Relokalisation – Ein theoretischer Ansatz in der Praxis*

Oliver Karch Hartmut Noltemeier Mathias Schwark Thomas Wahl

Lehrstuhl für Informatik I
Universität Würzburg
Am Hubland, 97074 Würzburg
Email: {karch,noltemei,schwark,wahl}@informatik.uni-wuerzburg.de
WWW: http://www-info1.informatik.uni-wuerzburg.de/

Zusammenfassung Wir betrachten das folgende Problem: Ein Roboter befindet sich an einer ihm unbekannten Position innerhalb eines Gebäudes und hat die Aufgabe, eine komplette *Relokalisation* durchzuführen, d.h. seine Position und Orientierung zu bestimmen. Dies ist beispielsweise notwendig, wenn der Roboter nach einer Störung (z.B. einem Stromausfall oder Wartungsarbeiten) wieder selbständig starten soll und die Möglichkeit besteht, daß er in der Zwischenzeit bewegt wurde.
Eine idealisierte Variante dieses Problems, bei der der Roboter einen Entfernungssensor und einen Kompaß besitzt, die jeweils exakte, unverrauschte Daten liefern, sowie eine (exakte) polygonale Umgebungskarte, wurde von Guibas *et al.* gelöst.
Wir beschreiben eine Möglichkeit, diesen theoretischen Ansatz so zu modifizieren, daß er auch in realistischen Umgebungen (beispielsweise bei ungenauer Sensorik oder ungenauer Karte) eingesetzt werden kann.

1 Problemstellung

Wir betrachten die erste Phase des *Lokalisationsproblems* [3]: Ein autonomer Roboter befindet sich an einer ihm unbekannten Position innerhalb eines Gebäudes. Seine Aufgabe besteht darin, seine Position und Orientierung ohne jegliches Vorwissen über frühere Positionen zu bestimmen. Dies ist beispielsweise notwendig, wenn der Roboter nach einer Störung (z.B. einem Stromausfall oder Wartungsarbeiten) wieder selbständig starten soll und die Möglichkeit besteht, daß er in der Zwischenzeit (von außen) bewegt wurde.

Der Roboter verfügt hierzu über eine Umgebungskarte und einen Entfernungssensor (z.B. einen Laser-Scanner) und soll die Lokalisation auch nur mit dieser Minimalausstattung an Sensoren bewerkstelligen; insbesondere dürfen also auch keine Ortsmarken (z.B. Markierungen an den Wänden oder auf dem Boden) benutzt werden. Auf diese Weise soll ein autonomer Roboter auch in Umgebungen eingesetzt werden können, in denen bauliche Veränderungen verboten oder zu kostenintensiv sind.

* Diese Arbeit wird von der Deutschen Forschungsgemeinschaft (DFG) gefördert, Projektnr. No 88/14-1.

In der Regel erfolgt die Lösung dieses Problems in zwei Phasen: Zunächst generiert der *stillstehende* Roboter alle hypothetischen Positionen in der Karte, die konsistent mit seinen Sensordaten sind. Falls die Karte gleichartige Bereiche an verschiedenen Stellen enthält (z.B. in Gebäuden mit vielen identischen Korridoren, wie in Krankenhäusern oder Lagerhallen), kann es mehrere potentielle Positionen geben. Diese besitzen alle dasselbe Sichtbarkeitspolygon und können von einem stillstehenden Roboter nicht unterschieden werden. Abbildung 2 zeigt ein Beispiel: Die Standorte am unteren Ende der beiden äußeren Nischen können aufgrund ihres Sichtbarkeitspolygons nicht unterschieden werden.

In einem solchen Fall *bewegt* sich der Roboter dann in einer zweiten Phase innerhalb seiner Umgebung, um die Mehrdeutigkeiten zu eliminieren. Dies ist ein typisches online-Problem, da der Roboter die Informationen online verarbeiten muß, um einen möglichst effizienten (d.h. kurzen) Weg zu bestimmen, auf dem die falschen Hypothesen eliminiert werden können. Dudek *et al.* [4] zeigten, daß die Bestimmung einer optimalen Lokalisierungsstrategie NP-hart ist, und stellten gleichzeitig eine kompetitive Greedy-Heuristik vor, deren Laufzeit von Schuierer [9] noch verbessert werden konnte.

Dieser Artikel konzentriert sich auf die erste Phase der Lokalisation, d.h. auf das Generieren der möglichen Roboterstandorte. Unter den zusätzlichen Voraussetzungen, daß der Roboter seine Orientierung bereits kennt (d.h. einen *Kompaß* besitzt) und sowohl alle Sensoren als auch die Karte *exakt* (d.h. ohne jegliche Verrauschungen) sind, erhalten wir das folgende geometrische Problem: Für ein gegebenes Kartenpolygon $\mathcal{P}$ und ein sternförmiges[1] Polygon $\mathcal{V}$ (das Sichtbarkeitspolygon des Roboters) sollen alle Punkte $p \in \mathcal{P}$ gefunden werden, deren Sichtbarkeitspolygon gleich $\mathcal{V}$ ist.

