Aufgabenoptimale Roboterkinematiken

Wie können individuelle Roboter effizient ausgelegt und betrieben werden?

vorausrobotik

Carl-Buderus-Str. 7 | 30455 Hannover | Deutschland | **voraus**robotik.com

Kurzzusammenfassung Individuell ausgelegte Roboter bieten gegenüber den oft kompromissbehafteten, standardisierten Kinematiken zahlreiche Vorteile: Sie lassen sich leichter in bestehende Anlagen integrieren, sind optimal auf den Prozess abgestimmt und verkürzen Zykluszeiten. Dem gegenüber stehen erhöhte Aufwände in der Auslegung und im Software-Engineering, die die zuvor genannten Vorteile konterkarieren.

In diesem White Paper zeigen wir, wie sich die Hemmnisse konkret überwinden und so signifikante Effizienzgewinne realisieren lassen. An konkreten Beispielen erläutern wir praxisnah den Auslegungs- und Modellierungsprozess sowie die Inbetriebnahme – am Ende resultieren u.a. eine signifikante Effizienzsteigerung und eine um 25 % verringerte Zykluszeit.

1. Einleitung und Motivation

Im September 2025 gab die International Federation of Robotics bekannt, dass im vergangenen Jahr zum vierten Mal in Folge mehr als eine halbe Million Roboter weltweit ausgeliefert wurden (Quelle: https://ifr.org/ifr-press-releases/news/global-robot-demand-in-factories-doubles-over-10-years). Der Trend zur (robotergestützten) Automatisierung ist ungebrochen und wird durch verschiedenste Faktoren wie Onshoring und demographischer Wandel getrieben. Dabei dominieren die standardisierten Kinematiken großer Hersteller (nachfolgend auch als Standardroboter bezeichnet). Die Gründe hierfür liegen unter anderem in der bei den Kunden vorhandenen Erfahrung (Programmierung, Instandhaltung, Lagerhaltung für Ersatzteile etc.) und der globalen Verfügbarkeit. Im Gegenzug sind bei der Wahl des konkreten Typs häufig Kompromisse bezüglich Traglast, Platzbedarf, Taktzeit etc. einzugehen. Aufgaben- bzw. applikationsindividuelle Roboter können diese Nachteile egalisieren – dabei sind sowohl einfache Anpassungen von Segmentlängen und Achsabständen (sog. Maßsynthese) möglich, aber auch komplett neue optimierte Strukturen (sog. Struktursynthese). Damit lassen sich zahlreiche Vorteile im Vergleich zu Standardrobotern erzielen, von denen nachfolgend einige genannt sind: (a) Höhere Genauigkeit, Dynamik oder Steifigkeit können Prozesse erst automatisierbar machen oder deren Effizienz signifikant steigern – wie die im Fallbeispiel gezeigte Reduktion der

Zykluszeit um 25 %. (b) Geringerer Platzbedarf erleichtert die Integration in Anlagen und verringert die mit der Stellfläche zusammenhängenden Betriebskosten (Klimatisierung, Reinigung, Abschreibung des Gebäudes etc.). (c) Eine an die Traglast bzw. den Prozess angepasste Auslegung vermeidet Überdimensionierung und senkt Energiekosten (da weniger Massen bewegt werden müssen).

Obwohl also individuelle Roboter signifikante Vorteile in Anwendung und Kosteneffizienz aufweisen, sind diese bisher kaum verbreitet. Die Ursachen liegen unter anderem in den Aufwendungen für Auslegung (inkl. Konstruktion), Steuerungsintegration und Inbetriebnahme. Aus diesen Gründen finden solche Lösungen (noch) bevorzugt in Nischenanwendungen Einsatz, die eine entsprechend hohe Marge bieten oder anderweitig nicht (effizient) automatisierbar sind (z. B. wegen Dynamik- oder Genauigkeitsanforderungen des Prozesses).

2. Von der Auslegung zur Inbetriebnahme

Wie lassen sich die vorgenannten Nachteile individueller Kinematiken egalisieren und die wirtschaftlichen Vorteile heben? Hierzu betrachten wir zunächst die wesentlichen Entwicklungsschritte – von der Auslegung (Abschnitte 2.1 und 2.2) bis zur Inbetriebnahme (Abschnitte 2.3 und 2.4). Die Zusammenhänge legt Bild 1 dar. Im oberen Teil ist die Optimierungsschleife skizziert: Ein Roboter wird modelliert und in die Simulation eingefügt. In dieser sind der Prozess sowie weitere Komponenten der Zelle enthalten und die Applikation wird virtuell ausgeführt. In Abhängigkeit von den so gewonnenen Test- und Performance-Ergebnissen wird entweder der Roboter angepasst oder die Software-Bausteine werden auf die reale Anlage übertragen (unterer Teil von Bild 1).

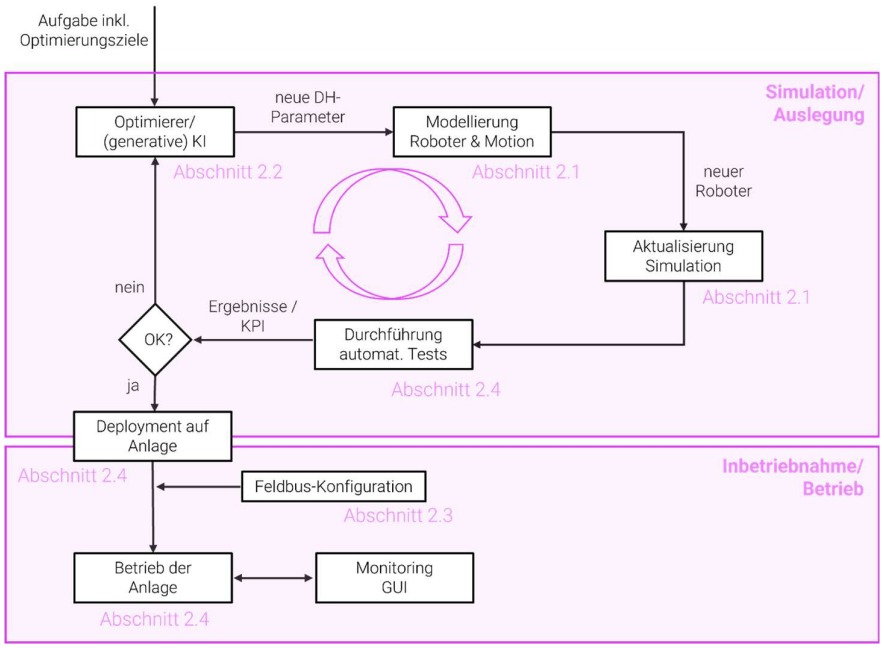


Bild 1: Prinzipielles Vorgehen zur Auslegung und zum Betrieb aufgabenoptimaler Roboter

2.1 Simulation, Modellierung und Applikation

Applikationsspezifische Roboter können ihre Vorteile naturgemäß nur dann vollständig ausspielen, wenn sie auf die Aufgabe (optimal) zugeschnitten sind. Dabei sollte der Prozess und die gesamte Anlage (zumindest jedoch die Komponenten, die in einem direkten Zusammenspiel mit dem Roboter stehen) gemeinsam betrachtet werden: Änderungen am Roboter (Geometrie, Stellfläche, Bahn) wirken sich auf umliegende Komponenten (erreichbarer Arbeitsraum, Störkonturen etc.) und den Prozess (z. B. Dynamik, Taktzeit) aus. Das Problem ist häufig, dass existierende Tools zur Simulation proprietär sind und nur Teile der Anlage beinhalten. Daher sind sie für diesen Anwendungsfall lediglich bedingt geeignet und stellen ein großes Hemmnis für einen effizienten Designprozess dar. Essenziell ist weiterhin, dass die Simulationsumgebung über offene Schnittstellen verfügt, die dem Auslegungsverfahren Zugriff auf die (virtualisierten) Komponenten und so die erforderlichen Freiheitsgrade für die Optimierung bietet. Dies ermöglicht nicht nur die Anpassung des Roboters selbst sondern auch die Auslegung der Zelle, wie bspw. die ideale Platzierung von Förderbändern – falls gewünscht.

