In: 14. Fachgespräch Autonome Mobile Systeme (AMS'98), Karlsruhe, 30. Nov. – 01. Dezember, 1998, Springer-Verlag, Reihe "Informatik Aktuell", pp 188–198.

Flächendeckende Bahnplanung in vollständig, teilweise oder nicht bekannten Umgebungen

Dominik HENRICH und René GRAF
Institut für Prozeßrechentechnik, Automation und Robotik (IPR)
Universität Karlsruhe, D-76128 Karlsruhe
E-mail: [dHenrich, Graf] @ira.uka.de,
http: //wwwipr.ira.uka.de/~dhenrich/

Zusammenfassung. *Es wird die Aufgabe der vollständigen räumlichen Abdeckung von Regionen in durch mobile Roboter betrachtet. Dabei können die Regionen in vollständig, teilweise oder nicht bekannten Umgebungen liegen. Zur Lösung wird ein Verfahren aus der Computergrafik zum Füllen von Bildregionen zugrunde gelegt. Das Verfahren hat eine lokale Sichtweise und läßt somit den Einsatz von Sensordaten und das Auftreten von unvorhergesehenen Hindernissen zu. Die Regionen können durch Karten off-line vorgegeben sein oder durch Sensordaten online aufgebaut werden. Dennoch ist eine vollständige und genau einmalige Flächenbearbeitung garantiert. Dies wird an Beispielen in einer graphischen Visualisierung der Realzeit-Steuerung des Roboters validiert.*

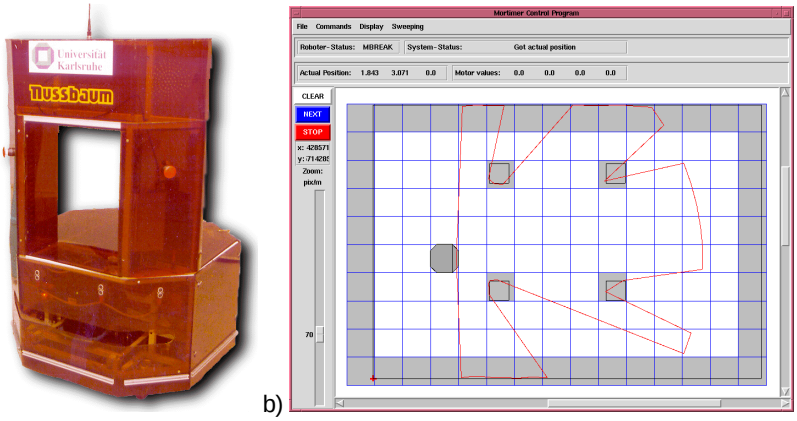
1 Einleitung

Für Serviceroboter bestehen eine Vielzahl von Planungsaufgaben. Bei mobilen Robotern sind dies insbesondere die kollisionsfreie Bahn- und Bewegungsplanung. Bei der *flächendeckenden* Bahnplanung (Sweeping, Complete Coverage) wird nicht nur ein kollisionsfreier Weg von Start zum Ziel gesucht, sondern der Roboter muß eine spezifizierte Region vollständig abdecken bzw. bearbeiten. In den meisten Anwendungsfällen kann die generierte Bahn an einem beliebigen Ziel enden. Optional kann gefordert werden, daß jeder Punkt in der Region genau einmal bearbeitet wird.

Diese Art von Bahnplanung kann für eine Reihe von verschiedenen Anwendungsgebieten eingesetzt werden. Dazu gehören zum Beispiel die Bodenreinigung, die Rasenpflege, die Boden- bzw. Rauminspektion und die Entfernung von Landminen.

Zur Untersuchung dieser Aufgabenstellung wird der mobile Roboter MORTIMER als Versuchsplattform verwendet (Abbildung 1a). Das Antriebskonzept erlaubt eine Translation in beliebige Richtungen bei gleichzeitiger Drehung [Graf98]. In dieser Arbeit wird ein Laserscanner zur Erkennung von Hindernissen verwendet. Der Laserscanner liefert 360 Abstandswerte bis 6 m Entfernung aus dem Bereich [0°, 180°] in Fahrtrichtung von denen aus Rechenzeitgründen nur jeder Zehnte ausgewertet wird.

