Zwei reflexe Ecken, pro reflexe Ecke zwei Möglichkeiten für die Platzierung der Öffnung, also vier Fälle.

Abbildung 11 dagegen zeigt alle fünf Fälle auf, bei denen Segmente auf G unerreichbar sind. Die Argumentation, warum es genau fünf Fälle sein müssen, ist ähnlich. Eine einzige reflexe Ecke kann entweder das untere oder das obere Stück der Hypotenuse des LVD unerreichbar machen. Das sind zwei Möglichkeiten. Sind zwei reflexe Ecken beteiligt, so können diese entweder auf zwei Arten die äußeren Bereiche der Hypotenuse unerreichbar machen oder aber einen mittleren Bereich. Mehr Kombinationen gibt es nicht. Insgesamt also fünf Fälle. Es fällt auf, dass die Fälle (a), (b) und (d) aus Abbildung 10 bis auf die Position der Tür den Fällen (c), (d) und (e)

aus Abbildung 11 gleichen.

Die Zahl der kritischen Punkte liegt in $O(n)$, denn jede reflexe Ecke verursacht genau drei kritische Punkte, das sind die reflexe Ecke selbst, ihre Forward-Extension und ihre Backward-Extension. Die $O(n)$ vielen kritischen Punkte eines Polygons kann man in Zeit $O(n\log n)$ mit Hilfe der in [5] vorgestellten Datenstruktur bestimmen und gemäß der $\prec_d$-Relation in einer Liste sortiert speichern, um das Polygon anschließend auf die Fälle aus Abbildung 10 und Abbildung 11 effizient testen zu können. Zur Bestimmung der Forward- und Backward-Extensions der reflexen Ecken eignet sich diese Datenstruktur (die in $O(n\log n)$ aufgebaut werden kann), da sie Lösungen folgender Probleme ermöglicht:

- Ermittlung der von einer „leuchtenden“ Kante beleuchteten Randstücke
- zu einem auf einer Kante startenden und ins Polygoninnere strahlenden Strahl s: Bestimmung des Treffpunktes von s mit ∂P, d. h. Bestimmung des ersten Schnittpunkts mit ∂P
- zu einem Randpunkt q: Ermittlung von Segmenten einer Kante, von denen aus q gesehen wird (wenn es solche gibt)

Das erste Problem kann mit der Datenstruktur in $O(n\log n)$ gelöst werden, die beiden anderen in $O(\log n)$. Siehe [5] für Details, der grobe Hintergedanke bei der Datenstruktur ist aber der folgende. Beleuchten zwei Lichtstrahlen dieselbe Kante, dann wird die Kante auch von allen Lichtstrahlen beleuchtet, die „zwischen“ diesen beiden Lichtstrahlen durch deren Schnittpunkt verlaufen. Die Datenstruktur nutzt die Dualität von Punkten und Geraden aus. Lichtstrahlen werden als duale Punkte betrachtet. Solche Lichtstrahlen in Punktform werden nach folgender Regel in Regionen unterteilt: Zwei Lichtstrahlen liegen in derselben Region, wenn sie dieselbe Kante treffen. Die Regionen werden mittels eines Divide & Conquer Ansatzes berechnet, der auf dem *Pologon-Cutting-Theorem* von Chazelle beruht. Jenes besagt, dass es mit $O(n\log n)$ Vorbereitungszeit möglich ist, zwei Eckpunkte p_i und p_j von P zu bestimmen, so dass die durch p_i und p_j gegebene Diagonale P in zwei einfache Polygone unterteilt mit maximal $\lceil 2n/3 \rceil + 1$ Ecken. Für unsere Anwendung ist die Lösung des zweiten Problems interessant, da wir pro reflexe Ecke zwei Strahlen losschicken und auf Schnitt mit ∂P testen müssen, um die Forward- und Backward-Extensions zu bestimmen. Die Datenstruktur eignet sich dafür deshalb so gut, weil die von den zu schießenden Strahlen angestrahlte Kante gerade durch die Region, in der die Strahlen liegen, bestimmt ist (zur Erinnerung, alle Strahlen, die dieselbe Kante beleuchten, liegen in derselben Region). Es muss also nur die Region gefunden werden, in der unser Strahl liegt, um die von ihm geschnittene Kante zu lokalisieren. Der anschließende Schnittest zur Bestimmung des genauen Schnittpunktes erfolgt in konstanter Zeit.

Den Test auf die Fälle aus Abbildung 10 wollen wir detailliert anhand des Falls (a) beschreiben.

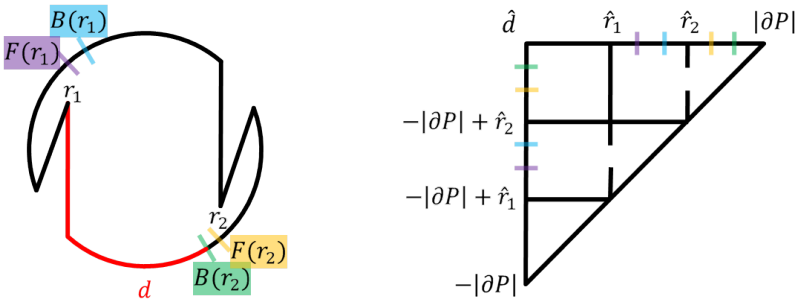

Abbildung 12: Fall (a) aus Abbildung 10.

