Automatisierte Palettierung mit Mehrfachgreifern

Dem Fachbereich Maschinenbau
der Universität Hannover
zur Erlangung des akademischen Grades
Doktor Ingenieur
genehmigte Dissertation

von
Dipl.- Ing. Thomas Kühn
geboren am 6. November 1968
in Berlin

1999

1. Referent Prof. Dr.-Ing. Dr.-Ing. E.h. mult. H. K. Tönshoff
2. Referent Prof. Dr.-Ing. B. Heimann
Tag der Promotion: 27.9.1999

Inhaltsverzeichnis

Formelzeichen

P	Feld mit Packstücken
i,j,k	Zählvariablen
B_x	Anzahl der Packstücke eines Palettenblocks in x
B_y	Anzahl der Packstücke eines Palettenblocks in y
G_x	Anzahl der Packstücke eines Greiferblocks in x
G_y	Anzahl der Packstücke eines Greiferblocks in y
N, M	Zählvariablen
++	inkrementiere um eins
$[i]$	Zugriff auf ein Feld an der Stelle i
v_{max}	maximale Geschwindigkeit
a_{max}	maximale Beschleunigung
T_{FIPO}	Feininterpolationszeit (z. B. 8ms)
$t_{\ddot{u}}$	Überschleifzeit
q_b	zu überschleifende Stützstelle
q_c	folgende Stützstelle
q_a	Ort zu Beginn des Satzüberganges
t_s	Satzlänge des folgenden Satzes
$B_0..B_5$	Polynomkoeffizienten
r_0	Ruck (Ableitung der Beschleunigung)
ü	Überschleiffaktor
.	Zugriffoperator auf einzelne Elemente einer Struktur
$\Delta\vec{D}$	Differenzvektors zum Mittelpunkt
$\vec{A}$	Anschlagvektor der Abgabe (Palette)
$\Delta\vec{N}$	relativen Position des Packstückes zur linken unteren Palettenecke
$Rest_x$	Anzahl der restlichen Packstücke in x-Richtung nach einer Subblockbildung
$Rest_y$	Anzahl der restlichen Packstücke in y-Richtung nach einer Subblockbildung

Begriffe, Definitionen

Packmuster	Anordnung von quaderförmigen Packstücken unterschiedlicher Abmaße auf einer Palette
Mehrfachgreifer	mehrere, quaderförmige Packstücke gleicher Abmessung werden gleichzeitig gegriffen
x-Block	Block aus Packstücken, die in x-Richtung auf einer Palette aneinander anliegen
Schnittblock	Block aus Packstücken, der an mindestens zwei weiteren Blöcken bündig anliegt
Subblock	Block in einem Block z. B. Greiferblock in einem Palettenblock
(Greifer-) Zyklus	Packstücke abholen und ablegen
Vereinzelung	Mehrere Packstücke abholen und einzeln ablegen
Einfügerichtung	Die Richtung aus der Packstücke auf die Palette gelegt werden
Zyklusabbruch	Vorzeitiges Zyklusende. Der Greifer wird nicht vollständig aufgefüllt
Metablock	Block, der in Subblöcke (Vielfaches des Greifers aufgeteilt werden kann).
IRL	Industrial Robot Language
SPS	Speicherprogrammierbare Steuerung
ASCII	American Code for Information Interchange
IO	Input Output
IPC	Industrie Personal Computer
CAN	Control Area Network
FIFO	First In First Out
ST	Strukturierter Text
PTP	Point to Point
OCX	Object linking and embedding control
RC	Robot Control
NC	Numerical Control
FIPO	Feininterpolation

Abstract

Am Ende des Produktionsprozesses erfordert die steigende Variantenvielfalt in immer kürzeren Zeitabständen eine schnelle und flexible Kommissionierung der Produkte. Dies kann im wesentlichen durch den Parallelbetrieb mehrerer Anlagen oder durch die Umrüstung einzelner Anlagen auf Mehrfachgreifer erreicht werden. Palettieranlagen mit Mehrfachgreifern werden aufgrund der großen Kollisionsgefahren mit Anlagenteilen vielfach noch im Lernbetrieb (Teachen) programmiert, was sehr zeitaufwendig ist und zu hohen Stillstandszeiten führt. Packmuster werden somit ausschließlich durch Techniker des Anlagenherstellers erstellt und geändert.

Im Rahmen dieser Arbeit wurde ein Palettiersystem zur automatischen, aber dennoch flexiblen Palettierung mit Handhabungssystemen beliebiger Kinematik unter Berücksichtigung von Mehrfachgreifern entwickelt. Dieses System bietet eine durchgängige Lösung von der Packmustererzeugung über die optimierte Bildung von Palettierzyklen bis hin zur schnellen mechanikschonenden Achspositionierung (PTP-Spline). Konkret wurden ein graphisches Off-line Programmiersystem für Mehrfachgreifer und eine PC-basierte (Palettieranlagen-) Steuerung entwickelt.

Ausgangspunkt ist die auftragsbezogene Kommissionierung eines beliebigen dreidimensionalen Packmusters mit Packstücken beliebiger Größe. Es werden einige Verfahren zur Bildung von geschwindigkeitsoptimierten und kollisionsfreie Ablagezyklen (Packprogramme) vorgestellt, die auf mathematischen Modellen für Palettieranlagen und Packmustern (Graph) basieren. Die Packprogramme werden direkt aus den Packmustern automatisch erzeugt, so daß eine Programmierung des Roboters für den Bediener vollständig entfällt.

Schlägwörter: Roboter, Palettierung, Greifer, Off-line, Steuerung, Algorithmus

Abstract

The increasing number of products forces a quick and flexible commission at the end of production's process. This can be achieved by the use of several parallel palletizing plants or by upgrading single plants to multiple grippers. Up to now the programming of these plants is done by teaching the positions, which takes time and is extremely expensive. The average worker (plant user) is not able to change or to add any pattern to the plant. This is mostly done by the plant manufacturer's technician.

This paper shows a new palletizing system that generates a robot programs from a given pattern automatically. Therefore it was necessary to build a mathmatical plant model and a pattern graph model. A search algorithm looks for the fastest way to place packages on the pallet without any collisions. The palletizing pattern can be generated automatically or manual on a graphic desktop.

Additionally a new PC-based control was made to replace Plc and robot controls by only one PC including a new PTP-spline axle positioning algorithm.

Keywords: robot, palletizing, gripper, off-line, control,algorithm

Kapitel 1

Einleitung

Kürzere Produktionszeiten und größere Variantenvielfalt erfordern gerade am Ende eines Fertigungsprozesses einen schnellen Abfluß der Produkte. Dies wird hauptsächlich durch den parallelen Einsatz mehrerer Palettieranlagen und/oder Aufrüstung auf Mehrfachgreifer innerhalb der Anlagen erreicht. Mehrfachgreifer fassen eine Vielzahl quaderförmiger, gleichartiger Packstücke und legen diese entweder vollständig oder teilweise nacheinander ab. Der Einsatz von Mehrfachgreifern führt einerseits zu einer deutlichen Geschwindigkeitssteigerung der Anlage, andererseits müssen größere Verfahrbereiche sowie aufwendigere Kollisionsüberwachung und Programmierung in Kauf genommen werden. Um flexibel auf die große Variantenvielfalt reagieren zu können, sind sehr kurze Programmierzeiten der Palettieranlagen gefordert, was insbesondere bei komplexeren Anlagen mit Mehrfachgreifern in der Regel nicht erfüllt werden kann. Die Programmierung solcher Anlagen erfolgt vielfach noch im Lernbetrieb (Teachen), der extrem zeitaufwendig ist und zu hohen Stillstandzeiten führt. Daraus resultiert, daß das Hinzufügen und Ändern von Packmustern fast ausschließlich Technikern vorbehalten ist. Aus diesem Grunde werden auch heute noch Palettieranlagen mit einer konstanten Zahl an Packmustern ausgeliefert, die für den Kunden ohne Programmierkenntisse nur schwer zu ändern sind.

Eine Palettieranlage (Abb. 1.1) besteht im wesentlichen aus den Komponenten:

- Abholstation (Zuführtisch, Transportband, Querschieber)
- Handhabungseinheit (Roboter oder Portal)
- Abgabestation (Palette).

Grundsätzlich sollen die auf dem Transportband ankommenden Packstücke möglichst schnell auf den Paletten plaziert werden. Das Handhabungssystem greift die Packstücke von der

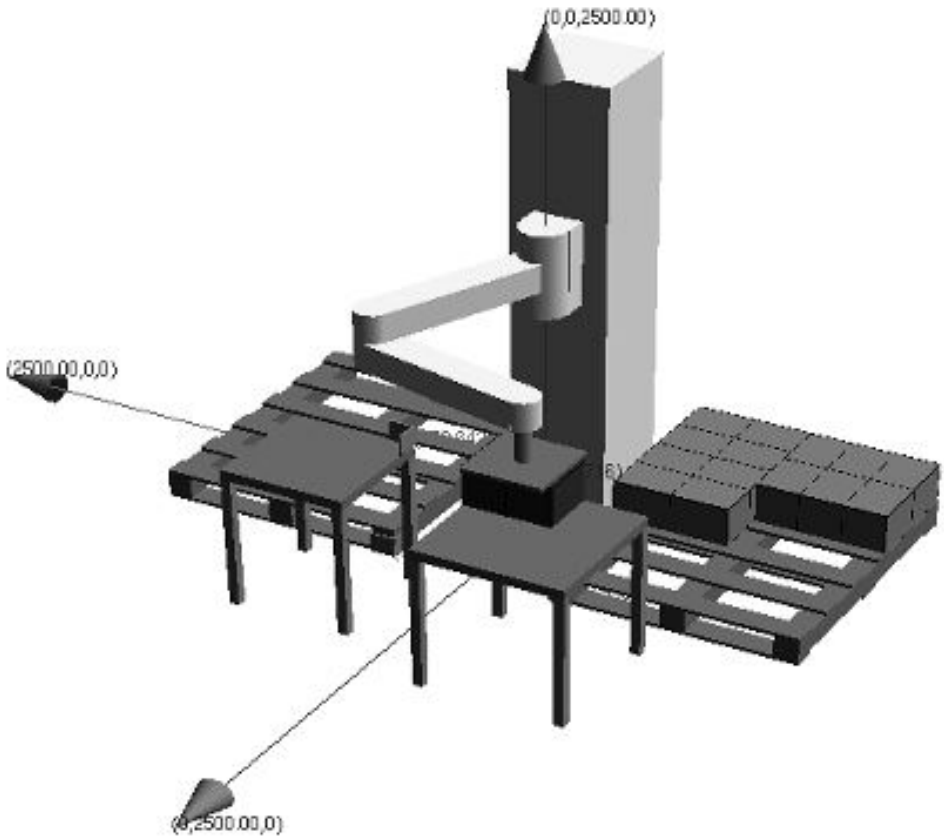


Abbildung 1.1: Beispielhafter Aufbau einer Palettieranlage

Abholstation, fährt einige Zwischenpunkte über der Abholstation und über der Palette an und legt die Packstücke dann auf einer Palette ab.

Die Greifer lassen sich allgemein nach folgenden Kriterien klassifizieren:

1. Anzahl der gleichzeitig greifbaren Packstücke
   - Einfachgreifer: Ein Packstück greifen und ablegen
   - Mehrfachgreifer: Gleichzeitig mehrere gleichgroße Packstücke von den Abholstationen aufnehmen und anschließend entweder komplett oder einzeln nacheinander abgelegen (Vereinzelung) .
2. Bauart des Greifers
   - Sauggreifer: Die Packstücke werden von oben angesaugt. Folglich wird diese Bauform überwiegend für leichte Packstücke eingesetzt.
   - Zangengreifer: Die Packstücke werden von zwei Greiferelementen eingeklemmt. Beim Ablegen der Packstücke entstehen evt. Lücken. Diese Bauform wird haupt-

sächlich für mittelschwere Packstücke verwendet. Die Abholtische müssen eine Pneumatikeinhalt enthalten, die das Packstück anhebt.

- Gabelgreifer: Die Packstücke werden auf gabelartige Stangen geschoben, die beim Ablegen in eine (Rückzug-) Richtung weggefahren werden. Diese Bauform ist vorwiegend für schwere Packstücke geeignet und erfordert spezielle Abholtische, die Lücken in Form der Greifergabeln enthalten. Beim Ablegen muß aureichend Platz für die zurückgezogenen Gabeln vorhanden sein.

Die Anordnung der abgelegten Packstücke auf der Palette ergibt ein bestimmtes Packmuster, das entweder zweidimensional (d. h. lagenweise typenrein) oder dreidimensional (d. h. unterschiedliche Packstückgrößen im Mix) ist.

Der prinzipielle Ablauf vom Auftragseingang bis zur Palettierung sieht grob wie folgt aus:

1. Auftragsdaten auswerten (Packstück- und Palettenmaße)
2. Packmuster erzeugen (Optimierungsprogramme, off-line)
3. Roboterprogramme generieren (automatisch oder Teachen, on-line)
4. Roboterprogramme laden (Download) und auswerten (Parsen)
5. Positionierung durchführen

Der Anlagenbediener extrahiert zuerst die Packstückmaße entweder manuell aus Auftragsblättern oder automatisch aus den Daten eines übergeordneten Leitstandes. Nachfolgend wird mit Hilfe eines Optimierungsprogramms oder manuell eine Anordnung von Packstücken auf der Palette (Packmuster) generiert. Erst jetzt kann aus den Packstücken ein Roboterprogramm erzeugt und anschließend gestartet werden. Zur vollständigen Automatisierung des Palettiersystemes müssen noch Schaltinformationen (-> SPS) für die Abhostationen und Greifer bereitgestellt werden. Die reine Erzeugung von Positionsinformationen ist nicht ausreichend (Abb. 1.2), da z. B. Informationen für die Abholstationen und Kippeinrichtungen evtl. einige Zyklen im Voraus benötigt werden. Kippeinrichtungen drehen die Packstücke um 90 Grad entweder um die x- oder y-Achse. Das ist notwendig, wenn diese Drehung zu einer besseren Raumausnutzung führt und das Handhabungssystem aufgrund der Kinematik keine derartige Drehung durchführen kann.

Bisherige Arbeiten auf diesem Gebiet haben sich hauptsächlich mit der Erzeugung von Packmustern beschäftigt, jedoch nicht mit der automatisierten Palettierung von dreidimensionalen Packmustern mit Mehrfachgreifern im Produktionsprozeß (siehe Stand des Wissens).

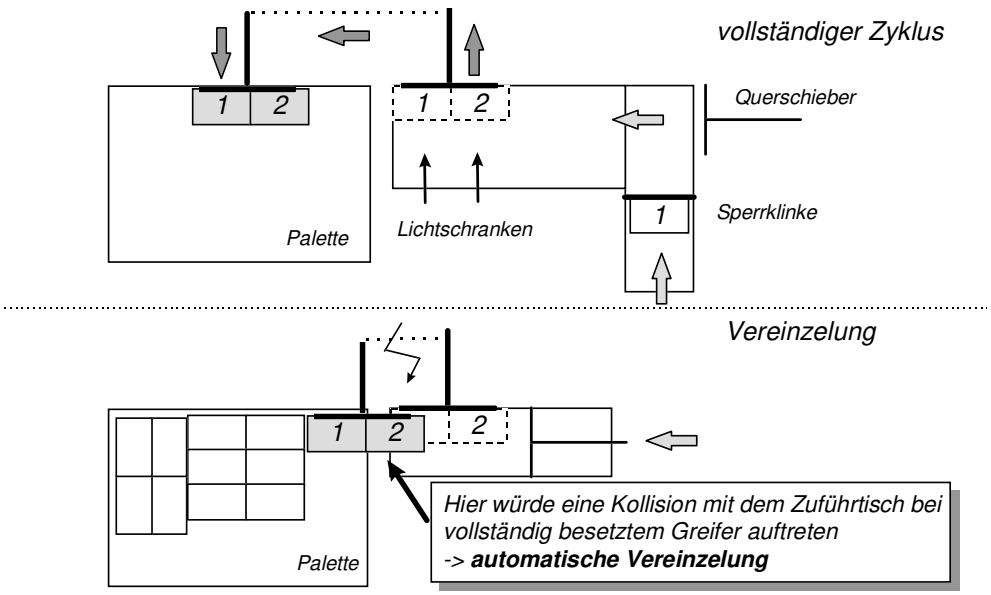


Abbildung 1.2: Daten für die SPS

Es existieren keine durchgängigen Konzepte von der off-line Programmierung bis hin zur Positionierung der Packstücke auf der Palette.

Diese Arbeit gliedert sich daher in zwei Teilbereiche:

- Off-line Programmiersystem (Palettiersystem) für Mehrfachgreifer und
- PC-basierte Steuerungskonzepte mit Positionierung .

Zuerst werden aus gegebenen Packmustern geschwindigkeitsoptimierte, kollisionsfreie Packprogramme automatisch erzeugt, die dann von einer PC-basierten Steuerung abgearbeitet werden. Da die eigentliche Roboterprogrammierung komplett entfällt, ist die Palettieranlage auch für Anwender ohne Programmierkenntnisse zu handhaben. Die durchgängig PC-basierte Systemlösung führt zu stark reduzierten Programmier- und Inbetriebnahmezeiten, insbesondere weil alle Steuerungskomponenten auf die Softwareebene verlagert wurden. Zur Reduzierung der on-line Packzeiten wurden im Rahmen des Steuerungskonzeptes einige Positionierungsverfahren untersucht und ein neues Positionierungsverfahren (Spline-Ptp) entwickelt. Die Implementierung der Positionierung erfolgte dabei komplett im Steuerungs-PC. In der Palettiertechnik ist die Positionierdauer von wesentlicher Bedeutung, weniger das exakte Einhalten einer bestimmten Bahn wie z. B. bei Werkzeugmaschinen. Wenn gewichtsreduzierte (Parallel-) Kinematiken vollständig ausgenutzt werden sollen, müssen Interpolationsverfahren mit Ruckbegrenzung eingesetzt werden. Im Rahmen dieser Arbeit wurden Spline Interpolationsverfahren auf die Einsatzmöglichkeit in der Palettiertechnik entwickelt und untersucht.

Kapitel 2

Stand des Wissens

Generell existieren drei Methoden, um Packstücke auf Paletten zu plazieren. Die erste ist die manuelle Palettierung, die nur für leichte Waren und geringe Geschwindigkeiten tauglich, aber sehr flexibel ist. Die zweite Methode ist die Nutzung von Palettierautomaten [26], die sehr schnell, aber unflexibel sind. Die dritte Methode ist der Einsatz von Robotern, die mit Hilfe von geeigneter Software schnell und flexibel arbeiten. Nachfolgend wird ausschließlich die Palettierung mit Robotern betrachtet.

In der Robotik wird zwischen off-line und on-line Programmierung getrennt. Unter on-line Programmierung versteht man hauptsächlich die Lernprogrammierung (Teachen), die aber für viele Anfahrpunkte zu aufwendig ist (siehe Einleitung). Für die Palettierung ist der Bereich off-line Programmierung von wesentlicher Bedeutung, der sich in

- Erzeugung von Packmustern,
- Palettierung bei unbekannter Kommissionierung und
- Palettierung bei bekannter Kommissionierung.

gliedert und in diesem Kapitel eingehender untersucht wird. Kommissionierung wird nach VDI 3590 als das *Zusammenstellen von bestimmten Teilmengen (Artikeln) aus einer bereitgestellten Gesamtmenge (Sortiment) aufgrund von Bedarfsinformationen (Aufträgen)* definiert.

Die off-line erzeugten Daten und (oder) Programme müssen letztlich von den Steuerungen ausgewertet und in Roboterbewegungen sowie Schaltinformationen umgesetzt werden. Am Ende dieses Kapitels werden die bisher in der Literatur erwähnten

- Steuerungskonzepte und

- Positionierungsalgorithmen

verglichen und die mögliche Eignung für Palettieranlagen überprüft.

2.1 Erzeugung von Packmustern

Im Bereich der Palettierung hat sich die Wissenschaft bisher hauptsächlich mit der Lösung des zwei- und dreidimensionalen Packproblems beschäftigt [30], [31], [33] , [34], [35], [36], [37], [38] , [48], [49], [50], [43]. Die Aufgabenstellung dieser Optimierung lautet hier, möglichst viele Pakete gleichen oder unterschiedlichen Typs in einem gegebenen Raum oder einer Fläche zu plazieren. Hier existieren vielfältige Lösungen, die auf heuristischen Suchverfahren oder zufälligen Ansätzen (z. B. genetischer Algorithmus [70]) basieren. Als Ergebnis des jeweiligen Algorithmus wird ein Packmuster geliefert, das nach verschiedenen Kriterien (z. B. Stabilität) optimiert wurde ([41], [42]). Dabei wird nach sogenannten Blockpackmustern und optimierten Mustern unterschieden.

Einfache Mustergeneratoren arbeiten mit Blockheuristiken [52]. Für die eingegebenen Packstückmaße werden mehrere fest gespeicherte Mustertypen durchlaufen und das Muster mit der größten Anzahl von Packstücken ausgegeben. Blockheuristiken liefern schnelle, aber nur suboptimale Lösungen.

Demgegenüber stehen reine Optimierungsprogramme, die meist auf branch and bound oder anderen Suchverfahren ([90], [61]) basieren. In der Regel wird ein Suchbaum aufgebaut, der in einer bestimmten Suchtiefe rekursiv durchlaufen wird. Eine Bewertungsfunktion entscheidet dann, in welcher Richtung der Graph weiter durchsucht wird. Reine Optimierungen beanspruchen bei großen Packmustern enorme Rechenzeiten, liefern daher aber eventuell auch bessere Lösungen als Blockheuristiken. Folglich werden Blockheuristiken überwiegend dann eingesetzt, wenn schnelle Lösungen benötigt werden. In der Literatur wird die Erzeugung von Packmustern auch häufig als Packproblem bezeichnet.

2.1.1 Das Packproblem

Packprobleme beschäftigen sich allgemein mit der Frage, wie sich eine vorhandene Ressource optimal ausnutzen läßt. Dabei sind in erster Linie solche Probleme gemeint, bei denen kleinere Einheiten in größere Einheiten (z.B. Behälter) gepackt werden und eine vorgegebene Optimierung erreicht wird. Damit unterscheiden sie sich von einer anderen Gruppe von Optimierungsproblemen, die zum Beispiel zum Ziel haben, die ungenutzte Fläche beim

Zuschnitt von Rechtecken aus einer rechteckigen Grundplatte zu minimieren [44], [45].

Diese als Zuschnittproblem zu klassifizierende Aufgabe stellt oftmals nur eine andere Betrachtung des gleichen Problems dar und kann in dieser Sichtweise auch als Packproblem angesehen werden. Es macht in vielen Fällen keinen wesentlichen Unterschied, ob eine vorliegende Grundfläche in Objekte zerlegt werden soll, so daß der Verschnitt minimiert wird, oder ob die Aufgabe darin besteht, die Grundfläche so mit Objekten zu bepacken, daß die ungenutzte Fläche minimal wird. Aufgrund dieser Dualität wird zwischen beiden Problemarten auch nicht streng unterschieden. Trotzdem lassen sich Algorithmen des Verschnittproblems nicht immer zur Lösung des Packproblems einsetzen. Bei eindimensionalen Problemen ist eine solche Übertragung noch möglich, doch gerade bei den besonders interessierenden Lösungsverfahren für zwei- und dreidimensionale Packprobleme ist eine Adaption von Zuschnitt- auf Packprobleme nicht möglich (vgl. [32], [51], [46]).

Bei zwei- und dreidimensionalen Zuschnittproblemen ist besonders problematisch, daß ein durchgängiger Schnitt von der einen zur anderen Seite des zu zerteilenden Rechtecks bzw. Körpers erforderlich ist. Diese Art wird als Guillotine-Schnitt bezeichnet.

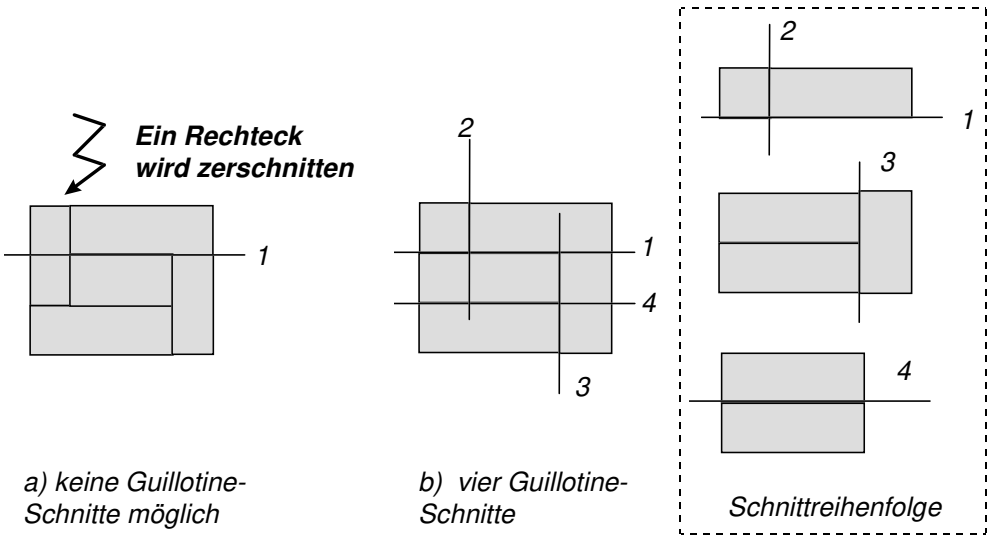


Abbildung 2.1: Guillotine-Schnitte

Die in Abb. 2.1 Bild a dargestellte Aufteilung ist mit Guillotine-Schnitten nicht erreichbar, da bei jedem durchgängigem Schnitt mindestens ein anderes Rechteck zerschnitten wird. Dahingegen alle Rechtecke in Bild b mit vier Guillotine-Schnitten realisiert werden kann.

Für zweidimensionale Zuschnittprobleme existieren Lösungstechniken, die aber Guillotine-Schnitte verlangen. Dies stellt in vielen Bereichen allerdings keine Einschränkung dar, da zum Beispiel in der glas- oder holzverarbeitenden Industrie ohnehin nur Guillotine-Schnitte

ausgeführt werden können. Bei Packproblemen ist eine Beschränkung auf Guillotine-Schnitte nicht realistisch und deshalb auch eine Adaption der Lösungsverfahren nicht sinnvoll.

2.1.1.1 Klassifizierung von Packproblemen

Aufgrund der großen Anzahl unterschiedlicher Zuschnitt- und Packprobleme ist eine Klassifizierung nach mehreren Kriterien notwendig (nach [40], [39]):

1. Dimension des Problems: N=1, N=2 (Abb. 2.2), N=3 (Abb. 2.3 a)) ;
2. Art der Zuordnung:

   a) Die begrenzte Anzahl von großen Objekten ist vorgegeben, in die möglichst viele kleine Objekte hineingepackt werden sollen (z. B. zweidimensionales Packproblem: Wie viele Packstücke von Typ A können auf eine Palette gepackt werden ?)

   b) Eine konstante Anzahl kleiner Objekte ist gegeben und möglichst wenig große Objekte gepackt werden. Ein praktisches Beispiel ist die auftragsbezogene Kommssionierung: z. B. 300 Packstücke von Typ A und 400 Packstücke von Typ B sind gegeben. Wie viele Paletten werden benötigt ?
3. Vorrat an großen Objekten (z. B. Paletten)

   a) es existiert nur ein großes Objekt (z. B. eine Palette),

   b) die Gestalt aller großen Objekte ist gleich (z. B. mehrere Paletten),

   c) es treten unterschiedliche Formen auf (z. B. mehrere Paletten mit unterschiedlichen Abmessungen)
4. Vorrat an kleinen Objekten (z. B. Packstücke)

   a) mit verschiedenen Formen oder Abmessungen (-> dreidimensionales Problem),

   b) alle sind gleichartig (-> zweidimensionales Problem),

Mit dieser Klassifizierung lassen sich zumindest annähernd Zuschnitt- und Packprobleme charakterisieren. Spezielle Eigenschaften, wie zum Beispiel die Beschränkung auf Guillotine-Schnitte, sind nicht berücksichtigt.

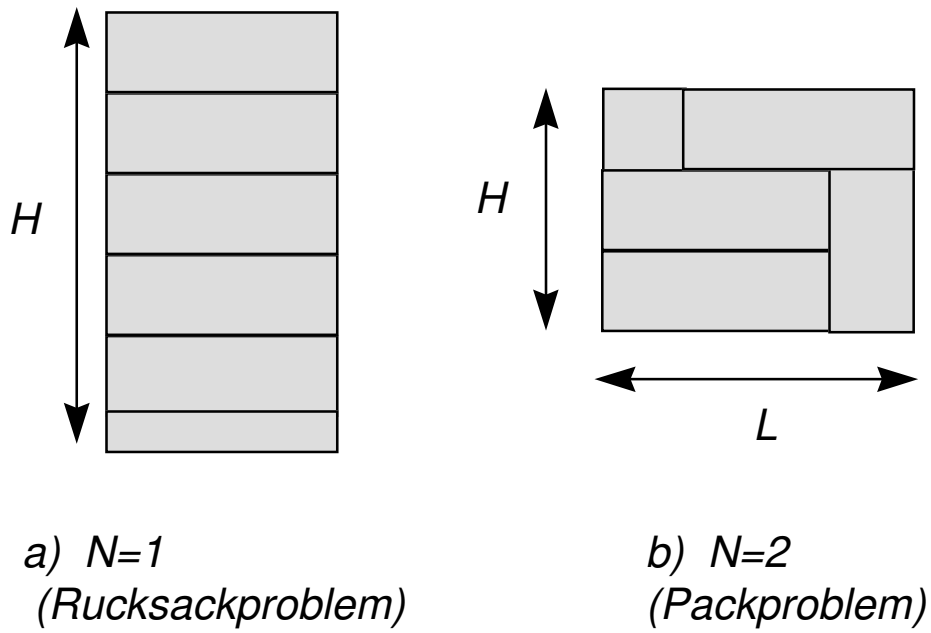


Abbildung 2.2: Klassifizierung von Packproblemen

2.1.1.2 Das Palettenbeladungsproblem

Das Problem der Palettenbeladung besteht darin, kleinere Einheiten in einem im allgemeinen erheblich größeren Quader anzuordnen, der durch die Grundfläche der Palette und deren maximaler Stapelhöhe begrenzt ist.

Hierbei handelt es sich um ein dreidimensionales Problem (Abb. 2.3), das aufgrund seiner Komplexität bisher kaum untersucht wurde. Durch die dritte Dimension müssen nicht nur Kapazitäts-, sondern auch Stabilitätsrestriktionen berücksichtigt werden, da die Anordnung einer Einheit oberhalb der ersten Ebene nur möglich ist, wenn diese durch darunterliegende Einheiten abgestützt ist.

Der Ansatz zur Lösung besteht also in einer Zerlegung des Problems in ein zweidimensionales und ein eindimensionales Problem. Zuerst wird bestimmt, wieviel Einheiten sich maximal auf die Grundfläche packen lassen (zweidimensionales Problem Abb. 2.3 c und d), in einem zweiten Schritt bestimmt man die maximale Anzahl der zu packenden Schichten (eindimensionales Problem).

Für die Lösung des eindimensionalen Problems existieren in der Literatur zahlreiche Lösungsmöglichkeiten (sogenanntes Rucksackproblem [42], [41]). Im folgenden wird daher nur noch auf das zweidimensionale Packproblem eingegangen.

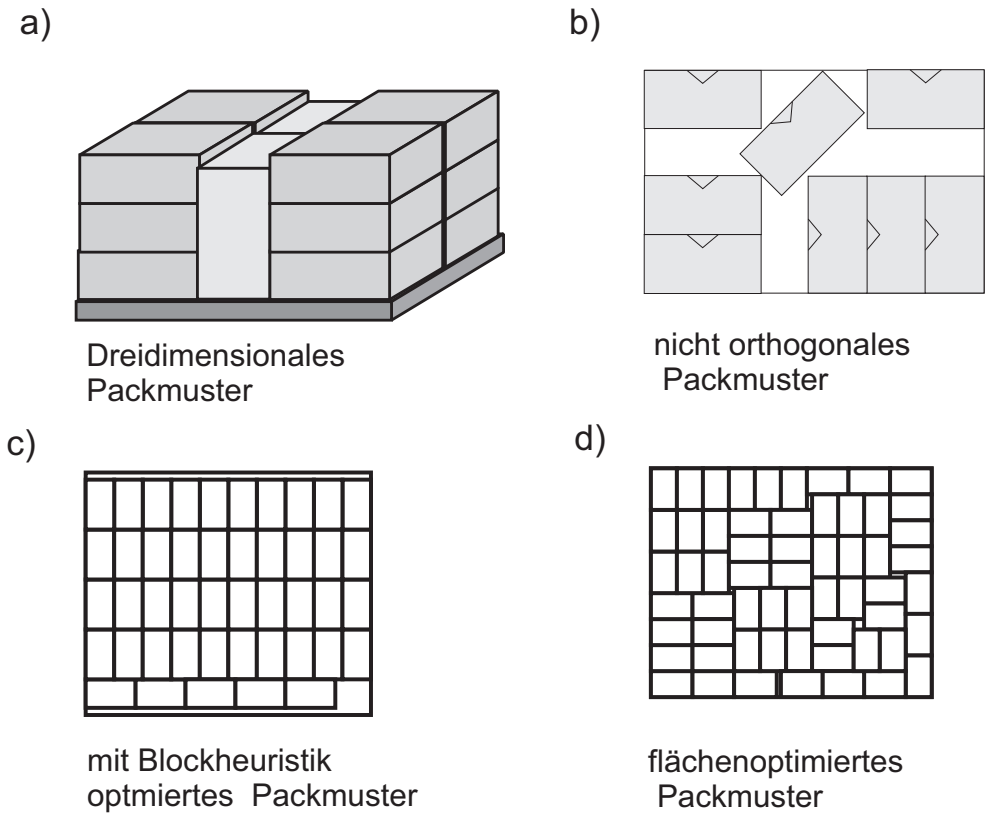


Abbildung 2.3: Packmuster

2.1.1.3 Das zweidimensionale Packproblem

Für praktische Anwendungen lassen sich eine Reihe von Vereinfachungen des allgemeinen zweidimensionalen Problems angeben. Als Grundfläche wird, wie oben schon angenommen, die Form eines Rechtecks gewählt. Ebenso werden als Packstücke nur Quader betrachtet, deren Grundfläche auch Rechtecke sind. Weiterhin wird vorausgesetzt, daß alle Packstücke kongruent sind (homogenes Packproblem), d.h. sie haben alle die gleiche Länge und die gleiche Breite. Eine Rotation der Einheiten ist natürlich zulässig, d.h. daß die Längsseite der Packstücke nicht parallel zur Längsseite der Palette gepackt werden muß. Schließlich kommt als Restriktion noch die Orthogonalitätsbedingung dazu, d.h. daß die Packstücke nur parallel zu den Seiten der Grundfläche angeordnet werden dürfen. Es gibt einige Fälle, in denen der Verzicht auf diese Bedingung zu Lösungen mit einer größeren Anzahl von Packstücken führt. Abb. 2.3 b) zeigt eine nicht-orthogonale Anordnung mit 8 Packstücken. Dabei sind die Maße der Palette 1200*800 und die der Packstücke 420*210. Ein orthogonales Packmuster erlaubt nur das Anordnen von 7 Packstücken.

In der Praxis haben nicht-orthogonale Anordnungen nahezu keine Bedeutung. Sie lassen sich rechnerisch nur mit großem Aufwand ermitteln (Problem der Normalisierung der nicht-

orthogonalen Packstücke) und sind aus Stabilitätsgründen, sowie wegen der Gefahr, daß sich Packstücke gegenseitig beschädigen, für die Praxis nicht relevant.

Zusammenfassend läßt sich das zweidimensionale orthogonale homogene Packproblem wie folgt definieren :

Gegeben ist eine Grundfläche mit der Länge X und der Breite Y und kleine kongruente Rechtecke mit der Länge a und der Breite b. Ziel ist es, unter Berücksichtigung von Kapazitätsrestriktionen, so viele Einheiten wie möglich auf der Grundfläche anzuordnen. Hierbei sind nur orthogonale Anordnungen erlaubt. Die Packstücke dürfen gedreht werden.

2.1.1.4 Das dreidimensionale Packproblem

Die Erweiterung des zweidimensionalen Problems um die dritte Dimension erschwert die Suche nach einer guten Lösung erheblich, da hier wesentlich mehr Möglichkeiten existieren. Daher existieren für dreidimensionale Probleme erst sehr wenige Algorithmen, die häufig auf Heuristiken oder linearer Programmierung (Optimierung) beruhen ([29]).

Schepers [28] entwickelte zwei neue Ansätze, die nicht auf linearer Programmierung basieren:

Im ersten Ansatz werden Packungen mit Hilfe von Intervallgraphen charakterisiert. Enthaltene Symmetrien werden dann zu ganzen Packmusterklassen zusammengefaßt. Die graphentheoretische Formulierung ermöglicht eine sehr kompakte Beschreibung und Speicherung der geometrischer Zusammenhänge. Beim zweiten Ansatz wurde eine allgemeine Konstruktionsmethode für effektive und effizient berechenbare Schranken entwickelt. Der auftretende Verschnitt wird dabei miteinbezogen.

2.2 Palettierung bei unbekannter Kommissionierung

Hauptmerkmal der Palettierung bei unbekannter Kommissionierung ist das Zuführen verschiedenartiger Packstücke in zufälliger Reihenfolge. Praktische Beispiele sind Postverteilstellen und Versandhäuser. Ein vollständiges Packmuster für eine Palette kann vorher nicht berechnet und nicht optimiert werden, weil nicht alle Packstücke bekannt sind. Ein (online) Palettieralgorithmus plaziert die ankommenden Packstücke nach bestimmten Kriterien (Heuristiken) auf der Palette, wobei im Vergleich zum dreidimensionalen Packproblem insgesamt nur wenig Möglichkeiten entstehen. Um bessere Gesamtlösungen, d. h. bessere Raumnutzung zu erhalten, werden die ankommenden Packstücke in einen Zwischenspeicher geschoben. Lee [23] verwendet einen Speicher für zwei, Strommer [21] einen Speicher

für sechs Packstücke. Strommers System erreicht nach eigenen Angaben einen Füllungsgrad der Palette von 70 bis 80 % bei einer Palettierzykluszeit von 8 s mit einem SCARA-Roboter.

Zusammenfassend kann das Problem der Palettierung bei unbekannter Kommissionierung als eine vereinfachte (on-line) Variante des dreidimensionalen Packproblemes aufgefaßt werden. Verallgemeinert stellt sich der vereinfachte Palettierablauf wie folgt dar:

1. Packstück vermessen (Bildverarbeitung)
2. Packstück zwischenspeichern
3. geeignetes Packstück aus Zwischenspeicher auswählen (Algorithmus)
4. Packstück auf der Palette plazieren.

Ähnliche Verfahren werden in [25], [73], [74] und [75] beschrieben. In allen diesen Fällen wurden Sauggreifer eingesetzt, die ein Packstück greifen und absetzen können. Nur Penington [25] berücksichtigte bisher einen Zangengreifer. Sein entwickelter Algorithmus vermeidet die Kollisionsgefahr der Greiferzangen mit bereits liegenden Packstücken durch eine geeignete Ablagereihenfolge.

2.3 Palettierung bei bekannter Kommissionierung

Die bisher untersuchte Palettierung bei bekannter Kommissionierung wurde oft als Koordinatenausgabe von Packmustergeneratoren realisiert ([71], [72], [104], [27], [91], [93], [101], [102]). Größtenteils werden nur Einfachgreifer berücksichtigt. Ausnahme ist Multiscience [104], die auch eine Lösung für Doppelgreifer anbietet. Das Packmuster wird von einer Einfügeecke ausgehend sortiert und die Daten in Anlagenkoordinaten ausgegeben. Dabei können vermehrt Kollisionen auftreten, wenn weder die Kinematik des Handhabungsgerätes noch die Anlagenkomponenten berücksichtigt werden. Die Art der Datenausgabe erfolgt entweder in Reinform oder in der spezifischen Steuerungssprache. Verallgemeinert stellt sich der vereinfachte Palettierablauf für Einfachgreifer wie folgt dar:

1. Packmuster erzeugen (lagenweise oder dreidimensional)
2. Packmusterdaten sortieren und in Anlagenkoordinaten ausgeben
3. (optional) Ablaufprogramm in einer Steuerungssprache mit den Packstückkoordinaten erzeugen

4. Übertragung zur Steuerung und Abarbeitung.

Der oben dargestellte Ablauf gilt nur für lagenweise sortenreine Packmuster und somit nicht für dreidimensionale Packmuster.

Tsai [92] zeigt eine Lösung für dreidimensionale Packmuster auf, indem er ein Netzwerk nach der sogenannten Critical Path Methode aufbaut, um eine kollisionsfreie Ablagenreihenfolge (hier nur in z-Richtung) zu garantieren. Abb. 2.4 zeigt ein solches Netzwerk für ein Packmusterbeispiel nach Tsai. Das dreidimensionale Packmuster ist in vier zweidimensionale Ebenen aufgeteilt. Die Knoten des Netzwerkes repräsentieren die Packstücknummern, die Pfeilespitzen zeigen in Richtung des nächsten ablegbaren Packstückes. Im Umkehrschluß gilt, daß kein Packstück abgelegt werden kann auf das noch Pfeile gerichtet sind - somit noch Vorgänger haben. Die Bildung einer Ablagenreihenfolge erfolgt durch Auswahl eines Knotens ohne Vorgänger. Abschließend werden die Knoten gelöscht. Tsai berücksichtigt weder Mehrfachgreifer noch eine Einfügerichtung. Seine Versuchsanlage besteht aus einem Roboter mit Einfachsauggreifer, einem Fließband, vier Palettenplätzen auf einem Drehtisch und einem Zwischenspeicher. Die zufällig ankommenden Packstücke werden identifiziert und das korrespondierende Packmuster ausgewählt. In der Robotersteuerung wurde vorher eine konstante Anzahl Packmuster hinterlegt. Gemäß der Netzreihenfolge werden schließlich alle Packstücke auf den Paletten des Drehtisches plaziert.