In [5] beschreiben Guibas *et al.* ein Verfahren zur effizienten Lösung des Lokalisationsproblems in obiger idealisierter Form, welches wir im folgenden Abschnitt kurz skizzieren wollen. Da diese Vorgehensweise eine exakte Sensorik, eine exakte Karte sowie einen Kompaß voraussetzt, ist das Verfahren nicht direkt in die Praxis übertragbar, wo die Daten im allgemeinen *fehlerbehaftet* sind. In Abschnitt 3 behandeln wir deshalb die auftretenden Probleme und stellen in den Abschnitten 4 und 5 einen Ansatz vor, diese zu umgehen. Hierbei werden Distanzfunktionen benutzt, um die Ähnlichkeiten zwischen den verrauschten Scans (des Entfernungssensors) und den im Preprocessing (aus der möglicherweise ungenauen Karte) bestimmten Sichtbarkeitsskeletten zu modellieren.

2 Ein theoretischer Ansatz

Im folgenden skizzieren wir das Verfahren von Guibas *et al.*, das als Grundlage für den in Abschnitt 4 vorgestellten Ansatz dient. Die Problemstellung lautet: Gegeben sind ein Polygon $\mathcal{P}$ (die *Umgebungskarte*) und ein sternförmiges Polygon $\mathcal{V}$ (das *Sichtbarkeitspolygon* des Roboters). Wir nehmen hierbei an, daß der

[1] Ein Polygon heißt *sternförmig*, wenn es (mindestens) einen Punkt gibt, von dem aus das komplette Polygoninnere einsehbar ist. Jeder solche Punkt ist ein *Kernpunkt*, und die Gesamtheit dieser Punkte ist der *Kern* des Polygons.

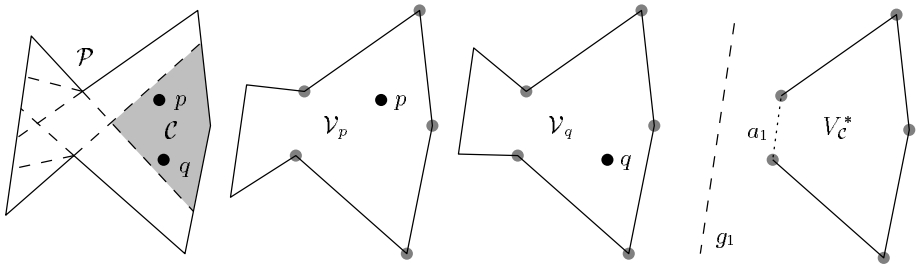

Abbildung 1. Zerlegung einer Karte in Sichtbarkeitszellen (links), zwei Sichtbarkeitspolygone (Mitte) und das zugehörige Skelett (rechts)

Roboter im Koordinatenursprung von $\mathcal{V}$ plaziert ist. Unter der Voraussetzung, daß die Orientierung von $\mathcal{V}$ bezüglich $\mathcal{P}$ schon bekannt ist, sind dann alle Punkte $p \in \mathcal{P}$ gesucht, deren Sichtbarkeitspolygon $\mathcal{V}_p$ gleich $\mathcal{V}$ ist.

Die zentrale Idee von Guibas *et al.* [5] zur Lösung dieses Problems ist, die Karte in *endlich* viele Sichtbarkeitszellen zu zerlegen, so daß sich das sog. Sichtbarkeitsskelett, das gewissermaßen eine Vergröberung des Sichtbarkeitspolygons darstellt, innerhalb einer Sichtbarkeitszelle nicht ändert.

Bei einer Lokalisationsanfrage wird dann das vom Sichtbarkeitspolygon des Roboters induzierte Skelett bestimmt und nur nach denjenigen Zellen gesucht, die dasselbe Skelett besitzen. Das kontinuierliche[2] Problem, ein Sichtbarkeitspolygon in die Karte einzupassen, wird dadurch auf natürliche Weise diskretisiert.

2.1 Die Zerlegung in Sichtbarkeitszellen

In einem Preprocessingschritt wird die Karte $\mathcal{P}$ durch Einfügen von Halbstrahlen in konvexe *Sichtbarkeitszellen* zerlegt, die die folgende Eigenschaft besitzen:

> Die Menge der sichtbaren Kartenecken ändert sich innerhalb einer Sichtbarkeitszelle nicht.

Die Sichtbarkeit einer Kartenecke ändert sich nur beim Überschreiten eines Halbstrahls, der durch diese Ecke und eine Reflexecke (mit Innenwinkel $\geq 180°$) induziert wird, welche die Kartenecke verdeckt. Abbildung 1 demonstriert diese Zerlegung in Sichtbarkeitszellen an einem einfachen Beispiel. Die Punkte p und q befinden sich in derselben Sichtbarkeitszelle $\mathcal{C}$ und sehen dieselben fünf grau markierten Ecken. Abbildung 2 zeigt eine Zerlegung für eine etwas kompliziertere Karte mit drei Hindernissen (grau gezeichnet), erzeugt mit unserer Software RoLoPro (siehe Abschnitt 6).

Besteht die Karte aus n Ecken, davon r Reflexecken, dann gibt es $\mathcal{O}(nr)$ solche Halbstrahlen und die Komplexität der entstehenden Zerlegung ist in $\mathcal{O}(n^2r^2)$, wobei auch leicht worst case-Szenen angegeben werden können, die zeigen, daß diese Schranken scharf sind.

[2] „Kontinuierlich“ in dem Sinne, daß sich kein $\varepsilon > 0$ finden läßt, so daß sich das Sichtbarkeitspolygon $\mathcal{V}_p$ eines sich um maximal ε bewegenden Punktes p nicht ändert.