Bevor wir uns der Optimierung (Abschnitt 2.2) widmen, müssen wir uns noch mit der Modellierung von Roboter und dessen Bewegungsplanung befassen. Wichtig ist: Das Verhalten des simulierten Roboters soll mit dem des realen möglichst exakt übereinstimmen – nur so lassen sich die berechneten Potentiale auch in der konkreten Umsetzung erreichen. Die in der Simulation verwendeten Modelle (oder allgemeiner die Berechnungsvorschriften) müssen daher auch in der Steuerung der Anlage eingesetzt werden; dies gilt insbesondere für die Bahnplanung – das Bewegungsverhalten des Roboters ist maßgeblich für z. B. Taktzeitoptimierung und Kollisionen.

Zunächst ist die Kinematik des Roboters aufzustellen – sie beschreibt den Zusammenhang zwischen Gelenkwinkeln (rotatorisch und translatorisch) und Roboter-Endeffektor (Spitze des am Flansch des Roboters angebrachten Werkzeugs). Dabei ist zwischen der direkten und inversen Kinematik zu unterscheiden: Direkte Kinematik bedeutet, dass, gegeben die Gelenkwinkel, die Position und die Orientierung (zusammen auch Pose oder Lage) des Roboter-Endeffektors berechnet werden können. Die inverse Kinematik löst das gegenteilige Problem: Aus der Lage des Endeffektors werden die zugehörigen Gelenkwinkel bestimmt. Ein gängiges Verfahren zur kinematischen Modellierung sind Denavit-Hartenberg (DH) Parameter. Diese beschreiben in effizienter Form (4 Stück je Achse) die relative Lage der Roboter-Achsen zueinander und erlauben so die Bestimmung der direkten und inversen Kinematik.

Diese Zusammenhänge sind Voraussetzung für die Berechnung von Arbeitsbereich und zahlreichen kinematischen Leistungsmerkmalen sowie der Pfadplanung. Verknüpfen wir den Pfad (die Menge aller Punkte/Orientierungen), entlang dessen sich der Roboter bewegt, mit der Zeit, so erhalten wir die Bahn (Trajektorie). Verschiedenste Methoden der Bahnplanung (bspw. Linear- und Kreisbewegung, ruckfreies Überschleifen zwischen

verschiedenen Bahnsegmenten) sind Voraussetzung für eine effiziente Prozessführung und Optimierung von Taktzeiten.

Kombinieren wir im nächsten Schritt die Bahn mit dynamischen Größen des Roboters (Masse und Trägheit einzelner Segmente, Reibung in den Antrieben etc.) so können wir direkte und inverse Dynamik bestimmen. Dynamik meint hier die Berechnung von Kräften und Momenten während einer Bewegung (und für die Mechaniker unter uns: auch im Stillstand). Diese Werte können wiederum zur Vorsteuerung der Antriebe verwendet werden, so dass eine exzellente Bahntreue auch bei sehr hohen Beschleunigungen erzielt wird. Die Kenntnis der Dynamik ist aber auch hilfreich bei der Auswahl der optimalen Motor-/Getriebe-Kombination, da sich mit ihr das Motordrehmoment bestimmen lässt.

Wir können festhalten: Das Aufstellen der beschreibenden Gleichungen für die Roboter erfordert Expertenwissen aus verschiedenen Bereichen, das nicht immer verfügbar ist. Jedoch ist deren automatische und (für die Optimierung) recheneffiziente Modellierung Voraussetzung für die Generierung aufgabenoptimaler Kinematiken.

Ein wichtiger Bestandteil der Simulation ist die eigentliche Applikation, da der Roboter nicht isoliert vom Prozess und den anderen Komponenten betrachtet werden kann. Die Simulation greift dabei auf die (virtualisierten) Komponenten der Zelle (z. B. Förderbänder, Greifer) zurück, die dieselben Schnittstellen wie in der realen Zelle aufweisen, und orchestriert sie. Modellierung, Simulation und Applikationsentwicklung erfolgen also im Idealfall in derselben Entwicklungsumgebung und in Hochsprache. Da zusätzlich alle Informationen der einzelnen Komponenten zentral verfügbar sind, wird das Erstellen der Applikation deutlich erleichtert und Kommunikationstotzeiten werden reduziert. Herstellerspezifische Silos, Tools und Programmiersprachen werden aufgebrochen. Neben den zahlreichen Bibliotheken zur Applikationserstellung stehen dann auch bspw. Physics-Engines wie PyBullet oder MuJoCo zur Simulation des Prozesses zur Verfügung.

2.2 Optimierung

In Abschnitt 2.1 ist deutlich geworden, dass zahlreiche Wechselwirkungen zwischen Roboter, Anlagenkomponenten und Prozess bestehen, die eine Optimierung erschweren und Expertenwissen voraussetzen. Eine gute Simulationsumgebung bietet einheitlichen Zugriff auf alle Komponenten und erlaubt eine holistische Betrachtung. Die nachfolgenden Ausführungen gelten unabhängig davon, ob die Auslegung mit „klassischen“ Verfahren oder mittels KI-Agenten erfolgt.

Betrachten wir den Roboter, so stellen wir fest, dass wir entweder die DH-Parameter optimieren können – in diesem Fall handelt es sich um ein typisches kontinuierliches Optimierungsproblem, wobei z. B. Kollisionen als Nebenbedingungen formuliert werden. Hierfür gibt es zahlreiche etablierte Lösungsverfahren (z. B. active set). Komplexer wird die Aufgabe, wenn zusätzlich noch die (diskrete) Anzahl und Art von Gelenken des Roboters bestimmt werden soll – man spricht dann von einem hybriden (diskret-

kontinuierlichen) Optimierungsproblem. Auch hier existieren Verfahren, die jedoch deutlich komplexer und rechenintensiver sind – in der Praxis wird man daher solche Problemstellungen eher vermeiden. Ein guter Kompromiss ist die automatische Auswahl von applikationstypischen Roboterstrukturen bspw. aus einer Bibliothek – im Nachgang können dann für jeden Typ ausgewählte Geometrieparameter wie z. B. die Segmentlängen als Teil der DH-Parameter optimiert werden. So erhalten wir eine einfache Problemstellung, die sehr gute Ergebnisse in kurzer Zeit liefert.

Zeitgleich zur Anpassung des Roboters ist auch die Bahn zu optimieren – sofern der Prozess dies zulässt. Dabei werden typischerweise zwei Aspekte unterschieden: Das Bahnprofil (also die Beschleunigungen und Geschwindigkeiten, mit denen der Roboter verfährt) und die Bahnpunkte, die die Geometrie festlegen. Hierbei handelt es sich erneut um ein kontinuierliches Optimierungsproblem mit Nebenbedingungen (wie max. Drehmoment, max. Geschwindigkeit der Antriebe, Position der Via-Punkte für die Bahn).