Der Fall (a) tritt auf, wenn es zwei reflexe Ecken r_1 und r_2 gibt, so dass $r_1 \prec_d B(r_1) \prec_d r_2 \prec_d B(r_2)$, wie man Abbildung 12 leicht entnehmen kann. Beginnt man nämlich bei d einen Uhrzeigersinnumlauf, so begegnet man zunächst der reflexen Ecke r_1, dann $F(r_1)$ und schließlich $B(r_1)$. Läuft man weiter, dann trifft man auf r_2, gefolgt von $F(r_2)$ und schließlich $B(r_2)$. Da hier $F(r_1)$ zwischen r_1 und $B(r_1)$ liegt, kann man die Forward-Extension von r_1 vernachlässigen (entsprechend bei r_2). Entscheidend sind hier die zwei kritischen Punktepaare $(r_i, B(r_i))$ mit der zuvor genannten Reihenfolge.

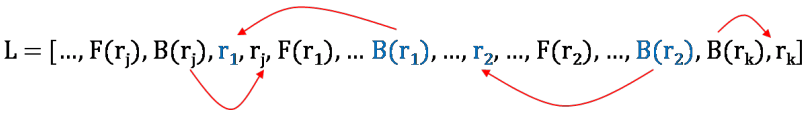

Abbildung 13: Liste für Fall (a) aus Abbildung 10.

Sei $|r|$ die Anzahl der reflexen Ecken. Um auf Fall (a) zu testen, muss man also in der nach der $\prec_d$-Relation sortierten Liste der kritischen Punkte nach dem ersten $B(r_i)$, $i \in \{1, ..., |r|\}$ suchen, so dass man auf der Suche r_i selbst schon begegnet ist (in der Abbildung ist das $(r_1, B(r_1))$). Hat man solch ein Paar gefunden, fährt man mit der Suche nach dem nächsten solchen Paar fort. Ist solch ein zweites Paar gefunden, gilt Fall (a) als erkannt. Den Test auf diesen Fall (wie auch die Tests auf die anderen Fälle) kann man in $O(n)$ Zeit erledigen, indem man die Liste derart verzeigert, dass man für den aktuell betrachteten kritischen Punkt an Stelle i in der Liste unmittelbar seine zugehörigen Extensions findet (falls es sich bei dem Punkt an Stelle i um eine reflexe Ecke handelt) bzw. die reflexe Ecke und die andere Extension (falls es sich um eine der Extensions handelt). Man kann dann nämlich jeden besuchten kritischen Punkt als besucht markieren und speziell in Fall (a) für ein gerade betrachtetes $B(r_i)$ prüfen, ob man dem zugehörigen r_i

schon begegnet ist. Ähnlich sieht es bei den anderen Fällen aus. Für Fall (b) müsste man analog die Liste auf $F(r_1) \prec_d r_1 \prec_d F(r_2) \prec_d r_2$ testen. Genauer: Man würde zunächst nach der ersten reflexen Ecke r_i suchen, so dass man auf der Suche nach ihr dem zugehörigen $F(r_i)$ schon begegnet ist. Anschließend würde man nach einer weiteren reflexen Ecke suchen, so dass man auch ihrer Forward Extesion schon begegnet ist. Dabei würde man in der Liste wie in Fall (a) beim Besuch eines kritischen Punktes diesen Punkt als besucht markieren, um beim Antreffen eines r_i mittels des Zeigers auf $F(r_i)$ schnell testen zu können, ob man $F(r_i)$ schon begegnet ist. Für Fall (c) müßte man auf $F(r_1) \prec_d r_1 \prec_d r_2 \prec_d B(r_2)$ testen und für Fall (d) wäre die Existenz von $r_1 \prec_d B(r_1)$ und $B(r_2) \prec_d r_2$ zu prüfen. Hat man einen der Fälle aus Abbildung 10 erkannt, so ist d eine unsichere Tür, $P(d)$ daher nicht durchsuchbar.

Anhand des Falls (a) erläutern wir den Test auf die Fälle aus Abbildung 11. Man will das für den Abschluss der Suche unerreichbare Randstück maximaler Länge ermitteln. Das ist in diesem Fall gerade das Randstück $\partial P_{cw}(d, r_1)$. Es muss das offene Randstück sein, weil r_1 als Treffpunkt der beiden Wächter erreicht werden kann, z. B. indem der linke Wächter zunächst im Uhrzeigersinn bis r_1 läuft während der rechte bei d wartet, um daraufhin gegen den Uhrzeigersinn zu ihm zu stoßen, ohne die gegenseitige Sichtbarkeit zu verletzen. Nun aber zurück zum Problem. Wir wollen das unerreichbare Randstück maximaler Länge ermitteln, hier $\partial P_{cw}(d, r_1)$. Dazu müssen wir in der Liste für Fall (a) einfach nach derjenigen reflexen Ecke r_i, $i \in \{1, ..., |r|\}$, suchen, die $r_i \prec_d B(r_i)$ erfüllt und unter allen anderen Ecken, die diese Bedingung erfüllen, $|\partial P_{cw}(d, r_i)|$ maximiert. Das kann man wie schon zuvor mit der nach der Vorgängerrelation sortierten und verzeigerten Liste der kritischen Punkte erledigen. Für den Test auf den Fall könnte man folgendes Vorgehen in Erwägung ziehen:

- Speichere in der Liste für alle i einen Zeiger von $B(r_i)$ auf r_i.
- Durchmustere die Liste von links nach rechts beginnend beim ersten Element.
- Markiere jede besuchte reflexe Ecke.
- Sobald ein $B(r_i)$ angetroffen wird, prüfe ob r_i markiert ist. Falls ja, lösche alle vor r_i gesetzten Markierungen. r_i ist dann die zu d aktuell am weitesten entfernte reflexe Ecke mit $r_i \prec_d B(r_i)$.