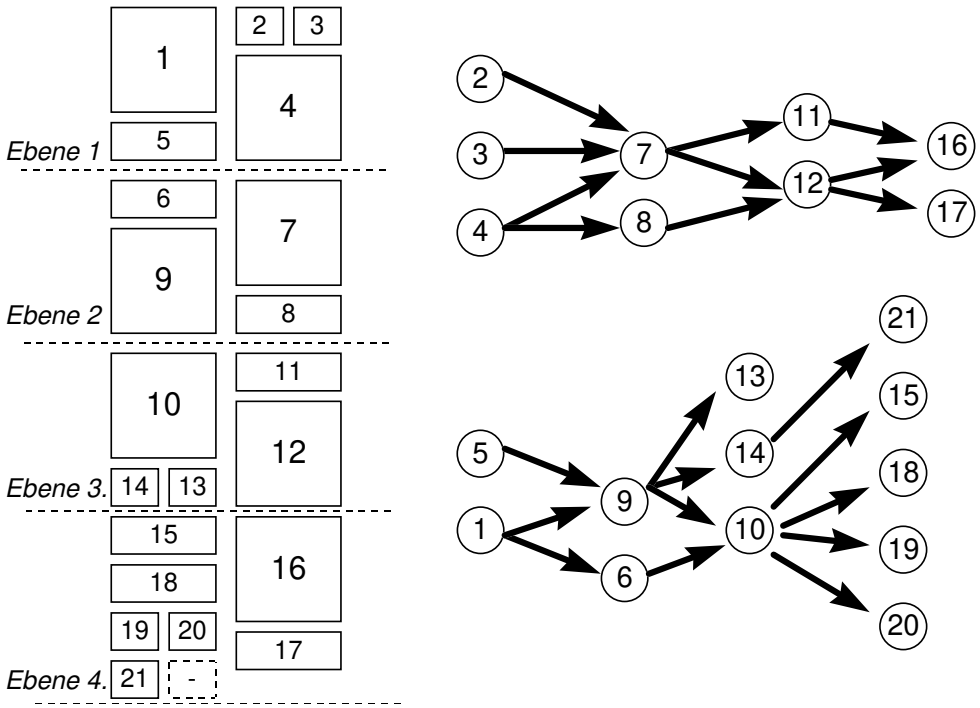


Abbildung 2.4: Bestimmung der Palettierreihenfolge mit Netzwerken nach Tsai

Menzel und Walter benutzen in [27] und [16] die Stauraumoptimierungssoftware PACKI , die Geometriedaten des Packmusters in eine ASCII-Datei exportiert. Die Datei wird dann in die Robotersteuerung (Bosch Rho3) geladen und anschließend on-line vom Roboterprogramm ausgewertet.

Busch setzt in [99] den Packmustergenerator Puzzle [103] ein, um zwei KUKA Roboter mit Packmusterdaten zu versorgen.

In [95] wird eine SCARA Palettieranlage mit Mehrfachgreifer präsentiert. Unter Mehrfachgreifer versteht der Autor nicht das gleichzeitige Greifen mehrerer Packstücke, sondern die Fähigkeit unterschiedliche Packstücke ohne Umrüstung palettieren zu können.

In keinem der erwähnten Fälle gibt es einen allgemeinen Ansatz zum Palettieren eines gegebenen Musters mit beliebigen n*m Mehrfachgreifer. Schwachpunkte bisheriger Systeme sind mangelnde steuerungsunabhängige Konzepte, Kollisionsvermeidung mit Anlagenteilen sowie einheitliche Schnittstellen zur SPS. Bei komplexeren Palettieranlagen müssen auch Daten für Zusatzkomponenten wie z. B. Kippeinrichtungen und Vorgruppiertische off-line erzeugt werden. Lediglich Tsai zeigt hier Lösungsansätze zur kollisionsfreien Palettierung eines dreidimensionalen Musters mit Einfachgreifer auf, wobei er allerdings keine Einfügerichtung berücksichtigt. Ferner ist die Ansteuerung von Mehrfachgreifer - insbesondere bei Vereinzelungen - erheblich schwieriger.

2.4 Steuerungskonzepte für Palettieranlagen

2.4.1 Herkömmliche Steuerungskonzepte

Bislang werden Palettieranlagen wie fast alle Fertigungsanlagen hauptsächlich von drei (Hardware-) Komponenten gesteuert ([16], [27]):

- SPS
- Positioniermodul für Handhabungseinheit (z. B. Roboter)
- Bedieneinheit (z. B. Einzeilendisplay mit Tasten)

Alle diese Geräte sind in einem Schaltschrank montiert und kommunizieren entweder über eine parallele (Bus-) oder serielle Schnittstelle (Abb. 2.5). Die Anlagenperipherie - wie z. B. Zuführbänder, Kippeinrichtungen, Gruppiertische - wird von der SPS gesteuert, während

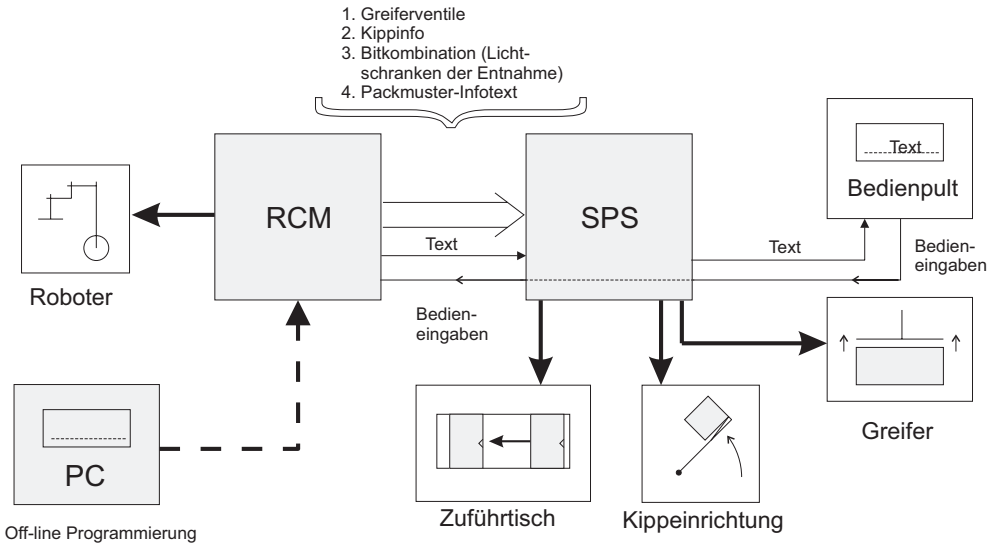


Abbildung 2.5: Herkömmlicher Steuerungsaufbau für Palettieranlagen

das Positioniermodul die Regelung der Verfahrbewegungen übernimmt. Die Kommunikationsbereiche (z. B. RC-SPS) sind dabei oft auf wenige Koppelbyte beschränkt. In der Regel werden die einzelnen Einheiten mit separaten Programmiersystemen programmiert, was jeweils drei unterschiedliche Kabel und Programme erfordert. Bei der Inbetriebnahme durch einen Techniker mit einem PC kann folglich nur eine Steuerungskomponente der Anlage programmiert und geändert werden, was zu erheblichen Verzögerungen führt. Ein softwaretechnisches Umschalten zwischen den Komponenten ist nicht möglich. Verschiedene Packmusterformate werden dabei in der Regel im Speicher der SPS oder der Robotersteuerung abgelegt. Diese Anlagenkonstellation weist bei großen Packmustern, d. h. großen Datenmengen, folgende Schwachstellen auf:

Der Arbeitsspeicher der SPS oder der Robotersteuerung (einige KByte bis wenige MByte) ist für größere Packmuster zu klein. Daher muß ein PC dauerhaft an die Anlage angeschlossen werden, der dann lediglich als externer Datenspeicher dient. Weiterhin ist die Kommunikationsfähigkeit zwischen SPS- und Robotermodule auf wenige Koppelbyte beschränkt, so daß größere Daten nur sehr langsam übertragen werden können. Eine Netzwerkanbindung an einen übergeordneten Leitstandrechner ist nur mit zusätzlicher Spezialhardware möglich. Weitere Probleme ergeben sich, wenn die Robotersteuerung keine ASCII -Daten einlesen kann. Dann müssen alle Daten vom off-line Programmiersystem in ein steuerungsabhängiges Roboterprogramm gewandelt werden.

In den meisten Fällen wird auch gerade im Forschungsbereich ein zusätzlicher PC als Zellenrechner eingesetzt, der zusätzliche Daten für die Robotersteuerung aufbereitet. Keines der Systeme ist für neue Kinematiken offen, d. h. es können nur die in der Robotersteuerung vor-

handenen Achskoordinaten-Transformationen genutzt werden. Ferner ist den meisten Fällen getrennte Hardware zur Ansteuerung der Peripherie notwendig - mit allen daraus resultierenden Kommunikationsproblemen. Es existiert kein Konzept, das alle Komponenten auf einer Plattform vereint oder zumindest dezentral über genormte Softwareschnittstellen (z. B. TCP/IP) anspricht. Die Kommunikation der Einzelkomponenten erfolgt hier auf Bitebene.

2.4.2 Echtzeit und SPS mit Standard-PC

Da moderne PC-Prozessoren in der Regel sehr leistungsfähig und kostengünstig sind, bietet es sich an, dieses Potential auch für die Automatisierungstechnik zu nutzen ([1], [6], [7], [8], [15], [82]). Das Hauptgegenargument war bisher die undeterminierte Antwortzeit des PC (keine Echtzeitfähigkeit). Es muß garantiert werden, daß Ein/Ausgänge innerhalb eines konstanten Zeitraumes ausgelesen, verknüpft und geschrieben werden.

Grundprinzip der Echtzeiterweiterung ist die Unterbrechung des Betriebssystemes durch einen Interrupt am Prozessor (Abb. 2.6). Die Abarbeitung der Betriebssystemaufgaben wird sofort unterbrochen und die Echtzeitaufgaben abgearbeitet. Innerhalb der Echtzeit kann entweder ein eigenständiges Betriebssystem oder eine priorisierte Taskliste abgearbeitet werden ([10], [11], [13], [14], [2]). Bei Palettiersteuerungen werden zunächst die höher priorisierten Positionierungsaufgaben und dann erst die SPS-Task abgearbeitet (Abb. 2.7). In Abhängigkeit der SPS-Programmgröße werden SPS-Tasks eventuell auf mehrere Echtzeittakte verteilt.

Bei Standard-PCs ist es sowohl mit Hardware- als auch Softwareerweiterungen möglich, Echtzeitverhalten zu garantieren ([9], [12], [3], [4], [5]). Eine Hardwarelösung wäre z. B. eine Timerkarte, die zyklisch einen Hardwareinterrupt auslöst und dann eine Interruptroutine aufruft. Bei der Softwarelösung wird der eingebaute Timer des PC benutzt. Die ursprüngliche Interruptaddresse wird überschrieben, der Timerbaustein umprogrammiert und die ursprüngliche Routine im alten Takt aufgerufen. Das Softwareverfahren funktioniert nur bei preemptiven Betriebssystemen, die durch den Timerinterrupt einen Rechenzeitverteiler (Dispatcher) aufrufen, um den einzelnen Teilnehmern Rechenzeit zuzuweisen.

Mit Hilfe der Echtzeiterweiterung für PCs kann für die Automatisierungstechnik ein Standard genutzt werden, der sowohl enormes know-how bietet (Vernetzung [85], Kommunikation, Programmierwerkzeuge [81]) als auch steigende Leistung bei sinkenden Kosten zur Verfügung stellt. Zur Zeit existiert kein durchgängiges und flexibles Steuerungskonzept für Palettieranlagen, das vollständig inklusive Achsinterpolation und Transformation auf Soft-

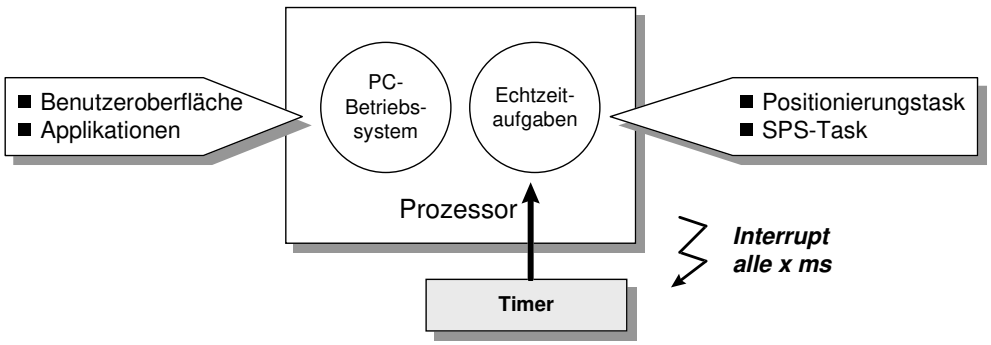


Abbildung 2.6: Echtzeit mit Standard-PC

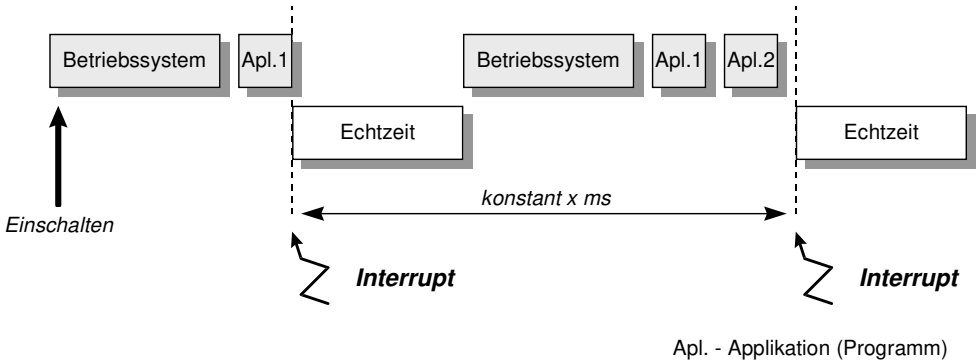


Abbildung 2.7: Zeitverhalten des Standard-PC mit Echtzeiterweiterung

wareebene realisiert ist. In den meisten Fällen gibt es immer noch die klassische Trennung auf Hardwareebene zwischen SPS und Positionierung.

2.5 Positionierungsalgorithmen

Kurze Palettierzeiten erfordern schnelle und mechanikschonende Positionierverfahren. Hier existieren zur Zeit entweder mehrphasige oder Spline basierte Verfahren. Phasenorientierte (PTP -) Verfahren sind in mindestens drei Phasen der Beschleungigung, Konstantfahrt und Bremsen aufgeteilt. Zur weiteren Ruckbegrenzung werden Filter einsetzt [53], [54], [56], [57], [58], [59]. Es ensteht dann ein $\sin^2$ ähnliches Profil mit sieben Phasen ([80]), die alle im Interpolator erkannt und unterschiedlich behandelt werden müssen.

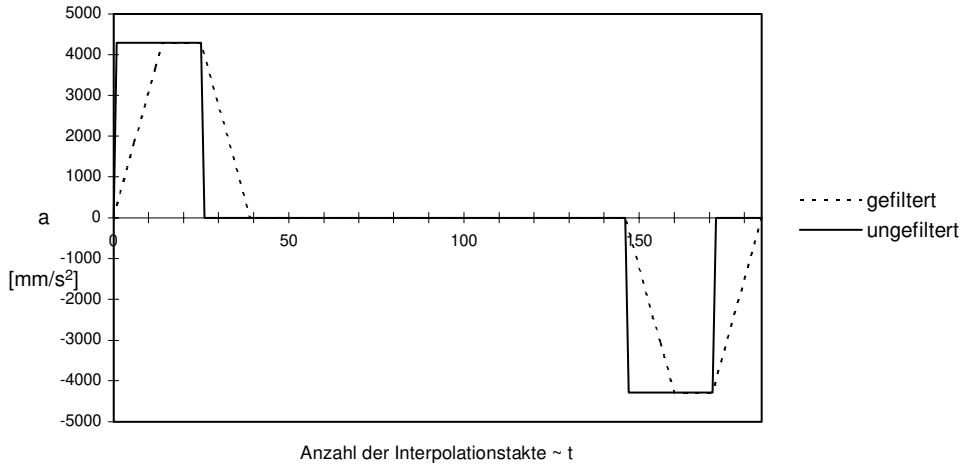


Abbildung 2.8: Beschleunigungsprofil beim PTP Verfahren

2.5.1 Synchrone PTP-Interpolation

Die synchrone PTP-Interpolation ist theoretisch die schnellste Möglichkeit, um Roboterachsen vom Start- zum Endpunkt zu verfahren. Synchron bedeutet, daß alle Achsen an die langsamste Achse angeglichen werden und zur gleichen Zeit ihre Endposition erreichen. Die Bewegung unterteilt sich in ein Drei-Phasen-Profil (Beschleunigen, Konstantfahrt, Bremsen). Es entsteht ein trapezförmiges Geschwindigkeitsprofil und ein rechteckförmiges Beschleunigungsprofil (Abb. 2.8). Der Nachteil dieser Interpolationsart ist, daß an den Phasenübergängen ein unendlich großer Ruck (zeitliche Ableitung der Beschleunigung) entsteht, d. h. daß die Mechanik sehr beansprucht wird. In der Praxis sind daher die vorgegebenen synchronen Geschwindigskeitswerte kaum erreichbar. Der Ruck kann allerdings durch eine anschließende Mittelwertfilterung des Geschwindigkeitsprofils begrenzt werden. Es ergibt sich dann ein trapezförmiges Beschleunigungsprofil und ein ruckbegrenzter Bahnverlauf (Abb. 2.8). Nachteil der Filterung ist der Mehraufwand im Interpolator sowie zusätzliche Probleme bei der Synchronisation.

2.5.2 Spline-Interpolation

Der Begriff Spline bezeichnete ursprünglich eine im Schiffsbau verwendete biegsame Latte. Die Form, die durch Biegung der Latte unter Vorgabe einiger Haltepunkte entsteht, hat die Eigenschaft, die Biegeenergie zu minimieren. Mit der Latte können möglichst "glatte" Kurven durch vorgegebene Haltepunkte gelegt werden, d.h. die Gesamtkrümmung der Kurven ist minimal. Diese Kurven werden durch kubische, krümmungsstetige Polynomsplines

3. Grades nachgebildet. Spline-Interpolationsverfahren werden bisher vorwiegend in Steuerung für Werkzeugmaschinen eingesetzt ([83], [84]), um im Bereich der Freiformflächen die erforderlichen weichen, stetigen Formen zu erzeugen. Die Sollbahn wird in kartesischen Koordinaten interpoliert. Schwinn beschreibt in [54] ein Verfahren zur Bahnplanung auf

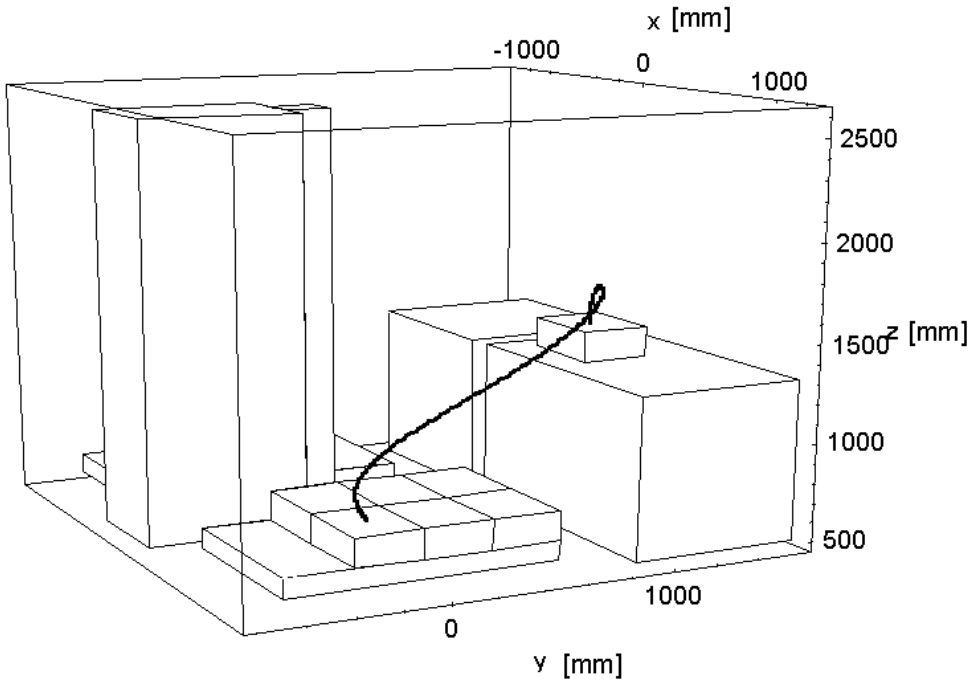


Abbildung 2.9: Bahnverlauf mit Spline Verfahren

Achskoordinatenebene, das auf kubischen Splines [55] basiert. Dabei sind n Stützstellen, Start-, Endpunkt und (n - 2) Zwischenpunkte vorgegeben. Geschwindigkeit und Beschleunigung sind im Start- und Endpunkt Null. Zwischen zwei Stützstellen wird ein Polynom dritten Grades, die kubische Spline-Funktion, berechnet und an den Stützstellen den benachbarten Polynomen mit Randbedingungen so angepaßt, daß die Gesamtbahn einen beschleunigungsstetigen Verlauf hat. Das Spline Verfahren wurde am Beispiel einiger Positionierungen innerhalb einer Palettieranlage mit SCARA-Roboter simuliert. Abb. 2.9 zeigt den Bahnverlauf für eine Positionierung von einer Zuführung zu einer Abgabe mit zugehörigem Geschwindigkeitsprofil Abb. 2.10. Lediglich eine Achse erreicht einmal während des Bewegungsablaufes ihre Geschwindigkeits- oder Beschleunigungsgrenze. In diesem Beispiel beschleunigt nur die lineare Achse (Achse 1) maximal. Dies liegt zum einen an der geometrischen Anordnung des verwendeten SCARA-Roboters, zum anderen im Grundprinzip des Spline Berechnungsverfahren begründet. Beim SCARA-Roboter wird die lineare Achse wesentlich stärker beansprucht als die Drehachsen, weil sie größere Wege als die Drehachsen Winkel zurücklegen muß. Aufgrund der notwendigen Synchronisation aller Achsen werden die Drehachsen deshalb weniger beansprucht.

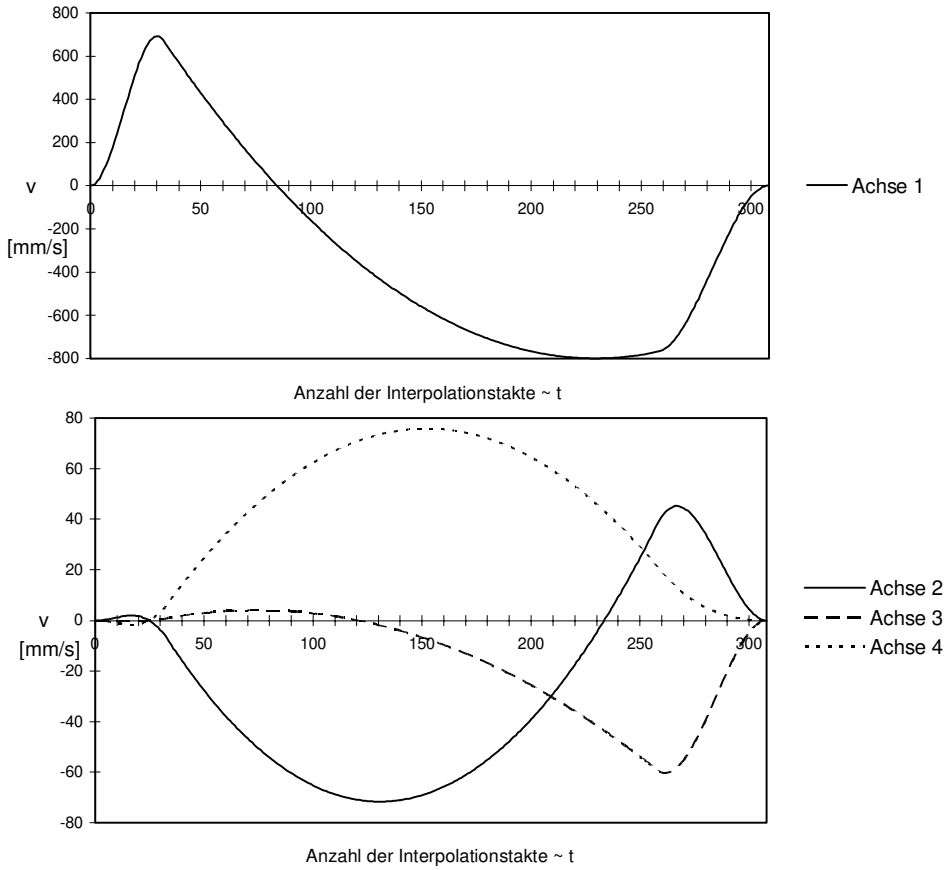


Abbildung 2.10: Geschwindigkeitsprofile beim Positionieren mit Spline Verfahren

Hauptproblem des Spline Verfahrens ist das ungewünschte Überschwingverhalten bei ungeschickter Wahl der Zwischenpunkte. Durch Einfügen eines zusätzlichen Zwischenpunktes beim vorherigen Palettierbeispiel ergibt sich der in Abb. 2.11 dargestellte Bahnverlauf. Die erste und zweite Drehachse kollidieren mit der Robotersäule . Sowohl die PTP als auch auch die reine Spline Interpolation auf Bahnebene sind zur Positionierung in der Palettiertechnik ungeeignet, da sie entweder zu langsam oder nicht ruckfrei sind. Tabelle 2.1 zeigt die Vor- und Nachteile im einzelnen: Es existiert kein Verfahren, das schnelle ruckfreie Bewegungen

Verfahren	Vorteil	Nachteil
PTP	Geschwindigkeit	nicht ruckfrei
PTP gefiltert	Geschwindigkeit	Mehrere Phasen, Synchronisation
Spline	ruckfrei	Überschwingen, Geschwindigkeit

Tabelle 2.1: Positionierungsverfahren: Vor- und Nachteile

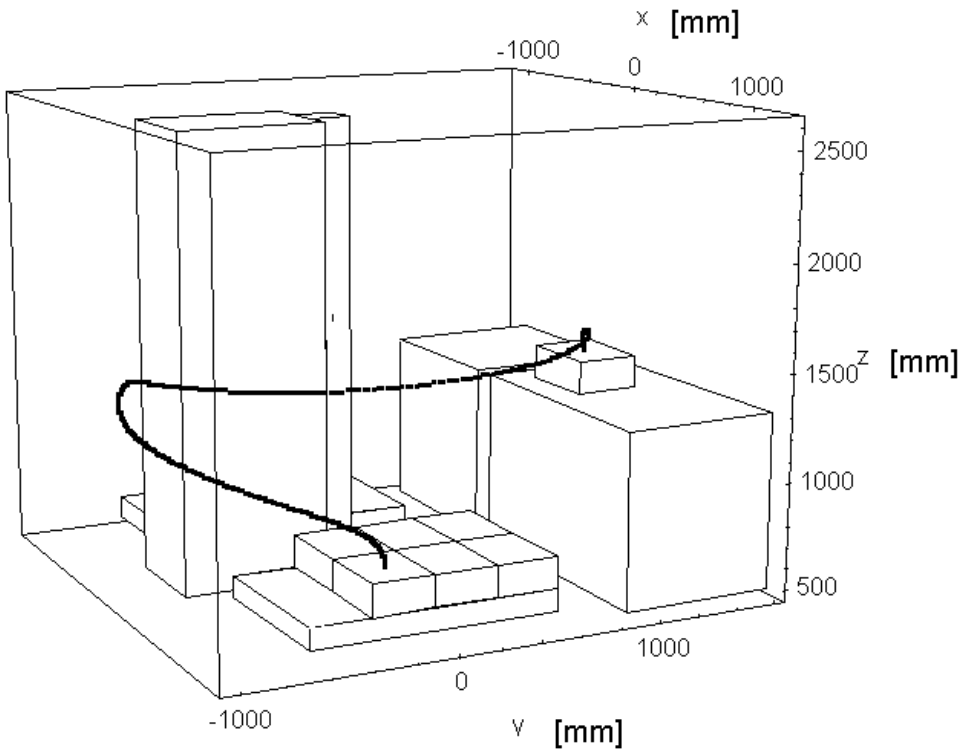


Abbildung 2.11: Bahnverlauf bei ungünstiger Stützstellenwahl

aber auch einfache Datenhaltung erlaubt.

Kapitel 3

Stand der Technik

3.1 Software zur Erzeugung von Packmustern

Tabelle 3.1 zeigt eine Übersicht der am Markt angebotenen Optimierungssoftware für Packmuster. Die meisten Produkte arbeiten mit Blockheuristiken, d. h. einer konstanten Zahl an Packmustern. Für alle Packmuster werden die Zahl der resultierenden Packstücke aufgrund der gewünschten Packstückgröße berechnet und nach maximaler Packstückzahl optimiert. Reine Suchverfahren sind aufgrund der erhöhten Rechenzeit selten zu finden. Die Anbindung an ein Handhabungssystem erfolgt dann in Form einer Datei mit den Ablagekoordinaten (für Einfachgreifer). Multipack kann als einziges Produkt eine weitere Optimierung der Ablagepunkte für Mehrfachgreifer durchführen. Einfügerichtungen und Kollisionen werden dabei nicht berücksichtigt.

Anbieter	Produkt	Packmustererzeugung	Koordinatenausgabe
Multiscience [64]	Multipack	lagenweise 2D	möglich
Multiscience [64]	Multipack plus	3D	möglich
Multiscience [64]	Multipack complex	3D, nicht quaderförmig	möglich
IML [65]	Puzzle	3D	möglich
Cape	Cape	2D	möglich
FHG	Ladeprog	2D	keine Angabe
Prolog	Stapel	2D	keine Angabe
MRM [68]		2D lagenweise	möglich (Einfachgreifer)
Fanuc [94]	PalletTool	2D Lagenweise	ja, Zeilengreifer

Tabelle 3.1: Stauraumoptimierungssoftware

3.2 Off-line Programmierung für Palettieranlagen

Die Programmierung der Palettieranlagen erfolgt häufig im Teachverfahren, d. h. der Techniker verfährt den Roboter oder das Portal per Hand an die gewünschte Position und speichert diese. Das Verfahren ist zeitaufwendig und teuer. Die Anlage steht während des Teachens nicht für den Produktionsprozeß zur Verfügung. Weiterhin kann das Teachen nur durch Fachpersonal erfolgen. Diese Nachteile können durch Einsatz von off-line Programmierung vermieden werden.

Am Markt existieren einige off-line Programmiersysteme (Tabelle 3.2) die aus einem gegebenen Packmuster die Anfahrpunkte des Roboters/Portals berechnen. Der Bediener generiert hier off-line am PC ein Packmuster und numeriert die einzelnen Pakete, so daß eine kollisionsfreie Palettierreihenfolge entsteht. Die Ansteuerung der Sauger muß hier manuell angegeben werden. Dieses Verfahren des "virtuellen Teachens" hat zwar erhebliche Vorteile gegenüber des reinen Teachens der Packmuster- eine Fehlbedienung kann allerdings zu katastrophalen Kollisionen führen. Off-line Programmiersysteme werden meist mit Zusatzprogrammen für die Palettierung vom Roboterhersteller angeboten.

Anbieter	Produkt	Packmuster-erzeugung	Palettier-programm	Greifer
Adept [69]	PalletWare	lagenweise manuell	manuell	mehrfach
Fanuc [96]	Pallettool	lagenweise manuell, automatisch	halbautomatisch, manuell	mehrfach (Reihe)
Gomes		lagenweise manuell, automatisch	manuell	Doppel
Ro-ber [66]	Puzzle	3D	automatisch	Einfach
Elettric 80 [67]	Freeway	lagenweise manuell		
ABB [98]	PalletWizard	2D lagenweise	keine Angabe	keine Angabe

Tabelle 3.2: Off-line Programmiersoftware für die Palettierung

3.3 Steuerungen für Palettieranlagen

Momentan sind im Steuerungsbereich für Palettieranlagen überwiegend noch klassische Strukturen zu finden - SPS, Robotersteuerung und Bedieneinheiten als entkoppelte Hardwareeinheiten. Die Marktübersicht der Tabelle 3.3 verdeutlicht den zwar zur Zeit noch relativ schwachen, aber durchaus vorhandenen Trend zu PC-basierten Steuerungslösungen.

Hauptprobleme klassischer Lösungen sind mangelnde Flexibilität (z. B. spezielle Transformationen) sowie Abhängigkeit von wenigen Herstellern und die damit verbundenen hohen Kosten. Ein Anlagenhersteller, der eine Sonderkinematik entwickelt hat, wird nur selten den Steuerungshersteller wechseln, wenn der Anpassungsaufwand zu groß ist. Hier fehlen Steuerungskonzepte, die auf einheitlichen Hardware- und Softwarestandards basieren.

Anbieter	Robotersteuerung	SPS	Bedienpult	Feldbus-anschluß
Adept [69]	VME Basis	-	-	-
Fanuc [100]	eigenes Betriebsystem	-	-	möglich
Gomes	PC - basiert	integriert	auf dem PC	-
Bosch Rho3	eigenes Betriebssystem	integriert	-	CAN
Siemens RCM 1P [86]	eigenes Betriebssystem	extern	-	-
IBM	PC - basiert, Sprache AML/2	-	auf dem PC	-
KUKA [10]	PC - basiert	als Softwaretask	-	CAN

Tabelle 3.3: Robotersteuerungen für die Palettierung

Kapitel 4

Aufgabenstellung

Die Analyse des Standes der Technik und der Wissenschaft zeigt eine deutliche Lücke zwischen der Generierung eines optimierten Packmusters und der automatischen, industriellen Palettierung eines Packmusters mit Handhabungssystemen im Produktionsprozeß. Die automatisch erzeugten Packmuster werden sehr oft manuell bzw. halbmanuell programmiert. In wenigen Fällen wurde die Palettierung eines gegebenen Packmusters ansatzweise mit Handhabungssystemen durchgängig automatisiert. Dabei wurden häufig halbautomatische Lösungen wie z. B. das "virtuelle Palettieren" aufgezeigt. Vereinfachte Sonderfälle wie z. B. die Palettierung mit Einfachsauggreifern berücksichtigen weder Kollisionen mit Anlagenteilen noch spezielle Greifertypen. Es existiert keine generelle strukturierte Modellierung von Palettieranlagen und somit keine allgemeine Lösung des Palettierproblems bei bekannter Kommissionierung für unterschiedliche Greifertypen.

Schwerpunkt dieser Arbeit ist die Lösung der Frage, wie ein gegebenes dreidimensionales Packmuster mit verschiedenen Packstücken (auftragsbezogene Kommissionierung) in minimaler Zeit palettiert werden kann. Mit Hilfe einer Modellbildung ist eine Lösung des Palettierproblems zu entwickeln, um eine automatisierte, flexible Palettierung in der Produktion mit beliebigen Handhabungssystemen zu realisieren. Dabei sollen die folgenden Randbedingungen gegeben sein:

- Eine Palettieranlage bestehend aus einem Handhabungsgerät (z. B. Roboter) mit unterschiedlichen Greiferbauformen, beliebig vielen Abhol-, Abgabestationen, sowie zusätzlichen Kollisionsobjekten (z. B. Schutzzaun).
- Eine Anordnung von Packstücken, die von einer übergeordneten Instanz (z. B. Optimierungsprogamm) erzeugt wurde und quaderförmige Packstücke von beliebiger Zahl und Abmessung enthält. Die Packstücke müssen nicht zwingend lagenweise angeordnet

sein. Ferner sind nur orthogonale Ausrichtungen der Packstücke zulässig.

Gesucht ist eine optimierte Reihenfolge von Anfahrpunkten und Schaltinformationen, die keine Kollisionen mit Objekten der Anlage (Handhabungsgerät, Packstück etc.) hervorruft. Um dies zu erreichen, muß zunächst ein Verfahren (Algorithmus) entwickelt werden, das anhand eines Modells die möglichen Palettierreihenfolgen ermittelt, bewertet und schließlich einen optimierten, kollisionsfreien Palettierablauf ausgibt. Dabei sind unterschiedliche Anlagenbauformen und Einfügerichtungen zu beachten. Die Einfügerichtung ist die Richtung, aus der ein neues abzulegendes Paket an bereits vorhandene angelegt wird.

Ferner soll ein Steuerungskonzept entwickelt werden, das die optimale Ansteuerung der automatisch generierten Palettierreihenfolge sicherstellt. Die Anforderungen lauten im einzelnen:

- automatische Erstellung von Packprogrammen für Mehrfachgreifer
- eine durchgängige Systemlösung von der Packmustererzeugung bis zur Positionierung
- schnelle Übertragung zur Steuerung
- einfache Bedienung, Inbetriebnahme und Programmierung des Gesamtsystems
- Berücksichtigung verschiedener Kinematiken und Anlagenvarianten
- Unabhängigkeit von Robotertyp und -steuerung
- Offenheit für manuelle Erweiterungen (z. B. Messung der Palettenhöhe)
- hohe Palettiergeschwindigkeit, mechanikschonende Positionierung.

Das resultierende Gesamtsystem sollte möglichst die in Abb. 4.1 dargestellte Struktur aufweisen.

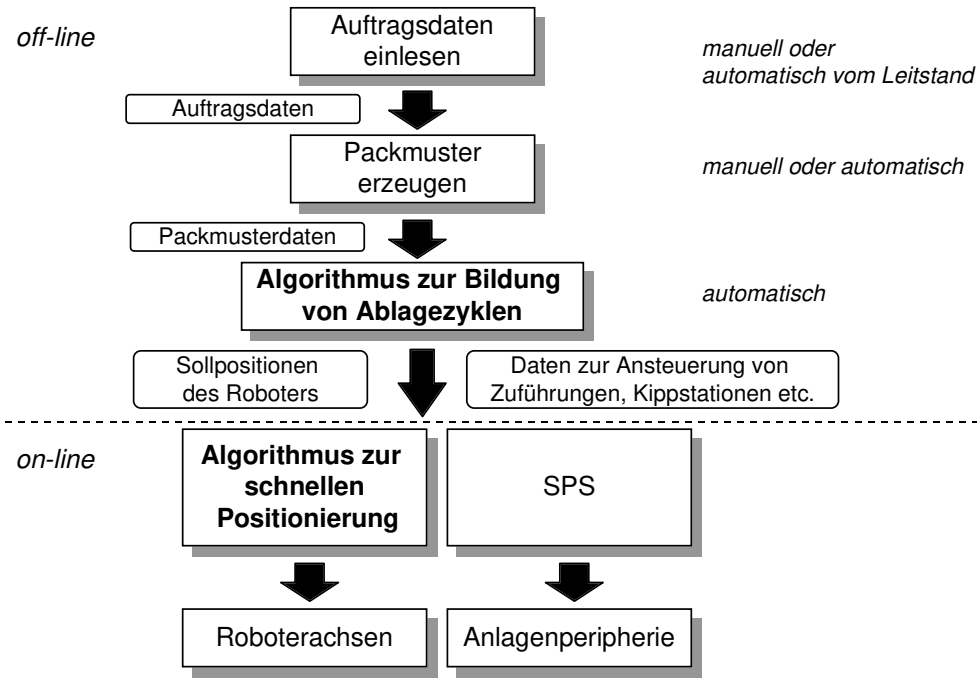


Abbildung 4.1: Sollstruktur eines Gesamtsystems zur Palettierung

Kapitel 5

Off-line Programmierung

Laut Aufgabenstellung soll zunächst ein Verfahren zur Optimierung des Palettierablaufes anhand einer Modellbildung für Palettieranlage und Packmuster entwickelt werden. Daraus resultiernd wird eine Referenzarchitektur für ein Palettiersystem entwickelt, das automatisch Packprogrammabläufe erzeugt (und abarbeitet), aber dennoch offen genug für manuelle Erweiterungen ist.

5.1 Beschreibung des allgemeinen Palettierproblems

Ein gegebenes Packmuster (d. h. beliebig viele Packstücke unterschiedlicher Größe) soll unter Berücksichtigung der Anlagenpherie und des Greifers so aufgeteilt werden, daß möglichst wenig Anfahrpunkte entstehen (Abb. 5.1). Die Anfahrpositionen müssen so berechnet werden, daß weder Greifer noch Handhabungsgerät mit der Anlagenperipherie oder den Packstücken kollidieren. Der eigentliche Palettiervorgang (on-line) wird in Zyklen aufgeteilt, die vereinfacht aus den folgenden Schritten bestehen:

1. Greife n Packstücke mit Greifer G_i der Bauform T_j von der Abholstation Abh_k.
2. Positioniere m Packstücke auf Abgabe Abg_l .
3. Wiederhole den letzten Schritt, bis alle Packstücke des Packmusters abgelegt sind.

mit m $<$ n $<$ Anzahl der Packstücke, 0 $<$ i $<$ Anzahl der Greifer, 0 $<$ j $<$ Anzahl der Greifertypen, 0 $<$ k $<$ Anzahl der Abholstationen und 0 $<$ l $<$ Anzahl der Abgabestationen.