Die meisten Planungsaufgaben bauen auf einer internen Repräsentation der Umwelt auf. Hierzu können zum Beispiel die polygon-basierten oder die zellen-basierten Repräsentationen verwendet werden. Entsprechend können die Regionen, in denen eine flächendeckende Bahn gefunden werden soll, durch ein umrandendes Polygon oder durch eine Reihe aufeinanderfolgender Zellen beschrieben werden. Im folgenden wird von der zweiten Art der Repräsentation ausgegangen. Sie ermöglicht später die einfache Berücksichtigung von Sensordaten. Zur flächendeckenden Bahnplanung wird die Zellen-



a) b)

Abbildung 1: Der Roboter MORTIMER als Versuchsplattform (a) und die Visualisierung der Robotersteuerung mit der Umgebung in Polygon- und Zellenrepräsentation sowie das Laserscanpolygon (b)

größe entsprechend der durch den Roboter bzw. die Bearbeitungseinheit abgedeckten Fläche gewählt (siehe Abbildung 1b).

Für jede derzeit abgedeckte Zelle kann der Roboter entscheiden, ob er sie mit oder ohne Bearbeitung überfahren möchte. Unbearbeitete Zellen müssen dann zu einen späteren Zeitpunkt erneut angefahren werden. Diese Entscheidungsfreiheit ist notwendig, z.B. wenn abzudeckende Zellen nur genau einmal bearbeitet werden dürfen. Somit ergeben sich bei der flächendeckenden Bahnplanung für jede Zelle die drei Zustände: *frei*, *belegt* und *bearbeitet*.

Der Rest des Artikels ist wie folgt gegliedert: Zunächst werden die bekannten Lösungen des oben genannten Bahnplanungsproblems in Abschnitt 2 dargestellt und bewertet. In Abschnitt 3 wird dann die Anwendung eines Algorithmus aus der Computergrafik vorgeschlagen. Wegen seiner lokalen Sichtweise kann dieser Ansatz in Abschnitt 4 gleichermaßen in vollständig, teilweise oder nicht bekannten Umgebungen eingesetzt werden.

2 Bisherige Ansätze

Im Vergleich mit der Bahnplanung für Punkt-zu-Punkt-Bewegungen gibt es in der Literatur für die flächendeckenden Bahnplanung mit mobilen Robotern relativ wenig Vorarbeiten. Es können die folgenden Ansätze unterschieden werden:

Durch eine regel-basierte Auswahl und Aneinanderketten von Bewegungsprimitiven können flächendeckende Parallelbahnen in polygonalen Umgebungen erzeugt werden [Hofner93, Carvalho97]. Einerseits sind diese Bahnen direkt von dem mobilen Roboter ausführbar. Andererseits werden zunächst keine unvorhergesehenen Hindernisse berücksichtigt, so daß eine Nachbearbeitungsphase notwendig ist [Hofner97b]. Darüber hinaus kann der Roboter in Sackgassen stecken bleiben und somit die Fläche nicht notwendigerweise vollständig bearbeiten [Carvalho97].

Durch Wellenausbreitung beginnend von der Zielzelle kann in einer gerasterten Umweltkarte ein globales Potentialfeld (distance transforms) berechnet werden. Zur Bahnplanung wird nun von einer beliebigen Startposition solange im Potential möglichst *auf*gestiegen und beim notwendigen *Ab*steigen oder Entlanggleiten im Potential die Fläche bearbeitet, bis im Zielpunkt jede Nachbarzelle bearbeitet ist [Zelinsky93, Zelinsky98]. Einerseits ist dies der einzige Ansatz, bei dem ein Zielpunkt angegeben werden kann (und muß). Andererseits ist seine Bestimmung für eine günstige Bahnplanung offen. Außerdem werden keine unvorhergesehenen Hindernisse berücksichtigt.

Durch Lösen des Handlungsreisenden-Problem kann auch eine flächendeckende Bahn geplant werden [Kurabayashi96a]. Dazu wird zuerst rekursiv und heuristisch eine Reihe von getrennten (Parallel- bzw. Kontur-) Teilbahnen gewählt, welche die Fläche vollständig abdecken. Die Start- und Endpunkte bzw. Eckpunkte der Teilbahnen bilden dann die Städte, für welche die kürzeste Gesamtbahn berechnet wird. Zwar ist die Lösung der Gesamtbahn für die gewählten Teilbahnen optimal, aber es ist unklar, mit welcher Heuristik günstige Teilbahnen gefunden werden können. Bei unvorhergesehenen Hindernissen muß eine lokale Umplanung erfolgen [Kurabayashi98a], welche einen exponentielle Aufwand in der Anzahl von betroffenen Kanten bzw. Eckpunkte hat.

Durch eine Vielzahl von Linearbahnen mit zufälligen Richtungswechseln bei einem Hinderniskontakt kann nach ausreichend langer Zeit auch eine Flächenabdeckung erreicht werden. In [Elektrolux97] wird dabei zuerst der Flächenrand einmal abgefahren. Einerseits werden hierbei das Planungsproblem umgangen und unvorhergesehene Hindernisse auf natürliche Weise berücksichtigt. Andererseits kann weder eine vollständige noch eine genau einmalige Flächenabdeckung innerhalb einer endlichen Bearbeitungszeit garantiert werden.