Der symmetrische Fall (b) wird analog behandelt, in den Fällen (c), (d) und (e) muss man nach zwei reflexen Ecken suchen, die die eingezeichnete Anordnung der kritischen Punkte erfüllen. Für Fall (c) ist das die Anordnung $B(r_2) \prec_d r_1 \prec_d B(r_1) \prec_d r_2$, für Fall (d) die Anordnung $r_1 \prec_d F(r_2) \prec_d r_2 \prec_d F(r_1)$ und für Fall (e) die Anordnung $B(r_2) \prec_d r_1 \prec_d r_2 \prec_d B(r_1)$.

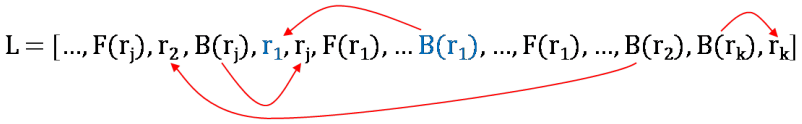

Abbildung 14: Liste für Fall (a) aus Abbildung 11.

Die beschriebenen Tests mit Hilfe der Liste sind in Zeit $O(n)$ möglich, damit können wir das Kapitel mit folgendem Theorem abschließen:

Theorem 3 *Sei P ein Polygon mit n Ecken. Sei $d \in \partial P$ eine Tür. Ob d sicher ist, kann mit zwei Wächtern in $O(nlogn)$ Zeit getestet werden.*

3.3 Ermittlung aller sicheren Türen

Im vorigen Abschnitt hat es für ein $d \in \partial P$ gereicht, das LVD(d) zu betrachten, da wir dort nur die Tür d auf Sicherheit testen wollten. Darin bestand die Gerade S aus genau einem Punkt, dem Punkt $(\widehat{d}, \widehat{d})$. Nun wollen wir aber alle sicheren Türen finden, also müssen wir auch alle potentiell sicheren Türen in Betracht ziehen. Das bedeutet, dass wir hierfür einen größeren Bereich im SVD betrachten müssen, nämlich den Bereich, der das durch $0 \leq x$ und $y < |\partial P|$ beschränkte Segment von S enthält. Wir geben diesem Bereich einen Namen und nennen ihn GVD (globales Visibilitäts-Diagramm, zur Illustration siehe Abbildung 9). Im GVD gilt $|S| = |\partial P| = \frac{|G|}{2}$. Wir werden für den Rest des Kapitels das GVD meinen, wenn wir von SVD oder V-Diagramm sprechen.

Da die Punkte von S nicht diskret sind und es selbst dann, wenn sie es wären, sehr naiv wäre, alle potentiellen Türen $d \in S$ auf Sicherheit zu testen, wollen wir uns ein geschickteres Verfahren überlegen. Dazu teilen wir den Rand ∂P in Segmente S_i auf, $i \in \{1, ..., 3|r|\}$, so dass jedes Segment einen kritischen Punkt als Start- und Endpunkt hat (und sonst keine kritischen Punkte enthält). Zur Erinnerung, die kritischen Punkte waren gerade die reflexen Ecken r, sowie ihre Erweiterungen $F(r)$ und $B(r)$ (siehe Abbildung 1 oder Abbildung 4).

Lemma 4 *Wenn ein Punkt $s_i \in S_i$ sicher ist, dann sind alle Punkte von S_i sicher. Wir sagen, s_i ist ein Repräsentant von S_i.*

Beweis. Die Behauptung folgt unmittelbar aus der Tatsache, dass im SVD alle Segmente in kritischen Punkten anfangen und enden. Befindet sich ein Punkt eines Segments in einer Sackgasse, dann ist das gesamte Segment von der Sackgasse berandet. □

Mit dem obigen Lemma können wir die Laufzeit für den Test auf sichere Türen wie folgt abschätzen. Um eine Tür zu testen, kostet es $O(n\log n)$ Zeit, wie wir in Abschnitt 3.2 gesehen haben. Führt man den Test für die $O(n)$ vielen Repräsentanten $s_i \in S_i$ durch, dann folgt eine Gesamtlaufzeit von $O(n^2\log n)$. Wir werden sehen, dass es besser geht.

3.3.1 Vorbereitendes zum Algorithmus

Damit der Leser einen Eindruck bekommt, wie unsichere und unerreichbare Bereiche im globalen V-Diagramm entstehen können, listen wir in Abbildung 15 und Abbildung 16 (ähnlich wie in Abschnitt 3.2) typische Formen von Polygonen und Anordnungen von Türen und reflexen Ecken auf, die Sackgassen zur Folge haben. Es genügt für uns jedoch, den Test nur für Sackgassen, wie sie in Abbildung 19 schematisch im GVD dargestellt sind, näher zu beschreiben. Solche Sackgassen finden wir in Abbildung 9 (grau markiert) bzw. in Abbildung 10 (c) oder Abbildung 11 (e) ebenfalls wieder. Es genügt, sich auf diese zu beschränken, da nur diese für den Algorithmus, den wir im nächsten Unterabschnitt kennenlernen werden, vorberechnet werden müssen. Alle anderen unsicheren Türpositionen wird der Algorithmus ohnehin erkennen, wobei die Begriffe $r_L(d)$ und $r_H(d)$, die weiter in diesem Abschnitt definiert werden, eine entscheidende Rolle spielen werden.