Fassen wir zusammen: Die Auswahl aus einer Roboter-Bibliothek mit anschließender Anpassung von Segmentlängen und Trajektorie ist aus Komplexitätsgründen zu bevorzugen. Diskret-kontinuierliche Optimierungsverfahren finden in der Praxis eher selten Anwendung. Soll die gesamte Zelle optimiert werden, können auch KI-Agenten unterstützen – sie haben dank standardisierter Schnittstellen und Programmierung in Hochsprache Zugriff auf alle virtualisierten Komponenten und sind so in der Lage, Anpassungen am Code und in den Modellen selbständig vorzunehmen und diese automatisiert zu testen (hierzu später mehr). Weitere Details zur Rolle der Simulation und dem Einsatz von KI-Agenten zur Optimierung im industriellen Umfeld sind unserem White Paper „KI Agenten in der Automatisierung“ zu entnehmen.

2.3 Antriebe und Steuerung

Die Mechanik applikationsspezifischer Roboter wird üblicherweise in kleinen Stückzahlen gefertigt. Anschließend sind die Antriebe (Motor, Getriebe, Leistungselektronik) mechanisch und steuerungstechnisch zu integrieren. Hier hat sich EtherCAT als Echtzeitfeldbus (in Kombination mit dem CiA 402 Standard) etabliert. Insofern sollte die Wahl des konkreten Lieferanten eine untergeordnete Rolle spielen, solange er sich an vorgenannte Standards hält.

In Abschnitt 2.2 werden optimierte Robotermodelle und -bahnen im Kontext der Applikation erzeugt. Dank der modernen (Simulations-)Tools stehen alle Softwarebausteine in Hochsprachen zur Verfügung. Können diese nahtlos – ohne Wechsel der Softwareumgebung und -sprache – auf die reale Anlage übertragen werden, so beschleunigt dies die Inbetriebnahme enorm. Eine Übersetzung in proprietäre (Roboter-)Sprachen und Bahnplanungsalgorithmen zur Ausführung in der Zelle führt darüber hinaus im Allgemeinen zu einem anderen geometrischen sowie zeitlichen Verhalten. Die in der Simulation gewonnenen Optimierungsergebnisse gehen so verloren. Aus diesem Grund sind klassische Steuerungsarchitekturen unter Einbindung von SPS nur bedingt geeignet. Zielführender sind hochsprachen- und

echtzeitfähige IPCs als einheitliche Steuerungsplattform und mit durchgängigem Motion-Stack – sowohl für die Entwicklung der Applikation als auch für die (echtzeitfähige) Steuerung aller Komponenten einschließlich Roboter via Feldbus.

2.4 Test, Deployment und Betrieb der Anlage

Im Idealfall erfolgen Auslegung des Roboters, Bahnplanung und Applikationsentwicklung ganzheitlich in der Simulation und ohne proprietäre Einschränkungen einzelner Hersteller. Lassen sich die realen Komponenten der Automatisierungszelle wie in Abschnitt 2.3 geschrieben durch die gleichen Schnittstellen und ohne Bruch in der Code-Basis ansteuern, so sind die Optimierungsergebnisse mit wenigen Klicks sofort in der Zelle ohne weitere Übersetzung ausführbar.

Um einen zuverlässigen Betrieb der Anlage zu gewährleisten ist die Software zuvor (selbstverständlich) ausgiebig und bevorzugt automatisiert zu testen. Geschieht dies erst während der Inbetriebnahme, wird die Fehlerbehebung teuer und zeitaufwändig – dies konterkariert die angestrebten Effizienzgewinne (siehe auch unser White Paper „Industrial DevOps“ zu den Kosten der Fehlerbehebung in den verschiedenen Projektphasen). Die automatisierte Testpipeline ist inhärenter Teil der Entwicklungsumgebung und baut auf den virtualisierten Zellenkomponenten auf. Sie kommt nicht nur zur finalen Prüfung vor dem Deployment (Aufspielen des Codes) auf die Anlage zur Anwendung, sondern kontinuierlich als integraler Bestandteil der Entwicklung – die generierten Ergebnisse und die Applikation sind jedes Mal unter anderem auf Kollisionen, Erreichbarkeit, korrektem Ablauf des Prozesses etc. zu prüfen. Es liegen dann nach Abschluss der Auslegungen für Roboter, den weiteren Zellenkomponenten und Applikation getestete Softwaremodule vor. Somit entsteht kaum Mehraufwand bei der Inbetriebnahme im Vergleich zu Standardrobotern – auch hier ist die Applikation zu testen und die Bahn bspw. auf mögliche Kollisionen zu prüfen. Dieser Vorteil ginge verloren, wenn der Code oder Teile davon vor dem Deployment in eine andere Sprache übersetzt werden müssten – die Testergebnisse würden dadurch drastisch an Aussagekraft verlieren; wie schon bei den Optimierungsergebnissen aus Abschnitten 2.2 und 2.3.

2.5. Fazit

Um die Vorteile optimierter Roboter wirtschaftlich zu nutzen, ist die Reduktion der Aufwände für die Auslegung und für das Software-Engineering der entscheidende Schlüssel. Gleichzeitig muss das erforderliche Robotik-Expertenwissen (siehe Abschnitt 2.1 und Abschnitt 2.2) auf ein Mindestmaß reduziert werden. Der industrielle DevOps-Ansatz (siehe Abschnitt 3.1) bietet hierfür ideale Voraussetzungen – Simulation und automatisiertes Testen sind integraler Bestandteil einer offenen und hochsprachenbasierten Entwicklungsumgebung. Soll die Auslegung des Roboters mittels KI-Agenten erfolgen, so ist dies ebenso problemlos möglich, da Infrastruktur, Schnittstellen und Modelle hierfür bereits existieren. Wir haben ebenfalls gesehen, dass

Roboter nicht isoliert betrachtet werden können – die Wechselwirkungen mit anderen Anlagenkomponenten und dem eigentlichen Prozess sind so umfangreich, dass eine holistische Simulation erforderlich ist. Auch dies ist bereits jetzt Teil der DevOps-Philosophie – proprietäre und zumeist auf einzelne Komponenten beschränkte Tools scheitern hier regelmäßig.

Der zweite große Schritt ist die Inbetriebnahme des Roboters – da wir in der Simulation schon alle erforderlichen Softwarebausteine erstellt und diese gegen die (virtuelle) Schnittstelle der Antriebe integriert und getestet haben, liegen die erforderlichen Komponenten für eine reibungslose Inbetriebnahme des Roboters vor. Voraussetzung ist, dass sich der Antriebslieferant an gängige Standards hält und diese sauber implementiert.

Auch für die weiteren Komponenten und die Prozesssteuerung liegt getesteter Code vor. Dieser kann mit wenig Aufwand auf einen zentralen Steuerungs-IPC gespielt werden, der sowohl die gesamte Anlage orchestriert als auch die Feldbuskomponenten in Echtzeit anspricht – eine zusätzliche SPS ist nicht mehr erforderlich.