Insgesamt handelt es sich bei den bisherigen Ansätzen entweder um reine off-line Planungen mit notwendiger Nachbearbeitung bei unvorhergesehenen Hindernissen oder um reine on-line Ansätze ohne garantierte Flächenabdeckung. Im folgenden wird nun eine Ansatz vorgestellt, der sowohl die Fläche garantiert abdeckt als auch detektierte Hindernisse berücksichtigt. Im Extremfall können auch gänzlich unbekannte Umgebungen bearbeitet werden. Darüber hinaus kann die Fläche genau einmal bearbeitet werden.

3 Grundansatz

Hier wird zur Lösung der flächendeckenden Bahnplanung ein Algorithmus aus der Computergrafik angewendet. Es handelt sich dabei um einen speichereffizienten Ansatz zum Füllen von Regionen aus [Henrich94a]. Gegenüber anderen Füllalgorithmen hat dieser die Eigenschaft, daß der sogenannte Cursor zum Füllen innerhalb der zu füllenden Region kontinuierlich herumwandert und dabei sukzessive Bildpunkte einfärbt. Andere Füllalgorithmen springen mit dem Curser u.U. oft in der zu füllenden Region und eignen sich somit nicht für die Anwendung bei mobilen Robotern. Im folgenden wird der Algorithmus im Rahmen der Anwendung auf die flächendeckenden Bahnplanung dargestellt.

Zur Steuerung des Bahnplanungsalgorithmus wird ein lokales Sichtfenster W des Roboters auf seine Karte bzw. Umwelt eingeführt (Abbildung 2). Das Fenster spei-

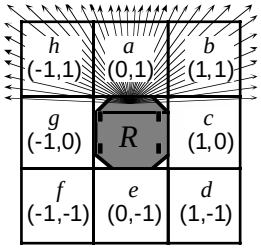


Abbildung 2: Lokales Sichtfenster des Roboters *R* mit Laserscanner auf das gerasterte Umweltmodell bzw. auf die reale Umgebung mit Zuständen (*a*, …, *h*) und deren relative Position zum Roboter

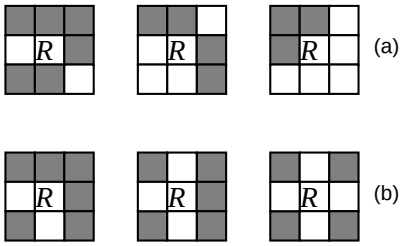


Abbildung 3: Beispiele für die Bewertung der Roboterposition *R* basierend auf den Zuständen der Nachbarzellen: *R* "nicht kritisch" (a) und *R* "kritisch" (b)

chert die Zustände der acht Nachbarzellen um den Roboter als 3x3-Ausschnitt der Umweltkarte. Dieses Fenster kann gleichermaßen für die Detektierung von bekannten bzw. unvorhergesehenen Hindernissen genutzt werden. Hierzu werden die Zustände des Fensters durch die Umweltkarte bzw. durch die entsprechenden Sensordaten bestimmt.

Mittels der Zustände in dem Fenster kann erkannt werden, ob die aktuelle Position des Roboters möglicherweise zwei (4-verbundene) Teilregionen miteinander verbindet. Mit einem lokalen Sichtfenster auf die zu bearbeitende Region kann dies zunächst nicht endgültig festgestellt werden. Daher beschränken wir uns hier auf die Detektion von *kritischen* Zellen, welche möglicherweise zwei Teilregionen verbinden (Abbildung 3). Diese Zellen werden durch die Funktion *CellCritical(W)* aus [Henrich94a] bestimmt.

Die Bewegung des Roboters erfolgt nach der Strategie "möglichst rechts" und ergibt eine spiralförmige Bahn innerhalb der abzudeckenden Region. In der Funktion *TurnRight(W, D, Stop)* wird die nächste Bewegungsrichtung *D* aufgrund des Sichtfensters *W* mit den Zuständen der Nachbarzellen ausgewählt. Die Priorität der Nachbarzellen ist dabei relativ zur aktuellen Bewegungsrichtung rechts, gerade, links und zurück. Damit bewegt sich der Roboter auf einer Bahn entlang den Hindernissen bzw. entlang den schon bearbeiteten Zellen. Falls keine direkte Nachbarzelle mehr frei ist, dann wird *Stop* auf *true* gesetzt.

Für die Funktionen *CellCritical(…)* und *TurnRight(…)* gelten im lokalen Sichtfenster neben den belegten auch die schon bearbeiteten Zellen als "Hindernis". Nur so kann garantiert werden, daß die Region vollständig abgedeckt und jede Zelle genau einmal bearbeitet wird.