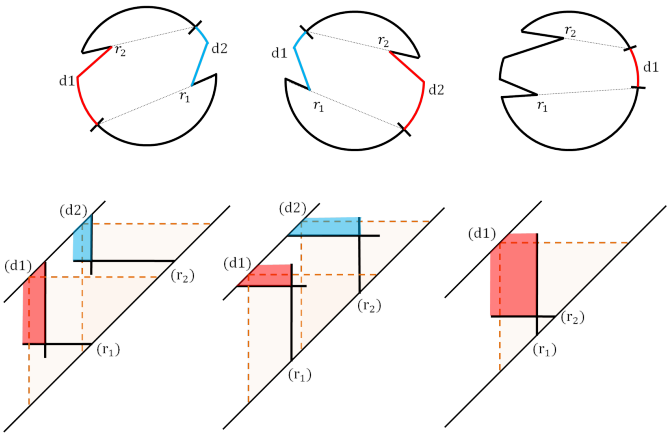

Abbildung 15: Türpositionen, die Sackgassen bei S verursachen.

In Abbildung 15 sehen wir Türpositionen, die S berandende Sackgassen verursachen. Abbildung 16 listet Türpositionen auf, die Sackgassen zur Folge haben, die G beranden. Es ist schematisch jeweils ein Ausschnitt des GVDs dargestellt, die gestrichelten Linien sollen das LVD einer Tür andeuten, die roten und blauen Bereiche sollen die unsicheren Türen bzw. unerreichbaren Bereiche markieren.

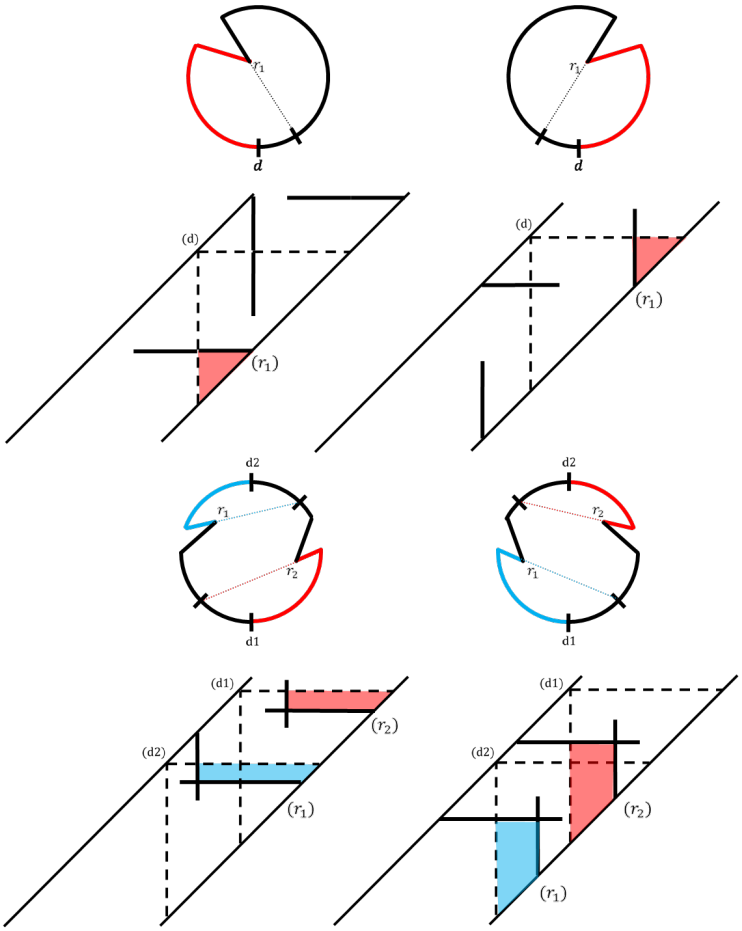

Abbildung 16: Türpositionen, die Sackgassen bei G verursachen.

Wir definieren $r_L(d)$ als die reflexe Ecke, deren CW-Komponente d nicht enthält und unter all solchen zu d im Uhrzeigersinn den größten Abstand hat. Ebenso definieren wir $r_H(d)$ als die reflexe Ecke, deren CCW-Komponente d nicht enthält und unter all solchen zu d gegen den Uhrzeigersinn den größten Abstand hat. Abbildung 17 veranschaulicht, welche Rolle $r_L(d)$ und $r_H(d)$ im SVD spielen. $r_L(d)$ und $r_H(d)$ sind die reflexen Ecken, die von d am weitesten entfernt sind und Teile von G unerreichbar machen. In der bereits bekannten Notation sind das die Randstücke $\partial P_{cw}(d, r_L(d))$ und $\partial P_{ccw}(d, r_H(d))$. Schauen wir uns in diesem Zusammenhang die Fallunterscheidung aus Abbildung 18 an, welche die vier denkbaren Fälle in einem LVD(d) zur Tür d illustriert, für die es mindestens eine der beiden reflexen Ecken $r_L(d)$ oder $r_H(d)$ gibt. Es ist klar, dass eine Tür d nie unsicher sein kann, wenn es zu der Tür kein $r_L(d)$ gibt, denn dann kann mit Sicherheit ein unteres Randstück der Hypotenuse des LVD(d) erreicht werden. Genauso kann die Tür d nie