3. Fallbeispiele

Nachfolgende Beispiele wurden mit der offenen Entwicklungsumgebung **voraus.pioneer** und dem containerisierten, hochsprachenfähigen Steuerungskern **voraus.core** umgesetzt. Zum besseren Verständnis und aus Gründen der Konsistenz, zunächst eine kurze Erläuterung der zugrundeliegenden Philosophie und eine Beschreibung der Funktionsweise (Abschnitt 3.1).

Ausgehend von Bild 1 zeigt Bild 2 das konkrete Zusammenspiel von voraus.pioneer und voraus.core. Das prinzipielle Vorgehen zur Auswahl bzw. zur Optimierung aufgabenspezifischer Roboter entspricht dem aus Abschnitt 2: Ausgehend von der Spezifikation der Aufgabe wird eine Roboter-Struktur erzeugt (oder aus einer Bibliothek ausgewählt). Anhand der DH-Parameter werden Kinematik und Motion berechnet (Abschnitt 3.2). Dadurch kann die Applikation in der Simulation gebaut und anhand automatisierter Tests geprüft werden. Sind die Performance-Kriterien erreicht und die Tests bestanden, wird der Code ohne Anpassungen auf die reale Anlage gespielt, die sofort betriebsbereit ist (Abschnitt 3.3 und Abschnitt 3.4). Falls nicht, nimmt der Optimierer bzw. die KI Modifikationen an der Kinematik vor oder wählt einen anderen Typ aus.

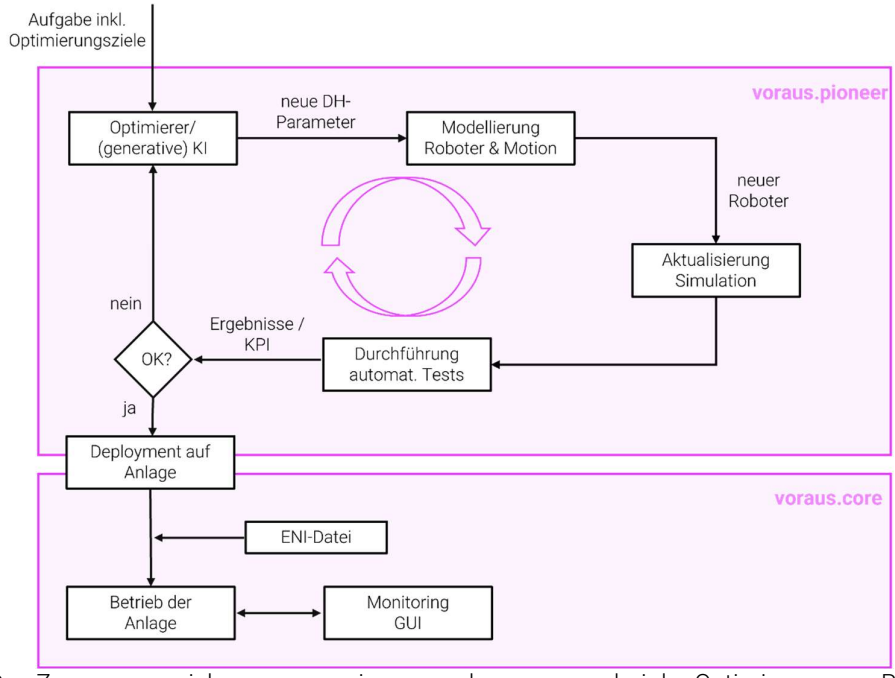


Bild 2: Das Zusammenspiel von voraus.pioneer und voraus.core bei der Optimierung von Robotern (oben) und deren Inbetriebnahme/Betrieb (unten)

3.1 Hochsprachensteuerung und offene Entwicklungsumgebung

Die beiden Software-Produkte voraus.core und voraus.pioneer setzen den Industrial DevOps Workflow um (Bild 3.). Ziel ist effizientes Entwickeln und Testen (linke Seite) mit Deployment, Betrieb und Überwachung der Anlage (rechte Seite), wie im IT-Bereich üblich, auch in der industriellen Automation (OT) zu etablieren. So werden schnelle Entwicklungszyklen bei hoher Software-Qualität möglich.

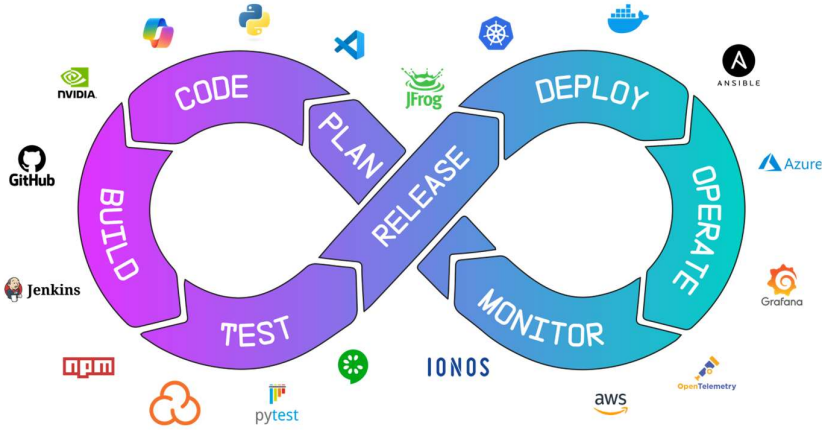


Bild 3: Industrial DevOps Workflow bei voraus.robotik inkl. beispielhafter Toolchain (Quelle Logos: Seiten der Anbieter)

Voraussetzung hierfür ist ein moderner, modularer und containerisierter (somit hardwareagnostischer) Steuerungskern (voraus.core), der die einzelnen Komponenten einer Zelle echtzeitfähig anbindet, z. B. über EtherCAT und PROFINET, gleichzeitig aber

deren Programmierung in Hochsprache ermöglicht, siehe Bild 4. Letztendlich bildet der voraus.core die Grundlage für die rechte Seite des in Bild 3 gezeigten Vorgehens.

