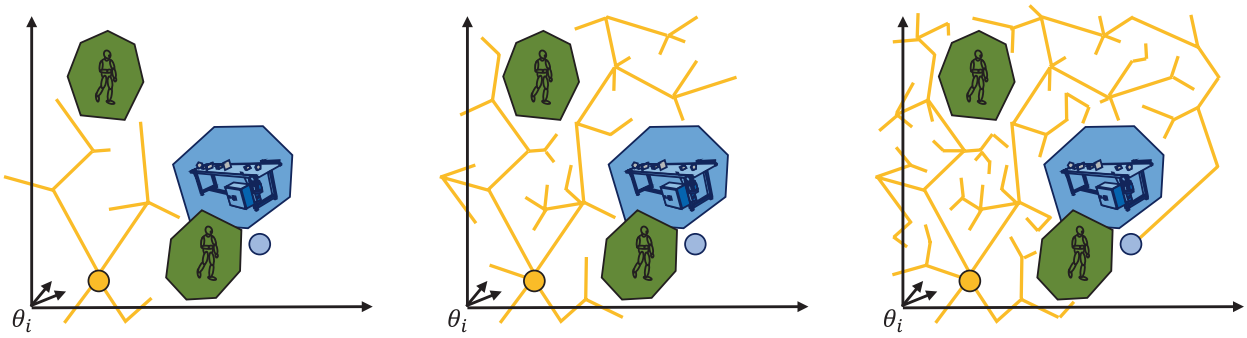

Bild 5.68 Die Planung von aktueller Roboterstellung (oranger Punkt) zu Zielstellung (blauer Punkt) im Gelenkwinkelraum mittels RRT-Verfahren erzeugt zufällig wachsende Baumstrukturen (links, Mitte, rechts) aus kollisionsfreien Bahnvorschlägen (orange Linien), die schließlich auch eine kollisionsfreie Bahn zum Ziel beinhalten (rechts). Im Voraus bekannte Objekte (blaue Flächen) und im Voraus unbekannte Hindernisse (grüne Flächen) behandelt das Verfahren gleich.

chische Darstellung der Hindernisse im Arbeitsraum des Roboters.

Intern erstellt das RRT-Verfahren zur Suche nach einer kollisionsfreien Bahn einen Baum aus kollisionsfreien Bahnvorschlägen im Gelenkwinkelraum des Roboters: Ausgehend von der Ist-Stellung des Roboters führen Äste dieses Baumes als kollisionsfreie Teilbahnen in verschiedene Richtungen davon. Äste können sich ferner verzweigen, in diesem Fall teilt sich ein Bahnvorschlag in mehrere Fortsetzungsmöglichkeiten für die Roboterbewegung auf. Schließlich endet nach erfolgreicher Bahnplanung zumindest einer der Baumäste an der Zielstellung des Roboters. Eine über den Baum verlaufende Gesamtbahn von aktueller Stellung zu Ziel ist dann insgesamt kollisionsfrei und kann vom Roboter abgefahren werden.

Zum Erstellen des Baums aus Bahnvorschlägen fließt in das RRT-Verfahren der Zufall ein: Zuerst wird die Ist-Position des Roboters als Wurzel des Baums festgelegt. Dann werden neue Roboterstellungen zufällig gezogen. Falls eine neue Roboterstellung zu keiner Kollision mit dem Weltmodell führt, so versucht das RRT-Verfahren die kollisionsfreie Stellung über eine (lineare oder nicht-lineare) Teilbahn mit dem nahsten Knotenpunkt des bestehenden Baums kollisionsfrei zu verbinden. Die Kollisionsfreiheit wird über eine (unregelmäßige oder regelmäßige) Abtastung der Teilbahn geprüft. Nur wenn die neue Stellung und die Teilbahn zum bestehenden Baum kollisionsfrei sind, nimmt das RRT-Verfahren die zufällig gewählte Stellung und die Teilbahn in den Baum auf. Um das Ziel zu erreichen, versucht das RRT-Verfahren im Anschluß, den Endpunkt einer erfolgreich hinzugefügten Teilbahn über eine weitere Teilbahn kollisionsfrei mit dem Zielpunkt der Roboterbewegung zu verbinden. Ist dies möglich, so hat das RRT-Verfahren eine kollisionsfreie Bewegung von Ist-Position über eine Reihe von Teilbahnen des Baums bis zum Zielpunkt gefunden. Die Bahnplanung kann in diesem Fall erfolgreich abgeschlossen werden. Ein Beispiel für die Planung mittels RRT-Verfahren findet sich in Bild 5.68, ferner zeigt Listing 5.13 eine knappe Implementierung des RRT-Verfahrens in Pseudo-Code.