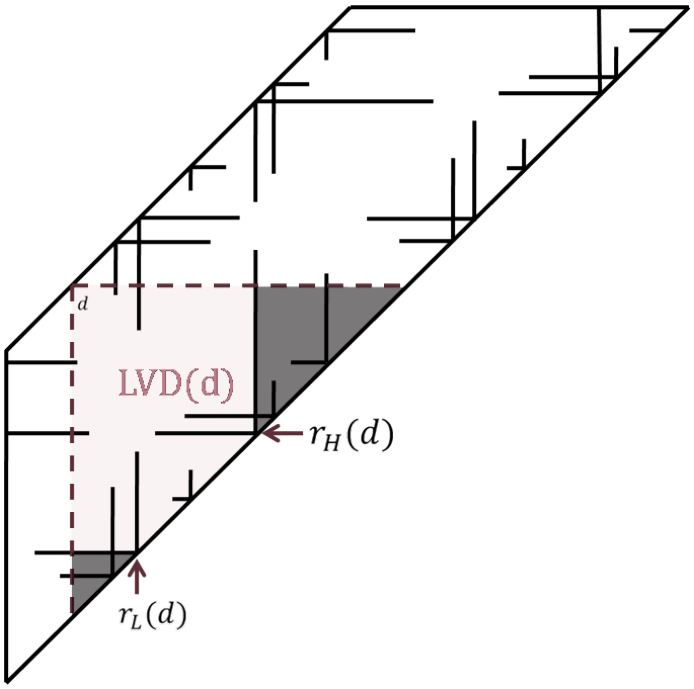

Abbildung 17: Illustration einiger Begriffe.

unsicher sein, wenn es zu der Tür kein $r_H(d)$ gibt, denn dann ist stets ein oberes Randstück der Hypotenuse des LVD(d) erreichbar. Eine Tür d ist auch dann sicher, wenn es $r_L(d)$ und $r_H(d)$ gibt mit $r_L(d) \prec_d r_H(d)$. Dann kann nämlich ein mittleres Segment erreicht werden. Wann ist also eine Tür unsicher? Eine Tür d ist nur dann unsicher, wenn es $r_L(d)$ und $r_H(d)$ gibt mit $r_H(d) \prec_d r_L(d)$, denn dann ist ganz G unerreichbar.

Zur Berechnung von $r_L(d)$ und $r_H(d)$ müssen die *nicht-redundanten* CW- bzw. CCW-Komponenten berechnet werden. Nicht-redundante Komponenten sind solche, die in keiner anderen Komponente enthalten sind. Diese lassen sich mit Hilfe einer sortierten Liste der kritischen Punkte in $O(n)$ Zeit berechnen (zur Erinnerung, die kritischen Punkte selbst kann man mit der Datenstruktur aus [5] in Zeit $O(n\log n)$ berechnen). Angenommen, wir wollen $r_L(d)$ berechnen. Dazu wählen wir einen beliebige kritischen Punkt k_0 als Ursprung, sortieren die kritischen Punke nach der Vorgängerrelation bzgl. k_0 und speichern sie in einer Liste SL. Mit Hilfe einer weiteren Liste L und eines Index-Arrays A kann man nun die nicht-redundanten CW-Komponenten in $O(n)$ Zeit bestimmen. Man beginnt einen Uhrzeigersinnumlauf beginnend bei k_0. L soll eine doppelt verkettete Liste sein, in die alle besuchten reflexen Ecken eingefügt werden, deren Backward-Extensions noch nicht besucht worden sind (das sind gerade die Kandidaten für $r_L(d)$).

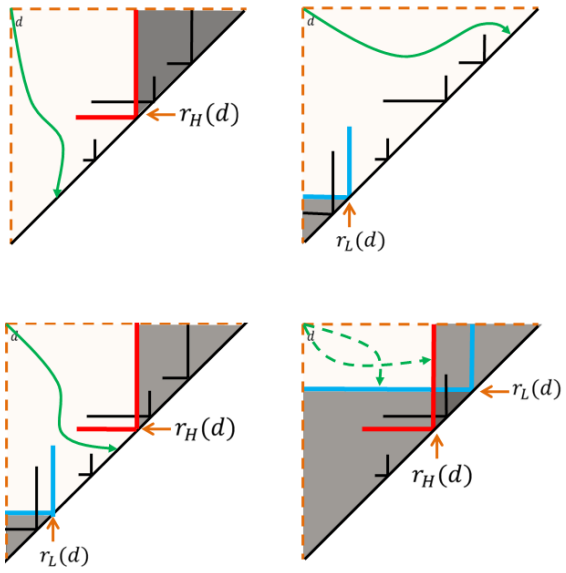

Abbildung 18: Fallunterscheidung bei $r_L(d)$ und $r_H(d)$.