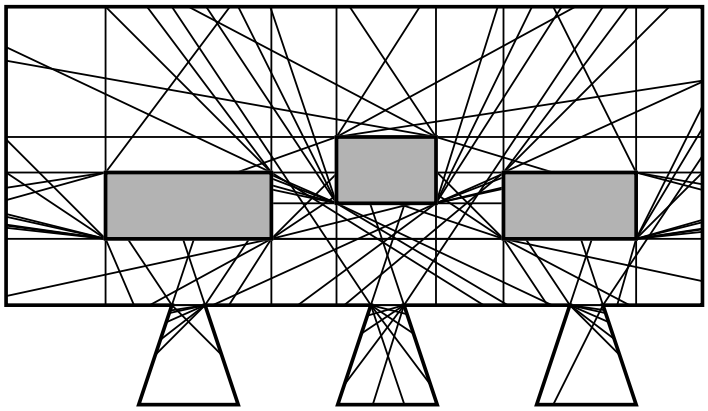

Abbildung 2. Sichtbarkeitszellenzerlegung einer komplizierteren Karte

2.2 Das Sichtbarkeitsskelett

Die Sichtbarkeitspolygone zweier Punkte aus derselben Zelle unterscheiden sich nur in einigen wenigen Stellen (siehe Abbildung 1), nämlich in den durch Verdeckungen induzierten *Scheinkanten* und den dazwischen liegenden *teilweise sichtbaren* Kanten der Karte. Die übrigen *echten* (d.h. komplett sichtbaren) Kanten sind in beiden Polygonen identisch (siehe die beiden Sichtbarkeitspolygone in Abbildung 1). Diese Beobachtung führt zur Definition einer Struktur, die sich innerhalb einer Zelle nicht ändert, des Sichtbarkeitsskeletts.

Für ein Sichtbarkeitspolygon $\mathcal{V}_p$ eines Punktes p erhält man das zugehörige *Sichtbarkeitsskelett* V_p^* durch Entfernen aller Scheinkanten, die kolinear zum Betrachter p liegen, und der dazwischen liegenden teilweise sichtbaren Kanten. Für jede teilweise sichtbare Kante wird eine *künstliche Kante* a_i zu V_p^* hinzugefügt, zusammen mit der Geraden g_i, auf der die ursprüngliche (teilweise sichtbare) Kante von $\mathcal{V}_p$ liegt.

Das so definierte Skelett ändert sich innerhalb einer Zelle nicht. Deshalb definieren wir als $V_\mathcal{C}^*$ das gemeinsame Skelett aller Punkte aus der Zelle $\mathcal{C}$. Abbildung 1 (rechts) zeigt als Beispiel das Skelett $V_\mathcal{C}^*$ der grau gefärbten Zelle $\mathcal{C}$.

3 Probleme in realistischen Szenarien

Die idealisierenden Annahmen, die das im letzten Abschnitt beschriebene Verfahren macht, verhindern seinen direkten Einsatz in realistischen Szenarien, in denen sich folgende Probleme ergeben:

- Ein realer Entfernungssensor liefert kein Sichtbarkeits*polygon* $\mathcal{V}$, sondern einen Scan $\mathcal{S}$, d.h. eine *endliche Folge* von Entfernungsmeßwerten. Zudem liegen die Scanpunkte nicht exakt auf dem Sichtbarkeitspolygon, sondern

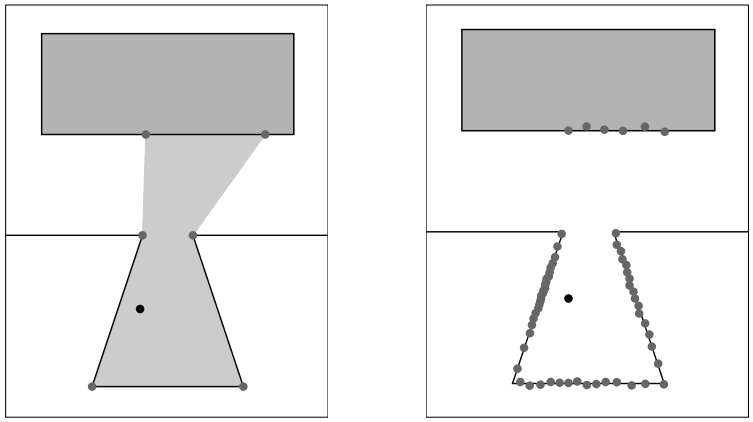

Abbildung 3. Exaktes Sichtbarkeitspolygon (links) und verrauschter Scan (rechts) für einen Roboter in der linken der drei Nischen von Abbildung 2

sind mit *Ungenauigkeiten* behaftet (siehe Abbildung 3). Selbst wenn wir diese Punkte durch Strecken verbinden, erhalten wir nur eine Approximation $\mathcal{V}_\mathcal{S}$ des exakten Sichtbarkeitspolygons $\mathcal{V}$.

- Die im letzten Abschnitt erwähnte Kompaßvoraussetzung wird von realen Robotern ebenfalls nicht erfüllt bzw. nur mit zu geringer Genauigkeit.
- In der Einsatzumgebung des Roboters können sich Hindernisse befinden, die in der Karte nicht verzeichnet sind und die seine Sicht beeinträchtigen. Dies können beispielsweise Hindernisse sein, die zu klein sind, um in die Karte aufgenommen zu werden (Stühle, Tische usw.), oder auch dynamische Hindernisse wie Menschen oder Roboter.