Für den Einsatz in der Praxis ist die oben stehende, naive Arbeitsfolge des RRT-Verfahrens noch in vielen Punkten an die Anforderungen der Echtzeitplanung anzupassen. So ist beispielsweise ein Abbruch nach einer bestimmten Anzahl Fehlversuche oder nach einer festen Zeitschranke dringend nötig, falls eine kollisionsfreie Bahn nur schwer zu finden ist oder gar nicht existiert.

Weitere Variationsmöglichkeiten bieten sich bei der Auswahl der zufälligen Roboterstellung und der Vernetzung mit bereits zuvor akzeptierten Roboterstellungen. So ist es zum Beispiel üblich, noch nicht abgetasteten Raum beim zufälligen Ziehen zu bevorzugen. Beim Verbinden berücksichtigt man üblicherweise nur Roboterstellungen, die in der Nähe der zufällig gezogenen Stellung liegen. Zuletzt ist man nicht notwendigerweise auf lineare Gelenkbewegungen zwischen zwei Roboterstellungen angewiesen, sondern kann alternative Bahnsegmente finden. Dazu eignen sich unter anderem die Verfahren zur Erzeugung von lokalen Ausweichbewegungen.

Wie bei den lokalen Bahnplanungsverfahren ist auch die Bewertung des RRT-Verfahrens bezüglich der *Rechenzeit*, des *Speicherbedarfs* und des *Implementierungsaufwands* stark vom genutzten *Weltmodell* abhängig. Daher berücksichtigt die folgende Bewertung zwei verschiedene Varianten des RRT-Verfahrens: Die erste

Listing 5.13 Planung kollisionsfreier Bahnen mit RRT

```
Eingabe: Arbeitsraumbelegung mit Kollisionstest
         Ist- und Zielstellung des Roboters
Ausgabe: Kollisionsfreie Bahn zwischen Ist- und Zielstellung
SET(roboterstellung) stellungen = EMPTY
SET(bahnsegment) segmente = EMPTY
stellungen.insert(iststellung)
BOOLEAN bahn_gefunden = FALSE
WHILE NOT bahn_gefunden DO
|  roboterstellung s = zufall()
|  FOR EACH roboterstellung t IN stellungen DO
|  |  IF kollisionsfrei(s) AND kollisionsfrei(t, s) THEN
|  |  |  stellungen.insert(s)
|  |  |  segmente.insert(t, s)
|  |  |  IF kollisionsfrei(s, ziel) THEN
|  |  |  |  stellungen.insert(ziel)
|  |  |  |  segmente.insert(s, ziel)
|  |  |  |  bahn_gefunden = TRUE
|  |  |  BREAK
```

Variante nutzt eine wenig komplexe Voxelrepräsentation der Umwelt und einen Abstandstest mit Brushfire zum Test auf Kollisionen, die zweite Variante setzt stattdessen auf komplexere hierarchische Roboter- und Umweltmodelle mit einem hierarchischen Abstandstest. Analog zu früheren Betrachtungen ergibt sich für die erste Variante ein sehr hoher Rechenzeit- und Speicherbedarf bei durchschnittlichem Implementierungsaufwand. Dem steht bei der zweiten Variante durchschnittlicher Rechenzeit- wie Speicherbedarf, aber ein hoher Implementierungsaufwand gegenüber.