Zum flächendeckenden Bearbeiten der Region wird das Sichtfenster der aktuellen Zelle betrachtet. Falls eine Zelle kritisch ist und somit die zu bearbeitende Region möglicherweise in zwei Teilregionen aufteilt, dann wird eine Teilregion sofort bearbeitet und die kritische Zelle zur späteren Weiterbearbeitung der anderen Teilregion(en) in einem Stack[1] abgespeichert. Nach Bearbeitung der aktuellen Zelle *C* durch *ProcessCell(C)* bewegt sich der Roboter gemäß der obigen Strategie in *MoveRobot(C, W, D)* um eine Zelle in Richtung *D* weiter und aktualisiert das Fenster *W*. Das Bewegen und Bearbei-

---

1 Auf die Datenstruktur Stack wird mit den bekannten Standardfunktionen *Empty*, *Push*, *Top* und *Pop* zugegriffen.

```
procedure MemoryFill(C    : Cell) ;
(*INPUT:     Start cell C within a 4-connected region*)
(*OUTPUT:    All cells in region are processed *)
var   D      : Direction;   (*current moving direction*)
      W      : Window;      (*3x3 region around the robot*)
      S      : Stack;       (*stack for critical cells*)
      Stop   : Boolean;     (*true, if no direction is free*)
begin
      InitMemoryFill(C, D, W) ;                    (*move to a corner*)
      Push(S, C) ;
      while not Empty(S) begin                     (*for all subregions*)
            Stop = false;
            C: = Top(S) ;                          (*get first cell in subregion*)
            while not Stop do begin                (*process current subregion*)
                  if not CellCritical(W) begin
                        ProcessCell(C) ;           (*process current cell*)
                        W [0, 0] : = Processed;(*update window*)
                  end else begin
                        Push(S, C) ;               (*store subregion*)
                        W [0, 0] : = Processed;(*assume cell to be processed*)
                  end;
                  TurnRight(W, D, Stop) ;          (*choose new direction*)
                  if not Stop then
                        MoveRobot(C, W, D) ;(*move robot and update window*)
            end; (*Stop*)
            do                                     (*select next subregion*)
                  Pop(S) ;                         (*delete processed subregion*)
            while (not Empty(S)) and CellProcessed(Top(S)) ;
            if not Empty(S) then
                  (*find path and move to Top(S) (next subregion) *)
      end; (*for all subregions*)
end; (*MemoryFill*)
```

Abbildung 4: Algorithmus *MemoryFill* zum flächendeckenden Bahnplanen in Pseudo-PASCAL-Notation

ten geschieht so lange, bis *Stop = true* gilt und damit die ganze (Teil-) Region bearbeitet ist.

Sind noch weitere Teilregionen aus dem Stack zu bearbeiten, dann fährt der Roboter zu der entsprechenden, abgespeicherten Zelle und setzt die Bearbeitung dort fort. Eine kollisionsfreie Bahn zwischen der aktuellen Zelle und der Zelle der nächsten Teilregion kann mittels bekannter Bahnplanungsverfahren für kollisionsfreie Punkt-zu-Punkt-Bewegungen generiert werden. Hier wird zunächst eine globale, diskrete Potentialfeldkarte durch Ausbreitung einer Wellenfront von der Zielzelle berechnet (distance transform). In dieser Potientialfeldkarte kann dann eine kollisionsfreie Bahn von der aktuellen Zelle zur nächsten Teilregion durch einen Gradientenabstieg erzeugt werden.

Beim Algorithmus *MemoryFill(C)* in Abbildung 4 wird davon ausgegangen, daß die Startzelle *C* des Roboters in einer einfachen, endlichen und 4-verbundenen Region befindet. In der Initialisierung durch *InitMemoryFill*(*C*, *D*, *W)* kann optional die Startzelle *C* bzw. der Roboter und das Fenster *W* zu einer Eckzelle verschoben werden, ohne dabei Zellen zu bearbeiten. Dadurch reduziert sich in der Bearbeitungsbahn die Anzahl von Richtungsänderungen. Die Laufrichtung *D* muß so gesetzt werden, als wenn der Roboter gerade an dem Rand entlang zu dieser Eckzelle gekommen wäre.