Obige Probleme haben zur Folge, daß das (approximierte) Sichtbarkeitsskelett $V^*_\mathcal{S}$, welches der Roboter aus dem approximierten Sichtbarkeitspolygon $\mathcal{V}_\mathcal{S}$ berechnet, in der Regel mit keinem der im Preprocessing berechneten Skelette exakt übereinstimmt. Deshalb ist der Roboter nicht in der Lage, den Scan einem der Skelette zuzuordnen, und die Lokalisationsanfrage liefert kein Ergebnis und schlägt fehl.

4 Anpassung des Verfahrens an die Praxis

Unser Ansatz zur Umgehung dieser Probleme ist, für einen gegebenen Scan $\mathcal{S}$ dasjenige Skelett zu suchen, das dem Scan am *ähnlichsten* ist, d.h. die ursprüngliche *Exact Match*-Anfrage durch eine *Nearest Neighbor*-Anfrage in der Menge der Skelette zu ersetzen. Die Ähnlichkeit zwischen einem Scan $\mathcal{S}$ und einem Skelett V^* soll hierbei durch eine geeignete Distanzfunktion $d(\mathcal{S}, V^*)$ modelliert werden.

Je nachdem, welche Distanzfunktion verwendet wird, muß auf den Scan und das so ermittelten Skelett zusätzlich noch ein Matching-Algorithmus angewandt

werden, um die Roboterposition zu bestimmen. Der Grund hierfür ist, daß nicht alle Verfahren zur Distanzberechnung auch ein optimales Matching liefern. Beispielsweise ermittelt der Algorithmus zur Berechnung der *Arkin-Metrik* [2] für Polygone nur den optimalen Rotationswinkel zwischen den Polygonen, aber keinen Translationsvektor. Im Gegensatz dazu bestimmen Algorithmen zur Berechnung der *Minimalen Hausdorff-Distanz* (bezüglich Euklidischer Transformationen) [1] sowohl den eigentlichen Funktionswert als auch das zugehörige Matching.

4.1 Forderungen an eine Distanzfunktion

Damit sich eine Distanzfunktion $d(\mathcal{S}, V^*)$ in der Praxis bewährt, sollte sie mindestens die folgenden Eigenschaften besitzen:

Stetigkeit Die Distanz sollte stetig sein in dem Sinne, daß kleine Veränderungen des Scans (beispielsweise verursacht durch fehlerbehaftete Sensoren) oder des Skeletts (z.B. aufgrund einer ungenauen Karte) auch nur kleine Änderungen der Distanz bewirken.

Dies wird inbesondere dadurch motiviert, daß bei der ursprünglichen Methode gerade die Einteilung der Kanten des Sichtbarkeitspolygons in verschiedene Typen (Scheinkanten, teilweise sichtbare Kanten usw.) das Verfahren anfällig für Störungen macht: Schon eine geringe Verschiebung eines Eckpunktes kann den Typ einer Kante ändern. Dies hat zur Folge, daß das zum Sichtbarkeitspolygon gehörende Skelett sich sogar strukturell ändert und in der Regel keinem der im Preprocessing bestimmten Skelette zugeordnet werden kann. In diesem Sinne kann die Exact Match-Anfrage des ursprünglichen Verfahrens auch als diskrete Distanz zwischen Polygon und Skelett aufgefaßt werden, die allerdings die Stetigkeit in krasser Weise verletzt, da sie nur zwei verschiedene Werte annimmt (0 – „paßt“ und 1 – „paßt nicht“).

Translationsunabhängigkeit Eine Verschiebung des Scans bzw. des Skeletts im jeweiligen lokalen Koordinatensystem sollte keine Veränderung der Distanz verursachen, da über die relative Lage der beiden Koordinatensysteme zueinander nichts bekannt ist. Diese zu bestimmen, ist ja gerade die Aufgabe des Lokalisationsalgorithmus'.

Rotationsunabhängigkeit Falls der Roboter keinen Kompaß besitzt, sollte die Distanz auch unabhängig gegenüber Rotationen des Scans bzw. des Skeletts sein.

Schnelle Berechenbarkeit Für eine Lokalisationsanfrage muß die Distanz mehrmals berechnet werden (jeweils für verschiedene Skelette, siehe Abschnitt 4.2). Deshalb sollte ihr Berechnungsaufwand nicht zu hoch sein.

Da wir bei einer Lokalisationsanfrage einen Scan nicht mit allen vorhandenen Skeletten vergleichen wollen (deren Zahl kann ja im worst case in $\Omega(n^2r^2)$ sein, siehe Abschnitt 2.1), sollten diese in einer geeigneten Datenstruktur gespeichert werden, um darin effizient suchen zu können.

4.2 Die Verwaltung der Skelette

Für die Verwaltung der Skelette bietet sich der *Monotone Bisektor-Baum* [8] an, ein räumlicher Index, mit dem die Skelette bezüglich einer weiteren Distanzfunktion $D(V_1^*, V_2^*)$, die die Ähnlichkeit zweier Skelette beschreibt, hierarchisch partitioniert werden können. Hierbei wird im Preprocessing die Menge der Skelette rekursiv in Cluster mit monoton fallenden Clusterradien aufgeteilt. Diese Aufteilung spiegelt dann die Ähnlichkeiten der Skelette untereinander wider.