Unabhängig der Wahl des Weltmodells sind Implementierungsaufwand, Rechenzeit und Speicherbedarf für die RRT-Verfahren allgemein höher als für lokale Ansätze. Grund ist die im RRT-Verfahren zwingend nötige, getrennt zu verwaltende Baumstruktur mit Pfadvorschlägen. Allerdings steht diesen Nachteilen eine deutlich höhere und insgesamt große *Flexibilität* durch die globale Bahnplanung gegenüber. Insbesondere findet ein RRT-Verfahren eine kollisionsfreie Bahn immer, sofern eine solche Bahn existiert und genügend Rechenzeit zur Verfügung steht. RRT-Verfahren sind somit *vollständig*. Falls bereits eine kollisionsfreie Bahn gefunden wurde, kann das RRT-Verfahren zusätzlich verfügbare Rechenzeit zudem für die Suche nach weiteren, besseren (z. B. kürzeren) kollisionsfreien Bahnen aufwenden. Somit weist das RRT-Verfahren *Anytime-Fähigkeiten* auf. Ist dagegen keine kollisionsfreie Bahn zwischen Start und Ziel zu finden, muss der Roboter im Zweifelsfall aus Gründen der *Sicherheit* stehen bleiben. Das ist insbesondere deswegen ein Problem, da das RRT-Verfahren sich nur schwer für eine *inkrementelle* Planung eignet: Einerseits liegt nach einer Bewegung des Roboters die aktuelle Roboterstellung nicht mehr an der Wurzel der Baumstruktur, andererseits kann die Bewegung von Hindernissen vollständige Äste des Baums ungültig machen. Zwar existieren komplexere Varianten des RRT-Verfahrens, die die bestehende Baumstruktur systematisch an Änderungen in der Roboterstellung und der Umgebung anpassen, allerdings ist beim naiven Ansatz ein regelmäßiges Verwerfen und Neuplanen der Baumstruktur üblich. Eine ähnliche Argumentation ergibt sich für *Glattheit* und *Vorhersagbarkeit*: Durch die zufällige Wahl von Roboterstellungen und durch das Neuplanen nach Änderungen unterscheiden sich die vom Roboter genutzten Bahnen selbst zwischen gleichen Start- und Zielstellungen deutlich. Dienen lineare Teilbahnen zum Verbinden von Gelenkstellungen, ergibt sich insgesamt eine zitternde und für den Menschen schwer vorherzusehende Roboterbewegung. Die Verwendung von nichtlinearen Teilbahnen oder eine nachträgliche Glättung der Gesamtbahn reduzieren dieses Problem, beheben es aber nicht vollständig. Zuletzt bieten die vielen Varianten der RRT-Planung (z. B. beim Ziehen von Stellungen) zumindest eine gute Möglichkeit, *externe Randbedingungen* in der Bahnfindung zu berücksichtigen.

Die ursprüngliche Publikation zum RRT-Verfahren ist (Lavalle 1998). Varianten finden sich unter anderem in (Kuffner 2000) und (Karaman, Frazzoli 2011).

Probabilistic Roadmaps

Die zuvor präsentierten RRT-Verfahren beginnen üblicherweise in jedem Abarbeitungsschritt des Bahnpla-

nungssystems eine vollständig neue Bahnplanung. Ergebnisse aus vorangehenden Abarbeitungsschritten finden dabei ebenso wenig Beachtung, wie Zusatzinformationen über fest montierte Bestandteile der Roboterzelle. Nachteilige Folgen sind unter anderem schlechte Vorhersagbarkeit und hoher Rechenaufwand. Das alternative Verfahren der Probabilistic Roadmaps umgeht diese Nachteile, indem das Verfahren über den Verlauf mehrerer Abarbeitungsschritte eine Straßenkarte sinnvoller, kollisionsfreier Roboterbahnen pflegt.