Die partielle Korrektheit bzw. die Terminierung und damit die Korrektheit selbst von *MemoryFill* werden in [Henrich94a] gezeigt. Obwohl *MemoryFill* nicht die garantiert

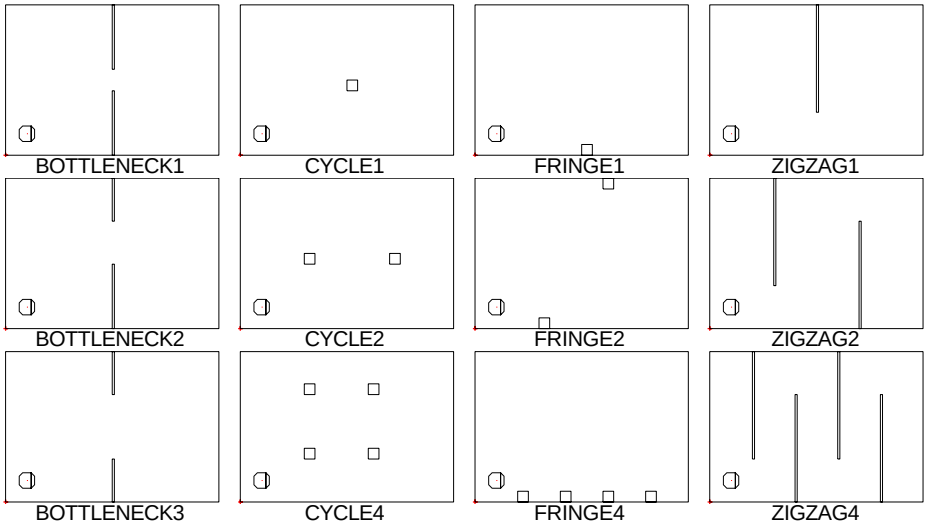


Abbildung 5: Benchmark-Probleme für die flächendeckende Bahnplanung mit Startposition in der linken unteren Ecke

kürzeste flächendeckende Bahn generieren kann, so werden doch nur wenige Zellen mehrfach angefahren bzw. jede Zelle nur genau einmal bearbeitet (siehe Abschnitt 4). Die Berechnung der kürzesten Bahn entspricht dem Problem des Handlungsreisenden. Hierbei korrespondieren die Zellen den Städten, welche mit bis zu vier benachbarten Zellen bzw. Städten verbunden sind.

4 Einsatzfelder

Der oben beschriebene Algorithmus kann in verschiedenen Situationen für die flächendeckende Bahnplanung mit einem mobilen Roboter eingesetzt werden. Sie unterscheiden sich vor allem durch den Grad des à priori bekannten Wissens über die Arbeitsumgebung. Hier wird nun auf Umgebungen (4.1) mit bekannten Hindernissen, (4.2) mit bekannten und unvorhergesehenen Hindernissen sowie (4.3) mit unbekannten Hindernissen eingegangen.

Dazu wurden verschiedene Umgebungsmodelle mit unterschiedlichen Problemtypen und Schwierigkeitsgraden entworfen (Abbildung 5) . Sie haben eine Standardgröße von 10 m x 7 m und sind für einen Roboter der Größe 0, 75 m x 0, 75 m ausgelegt. Die Startposition der flächendeckenden Bearbeitung ist jeweils in der linken unteren Ecke.[2]

4.1 Bekannte Umgebungen

In diesem Abschnitt wird davon ausgegangen, daß die zu bearbeitende 2-dimensionale Arbeitsumgebung des Roboters vollständig bekannt ist und keine unvorhergesehenen Hindernisse auftreten. Außerdem sind alle Hindernisse stationär. Unter diesen Voraus-

---

2 Die Daten dieser Umgebungen können heruntergeladen werden von der Internet-Seite: `http://wwwipr.ira.uka.de/~paro/sweeping/`