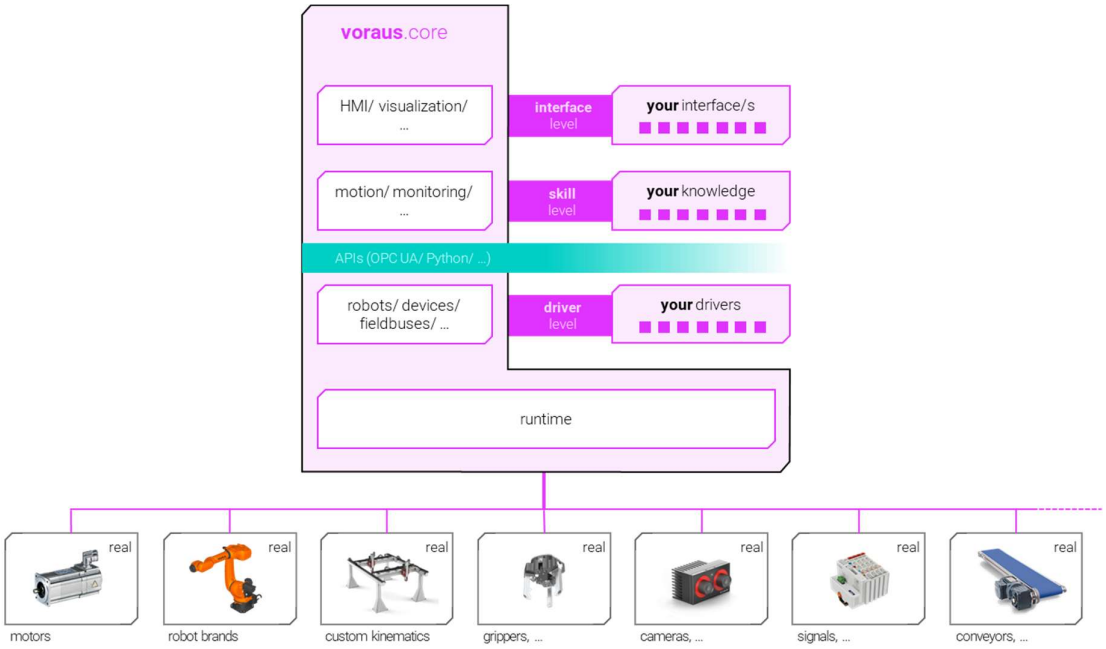


Bild 4: voraus.core – von der Feldbusebene bis zur Applikation in Hochsprache

Durch die Hochsprachenfähigkeit eröffnen sich für die Entwicklung völlig neue Möglichkeiten – komponentenübergreifend kann auf die leistungsfähigen und offenen (z. T. auch open source) Bibliotheken und Tools der IT-Welt zugegriffen werden; ein Vendor-Lock wird vermieden. Voraussetzung hierfür ist eine Virtualisierung der Zellenkomponenten unter Beibehaltung der Schnittstellen – diese stehen somit dem Entwickler bereits vor der eigentlichen Hardware zur Verfügung. Programmierung, Simulation und automatisiertes Testen werden von der Hardwareverfügbarkeit entkoppelt. Dies deckt die linke Seite des in Bild 3 gezeigten DevOps-Zyklus ab und wird bei voraus voraus.pioneer (siehe Bild 5) genannt. Dank der offenen Schnittstellen und Programmierung in Python oder Rust können existierende Bibliotheken zur Simulation (wie z. B. PyBullet, MuJoCo oder NVIDIA Isaac Sim für das physikalische Verhalten) oder Frameworks zum Testen (z. B. Jenkins) effizient eingebunden werden. Dies ist auch die Grundlage für nachfolgend beschriebene Optimierung und Auslegung. Wichtig ist, dass so Silos aufgebrochen und alle Komponenten einheitlich angesprochen werden; auch der Zugriff auf alle im System anfallenden Informationen und Daten ist sichergestellt. Das frühzeitige und ganzheitliche Testen in der Simulation ermöglicht es, Fehler früh zu erkennen und reduziert das Risiko eines Anlagenausfalls deutlich.

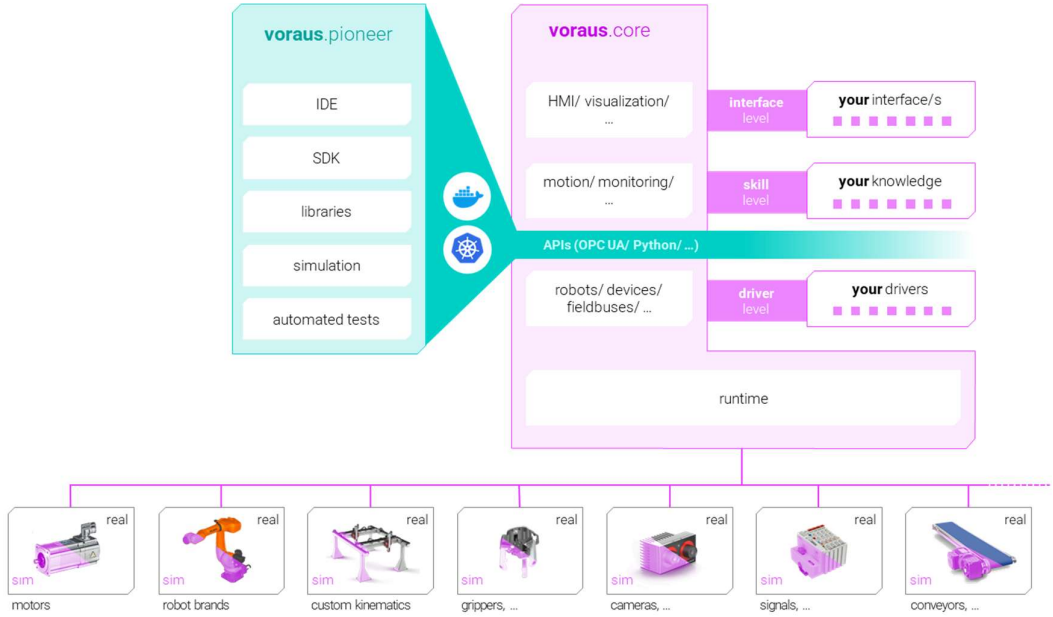


Bild 5: Der voraus.pioneer erlaubt das holistische Testen der gesamten Applikation in der Simulation und verwendet dieselben Schnittstellen zu den Komponenten wie die reale Anlage

3.2. Simulation und Modellierung

Wie eingangs erwähnt, soll die Modellierung einer individuellen Roboterkinematik so wenig Aufwand und Expertenwissen wie möglich erfordern. In Abschnitt 2.1 wurden die DH-Parameter, die auch die Grundlage für einen automatisierten und recheneffizienten Auslegungsprozess bilden, als eine etablierte Lösung kurz vorgestellt: Diese Parameter können dann bspw. einem Optimierungsalgorithmus oder einer KI als Freiheitsgrade zur Anpassung übergegeben werden. Das prinzipielle Vorgehen zur Modellierung sei im Folgenden an einer Portalkinematik (Bild 6) erläutert.

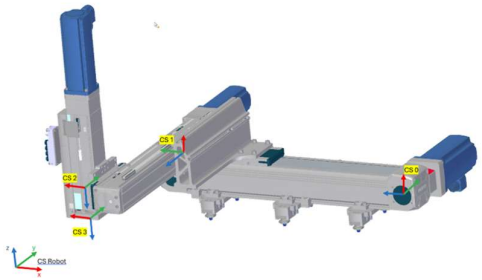
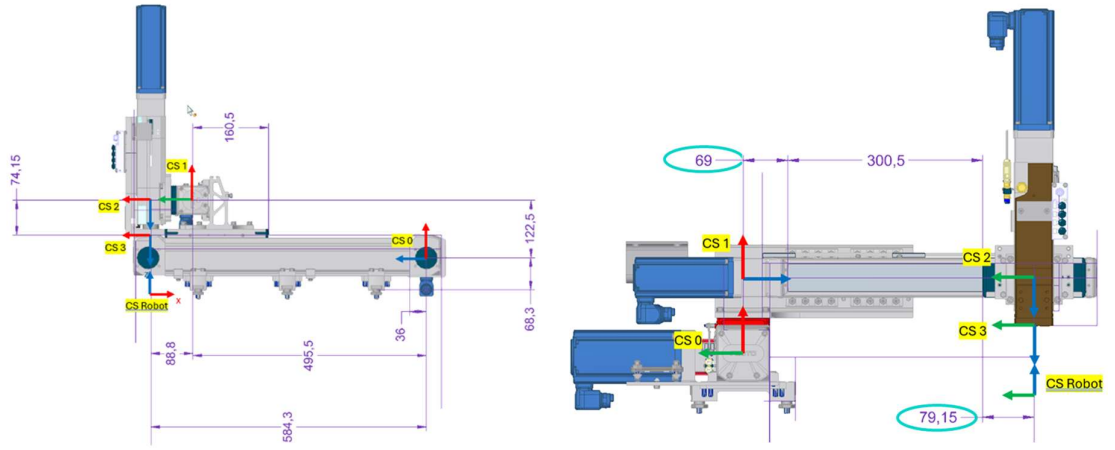


Bild 6: Betrachtete Portalkinematik mit drei Freiheitsgraden (x -, y - und z -Richtung)

Die DH-Parameter lassen sich direkt aus dem Robotermodell ablesen und sind tabellarisch dargestellt (Bild 7).