Die Elemente des Arrays A enthalten einerseits einen Zeiger $A[i].p$ in die Kandidatenliste L und andererseits einen Status $A[i].s$, der angibt, ob es sich um eine redundante, nicht-redundante oder noch unbekannte Komponente handelt, $i = 1, ..., |r|$. Man initialisiert für alle reflexen Ecken i den Status $A[i].s$ als unbekannt und setzt den Zeiger $A[i].p$ auf $NULL$. Zur Initialisierung von L druchmustert man die sortierte Liste SL von links nach rechts, und wenn der gerade betrachtete kritische Punkt k eine reflexe Ecke r ist, dann hängt man r ans Ende von L an und aktualisiert den Zeiger $A[r].p$. Wenn es sich bei k um eine Backward-Extension von r handelt und r schon in L enthalten ist, d. h. $A[r].p \neq NULL$, dann entfernt man r aus L und setzt $A[r].p = NULL$. Nach dieser Initialisierung von L und A (d. h. nach einem Uhrzeigersinnumlauf bzw. einer Durchmusterung von SL von links nach rechts) durchmustert man SL erneut. Wenn der gerade betrachtete kritische Punkt k eine reflexe Ecke r mit noch unbekannten Status $A[r].s$ ist, so fügt man r ans Ende von L an und aktualisiert den Zeiger $A[r].p$. Wenn es sich bei k um eine Backward-Extension von r handelt und r noch unbekannten Status hat, so benutzt man den Zeiger $A[r].p$ für den Zugriff auf r in L und greift daraufhin mit dem Rückwärtszeiger der doppelt verketteten Liste L auf alle Elemente i vor r in L zu, um ihren Status $A[i].s$ als redundant zu markieren und die Elemente selbst aus L zu entfernen. r dagegen wird nun als nicht-redundant markiert. Die Initialisierung kostet einen Uhrzeigersinnumlauf, die Aktualisierung einen weiteren, also $O(n)$ Zeit. Die Ermittlung der CCW-Komponenten erfolgt analog, man läuft dazu ledig-

lich gegen den Uhrzeigersinn statt im Uhrzeigersinn und behandelt die Entdeckung von Forward-Extensions statt der Backward-Extensions. Hat man die nicht-redundanten Komponenten ermittelt, kann man leicht $r_L(d)$ und $r_H(d)$ berechnen. Zur Berechnung von $r_L(d)$ genügt nämlich ein Schnitttest der nicht-redundanten CW-Komponenten (das sind horizontale Linien im SVD) mit der vertikal verlaufenden Seite des LVD(d). Die CW-Komponente, die das LVD(d) schneidet und deren reflexe Ecke zu d im Uhrzeigersinn maximalen Abstand hat, ist das gesuchte $r_L(d)$. Entsprechend testet man für die Berechnung von $r_H(d)$ die nicht-redundanten CCW-Komponenten (das sind vertikale Linien im SVD) auf Schnitt mit der horizontalen Seite des LVD(d). Um Sackgassen wie in Abbildung 19 zu entdecken, d. h. solche, die durch den Schnitt je einer horizontalen und einer vertikalen Linie im SVD entstehen, für die nicht-redundante Komponenten verantwortlich sind, muss man den Schnitttest lediglich auf benachbarte horizontale und vertikale Linien im GVD ausweiten, die durch nicht-redundante Komponenten entstehen.

Für den folgenden Algorithmus berechnen wir als Vorverarbeitungsschritt die offensichtlich unsicheren Segmente auf S, die zugleich die offensichtlich unerreichbaren Segmente auf G sind, damit sind diejenigen gemeint, die von Sackgassen wie in Abbildung 19 berandet sind. Alle anderen potentiell sicheren Segmente auf S gruppieren wir nach gleichem $r_L(d)$ und speichern die Gruppen in einer Liste Rep_S, wobei die Speicherung jedes potentiell sicheren Segments einer Gruppe mit Hilfe seines Repräsentanten erfolgt. Segmente auf S, für die es kein $r_L(d)$ gibt, sind trivialerweise sicher, da für jedes d auf solch einem Segment auf jeden Fall ein unteres Segment von G im LVD(d) erreicht werden kann (vgl. Abbildung 18). Entsprechend merken wir uns in einer Liste Sec_G alle Segmente auf G, für die der Test auf die oben beschriebene offensichtliche Unerreichbarkeit negativ ausfällt, in anderen Worten, die also erreichbar sind. Die Segmente in Sec_G speichern wir als Paare von Anfangs- und Endpunkten.

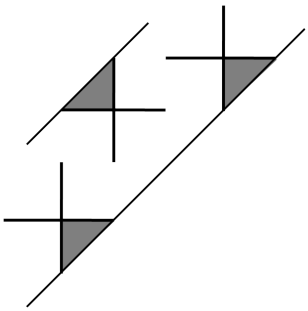

Abbildung 19: Offensichtliche Sackgassen im globalen V-Diagramm.