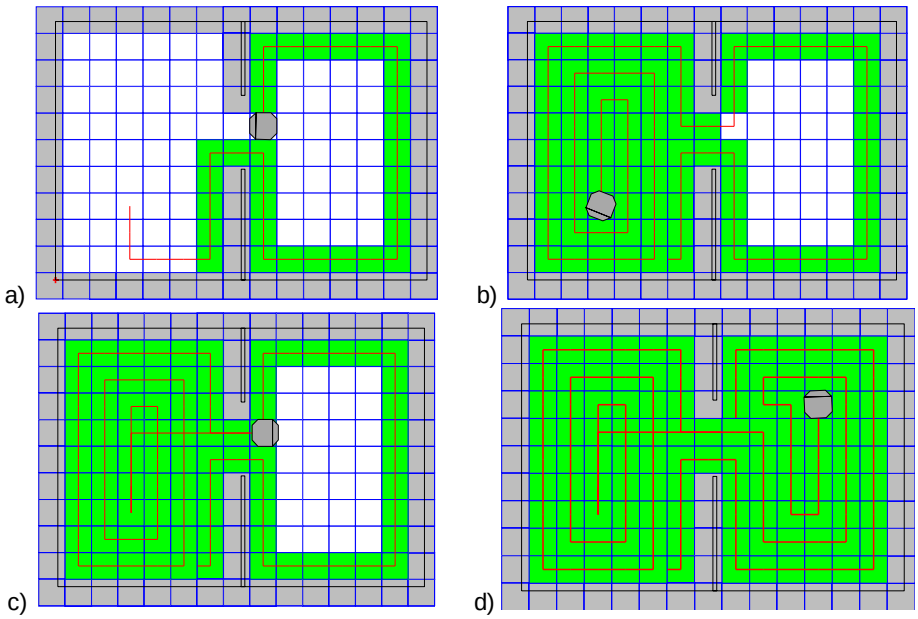


Abbildung 6: Vier Simulationsphasen für den Benchmark BOTTLENECK2 als bekannte Umgebung (Flächenbearbeitung: 89%, Mehrfachabdeckung: 11%)

setzungen kann die flächendeckende Bahn vollständig vor ihrer Ausführung geplant werden.

Ist ein polygonales Umweltmodell gegeben, so muß man zunächst für *MemoryFill* die entsprechende Zellenrepräsentation berechnen. Dazu werden hier alle Polygonkanten mit allen horizontalen und vertikalen durch die Zellenränder sich ergebenden Gitterlinien geschnitten. Die den Schnittpunkten angrenzenden Zellen werden auf den Zustand *belegt* gesetzt. (Da dieser einfache Algorithmus die meisten Zellen zweimal als belegt markiert, ist er sicherlich nicht der schnellste, aber für unsere Anwendung vollkommen ausreichend.)

Die flächendeckenden Bahn für den Benchmark BOTTLENECK2 als bekannte Umgebung ist in Abbildung 6 zu sehen. Zur Initialisierung fährt der Roboter an eine Ecke und beginnt dort die Bearbeitung. In Abbildung 6a steht der Roboter auf einer kritischen Zelle, welche die linke und rechte Teilregion separiert. Die Bahn wird zur linken Teilregion fortgesetzt und führt zunächst nicht zur kritischen Zelle zurück, da sie von dann schon bearbeiteten Zellen blockiert wird (Abbildung 6b). Daher wird die kritische Zelle auf dem Stack abgespeichert. Sie wird später als Repräsentant der noch nicht bearbeiteten rechten Teilregion durch eine potentialfeld-basierte Punkt-zu-Punkt-Bahnplanung erneut angefahren (Abbildung 6c). So kann dann auch die rechte Teilregion vollständig bearbeitet werden (Abbildung 6d).

4.2 Teilweise bekannte Umgebungen

In diesem Abschnitt wird davon ausgegangen, daß ein Teil der Umgebung bekannt ist, aber dennoch unvorhergesehene Hindernisse auftreten können. Die unvorhergesehenen

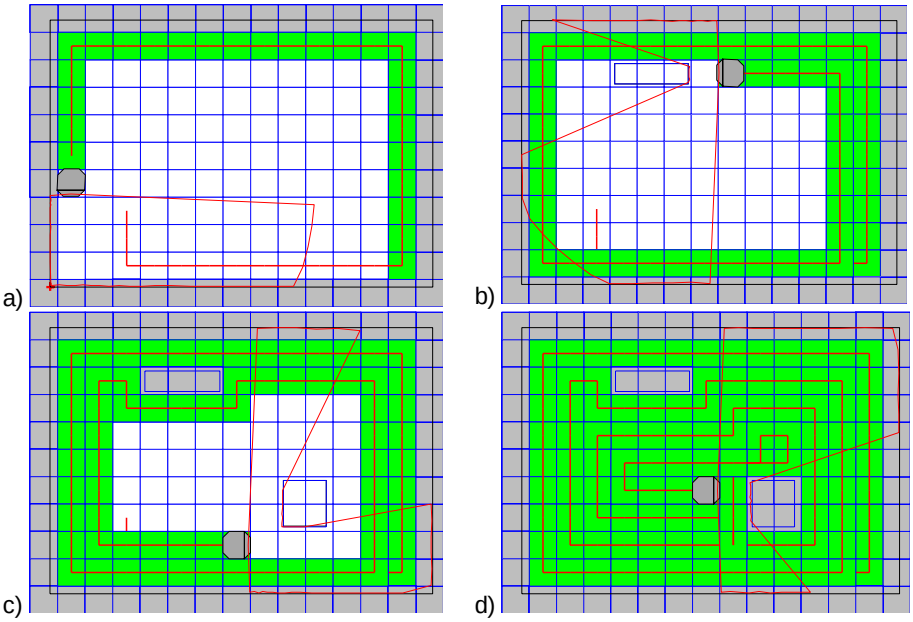


Abbildung 7: Vier Simulationsphasen (mit Laserscanpolygon) für den Benchmark SIMPLE als bekannte Umgebung mit zwei (unvermittelt auftretenden) unvorhergesehenen Hindernissen (Flächenbearbeitung: 94%, Mehrfachabdeckung: 13%)