Damit bei der Nearest Neighbor-Anfrage nicht immer der komplette Baum durchsucht werden muß, sollte die Distanz $D(V_1^*, V_2^*)$ „kompatibel" zur Distanz $d(\mathcal{S}, V^*)$ gewählt werden, d.h. es sollte die *Dreiecksungleichung*

$$d(\mathcal{S}, V_2^*) \leq d(\mathcal{S}, V_1^*) + D(V_1^*, V_2^*)$$

erfüllt sein. Auf diese Weise können beim Durchlaufen des Baumes die Distanzwerte $d(\mathcal{S}, V^*)$ ganzer Teilcluster von Skeletten nach unten abgeschätzt werden. Ein solcher Cluster kann dann komplett verworfen werden und ist nicht weiter zu durchsuchen.

5 Sinnvolle Distanzen für $d(\mathcal{S}, V^*)$

Distanzfunktionen zu finden, die alle Anforderungen aus Abschnitt 4.1 erfüllen, ist nicht einfach. Insbesondere die Forderung nach der schnellen Berechenbarkeit, die den drei übrigen Forderungen entgegen wirkt, schränkt die Wahl von $d(\mathcal{S}, V^*)$ stark ein.

Weiterhin ist es oftmals nicht möglich, schon vorhandene Distanzfunktionen (z.B. für Polygone) einfach zu übernehmen, da wir es in unserer Problemstellung mit Scans bzw. Skeletten und nicht mit Polygonen zu tun haben. Eine Anpassung der Distanz ist somit fast immer notwendig. Im folgenden wollen wir zwei mögliche Distanzfunktionen untersuchen, die Hausdorff-Distanz und die Polarkoordinaten-Metrik, und anhand derer die Probleme erläutern.

5.1 Die Hausdorff-Distanz

Für zwei Punktmengen $A, B \subset \mathbb{R}^2$ ist deren *Hausdorff-Distanz* $\delta(A, B)$ definiert als

$$\delta(A, B) := \max\{\vec{\delta}(A, B), \vec{\delta}(B, A)\}, \quad \text{wobei} \quad \vec{\delta}(A, B) := \sup_{a \in A} \inf_{b \in B} \|a - b\|$$

die gerichtete Hausdorff-Distanz von A nach B beschreibt und $\|\cdot\|$ die Euklidische Norm darstellt. Der Ausdruck $\vec{\delta}(A, B)$ steht demnach für den maximalen Abstand eines Punktes aus A zur Menge B.

Sei $\mathcal{T}$ die Menge aller Euklidischen Transformationen (d.h. Kombinationen von Translationen und Rotationen), dann ist die ungerichtete bzw. gerichtete *Minimale Hausdorff-Distanz* bezüglich dieser Transformationen definiert durch

$$\delta_{\min}(A, B) := \inf_{t \in \mathcal{T}} \delta(A, t(B)) \quad \text{und} \quad \vec{\delta}_{\min}(A, B) := \inf_{t \in \mathcal{T}} \vec{\delta}(A, t(B)) \ .$$

Man sieht leicht, daß die Minimale Hausdorff-Distanz stetig ist und (nach Definition) auch die zweite und die dritte Anforderung aus Abschnitt 4.1 erfüllt. Allerdings ist ihre Berechnung äußerst aufwendig. Nach [1] ist dies mit einem Zeitaufwand von $\mathcal{O}((ms)^4(m+s)\log(m+s))$ möglich, wenn m die Komplexität des Scans und s die des Skeletts bezeichnet. Dagegen kann die Hausdorff-Distanz ohne Minimierung verhältnismäßig schnell bestimmt werden [1], nämlich mit einem Zeitbedarf von $\mathcal{O}((m+s)\log(m+s))$. Die Stetigkeitseigenschaft geht hierdurch nicht verloren, allerdings müssen wir jetzt einen geeigneten Translationsvektor und einen Rotationswinkel „von Hand" wählen.

Ein naheliegender Ansatz für die Wahl des Translationsvektors ist, für einen Scan $\mathcal{S}$ und ein Skelett V^* denjenigen Vektor zu wählen, der den Ursprung des Scan-Koordinatensystems (d.h. den Roboterstandort) in die zum Skelett gehörende Sichtbarkeitszelle $\mathcal{C}_{V^*}$ (beispielsweise in ihren Schwerpunkt) verschiebt. Dies ist sinnvoll, weil ja nach Definition der Sichtbarkeitszellen gerade die innerhalb von $\mathcal{C}_{V^*}$ gelegenen Roboterstandorte das Skelett V^* induzieren. Die Folge ist natürlich, daß Zellen, die dasselbe Skelett V^* besitzen (beispielsweise die großen Zellen in den beiden äußeren Nischen in Abbildung 2), getrennt behandelt werden müssen, da die Distanz $d(\mathcal{S}, V^*)$ jetzt nicht nur von V^*, sondern auch von der Zelle selbst abhängt.[3] Außerdem erkennt man sofort, daß der Schnitt der Zellen auch leer sein kann, so daß es i.allg. keinen gemeinsamen Vektor für alle Zellen gibt. Der Fehler bei diesem Ansatz, verglichen mit der Minimalen Hausdorff-Distanz, ist natürlich um so größer, je größer die Zelle ist, in die der Scan plaziert werden muß.