3.3.2 Algorithmus

```
// pSecG zeigt auf das erste Element in Sec_G
// pRepS zeigt auf die erste Gruppe in Rep_S
// d soll eine potentielle Tür sein
// d_R ist ein Element der Gruppe, auf die pRepS zeigt

1: WHILE (pRepS ≠ NULL) {
2:     IF (pSecG.EndPoint ≺_d pRepS.r_L(d)) {
3:         pSecG := pSecG.Next
4:         IF (pSecG = NULL) {
5:             „Alle übrigen Segmente in den Gruppen beginnend
6:             bei pRepS sind unsicher.“
7:             BREAK
8:         }
9:     }
10:    ELSE {
11:        IF (pSecG.StartPoint ≺_d pRepS.r_L(d)) x := pRepS.r_L(d)
12:        ELSE x := pSecG.StartPoint
13:        FOR EACH (d_R) {
14:            IF (x ≺_{d_R} r_H(d_R) OR r_H(d_R) = NULL) „Das durch d_R repr.
15:            Segment ist sicher.“
16:            ELSE „Das durch d_R repr. Segment ist unsicher.“
17:        }
18:        pRepS := pRepS.Next
19:    }
20: }
```

Der Algorithmus unterhält zwei Zeiger. Mit dem Zeiger $pSecG$ werden die Elemente von Sec_G durchlaufen, mit dem Zeiger $pRepS$ die Elemente von Rep_S (nach $\prec_d$-Relation sortiert). $pRepS$ zeigt jeweils auf eine Gruppe von Punkten in Rep_S, die dasselbe $r_L(d)$ haben. Beginnend bei der ersten Gruppe läuft die Hauptschleife über alle Gruppen in Rep_S. Die IF-Schleife aus Zeile 2, dient dazu, so lange $pSecG$ zu traversieren, bis der Endpunkt des gerade betrachteten Zielrandstücks von G bzgl. der Vorgängerrelation hinter dem $r_L(d)$ der gerade betrachteten Gruppe von potentiellen Türen liegt. Ein solches Stück ist potentiell erreichbar. Die tatsächliche Erreichbarkeit hängt aber von der Reihenfolge von r_L und r_H ab. Ist so ein Stück gefunden, dann greift die ELSE-Schleife aus Zeile 10. Diese prüft jeden Repräsentanten d_R der aktuellen Gruppe auf das Auftreten der Reihenfolge $r_L(d) = r_L(d_R) \prec_{d_R} r_H(d_R)$. Falls diese Reihenfolge erkannt wird, ist d_R sicher und ansonsten

unsicher. Falls es kein $r_H(d_R)$ gibt, ist d_R trivialerweise sicher, da dann ein oberes Segment des LVD(d_R) mit Sicherheit erreicht werden kann. Zur Veranschaulichung sei hier erneut auf Abbildung 18 verwiesen.

Das Bestimmen und Sortieren der kritischen Punke, sowie die Ermittlung von Rep_S und Sec_G ist in $O(n\log n)$ möglich. Sowohl das Traversieren von $pSecG$ als auch von $pRepS$ und seiner Elemente liegt in $O(n)$ (es geschieht nur einmal). Daher folgt das Theorem:

Theorem 5 *Sei P ein Polygon mit n Ecken. In $O(nlogn)$ Zeit kann man mit zwei Wächtern alle sicheren Türen $d \in \partial P$ bestimmen.*

4 Zwei Wächter vs Boundary 1-Searcher

Unter Verwendung des Boundary 1-Searchers muss, wie der Text zu Abbildung 8 schon andeutet, eine Konvention getroffen werden. Und zwar muss man festlegen, welche Achse im V-Diagramm die Position des Boundary 1-Searchers repräsentieren soll. Die andere Achse repräsentiert folglich die Position des Laserpunktes. Wir legen o.B.d.A. die Ordinate als die Achse fest, durch die die Position des Boundary 1-Searchers repräsentiert wird und damit zugleich die Abszisse als die Achse, die die Position des Laserpunktes darstellt.

Auf den ersten Blick scheint sich der Boundary 1-Searcher von den zwei Wächtern nicht zu sehr zu unterscheiden. Er bewegt sich stetig entlang des Randes, er hält den Bereich, in dem sich die Tür befindet sauber, er macht einen Eindringling ausfindig, wenn dieser seinen Laserstrahl kreuzt. Doch gibt es Räume, an denen die zwei Wächter bei ihrer Suche scheitern, der Boundary 1-Searcher aber Erfolg hat. Doch wie kommt es, dass der Boundary 1-Searcher den zwei Wächtern überlegen ist? Seine Überlegenheit ist in der Tatsache begründet, die in Abschnitt 2.4 beschrieben ist: Legale Sprünge des Laserpunkts erlauben dem Boundary 1-Searcher im V-Diagramm das Kreuzen von $\mathcal{V}_i$, wenn auch (nach obiger Konvention) nur von rechts nach links (vgl. Abbildung 8). Dass das ein entscheidender Vorteil sein kann, soll folgendes Beispiel verdeutlichen:

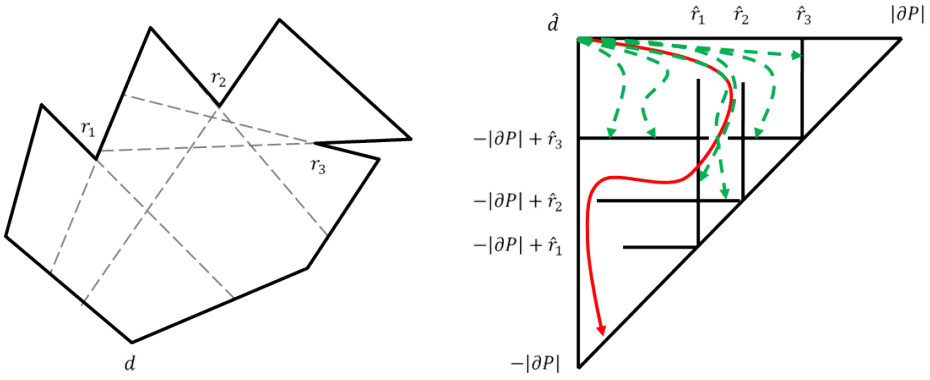

Abbildung 20: Ein Polygon $P(d)$ und das zugehörige LVD(d). $P(d)$ kann von zwei Wächtern nicht durchsucht werden, sehr wohl aber von einem Boundary 1-Searcher.