Hindernisse sind nicht-dynamisch und müssen durch einen Sensor, wie hier der Laserscanner zuverlässig detektierbar sein. Die Auswertung der Laserscanner-Daten beschränkt sich hier auf die Detektion von Hindernissen innerhalb des lokalen Sichtfensters. Dies sind insbesondere die drei Zellen vor dem Roboter in Fahrtrichtung (Zellen *h*, *a* und *b* in Abbildung 2). Dazu wird für jeden Abstandswert mit dem korrespondierenden Scanwinkel berechnet, in welcher Zelle sich der gemessene Punkt des (polygonalen) Hindernis befindet und die Zellenzustände im Sichtfenster aktualisiert. Bei Bedarf könnten auch Zellen außerhalb des Sichtfensters aktualisiert und somit eine weitere Vorausschau realisiert werden. Die Sensordatenauswertung findet statt, nachdem sich der Roboter in eine neue Zelle bewegt hat, so daß das lokale Sichtfenster immer auf dem neuesten Stand ist.

Die flächendeckenden Bahn für den Benchmark SIMPLE als teilweise bekannte Umgebung ist in Abbildung 7 zu sehen. Zunächst stellt sich die Umgebung wie in der offline verfügbaren Umweltkarte dar und die Bahn wird entsprechend berechnet (Abbildung 7a). Dann werden unvorhergesehene Hindernisse detektiert (Abbildung 7b+c). Diese können durch die on-line Abarbeitung der Bahnplanung entsprechend umfahren werden (Abbildung 7c+d).

Sollten während der Bearbeitung neue oder nicht-stationäre Hindernissen auftreten, so sind die Voraussetzungen zunächst verletzt. In ungünstigen Fällen könnten dadurch unbearbeitete Teile der Fläche vom Roboter abgetrennt werden und bleiben unbearbeitet. Diese Art von dynamischen Umgebungen kann durch Anschluß einer der üblichen Nachbearbeitungsphasen vollständig bearbeitet werden.

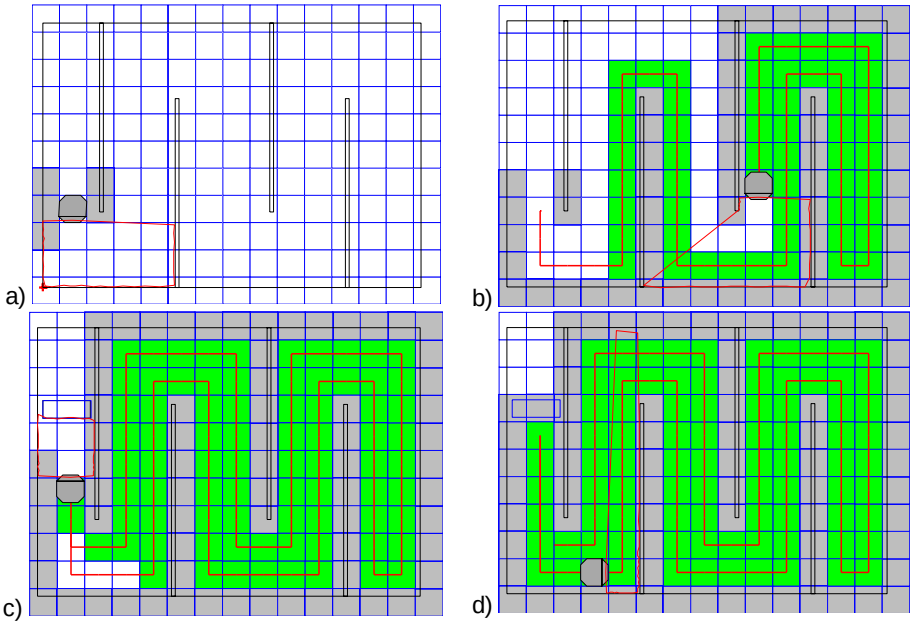


Abbildung 8: Vier Simulationsphasen (mit Laserscanpolygon) für den Benchmark ZIGZAG4 als unbekannte Umgebung (Flächenbearbeitung: 71%, Mehrfachabdeckung: 8%)

4.3 Unbekannte Umgebungen

In diesem Abschnitt wird davon ausgegangen, daß die Umgebung vollständig unbekannt ist. Somit müssen zunächst die Hindernisse um die Startposition des Roboters detektiert werden. Dazu dreht sich der Roboter einmal auf der Stelle um 360° und erfaßt damit alle möglichen Hindernisse innerhalb des lokalen Sichtfensters. Ein Scan in Fahrtrichtung reicht nicht, da die Startzelle schon auf Grund der Zustände der Nachbarzellen eine kritische Zelle sein könnte. Außerdem kann in unbekannten Umgebungen auch keine günstige Lage der Zellenrepräsentation relativ zu den polygonalen Hindernissen berechnet werden. Daher werden hier die Zellen an der Startposition des Roboters ausgerichtet.