Allgemein ist das Ziel beim PRM-Verfahren das gleiche wie bei der RRT-Variante: Von einer gegebenen Ist-Stellung des Roboterarms zu einer ebenso gegebene Zielstellung ist eine kollisionsfreie Roboterbahn zu finden. Voraussetzung ist erneut eine geeignete, vollständige Repräsentation der Hindernisse im Arbeitsraum des Roboters, beispielsweise als Ergebnis einer vorangehenden Hindernisdetektion und Hindernislokalisation. Um das grundlegende Vorgehen des PRM-Verfahrens zu erklären, bleiben Hindernisse zunächst jedoch unberücksichtigt. Vielmehr geht die folgende Erklärung vorerst von einem vollständig objektfreien Arbeitsraum aus.

Das PRM-Verfahren beginnt mit der Erstellung einer initialen Karte aus kollisionsfreien Teilbahnen. Zum Erstellen der initialen Karte wählt das Verfahren iterativ zufällige Stellungen im Gelenkwinkelraum des Roboterarms aus. Diese Stellungen sind als mögliche Endpunkte von Teilbahnen in die Karte einzufügen. Nachdem eine neue Stellung zur Karte hinzugefügt wurde, verbindet das PRM-Verfahren die neue Stellung über neue Teilbahnen (z. B. lineare Bahnen) mit bereits in der Karte eingetragenen, benachbarten Stellungen. Dieses Vorgehen setzt sich fort, bis die Karte eine Mindestanzahl an Roboterstellungen enthält. Dadurch, dass im Verfahrensverlauf gegebenenfalls eine Verbindung zu mehreren Nachbarstellungen hergestellt wird, ensteht im Gegensatz zur Baumstruktur des RRT-Verfahrens beim PRM-Verfahren die besagte Kartenstruktur.

Zuständig für das Erstellen der initialen Karte ist zumeist ein separater offline-Schritt (z. B. bei Start des Programms zur Ansteuerung des Roboters). Dies erspart das Neuberechnen der Gesamtkarte bei jedem weiteren Verarbeitungsschritt.

Ist die initiale Karte erst erstellt, ist mit der Karte prinzipiell eine Planung von einer beliebigen Ist-Stellung des Roboters zu einer ebenso beliebigen Zielstellung möglich. Da weder Ist- noch Zielstellung notwendigerweise auf der Karte liegen müssen, benötigt das Planungsverfahren dazu drei getrennte Schritte: In einem ersten Schritt ist ein Weg von der Ist-Stellung des Roboters auf eine naheliegende Teilbahn der Straßenkarte zu finden. Im zweiten Schritt ist über die Straßenkarte mit graphentheoretischen Ansätzen (z. B. A*-Verfahren) ein Weg zu einer hinreichend nahe an der Zielstellung liegenden Roboterstellung zu finden. Im dritten und letzten Schritt ist schließlich diese Roboterstellung mit der gewünschten Zielstellung zu verbinden, dabei verlässt der Roboter die auf der Straßenkarte liegenden Teilbahnen wieder. Der erste und der letzte Schritt können gegebenenfalls entfallen, wenn sich die Ist-Position des Roboters oder die Zielstellung bereits auf der Straßenkarte befinden. Dieses Vorgehen ist in Bild 5.69 dargestellt.