In obiger Aufzählung werden keine Zwischenpunkte berücksichtigt, da diese zur Erstellung eines Palettierablaufes zunächst nicht erforderlich sind. Eine genaue Bahnplanung mit den

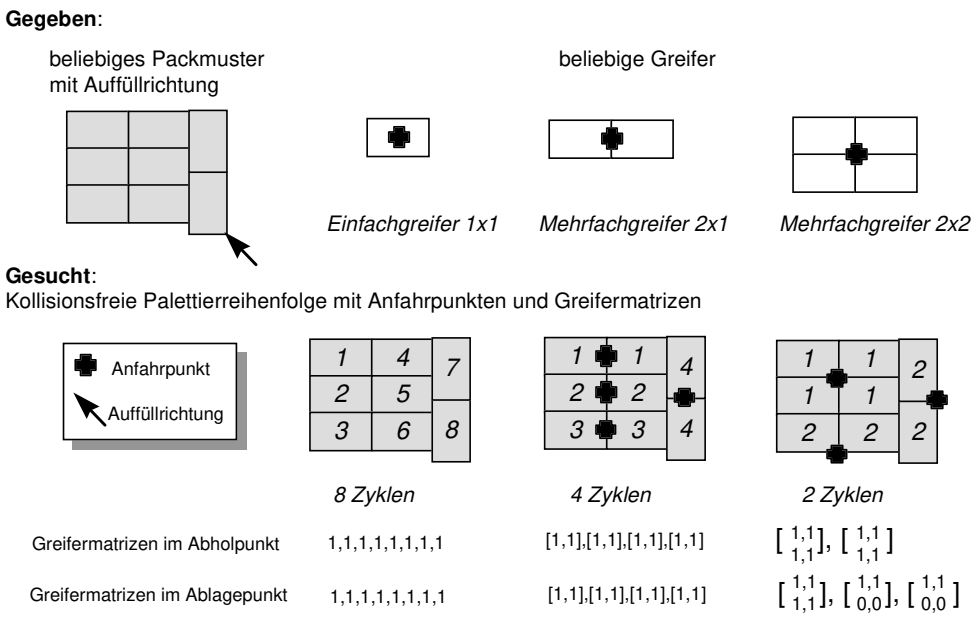


Abbildung 5.1: Das allgemeine Palettierproblem

erforderlichen Zwischenpunkten ist in Kapitel 7.6 beschrieben. Die Packstücke stehen in Form einer zweidimensionalen Matrix an der Abholstation zur Verfügung. Letztlich werden für jeden Zyklus die Abhol- und Ablagepositionen, aber auch die Matrizen zur Ansteuerung der Greifer und Abholstationen benötigt.

5.2 Palettieranlagenmodell

Eine Palettieranlage besteht allgemein aus einem Handhabungsgerät und einer beliebigen Anzahl von Abhol- und Abgabestationen sowie Greifern. Abb. 5.2 zeigt das vereinfachte Modell, das nachfolgend anhand eines Palettierzyklus erläutert wird:

Zunächst wird für jeden Palettierzyklus eine zweidimensionale Matrix von Packstücken an der Abholstation bereitgestellt. Die Packstücke erzwingen automatisch eine Matrix von Sensor bzw. Schalterwerten (1,0). Diese beiden Matrizen können, müssen aber nicht zwingend identisch sein (siehe virtuelle Greifer). Im nächsten Schritt werden die Packstücke durch Ansteuerung der Greiferaktoren (Ventile für Sauger oder Zangen) gegriffen. Auch hier kann die Packstückmatrix ungleich der Aktormatrix sein. Zum Schluß erfolgt das Ablegen der Packstücke auf der Palette durch Positionierung des Handhabungssystems und Ausgabe der Greiferaktorenmatrix.

Die in der Einleitung vorgestellten Greiferbauformen können durch eine zweidimensionale

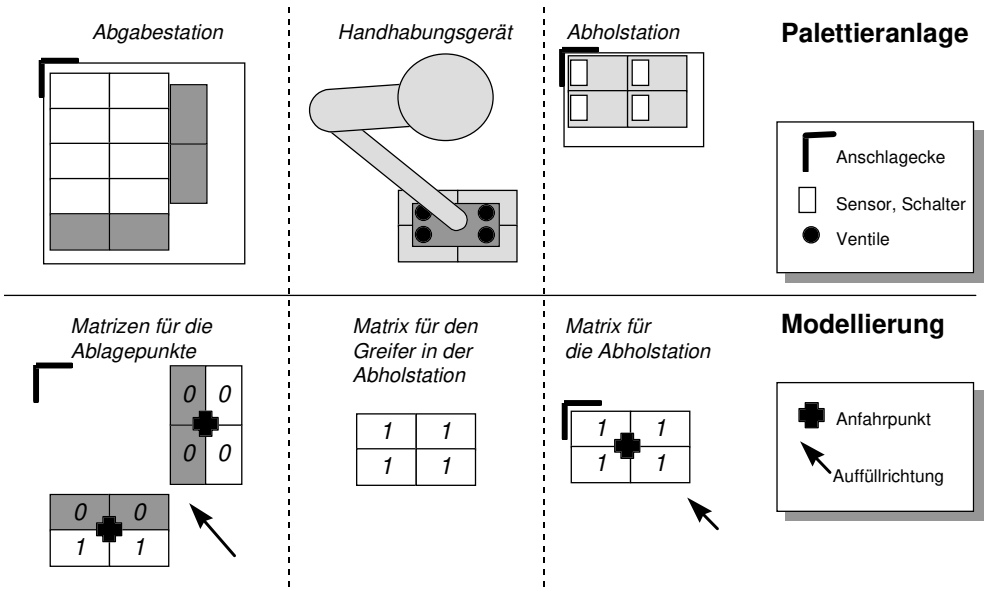


Abbildung 5.2: Modellierung von Palettieranlagen

Aktormatrix, Rechtecke (Grundplatten) und der Werkzeugkorrektur vollständig beschrieben werden. Tabelle 5.1 zeigt eine Zusammenfassung der relevanten Komponenten einer Palettieranlage und deren mathematische Darstellung.

Komponente	mathematische Darstellung
Abholstation	2D Matrix für Sensorwerte
Greifer	2D Matrix für die Aktoren, Rechtecke, Werkzeugkorrekturvektor
Ablage (Palette)	kartesische Koordinaten, 2D Matrix für die Aktoren
Roboterarme	Gerade, Polygon
Begrenzungsraum	Polygon, Rechteck
weitere Kollisionsobjekte	Rechtecke, Kreise

Tabelle 5.1: Mathematische Darstellung von Komponenten einer Palettieranlage

5.2.1 Virtuelle Greifer

Die Packstückmatrizen und die Matrizen zur Ansteuerung der Greiferaktoren können durchaus unterschiedlich sein, weil diese sich in Abhängigkeit der Packstückgröße ändern. Somit entstehen pro Packstückmaß sogenannte virtuelle Greifer aus deren ganzzahligen Vielfachen

die Palettierreihenfolge gebildet wird. Zur Ansteuerung der vorhandenen Aktoren muß die virtuelle Greifermatrix noch in eine physikalische Matrix transformiert werden.

Abb. 5.3 zeigt das Beispiel zweier virtueller Greifer für eine Zuführung. Aufgrund der Packstückmaße wurde der erste virtuelle Greifer als 1x2 Matrix und der zweite als 2x2 Matrix gewählt. Das Packmuster wird entsprechend in Vielfache der 1x2 und 2x2 Matrizen aufgeteilt. Die Ansteuerung der Aktoren (Ventile/Sauger) erfolgt letztlich durch die physikalisch gegebenen Matrizen (hier 2x2 Matrix).

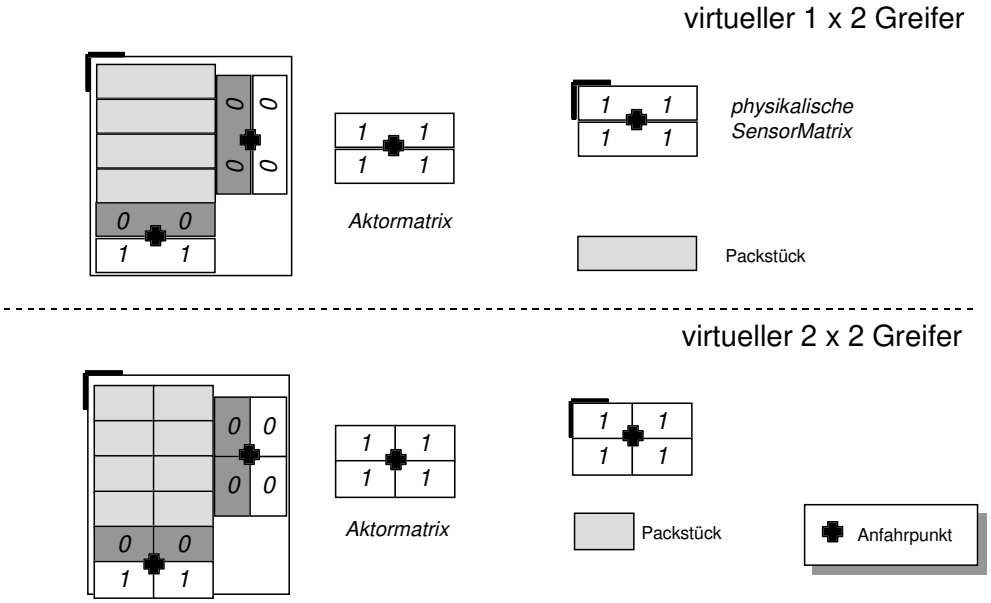


Abbildung 5.3: Virtuelle Greifer

Aus der Anlagenmodellierung ergeben sich die konkreten Aufgaben für einen Palettieralgorithmus, der im weiteren auch Anordnungsalgorithmus genannt wird:

- Ordne die Packstücke unter Berücksichtigung der Auffüllrichtungen (Abgabe und Abholstation!) in Greiferzyklen an, so daß möglichst viele Packstücke zugleich abgelegt werden können und somit möglichst wenig Anfahrpunkte entstehen.
- Überwache auf Kollisionen mit Anlagenteilen und bereits abgelegten Packstücken.
- Erzeuge Ablagepunkte mit Angabe der Greiferdrehung sowie Steuerinformationen für Zuführung und Greifer. Bei Vereinzelungen sind die Ablagepunkte nicht immer automatisch die Mittelpunkte und müssen anhand der relativen Matrixposition der abzulegenden Packstücke korrigiert werden.

Das Hauptproblem sind die vielen Möglichkeiten zur Anordnung von Packstücken bei der Bildung der Greiferzyklen. Hier muß eine Abbildung des Packmusters in eine mathematische

Struktur oder in ein Modell gefunden werden, um die Möglichkeiten zu reduzieren. Ein mathematischer Graph ist eine solche geeignete Struktur.

5.3 Graphmodell für Packmuster

Bisher werden Packprobleme in Form von Netzen dargestellt (siehe S.5), was bei Packmustern schon in geringer Suchtiefe zu unübersichtlichen Darstellungen und schwer nachvollziehbaren Lösungen führt. Aus diesem Grunde wurde die nachfolgend beschriebene dreidimensionale Darstellungsweise entwickelt. Die Darstellung eines Packmusters als Graph ist sowohl dafür geeignet, das Optimierungsproblem mit Hilfe von Suchverfahren zu lösen, als auch das Problem und die Lösungen zu visualisieren.

5.3.1 Aufbau eines vollständigen Graphen

Jeder Knoten des Graphen stellt ein Packstück bzw. einen Block (-> Blockalgorithmus) dar. Unter Berücksichtigung der Einfügerichtung bedeutet eine eingerichtete Kante von P[i] nach P[i+1], daß P[i] vor P[i+1] abgelegt werden muß (Abb. 5.4). Somit ist P[i] Vorgänger von P[i+1]. Um alle Vorgänger eines Graphen bestimmen zu können, wird das Packmuster sortiert, die untere Einfügeecke und die obere Einfügeecke berechnet. Die Einfügeecken sind diejenigen Ecken, die zuerst mit Packstücken belegt werden. Daher gilt in Abhängigkeit der Einfügerichtung:

Packstück $P[i]$ ist Vorgänger von Packstück $P[i+1]$, wenn

$$P[i]_{\text{obere Einfügeecke}} > P[i+1]_{\text{untere Einfügeecke}} \tag{5.1}$$

Der (überladene) $>$ Operator liefert true, wenn

$$(P[i].X_{unten} \leq P[i+1].Y_{oben}) \wedge (P[i].Y_{unten} \leq P[i+1].Y_{oben}). \tag{5.2}$$

Somit kann der Algorithmus zur Bestimmung aller Vorgänger (in C++) wie folgt angegeben werden:

```
// Durchlaufe das gesamte Feld mit Ausnahme des aktuellen Index
for(int i=0;i< MaxAnzahlPackstücke;i++)
{
 // Überspringe aktuellen Index
 if(i==AktuellerPackstückIndex)
```

```
  i++;
 if(i >= MaxAnzahlPackstücke)
  break;
 // Ist Packstück mit Index i Vorgänger von AktuellerPackstückIndex ?
 if(P[AktuellerPackstückIndex].ObereEinfügeecke > P[i].UntereEinfügeecke)
 {
  // Index einfügen
  P[AktuellerPackstückIndex].VorgängerPackstückListe.Insert(i);
 }
}
```

Alle Packstücke, die im oberen Einfügerechteck eines Packstücks liegen oder dieses schneiden, sind Vorgänger des Packstücks. Für jeden Knoten werden alle Vorgänger in einer Liste gespeichert, z. B. in Abb. 5.4 sind die Packstücke 1,2,4 Vorgänger von Packstück 5.

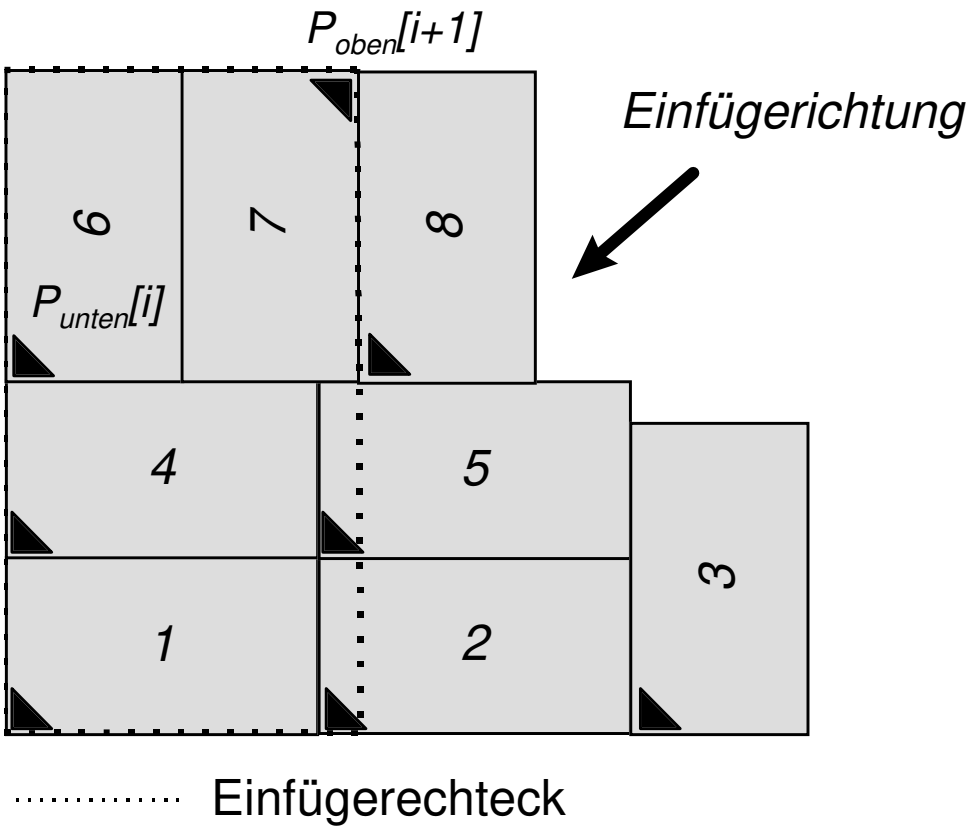


Abbildung 5.4: Erstellung einer Vorgängerliste mit Einfügeecken

Zum Aufbau eines Graphen werden entweder die Einfügeecken oder die Mittelpunkte der Packstücke als Knotenmittelpunkte des Graphen benutzt. Die Verwendung der Einfügeecken

hat den zusätzlichen Vorteil, daß zusammenliegende Packstücke direkt über den Differenzvektor der Eckpunkte und einen Vergleich der Packstückmaße erkannt werden können.

5.3.2 Aufbau eines reduzierten (Vorgänger-) Graphen

Da die Darstellung des vollständigen Graphen auch für kleine Packmuster bereits zu unübersichtlich wird, ist eine komprimierte Darstellung ohne Verlust des Informationsgehaltes entwickelt worden. Diese Form der Darstellung wird reduzierter Graph genannt und basiert auf der Tatsache, daß nur direkte Vorgänger vorhanden sein müssen. In Abb. 5.4 ist es z. B. ausreichend, daß Block 5 die direkten Vorgänger 4 und 2 hat - Packstück 1 kann aus der Vorgängerliste eliminiert werden. Der Graph wird somit erheblich reduziert. Der Algorithmus zur Erkennung der direkten Vorgänger kann wie folgt dargestellt werden:

```
// Alle direkten Vorgänger ermitteln und indirekte zum Loeschen markieren
for( i=0;i<MaxAnzahlPackstücke;i++)
{
  for(int j=0;j<P[i].AnzahlVorgängerPackstücke;j++)
  {
 if(P[i].VorgängerPackstückListe[j]==-1)
j++;
 int AktuellerVorgängerIndex=P[i].VorgängerPackstückListe[j] ;
 for(int k=0;k<P[i].AnzahlVorgängerPackstücke;k++)
 {
 if(k==j)
k++;
 // wenn Vorgänger geloescht werden sollen ->
 //keine Untersuchung seiner Vorgänger
 if(P[i].VorgängerPackstückListe[k].KannGelöschtWerden)
k++;
 // hier alle indirekten Vorgänger zum Loeschen eintragen
 for(int l=0;l< P[AktuellerVorgängerIndex].AnzahlVorgängerPackstücke;l++)
if( P[AktuellerVorgängerIndex].VorgängerPackstückListe[l] ==
P[i].VorgängerPackstückListe[k])
P[i].VorgängerPackstückListe[k].KannGelöschtWerden =true;
 }
  }
}
```

Knoten	Vorgänger		Nachfolger	
P[i]	vollständig	reduziert	vollständig	reduziert
1	-	-	2 3 4 5 6 7 8	2 4
2	1	1	3 5 7 8	5
3	1 2 4 5	5	-	-
4	1	1	3 5 6 7 8	5 6
5	1 2 4	2 4	3 7 8	7 8 3
6	4 1	4	7 8	7
7	1 2 4 5 6	5 6	8	8
8	1 2 4 5 6 7	5 7	-	-

Tabelle 5.2: Vorgänger- und Nachfolgerlisten

Die Listen der vollständigen und reduzierten Vorgänger und Nachfolger für das in Abb. 5.4 abgebildete Packmuster sind in der Tabelle 5.2 dargestellt. Am Beispiel eines weiteren zweidimensionalen Packmusters wird in Abb. 5.5 ein reduzierter Nachfolger-Graph unter Berücksichtigung einer Einfügerichtung gezeigt.

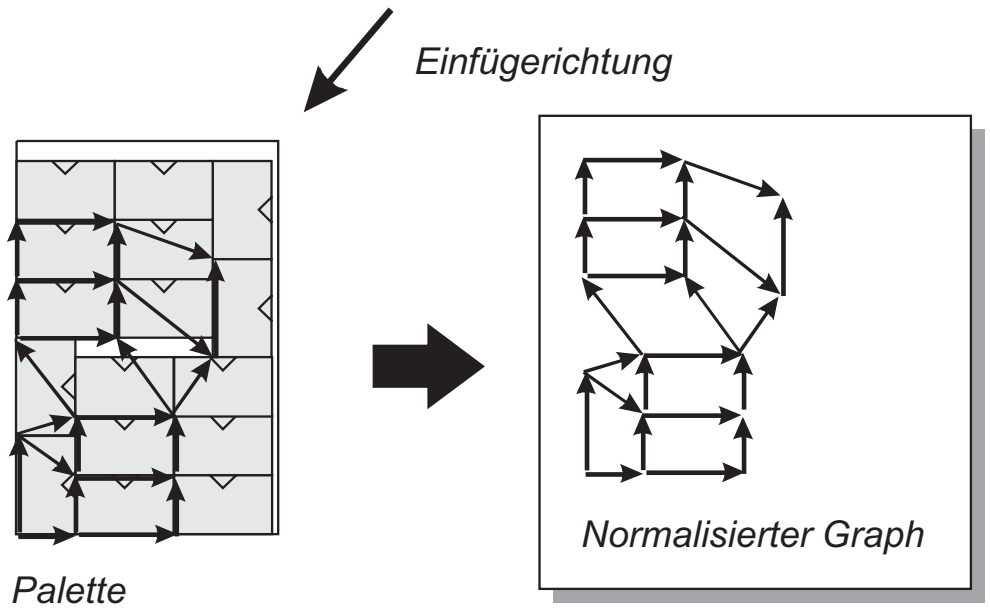


Abbildung 5.5: Reduzierter zweidimensionaler Packstückgraph

Das Graphmodell läßt sich auch auf dreidimensionale Muster anwenden, indem in Abb. 5.4 auch die z-Koordinate ausgewertet wird. Abb. 5.6 zeigt einen reduzierten dreidimensionalen Graphen für das Packmuster "z1-orang" mit 320 Packstücken. Eine dreidimensionale Darstellung des Packmusters ist im Anhang A.1 in Abb. A.4 zu finden. Die Packstücke wurden

hier bereits zu Greifer-Blöcken (siehe 5.4.3) zusammengefaßt, wobei die Pfeile in Richtung des jeweiligen Vorgängerblockes zeigen.

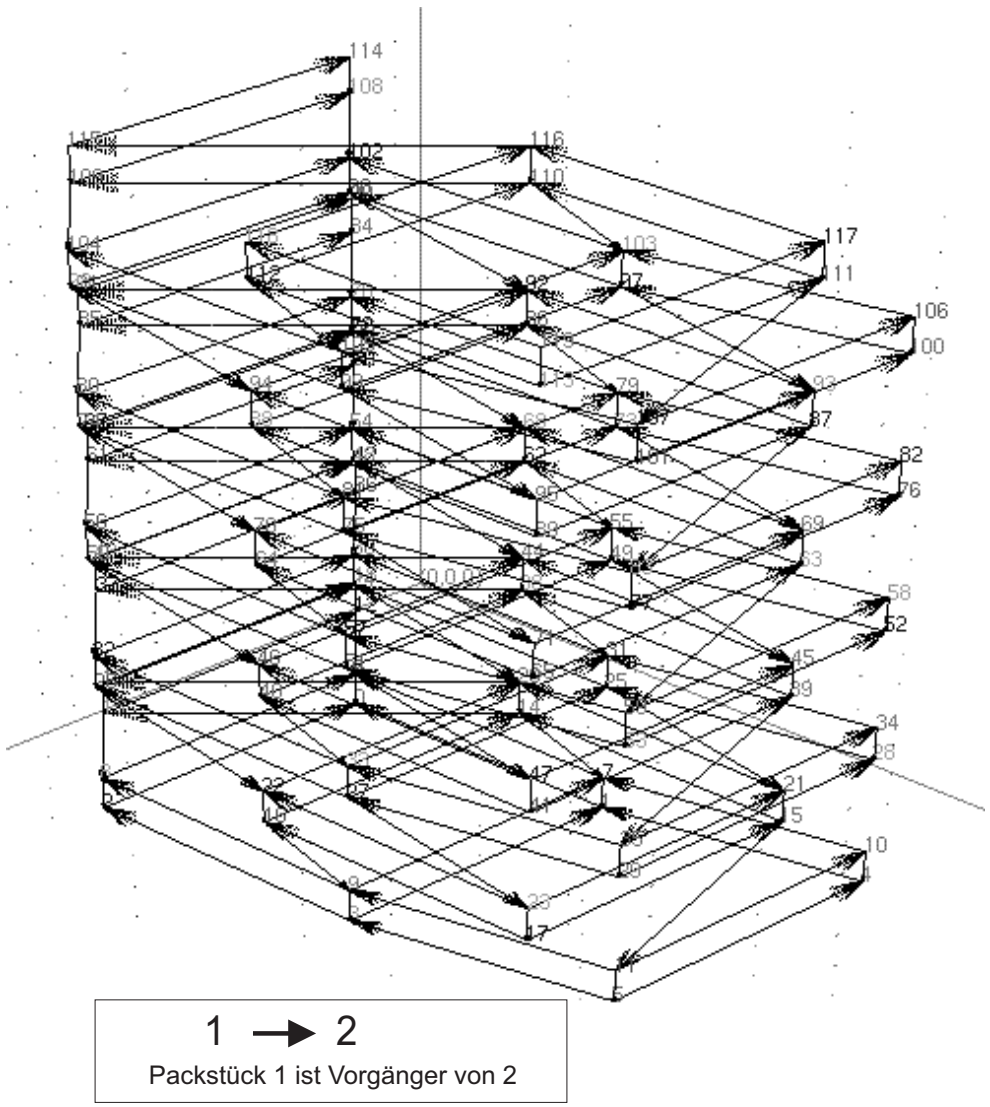


Abbildung 5.6: Reduzierter dreidimensionaler (Nachfolger-) Packstückgraph

Abb. 5.7 verdeutlicht, warum die Reduzierung des vollständigen für größere Packmuster zwingend erforderlich ist. Wie im Bild zu sehen, entsteht hier eine große Verflechtung von Pfeilen und Knoten. Um eine optimale Lösung zu erhalten, müßten alle diese Verflechtungen durchlaufen und bewertet werden. Da in Abhängigkeit des gegebenen Packmusters nicht unbedingt ein Binärbaum (genau zwei Nachfolger) vorliegt, ergibt sich die Zahl der möglichen Reihenfolgen n in Abhängigkeit der Knotenzahl N zu:

$$2^N < n < N! \tag{5.3}$$

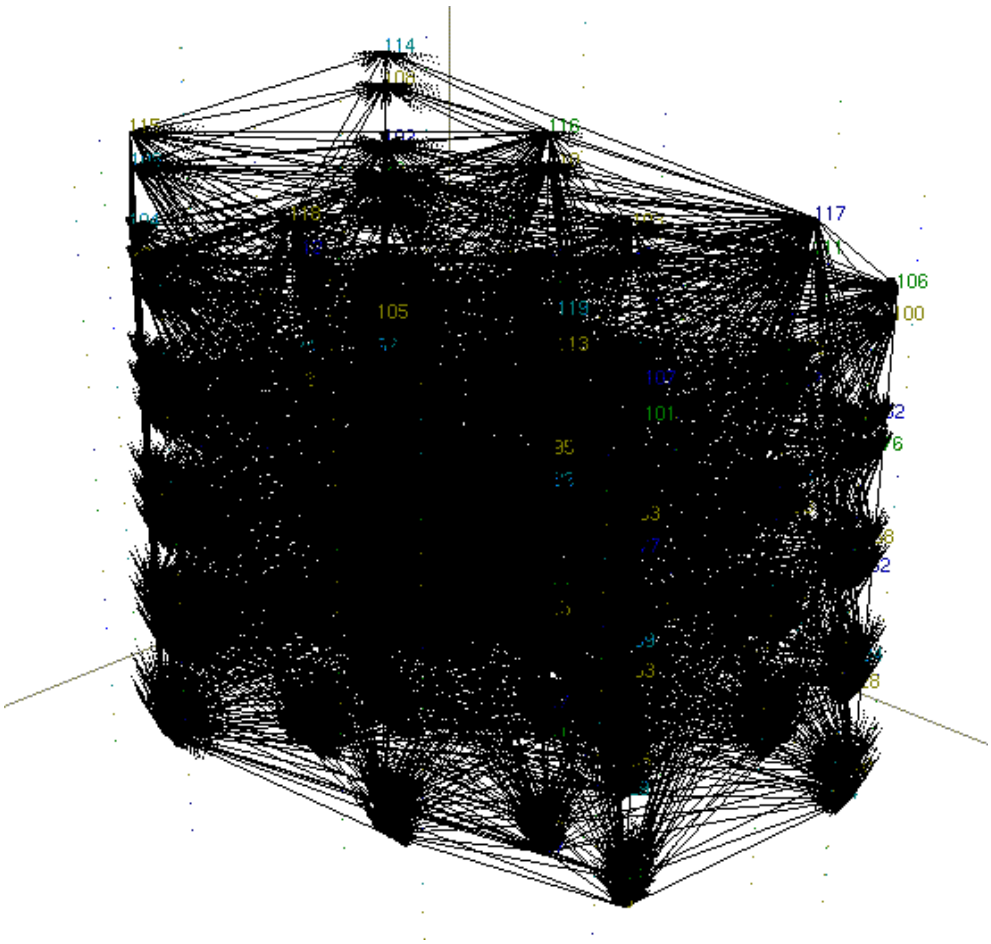


Abbildung 5.7: Vollständiger dreidimensionaler Packstückgraph

Aus obiger Formel und aus Abb. 5.7 ist ersichtlich , daß eine optimale Lösung mit vollständiger Suche bei großer Packstückzahl nahezu hoffnungslos ist, da sich bei 15 Packstücken bereits $3 * 10^{13}$ mögliche Reihenfolgen ohne Berücksichtigung einer Einfügerichtung ergeben. Somit müssen Algorithmen mit neuen Ansätzen entwickelt werden, die die Anzahl der Möglichkeiten reduzieren aber dennoch möglichst optimale Lösungen liefern.

5.4 Verfahren zur Bildung von Ablagezyklen

Das Problem der Bildung von Ablagereihenfolgen ist eine Kombination aus Lösen des zweidimensionalen Packproblems und optimalem Durchlauf eines Graphen. Alle Packstücke einer Palette sollen so abgelegt werden, daß möglichst wenig Anfahrpunkte (zeitoptimal) und keine Kollisionen entstehen.

Mathematisch formuliert lautet das Problem also wie folgt:

Gegeben sind

- ein virtueller Greifer mit nxm Matrix und
- mehrere Reihenfolgen von Palettenblöcken mit kxl Matrix

und gesucht wird die beste aller möglichen gültigen Reihenfolgen der Greiferzyklen. Prinzipiell gibt es zwei Vorgehensweisen :

1. Virtuellen Greiferblock mit Palettenblöcken füllen.
2. Virtuellen Greiferblock auf Palettenblöcke aufteilen.

Es wird die erste Möglichkeit gewählt, da der Greiferblock nicht immer vollständig gefüllt ist und somit ein erneutes Durchsuchen für die verringerte Packstückzahl am Greifer notwendig ist. Weiterhin kann das Verfahren auch zur Depalettierung benutzt werden.

Unter Berücksichtigung einer Einfügerichtung wird eine Blockreihenfolge gesucht, die den Greifer immer möglichst vollständig auffüllt. Das Einfügen aus einer Richtung ermöglicht ein schnelleres Ablegen der Packstücke bei einigen Roboterkinematiken (z. B. SCARA) und somit eine Geschwindigkeitssteigerung.

Anders formuliert, möglichst viele Packstücke müssen pro Zyklus in ein Greiferrechteck gelegt werden ($\rightarrow$ zweidimensionales Packproblem). Bei der Bildung der endgültigen Ablagereihenfolge gibt es zusätzlich zu den Möglichkeiten nach (5.3) noch g! Möglichkeiten für die Greiferpackstücke g. Bei der Komplexität des Problems - sprich Größe des Suchraumes -, erscheint es ratsam, das Problem entweder in Teilstücke (Blöcke) zusammenzufassen oder zufallsbasierte Algorithmen anzuwenden, um globale bzw. zumindest lokale Maxima zu finden. Allerdings können gute Lösungen durch die Einteilung in Blöcke entfallen, was extrem negative Auswirkung auf die Gesamtlösung hat. Daher ist eine geschickte Aufteilung in Subblöcke eventuell sinnvoll.

Tabelle 5.3 zeigt eine Übersicht der nachfolgend beschriebenen Verfahren zur Erzeugung einer Packstück- bzw. Blockreihenfolge.

5.4.1 Genetischer Algorithmus

Das erste Verfahren basiert auf einem genetischen Algorithmus, der zufällige Reihenfolgen durch Rekombination und Mutation erzeugt. Eine Bewertungsfunktion entscheidet anschließend, ob sich eine Teillösung weiter fortpflanzt oder nicht. Die Mutationsfunktions hilft, lokale Minima zu verlassen. Ein Vorteil des genetischen Algorithmus ist die konstante Laufzeit

Verfahren	Prinzip
Genetischer Algorithmus	Packmuster auf Bitstring abbilden, Bitstring zufällig mutieren, rekombinieren und bewerten
Packstückgraph	Graph aus Packstücken aufbauen, Durchsuchen des Packstückgraphen
Blockgraph	Packstücke zu Blöcken zusammenfassen, Graph aus Blöcken aufbauen, Durchsuchen des Blockgraphen
Blockgraph mit vorzeitiger Subblockaufteilung	Packstücke zu Blöcken zusammenfassen, Blöcke in Vielfache des Greifers aufteilen, Graph aus Subblöcken aufbauen, Durchsuchen des Blockgraphen

Tabelle 5.3: Verfahren zur Zyklusbildung

- nach x Generationen wird der Algorithmus beendet und eine Lösung ausgegeben. Nachteile sind die fehlende Reproduzierbarkeit und die problematische Abbildung eines zweidimensionalen Problems auf einen eindimensionalen Bitstring sowie dessen Bewertung. Der gravierende Nachteil des diskontinuierlichen Lösungsgrades verhindert eine weitere Nutzung von genetischen Algorithmen bei Palettierproblemen.

5.4.2 Packstückgraphverfahren

Hier wird der aufgebaute Packstückgraph rekursiv in einer bestimmten Suchtiefe durchlaufen. Innerhalb der Suchtiefe werden alle Möglichkeiten, die Packstücke in Zyklen aufzuteilen, vollständig durchsucht und bewertet. Der vereinfachte rekursive Algorithmus in Form der Funktion BesuchePackstückKnoten() lautet wie folgt:

```
int BesuchePackstückKnoten(int FeldIndex)
{
 // Suchtiefe hochzählen
 SuchTiefe++;
 if(P[FeldIndex].AnzahlNachfolger <= 0 && SuchTiefe >= MaxAnzahlPackstücke)
 {
// keine naechsten Packstuecke mehr -> Bewertungsfunktion aufrufen
SchreibeLösungSuchReihenfolge();
```

```
 }
 else if( P[FeldIndex].AnzahlNachfolger >0) // Nachfolger vorhanden
 {
// Besuche alle nächsten Knoten
for(int j=0;j<NachfolgerPackstücke;j++)
{
  // Vorgänger des Nachfolgers ermitteln
  // keine weiteren Vorgänger im Knoten vorhanden ->besuche Knoten
  if(P[P[FeldIndex].NachfolgerPackstückeListe[j].AnzahlVorgängerPackstücke==1)
  {
 // rekursiver Aufruf -> besuche nächsten Knoten
 if(!BesuchePackstückKnoten(P[FeldIndex].NachfolgerPackstückeListe[j]))
return false;
  }
}
 }
 // Suchtiefe zurücksetzen
 SuchTiefe--;
 return true;
}
```

Die Funktion liefert den Wert true (wahr), wenn ein Knoten erfolgreich besucht werden konnte. Bei Abbruch der Suche wird false (falsch) zurückgegeben.

Die Abbildung Abb. 5.8 verdeutlicht den Suchvorgang anhand eines einfachen Packmusters mit 2*2 Mehrfachgreifer. Sämtliche Kombinationsmöglichkeiten sowohl für den Greifer als für die Palette werden durchsucht.

5.4.3 Blockgraphverfahren

Alle gleichartigen Packstücke werden zu Blöcken zusammengefaßt, der Graph aufgebaut und durchlaufen. Durch die Zusammenfassung wird die Anzahl der Möglichkeiten erheblich reduziert - allerdings unter Umständen auch gute Möglichkeiten nicht erkannt. Der Ablauf beim Algorithmus mit Blockbildung Abb. 5.9 sieht allgemein wie folgt aus:

1. Blockbildung (natürliche x-Blöcke)
2. Schnittblockaufteilung
3. Blockgraph durchlaufen → Blockreihenfolgen ermitteln

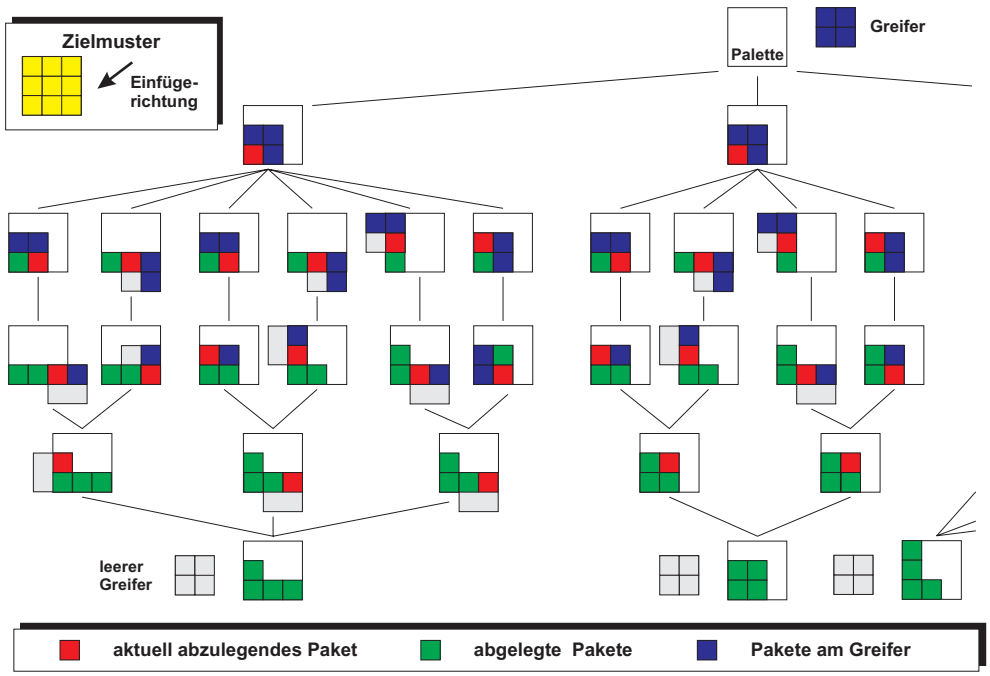


Abbildung 5.8: Suchbaum des ersten Zyklus bei vollständiger Suche mit Vierfachgreifer

4. Aufteilung in Subblöcke: Vielfache des Greiferblocks $v_x N, M v_y$
5. Auffüllen des Greiferblocks aus zulässiger Richtung
6. Ausgabe der Anfahrpunkte und Bitkombinationen für Zuführungen und Greifer

5.4.4 Blockgraphverfahren mit vorzeitiger Subblockaufteilung

Um bessere Lösungen zu erhalten, wird eine sogenannte vorzeitige Subblockaufteilung von Schnittblöcken eingeführt. Alle Blöcke werden dabei vor dem Aufstellen des Blockgraphen in Subblöcke sprich Vielfache des Greifers eingeteilt. Der resultierende Blockgraph wird dadurch komplexer. Der Ablauf stellt sich somit wie folgt Abb. 5.9 dar:

1. Blockbildung (natürliche x-Blöcke)
2. Schnittblockaufteilung
3. Aufteilung in Subblöcke: Vielfache des Greiferblocks $v_x N, M v_y$
4. Blockgraph durchlaufen $\rightarrow$ Blockreihenfolgen ermitteln
5. Auffüllen des Greiferblocks aus zulässiger Richtung

6. Ausgabe der Anfahrpunkte und Bitkombinationen für Zuführungen und Greifer

Das Blockbildungsverfahren kann prinzipiell mit oder ohne Aufteilung in Greifersubblöcke realisiert werden. Die vorzeitige Aufteilung der Palettenblöcke in Subblöcke des Greifers hat den Vorteil, daß evtl. bessere Lösungen durch einen komplexeren Graph gefunden werden. Nachteil ist die größere Rechenzeit. Abb. 5.9 zeigt den Ablauf für beide Verfahren.