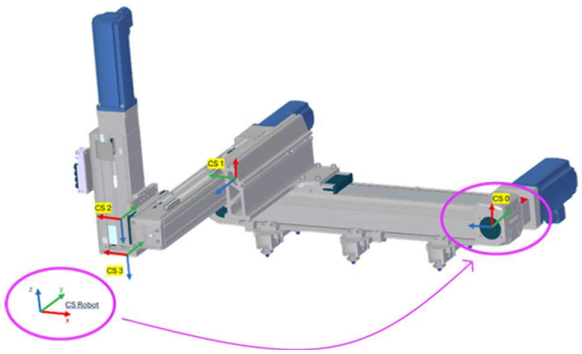


Joint	θ	d [mm]	a [mm]	α [rad]
1	0	q1 + 196.5	122.5	π/2
2	π/2	q2 + 148.15	88.8	-π/2
3	0	q3 + 74.15	0	0

Joint	θ	d [mm]	a [mm]	α [rad]
1	0	q1 + 196.5	122.5	π/2
2	π/2	q2 + 148.15	88.8	-π/2
3	0	q3 + 74.15	0	0

Bild 7: DH-Parameter und tabellarische Übersicht (auf die Definition der Koordinatensysteme sei hier nicht eingegangen)

Schlussendlich fehlt noch das Basiskoordinatensystem (siehe Bild 8). Dieses legt fest, wo sich die Roboterbasis in Bezug auf die Zelle befindet.



x [mm]	y [mm]	z [mm]	Rx [rad]	Ry' [rad]	Rz'' [rad]
584.3	448.65	68.3	0	-π/2	0

Bild 8: Definition des Basiskoordinatensystems

Diese Parameter werden in eine json-Datei übertragen (siehe Bild 9). Danach sind sowohl der voraus.core als auch der voraus.pioneer in der Lage, Roboter-Kinematiken inkl. direkter und inverser Kinematik automatisiert zu berechnen – ab hier ist kein weiteres Expertenwissen mehr erforderlich. Die Algorithmen sind sowohl in der Entwicklungsumgebung als auch in der Steuerung identisch – somit resultiert das oben geforderte gleiche Verhalten. Direkte und inverse Kinematik finden auch Anwendung im Motion-Stack, der unter anderem Bahnplanung, Überschleifen und verschiedene Arten von Stopp zur Verfügung stellt. Damit können Arbeitsraumanalysen, Erreichbarkeit und Pfade sowie Taktzeiten bestimmt werden. Versieht man die Kinematiken noch mit einfachen Hüllkurven, die mit den kinematischen Parametern skalieren (z. B. Länge der

Segmente), dann können in der Simulation bereits Kollisionen mit anderen Störkonturen berechnet werden. Dies bildet die Grundlage für eine vollautomatische Auslegung in der Simulation.

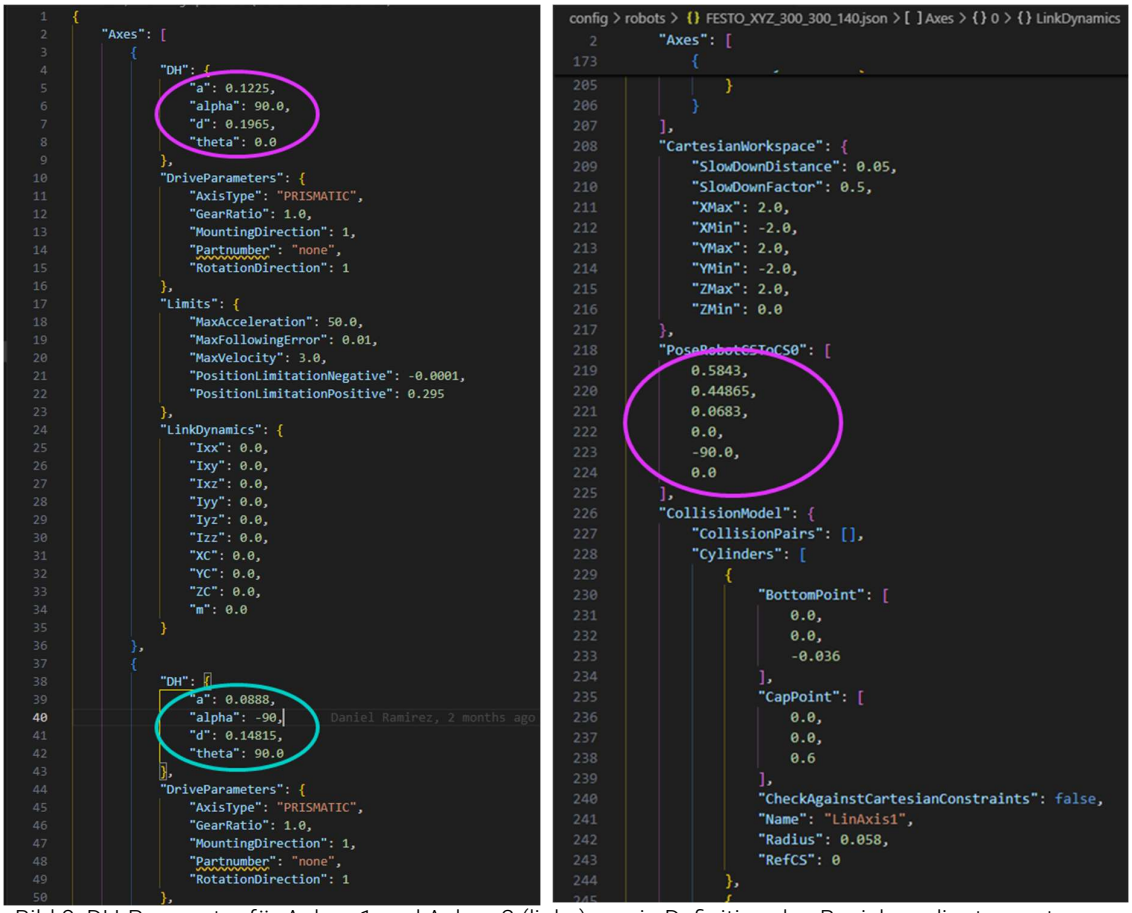


Bild 9: DH-Parameter für Achse 1 und Achse 2 (links) sowie Definition des Basiskoordinatensystems (rechts) in der Roboterkonfigurationsdatei

Zusätzlich ist es möglich – auch wenn eher für den tatsächlichen Betrieb des Roboters in der Zelle relevant – Massenträgheiten und Reibparameter anzugeben, die dann in der Dynamik und Regelung (z. B. Vorsteuerung für eine bessere Bahntreue) Anwendung finden.