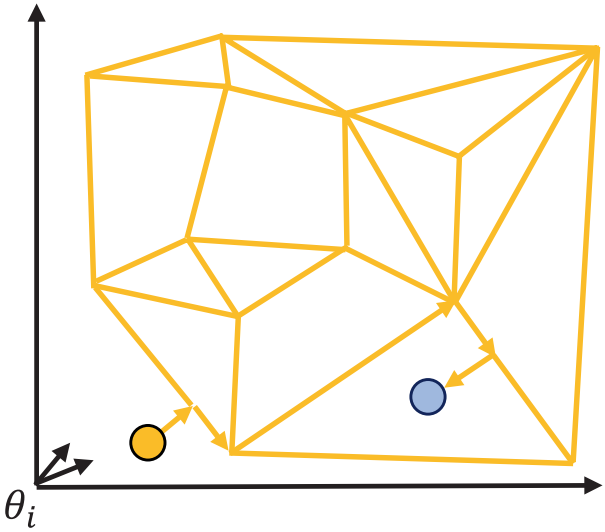

Bild 5.69 Die Planung von aktueller Roboterstellung (oranger Punkt) zu Zielstellung (blauer Punkt) im Gelenkwinkelraum mittels PRM-Verfahren arbeitet auf einer festen Karte aus mit Teilbahnen verbundenen Roboterstellungen (orange, gepunktete Linien), ähnlich einer Straßenkarte: Zuerst ist ein Weg von der aktuellen Roboterstellung auf die Teilbahnen der Karte zu finden (oranger Pfeil). Auf der Karte führt der Roboter dann eine optimierte Bewegung zum am nahsten am Ziel liegenden Punkt einer Teilbahn aus (orange Pfeile). Dort verlässt der Roboter schließlich die vorgespeicherten Teilbahnen und bewegt sich zum Ziel (oranger Pfeil).

Die obige Beschreibung des PRM-Verfahrens genügt bereits, um Roboterbahnen im objektfreien Arbeitsraum zu planen. Seine Stärken zeigt das PRM-Verfahren jedoch erst, wenn man Objekte im Arbeitsraum zulässt. Zwar ist es möglich, alle Objekte im Arbeitsraum als unbekannte Hindernisse zu betrachten, allerdings ergibt sich beim PRM-Verfahren ein großer

Listing 5.14 Initialbefüllung eines PRM-Netzwerks unter Berücksichtigung von unbeweglichen Objekten

```
01Eingabe: Arbeitsraumbelegung mit Kollisionstest
02         Zu erzeugende Zahl an Knoten auf der Straßenkarte
03Ausgabe: Netzwerk aus kollisionsfreien Bahnsegmenten
04SET(roboterstellung) stellungen = EMPTY
05SET(bahnsegment) segmente = EMPTY
06WHILE stellungen.size() < maximale_knotenzahl DO
07|  roboterstellung s = zufall()
08|
09|  IF kollisionsfrei(s) THEN
10|  |  FOR EACH roboterstellung t IN stellungen DO
11|  |  |  IF kollisionsfrei(s, t) THEN
12|  |  |  |  segmente.insert(s, t)
13|  |
14|  |  stellungen.insert(s)
```

Vorteil, wenn man zwischen im Voraus bekannten, unbeweglichen Objekten (z. B. fest montierte Teile der Arbeitszelle) und im Voraus unbekannten Hindernissen (z. B. Menschen und eingebrachte Werkstücke) trennt. Nachstehende Erklärung geht daher zunächst auf die Berücksichtigung von im Voraus bekannten, unbeweglichen Objekten ein, ehe unbekannte Hindernisse in die PRM-Planung aufgenommen werden.

Zur Berücksichtigung von im Voraus bekannten, unbeweglichen Objekten ist lediglich das Erstellen der initialen Karte anzupassen: Vor dem Einfügen einer Roboterstellung ist diese Stellung auf eine Kollision (z. B. mittels Abstandstest) mit den unbeweglichen Objekten zu prüfen. Bei Vorliegen einer Kollision ist die Roboterstellung zu verwerfen, sonst wieder mit Nachbarstellungen der Karte zu verbinden. Beim Verbinden mit Nachbarstellungen sind zudem die Teilbahnen jeder Verbindung (z. B. durch Abtasten oder durch einen lokalen Planer) auf Kollision zu prüfen. Ist zwischen einer neuen Roboterstellung und einer Nachbarstellung keine kollisionsfreie Teilbahn zu finden, so werden beide Stellungen nicht verbunden. Das Ergebnis der Initialbefüllung ist folglich eine Karte an Teilbahnen, die in keinem Fall eine Kollision mit einem im Voraus bekannten, unbeweglichen Objekt verursachen. Pseudocode für diese Variante der Initialbefüllung findet sich in Listing 5.14