Alle Punkte von G sind für die zwei Wächter unerreichbar. Jeder Pfad der zwei Wächter (grün-gestrichelt) scheitert an einer Barriere im V-Diagramm. Das trifft aber nicht auf den Boundary 1-Searcher zu. Für ihn existiert ein

Suchpfad (rot). Er führt auf seinem Weg an der reflexen Ecke r_1 lediglich einen legalen Laserpunktsprung durch und kann so die Zielgerade erreichen.

Es ist klar, dass alle für die zwei Wächter gemachten Beobachtungen auch auf den Boundary 1-Searcher zutreffen, eben mit der einen Ausnahme, dass es für ihn erlaubt ist, dunkelgraue Bereiche im V-Diagramm von rechts nach links zu kreuzen. Da der Boundary 1-Searcher während der Suche dieselben Konfigurationen annehmen kann wie die zwei Wächter, kann man auch für ihn die zwei Theoreme folgern:

Theorem 6 *Sei P ein Polygon mit n Ecken. Sei $d \in \partial P$ eine Tür. Ob d sicher ist, kann mit einem Boundary 1-Searcher in $O(nlogn)$ Zeit getestet werden.*

Theorem 7 *Sei P ein Polygon mit n Ecken. In $O(nlogn)$ Zeit kann man mit einem Boundary 1-Searcher alle sicheren Türen $d \in \partial P$ bestimmen.*

5 Abschließende Bemerkungen

In der Ausarbeitung wurden zwei mögliche Strategien vorgestellt, um ein gegebenes Polygon P mit n Ecken und einer Tür in Randpunkt d nach Eindringlingen zu durchsuchen: Die Suche mit zwei Wächtern und die Suche mit dem Boundary 1-Searcher. Es wurden die für beide Strategien äußerst hilfreichen V-Diagramme und ihre Variationen vorgestellt. Wir haben der Begriff der Durchsuchbarkeit eines Polygons / Sicherheit einer Tür eingeführt und ein Verfahren erläutert, mit dem es in $O(n\log n)$ Zeit möglich ist, eine Tür d auf Sicherheit zu testen. Die naheliegende Erweiterung dieses Verfahrens zur Lokalisierung aller sicheren Türen mit Hilfe der $O(n)$ vielen Repräsentanten liegt in $O(n^2\log n)$. Dass die Lokalisierung aller Türen aber auch in $O(n\log n)$ möglich ist, zeigt der Algorithmus aus Abschnitt 3.3.2, der selbst in $O(n)$ Zeit läuft, aber $O(n\log n)$ Vorbereitunszeit benötigt. Wir haben die zwei Wächter dem Boundary 1-Searcher gegenübergestellt, mit dem Ergebnis, dass der Boundary 1-Searcher den zwei Wächtern aufgrund seiner Fähigkeit, den Laserpunkt von einer Kante auf eine nicht angrenzende andere „springen“ zu lassen, überlegen ist, mit anderen Worten, es gibt Polygone die von dem Boundary 1-Searcher durchsucht werden können, von den zwei Wächtern aber nicht. Da der Boundary 1-Searcher bei seiner Suche jedoch insbesondere all die Konfigurationen einnehmen kann, die auch die zwei Wächter einnehmen können, konnten wir schließen, dass die für die zwei Wächter beobachteten Ergebnisse, was Durchsuchbarkeit und Laufzeiten angeht, auch für den Boundary 1-Searcher gelten müssen.

Die Ausarbeitung gibt in großen Teilen die im Paper „Where to Build a Door“ [1] erarbeiteten Verfahren und erzielten Ergebnisse wieder. Die meisten der beigefügten Abbildungen basieren auf Abbildungen aus dieser Quelle, einige wurden lediglich überarbeitet oder ergänzt. Abschließend sei an dieser Stelle noch erwähnt, dass man die Bestimmung aller sicheren Türen von $O(n\log n)$ auf $\mathrm{O}(n)$ Zeit senken kann. Das wurde in der Dissertation von Z. Zhang [2] unter Zuhilfenahme der so genannten *LR-Sichtbarkeit* gezeigt.

Literatur

[1] Z. Zhang, T. Kameda. Where to Build a Door. *Proc. IEEE/RSJ Intn'l Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[2] Z. Zhang. Applications of Visibility Space in Polygon Search Problems. *Ph. D. dissertation, Simon Fraser University*, Seiten 30–62, 2005.

[3] T. Kameda, Z. Zhang, M. Yamashita. Simple Characterization of Polygons Searchable by a Boundary 1-Searcher. *Proc. Canadian Conf. on Computational Geometry*, 2006.

[4] T. Kameda, M. Yamashita, I. Suzuki. On-Line Polygon Search by a Six-State Boundary 1-Searcher. *Technical Report CMPT-TR 2003-07, School of Computing Science, SFU*, 2003.

[5] B. Chazelle, L. Guibas. Visibility and Intersection Problems in Plane Geometry. *Discrete & Computational Geometry*, Seiten 551-557, Springer, 1989.