Ein weiterer Punkt, den es zu beachten gilt, ist, daß ein Skelett (als Punktmenge aufgefaßt) i.allg. nicht beschränkt ist, da es für jede künstliche Kante eine Gerade enthält. Dies hat zur Folge, daß die gerichteten Distanzen $\vec{\delta}(V^*, \mathcal{S})$ und $\vec{\delta}_{\min}(V^*, \mathcal{S})$ fast immer den Wert ∞ besitzen (außer im trivialen Fall, wenn V^* gleich dem konvexen Kartenpolygon ist und keine künstlichen Kanten besitzt). Deshalb müssen zur Distanzberechnung entweder die Skelette modifiziert werden, oder es sind nur die gerichteten Distanzen $\vec{\delta}(\mathcal{S}, V^*)$ und $\vec{\delta}_{\min}(\mathcal{S}, V^*)$ verwendbar.

5.2 Die Polarkoordinatenmetrik

Eine Distanzfunktion, die die Besonderheiten unserer Problemstellung stärker berücksichtigt als die Hausdorff-Distanz, ist die *Polarkoordinatenmetrik* (kurz PKM) [10], die eine fundamentale Eigenschaft der Problemstellung ausnutzt: Alle auftretenden Polygone und Skelette sind *sternförmig* im folgenden Sinne, und wir können sogar jeweils einen Kernpunkt angeben:

[3] Die Notation $d(\mathcal{S}, V^*)$ ist dann etwas irreführend, da es zu einem Skelett V^* mehrere Zellen geben kann. Die korrekte Schreibweise wäre $d(\mathcal{S}, \mathcal{C}_{V^*})$, in der auch die Abhängigkeit der Distanz von der Zelle deutlich wird. Wir belassen es jedoch hier bei der intuitiv verständlicheren Form $d(\mathcal{S}, V^*)$.

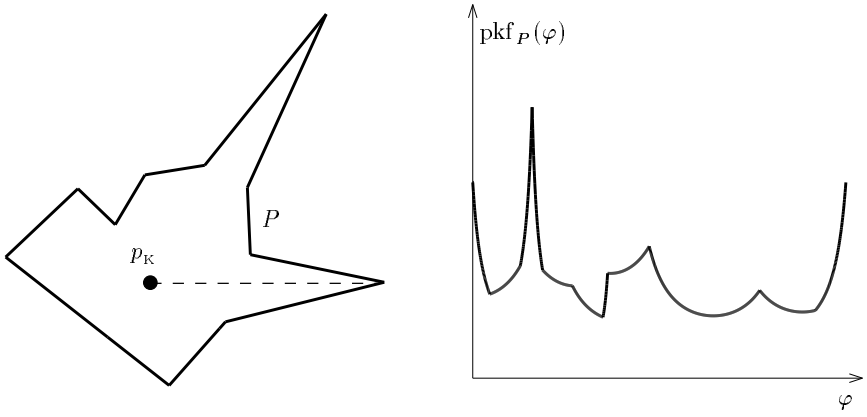

Abbildung 4. Die Funktion $\mathrm{pkf}_P(\varphi)$ für das Polygon P

- Das aus den Entfernungsdaten approximierte Sichtbarkeitspolygon $\mathcal{V}_S$ ist nach Konstruktion sternförmig mit dem Koordinatenursprung (Roboterstandort) als Kernpunkt.
- Jedes Skelett V^* ist in dem Sinne sternförmig, daß von jedem Punkt der zugehörigen Zelle $\mathcal{C}_{V^*}$ alle echten Kanten komplett sichtbar sind und für jede künstliche Kante a_i ein Teil der zugehörigen Geraden g_i sichtbar ist.

Zur Berechnung der PKM für zwei (sternförmige) Polygone P und Q mit Kernpunkten p_K und q_K definieren wir zunächst den Wert der *Polarkoordinatenfunktion*

$$\mathrm{pkf}_P(\varphi) : \mathbb{R} \to \mathbb{R}_{\geq 0}$$

als den Abstand des Kernpunktes p_K zum Schnittpunkt des von p_K in Richtung φ ausgehenden Halbstrahls mit dem Rand von P. Die Funktion $\mathrm{pkf}_P(\varphi)$ spiegelt demnach die Polarkoordinatendarstellung des Polygons P wider (mit p_K als Koordinatenursprung) und ist periodisch mit Periodenlänge 2π. Abbildung 4 zeigt ein Beispiel. In gleicher Weise definieren wir die Funktion $\mathrm{pkf}_Q(\varphi)$ für das Polygon Q.

Dann ist die PKM für die beiden Polygone P und Q definiert als die minimale Integraldistanz für die Funktionen pkf_P und pkf_Q im Intervall $[0, 2\pi]$ unter allen horizontalen Verschiebungen der beiden Funktionen (d.h. Rotationen der korrespondierenden Polygone) gegeneinander:

$$\mathrm{pkm}(P,Q) := \min_{t \in [0,2\pi]} \sqrt{\int_0^{2\pi} \left(\mathrm{pkf}_P(\varphi - t) - \mathrm{pkf}_Q(\varphi)\right)^2 \mathrm{d}\varphi}$$

Die Funktion pkf_P ist stetig in φ außer für den Spezialfall, wenn eine Polygonkante kolinear zu p_K liegt. Dann nämlich besitzt pkf_P am entsprechenden Winkel einen Sprung, der der Länge der kolinearen Kante entspricht. Weiterhin

is pkf_P auch stetig gegenüber Bewegungen der Polygonecken, sofern dieser Spezialfall nicht auftritt. Da jedoch pkf_P und pkf_Q jeweils nur endlich viele solcher Unstetigkeitsstellen aufweisen können, verschwinden diese beim Integrieren, d.h. die PKM ist, ebenso wie die Hausdorff-Distanzen, stetig gegenüber *allen* Bewegungen der Polygonecken von P und Q.