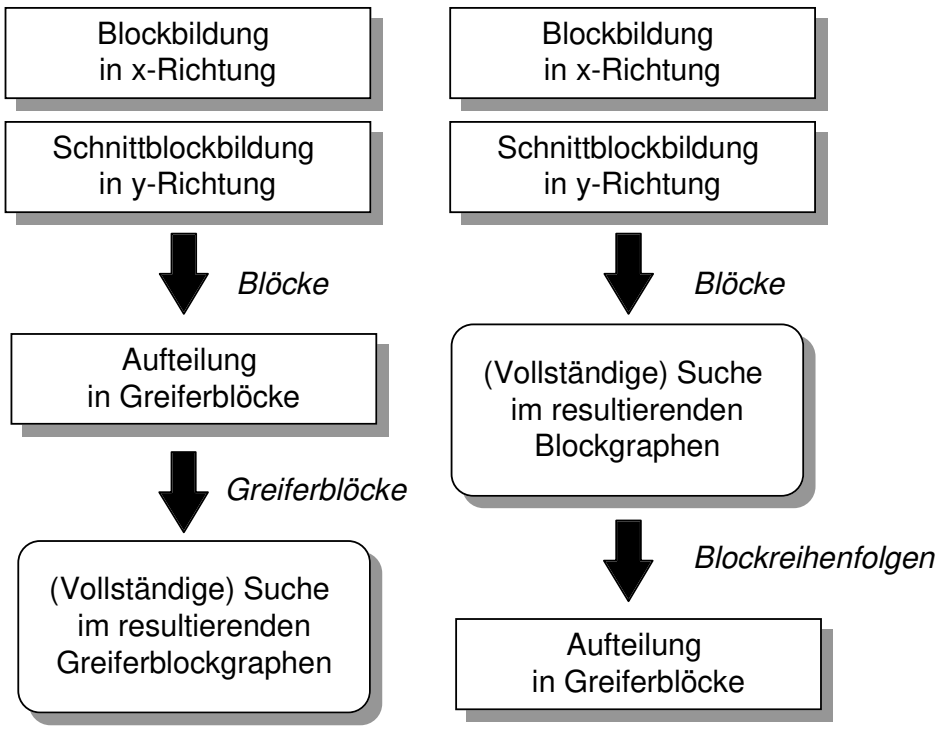


Abbildung 5.9: Blockheuristiken zur Zyklusbildung

Im Folgenden werden die einzelnen Unterpunkte der Blockbildung näher beschrieben.

5.4.4.1 Algorithmus zur Erzeugung natürlicher x-Blöcke

Zuerst werden alle Packstücke mit Insertionsort [61] nach z y x sortiert und dann in x-Richtung durchsucht. Das Packmuster wird einmal durchlaufen und die Packstücke nach verschiedenen Kriterien verglichen. Liegen Packstücke in einer Zeile an, werden diese in einem temporären Zeilenblock gespeichert (Abb. 5.10). Wenn die Packstücke nicht anliegen, entsteht ein Umbruch, d. h. ein neuer temporärer Zeilenblock muß geschaffen werden. Gleichzeitig wird nach einem Umbruch geprüft, ob der vorherige temporäre Zeilenblock in y - Richtung an bereits vorhandene Blocke angelegt werden kann. Der Blockbildungsalgorithmus kann vereinfacht wie folgt dargestellt werden:

```
for(int i=1;i<=MaxAnzahlPackstücke;i++)
```

```
{
// Sind die Packstücke in der gleichen Zeile ?
if(GleicheZeile(P[i], P[i-1])
{
  // wenn das nachfolgende Packstück an das vorherige (in y) anliegt
  // -> Eintrag in die tmp Zeile
  TemporäreZeile.Insert(i); // insert i
}
else // neue Zeile, Zeilenwechsel
{
// Durchsuche alle bisher bekannten Blöcke
for(int b=(BlockZähler-1) ; b>=0; b--)
{
// der zugehörige Block muß die gleiche Anzahl an Packstücken in x haben
if( BlockArray[b].PackagesInX() != TemporäreZeile.PackstückeInX()   )
  continue;
// Überprüfe Packstückmaße
if(ÜberprüfePackstückmaße())
  continue;
// wenn Blockhoehe nicht identisch -> Abbruch, da nach z sortiert
if(ÜberprüfeBlockHöhe())
  break;
// wenn TemporäreZeile nicht einem Block war,
// TemporäreZeile in einem neuen BLock speichern
if(!SucheBlock(b))
  ErzeugeNeuenBlock(TemporäreZeile);
 }
}
}
```

5.4.4.2 Schnittblockbildung

Schnittblöcke dienen der Rückgewinnung von Kombinationsmöglichkeiten zur Erzeugung besserer Lösungen. Abb. 5.11 zeigt drei Blöcke, die in y-Richtung aneinander anliegen und somit zwei Schnittblöcke bilden.

Die Bedingungen zur Bildung von Schnittblöcken sind

1. identische Packstückmaße (Länge, Höhe, Breite),

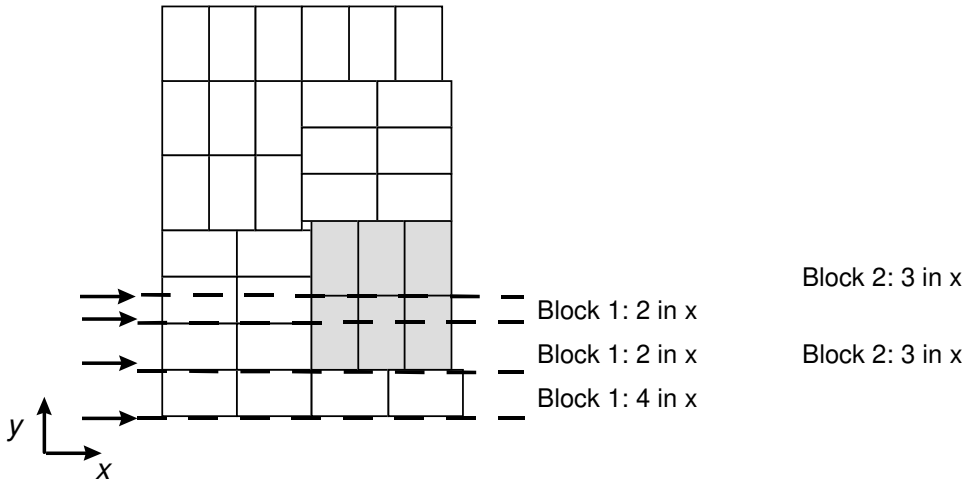


Abbildung 5.10: Blockbildung

2. bündiges Anliegen in y-Richtung und

3. Überlappung in x-Richtung.

Nachteil dieses Verfahrens ist der erhöhte Bedarf an Rechenzeit.

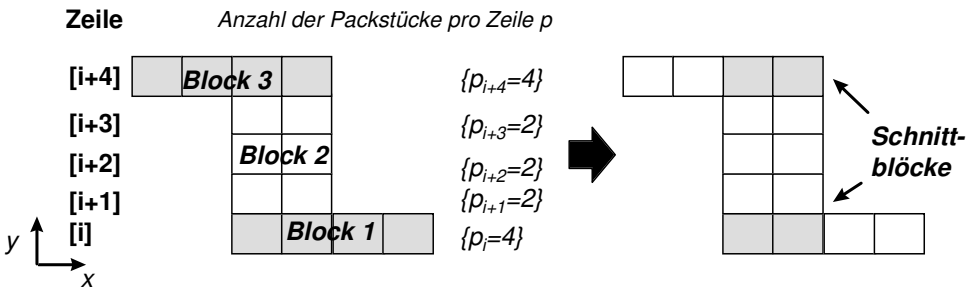


Abbildung 5.11: Schnittblockbildung

5.4.4.3 Subblockbildung

Zur endgültigen Bildung eines Zyklus ist die Aufteilung von (Paletten-) Blöcken B_xB_y in Subblöcke des Greifers G_xG_y erforderlich. B_x beschreibt die Anzahl der Packstücke in x-Richtung eines Blockes, B_y in y-Richtung. G_x und G_y gelten entsprechend für den Greifer.

Bei der Aufspaltung können prinzipiell drei Fälle auftreten (Abb. 5.12):

1. Der Palettenblock ist ein Vielfaches des Greiferblocks.

2. Der Palettenblock ist ein Vielfaches des Greiferblocks plus Rest in x oder y.

3. Der Palettenblock ist ein Vielfaches des Greiferblocks plus Rest in x und y.

Die resultierenden Subblockgrößen berechnen sich nach:

$$Rest_x = B_x - X_n G_x \text{ und } Rest_y = B_y - Y_n G_y$$

mit

$$X_n = floor(\frac{B_x}{G_x}) \text{ und } Y_n = floor(\frac{B_y}{G_y})$$

Die Funktion floor bewirkt das Abschneiden der Nachkommastellen. Die Anzahl der neuen Blöcke beträgt:

$$B_{neu} = X_n Y_n + 1(Rest_x)Y_n + 1(Rest_y)X_n + 1(Rest_x)1(Rest_y)$$

mit

$$1(n) = \begin{cases} 0 & \forall \quad n <= 0 \\ 1 & \forall \quad n > 0 \end{cases}$$

Im Ablagealgorithmus wird die genaue Anzahl der neuen Blöcke u. a. zur Anforderung von Arbeitsspeicher benötigt. Darüber hinaus kann die benötigte anhand der insgesamt vorhandenen Blöcke abgeschätzt werden. Für den dritten Fall in Abb. 5.12 beträgt $B_{neu} = 4$ mit $B_x = B_y = 3$ und $G_x = G_y = 2$ bzw. $X_n = Y_n = 1$ und $Rest_x = Rest_y = 1$.

5.4.4.4 Berechnung von Anfahrpunkten

Zur Erzeugung von Anfahrpunkten müssen die Greifermatrizen in Zyklen mit Anfahrpunkten und Greiferansteuerbits konvertiert werden. Es wird zwischen Abhol- und Ablagepunkt unterschieden.

Die Abholmittelpunkte (Zuführung) werden unter Berücksichtigung der Anschlageckenausrichtung berechnet. Die Mittelpunkte M ergeben sich aus Eckenvektor der Zuführung Z plus Verschiebung ΔV:

$$\vec{M} = \vec{Z} + \Delta\vec{V}$$

In Abhängigkeit der Ausrichtung der Anschlagecke existieren prinzipiell vier Fälle:

$$\Delta\vec{V}_{obenlinks} = \begin{pmatrix} -\Delta x \\ \Delta y \end{pmatrix}; \Delta\vec{V}_{untenlinks} = \begin{pmatrix} -\Delta x \\ -\Delta y \end{pmatrix}$$

$$\Delta\vec{V}_{obenrechts} = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}; \Delta\vec{V}_{untenrechts} = \begin{pmatrix} \Delta x \\ -\Delta y \end{pmatrix}$$

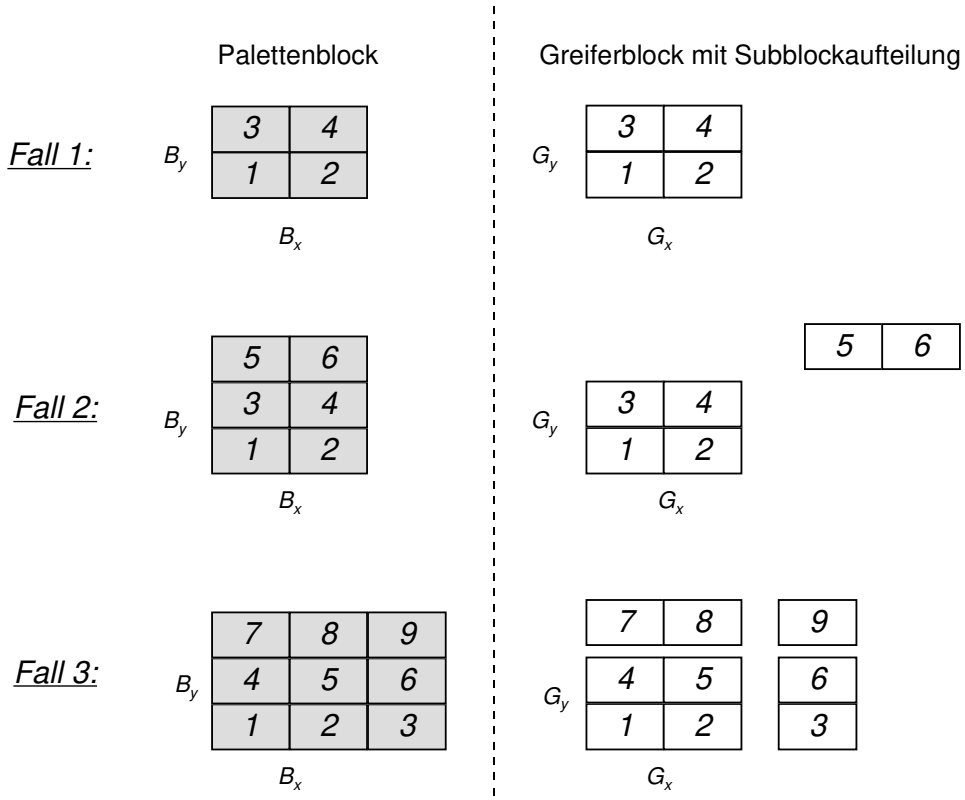


Abbildung 5.12: Subblockbildung

mit $\Delta x = \frac{G_{XL} * P_{XL}}{2}$ und $\Delta y = \frac{G_{YL} * P_{YL}}{2}$.

Die Ablagepunkte ergeben sich aus Anschlagvektor der Abgabe (Palette) A, der relativen Position des Packstückes zur linken unteren Palettenecke ΔN und des Differenzvektors zur Anschlagecke sowie des Differenzvektors zum Mittelpunkt ΔD:

$$\vec{M_{Ablage}} = \vec{A} + \Delta\vec{N} + \Delta\vec{D}$$

Für die relative Position des Packstücks zur linken unteren Palettenecke existieren vier Fälle:

$$\Delta\vec{N}_{obenlinks} = \begin{pmatrix} X_{LPalette} \\ 0 \end{pmatrix}; \Delta\vec{N}_{untenlinks} = \begin{pmatrix} -X_{LPalette} \\ -Y_{LPalette} \end{pmatrix}$$

$$\Delta\vec{V}_{obenrechts} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \Delta\vec{V}_{untenrechts} = \begin{pmatrix} 0 \\ -Y_{LPalette} \end{pmatrix}$$

Der Differenzvektor eines Packstückes oder Blocks zum Greifermittelpunkt läßt wie folgt ermitteln (Grundausrichtung):

$$\Delta\vec{D}(n,m) = \begin{pmatrix} X_{LPackstück}(\frac{X_N}{2} - n) \\ Y_{LPackstück}(\frac{Y_N}{2} - m) \end{pmatrix}$$

mit n Index in X-Richtung des Greifers und m Index in Y-Richtung des Greifers.

5.4.4.5 Berücksichtigung von Greiferdrehungen

Gedrehte Packstücke ($+-90°$ und $180°$) bzw. Blöcke müssen erkannt und gesondert behandelt werden, weil sonst ein neuer virtueller Greifer für das gedrehte Packstückmaß erzeugt wird (Abb. 5.13), was zur Folge hat, daß der Greifer beim Blockwechsel von ungedrehten auf gedrehte Packstücke nicht vollständig gefüllt wird. Somit steigt die Anzahl der Anfahrpunkte und die Gesamtlösung wird schlechter. Für das Beispiel in Abb. 5.13 ist ein zusätzlicher Anfahrpunkt notwendig. Insbesondere bei der Subblockbildung müssen Greiferdrehungen vorzeitig berücksichtigt werden.

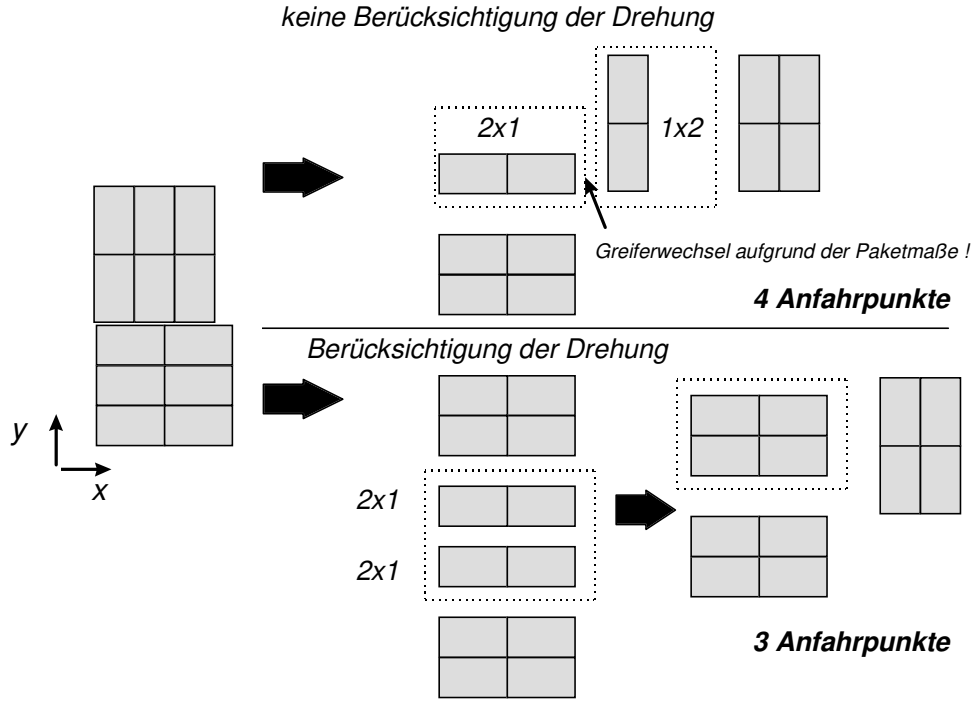


Abbildung 5.13: Berücksichtigung von Greiferdrehungen

5.5 Ergebnisse der entwickelten Verfahren

In diesem Kapitel werden die Lösungen und das Laufzeitverhalten der einzelnen Verfahren näher untersucht.

Zum Test der Algorithmen wurden neun zweidimensionale und ein dreidimensionales Packmuster ausgewählt. In Abb. 5.15 und Abb. 5.14 sind beispielhaft zwei Muster dargestellt. Die übrigen Muster sind im Anhang A.1 zu finden. Um praxistaugliche Rahmenbedingungen zu erfüllen, wurden sieben dieser Testmuster von einem Industriepartner zur Verfügung gestellt.

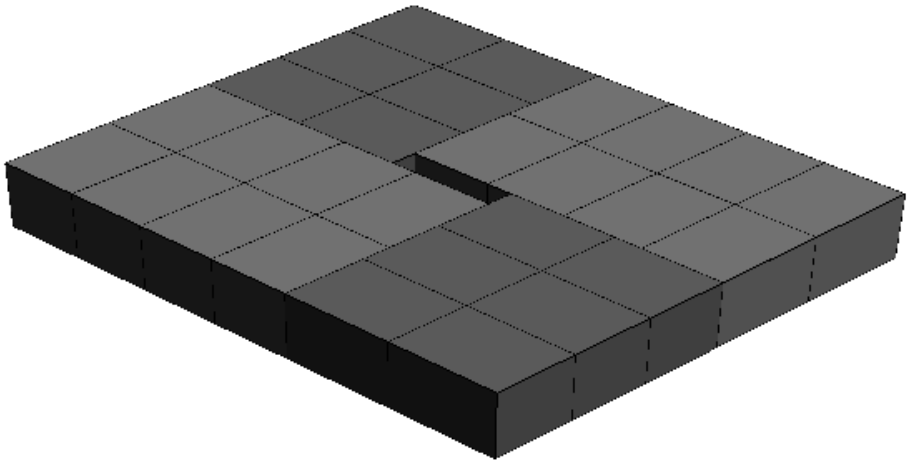


Abbildung 5.14: Packmuster 'a-jaffa'

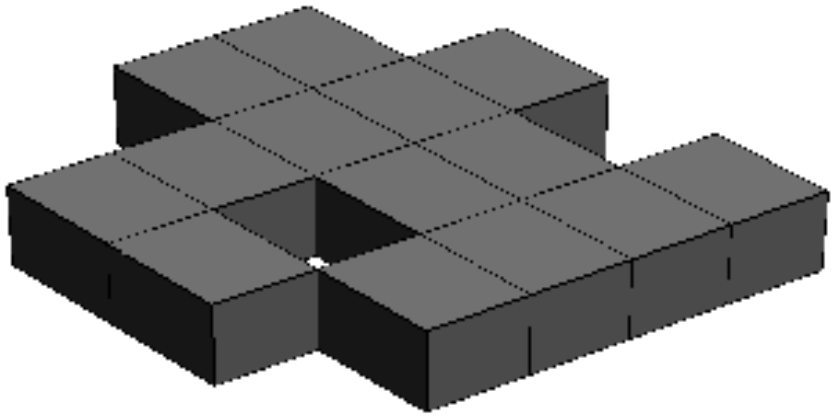


Abbildung 5.15: Packmuster 'fract'

Somit wird ein Teil der in der Praxis häufig vorkommenden Packmuster abgedeckt. Tabelle 5.4 zeigt einen Ausschnitt der wichtigsten Ergebnisse für unterschiedliche Packmuster mit verschiedenen Greifertypen bei Verwendung des Blockalgorithmus und des Packstückgraphen. Eine Auflistung aller Meßwerte ist im Anhang zu finden. Die Messungen wurden auf einem PC mit Pentium 200 Prozessor und 64 MB Hauptspeicher durchgeführt.

Erwartungsgemäß zeigt der Blockalgorithmus bei blockorientierten Packmustern einen deutlich geringeren Rechenzeitbedarf. Dies gilt besonders für Mehrfachgreifer mit mehr als drei Packstücken in x oder y Richtung . Hier beträgt die Zeitdifferenz teilweise mehrere hundert Sekunden. Im Bereich kleinerer Greifer (< 2) weisen beide Algorithmen einen ähnlichen Re-

Packmuster	Greifer	Rechenzeit Block	Rechenzeit Packstückgraph
a-jaffa (28 Packstücke), 4 Blöcke	2*1	0.691 s	1 s
	3*2	0.06 s	14 s
	4*2	0.06 s	301 s
	4*3	0.06 s	$>$ 24 h
fract (14 Packstücke), Fraktal	2*1	0.16 s	$<$ 1 s
	2*2	0.10 s	$<$ 1 s
	4*2	0.33 s	5 s
	4*3	0.16 s	1310 s
	4*4	0.05 s	31266 s

Tabelle 5.4: Vergleich der Rechenzeiten (Ausschnitt)

chenzeitbedarf auf. Die Qualität der Lösungen ist in den getesteten Fällen nahezu gleich. In nur einem Fall schnitt der Blockalgorithmus schlechter ab. Die Vermutung, daß die Blockbildung zu einer erheblichen Reduzierung von Lösungsmöglichkeiten führt, konnte im Versuch bestätigt werden. Ferner lieferte der Blockalgorithmus auch bei optimierten Packmustern gute und schnelle Lösungen. In Abb. 5.16 ist die Rechenzeit und die Anzahl der Greiferpackstücke aufgetragen. Die Rechenzeit blieb für alle Greifer nahezu konstant. Dahingegen nahm die Rechenzeit beim Packstückgraphalgorithmus mit der Anzahl der Packstücke am Greifer deutlich zu. Dieses Verhalten liegt im Verfahren begründet, weil hier pro Packstückreihenfolge noch zusätzlich g! Greifermöglichkeiten durchsucht werden. Im Falle des Testmusters 'fract' mit 4*3 Greifer ergeben sich somit statt der 14! insgesamt $14! \cdot (12!)$ Möglichkeiten. Abschließend können die Verfahren wie folgt bewertet werden: Das Packstückgraphverfahren

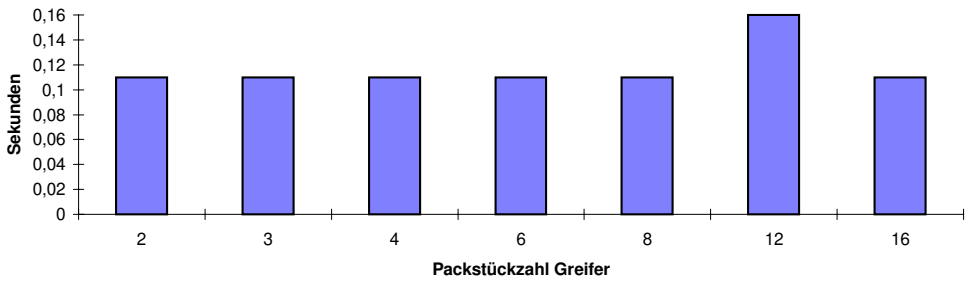


Abbildung 5.16: Rechenzeit Blockverfahren (Packmuster 'fract')

Verfahren	Bewertung
Genetischer Algorithmus	ungeeignet, da inkonsistente Lösung
Packstückgraph	bis 3*3 Greifer gut geeignet, ab 3*3 bedingt praxistauglich (hohe Rechenzeit)
Blockgraph	durchgängig gut geeignet, eventuell schlechtere Lösung bei optimierten Packmustern

Tabelle 5.5: Bewertung der Verfahren zur Zyklusbildung

liefert für kleine Greifer bis zu drei Packstücken in jede Richtung bei schneller Geschwindigkeit gute Lösungen. Bei größeren Greifern jedoch (mehr als 6 Packstücke) , steigt die Rechenzeit deutlich an, so daß die Berechnung aller Zyklen bis zu mehreren Stunden dauern kann. Für einen Praxiseinsatz ist dieser Zeitbedarf nicht mehr akzeptabel, da hier z. B. geänderte Packmuster nach wenigen Minuten gestapelt werden müssen. Der Blockalgorithmus erweist sich als geeignetes Verfahren, um schnell eine gültige Ablagereihenfolge zu erhalten. Der Rechenzeitbedarf lag im Falle aller Testmuster im Bereich weniger Sekunden. Das Packstückgraphverfahren eignet sich vorwiegend für kleine Greifer und sogenannte optimierte Packmuster (siehe Packmuster opt im Anhang), die mehrfach verschachtelte Packstücke enthalten. In Tabelle 5.5 ist eine kurze Zusammenfassung der Bewertungen aufgeführt.

5.6 Graphische off-line "Programmierung"

Im Gegensatz zu herkömmlichen textuellen off-line Programmiersystemen ist die graphische off-line Programmierung auch technisch nicht versierten Anwendern zugänglich. Die Programmierung beschränkt sich bei diesem System auf einfaches Ziehen und Loslassen (drag and drop) von Packstücken. Da die Programmierung im ursprünglichen Sinn komplett entfällt, wird an dieser Stelle der Begriff off-line Palettierung eingeführt. Abb. 5.17 zeigt diesen Ablauf im Detail.

Grundsätzlich wird zwischen den Benutzergruppen Systemkonfigurierer und Bediener (Operator) unterschieden. Zunächst konfiguriert ein Techniker das System einmalig anhand der spezifischen Anlagendaten. Im Anschluß daran kann auch der technisch nicht versierte Benutzer neue Packmuster graphisch erzeugen und diese automatisch palettieren. Der Ablagealgorithmus optimiert die Reihenfolge der Plazierung von Packstücken nach Schnelligkeit und Kollisionsfreiheit unter Berücksichtigung des Anlagenlayouts. Die generierten Zyklusdaten enthalten sowohl Informationen für die SPS als auch für die Positionierungseinheit

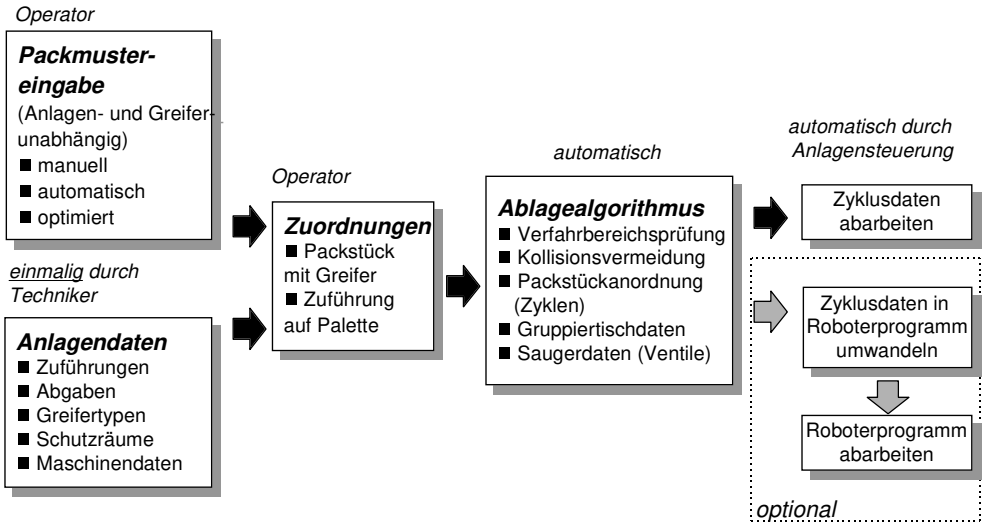


Abbildung 5.17: Ablauf der graphischen off-line Palettierung

(Robotersteuerung).

Die grundsätzliche Ausgabeschnittstelle zwischen off-line System und Anlagensteuerung ist eine ASCII-Datei, die dann prinzipiell von jeder Steuerung eingelesen und ausgewertet werden kann. Der wesentliche Vorteil ist die Unabhängigkeit vom Steuerungstyp des jeweiligen Herstellers. Allerdings werden erhöhte Anforderungen an die Steuerungssprache gestellt - diese muß die ASCII-Datei lesen können, was bei allen Steuerungen der Fall ist, die IRL [60] unterstützten. Falls die Steuerung keine ASCII-Daten einlesen kann, wird optional ein Postprozessor aufgerufen, der die Zyklusdaten in ein steuerungsspezifisches Programm umwandelt. Die Umwandlung verbraucht allerdings zusätzliche Rechenzeit.

Kapitel 6

PC-basierte Palettieranlagensteuerung

Wie bereits in den Kapiteln Stand der Technik und des Wissens angedeutet, gibt es im Bereich der Robotik besonders auf Hardwareebene zu unflexible Speziallösungen. Daher wurde ein Automatisierungskonzept entwickelt, das auf Standards basiert, aber dennoch flexibel genug ist, um spezifische Kundenlösungen zu realisieren. Das Konzept ist prinzipiell auch auf andere Fertigungsanlagen übertragbar.

Im Rahmen dieser Arbeit wurden zwei unterschiedliche Steuerungstypen aufgebaut:

- PC-basierte Palettieranlagensteuerung auf DOS Basis, die mittlerweile in der Industrie mehrfach eingesetzt wird.
- PC-basierte Palettieranlagensteuerung auf Basis von Windows NT, die noch im Stadium eines Prototypen ist.

Das Hardwarekonzept ist für beide Steuerungen identisch. Nachfolgend werden daher einmal der Hardwareaufbau und zwei unterschiedliche Ausprägungen des allgemeinen Softwarekonzeptes inklusive der Achspositionierung im Detail beschrieben.

6.1 Allgemeines Hardwarekonzept

Das Hardwarekonzept Abb. 6.1 basiert im wesentlichen auf einem IPC mit Feldbus. Es sind alle Steuerungsmodule (inklusive Module für den Roboter) an einem Feldbus angeschlossen und gewährleisten somit Flexibilität und Dezentralität. Die Liste der Hardwarekomponenten sieht allgemein wie folgt aus :

- IPC

- Feldbuskarte (Profibus, CAN, Interbus, firmenspezifisch)
- I/O Feldbusmodule (Digital, Analog, Inkrementalgeber etc.)
- Antriebsmodule mit Feldbusanschluß

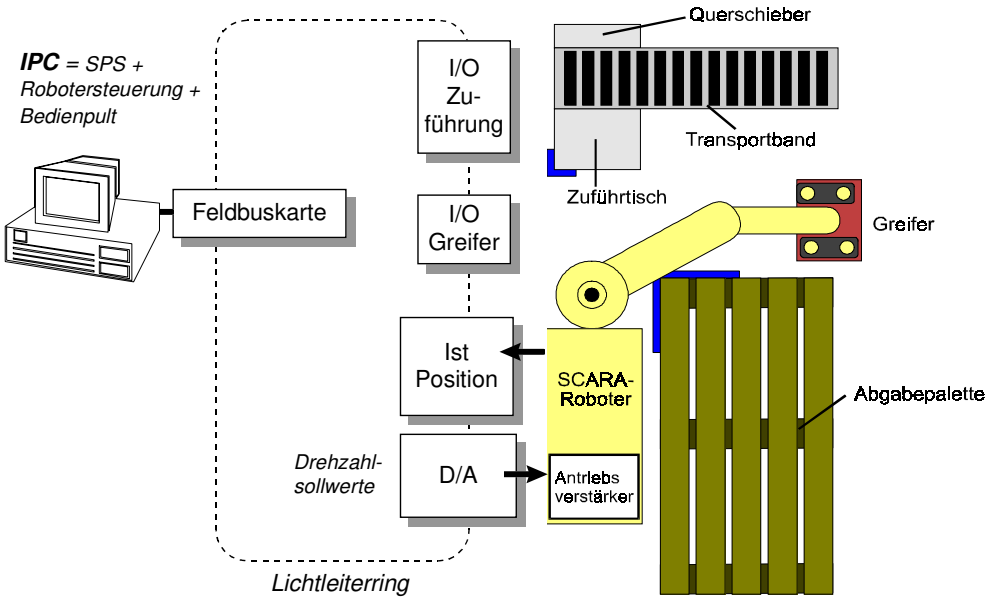


Abbildung 6.1: Hardwareaufbau mit Lichtleiterfeldbus

6.2 Allgemeines Softwarekonzept

Das neue Softwarekonzept sieht eine Verlagerung aller Funktionsgruppen auf die Softwareebene eines Standard IPC vor, d. h. im Falle einer Palettieranlage werden SPS, Bedienpult und Achspositionierung (Abb. 6.2) als Softwaremodule realisiert. Aus Gründen der Übersichtlichkeit wurde die funktionale Trennung von SPS und reiner Ablaufsteuerung für die Achspositionierung beibehalten.

Zur Inbetriebnahme einer Palettieranlage sind einmalig folgende Schritte notwendig:

1. SPS-Programm für die Anlage schreiben (Notaus, Sicherheitsabfragen).
2. Achsdaten für die Positionierung eintragen und im Handbetrieb testen.
3. Roboterprogramm zur kartesischen Positionierung mit allen Achsen schreiben und testen.

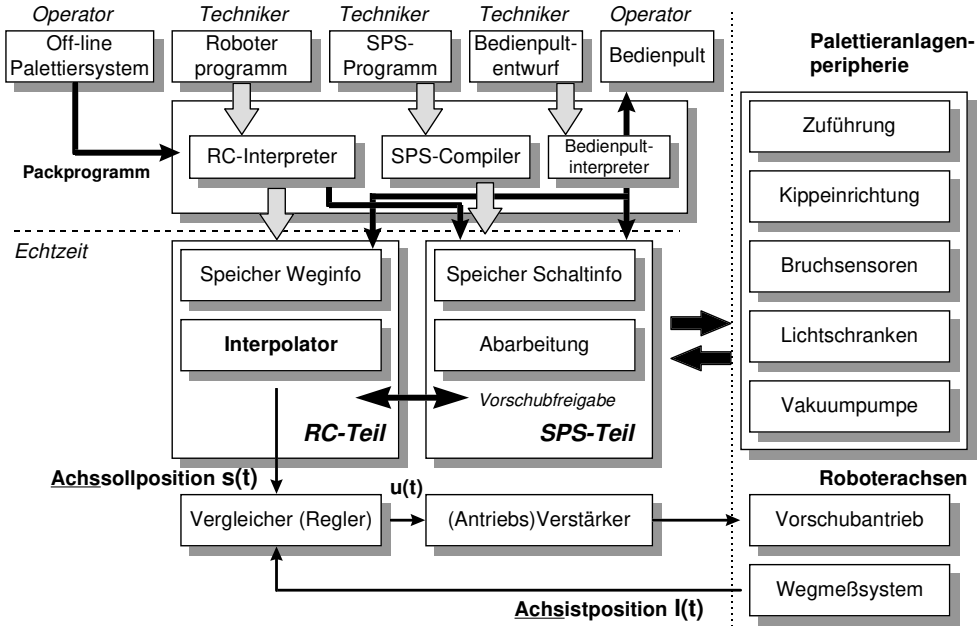


Abbildung 6.2: Allgemeines PC-basiertes Steuerungskonzept

4. Roboterprogramm zum Einlesen von Packmusterdateien entwickeln.

5. Packmuster mit off-line Palettiersystem erzeugen und testen.

6. Bedienpult entwerfen und testen.

Nach erfolgreicher Inbetriebnahme kann ein beliebiges Packmuster erzeugt und ohne zusätzlichen Programmieraufwand palettiert werden. Das Gesamtsystem unterscheidet zwischen den Benutzern Anlageneinrichter und Anlagenbediener. Ein Techniker des Anlagenhersteller entwickelt einmalig Roboter- und SPS-Programme und muß somit vollen Zugriff auf alle Funktionen der Steuerung haben. Der Operator - letztlich der Anwender der Palettieranlage - hat nur eingeschränkten Zugriff auf eine Visualisierung (Bedienpult) mit Packmusterübertragung.

Im Palettierbetrieb liest das Roboterprogramm die Packmusterdaten ein und generiert die Weginformationen für den Interpolator, der dann Sollwerte für die Antriebe erzeugt. Ferner übergibt die Robotersteuerung Ansteuerinformationen für die Kippeinrichtungen, Lichtschranken und Zuführungen an die SPS. Die Anlagenperipherie inklusive der sicherheitrelevanten Funktionen wie Notaus und Bruchsensoren werden zyklisch von einem SPS-Programm ausgewertet. Betriebsdaten und Störungen werden protokolliert und zyklisch in eine Logdatei auf die Festplatte des Steuerungs-PC geschrieben.

6.3 Roboterpositionierung mit Splines

Damit beliebige (Roboter-) Kinematiken flexibel eingesetzt werden können, müssen Transformations - oder Regleralgorithmen möglichst einfach, d. h. möglichst auf Softwareebene zu ändern sein. Aus diesem Grund scheidet der Einsatz spezieller Steuerungshardwarekomponenten wie z. B. Einschubkarten aus. Nicht nur in der Handhabungstechnik geht der Trend allgemein zu immer leichteren (Parallel-) Kinematiken, die wesentlich schneller beschleunigt werden können. Daher sind besonders mechanikschonende Interpolationsverfahren gefordert, die zugleich schnelle Bewegungen ermöglichen. Zusammenfassend lauten die Anforderungen an eine Positionierung:

- schnelle und ruckfreie Bewegungen
- einfache Änderung der Transformation
- speicherplatzsparende Datenhaltung und sicheres Timingverhalten

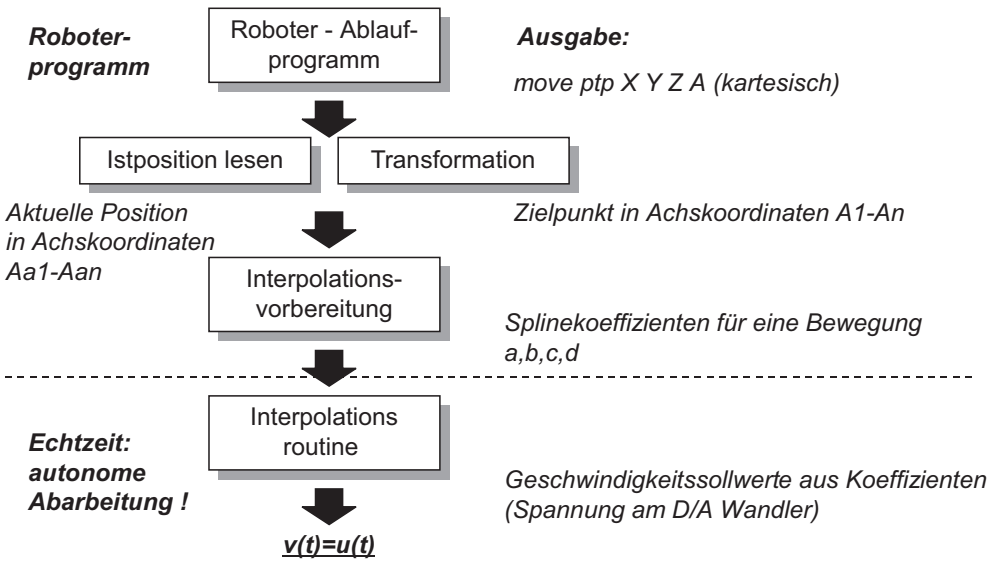


Abbildung 6.3: Roboterpositionierung mit Splines

Bisher eingesetzte Positionierungsverfahren arbeiten meist phasenorientiert, d. h. daß das Geschwindigkeitsprofil in Phasen der Beschleunigung und des Konstantfahrt aufgeteilt wird. Um den Ruck begrenzen, werden die Phasen aufintegriert. Es entsteht dann ein $\sin^2$ ähnliches Profil mit sieben Phasen, die alle im Interpolator erkannt und unterschiedlich behandelt werden müssen. Daraus resultiert eine ungünstige Datenhaltung.

Eine Lösung dieser Probleme ist der Einsatz von Splines Abb. 6.3. Das komplette Geschwindigkeitsprofil der Achsen soll möglichst durch eine Splinefunktion beschrieben werden, was

den Vorteil hat, daß nur noch die Splinekoeffizienten an die Interpolationsroutine übergeben werden müssen. Ferner kann die Interpolationsroutine (Echtzeitebene) nach Übergabe der Koeffizienten vollständig autonom arbeiten. Eventuelle Timingprobleme durch aufwendige Grobinterpolatoren, die über FIFO-Speicher arbeiten, entfallen.

In der Palettiertechnik liegt der Schwerpunkt in der schnellen Abarbeitung der Zyklen und nicht in der exakten Einhalten einer vorgegebenen Bahn wie bei Werkzeugmaschinensteuerungen. Daher werden die nachfolgend beschriebenen Splineverfahren zur Interpolation im Geschwindigkeitsprofil eingesetzt. Das Bahnprofil resultiert aus der Überlagerung der Wegprofile der einzelnen Roboterachsen und muß nur die Bedingung der Kollisionsfreiheit erfüllen.