Im Gegensatz zu im Voraus bekannten, unbeweglichen Objekten lassen sich im Voraus unbekannte, möglicherweise bewegliche Hindernisse nicht in einem Vorverarbeitungsschritt auswerten. Stattdessen ist die bestehende Straßenkarte im Hinblick auf die aus einer Hindernisdetektion und -lokalisation hervorgehende Belegung des Arbeitsraums anzupassen. Grundidee ist, von Hindernissen belegte Teilbahnen der Karte über Abtastung und einen Abstandstest zu erkennen und für den Roboter kurzfristig zu sperren. Einmal gesperrte Teilbahnen können für einige Zeit gesperrt bleiben, zudem brauchen nur vom Roboter aktuell in Betracht gezogene Teilbahnen auf Hindernisse geprüft werden. So entfällt eine rechenaufwendige Prüfung der Gesamtkarte in jedem Verarbeitungsschritt, stattdessen kann die Karte iterativ an die aktuelle Hindernissituation angeglichen werden.

Ein Beispiel für eine PRM-Planung, die separat unbewegliche Objekte und möglicherweise bewegliche Hindernisse berücksichtigt, ist in Bild 5.70 dargestellt.

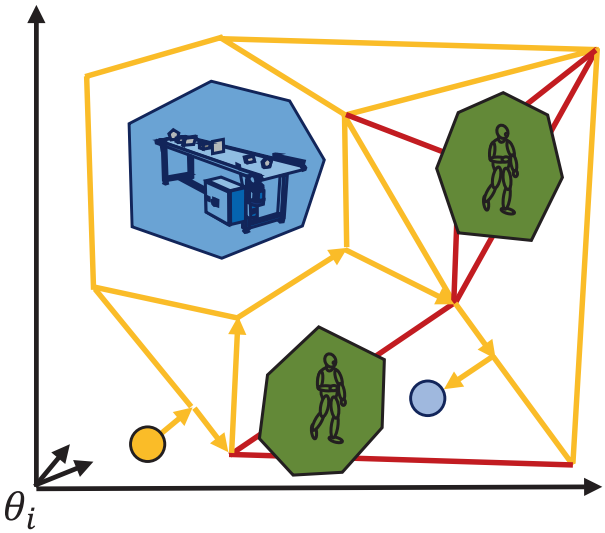

Bild 5.70 Im Voraus bekannte Objekte (blaue Flächen) beinhalten von vornherein keine Verbindungen, während bestehende Verbindungen durch unbekannte Hindernisse (grüne Flächen) kurzfristig für den Roboter gesperrt werden (rote, gepunktete Linien).

Die Trennung zwischen Initialisierung der Straßenkarte und dem Sperren von Verbindungen bei blockierenden Hindernissen führt in der Bewertung des PRM-Verfahrens zu Vorteilen: Das Verfahren ist sowohl *anytime-fähig*, denn zusätzliche Zeit kann zum Aktualisieren der Karte genutzt werden, als auch *inkrementell*, denn eine Kartenbelegung wird von einem Abarbeitungsschritt zum nächsten entnommen. Somit liegt die nötige *Rechenzeit* bei gleichem Weltmodell unter der Rechenzeit eines RRT-Ansatzes. Weitere Vorteile in der Rechenzeit folgen aus dem nun nicht mehr zur Laufzeit nötigen Kollisionstest gegen unbewegte Objekte. Der gesparten Rechenzeit gegenüber steht verständlicherweise ein höherer *Implementierungsaufwand* für die durchgehende Pflege der Karte, einhergehend mit einem deutlich komplexeren *Weltmodell*, das eine Trennung zwischen Hindernissen (z. B. als Voxel) und im Voraus bekannten Objekten (z. B. als Hierarchie) benötigt. Immerhin steigt durch die Karte der *Speicherverbrauch* im Vergleich zu einer Baumstruktur nur unwesentlich.