In [10] wurde gezeigt, daß die Funktion $\mathrm{pkm}(P,Q)$ eine Polygonmetrik ist, sofern die Kernpunkte *deterministisch* aus den Polygonen bestimmt werden, z.B. durch Wahl des dem Schwerpunkt nächstgelegenen Kernpunktes, der zudem translations- und rotationsinvariant ist. Weiterhin wurde eine lineare Approximation der PKM vorgestellt, die ebenfalls alle Metrikeigenschaften besitzt. Diese ist für unsere Anwendungen völlig ausreichend, und der Algorithmus zu ihrer Berechnung hat eine Laufzeit von $\mathcal{O}(pq \cdot (p+q))$ bzw. $\mathcal{O}(p+q)$, falls nicht über die Rotationen minimiert werden soll. Hierbei stehen p und q für die Komplexitäten der beiden Polygone.

Um die PKM als Distanzfunktion $d(\mathcal{S}, V^*)$ verwenden zu können, benötigen wir zwei zu $\mathcal{S}$ und V^* korrespondierende sternförmige Polygone:

- Für den Scan wählen wir das (nach Konstruktion sternförmige) approximierte Sichtbarkeitspolygon $\mathcal{V}_\mathcal{S}$. Als Kernpunkt kann wieder der Koordinatenursprung verwendet werden.
- Um aus einem Skelett V^* (mit zugehöriger Zelle $\mathcal{C}_{V^*}$) ein Polygon zu erhalten, wählen wir einen Punkt c innerhalb der Zelle $\mathcal{C}_{V^*}$ (z.B. deren Schwerpunkt) und bestimmen für diesen Punkt das Sichtbarkeitspolygon $\mathcal{V}_c$, für welches c dann auch ein Kernpunkt ist.

Als Distanzfunktion wählen wir dann $d(\mathcal{S}, V^*) := \mathrm{pkm}(\mathcal{V}_\mathcal{S}, \mathcal{V}_c)$ und erhalten eine Distanz, die alle Forderungen aus Abschnitt 4.1 (Stetigkeit, Translations-, Rotationsinvarianz, schnelle Berechenbarkeit) erfüllt.

6 Implementierung und erste experimentelle Ergebnisse

Sowohl der in Abschnitt 2 beschriebene Ansatz für exakte Sensorik als auch die in den beiden letzten Abschnitten vorgestellte Modifikation für realistische Szenarien wurden von uns in C++ unter Verwendung der LEDA-Klassenbibliothek [7] implementiert. Das ursprüngliche Verfahren wurde dabei an einigen Stellen modifiziert und vereinfacht, da unser Hauptaugenmerk nicht auf der exakten Implementierung der von Guibas *et al.* vorgeschlagenen (und teilweise sehr komplizierten) Datenstrukturen und Algorithmen lag. Stattdessen war unser Ziel ein Programm, das sich für alle Eingaben stabil verhält und das als Experimentierplattform und als Ausgangspunkt für eigene Modifikationen dienen kann. Die Folge ist natürlich, daß nicht mehr alle in [5, 6] gezeigten worst case-Schranken für Zeit und Speicherplatz eingehalten werden, da dies immensen Programmieraufwand erfordert hätte. Gleichwohl ist unsere Implementierung genügend effizient und robust. Abbildung 5 zeigt einen Screenshot unserer Software ROLOPRO (Roboter-Lokalisations-Programm), bei der gerade eine Anfrage für einen (simulierten) verrauschten Scan bearbeitet wird.