6.3.1 PTP-Interpolation mit Spline-Überschleifen

Das PTP-Interpolationsverfahren benutzt einen Approximationsansatz von Paul [56], der eine Roboterbahn durch eine Approximation linearer Bahnstücke beschreibt.

Grundidee der PTP-Spline Interpolation ist die Umkehrung des Paul Ansatzes. Rechteckige Geschwindigkeitprofile werden so angepaßt, daß diese mit Splineprofilen zu einem zeitoptimalen Bahnverlauf führen. Abb. 6.4 zeigt die Weg-, Geschwindigkeits- und Beschleunigungsprofile der Grundelemente. Zwischen den n Stützstellen (Bahnpunkten) ist das Wegprofil abschnittsweise linear. Innerhalb der Sätze mit den Satzzeiten werden die Achsen mit den zugehörigen konstanten, synchronen Geschwindigkeiten verfahren. Das Wegprofil ist in dieser Form nicht zu realisieren, weil im Beschleunigungsprofil an den Stützstellen Dirac-Stöße auftreten. Die Lösung des Problems sind Überschleiffenster, die die Stützstellen umschließen. In diesen Fenstern wird mit Splinefunktionen ein beschleunigungsstetiger und ruckbegrenzter Übergang zwischen den unterschiedlichen Satzgeschwindigkeiten berechnet.

Berechnung der Satzzeiten und synchronen Geschwindigkeiten:

Die Satzzeiten haben zwei Kriterien zu erfüllen. Zum einen dürfen sich die Überschleiffenster nicht überschneiden. Deren Größe wird vorläufig so berechnet, daß in keinem Fall die maximale Beschleunigung überschritten wird. Sie sind so zu bemessen, daß innerhalb des Überschleifzeitraumes alle Achsen von der maximalen negativen Geschwindigkeit auf die maximale positive Geschwindigkeit beschleunigt werden können. Für den Überschleifzeitraum ergibt sich:

$$2v_{max_i} = a_{max_i}(2_{t_{\ddot{u}}}) \leftrightarrow t_{\ddot{u}_i} = \frac{v_{max_i}}{a_{max_i}} \tag{6.1}$$

$$\rightarrow t_{\ddot{u}} = \max_{i=1..4}\{t_{(\ddot{u}_i)}\} \tag{6.2}$$

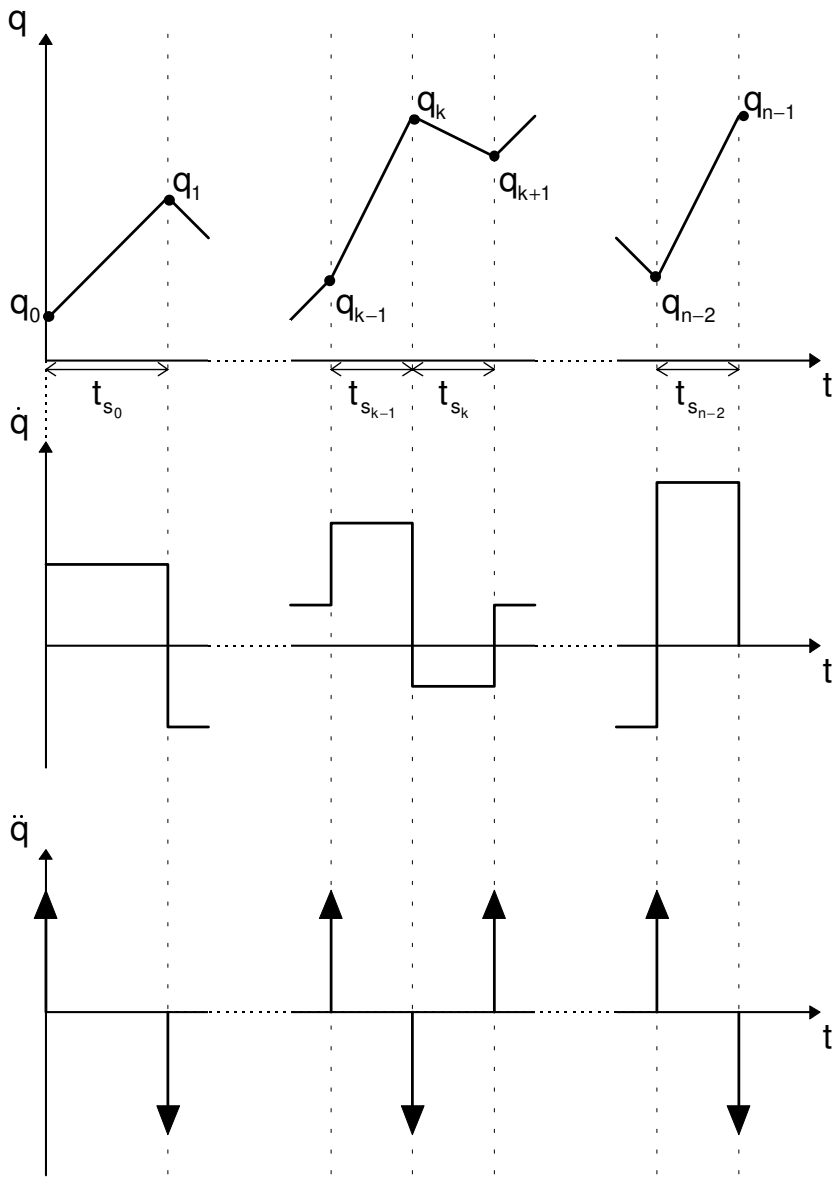


Abbildung 6.4: Weg-, Geschwindigkeits- und Beschleunigungsprofil

Die Überschleiffenster sind symmetrisch um die Stützstellen positioniert. Für die Berechnung der Größe der Fenster wird später ein Verfahren vorgestellt, das an die vorhandenen Geschwindigkeitsunterschiede zwischen den Sätzen angepaßt ist. Zum anderen sind die Satzzeiten so zu wählen, daß die maximalen Geschwindigkeiten nicht überschritten werden. Dafür wird die minimale Verfahrzeit zwischen den Stützstellen berechnet.

$$t_{k_i} = \frac{|q_{k+1_i} - q_{k_i}|}{v_{max_i}} \tag{6.3}$$

$$\rightarrow t_{v_k} = \max_{i=1..4}\{t_{k_i}\} \tag{6.4}$$

Die Satzzeiten ergeben sich folgendermaßen aus den beiden Kriterien:

$$\rightarrow t_{s_k} = max\{t_{v_k}, 2t_{\ddot{u}}\} \tag{6.5}$$

Für die synchronen Geschwindigkeiten gilt die Berechnungsvorschrift

$$v_{sync_{k,i}} = \frac{q_{k+1_i} - q_{k_i}}{t_{s_k}} \tag{6.6}$$

Berechnung der Konstantfahrtphasen:

Zwischen den Satzübergängen werden die Roboterachsen mit konstanter, synchroner Geschwindigkeit verfahren. Die diskreten Bahnpunkte ergeben sich in folgender Weise:

$$\begin{aligned} q_{neu} &= q_{alt} + v_{sync} T_{FIPO} \\ v_{neu} &= v_{sync} \\ a_{neu} &= 0 \\ r_{neu} &= 0 \end{aligned} \tag{6.7}$$

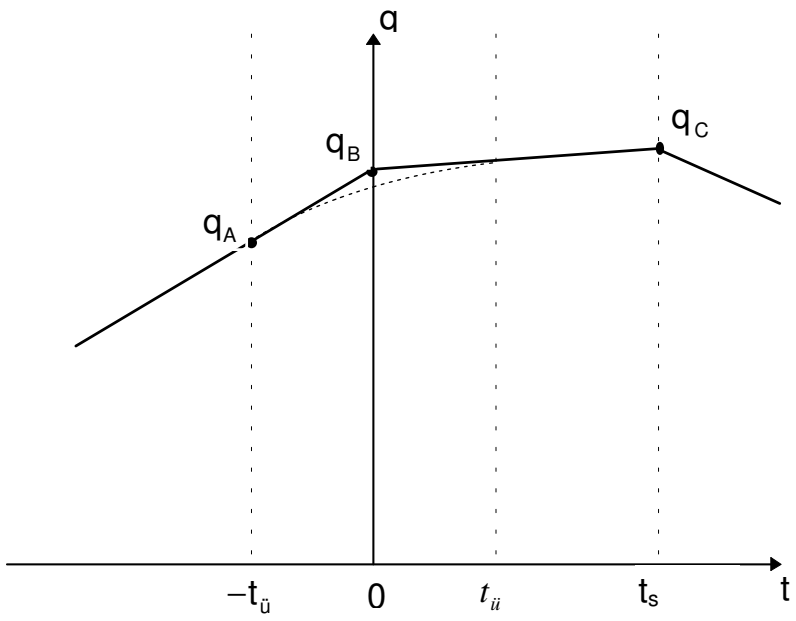
Berechnung der Satzübergänge: Alle Satzübergänge haben zunächst eine Zeitdauer von $2t_{\ddot{u}}$. Die Zeitachse des Satzübergangs ist symmetrisch zur Stützstelle. Zu Beginn des Überschleiffensters ist $t = -t_{\ddot{u}}$, am Ende ist $t = t_{\ddot{u}}$. Zuerst sollen anhand Abb. 6.5 die Bezeichnungen erläutert werden.

Das den Satzübergang beschreibende Polynom hat 6 Randbedingungen zu erfüllen:

$$\begin{aligned} q(-t_{\ddot{u}}) = q_a &= q_b - v_{sync_{alt}} t_{\ddot{u}} \\ q(t_{\ddot{u}}) = q_b + q_{cb}\frac{t_{\ddot{u}}}{t_s} &= q_b + v_{sync_{alt}} t_{\ddot{u}} \\ \dot{q}(-t_{\ddot{u}}) = -\frac{q_{ab}}{t_s} &= v_{synch_{alt}} \\ \dot{q}(t_{\ddot{u}}) = -\frac{q_{cb}}{t_s} &= v_{synch_{neu}} \\ \ddot{q}(-t_{\ddot{u}}) &= 0 \\ \ddot{q}(t_{\ddot{u}}) &= 0 \end{aligned} \tag{6.8}$$

Die Randgeschwindigkeiten entsprechen den synchronen Geschwindigkeiten der benachbarten Sätze. Als Ansatz ist ein Polynom 5. Grades zu wählen.

$$\begin{aligned} q(t) &= B_0 + B_1 t + B_2 t^2 + B_3 t^3 + B_4 t^4 + B_5 t^5 \\ \dot{q}(t) &= B_1 + 2B_2 t + 3B_3 t^2 + 4B_4 t^3 + 5B_5 t^4 \\ \ddot{q}(t) &= 2B_2 + 6B_3 t + 12B_4 t^2 + 20B_5 t^3 \end{aligned} \tag{6.9}$$



q_B : Stützstelle, die überschliffen werden soll
q_C : folgende Stützstelle
q_A : Ort zu Beginn des Satzüberganges
$q_{AB} = q_A - q_B$
$q_{CB} = q_C - q_B$
t_s : Satzlänge des folgenden Satzes

Abbildung 6.5: Satzübergang an der Stützstelle

Nach dem Auswerten der Randbedingungen ergeben sich die Polynomkoeffizienten folgendermaßen:

$$\begin{aligned} B_0 &= q_b + \frac{3}{15}(q_{cb}\frac{t_{\ddot{u}}}{t_s} + q_{ab}) \\ B_1 &= \frac{1}{2t_{\ddot{u}}}(q_{cb}\frac{t_{\ddot{u}}}{t_s} - q_{ab}) \\ B_2 &= \frac{3}{8{t_{\ddot{u}}}^2}(q_{cb}\frac{t_{\ddot{u}}}{t_s} + q_{ab}) \\ B_3 &= 0 \\ B_4 &= \frac{-1}{16{t_{\ddot{u}}}^4}(q_{cb}\frac{t_{\ddot{u}}}{t_s} + q_{ab}) \\ B_5 &= 0 \end{aligned} \tag{6.10}$$

Aufgrund des symmetrischen Aufbaus der Überschleiffenster sind die Koeffizienten $B_3 =$

$B_5 = 0$. Das berechnete Polynom hat die folgende Form:

$$\begin{aligned} q(t) &= B_0 + B_1 t + B_2 t^2 + B_4 t^4 \\ \dot{q}(t) &= B_1 + 2B_2 t + 4B_4 t^3 \\ \ddot{q}(t) &= 2B_2 + 12B_4 t^2 \end{aligned} \tag{6.11}$$

Die Stützstelle wird nicht genau angefahren. Der Bahnverlauf wird während des Satzübergangs vorausschauend auf die folgende Stützstelle geplant.

Start- und Endphase als Spezialfall der Satzübergänge:

Vom Startpunkt ist aus der Ruhelage bis zur synchronen Geschwindigkeit des ersten Satzes zu beschleunigen, zum Endpunkt aus der synchronen Geschwindigkeit des letzten Satzes bis zum Stillstand abzubremsen. Diese beiden Bewegungsphasen können aus dem Berechnungsverfahren für die Satzübergänge abgeleitet werden. Für die Startphase gilt

$$\begin{aligned} q_a = q_b = 0 \rightarrow q_a b = 0 \\ q_c = q_1 \end{aligned} \tag{6.12}$$

und für die Endphase

$$\begin{aligned} q_a &= q_{n-1} - \frac{q_{n-1} - q_{n-2}}{t_{s_{n-2}}} \\ q_b = q_c = q_{n-1} &\rightarrow q_{cb} = 0 \end{aligned} \tag{6.13}$$

Die Gesamtbewegungsdauer vergrößert sich aufgrund der Start- und Endphase jeweils um $t_{\ddot{u}}$.

Modifizierte Berechnung der Überschleifzeiträume:

Bisher haben alle Überschleiffenster die gleiche Größe. Für die Berechnung dieser Größe wird davon ausgegangen, daß im Überschleifzeitraum der größtmögliche Geschwindigkeitsunterschied zwischen den Sätzen zu überwinden ist. Die Stützstellen sind jedoch selten so angeordnet, daß diese großen Geschwindigkeitsunterschiede an den Satzübergängen auftreten. Der überwiegende Teil der Überschleiffenster ist überdimensioniert. Dadurch wird die Bewegungsdauer unnötig verlängert, weil die Satzübergangszeiten maßgeblich die Satzzeiten bestimmen. Abhilfe schafft hier eine genauere Betrachtung der Satzübergangspolynome:

$$\begin{aligned} q(t) &= B_0 + B_1 t + B_2 t^2 + B_4 t^4 \\ \dot{q}(t) &= B_1 + 2B_2 t + 4B_4 t^3 \\ \ddot{q}(t) &= 2B_2 + 12B_4 t^2 \end{aligned} \tag{6.14}$$

Weil die Koeffizienten $B_3 = B_5 = 0$ sind, ergibt sich ein lineares Ruckprofil mit Nulldurchgang bei $t = 0$. Das Beschleunigungsprofil ist parabelförmig mit dem Scheitelpunkt $t = 0$ und den Nulldurchgängen $t = -t_{\ddot{u}}$ und $t = t_{\ddot{u}}$. Das kubische Geschwindigkeitsprofil erreicht seine maximale Steigung also bei $t = 0$ und schließt an den Rändern des Überschleifbereiches mit der Steigung $dv/dt = \ddot{q} = 0$ an die synchronen Geschwindigkeiten der Konstantfahrtphasen an. Die Profile sind in Abb. 6.6 dargestellt.

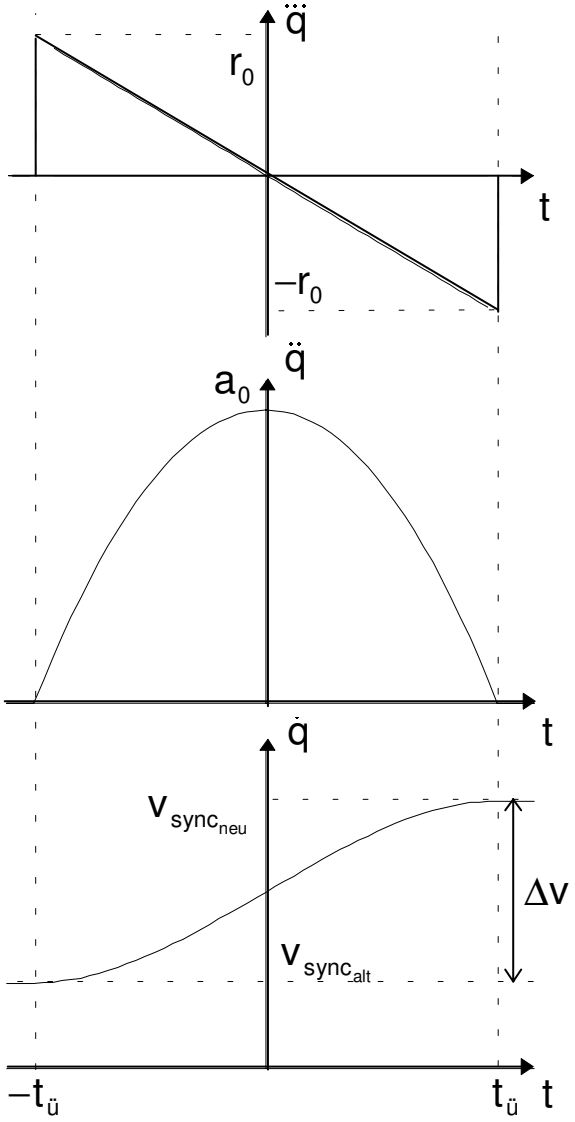


Abbildung 6.6: Ruck-, Beschleunigungs- und Geschwindigkeitsverlauf im Satzübergang

Während des Überwindens des Geschwindigkeitsunterschieds Δv tritt die maximale Beschleunigung a_0 und der maximale Ruck r_0 auf. Diese beiden Werte sind zu berechnen.

$$a_0 = \ddot{q}(t = 0) = 2B_2 = \frac{3}{4{t_{\ddot{u}}}^2}(q_{cb}\frac{t_{\ddot{u}}}{t_s} + q_{ab}) = \frac{3}{4t_{\ddot{u}}}(v_{synch_{neu}} - v_{synch_{alt}}) \tag{6.15}$$

$$r_0 = \frac{\partial^3 q}{\partial t}(t = -t_{\ddot{u}}) = -24B_4 t_{\ddot{u}} = \frac{3}{2{t_{\ddot{u}}}^3}(q_{cb}\frac{t_{\ddot{u}}}{t_s} + q_{ab}) = \frac{3}{2{t_{\ddot{u}}}^2}(v_{synch_{neu}} - v_{synch_{alt}}) \quad (6.16)$$

Bis auf die Satzübergangszeit $2t_{\ddot{u}}$ sind alle Größen bekannt. Für die maximal auftretende Beschleunigung a_0 und den maximal auftretenden Ruck r_0 gelten die folgenden Einschränkungen:

$$|a_0| \leq a_{max} \quad (6.17)$$

$$|r_0| \leq r_{max} \quad (6.18)$$

Für die Zeit $t_{\ddot{u}}$ ergeben sich zwei Berechnungsvorschriften.

$$t_{\ddot{u}} = \frac{3|v_{synch_{neu}} - v_{synch_{alt}}|}{4a_{max}} \quad (6.19)$$

$$t_{\ddot{u}} = \sqrt{\frac{3|v_{synch_{neu}} - v_{synch_{alt}}|}{2r_{max}}} \quad (6.20)$$

Diese Gleichungen sind für alle Achsen zu berechnen. Als Endergebnis für die halbe Satzübergangszeit $t_{\ddot{u}}$ ist der größte Wert zu wählen, damit die maximalen Beschleunigungen und Rucke aller Achsen eingehalten werden. Auf diese Weise ist es möglich, jeden Satzübergang individuell zu berechnen. Die Satzzeiten werden iterativ ermittelt. Zu Beginn werden sie einmal mit Hilfe der Stützstellenabstände berechnet. In jeder Iterationsschleife werden die synchronen Geschwindigkeiten und die Satzübergangszeiten neu errechnet. Die Satzzeiten und Satzübergangszeiten werden auf eventuelle Überschneidung der Überschleiffenster geprüft. Im Falle einer Überschneidung werden die entsprechenden Satzzeiten inkrementiert. Die Schleife wird solange durchlaufen, bis keine Überschneidungen mehr auftreten. Einhalten einer vorgegebenen Überschleifrate: Des weiteren kann in die Iterationsschleife eine Überprüfung der Überschleifrate integriert werden. Die Überschleifrate ü ist ein Maß für die maximale Größe der Überschleiffenster. So ist es sinnvoll, während des senkrechten Hochfahrens vom Zuführtisch sicherzustellen, daß der Roboter seine Bewegungsrichtung über einen definiertes Teilstück beibehält und nicht in eine Überschleifbewegung eintritt. Dazu sind folgende Überlegungen notwendig. Die Überschleifrate gibt das Verhältnis vom Überschleifbereich zum halben Stützstellenabstand wieder:

$$|q_k - q_a| = \ddot{u}\frac{q_k - q_{k-1}}{2} \quad (6.21)$$

wobei q_k der Stützstelle entspricht, die überschliffen wird, q_{k-1} die vorherige Stützstelle ist und q_a den Startpunkt des Überschleifens darstellt. Theoretisch ist es möglich, Überschleifraten $0 < \ddot{u} \leq 2$ vorzugeben. $\ddot{u} = 0$ würde bedeuten, daß die Stützstelle genau angefahren wird.

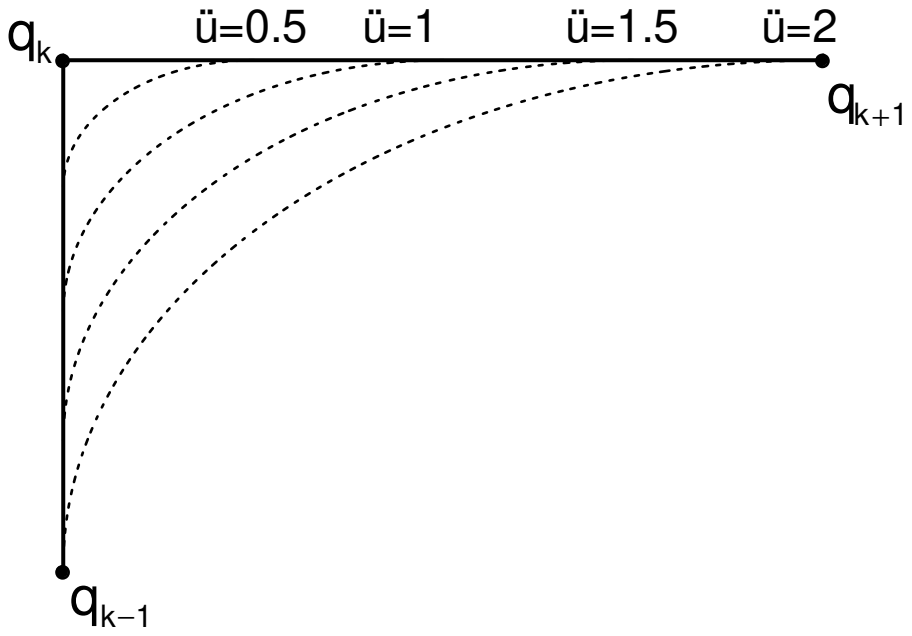


Abbildung 6.7: Unterschiedliche Überschleifraten

Für $\ddot{u} = 1$ hat das Überschleiffenster maximal die Größe des halben Stützstellenabstandes und für $\ddot{u} = 2$ wird der Satzübergang nicht durch die Überschleifrate beeinflußt.

Von praktischer Bedeutung ist ein Wertebereich $0.5 < \ddot{u} \leq 2$. Durch die Vorgabe zu kleiner Überschleifraten steigt die Bewegungsdauer überproportional an. Die Überschleifrate ist auch mit den synchronen Geschwindigkeiten zu berechnen:

$$|v_{sync_{k-1}}|t_{\ddot{u}} = \ddot{u}\frac{|v_{sync_{k-1}}|t_{s_{k-1}}}{2} \rightarrow \ddot{u} = \frac{t_{\ddot{u}}}{\frac{t_{s_{k-1}}}{2}} \tag{6.22}$$

Die Überschleifrate ü entspricht also dem Verhältnis der halben Satzübergangszeit $t_{\ddot{u}}$ zur halben Satzzeit $\frac{t_{sk-1}}{2}$. Die Überschleifrate ist in den beiden angrenzenden Sätzen des Satzüberganges einzuhalten. Für die Größe der Überschleiffenster ergeben sich die folgenden Forderungen:

$$t_{\ddot{u}} \leq \ddot{u}\frac{t_{s_{k-1}}}{2} \tag{6.23}$$

und

$$t_{\ddot{u}} \leq \ddot{u}\frac{t_{s_k}}{2} \tag{6.24}$$

Wird eine dieser Bedingungen nicht erfüllt, ist die entsprechende Satzzeit zu inkrementieren. Dadurch gleichen sich die synchronen Geschwindigkeiten aufeinanderfolgender Sätze einander an und die Satzübergangszeiten werden kürzer.

Gesamtbewegungsdauer:

Die Gesamtbewegungsdauer ergibt sich aus der Summe aller Satzzeiten zuzüglich der halben Satzübergangszeiten der Start- und Endphase.

$$t_{ges} = t_{\ddot{u}_0} + t_{s_0} + t_{s_1} + .. + ..t_{s_{n-2}} + t_{\ddot{u}_{n-1}} \tag{6.25}$$

6.3.2 Ergebnisse des PTP-Spline Verfahren

Zur Validierung des Verfahrens wurde ein Zyklus einer Palettierung simuliert werden und die Positionierungszeiten berechnet. Dabei wird der Transport eines Paketes vom Zuführtisch zur Palette betrachtet und eine Überschleifrate ü=1 gewählt. Der erste Zwischenpunkt wird 200 mm senkrecht über der Zuführung gesetzt. Dadurch ist sichergestellt, daß sich der Roboter bei der gewählten Überschleifrate mindestens 100 mm senkrecht nach oben bewegt. Der zweite Zwischenpunkt befindet sich auf gleicher Höhe wie der erste senkrecht über der Einfügeposition, um eine Kollision des Roboters und Paketes mit dem Zuführtisch zu vermeiden. Der dritte Zwischenpunkt auch Einfügepunkt genannt ist schräg versetzt über dem Ablagepunkt. Er ist notwendig, um das Paket ohne Kollision mit anderen Paketen auf der Palette abzulegen. Für die Ablage eines Paketes werden somit 5 Stützstellen (Start- und Endpunkt sowie drei Zwischenpunkte) vorgegeben. Für deren Berechnung werden benötigt:

- Koordinaten des Paketes auf dem Zuführtisch,
- Ablageort auf der Palette,
- Einfügerichtung des Paketes,
- aktuelle Palettenhöhe,
- Höhe der Zuführung

Start- und Endpunkt entsprechen der Startposition auf dem Zuführtisch und dem Ablageort auf der Palette. Die Höhe der ersten beiden Zwischenpunkte ist so zu wählen, daß sie größer als die Höhe der Zuführung und der aktuellen Palettenhöhe ist. Der letzte Zwischenpunkt ergibt sich aus der vorgegebenen Einfügerichtung, damit das Paket nicht mit bereits abgelegten Paketen kollidiert. Für die Rückkehr des Greifers von der Paketablage zur Zuführung werden 4 Stützstellen vorgegeben - die Einfügebewegung zur Ablage des Paketes auf die Palette entfällt.

Bei einem Interpolationstakt $T_{FIPO} = 8ms$ ergeben sich für dieses Beispiel 309 Bahnpunkte und somit eine Bewegungsdauer $t_{ges} = 2,464s$. In Abb. 6.8 und Abb. 6.9 sind der kartesische Bahnverlauf sowie Geschwindigkeits- und Beschleunigungsprofile dargestellt.

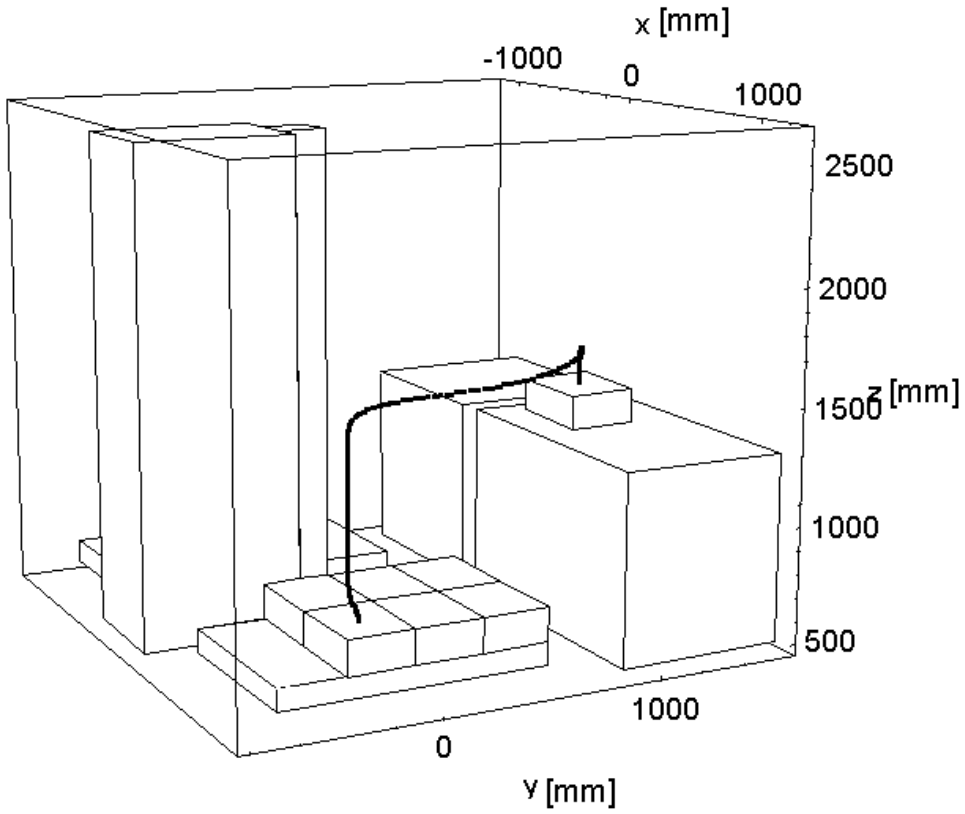


Abbildung 6.8: Bahnverlauf in kartesischen Koordinaten

Durch die primäre Planung der Sätze mit der PTP-Interpolation ist das schnellstmögliche Verfahren zwischen allen Stützstellen gewährleistet. Die Bahnberechnung mit Hilfe der Spline-Interpolation in den Satzübergängen stellt den beschleunigungsstetigen und ruckbegrenzten Bahnverlauf sicher. Aufgrund dieser mechanikschonenden Verfahrart ist es möglich, größere Maximalwerte für Geschwindigkeit, Beschleunigung und Ruck zuzulassen. Für eine 30 %-ige Erhöhung dieser Werte ergeben sich für das vorangegangene Positionierungsbeispiel 259 Bahnpunkte und eine Bewegungsdauer $t_{ges} = 2,064s$. Dies ist eine Zeitersparnis von $\Delta t = 0,4s$. Der Bahnverlauf variiert geringfügig in den Überschleifbereichen, die Überschleifrate wird aber eingehalten. Dies ist ein großer Vorteil der PTP-Interpolation mit Spline-Überschleifen. Durch die Vorgabe der Stützstellen und der zulässigen Überschleifrate ist der Bahnverlauf in engen Grenzen definiert. Da ein Überschwingen kann nicht auftreten, kann auf eine explizite Kollisionsüberprüfung der berechneten Bahn bei geeigneter Stützstellenvorgabe und korrekter Überschleifrate verzichtet werden.

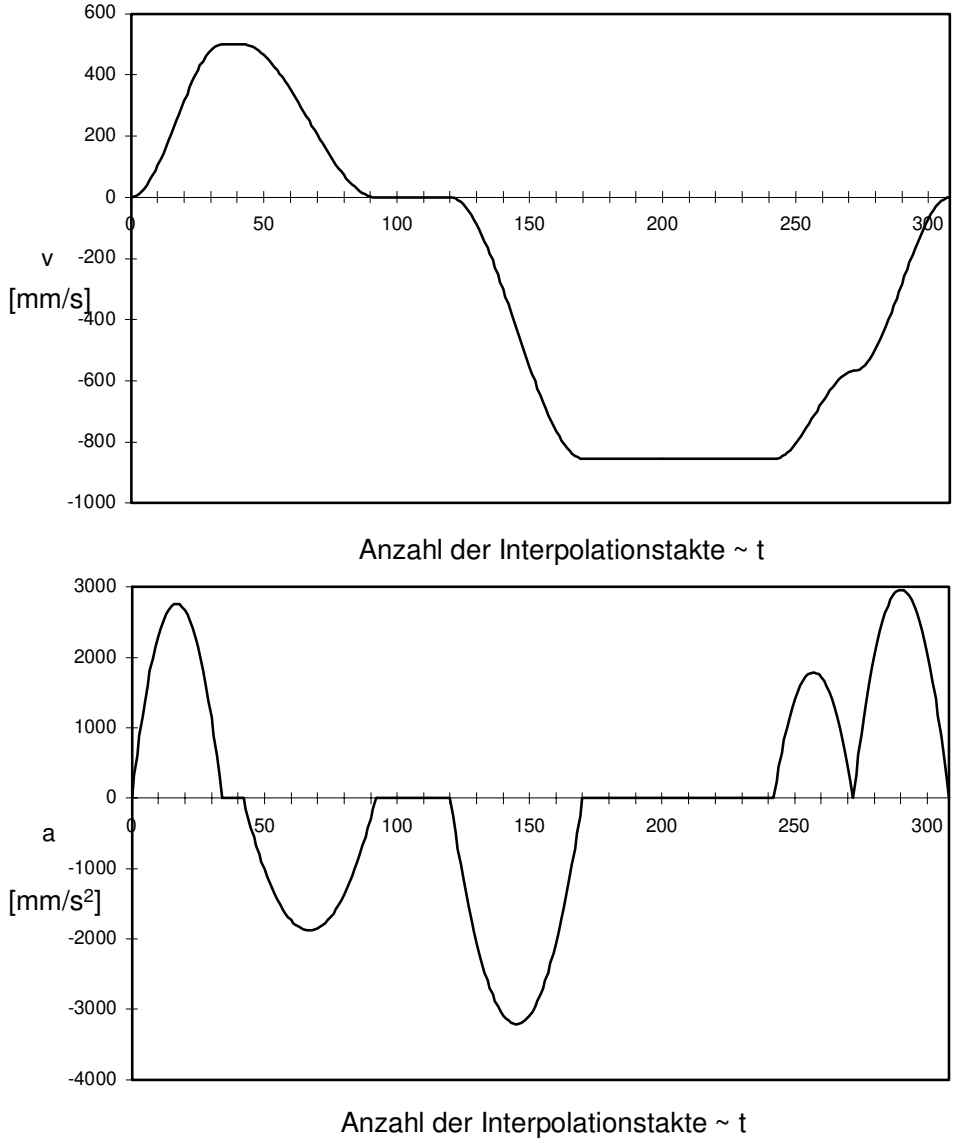


Abbildung 6.9: Geschwindigkeits- und Beschleunigungsprofile der z-Achse

6.3.3 PTP-Spline im Vergleich mit anderen Verfahren

Folgende Verfahren werden verglichen und bewertet:

- gefilterte und ungefilterte PTP Interpolation
- Spline Interpolation nach Schwinn [54]
- PTP Spline

Die synchrone PTP-Interpolation (siehe Stand der Technik) stellt die schnellstmögliche Verfahrart zwischen zwei Punkten dar. Sie hat den Nachteil, daß erst durch eine anschließende Filterung ein beschleunigungsstetiger und ruckbegrenzter Bahnverlauf erzeugt wird. Wesentlicher Nachteil ist die eingeschränkte Anwendbarkeit, weil lediglich zwei Stützstellen vorgegeben werden, zwischen denen die Bahn berechnet wird. Wenn Zwischenpunkte zur Kollisionsvermeidung notwendig sind, ist für jeden Satz eine eigene Bahn zu berechnen. Der Roboter bremst dabei an jedem Zwischenpunkt bis zum Stillstand ab. Dies verstärkt den Verschleiß an der Robotermechanik. Das Problem ist mit Überschleifverfahren an den Zwischenpunkten zu beheben, die eine vorausschauende Bahnplanung auf den nächsten Zwischenpunkt ermöglichen.

Mit der Spline-Interpolation ist dagegen eine geschlossene Bahnplanung über mehrere Stützstellen möglich. Alle Zwischenpunkte werden dabei genau angefahren, aber der Roboter bremst nicht bis zum Stillstand ab. Die berechnete Bahn hat einen beschleunigungsstetigen und ruckbegrenzten Verlauf. Ein Nachteil dieses Verfahrens ist die schlechte Ausnutzung des Leistungsvermögens. Während des gesamten Bewegungsablaufes erreicht eine Roboterachse nur einmal ihre maximale Geschwindigkeit oder Beschleunigung. Dies widerspricht einer Bahnplanung mit einer minimalen Verfahrzeit. Des weiteren ist die Auswahl der Stützstellen kritisch. Es ist möglich, daß aufgrund von Überschwingerscheinungen zwischen zwei Stützstellen eine Bahn berechnet wird, die außerhalb der Achsverfahrbereiche liegt. Dieses Überschwingen erschwert das Planen einer kollisionsfreien Bahn.

Die PTP-Interpolation mit Spline-Überschleifen verknüpft die Vorteile der beiden anderen Verfahren miteinander. Sie ermöglicht eine geschlossene Bahnplanung basierend auf der synchronen PTP-Interpolation. Dadurch ist eine minimale Verfahrzeit sichergestellt. Das Spline-Überschleifen erzeugt beschleunigungsstetige und ruckbegrenzte Satzübergänge an den Stützstellen. Die Zwischenpunkte werden dabei nicht genau angefahren, wobei die Größe der Überschleifbereiche durch die vorgebbare Überschleifrate eingegrenzt ist. In Tabelle 6.1 ist eine Übersicht der Charakteristika der untersuchten Interpolationsarten zusammengestellt:

In Tabelle 6.2 werden die Interpolationsarten anhand von Simulation von einigen Positionierungsbeispielen verglichen. In der Simulation wird die z-Achse um 1000 mm verfahren, die anderen Achsen stehen still. Bei der zweiten Positionierung wurde zusätzlich ein Zwischenpunkt auf der Hälfte der Distanz eingefügt. Simulation drei beschreibt die Bahn vom Abholen eines Paketes vom Zuführtisch bis zur Ablage auf der Palette. Abb. 6.11 zeigt die resultierenden Bahnen der dritten Simulation.

Eigenschaften	PTP gefiltert	PTP ungefiltert	Spline	PTP-Spline
Stetigkeit der	Geschwindigkeit	Beschleunigung	Beschleunigung	Beschleunigung
Ruckbegrenzung	nein	ja	ja	ja
Überschwingen	nein	nein	ja	nein
Einfluß der Stützstellen-zahl auf die Bewegungsdauer	groß	groß	groß	gering
Rechenaufwand	gering	gering	groß	gering

Tabelle 6.1: Allgemeiner Vergleich der Interpolationsarten

Simulation	PTP ungefiltert	PTP gefiltert	Spline	PTP-Spline
z-Achse 1000 mm verfahren	1,368 s	1,472 s	2,176	1,52
zusätzlicher Zwischenpunkt bei 500 mm	1,568 s	1,776 s	2,328 s	1,52 s
Ablage eines Paketes auf der Palette	2,808 s	3,016 s	2,464 s	2,464 s

Tabelle 6.2: Simulation der Positionierungszeiten im Vergleich

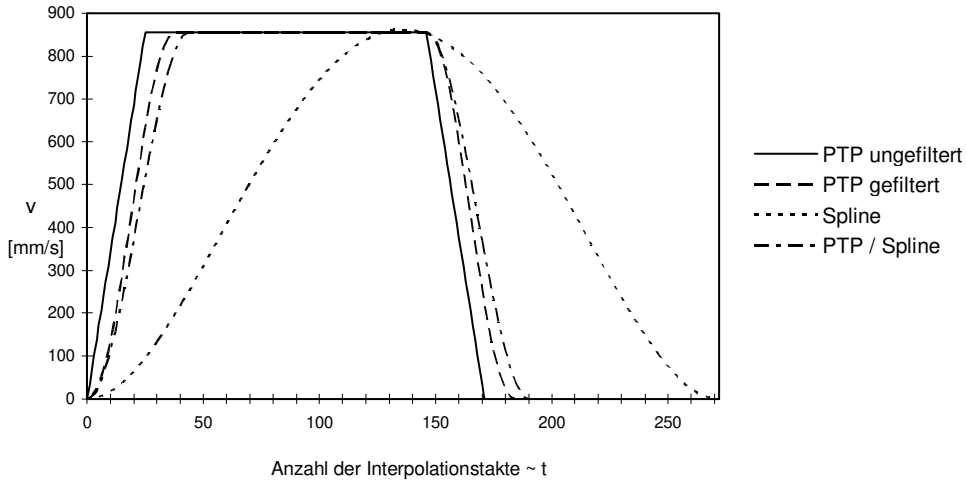


Abbildung 6.10: Geschwindigkeitsprofile beim Verfahren der z-Achse

Die synchrone PTP-Interpolation berechnet theoretisch die schnellste Bahn. In der Praxis kommt es jedoch bereits bei geringen Geschwindigkeiten zu einem unruhigen Lauf durch Beschleunigungsstöße (Dirac). Die Spline-Interpolation, deren Ansatz vollständig auf Polynomfunktionen basiert, erzeugt eine Bahn mit einer erheblich größeren Bewegungsdauer. Die PTP-Interpolation mit Spline-Überschleifen, bei der nur die Satzübergänge mit Polynomansätzen berechnet werden, liefert eine der synchronen PTP-Interpolation ähnliche Bahn. Der große Vorteil der PTP-Interpolation mit Spline-Überschleifen wird deutlich, wenn wie im zweiten Versuch ein Zwischenpunkt eingefügt wird. Die Verfahrzeit ändert sich nicht, weil der Zwischenpunkt sich auf der Strecke zwischen Start- und Endpunkt befindet. Die synchrone PTP-Interpolation berechnet eine größere Bewegungsdauer, da der Roboter am Zwischenpunkt bis zum Stillstand abbremst. Auch für die Spline-Interpolation vergrößert sich die Verfahrzeit, obwohl der Zwischenpunkt in jedem Fall angefahren wird.