3.3 Optimierung mittels Roboterbibliothek

Das in Abschnitt 3.2 beschriebene Vorgehen zur Modellierung erfordert, trotz der Simplifizierungen, immer noch Expertenwissen (wenn auch in geringem Umfang). Ein naheliegender Schritt ist daher der Rückgriff auf eine Bibliothek existierender Roboter, die vollautomatisch in die Simulation integriert werden. Es erfolgt dann keine individuelle Anpassung mehr, sondern vielmehr die optimale Auswahl eines bestimmten Typs. Da hier typischerweise die exakten Störkonturen und dynamischen Größen bereits hinterlegt sind, gewinnt die Simulation zusätzlich an Aussagekraft. Ein Beispiel ist die Kombination des voraus.pioneer mit dem autonox Finder der Firma autonox Robotics oder dem Roboter-Konfigurator der Firma igus.

Im Folgenden kommen Roboter der Firma autonox Robotics in einer Pick-&-Place-Anwendung zum Einsatz. Hierfür wird in einem ersten Schritt im voraus.pioneer die Zelle mit ihren relevanten Komponenten erstellt und die Applikation in Python programmiert (Bild 10). Der rechts gezeigte Roboter steht dabei exemplarisch für eine getroffene Auswahl aus dem Produktportfolio.

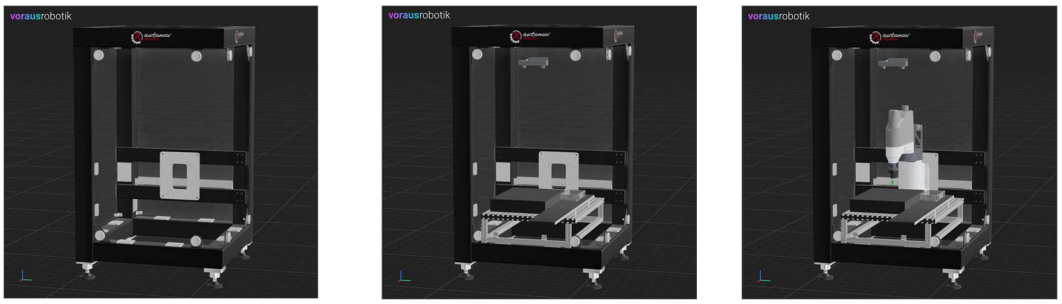


Bild 10: Schrittweiser Aufbau einer Pick-&-Place-Anwendung in der Simulation mit autonox-Kinematik

Das Optimierungsziel ist bei Pick-&-Place typischerweise die Minimierung der Zykluszeit für einen (oder mehrere) Picks. Gleichzeitig muss die Bahn kollisionsfrei und die Erreichbarkeit von Förderband und Boxen (mit den zu greifenden Gegenständen) sichergestellt sein.

Wird nun die Optimierung gestartet, wählt der Algorithmus einen Roboter aus der Bibliothek, aktualisiert Simulation sowie Bahnplanung und bestimmt die Zykluszeit. Diese kann dann über verschiedene Lagen der zu greifenden Gegenstände gemittelt werden. Der Bediener entscheidet sich in einem zweiten Schritt für die bevorzugte Konfiguration. So lassen sich auch Zellengröße/Boxanzahl und Roboter kombinieren, siehe Bild 11. Hier kann der Benutzer final wählen, ob er die zeitlich optimale Variante (rechts) bevorzugt, oder eine zukünftig ggf. erfolgende Erhöhung der Anzahl der Boxen bereits jetzt berücksichtigt (links) und dafür eine minimal längere Zykluszeit sowie einen größeren Zellenaufbau in Kauf nimmt. Letztendlich werden so Investitionsentscheidungen transparenter und der Return on Invest besser kalkulierbar.

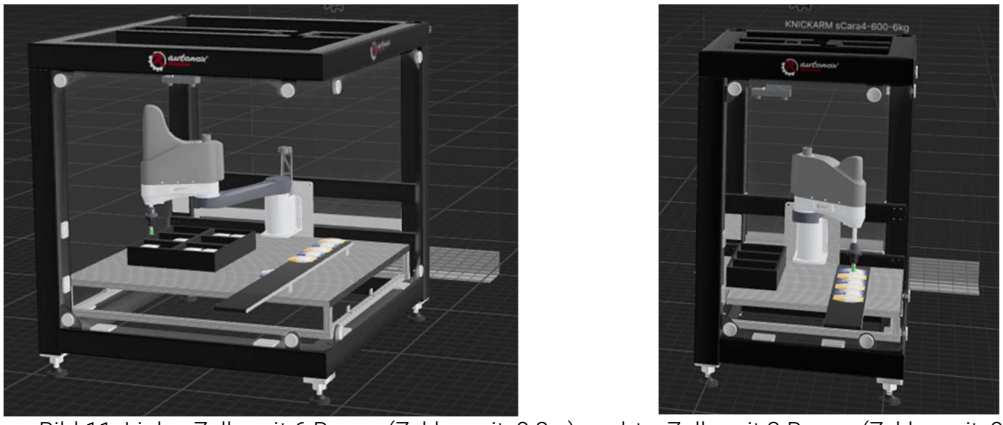


Bild 11: Links: Zelle mit 6 Boxen (Zykluszeit: 9,8 s); rechts: Zelle mit 3 Boxen (Zykluszeit: 9,4 s)

Selbstverständlich können auch Roboter verschiedener Hersteller verglichen werden – der voraus robotik Software-Stack ist aus Applikationssicht roboterunabhängig (siehe

Bild 12 – beispielhaft für autonox, Fanuc und KUKA). In der Simulation werden für den jeweiligen Robotertyp bzw. Hersteller Taktzeit, Erreichbarkeit, Kollisionen etc. bestimmt und die beste Lösung ausgewählt.

Darüber hinaus sind Applikation und Parametrierung strikt getrennt. Bild 12 zeigt dies exemplarisch für eine Palettierlösung mit Rollenförderer: Roboter unterschiedlicher Hersteller führen in verschiedenen Layouts dieselbe Aufgabe durch. Die jeweiligen Zellen lassen sich anhand individueller Kriterien automatisiert optimieren und testen: z.B. Roboterauswahl und -bewegung unter Taktzeitaspekten inklusive Kollisionsprüfung oder Position der einzelnen Komponenten zur Minimierung des Platzbedarfs. So ist bspw. bei der einfachen Pick-&-Place-Zelle eine Reduktion der Zykluszeiten von 3,4 s auf 2,5 s mithin also um mehr als 25 % auf Knopfdruck möglich – ein enormer Effizienzgewinn. Nach Auswahl der finalen Konfiguration wird die Zelle aufgebaut (Bild 12 links unten) und das Steuerungsprogramm (Roboter, Förderband, Einbindung Kamera etc.) auf Knopfdruck aufgespielt.