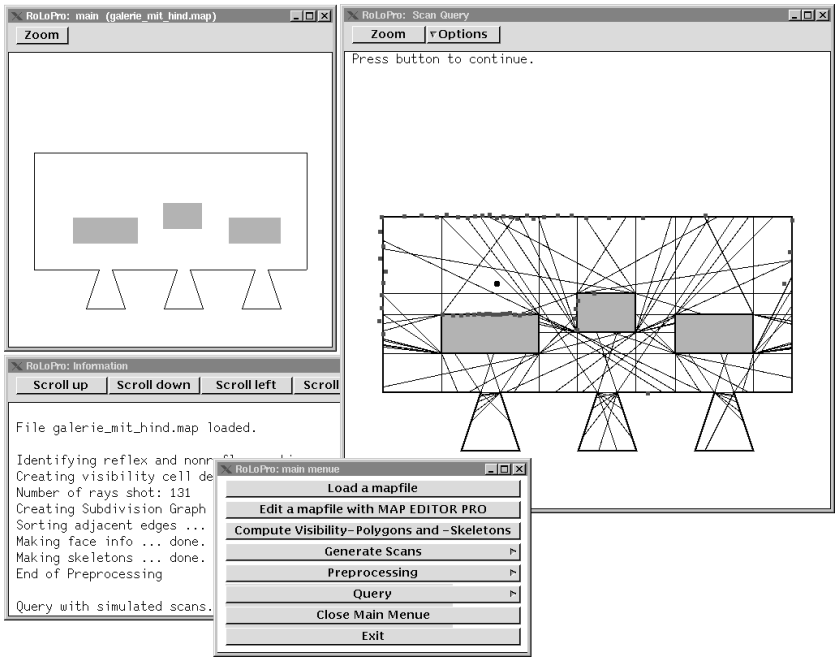

Abbildung 5. Screenshot von RoLoPro

Als Distanzfunktion $d(\mathcal{S}, V^*)$ wurden die in Abschnitt 5 beschriebene Hausdorff-Distanz und die Polarkoordinatenmetrik implementiert. Erste Tests in kleineren Szenen ergaben für die Hausdorff-Distanz eine Trefferrate von ca. 60%, d.h. bei etwa 60 von 100 Scans lag der Scanursprung in derjenigen Zelle, deren Skelett dem Scan am ähnlichsten war. Für die Polarkoordinatenmetrik ergab sich in denselben Szenen eine Trefferrate von ca. 90%.

7 Zukünftige Arbeiten

Der in den letzten Abschnitten beschriebene Ansatz wird von uns momentan ausgiebig in unserem Simulationssystem getestet und soll demnächst auch in zwei verschiedenen Szenarien für Serviceroboter in der Praxis erprobt werden.

Für die nähere Zukunft sind unsere wichtigsten Ziele

- die in Abschnitt 4.2 beschriebene hierarchische Verwaltung der Skelette zur Steigerung der Geschwindigkeit einer Lokalisationsanfrage. Als Distanzfunktion $D(V_1^*, V_2^*)$ bietet sich hier wieder die Polarkoordinatenmetrik an, da sie auch die geforderte Dreiecksungleichung erfüllt;
- die Untersuchung weiterer Distanzfunktionen bzw. deren Modifikation. So gibt es beispielsweise für die Wahl des in Abschnitt 5.1 beschriebenen Translationsvektors oder die Wahl der in Abschnitt 5.2 beschriebenen Kernpunkte

auch noch andere Strategien als die genannten. Diese könnten die Güte der beiden Distanzfunktionen weiter verbessern;
- die Modifikation der Distanzfunktionen, um diese robust gegenüber kleineren Verdeckungen (z.B. Stühlen, Tischen oder kleineren dynamischen Hindernissen im Sichtfeld des Roboters) zu machen.

Ein weiteres Ziel ist beispielsweise die Implementierung des Matching-Algorithmus' (siehe Abschnitt 4), der letztendlich aus dem Scan und demjenigen Skelett, das dem Scan am ähnlichsten ist, die Position des Roboters bestimmt. Auch ist es sinnvoll, die Distanzen $d(\mathcal{S}, V^*)$ robust gegenüber „Ausreißern" im Scan zu machen. Bei der Hausdorff-Distanz kann man dies z.B. durch Verwendung der k-Hausdorff-Distanz erreichen, die bei der Supremumsbildung anstatt des größten Abstandswertes nur den k-größten Wert verwendet, mit einem Parameter k aus dem Intervall [1, Anzahl der Scanpunkte].

Literatur

1. H. Alt, B. Behrends, J. Blömer. Approximate Matching of Polygonal Shapes. In *Proceedings of the 7th Annual ACM Symposium on Computational Geometry*, S. 186–193, 1991.
2. E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, J. S. B. Mitchell. An Efficiently Computable Metric for Comparing Polygonal Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 13, S. 209–216, 1991.
3. I. J. Cox. Blanche — An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle. *IEEE Transactions on Robotics and Automation*, Bd. 7(2), S. 193–204, April 1991.
4. G. Dudek, K. Romanik, S. Whitesides. Localizing a Robot with Minimum Travel. In *Proceedings of the 6th Annual ACM–SIAM Symposium on Discrete Algorithms*, S. 437–446, 1995.
5. L. J. Guibas, R. Motwani, P. Raghavan. The Robot Localization Problem. In K. Goldberg, D. Halperin, J.-C. Latombe, R. Wilson (Hrsg.), *Algorithmic Foundations of Robotics*, S. 269–282. A K Peters, 1995. `http://theory.stanford.edu/people/motwani/postscripts/localiz.ps.Z`.
6. O. Karch, H. Noltemeier. Robot Localization—Theory and Practice. In *Proceedings of the 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '97)*, 1997.
7. K. Mehlhorn, S. Näher. LEDA – A Platform for Combinatorial and Geometric Computing. *Communications of the ACM*, Bd. 38, S. 96–102, 1995. `http://www.mpi-sb.mpg.de/guide/staff/uhrig/ledapub/reports/leda.ps.Z`.
8. H. Noltemeier, K. Verbarg, C. Zirkelbach. A Data Structure for Representing and Efficient Querying Large Scenes of Geometric Objects: MB*–Trees. In G. Farin, H. Hagen, H. Noltemeier (Hrsg.), *Geometric Modelling*, Bd. 8 von *Computing Supplement*, S. 211–226. Springer, 1993.
9. S. Schuierer. Efficient Robot Self-Localization in Simple Polygons. In O. Karch, H. Noltemeier, K. Verbarg (Hrsg.), *13th European Workshop on Computational Geometry (CG '97)*, S. 20–22. Universität Würzburg, 1997.
10. Th. Wahl. Distanzfunktionen für Polygone und ihre Anwendung in der Roboterlokalisation. Diplomarbeit, Universität Würzburg, Juni 1997.