Betrachtet man lediglich die erzeugten Bahnen, so ist das PRM-Verfahren durch die Nutzung fester Pfade weniger *flexibel*, ebenso wenig *glatt*, doch deutlich besser *vorhersagbar* als das RRT-Verfahren. *Vollständig* ist die PRM-Planung indes nur, wenn für die Straßenkarte zu Beginn beliebig viele Punkte zur Verfügung stehen. Zur Laufzeit ist das PRM-Verfahren dagegen durch die Beschränkung auf bestehende Bahnen im Allgemeinen nicht vollständig. Für *zusätzliche Randbedingungen* ist das PRM-Verfahren dagegen ideal geeignet. Grund ist, dass die durchgängig vorgehaltenen Teilbahnen der Karte mit zusätzlichen Informationen für die Bahnplanung (z. B. über Geschwindigkeitsbegrenzungen) annotiert werden können.

Unter der Voraussetzung einer zuverlässigen Hindernislokalisation ergibt sich für das PRM-Verfahren zuletzt eine hohe *Sicherheit*. Diese folgt aus den für Benutzer intuitiven Bahnen ebenso wie aus der sicheren Sperrung gegenüber bekannten Objekten kollisionsbehafteter Bahnen.

Die Erstveröffentlichung zum PRM-Verfahren ist (Kavraki et al. 1996). Varianten finden sich unter anderem in (Gecks 2011) und (Bohlin, Kavraki 2000).

5.5.5 Vergleich der Verfahren

In diesem Kapitel wurde eine Reihe von Reaktionsstrategien vorgestellt, die zum Behandeln möglicher Kollisionen zwischen Roboter und Hindernissen in Echtzeit geeignet sind. Für jedes genannte Verfahren erfolgte eine Bewertung bezüglich verschiedener Kriterien, um die Auswahl einer geeigneten Strategie für eine gegebene Anwendung zu unterstützen. Tabelle 5.3 fasst die Ergebnisse der Evaluation auf einen Blick zusammen.

Im Rückblick auf die einzelnen Bewertungsergebnisse ist festzuhalten, dass es keine universelle Reaktionsstrategie gibt, die für jede Anwendung geeignet ist. Stattdessen hat jede Strategie individuelle Stärken und Schwächen, sowohl im Hinblick auf die Sicherheit als auch auf die Qualität der entsprechenden Reaktion zur Kollisionsvermeidung. Grundsätzlich lässt sich allerdings bemerken, dass Verfahren mit zunehmender Funktionsvielfalt auch teurer in der Implementierung werden und erhöhte Anforderungen an das Gesamtsystem stellen.

Je nach Anwendung wird also eines der Verfahren im Vordergrund stehen. Ist eine zertifizierbare Sicherheit nötig (beispielsweise im Umfeld der industriellen Fertigung bzw. Produktion), so bietet die Kollisionsentschärfung bei wenigen Annahmen an Sensoren und Robotersystem die härtesten, validierbaren Sicherheitsgarantien. Andersherum ist die Sicherheit bei einem Systementwurf mit globaler Bahnplanung über die nötigen Vorverarbeitungsschritte inklusive Hindernisdetektion und Hindernislokalisation schwer zu validieren. Dafür realisiert ein solches System über dynamisch angepasste Bahnen Roboterreaktionen, die im Kontext der engen Mensch-Roboter-Kooperation deutlich überlegen sind.

In der Praxis ist es auch möglich, verschiedene Strategien in einem Gesamtsystem gleichzeitig zu nutzen. Ein Gedankenexperiment zur Veranschaulichung: Seien Aufwand und Kosten keine wesentlichen Kriterien. Dann ist es ohne Probleme möglich, einen nachgiebigen Roboter mit elastischer Ummantelung zu nutzen, der abstandsabhängig seine Geschwindigkeit auf Hindernisse anpasst. Schließlich kann der Roboter einen lokalen und globalen Bahnplaner kombinieren, um in jeder erdenklichen Hinderniskonstellation sinnvolle Bahnen für die Mensch-Roboter-Zusammenarbeit zu erzeugen.