Die typischen Merkmale der drei Interpolationsarten werden in Abb. 6.11 noch einmal verdeutlicht. Die PTP-Spline-Interpolation und die synchrone PTP-Interpolation haben annähernd die gleiche Bahn. Während bei letzterer der Roboter an den Zwischenpunkten bis zum Stillstand abbremst und diese genau anfährt, werden bei der ersten Interpolationsart die Zwischenpunkte überschliffen. Für die Spline-Interpolation ist ein Zwischenpunkt weniger vorgegeben. Der berechnete Weg ist räumlich kürzer als bei den beiden anderen Interpolationsarten. Trotzdem ergibt sich die gleiche Verfahrzeit wie für die PTP-Interpolation mit Spline-Überschleifen, da das Leistungsvermögen des Roboters von der Spline-Interpolation nicht ausgenutzt wird.

Abschließend ist festzustellen, daß die PTP-Interpolation mit Spline-Überschleifen ein sehr geeignetes Bahnplanungsverfahren für einen Palettierroboter darstellt. Durch die Kombi-

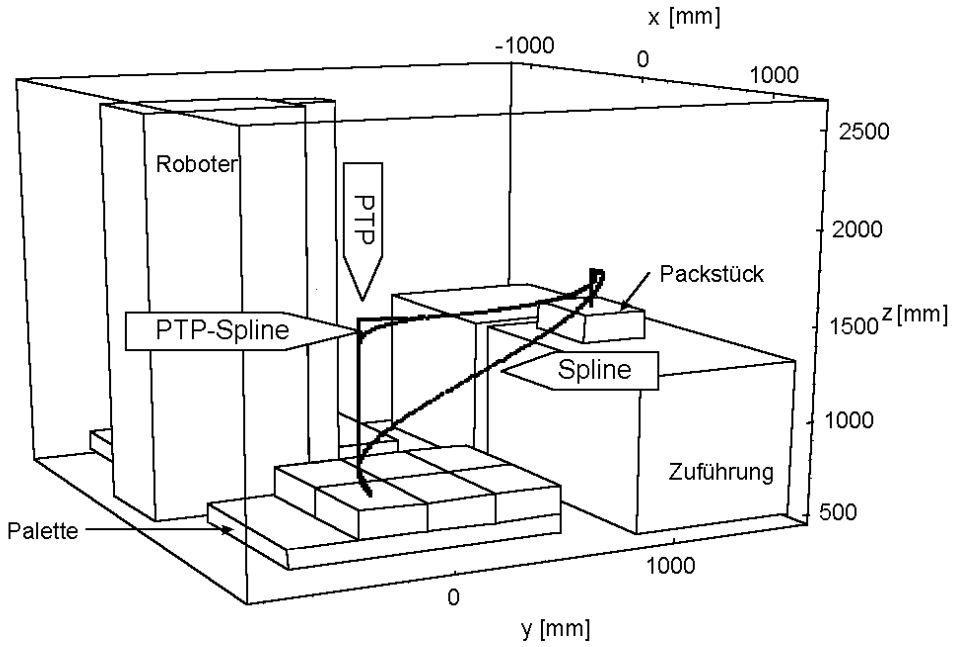


Abbildung 6.11: Darstellung der Bahnverläufe in kartesischen Koordinaten

nation der synchronen PTP-Interpolation als schnellstmögliche Bewegungsform mit dem Spline-Überschleifen zum Erzeugen eines zweimal stetig differenzierbaren Bahnverlaufes erhält man ein beschleunigungsstetiges und ruckbegrenztes Bahnprofil mit einer minimal möglichen Verfahrzeit. Durch eine geeignete Vorgabe der Stützstellen und der Überschleifrate ist es möglich, auf eine nachträgliche Kollisionsüberprüfung des berechneten Bahnprofils zu verzichten, weil durch die Vorgaben der Bahnverlauf in engen Grenzen festgelegt ist. Daher ist es denkbar, die Bahnprofilberechnung parallel zur Bewegung des Roboters auszuführen. Sobald der iterative Teil der Berechnung abgeschlossen ist, beginnt der Roboter mit der Bewegung. Die berechneten Splinekoeffiezienten und die Daten für die Konstantfahrtphase werden in ein Feld geschrieben und an die Positionierungsroutine übergeben. Die Positionierungsroutine (Interruptroutine) verarbeitet die Daten vollständig autonom, so daß quasi gleichzeitig zur aktuellen Bewegung ein neuer Bewegungssatz vorbereitet werden kann. Details sind allerdings vom verwendeten Betriebssystem abhängig.

Kapitel 7

Anwendung und Implementierung

Im Rahmen dieser Arbeit wurden zwei Systeme zur automatisierten Palettierung entwickelt:

- Zweidimensionales off-line Palettiersystem mit Robotersteuerung unter DOS[1]:

  Die automatische Zyklusbildung basiert auf einem Packstückgraphverfahren (siehe Kapitel Off-line Programmierung) , das in einer bestimmten Suchtiefe das lokale Optimum aufeinanderfolgender Packstücke durch vollständige Suche ermittelt. Dieses System wurde in der Industrie bereits mehrfach eingesetzt.

- Dreidimensionales Palettiersystem mit Robotersteuerung unter Windows NT:

  Die Zyklusbildung erfolgt hier mit Hilfe des Blockgraphenverfahrens. Die Robotersteuerung wurde als SPS Task unter Windows NT realisiert. Das off-line Programmiersystem kann aufgrund der Windows Plattform auch on-line benutzt werden. Das Gesamtsystem befindet sich noch im Stadium eines Prototypen.

Nachfolgend werden der Aufbau der Versuchsanlage und beide Systeme im Detail vorgestellt.

7.1 Versuchseinrichtung

Im Versuchsfeld wurde eine Palettieranlage (Abb. 7.1,Abb. 7.2) mit SCARA-Roboter, die ursprünglich von einer Hardware SPS , Robotersteuerung und einem Bedienpult gesteuert wurde, auf eine PC basierte Steuerung umgerüstet. Servomodule und Inkrementalgeber wurden auch weiterhin genutzt Abb. 7.3. Die Anlagenhardware wurde zu Positionier- und Palettierversuchen sowohl für die DOS-Lösung als auch für die Windows-Lösung benutzt.

[1] disk operating system

Mechanisch besteht die Anlage aus zwei Zuführungen und zwei Abgaben. Die erste Zuführung ist für ein Packstück ausgelegt, die zweite für maximal vier Packstücke in y-Richtung. Optional können die Packstücke an der zweiten Zuführung auch gekippt werden. Der Greifer am Roboter kann bis zu vier Packstücke mit jeweils einer Saugleiste greifen. Jede Zuführung ist weiterhin an Förderbänder angeschlossen.

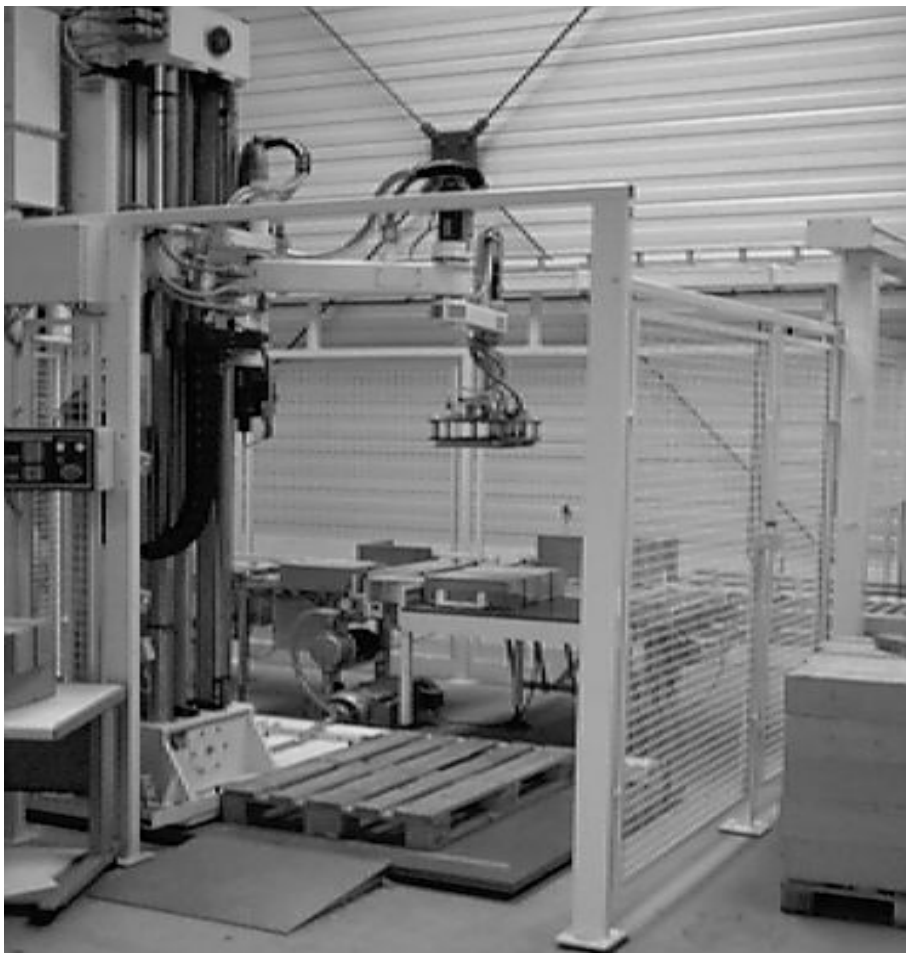


Abbildung 7.1: Palettieranlage im Versuchsfeld

Die neue Steuerungshardware (Firma Beckhoff) für die Palettieranlage besteht aus

- einem IPC mit Pentium 133 und 80 MB RAM
- einer Lichtleiterfeldbuskarte
- ein Lichtleiterfeldbus (Ring)
- sechs Digital IO[2] Boxen

[2]input output

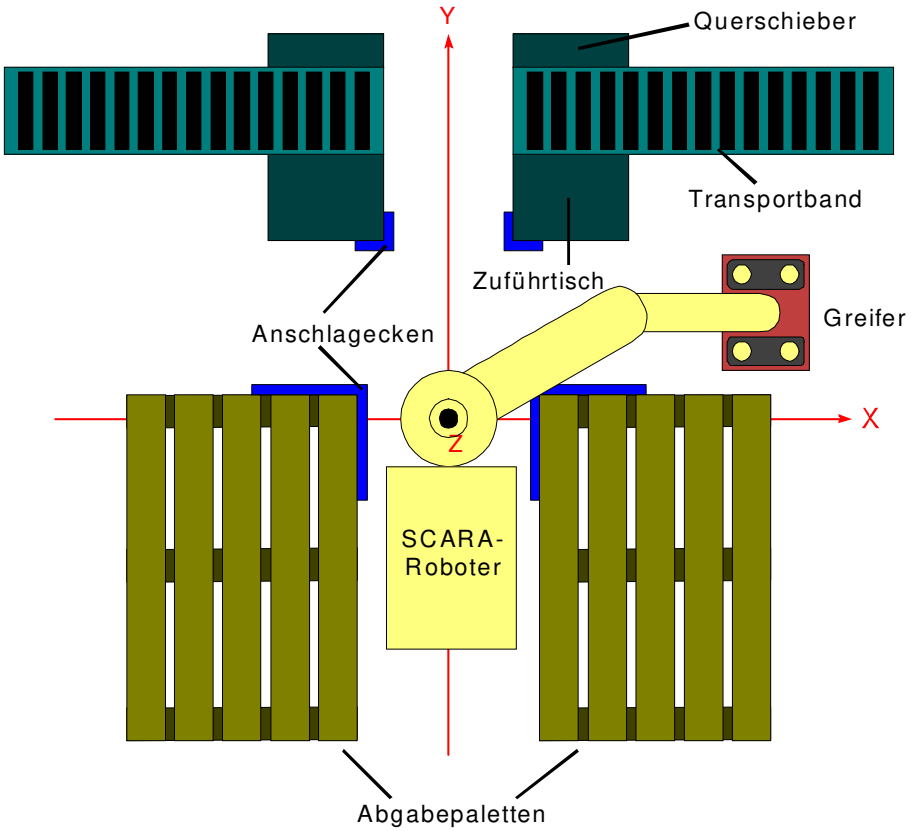


Abbildung 7.2: Systembild der Versuchsanlage

- eine Digital Analog Wandler Box für vier Achsen
- eine Inkrementalgeber Box für vier Geber.

Alle Steuerungsmodule sind in einem Lichtleiterring angeschlossen und werden von der jeweiligen Software SPS angesteuert. Der Lichtleiterring ist so verdrahtet, daß sowohl die alte als auch die neue Steuerung durch Umstecken aktiviert werden kann.

7.2 Zweidimensionales Palettiersystem

Zur Palettierung eines Packmusters sind hier folgende Schritte notwendig:

1. Konfiguration der Anlage. Alle Anlagendaten werden inklusive der Roboterdaten eingegeben. Nachfolgend kann die Anlagenkonfiguration durch die Simulation der Roboterbahn für die Anfahrpunkte der Zuführungen und der Abgaben inklusive der Einfügepunkte getestet werden (Abb. 7.5).
2. Graphische oder automatische Erzeugung eines Packmusters (Abb. 7.6).

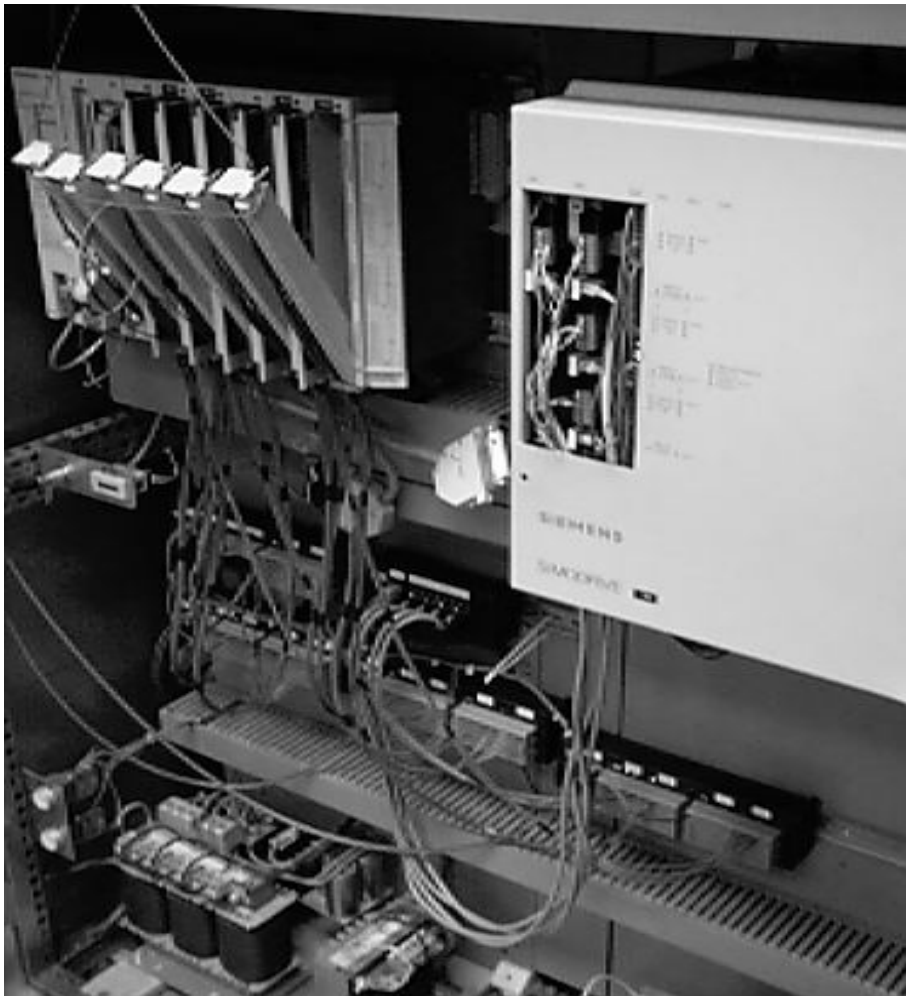


Abbildung 7.3: Umrüstung der Palettieranlage auf PC mit Lichtleiterfeldbus

3. Packprogramm aus mehreren Packmustern erzeugen (Abb. 7.7) und speichern.

4. Einlesen der Packprogramme und Anwahl an der Steuerung (Abb. 7.8).

Das System unterscheidet zwischen den Benutzergruppen Techniker und Kunde bzw. Operator. Der Techniker konfiguriert das System zunächst einmalig anhand kundenspezifischer Daten. Die wesentlichen Einstellungen sind

- Schutzräume $(x_{min}, y_{min}, x_{max}, y_{max})$,
- Robotertyp und Maschinendaten inklusive Verfahrbereiche (siehe Abb. 7.4),
- Zuführungen (Anschlagecke (X, Y, Z, A), Schutzraum $x_{min}, y_{min}, x_{max}, y_{max}$, Kippinformation, Lichtschrankenpositionen, Zusatzinformation (z. B. Armdurchbiegung)), Auffüllrichtung

- Abgaben (Anschlagecke(X, Y, Z, A), Zusatzinformation (z. B. Armdurchbiegung), Auffüllrichtung)
- Greifer (Typ, Werkzeugkorrektur).

Abb. 7.4 zeigt beispielhaft den Dialog zur Konfiguration der Zuführungen. Der Operator

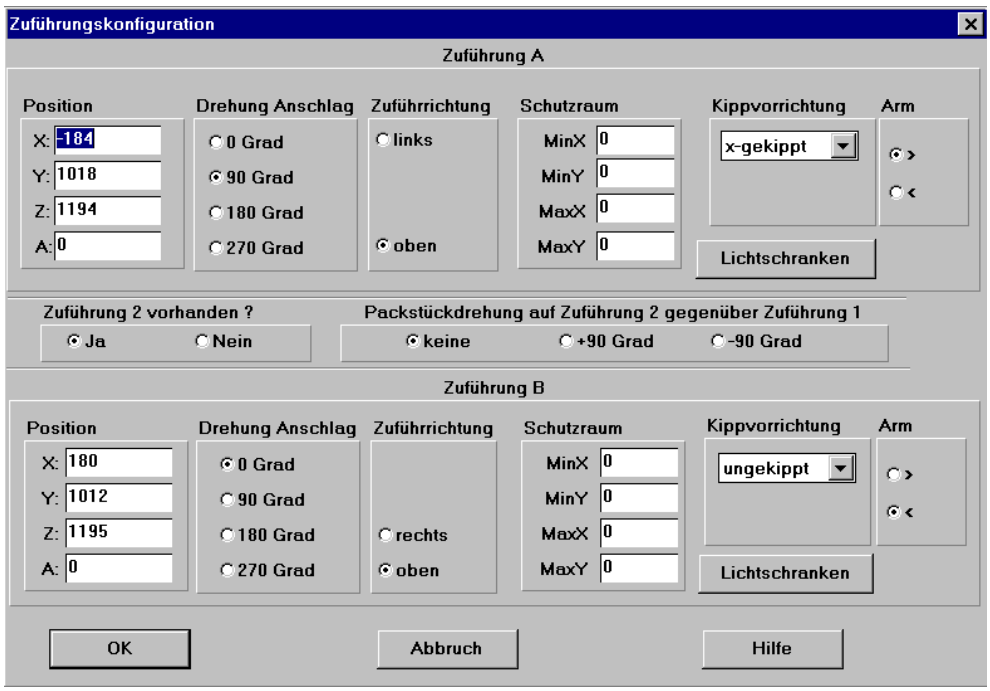


Abbildung 7.4: Dialog zur Konfiguration der Zuführungen

ist im Gegensatz zum Techniker nicht berechtigt, die Anlagenkonfiguration zu ändern. Er darf nur neue Packmuster erzeugen und Packprogramme erstellen. Bei der Erzeugung von Packprogrammen werden die gewünschten Packmuster selektiert und dann die Zuführungs- und Abgabennummern eingestellt (Abb. 7.7). Ferner kann noch eingestellt werden, ob Lagen abgeschlossen werden sollen, was z. B. beim Einfügen von Zwischenlagen (Papierbögen zur Erhöhung der Stabilität) sinnvoll ist. Anschließend werden die Packprogramme entweder via Diskette oder Netzwerk zur Steuerung transferiert. Im (on-line) Packmusterauswahldialog (Abb. 7.8) werden auf Tastendruck (copy) alle Daten automatisch zur Steuerung kopiert. Letztlich muß nur noch das gewünschte Packprogramm selektiert werden. Das ist ein wesentlicher Vorteil gegenüber herkömmlichen Steuerungen - es muß kein PC zusätzlich angeschlossen werden und Baudraten etc. eingestellt werden.

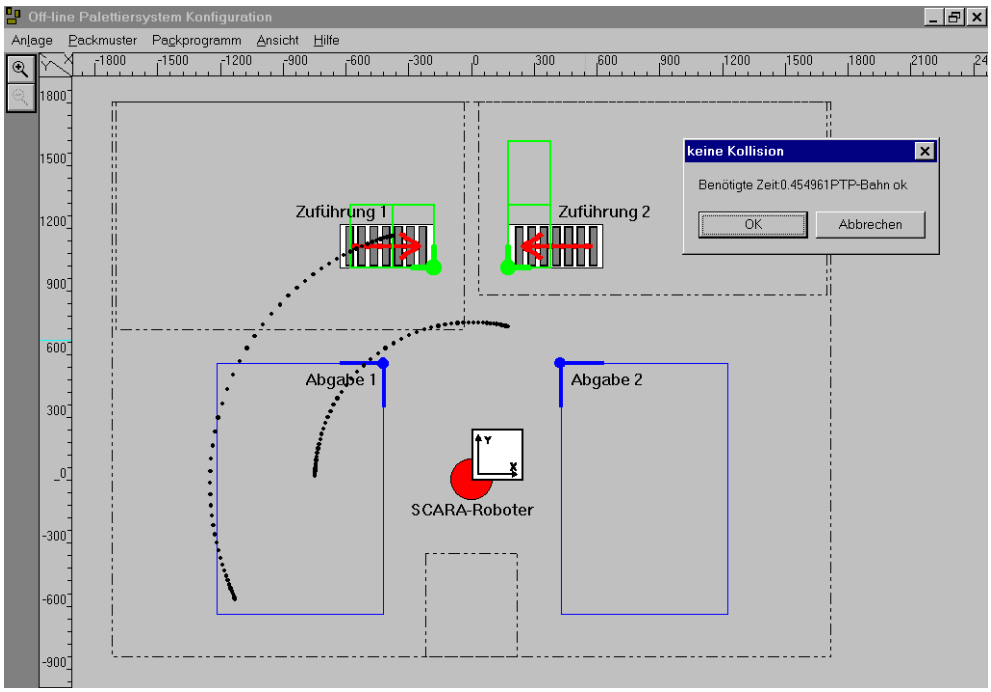


Abbildung 7.5: Eingabe und Überprüfung der Anlagenkonfiguration

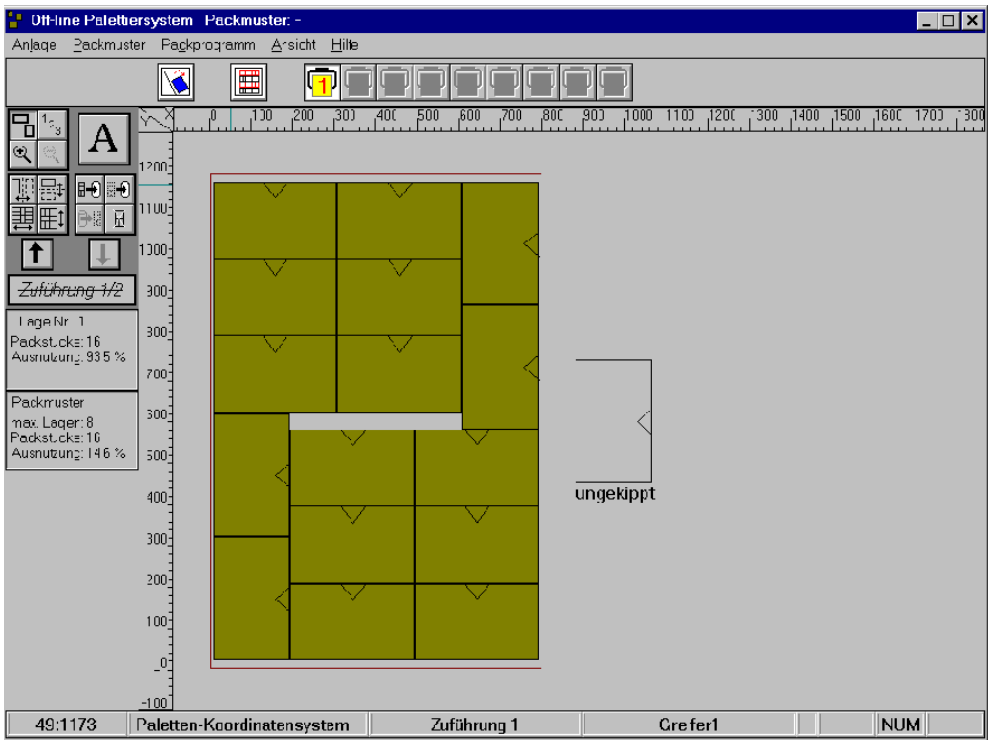


Abbildung 7.6: Graphische Erzeugung des Packmusters

7.3 Anlagensteuerung unter DOS

Das allgemeine Softwarekonzept zur Anlagensteuerung wurde zunächst auf einem DOS Rechner umgesetzt. Dabei wurde eine kommerzielle Software SPS (Beckhoff S2x00) um

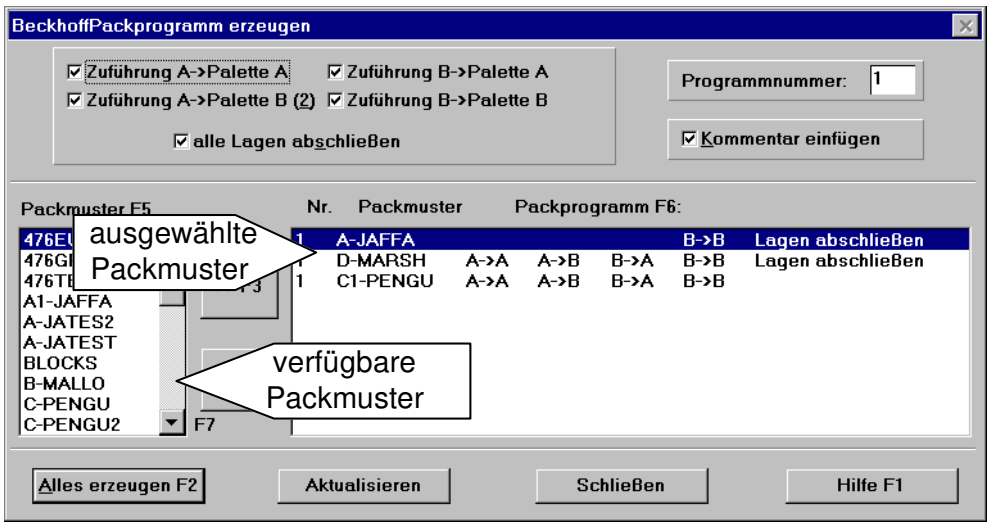


Abbildung 7.7: Erzeugung eines Packprogrammes

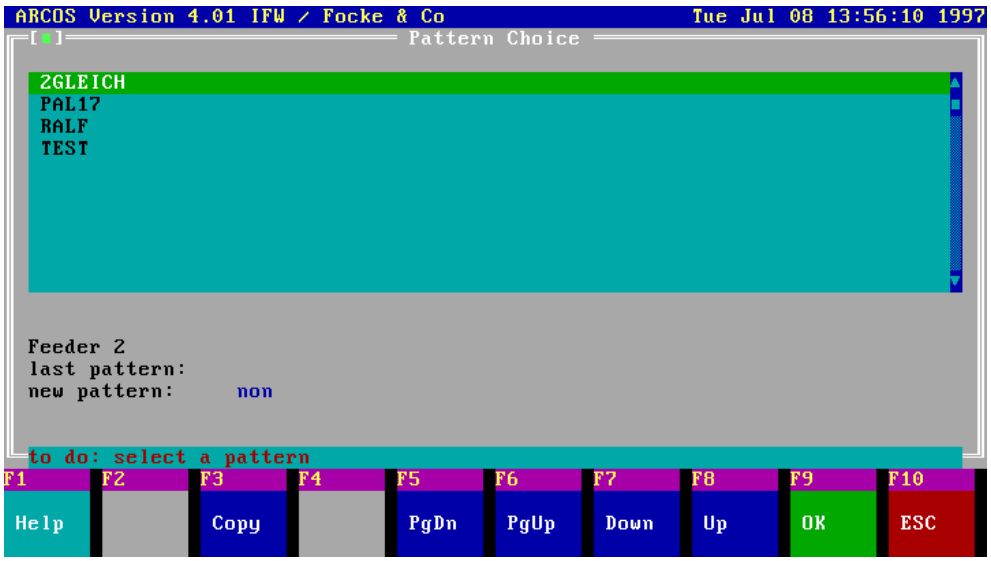


Abbildung 7.8: Übertragung/Anwahl eines Packprogrammes an der Robotersteuerung

Roboter - und Bedienpultfunktionalität erweitert und die Positionierungsalgorithmen in eine Interruptroutine der SPS eingebaut (Abb. 7.9). Aufgrund der höheren Priorität wird erst die Bewegungsinterpolation und dann der SPS Code abgearbeitet.

Der Robotersteuerungsteil arbeitet in Form eines IRL Interpreters kontinuierlich als Hauptapplikation. Die Interpolationsvorbereitung wird vom Interpreter nach der Auswertung eines Bewegungsbefehles aufgerufen und übergibt dann die berechneten Daten (Koeffizienten) an den Positionierungsteil der Interruptroutine. Der Interpreter wartet dann auf das Ende der Bewegung bzw. auf eventuelle Störungen oder einen Wiederanlauf. Im System wird zwischen den Stoparten Notstop und Softstop unterschieden. Beim Notstop werden die Antriebsver-

stärker hardwarebedingt spannungslos geschaltet und die Bremsen aktiviert - der Roboter bremst ruckartig ab. Da diese Stopart nicht sehr mechanikschonend ist, wurde eine Softstopfunktion entwickelt, die den Roboter auf der bereits berechneten Bahn mit den in den Maschinendaten eingestellten Grenzwerten der Beschleunigung abbremst. Diese Funktion kann prinzipiell auch zu einer on-line override Regelung benutzt werden.

Da unter DOS kein Multitasking zur Verfügung steht, muß das Bedienpult quasiparallel von der Hauptapplikation aufgerufen werden. Das hat den wesentlichen Nachteil, daß Bedieneingaben den Ablauf des Roboterprogrammes eventuell verzögern können. Bei Einsatz eines Multitasking Betriebssystemes wird dies allerdings entfallen. Die erreichbare Zykluszeit der Software-SPS liegt bei 1 bis 10 ms in Abhängigkeit der Anzahl der zu interpolierenden Roboterachsen und der eingestellten Interpolationszeit (1 - 2 ms) bei Verwendung eines Pentium P133 Prozessors. Der Leistungsumfang der Software-SPS entspricht einer schnellen Hardware-SPS, wobei die Programmiersprache an AWL [3] (Siemens S5 ähnlich) angelehnt ist.

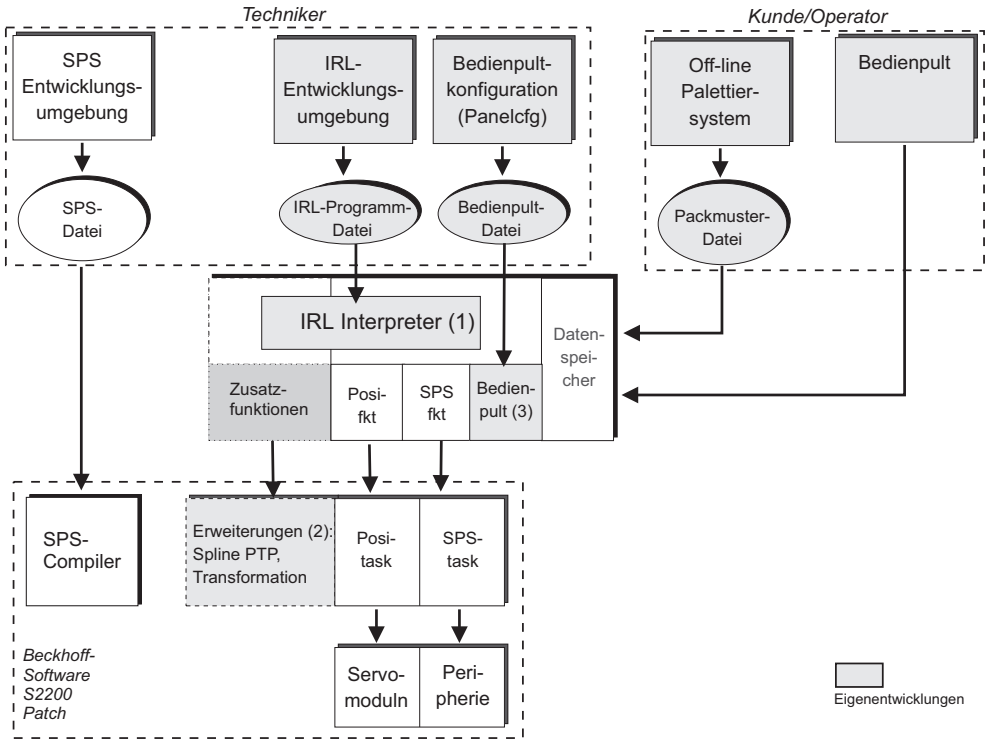


Abbildung 7.9: PC-basiertes Softwarekonzept unter DOS

[3] Anweisungsliste

7.3.1 Roboterprogrammierung in IRL

Die Robotersteuerung wird nach DIN 66312 in IRL[4] programmiert. Die IRL-Syntax ist sehr PASCAL ähnlich und bietet alle Programmiermöglichkeiten einer Hochsprache - Strukturen und Funktionen beliebiger Komplexität können definiert werden. Ein IRL-Interpreter setzt die Befehle wie z. B. "move" in Verfahrbewegungen um. Im Rahmen der Arbeit entstand eine IRL-Entwicklungsumgebung Abb. 7.10, die aus einem Editor mit Makros, Debugger, Bedienpultsimulationsmodul und einem Teachmodul besteht. Kernstück der Umgebung ist der in die Oberfläche integrierte IRL-Interpreter. Dieser basiert auf einer weiterentwickelten Kommandzeilenversion, die bereits am Institut vorhanden war. Mit Hilfe der eingebauten Makros, die beliebig erweiterbar sind, kann wie folgt sehr einfach ein IRL-Programm erzeugt werden:

1. Neues IRL -Programm erzeugen
2. Anfahrpunkt anfahren und speichern, die Koordinaten werden automatisch in die aktuelle Zeile des jeweiligen Editortextes übernommen
3. Programm eventuell anpassen und ausführen. Das Ergebnis zeigt Abb. 7.10.

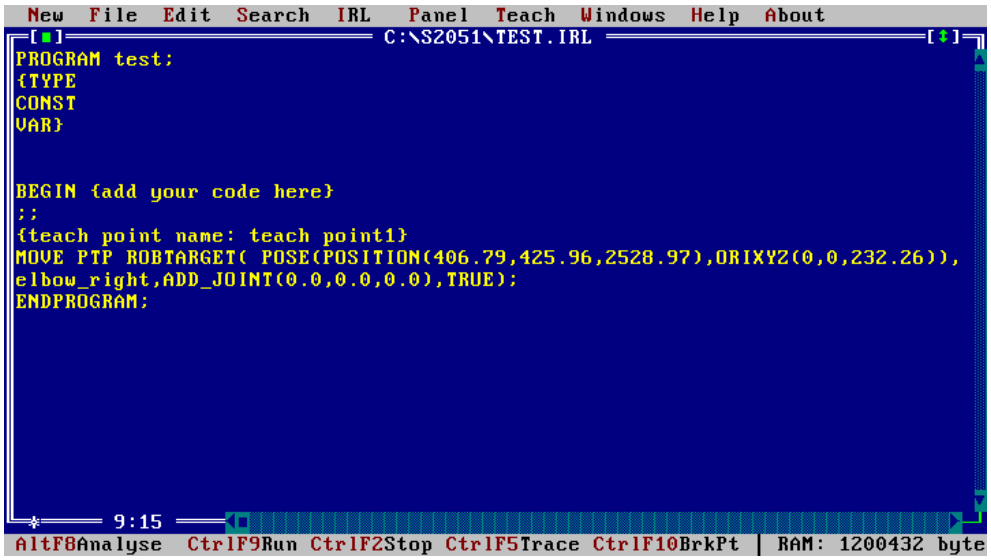


Abbildung 7.10: IRL-Entwicklungsumgebung zur Programmierung von Robotern

Prinzipiell kann die Entwicklungsumgebung, die ursprünglich als einfaches Hilfsmittel zur Inbetriebnahme von Palettieranlagen konstruiert wurde, auch für andere Anwendungen (z. B. Montage) eingesetzt werden. Zur Zeit wird der Bewegungsmodus Punkt zu Punkt (siehe

[4]Industrial Robot Language

Kapitel PTP-Spline) unterstützt. Das nachfolgende kleine Roboterprogramm soll die Möglichkeiten und Komplexität der Robotersprache IRL aufzeigen:

```
PROGRAM ptp_test;
VAR
 ROBTARGET:target_point:= ( ((1000,400,2000),ORIXYZ(0,0,90)));
BEGIN
 r_speed_ptp := 0.1; { globale Einstellungen: setze Override auf 10%}
 R_ACC_PTP := 0.20; {Beschleunigung auf 20%}
 R_RET_PTP:= 0.20;  {Abbremsen auf 20%}
 writeln(SCREEN,'Demo zum Winkelstatus elbow_left ');
 target_point.is_valid := TRUE;
 target_point.status:= elbow_left;
 MOVE PTP target_point ;{ Fahrt mit 10 % Override}
ENDPROGRAM;
```

Der Roboter fährt in diesem Programm von der aktuellen Position bis zum Punkt 1000mm, 400mm, 2000mm mit der Drehung 90 Grad und dem Winkelstatus Ellenbogen links. Mittlerweile wurden Roboterprogramme implementiert, die mehr als 50 KByte groß sind und nicht nur den Roboter positionieren, sondern auch Dateien einlesen und auswerten (siehe Kapitel On-line Packprogramm). Für sehr schnelle und sanfte Bewegungen kann ein Spline-Überschleifen aktiviert werden. Die einzelnen Zielpunkte werden dann in einem Toleranzbereich mit angepaßten Geschwindigkeiten ruckfrei angefahren. Der Interpreter und die Spline-Positionierung wurden als dynamische Bibliothek (DLL [5]) in die Hauptroutine der SPS (Hauptapplikation) eingebunden. Das Robotermodul ist durch Änderung der Transformation an jede Kinematik anpaßbar. Bisher wurden Transformationen für Horizontalknickarm- und Parallelkinematiken programmiert.

7.3.2 Frei konfigurierbares Software-Bedienpult

Die dritte Softwarekomponente der PC-basierten Steuerung ist das Bedienpult. Die Erzeugung der einzelnen Bedienpultfenster erfolgt graphisch über das Konfigurationstool Panelcfg. Mit Hilfe von Panelcfg können Texte, SPS-Werte und IRL-Strings per Maus beliebig im Bedienpult angeordnet werden. Der Funktionalitätsumfang herkömmlicher Mehrzeilenbedienpulte wird somit bei weitem übertroffen.

[5] dynamic link library

Das Software-Bedienpult unterscheidet zwischen System- und Benutzerfenstern. Benutzerfenster (Abb. 7.11) können frei konfiguriert werden und beinhalten die Anzeige von Merkerbits, -bytes, -wörtern sowie Zeichenketten (Strings). Als statische Systemfenstertypen sind zur Zeit Packmusteranwahl- (Abb. 7.8) und Störungsfenster realisiert, weitere Systemfenster sind jedoch denkbar. Im Packmusteranwahldialog können Packmuster von Diskette auf die Steuerungsfestplatte kopiert und angewählt werden. Das Störungsfenster zeigt alle aktuelle anliegenden Störungen inklusive der Hilfetexte. Beseitigte Störungen werden in einem Historyfenster gespeichert, das bis zu 1000 Störungen protokollieren kann.