Die flächendeckenden Bahn für den Benchmark ZIGZAG4 als vollständig unbekannte Umgebung ist in Abbildung 8 zu sehen. Zunächst wird durch eine 360°-Drehung die direkte Umgebung erfaßt (Abbildung 8a). Nach dem Anfahren einer Eckzelle wird die Bearbeitung vorgenommen (Abbildung 8b). Die hier auftretenden nicht-stationären Hindernisse können ohne Modell nicht von den stationären unterschieden werden. Daher können abgetrennte Teilregionen unbearbeitet bleiben (Abbildung 8c+d).

5 Schlußfolgerungen und Ausblick

In diesem Artikel wurde die flächendeckende Bahnplanung für Serviceroboter betrachtet. Gegenüber bisherigen Ansätzen kann der hier vorgeschlagene Lösungsansatz gleichermaßen für vollständig, teilweise und nicht bekannte Umgebungen eingesetzt wer-

den. Hierzu müssen nur die Eingabedaten und die Zustandsinterpretation des Algorithmus angepaßt werden; der Algorithmus selbst ändert sich nicht. Trotz rein lokaler (sensorbasierter) Sichtweise wird eine globale und einmalige Flächenbearbeitung erreicht.

Der vorgestellte Ansatz kann direkt auf andere Sensoren zur Hindernisdetektion angewendet werden, vorausgesetzt, die Sensordaten können in eine Zustandsbelegung im Sichtfenster umgerechnet werden. Zunächst ist geplant neben der dargestellten simulierten Ausführung Experimente mit dem Robotersystem MORTIMER durchzuführen. Interessant ist sicherlich auch eine Erweiterung des Ansatzes zur Bearbeitung von nur teilweise durch Hindernisse belegten Zellen. Fraglich dabei ist, inwieweit man dann von der Zellenrepräsentation auf eine rein polygonale Umweltmodellierung übergehen kann und muß.

Literaturverzeichnis

[Carvalho97] Neumann de Carvalho R., et al: "Complete coverage path planning and guidance for cleaning robots". In: IEEE Int. Symp. on Industrial Electronics (ISIE'97), Guimaraes, Portugal, 1997, pp 677-682.

[Electrolux97] Pressemitteilung der Firma Electrolux zur Vorstellung eines autonomen Bodenreinigungsroboters, http: //www.electrolux.se/corporate/pressnotes/274a_4b2.htm, (siehe auch http: //www.electrolux.se/robot/), Dez. 1997.

[Graf98] Graf R., Weckesser P.: "Roomservice in a hotel". In: IFAC Symp. on Intellegnt Autonomous (IAV'98), Madrid., Eds: M.A. Salichs and A. Halme, March 1998, vol 2, pp 641-647.

[Henrich94a] Henrich D.: "Space-efficient region filling in raster graphics". In: The Visual Computer: International Journal of Computer Graphics, vol 10, no 4, pp 205-215, 1994.

[Hofner93] Hofner C., Schmidt G.: "Automatische Kursplanung und Führung autonomer Reinigungsfahrzeuge". In: 9. Fachgespräch Autonome mobile Systeme (AMS'93), München, 1993, pp 27-38.

[Hofner97b] Hofner Ch., Schmidt G.: "Nachbearbeitungsaspekte im Kontext der Automatisierung flächendeckender Bearbeitungsaufgaben". In: Fachgespräche Autonome mobile Systeme (AMS'97), Stuttgart, 1997, pp 232-245.

[Kurabayashi96a] Kurabayashi D., et al.: "Cooperative sweeping by multiple mobile robots". In: Proc. of the Int. Conf. on Robotics and Automation (ICRA´96), Minneapolis, 1996, vol 2, pp 1744-1749.

[Kurabayashi98a] Kurabayashi D., et al.: "Local path re-planning for unforeseen obstacle avoidance by an automous sweeping robot". In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA´98), Leuven, Belgium, 1998, pp 3153-3159.

[Zelinsky93] Zelinsky A., Jarvis R., Byrne J., and Yuta S.: "Planning paths of complete coverage of an unstructured environment by a mobile robot". In: Proc. Int. Conf. on Advanced Robotics (ICAR'93), November 1993, Tokyo, Japan.

[Zelinsky98] Zelinsky A.: "Paths of complete coverage of an environment". Personal communication, Sept., 1998.