Tabelle 5.3 Reaktionsverfahren im Überblick

	Sicherheit	Flexibilität	Rechenaufwand	Speicheraufwand	Weltmodell-Komplexität	inkrementelle Planung	Anytime-Fähigkeit	Vorhersagbarkeit	Glattheit	Vollständigkeit	zusätzliche Randbedingungen	Implementierungs-Aufwand
elastische Roboterverkleidung	👎	👎	✖	✖	✖	✖	✖	👎	✖	👎	👎	✖
Notfallbremsung	👍	👎	👍👍	👍👍	✖	✖	✖	😐	✖	👎	👎	👍👍
Nachgiebiger Roboter	👍👍	👎	👍👍	👍👍	✖	✖	✖	😐	✖	👎	👎	👍👍
Geschwindigkeits-R. voxelbasiert	👍	😐	😐	😐	👍	👎	😐	😐	✖	👎	😐	👍
Geschwindigkeits-R. hierarchisch	👍	😐	👍	👍	😐	😐	😐	😐	✖	👎	😐	😐
Potentialfeld voxelbasiert	😐	👍	👎	👎	👍	😐	😐	👎	😐	👎	😐	👎
Potentialfeld hierarchisch	😐	👍	😐	😐	😐	😐	😐	👎	😐	👎	😐	👎👎
Elastic Bands voxelbasiert	😐	👍	👎👎	👎	👍	😐	😐	😐	👍	👎	👍	👎👎
Elastic Bands hierarchisch	😐	👍	👎	😐	😐	😐	😐	😐	👍	👎	👍	👎👎👎
RRT voxelbasiert	👍	👍👍	👎👎	👎👎	👍	👎	👍	👎	😐	👍	😐	👎👎
RRT hierarchisch	👍	👍👍	👎	👎	😐	👎	👍	👎	😐	👍	😐	👎👎👎
PRM (Voxel + Hierarchie)	👍	👍👍	👎	👎👎👎	👎	👍	👍	👍	👎	👍	👍👍	👎👎👎👎

5.5.6 Systemstudie SIMERO

Wie den vorangehenden Kapiteln zu entnehmen ist, gibt es zahlreiche Verfahren zum Erkennen und Lokalisieren von Hindernissen in einer Roboterarbeitszelle und zum Ermitteln einer entsprechenden Roboterreaktion. Zur Implementierung eines sicheren Systems zur Mensch-Roboter-Kooperation ist es daher nötig, eine geeignete Auswahl an Verfahren zu treffen. Diese Verfahren sind dann in eine System-Architektur aus Hardware (z. B. Sensoren, Rechner) und Software (z. B. Roboteransteuerung, Bewegungsplanung) einzubetten.
Ein Beispiel für eine Systemarchitektur, die sich zur Absicherung von Szenarien der Mensch-Roboter-Kooperation eignet, stellt dieser Abschnitt vor. Grundlage der Systemarchitektur bildet die SIMERO-Zelle (SIchere-MEnsch-ROboter-Zelle). Diese Roboterarbeitszelle entstand im Rahmen des DFG-geförderten Projekts SIMERO mit dem Zweck, neuartige Verfahren zur Hinderniserkennung und zur Kollisionsvermeidung zu finden, zu realisieren und zu evaluieren. Bild 5.71 zeigt einen Blick in die Arbeitszelle.

Systemüberblick

Zuerst folgt ein Überblick über Aufgaben und Zusammenhang der in der Zelle genutzten Hardware-Komponenten: Den Kern der Arbeitszelle bildet ein Stäubli-RX130-Roboterarm, der an einer CS7-Steuerung betrieben wird. An der Decke des Raums überwacht während des Roboterbetriebs ein Multikamera-System aus acht Kameras den Arbeitsraum auf dynamische Hindernisse. Bei den Kameras handelt es sich um kostengünstige, hochauflösende Endanwender-Kameras vom Typ Logitech C930e. Jede Kamera ist über USB 3.0 mit einem Vorverarbeitungs-Rechner verbunden. Als Vorverarbeitungs-Rechner sind in der SIMERO-Zelle Kleinstrechner von Shuttle im Einsatz. Der aufwendige Schritt der Hindernisdetektion wird dort auf preisgünstigen Grafikkarten vom Typ NVIDIA