In Benutzerfenstern ist es möglich, Bits in der SPS zu setzen und somit SPS-Funktionen auszulösen. Durch SPS-Ereignisse (Setzen eines Merkerbits) können verschiedene Fenster in den Vordergrund treten (Popup). Damit ist eine einfache (kontextsensitive) Benutzerführung in Abhängigkeit des jeweiligen Anlagenzustandes möglich. Jedes Benutzerfenster wird in Form einer ASCII-Datei gespeichert und on-line interpretiert. Das Dateiformat wird im Anhang anhand eines Beispielfensters erläutert.

Abbildung 7.11: Konfigurierbares Bedienpultfenster (Beispiel)

7.4 Dreidimensionales Palettiersystem

Beim Design des dreidimensionalen Systems wurde großen Wert auf Plattformunabhängigkeit des Quellcodes gelegt, d. h. alle vom System verwendeten Dateien sollen auf jedem System gelesen und geschrieben werden können. Weil der Quellcode auf mehrere Plattformen portierbar sein sollte, wurde C/C++ als Programmiersprache gewählt. JAVA wurde

aus Geschwindigkeitsgründen noch nicht eingesetzt. Prinzipiell ist dies aber denkbar und hätte den Vorteil, daß ausführbarer Code pro Plattform nicht zusätzlich kompiliert werden muß. Generell wird zwischen graphischen Eingaben und reinen Algorithmen zur Palettierung strikt getrennt. Benutzereingaben erfolgen über eine plattformunabhängige wxwin Bibliothek für Dialoge und der OpenGL/GLUT Bibliothek für dreidimensionale Darstellungen (Abb. 7.13). Die reinen Algorithmen wurden in Standard C++ unter Benutzung der STL Bibliothek geschrieben. Die allgemeine Softwarestruktur ist in Abb. 7.12 dargestellt.

Im Gegensatz zur vorherigen zweidimensionalen Version werden die Packmuster anlagenunabhängig betrachtet, d. h. vor Erzeugung eines Packprogrammes wird das gesamte Packmuster nach unterschiedlichen Packstückmaßen durchsucht. Danach wird jedem Packstückmaß entweder automatisch oder manuell ein virtueller bzw. physikalischer Greifer zugeordnet. Wie beim Vorgänger müssen die Zuführungen den gewünschten Abgaben (Paletten) zugeordnet werden. Die Einfügerichtung ist für jedes Packmuster frei einstellbar. Alle Daten werden in ASCII-Dateien abgelegt, deren Formate im Anhang erläutert werden. Die Schnittstelle zur Anlagensteuerung ist nach wie vor eine ASCII-Datei, deren Format geringfügig geändert wurde.

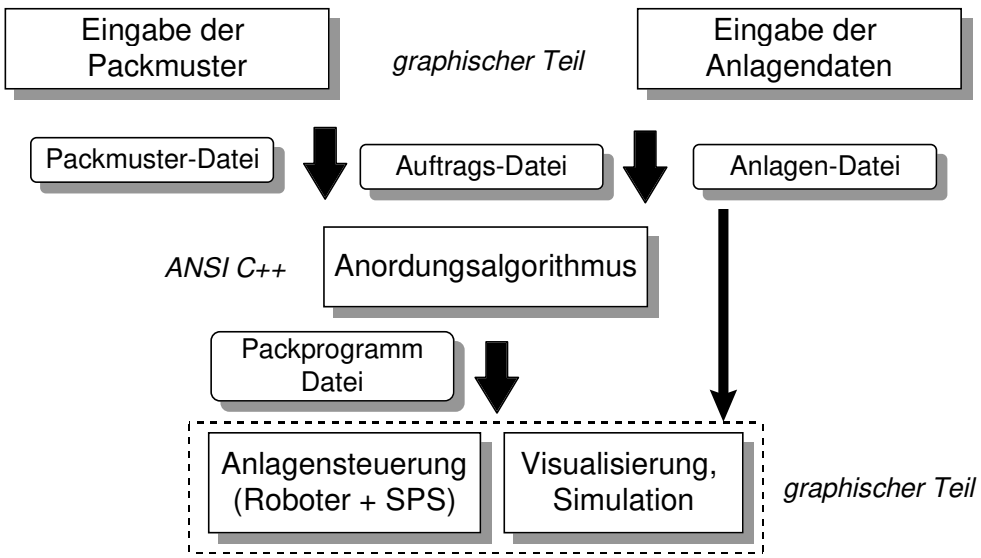


Abbildung 7.12: Softwarestruktur des dreidimensionalen Palettiersystems

7.5 Anlagensteuerung unter Windows NT

Durch den Einsatz von Windows NT als Echtzeitplattform ergeben sich gegenüber dem DOS basierten System drei wesentliche Vorteile:

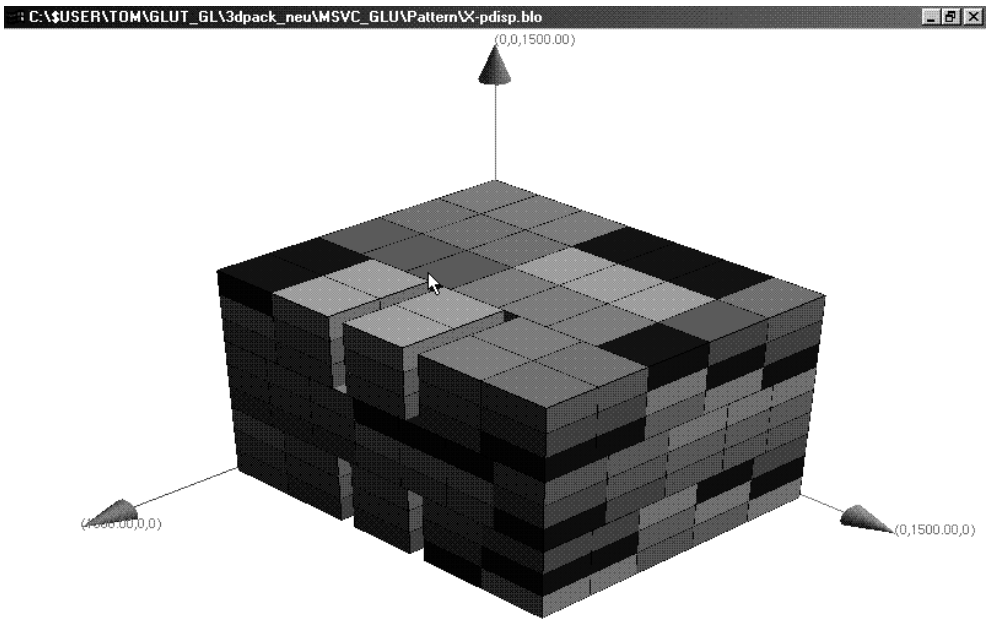


Abbildung 7.13: Oberfläche des dreidimensionalen Palettiersystems

- Multitasking (mehrere Programme können gleichzeitig parallel abgearbeitet werden)
- Einheitliche Bedienung durch Standard-Oberfläche
- Nutzung von PC-Standards: z. B. Programmierumgebungen und Visualisierungen

Die neuen Eigenschaften des verwendeten Betriebssystem Windows NT können zu strukturellen Vereinfachungen des allgemeinen Softwarekonzeptes genutzt werden. Ferner kann in erhöhtem Maße auf Standards zurückgriffen werden und somit eine Art Baukastensystem implementiert werden. Es war z. B. unter DOS noch notwendig, die Positionierungsroutine direkt in den Quellcode des Software SPS Herstellers einzubauen, was zusätzlichen Aufwand bei Updates und zusätzliche Kosten bei der Entwicklung erforderte. Das neue Konzept basiert auf einer Standard IEC 1131 ohne Zugriff auf Herstellerquelltexte, was letztendlich zusätzliche Freiheit für den Anlagenhersteller bedeutet.

Strukturell wird aus Gründen der Übersichtlichkeit nach wie vor zwischen Positionierung und der Peripheriesteuerung getrennt. Allgemein werden dann folgende Komponenten zur Steuerung der Gesamtanlage (Abb. 7.14) benötigt:

1. IEC1131 Software-SPS mit OCX Schnittstelle und mindestens zwei SPS-Tasks mit einer Zykluszeit $<= 1$ ms

2. Roboterprogramm (Ablaufauswertung, Interpolationsvorbereitung)

3. Bedienpult, Prozeßvisualisierung (optional).

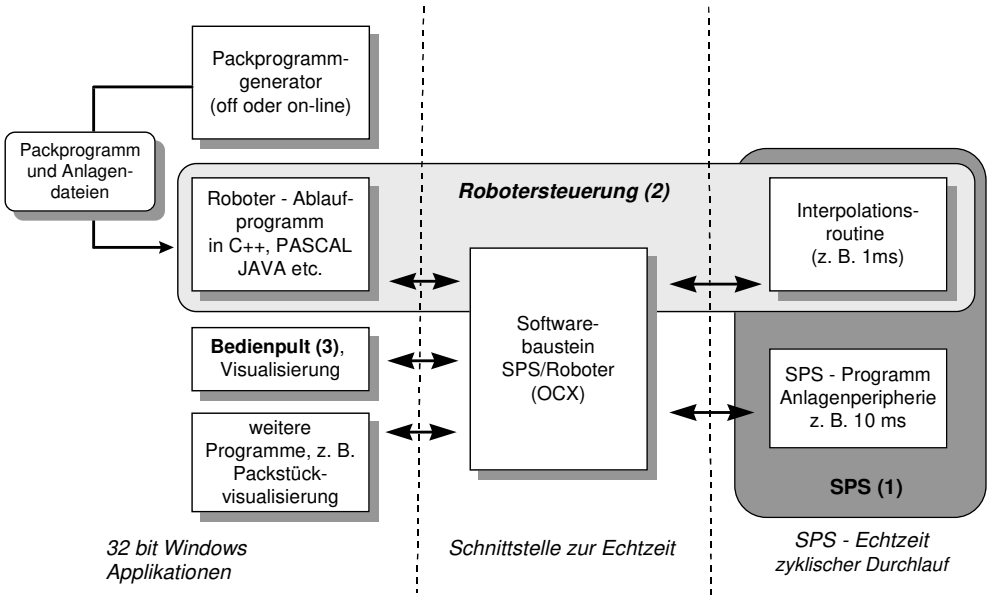


Abbildung 7.14: Softwarekonzept der Steuerung unter Windows NT

Windows NT stellt mit OCX[6] eine Objektschnittstelle zur Verfügung, die einen Zugriff auf interne Teile einer Applikation erlaubt. Im Falle der Steuerung für Palettieranlagen wurde ein OCX zum Zugriff auf die Echtzeitebene einer Software SPS benutzt (Abb. 7.14). Dabei ist es unerheblich, ob es sich bei der aufrufenden Instanz um einen Interpreter eines Bewegungsprogrammes oder um eine Prozeßvisualisierung handelt. Die Robotersteuerung besteht aus einem Roboterablaufprogramm, einem Roboter-OCX mit Interpolationsvorbereitung und einem SPS-Interpolationsprogramm. Das SPS-Programm zur Peripheriesteuerung läuft dann entweder als eigene Task mit niedriger Priorität oder als angehängte Funktion nach der Interpolation. Bedienpult und Prozeßvisualisierungen lesen die Anlagenzustände auch über das ein SPS-OCX ein.

7.5.1 Roboterpositionierung mit SPS-Task

Ein Teil der SPS wird jetzt zur Steuerung des Roboters genutzt, da mit ST[7] in IEC 1131 mittlerweile eine leistungsfähige Hochsprache zur Verfügung steht, so daß die Interpolation der Positionierung komplett aus der C-Interruptroutine in eine SPS-Task verlagert werden

[6] Object linking and embedding control

[7] Strukturierter Text

konnte. Die Positionierungsalgorithmen können somit komfortabel in der IEC 1131 Entwicklungsumgebung überprüft werden (debug), was vorher aufgrund der Einbindung als Interruptroutine nicht möglich war.

Die Interpolationsroutine wird im Millisekundentakt aufgerufen und generiert Geschwindigkeitssollwerte, die dann über die Feldbusmodule an die Servomodule der einzelnen Achsen ausgegeben werden. Die Interpolationsvorbereitung des vorher beschriebenen Spline-PTP-Verfahrens berechnet die gesamten Weg- , Geschwindigkeits- und Beschleunigungsprofile im Voraus und übergibt nur die Splinekoeffizienten und die Daten der Konstantfahrtphase für die einzelnen Achsen über einen Softwarebaustein (OCX) an die SPS. In der SPS-Interpolationsroutine kann dann die Ausgabegeschwindigkeit ($\sim$ Spannung) für die Motoren sehr schnell durch Einsetzen der aktuellen Zeit berechnet werden.

$$u_i(t) \sim v_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Im Anhang ist eine Berechnung der Geschwindigkeiten und Spannngen anhand der Achsdaten der Versuchsanlage zu finden.

Es wird lediglich zwischen Spline und Konstantfahrtphasen unterschieden. Diese Art der Positionierungssteuerung weist für den Palettieranlagenhersteller erhebliche Vorteile auf:

- Unabhängigkeit vom Steuerungshersteller
- Sprachunabhängige Roboterprogrammierung
- Einsatz für beliebige Kinematiken, da keine Quelltexte des Steuerungsherstellers mehr benötigt werden
- Zusätzliche Erweiterungen, z. B. spezielle Regler, können einfach eingebaut werden.

Abb. 7.15 zeigt alle Programmbausteine zur Steuerung einer Palettieranlage. Der Baustein ROBOT ist ein Systembaustein und beinhaltet alle Positionierungsfunktionen: Handverfahren, Referieren, Regler, Spline-Interpolation und Bremsfunktion. Ein Systembaustein kann vom Anlagenprogrammierer benutzt, aber nicht geändert werden. Bei der Inbetriebnahme einer Anlage programmiert der Techniker einen Anlagenbaustein, der dann entweder an den ROBOT-Systembaustein angehängt wird oder als zweite SPS-Task mit niedriger Priorität abgearbeitet wird.

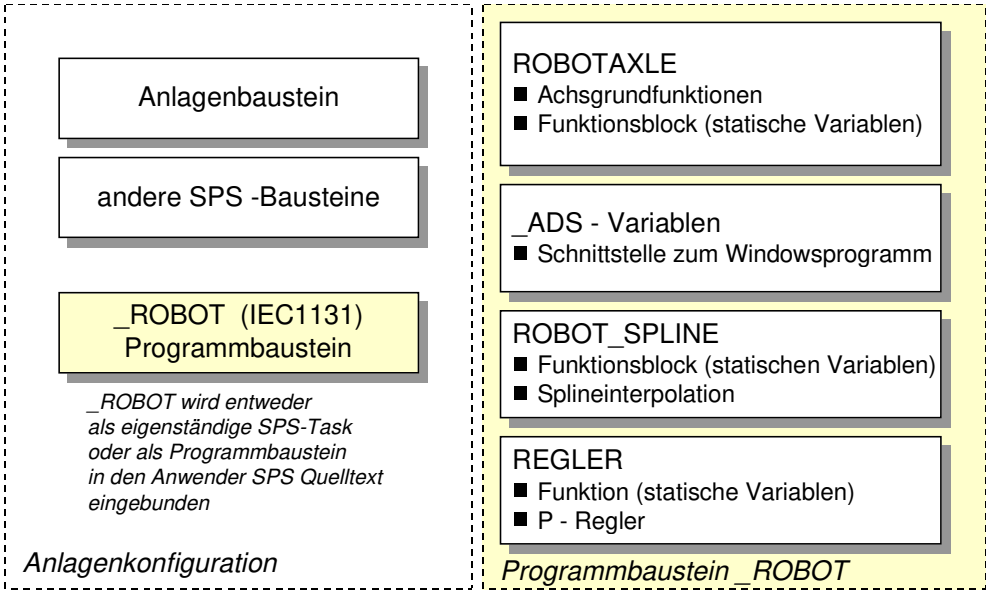


Abbildung 7.15: Programmbausteine der Software SPS

7.5.2 Roboterprogrammierung in beliebiger Sprache

Die Einbindung der Positionierungsalgorithmen (C++ Funktionsbibliotheken) in ein OCX ermöglicht eine sprachunabhängige Programmierung des Roboters. Eine PTP-Funktion (siehe Abb. 7.16) kann sowohl aus C++ Umgebung als auch aus einer Pascal Umgebung heraus aufgerufen werden. Optional ist die Vorverarbeitung durch beliebige Sprachinterpreter für spezielle Anwendungsfälle z. B. IRL realisierbar. Allerdings sollte berücksichtigt werden, daß zukünftig immer mehr graphische Darstellungen und interaktive Aktionen der jeweiligen Anlage verlangt werden. Deshalb scheinen spezielle Programmiersprachen (wie IRL), die fast ausschließlich zur Verarbeitung von Positionsinformationen gedacht sind, ungeeignet für komplexe Projekte mit umfangreichen Darstellungen. Dies gilt vor allem für die Objektorientierung, die heute für größere Softwareprojekte zwingend erforderlich ist. In erster Linie bietet sich C++ als leistungsfähige und objektorientierte Sprache an.

Abb. 7.17 zeigt die Oberfläche des Roboterablaufprogrammes für Hand- und Automatikbetrieb der Windows Lösung. Dieses Programm wurde in C++ mit dem RAD[8]-Werkzeug Power++ geschrieben. Die Zykluszeit für das gesamte SPS-Projekt (Peripherie und Positionierung) beträgt zur Zeit eine Millisekunde bei 30 KByte Code (170 KByte Daten) mit einem Pentium 133. Es ist somit ausreichend Rechenleistung für größere Anlagen vorhanden besonders im Hinblick darauf, daß mittlerweile schon wesentlich leistungsfähigere Prozessoren mit höheren Taktraten ($>$ 350 MHz im Vergleich zu 133 MHz) zur Verfügung stehen.

[8]rapid application development

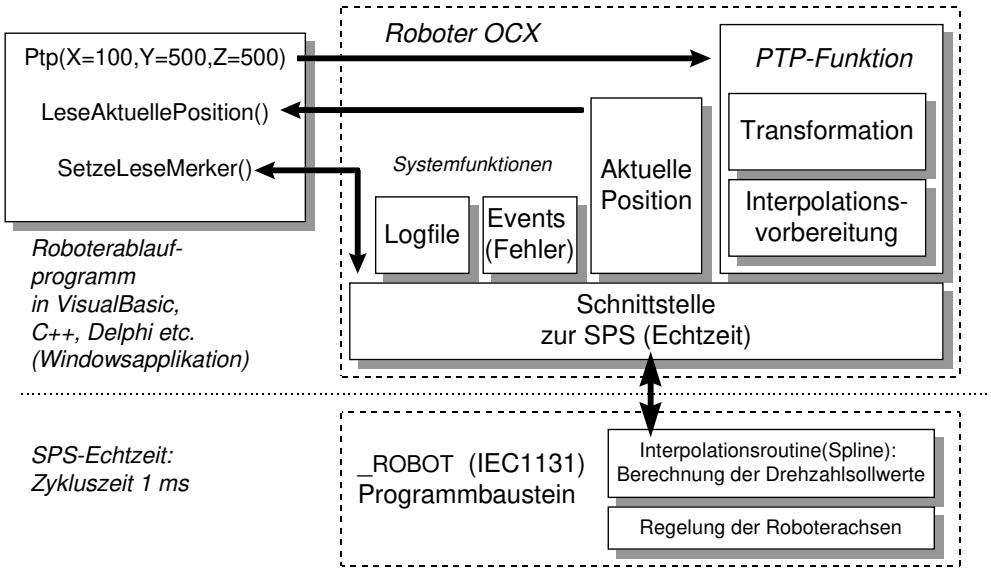


Abbildung 7.16: Ablauf eines Roboterprogrammes

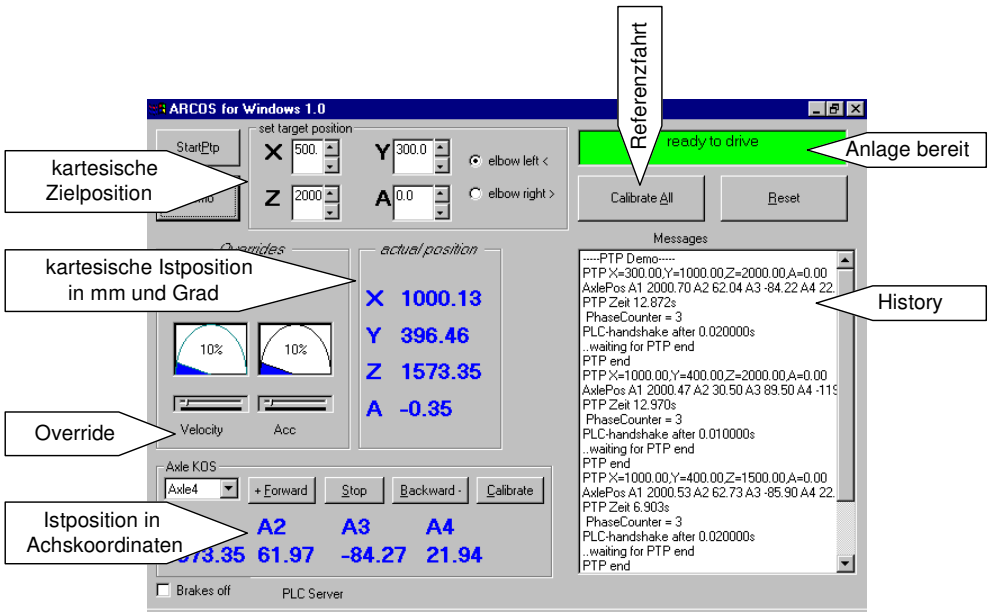


Abbildung 7.17: Verfahrdialog der Robotersteuerung unter Windows NT

7.5.3 Ereignisgesteuerte Anlagenprogrammierung

Multitasking-Betriebssysteme benutzen u. a. das Prinzip der Ereignissteuerung (events). Jedes Anwendungsprogramm ist an einen Kanal des Betriebsystems angeschlossen und reagiert auf die Botschaften (messages) des Betriebssystems. Wenn der Anwender z. B. ein Fenster schließt, wird dem Anwendungsprogramm vom Betriebssystem die Botschaft 'Fenster <x> geschlossen' mitgeteilt. Wenn keine Botschaften für die Anwendung vorliegen,

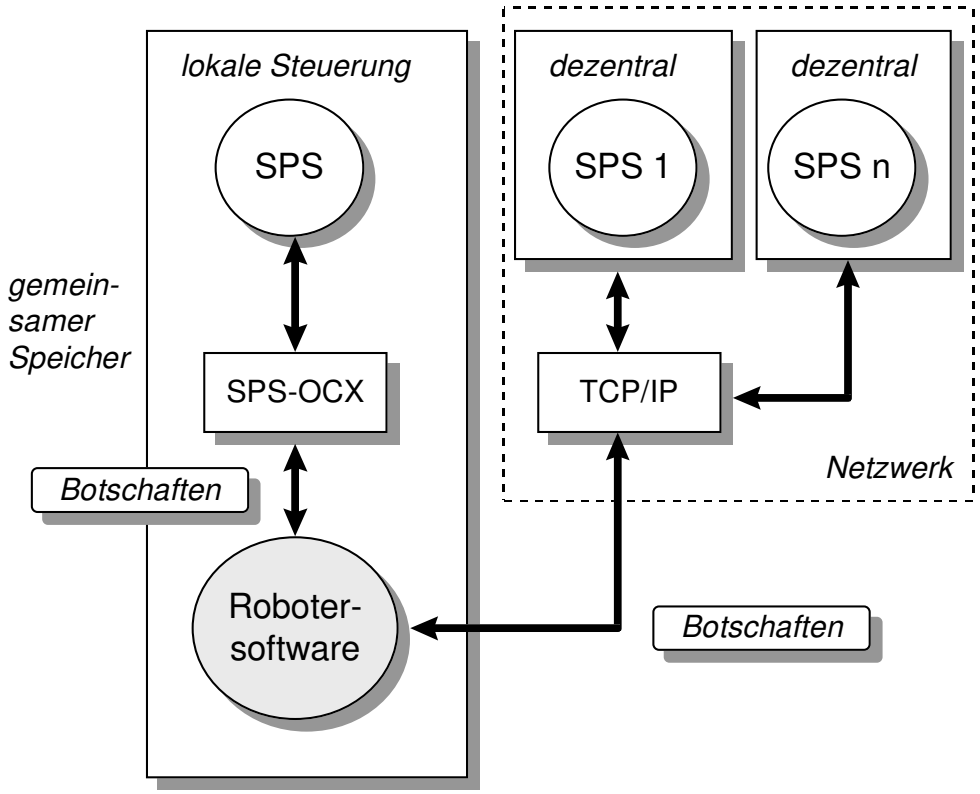


Abbildung 7.18: Ereignisorientierte, verteilte Anlagenprogrammierung

wird die Rechenzeit an das Betriebssystem zurückgegeben. Dieses Prinzip kann auch bei Palettieranlagen bzw. allgemein bei beliebigen Fertigungsanlagen verwendet werden.

Am Beispiel Palettieranlage kann die Kommunikation der einzelnen Objekte (SPS, RC, Bedienpult) ausschließlich ereignisgesteuert über Botschaften erfolgen, was insgesamt folgende Vorteile aufweist

- beliebig viele Teilnehmer,
- Einsparung von Rechenzeit,
- definierte Schnittstellen - keine Speicherbereichskonflikte,
- Dezentralisierung von Rechenleistungen - einzelne Objekte können auch in einem Netzwerk verteilt sein (siehe Abb. 7.18).

Nachteile der Ereignissteuerung sind der höhere Programmieraufwand und die zunächst für den Techniker ungewohnte ereignissteuerte, botschaftenorientierte Denkweise bei der Programmentwicklung. Zur Umsetzung der Ereignissteuerung muß die SPS in der Lage sein, nur dann Meldungen (Messages) zu senden, wenn sich tatsächlich Variablenwerte in

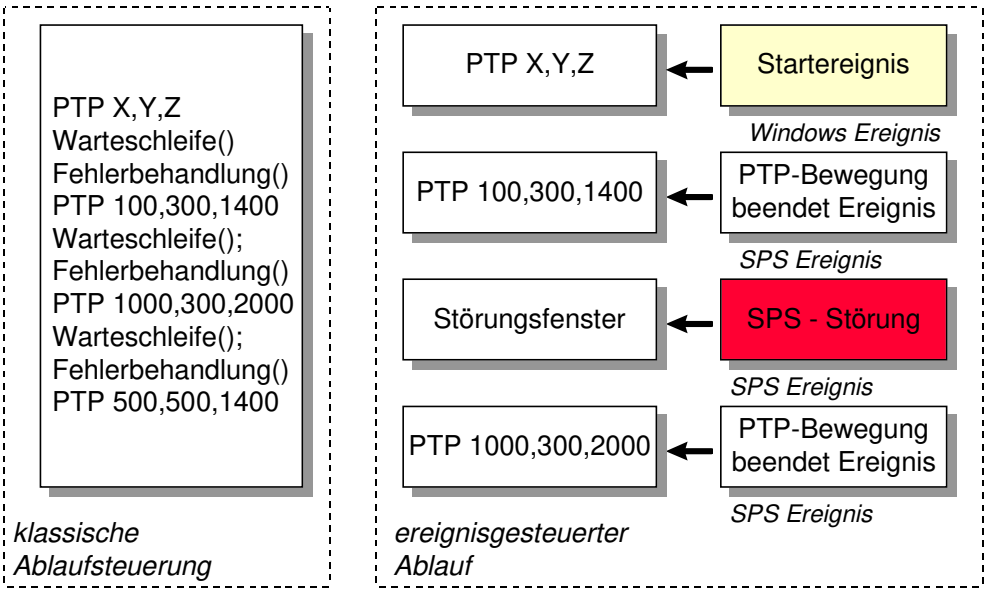


Abbildung 7.19: Klassische und Ereignisgesteuerte Anlagenprogrammierung

der SPS verändert haben. Warteschleifen werden somit effektiv vermieden. Abb. 7.19 zeigt beispielhaft einen Vergleich zwischen dem klassischen Ablauf mit Warteschleifen und einem ereignisgesteuerten Ablauf. Beim klassischen Ablauf werden bestimmte SPS-Merker (z. B. Positionierung beendet) bis zum Ende der Positionierung permanent abgefragt (polling). Der Prozessor wird somit nutzlos belastet. Dahingegen gibt der Positionierungprozeß beim ereignisgesteuerten Ablauf nach Aufruf der PTP-Funktion die Rechenzeit wieder an das Betriebsystem ab.

Der gesamte Palettierablauf (allgemein Fertigungsablauf) ergibt sich aus der Aneinanderreihung der einzelnen Botschaften. Für die erste Phase eines Palettierzyklus der Abb. 7.21 würde sich folgende Reihenfolge ergeben:

1. Starte Packprogramm (1. Phase: hole Pakete (einmal pro Zyklus, 6 Aktionen))
2. Message an SPS, setze Greifernummer
3. PTP in Höhe des Vorpunktes der Zuführung
4. PTP zum Vorpunkt der Zuführung
5. PTP zum Abholpunkt der Zuführung
6. Message an SPS, Sauger aktivieren
7. PTP über den Abholpunkt der Zuführung

Botschaft	Bedeutung	Daten
START PATTERN PROGRAM	Starte Packprogramm	ProgrammIndex, Zykluszahl, Zyklusarray
END PATTERN	Beende Packprogramm	ProgrammIndex
GRIPPER ABOVE FEEDER	Greifer aus Entnahme	-
LAST CYCLE ENDED	SPS darf mit Zyklus fortfahren	ProgrammIndex
SOFTBUTTON PRESSED	Softwareknopf am Bedienpult gedrückt	Knopfkennung
PTP START	Starte PTP	Spline Koeffizienten

Tabelle 7.1: Botschaften des Robotermoduls an die SPS

8. Message an SPS, Greifer aus Entnahme

Es ist hier zu beachten, daß der Roboterarm an einer beliebigen Stelle stehen kann und daher erst eine Fahrt in eine sichere Position in Höhe des ersten Vorpunktes über der Zuführung notwendig ist. Ferner erfordert jede PTP-Fahrt wiederum zwei weitere Messages. Bei der ersten Message (PTP START) werden der SPS die berechneten Spline-PTP Koeffizienten übergeben. Die SPS berechnet dann die Ausgabespannungen für die Antriebe. Wenn die Bewegung beendet ist, erhält die Robotersteuerung die Aufforderung (PTP END) neue Koeffizienten zu senden.

Nachfolgend werden exemplarisch einige Nachrichten gezeigt, die zur Steuerung einer Palettieranlage notwendig sind. Die Botschaften des Robotermoduls an die SPS sind in Tabelle 7.1 und die umgekehrte Richtung in Tabelle 7.2 aufgelistet. Bei der Implementierung der Botschaftenkommunikation muß zwingend ein ausreichender Speicherbereich für die in den Tabellen aufgelisteten Daten zur Verfügung stehen.

7.5.4 Neue Möglichkeiten durch Multitasking-Plattform

Der Einsatz von Windows NT als Echtzeitplattform ermöglicht die Verschmelzung von off und on-line Programmierung auf einem System. Mit anderen Worten kann das off-line Palettiersystem jetzt auch on-line benutzt werden. Parallel zur Palettierung kann jetzt ein neues Muster erzeugt und in ein Packprogramm umgewandelt werden. Desweiteren können Standardkommunikationsmöglichkeiten wie z. B. Modem oder Netzwerkkarte genutzt werden. So läßt z. B. sich die Robotersteuerung per Modem mit dem Softwareprodukt PC-Anywhere

Botschaft	Bedeutung	Daten
FEEDER READY	Packstücke zur Abholung bereit	ProgrammIndex, ZyklusNr
FAULT	Störung	Störungsnummer oder String
EMERGENCY STOP	Notaus	-
PLANT READY	Anlage ist hochgefahren	-
HARDBUTTON PRESSED	Hardwarebutton gedrückt (Taster)	Buttonkennung
PLANT STATE CHANGED	Anlagezustand hat sich geändert	Zustandskennung
PTP END	Ende einer PTP-Fahrt	-

Tabelle 7.2: Botschaften der SPS an das Robotermodul

komplett fernwarten. Palettieranlagen können via Ethernet zu komplexen Palettierstraßen vernetzt werden. Abb. 7.20 zeigt vernetzte Palettieranlagen mit Fernwartungsmöglichkeit via Modem. Ein Leitstandrechner plant und überwacht die einzelnen Palettierprogramme und verteilt diese über das Ethernet an die einzelnen Palettierer. Im Störungsfall können die Zustände der einzelnen Anlagen entweder am Leitstandrechner oder wahlweise über Modem bzw. Internet an einem PC des Servicetechnikers visualisiert werden. Ein Anschluß an das Internet hat einen großen Preisvorteil, weil nur jeweils die lokalen Anschluß- und Telefongebühren bis zu einem (Provider-) Einwahlknoten bezahlt werden müssen.

7.6 On-line Packprogramm

Das On-line Packprogramm wertet eine vom Off-line Palettiersystem erzeugte Packprogrammdatei aus und generiert Verfahrbefehle für die Palettierzyklen. In Abhängigkeit der Anlagenkonfiguration und des Zyklus werden unterschiedliche Zwischenpunkte erzeugt, da das off-line Palettiersystem nur die Abholpunkte und Ablagepunkte zur Verfügung stellt. An einer Anlage können mehrere Packprogramme quasiparallel abgearbeitet werden, wobei immer nur ein Programm aktiv sein. Die SPS teilt dem Roboterprogramm mit, von welcher Zuführung Packstücke entnommen werden d. h. welches Packprogramm abgearbeitet werden soll.

Am Zyklusbeginn wird ein Zwischenpunkt über der Entnahme berechnet, der Kollisionen

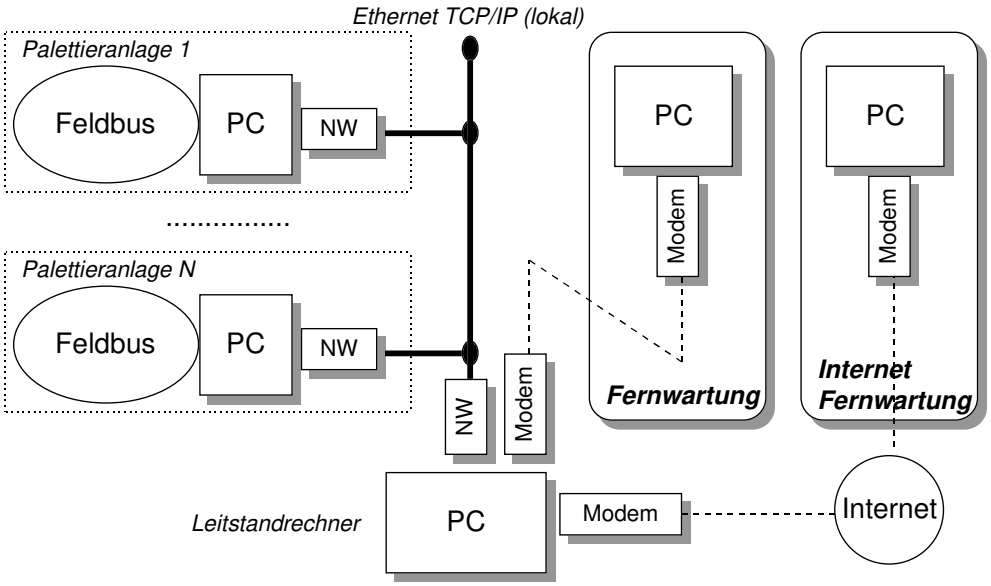


Abbildung 7.20: Vernetzte Palettieranlagen mit Fernwartung

mit den Anschlageecken oder den bereits palettierten Packstücken verhindert (Abb. 7.21). Danach wird die Höhe beibehalten und in xy Richtung ein Einfügepunkt angefahren. Der Einfügepunkt soll später eine Kollision verhindern, die beim direkten Positionieren von oben durch Packstücktoleranzen mit bereits liegenden Packstücken entstehen könnte. Letztlich wird der Ablagepunkt angefahren und das Packstück abgelegt. Mehrfachgreifer erfordern eine zusätzliche Erzeugung von Zwischenpunkten, da hier mehrere Packstücke nacheinander abgelegt werden können. Abb. 7.22 zeigt einen solchen Zyklus. Das erste Packstück wird wie vorher beschrieben abgelegt. Nachfolgend fährt der Greifer nicht zur Entnahme sondern in Lagenhöhe plus einem Offset und legt das zweite Packstück ab. Bei Mehrfachgreifern wird diese Vorgehensweise entsprechend wiederholt. Unter Ausnutzung des PTP-Spline Verfahrens werden alle Zwischenpunkte überschliffen, so daß eine kollisionsfreie geschwindigkeitsoptimierte Bahn entsteht. Der Roboter kommt nur in den Start- und Endpunkten vollständig zum Stillstand.

Eine andere Aufgabe des on-line Packprogrammes in Zusammenarbeit mit der SPS ist die Reaktion auf Störzustände der Anlage. In jedem Störzustand muß ein korrekter und kollisionsfreier Wiederanlauf gewährleistet sein. Das bezieht sich auch auf das Ausschalten der Anlage. Hier muß remanent gespeichert werden, wie viele Packstücke bereits auf der Palette liegen und welche Packprogramme aktiviert sind. Abb. 7.23 zeigt die wesentlichen Zustände einer Palettieranlage. Nach dem Einschalten und dem Reset befindet sich die Anlage im bereit Zustand - ein Packprogramm kann jetzt angewählt und abgearbeitet werden. Während der Palettierung können die Zustände Softstop, Notaus oder Palettenwechsel auftreten.

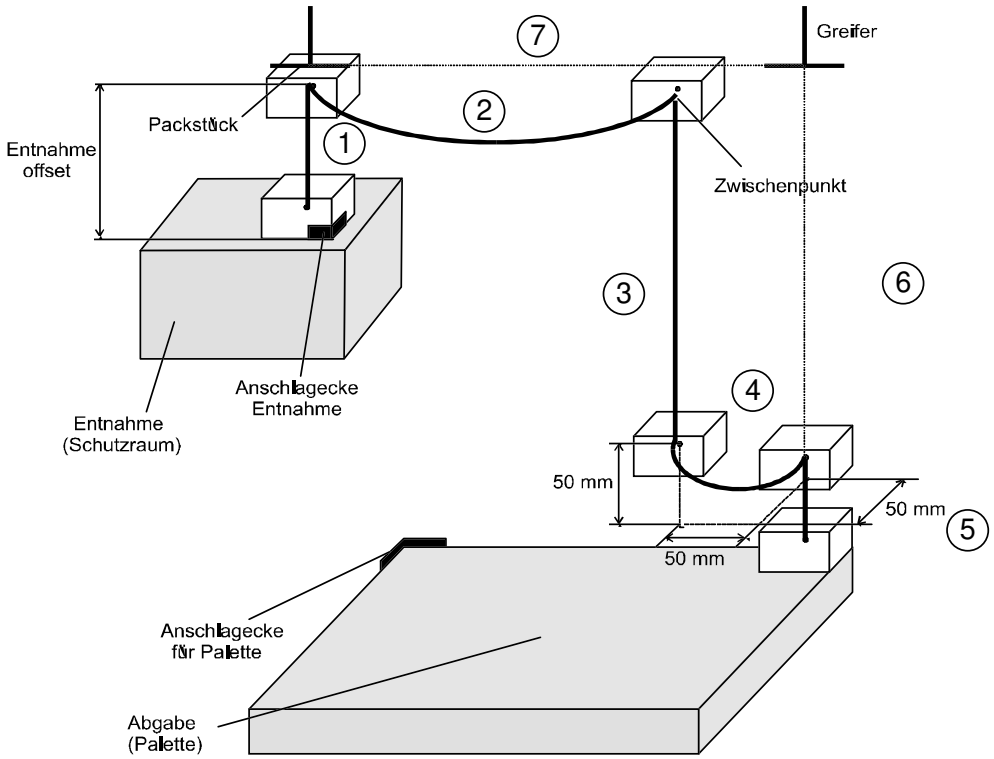


Abbildung 7.21: Zwischenpunkte beim on-line Packprogramm

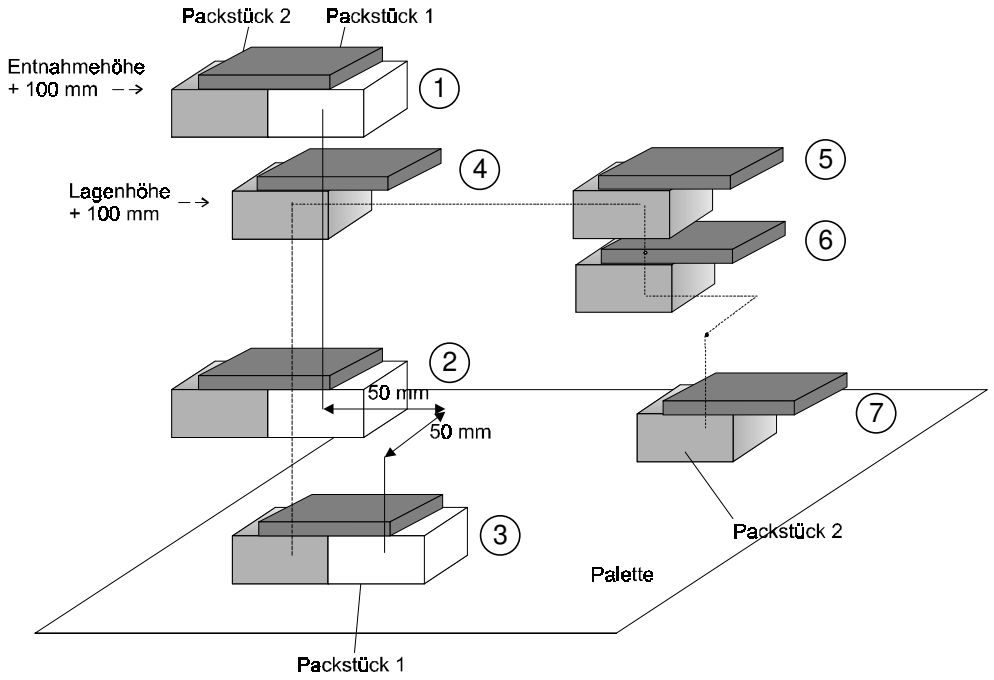


Abbildung 7.22: Zwischenpunkte bei der Vereinzelung mit Mehrfachgreifer

Nach Bearbeitung dieser Ereignisse ist die Anlage wieder im Aktiv-Zustand und setzt die Palettierung fort.

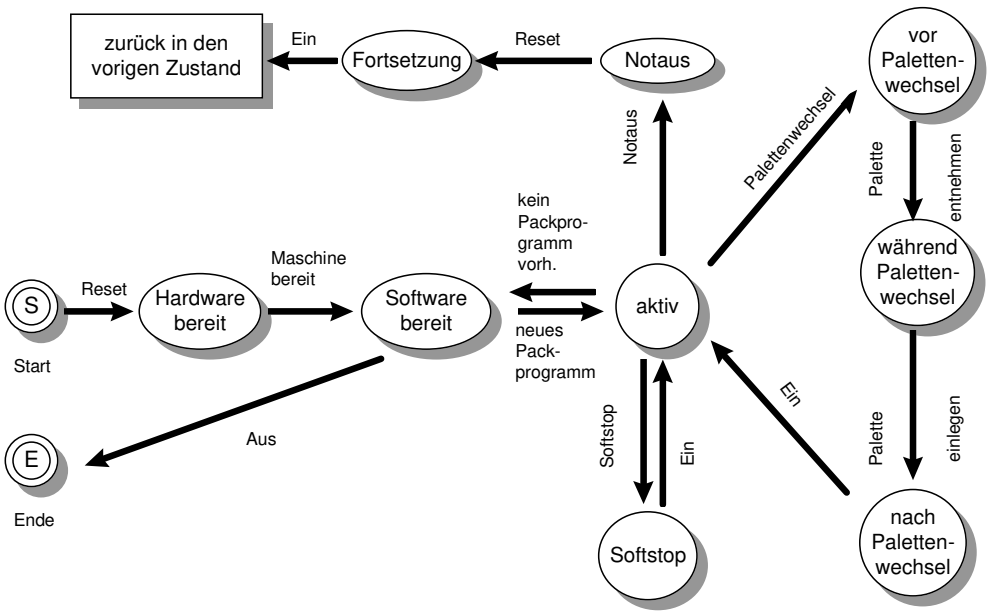


Abbildung 7.23: Vereinfachtes Zustandsdiagramm einer Palettieranlage

Nachfolgend werden die Unterschiede der on-line Varianten für die DOS und Windows Konzepte diskutiert. Beim DOS Konzept wurde das Packprogramm in IRL geschrieben. Es wird on-line vom IRL-Interpreter ausgewertet. Die Kommunikation erfolgt über direktes Setzen von SPS Variablen (Merkern). Im Gegensatz dazu wurde unter Windows ein C++ Programm geschrieben und in Maschincode übersetzt (Abb. 7.24) . Diese direkte Implementierung hat Geschwindigkeitsvorteile und bietet die Möglichkeit zusätzlicher Visualisierung z. B. der Packstücke. Nachteil ist die erforderliche Neuübersetzung (Compilierung) nach Änderung des Packprogrammaufbaus, wobei fraglich ist wie oft eine Änderung des on-line Programmes wirklich notwendig ist. Die IRL Variante scheint momentan besser für Techniker geeignet und somit praxistauglicher zu sein.

Abbildung 7.24: On-line Packprogramm unter Windows NT

Kapitel 8

Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein Palettiersystem zur automatischen, aber dennoch flexiblen Palettierung mit Handhabungssystemen beliebiger Kinematik unter Berücksichtigung von Mehrfachgreifern entwickelt. Dieses System bietet eine durchgängige Lösung von der Packmustererzeugung über die optimierte Bildung von Ablagezyklen(für die on-line Palettierung) bis hin zur schnellen meachanikschonenden Achspositionierung. Daraus resultierten folgende Schwerpunkte:

- Off-line Programmiersystem für Mehrfachgreifer und
- PC-basierte Steuerung mit Positionierung .

Ausgangspunkt ist die auftragsbezogene Kommissionierung eines beliebigen dreidimensionalen Packmusters mit Packstücken beliebiger Größe. Das Packmuster wird unter Berücksichtigung eines mathematischen Modells der gesamten Palettieranlage und einer Einfügerichtung in ein gerichtetes Graphmodell transformiert. Sogenannte Anordungsalgorithmen, die im off-line Programmiersystem integriert sind, durchlaufen den Graphen und bilden mit Hilfe eines Greiferauffüllalgorithmus automatisch geschwindigkeitsoptimierte, kollisionsfreie Ablagezyklen (Packprogramme). Eine Kollisionsroutine prüft abschließend Kollisionen mit Anlagenteilen und bereits liegenden Packstücken. Zusätzlich werden auch die erforderlichen Schaltinformationen zur Ansteuerung der Zuführstationen und der Greifer erzeugt. Da die eigentliche Roboterprogrammierung komplett entfällt, ist die Palettieranlage auch für Anwender ohne Programmierkenntnisse zu bedienen. In dieser Arbeit wurden vier verschiedene Varianten von Anordungsalgorithmen entwickelt und untersucht. Diese basieren auf:

- genetischen Algorithmen

- Packstückgraphen
- Blockgraphen mit und ohne Subblockaufteilung

Das Blockgraphverfahren lieferte hier die besten Ergebnisse, um in kurzer Zeit gute Lösungen zu erhalten. Die Lösungen werden später entweder in Reinform (nur die Daten) oder als Steuerungsprogramm in der Sprache der verwendeten Positionierungseinheit (z. B. IRL) abgespeichert. Herkömmliche Steuerunge erwiesen sich aufgrund kleiner Arbeitspeicher und hoher Übertragungszeiten gerade für größere Packmuster (> 100 Packstücken) als ungeeignet.

Folglich wurde ein Steuerungskonzept auf Basis eines Standard-PC geschaffen, das durch eine optimale Umsetzung Packprogrammdaten kurze Übertragungs- und Palettierzeiten sicherstellt. Das neue Steuerungskonzept verlagert sämtliche Bedien-, Steuerungs- und Positionierungsaufgaben auf die Softwareebene. Es wird nur noch ein Industrie-PC für die gesamte Anlage benötigt. Ausgehend von einer Standard-Software-SPS wurde eine Robotersteuerung aufgebaut, die aus einem Interpreter für Roboterprogramme (IRL) und einem Spline-PTP-Verfahren besteht. Aus der Untersuchung verschiedener herkömmlicher Positionierverfahren resultierte die Entwicklung eines Spline-PTP-Verfahrens, das schnelle und mechanikschonende Palettierung ermöglicht. Weitere Vorteile der PC-basierten Steuerung sind die vereinfachte Inbetriebnahme und Programmierung sowie die generelle Offenheit, insbesondere für verschiedene Kinematiken. Die praktische Umsetzung des Steuerungskonzeptes führte zur Entwicklung von zwei Varianten:

- PC-basierte Palettieranlagensteuerung auf DOS Basis, die mittlerweile in der Industrie mehrfach eingesetzt wird. Diese Variante besteht aus einer IRL-Entwicklungsumgebung, einem frei konfigurierbarem Bedienpult inklusive graphischer Konfiguration und einer SPS.
- PC-basierte Palettieranlagensteuerung auf Basis von Windows NT, die noch im Stadium eines Prototypen ist und auf einer Standard IEC 1131 SPS basiert. Zur Änderung der (Kinematik-) Transformation sind hier keine Quelltexte des SPS-Herstellers notwendig.

Das Hardwarekonzept ist für beide Steuerungen identisch. Wesentliche Vorteile des Steuerungskonzept unter Windows NT sind Multitasking-Fähigkeit und die Möglichkeit der vollständigen Simulation auf dem Bürorechner. Die Software-Inbetriebnahme kann somit bereits vor- oder gleichzeitig zum mechanischen Aufbau erfolgen - die erstellten Programme für Roboter und SPS sind später übertragbar. Mit Hilfe einer Anbindung an eine dreidimensionale

Simulation können ganze Anlagen - nicht zwingend ausschließlich Palettieranlagen - in Echtzeit getestet und eingerichtet werden.

Bewertung und industrielle Bedeutung

Das off-line Palettiersystem und die PC-basierte Steuerung wurden bereits in der Industrie erfolgreich eingesetzt und bewiesen Praxistauglichkeit. Ferner reduzierte die neue Architektur die Inbetriebnahmezeiten. Inzwischen ist die off-line Palettiersoftware ein zusätzliches Verkaufsargument für die Anlagen des Industriepartners, gerade weil diese auch von Kunden ohne Kenntnisse einer (Roboter-) Programmiersprache bedient werden kann. Aufgrund der PC-basierten Steuerung konnten Palettieranlagen einfach vernetzt und an übergeordnete Fertigungsleitstände angebunden sowie ferngewartet werden. Mittlerweile existieren mehrere (Palettieranlagen-) Applikationen mit der PC-Steuerung. Dies sind zum einen stand-alone Lösungen ohne off-line Palettiersystem, zum anderen aber auch mehrere über Ethernet vernetzte Palettieranlagen, die von einem Leitstandrechner aus angesteuert werden. In einem Fall wurde die Steuerung auf eine pick-and-place Anlage mit Parallelkinematik angepaßt. Bereits nach zwei Wochen konnten erste Tests der Anlage nach Änderung der Transformation durchgeführt werden.

Ausblick

Zukünftige Arbeiten im Bereich der Palettierung könnten einige Algorithmen dieser Arbeit zur Palettierung bei unbekannter Kommissionierung nutzen, da diese Art der Palettierung eine Untermenge des im Rahmen dieser Arbeit Realisierten bildet. Erweiterungen im Bereich der übergeordneten Auftragsplanung und Anbindung an ein PPS-System sind genauso denkbar wie neue Heuristiken für den Ablagealgorithmus. Dabei können eventuell auch neuronale Netze auf Tauglichkeit hin untersucht und benutzt werden. Ein weiterer sinnvoller Schritt wäre eine palettenübergreifende Planung für mehrere Palettieranlagen. Ein Kundenauftrag wird automatisch auf mehrere Palettierstationen verteilt - die Packmuster werden dabei automatisch optimiert erstellt. Als Rückmeldung erhält das PPS-System u. a. die Anzahl der benötigten Paletten.

Auch für den Bereich der Steuerungen wird die Vernetzung immer wichtiger. Ein echtzeitfähiges Ethernet könnte hier langfristig evtl. die Feldbusebene verdrängen, wodurch Fertigungsabläufe erheblich transparenter werden. Das vorgestellte ereignisorientierte Steuerungskonzept kann dabei als Grundlage zur Steuerung von vernetzten Fertigungsinseln bzw. Linien dienen. Die Ereignisstruktur bleibt erhalten, das Übertragungsmedium wird jedoch

das Netzwerk und nicht der Speicherbereich sein. Jede Steuerungskomponente SPS, RC, NC oder PC kann als Softwareobjekt mit Eigenschaften und Funktionen aufgefaßt werden. Ein Fertigungsablauf entsteht durch entsprechende Aneinanderreihung von Funktionsaufrufen bzw. Botschaften. Das widerspricht nicht dem in dieser Arbeit vorgestellten Konzept einer all-in-one Lösung RC, SPS und Bedienpult für Palettieranlagen, sondern stellt lediglich die Erweiterung der objektorientierten, ereignisgesteuerten Anlagenprogrammierung dar. Zukünftig wird es je nach Anwendungsfall wie bisher eine Koexistenz von zentralen und dezentralen Steuerungslösungen geben. Aufgrund der Technologieentwicklung im Prozessor bzw. PC-Bereich wird vermutlich ein zunehmender Überhang von PC-Lösungen gerade im Bereich von Fertigungsinseln eintreten. Die komplette vernetzte Steuerung von Fertigungsanlagen mit nur einem PC ist, wie am Beispiel Palettieranlage in dieser Arbeit demonstriert, zumindest möglich.

Anhang A

Algorithmentests

A.1 Testmuster

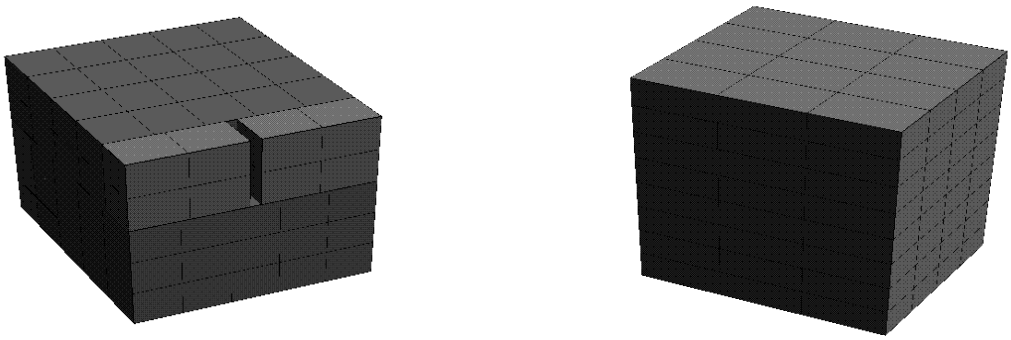


Abbildung A.1: Packmuster 'b-mallo' und 'm-zebra'

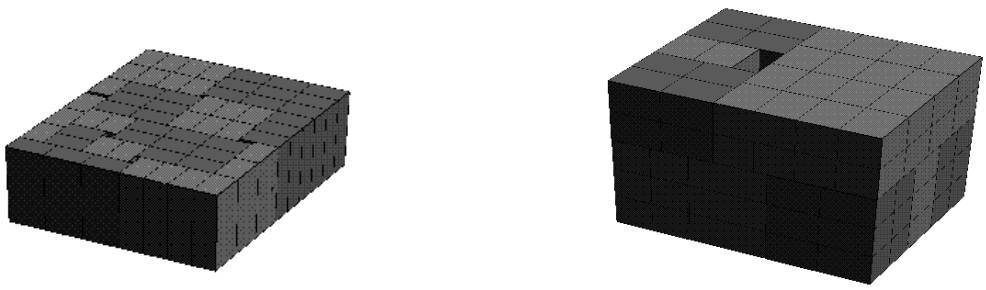


Abbildung A.2: Packmuster 'opt' und 'c-pengu2'

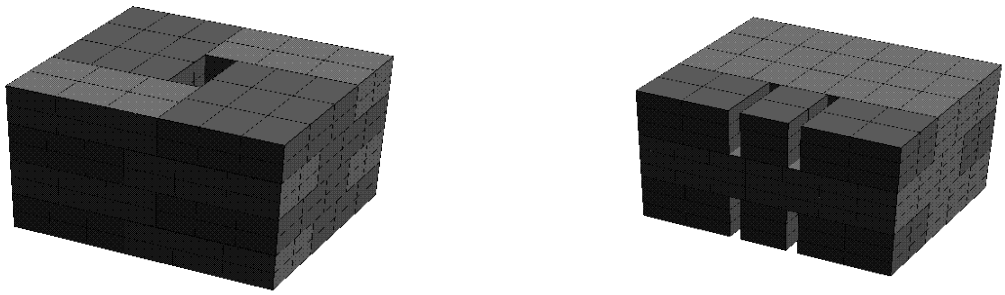


Abbildung A.3: Packmuster 'w-misp' und 'x-disp'

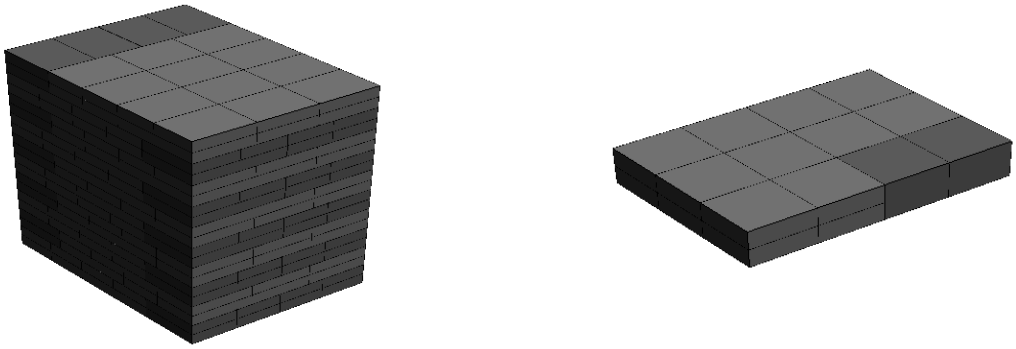


Abbildung A.4: Packmuster 'z1-orang' und '3DTest'

A.2 Rechenzeiten

A.3 Dateiformat für Packmuster

```
FILENAME 3DTest.PTT
NUMBER_OF_PACKAGES 22
;Syntax: PACKAGE PackageNumber X Y Z A Label XL YL ZL GripperNo
PACKAGE 1 710 100 0 90 0 200 210 82 0
PACKAGE 2 500 100 0 90 0 200 210 82 0
PACKAGE 3 710 300 0 90 0 200 210 82 0
PACKAGE 4 500 300 0 90 0 200 210 82 0
PACKAGE 5 290 100 0 90 0 200 210 82 0
PACKAGE 6 80 100 0 90 0 200 210 82 0
PACKAGE 7 290 300 0 90 0 200 210 82 0
PACKAGE 8 80 300 0 90 0 200 210 82 0
PACKAGE 9 710 500 0 90 0 200 210 82 0
PACKAGE 10 500 500 0 90 0 200 210 82 0
```

Packmuster	Greifer	Rechenzeit [s] Block	Rechenzeit [s] Packstückgraph
a-jaffa (28 Packstücke), 4 Blöcke	2*1	0.691 s	1 s
	3*2	0.06 s	14 s
	4*2	0.06 s	301 s
	4*3	0.06 s	$>$ 24 h
fract (14 Packstücke), Fraktal	2*1	0.16 s	$<$ 1 s
	2*2	0.10 s	$<$ 1 s
	4*2	0.33 s	5 s
	4*3	0.16 s	1310 s
	4*4	0.05 s	31266 s
opt (59 Packstücke), optimiertes Muster	2*1	1.21	$<$ 1
	2*2	2.09	$<$ 1
	3*1	2.36	$<$ 1
	3*2	1.49	18
	3*3	3.31	120

Tabelle A.1: Vollständiger Vergleich der Rechenzeiten

Packmuster	Greifer	Rechenzeit [s] Block
b-mallo (120 Packstücke)	2*1	0.57
	2*2	0.09 s
	3*1	0.13 s
	3*2	0.08 s
	4*2	0.07 s
	4*3	0.07 s
c-pengu2 (130 Packstücke)	2*1	0.24
	2*2	0.07 s
	3*1	0.21 s
	3*2	0.07 s
	3*3	0.07 s
	4*2	0.06 s
	4*3	0.06 s
m-zebra (96 Packstücke)	2*1	0.09
	2*2	0.05 s
	3*1	0.07 s
	3*2	0.05 s
w-mdisp (240 Packstücke)	2*1	0.651
	2*2	0.13 s
	3*1	0.11 s
	3*2	0.08 s
	3*3	0.06 s
	4*3	0.05 s
x-disp (279 Packstücke)	2*1	0.661
	2*2	0.221 s
	3*2	0.24 s
	3*3	0.07 s
	4*3	0.08 s
	4*4	0.08 s
z1-orang (320 Packstücke)	2*1	0.13
	2*2	0.07 s
	3*1	0.09 s
	3*2	0.06 s
	3*3	0.04 s
	4*3	0.04 s

Tabelle A.2: Rechenzeiten des 3D Algorithmus

```
PACKAGE 11 80 500 0 90 0 200 210 164 0
PACKAGE 12 290 500 0 90 0 200 210 164 0
PACKAGE 13 710 100 82 90 0 200 210 82 0
PACKAGE 14 500 100 82 90 0 200 210 82 0
PACKAGE 15 710 300 82 90 0 200 210 82 0
PACKAGE 16 500 300 82 90 0 200 210 82 0
PACKAGE 17 290 100 82 90 0 200 210 82 0
PACKAGE 18 80 100 82 90 0 200 210 82 0
PACKAGE 19 290 300 82 90 0 200 210 82 0
PACKAGE 20 80 300 82 90 0 200 210 82 0
PACKAGE 21 710 500 82 90 0 200 210 82 0
PACKAGE 22 500 500 82 90 0 200 210 82 0
```

A.4 Dateiformat für Palettieranlagen

```
PLANT_FILE VERSION 1
; this is a plant file for 3D palletizing
;
; history
; since 13/1/99 read plate size for infeed and gripper
;
; GRIPPER <Type> <fill direction> <x> <y> <z> <a> <xl> <yl> <zl>
; light barrier positions <NumberOfLights> <x1> <y1> ..<xn> <yn>
GRIPPER FORK TOP_LEFT 0 0 0 0 320 320 20 1 10 10
GRIPPER VACUUM TOP_LEFT 0 0 0 0 320 570 20 4 -75 -155 75 -155 75 155 -75 155
;
; INFEED  <x> <y> <z> <a> <insertionedge> <plateXL> <plateYL> <plateZL> <tippertype>
; light barrier positions <NumberOfLights> <x1> <y1> ..<xn> <yn>
INFEED -208 1123 716 0 TOP_LEFT 600 600 50 TIPPER_NON 4 -20 1 -20 400 -265 1 -265 400
INFEED 170 1042 716 0 TOP_RIGHT 500 500 50 TIPPER_NON 4 -20 1 -20 400 -265 1 -265 400
;
; pallets
; PALLET  <x> <y> <z> <a> <insertionedge> <xl> <yl> <zl>
PALLET -330 559 4 0 BOTTOM_LEFT 1000 1200 144
PALLET 325 554 4 0 BOTTOM_RIGHT 1000 1200 144
;
; additional plant elements
CORNER_STUD_SIZE 100 10 100
```

A.5 Dateiformat für Packprogramme

Diese Datei ist die Schnittstelle zur Robotersteuerung. Die Steuerung liest die Datei und interpretiert die einzelnen Zeilen entweder einmalig oder zyklisch.

```
ROBOT_PALLETIZING_DATA VERSION 1
;syntax :
;NEWGRIPPEER PhysicalGripperNo PackageNo PackagesInX PackagesInY
; PackageBarcodeId InfeedNo Rotation
NEW_GRIPPER 0 0 3 2 200 210 82 80 0 0
GET -508 1333 798 0
  1  1  0
  1  1  0
PUT -620 -241 230 90
  0  0  0
  0  0  0
GET -508 1333 798 0
  1  1  0
  1  1  0
PUT -1040 -241 230 90
  0  0  0
  0  0  0
GET -508 1333 798 0
  1  1  1
  1  1  1
PUT -620 -241 312 90
  0  0  1
  0  0  1
PUT -620 -241 230 90
  0  0  0
  0  0  0
GET -508 1333 798 0
  1  1  1
  1  1  1
PUT -620 159 312 90
  0  1  1
  0  1  1
PUT -1040 -441 312 90
  0  0  0
```

```
  0  0  0
NEW_GRIPPER 0 1 2 1 200 210 164 90 1 0
GET -408 1228 880 0
  1  1
PUT -935 59 312 90
  0  1
PUT -1145 -141 312 90
  0  0
```

A.6 Roboterachsdaten der Versuchsanlage

Achsen	Motordrehzahl $[\frac{1}{min}]$	Getriebeübersetzung	Achsgeschwindigkeiten
1	4000	29 (Zähne=25)	1 [m/s]
2	2000	89	135 [Grad/s]
3	2000	89	150 [Grad/s]
4	3000	78	230 [Grad/s]

A.6.1 Berechnung der maximalen Achsgeschwindigkeiten für rotatorische Achsen

$$v_{max}[\frac{Grad}{s}] = \frac{Motordrehzahl[\frac{1}{min}] * 360}{\ddot{U}bersetzung * 60}$$

A.6.2 Skalierung des Beckhoff D/A-Wandlers

Maximale Ausgangsspannung: v_{max}= 10 Volt

D/A Wert (12Bit)	Spannung
0	$+V_{max}$
2048	0 Volt
4095	$-V_{max}$

Vorzeichenbehaftete Geschwindigkeit v im Bereich [0..2048]

$$v_{D/A-Wandler} = 2048 - v$$

Bei maximaler Ausgangsspannung wird der Motor mit Nenndrehzahl betrieben und die Achse somit mit maximaler Achsgeschwindigkeit verfahren ⇒ Skalierung auf maximale Achsgeschwindigkeiten!

A.6.3 Skalierung des Beckhoff Inkrementalgebers

Berechnung der Inkrementalgeberskalierung

$$Skalierungsfaktor[\frac{Inkr}{Grad}] = \frac{\ddot{U}bersetzung * Impulsvervielfachung * Z\ddot{a}hlerimpulse}{360}$$

Beispiel für Achse 4:

Übersetzung des Getriebes = 78

Zählerimpulse pro Umdrehung des Meßgebers = 1000

Impulsvervielfachung Meßgeber = 4

→ Skalierungsfaktor=866,666

Berechnung der aktuellen Achsposition in Grad

$$Position[Grad] = Skalierungsfaktor * Postion[Inkr]$$

Literaturverzeichnis

[1] Arbeitsgruppe Open Control: Open Control. http://www.interbusclub.com, (1998)

[2] unbekannt: Impressionen und Innovationen *Roboter und Automation*, (1996) 6, S. 30-31

[3] Hoppe, G.: Die Industrie setzt zunehmend auf PC - kompatible Steuerungstechnik *etz*, (1996) 15-16, S. 6 -10

[4] unbekannt: Realisierte Zukunftstechnologie *iee*, (1997) 1, S. 14 - 15

[5] Bechtloff, J.: Im Trio automatisieren *Elektronik*, (1992) 20, S. 104

[6] Happacher, M.: Das trojanische Pferd. *Elektronik*, (1998) 8, S. 2

[7] Hohenadel, J.: PC-basierte offene Steuerungssysteme. *Werkstattstechnik*, 88(1998) 1/2, S. 18–22

[8] Mertin, A.; Sieverding P.: Bloß nicht zu viele Interrupts. *Elektronik*, (1998) 8, S. 118–122

[9] Pritschow, G. u. a.: Tendenzen in der Steuerungstechnik und Antriebstechnik. *Werkstattstechnik*, 88(1998) 1/2, S. 13–17

[10] Schneider, G.: Feuerprobe bestanden. *Elektronik*, (1998) 8, S. 82–88

[11] Süss, G.: Weiche und harte Steuerungskonzepte. *Elektronik*, (1998) 8, S. 56–58

[12] Wollert, J.: Zeitweise. *Elrad*, (1997) 4, S. 71–73

[13] unbekannt: Durchgängig. *Elektrotechnik für die Automatisierung*, (1996) 3, S. 70–72

[14] Wollert, J.; Fiedler J.: Atp-Marktanalyse Echtzeitbetriebssysteme. *atp*, (1996) 1, S. 33–44

[15] GEE, D.: PC or Plc: What's in a name. *InTech*, (1995) 9, S. 48–51

[16] WALTER, A.: Gut gelernt ist halb gepackt. *Roboter*, (1994) 4, S. 34–37

[17] SVERIN, H. G.: Einsatz von Robotern im Materialfluß. *Neue Verpackung*, (1993) 6, S. 23–26

[18] SPEE, D.: Automatisches Palettieren nicht nur mit uniformen Größen. *Fördern und Heben*, (1996) 7, S. 516–517

[19] FOITZIK, B.: *CNC-Verpackungsmaschinen* : Moderne Industrie, 1995

[20] GRÄFENSTEIN, T.; FREY F.: Eine neue Verpackungsgeneration zum Palettieren (1993)

[21] STROMMER, W. M.: *Verfahren zum Palettieren von quaderförmigen Packstücken im beliebigen Sortenmix.* Berlin: Springer, 1992

[22] DIETZE, D.; KLAR A.: Palettieren mit Roboter. *Packung und Transport*, (1991) 7, S. 33

[23] LEE, E. C.: *Automated Palletization of Multiple Box Sizes.* Windsor University , 1990

[24] BERNARD, J.: Roboter in der Praxis - Neue Anwendungsbeispiele für das Palettieren. *Depalettieren und Kommissionieren*, (1988) 6, S. 117–128

[25] PENINGTON, A.; TANCHOCO J. M. A.: Robotic Palletization of Multiple Box Sizes *International Journal of Production Research*, (1988) 26, S. 95–105

[26] MÜLLER, K.: *Beitrag zur Optimierung der automatischen Palettierung - ein Verfahren zur Berechnung der Leistungsdaten von Palettieranlagen* Dr.-Ing. Dissertation Universität Berlin, 1972

[27] MENZEL, E. U. A.: Roboter auf Trab gebracht. *Neue Verpackung*, (1993) 6, S. 23–26

[28] SCHEPERS, J. *Exakte Algorithmen für orthogonale Packungsprobleme* Mathematisches Institut, Universität zu Köln, 1997

[29] WOTTAWA, M. *Struktur und algorithmische Behandlung von praxisorientierten dreidimensionalen Packungsproblemen* Mathematisches Institut, Universität zu Köln, 1996

[30] Bischoff, E. E. ; Marriott M. D.: A comparative evaluation of heuristics for container loading *European Journal of Operational Research* , (1990) 44, S. 267-276

[31] De Cani, P. *Packing Problems In Theory And Practice* Department of Engineering Production, University of Birmingham, 1979

[32] Daniels J. J. ; Ghandforoush P. An improved algorithm for the non-guillotine-constrained cutting-stock problem *Journal of the Operational Research Society*,41 (1990) 2, pp. 141-149

[33] Dowsland K. A. The three-dimensional pallet chart: An analysis of the factors affecting the set of feasible layouts for a class of two-dimensional packing problems *Journal of the Operational Research Society*,35 (1984) 10, pp. 895-905

[34] Dowsland K. A. Determining an upper bound for a class of rectangular packing problems *Computers and Operations Research*, 12 (1985) 2, pp. 201-205

[35] Dowsland K. A. A combined data-base and algorithmic approach to the pallet-loading problem *Journal of the Operational Research Society*,38 (1987) 4, pp. 341-345

[36] Dowsland K. A. An exact algorithm for the pallet loading problem *European Journal of Operational Research*, (1987) 31, S. 78-84

[37] Dowsland K. A. Efficient automated pallet loading *European Journal of Operational Research*, (1990) 44, S. 232-238

[38] Dowsland K. A. ; Dowsland W. B. Packing problems *European Journal of Operational Research* (1992) 56, S. 2-14

[39] Dyckhoff H., u. a. Standardsoftware für Zuschneideprozesse *Zeitschrift für wirtschaftliche Fertigung*,82 (1987) 8, S. 472-477

[40] Dyckhoff H. A typology of cutting and packing problems *European Journal of Operational Research*,(1990) 44, S. 145-159

[41] Exeler, H. Upper bounds for the homogeneous case of a two-dimensional packing problem *Zeitschrift für Operations Research - Methods and Models of Operations* ,(1991) 35, S. 45-58

[42] Exeler, H. *Das homogene Packproblem in der betriebswirtschaftlichen Logistik* Physica-Verlag Heidelberg, 1988

[43] Frerich-Sagurna R. ; Li H. Packmustergenerierung - Projektbericht und Perspektiven *OR Spektrum*, (1991) 13, S. 249-253

[44] ISERMANN H. *Ein Planungssystem zur Optimierung der Palettenbeladung mit kongruenten rechteckigen Versandgebinden* Diskussionsarbeiten der Fakultät für Wirtschaftswissenschaften der Universität Bielefeld, Nr. 155, 1985

[45] Isermann H. Ein Planungssystem zur Optimierung der Palettenbeladung mit kongruenten rechteckigen Versandgebinden *OR Spektrum*, (1987) 9, S. 235-249

[46] Nelißen J. Die Optimierung zweidimensionaler Zuschnittprobleme Schriften zur Informatik und angewandten Mathematik,(1991), Nr. 150

[47] NELISSEN J. *Neue Ansätze zur Lösung des Palettenbeladungsproblems* Verlag Shaker, Aachen, 1995

[48] NAUJOKS G. Neue Heuristiken und Strukturanalysen zum zweidimensionalen homogenen Packproblem *Operations Research Proceedings 1989*, Springer-Verlag Berlin Heidelberg, 1990, S. 257-263

[49] NAUJOKS G. Ein neuer Ansatz zur Bestimmung theoretischer Obergrenzen für das zweidimensionale orthogonale homogene Packproblem *OR Spektrum*, (1991) 13, S. 224-228

[50] SMITH A. ; DE CANI P. An algorithm to optimize the layout of boxes in pallets *Journal of the Operational Research Society*,(1980) 31, pp. 573-578

[51] SWEENEY P. E. ; PATERNOSTER E. R. Cutting and packing problems: A categorized, application-orientated research bibliography *Journal of the Operational Research Society*, 43 (1992) 7, pp. 691-706

[52] HEIMANN D. Ganzheitliche Betrachtung eröffnet Einsparpotentiale *Neue Verpackung*,(1995) 9, S. 26 - 30

[53] JACOBI, A. N. *Beitrag zur Parametrierung und Integration von Steuerungssimulationsmodellen* Verlag Hanser, München, 1994

[54] SCHWINN; WALTER *Grundlagen der Roboterkinematik* Verlag Schwinn, Schwalbach, 1992

[55] FELDMANN C.; DIETRICH *Repetitorium der Ingenieur-Mathematik, Teil 2* Verlag C. Feldmann,Springe, 1989, 3.Aufl.

[56] PAUL, RICHARD P. *Robot manipulators: Mathematics, programming, and control* MIT Press, Cambridge, 1981

[57] FEDROWITZ, C. *Ein PC - basiertes Off - line - Programmiersystem mit schneller Kollisionserkennung* Verlag VDI, Düsseldorf, 1994

[58] PRITSCHOW, G. U. A. *Roboteranwendung für die flexible Fertigung* Hanser, München, 1994

[59] EBACH, H. *Digitale Algorithmen zur Regelung und dynamischen Entkopplung von Industrieroboterachsen* Dissertation, Universität Erlangen-Nürnberg, 1985

[60] NORMENAUSSSCHUSS MASCHINENBAU *Programmiersprache Industrial Robot Language (IRL) DIN 66312* Beuth, Berlin, 1993

[61] SEDGEWICK, R. *Algorithmen in C/C++* Sybex, New York, 1993

[62] RECHENBERG P., POMBERGER G. *Informatikhandbuch* Hanser, München, 1997

[63] MAK, R. *Writing Compilers and Interpreters* Wiley, New York, 1991

[64] HEINER, V. Automatische Ladeplanung als logistisches Bindeglied *Hebezeuge und Fördermittel*, (1997) 6, S. 262-264

[65] DIETZE G. Auftragsbezogene Kommissionierung mit Portalroboter *Frauenhofer Institut für Materialfluß und Logistik* (1998), www.iml.fhg.de/Projekte/Projekte/Portal

[66] RO-BER INDUSTRIEROBOTER GMBH Südfeld 5, 59174 Kamen-Herren

[67] GRUPPO ELETTRIC 80 www.rcs.re.it

[68] MRM M.R.MÜLLER ROBOTERTECHNIK *Roboworker - Dreiachsportal für die Palettierung von Bauteilen* Hannover Messe, 1996

[69] SINGH, TED Introducing AIM PalletWare Adept technology, inc. San Jose, 1994, 14 Seiten

[70] KOIDE SEIJI, SUZUKI SHUNATOR, DEGAWA SADAO Palletize - planning system for multiple kinds of loads using GA search and traditional search.*Conference Artikel*, (1995)

[71] SHARP J.M.,DORE A.M., WINTEL S. Flexible robotic palletising of mixed size boxes. *conference Paper* INSPEC 5256841 C9606-1290F-065

[72] CHOI, PARK, LEE Event graph modelling of automated sorting and buffering *Journal Paper* International Journal of Computer Integrated Manufacturing 9 () 5, Seite 369-80

[73] AUTOMATISCH CHAOTISCH PALETTIEREN Frauenhofer Institut für Produkionstechnik und Automatisierung *www.ipa.fhg.de/300/320/Projekte/IPA/Projekt/ACP.htm*

[74] HALL, E. L. Intelligent Robot Trends for Factory Automation *www.eng.uc.edu/robotics/trends*,1998

[75] UNBEKANNT Chaotische Mischung *Lagertechnik*, (1997) 3, S.12 - 14

[76] UNBEKANNT Kommissioniertechnik *Lagertechnik*, (1994) 11, S.27

[77] H. K. TÖNSHOFF, T. KÜHN Per PC palettiert *Elektronik*, (1998) 24, S. 71

[78] H. K. TÖNSHOFF, T. KÜHN Software als Robotersteuerung unter Windows NT *wt Werkstattstechnik*,88, (1999) 1/2, S. 73

[79] H. K. TÖNSHOFF, T. KÜHN Vollautomatische PC-basierte Palettierung mit Mehrfachgreifern *wt Werkstattstechnik*,88, (1999) 2/3, S. 77

[80] UNBEKANNT S2000 Handbuch *Beckhoff*, (1993)

[81] SINGER, H. SPS - Ablaufsteuerung, programmierbar in der Sprache C *Kolloquim Softwareentwicklung - Entwicklung - Methoden, Werkzeuge, Erfahrungen* , (1995)

[82] TEMMES, J. Universal PC software for robot cells - still a dream *HDI - S3*, (1993) 25, S.161 - 168

[83] PRITSCHOW, J. U. A. Durchgängiger Einsatz von Splines in offenen numerischen Steuerungssystemen *WT - Werkstattstechnik*, (1998) 1/2, S.23 - 29

[84] VISSER, A; KOHLHAAS E. Zirkulare Splines zur Roboterprogrammierung mittels Standardinterpolatoren *WT - Werkstattstechnik*, (1990) 10, S.527 - 531

[85] STOTZ, C. Optimistisch in die Zukunft *MM Maschinemarkt*, (1996) 44, S.18 - 21

[86] BURKHARDT, G.; U. A. Neue Robotersteuerungs - und Programmiersysteme mit optimalem Preis-/Leisungsverhältnis *Produktionsautomatisierung*, (1994) 3, S.40 - 43

[87] SPUR, G.; ODER B. Spline-Verfahren für Fräsbearbeitung *unbekannt*

[88] WECK, M.; KLEIN F. Interpolation in der NC Bearbeitung *unbekannt*

[89] WECK, M. U. A. Werkzeugmaschinen und Fertigungssysteme Band 3.1 *VDI-Verlag* 1997

[90] SCHOLL, A. U. A. Besser beschränkt *ct*, (1998) Nr. 10, S.337 - 345

[91] KLAR, A. ; DIETZE, G. Zeitgerecht kommissioniert *Fördertechnik*, (1993) 12, S.12 - 14

[92] DU - MING TSAI *Modeling and analysis of three - dimensional robotic palletising systems for mixed carton sizes* Iowa State Univ. (Ph. D. Thesis),Ann Arbor, 1987

[93] MERTENS, P. *An automated palletizing System for industrial robots* Virginia Polytechnic Inst. State Univ. (M.Sc.Thesis),Blacksburg, 1985

[94] SAVOLDI, M. Flexible Palletizing *Robotics World*, (1995) Herbst, S.24 - 27

[95] UNBEKANNT Paralleles Palettieren *Fördermittel Journal*, (1998) 6, S.48 - 49

[96] UNBEKANNT Stapeln im Sekundentakt *Fördermittel Journal*, (1998) 4, S.51 - 53

[97] UNBEKANNT Näher ans Ziel *Fördermittel Journal*, (1994) 11, S.26 - 28

[98] UNBEKANNT Flexibel Palettieren *Fördern und Heben*, (1998) 5, S.359 - 360

[99] BUSCH, C. Gemischt zusammengesetzte Aufträge automatisch palettieren *Fördern und Heben*, (1998) 48, S.832 - 833

[100] UNBEKANNT Kommissionieranlage: Im Mittelpunkt steht der Roboter *Fördern und Heben*, (1995) 11, S.816

[101] SEVERIN, H. G. Roboter schaffen Bewegung *Produktion*, (1995) 29/30, S.16

[102] RUTHENBERGER, R. Sanft gehandhabt, schnell verpackt *Produktion*, (1997) 44, S.17

[103] WICHMANN G. Puzzle - Optimierung von Lade- und Transporteinheiten *Fraunhofer Institut für Materialfluß und Logistik*, (1997)

[104] MULTISCIENCE: Multimix Mischpalettenbeladung www.multiscience.com/prods/multimix, (1998)

Lebenslauf

Persönliches:

	Thomas Kühn
geboren am:	6.11.68 in Berlin
Eltern:	Leopold Kühn
	Ursula Kühn geb. Starke
Familienstand:	ledig

Schulbildung:

1975-1979	Grundschule Kirchdorf
1979-1981	Orientierungsstufe Spalterhals
1981-1984	Ganztagsgymnasium Barsinghausen
1984-1988	Gymnasium Bad Nenndorf

Studium:

1988-1994	Elektrotechnik an der Universität Hannover, Diplomzeugnis vom 9.3.1994

Berufstätigkeit:

8.1994 - 6.1999	Wissenschaftlicher Mitarbeiter am Institut für Fertigungstechnik der Universität Hannover