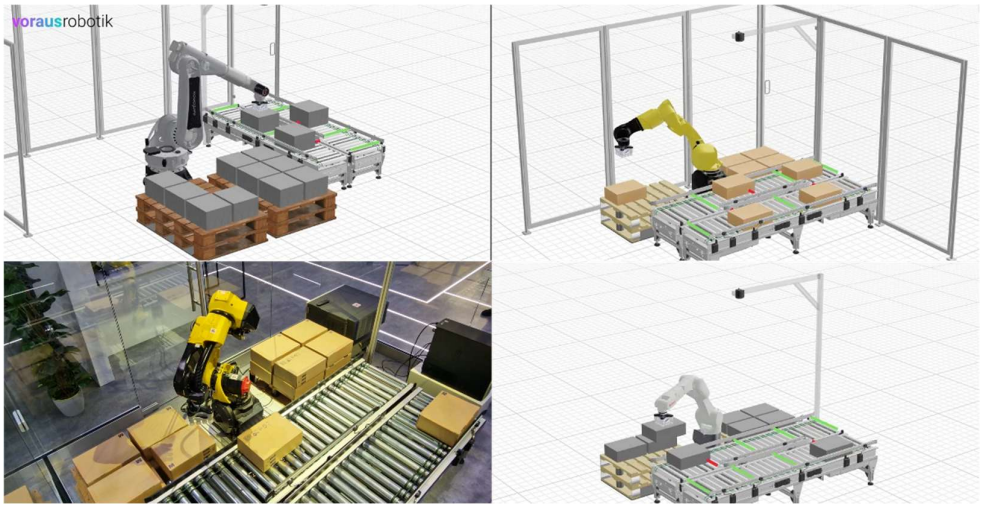


Bild 12: Roboter verschiedener Hersteller lassen sich in der Simulation in verschiedenen Zellenlayouts vollautomatisch vergleichen und so ein optimales Set-Up zu garantieren

Im Falle der oben gezeigten autonox-Beispiele sind nach Festlegung der optimalen Mechanik noch Antriebe und Steuerung zu integrieren – dies ist Gegenstand von Abschnitt 3.4. Sowohl Applikation als auch Roboteransteuerung können auch hier direkt für die reale Zelle verwendet werden – ein zusätzlicher Aufwand beim Deployment entsteht nicht.

3.4 Antriebe und Steuerung

Wie oben geschrieben, ist ein großer Vorteil des hier skizzierten Vorgehens die zentrale Steuerung aller Komponenten in gängigen Hochsprachen und ohne (proprietäre) Silos. Folgerichtig kommt auch für die Echtzeitkommunikation (Feldbus) keine SPS zum Einsatz, sondern ein kostengünstiger IPC. Auf diesem ist der voraus.core installiert, der über Software-Stacks der Feldbusprotokolle EtherCAT und PROFINET verfügt.

Im Folgenden betrachten wir die Einbindung der Antriebe mittels EtherCAT: Es wird in einem ersten Schritt die ENI-Datei (EtherCAT Network Information) generiert, die unter anderem die individuelle Netzwerktopologie beschreibt. Sind darüber hinaus die Antriebe CiA 402-konform, so ist beim voraus.core kein weiterer Integrationsaufwand vonnöten. Zusammen mit den Roboter-Parametern wird die ENI-Datei dem voraus.core übergeben und ein Zugriff auf die Antriebe des Roboters über Hochsprache ist sofort möglich (siehe Bild 13). Der voraus Motion-Stack stellt eine synchronisierte Mehrachsbewegung sicher und der Programmierer kann sich auf das Schreiben der Applikation konzentrieren, wobei ein zentraler und einheitlicher Zugriff auf alle Komponenten möglich ist.

Ein Dashboard bspw. ist problemlos über OPC UA integrierbar – dessen Erstellung erfolgt mit open source Tools (z. B. Grafana, pydash, rerun.io) in kürzester Zeit. Da der voraus.core in Hochsprache programmiert wird, ist ebenfalls die Einbindung in übergeordnete Steuerungssysteme (z. B. ERP, MES) keine Herausforderung.

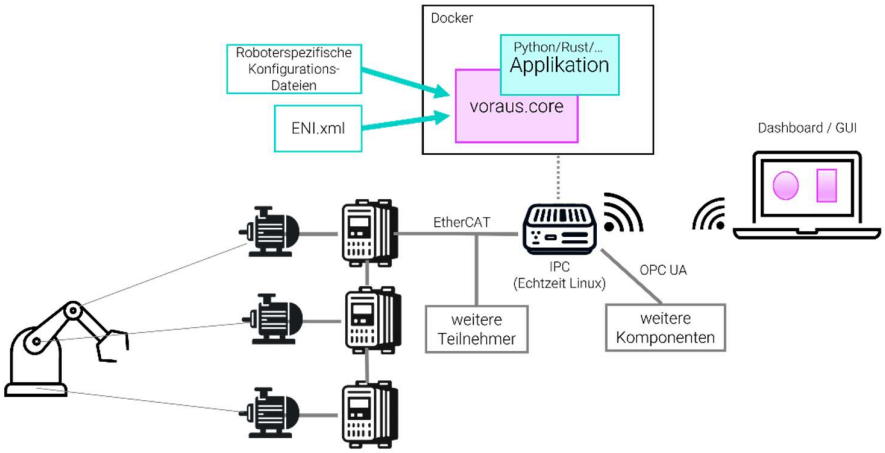


Bild 13: Integration der Roboterantriebe (ENI-Datei), der Roboterkinematik sowie weiterer Komponenten der Zelle (inkl. Dashboard/GUI)

Abschließend sei erwähnt, dass der einheitliche Zugriff auf alle Daten, die zentrale Steuerungsarchitektur und das konsequente Verwenden von Hochsprache für alle Komponenten eine wesentliche Voraussetzung für die Datenanalyse (z. B. Detektion von Anomalien) und für den Einsatz von KI-Agenten ist. Details hierzu sind im White Paper „KI-Agenten in der Automatisierung“ zu finden.

3.5 Test und Deployment

Dank der Abstraktion und Virtualisierung aller Komponenten einer Zelle kann die Programmierung unabhängig von der konkreten Auswahl und Verfügbarkeit der realen Hardware beginnen. Die Applikation selbst und insbesondere die Behandlung von Fehler- sowie Edge-Cases können in einer virtuellen Umgebung automatisiert abgesichert werden. Dies ist integraler Bestandteil der DevOps-Philosophie und kommt selbstverständlich auch bei der Optimierung bzw. Auswahl der Roboterkinematik zur Anwendung – ohne Mehraufwand. Die Testpipeline deckt alle Schritte zum Erstellen des

Releases inklusive statischer Code-Analyse, Durchführen von Unit-, Motion- und Docker-Tests sowie Erstellen der SW-Artefakte ab. Hierbei können auch applikationsspezifische Analysen wie Kollisionen sowie KPIs (z. B. Taktzeit) problemlos eingebunden werden. In jedem Optimierungsschritt werden dann die Tests automatisiert durchlaufen, um Code mit hoher Qualität zur Verfügung zu stellen. So lässt sich die Zeit für ein Update von Tagen oder Wochen auf wenige Minuten reduzieren, bei gleichzeitig signifikant geringerem Risiko.

Das Ergebnis der in Abschnitt 3.2 behandelten Kinematik ist in Bild 14 gezeigt – der in der Simulation getestete Code kann ohne Brüche in der Toolchain mit wenigen Klicks auf die reale Anlage gespielt werden, da nur ein zentraler Steuerungsrechner zu aktualisieren ist.

Bild 14: Individuelle Kinematik inklusive automatisiert getesteter Steuerungscode – reale Anlage und Simulation

4. Fazit und Ausblick

Individuelle Roboter aufgabenoptimal gestalten oder aus einem Katalog automatisiert auswählen – eine durchgängige Simulation ist der Schlüssel. So kann getestet werden, welche Auswirkungen eine geänderte Reichweite oder eine angepasste Bahn auf den Prozess und andere Komponenten der Zelle haben. Werden Simulation, Applikation und Steuerung der Anlage in derselben Hochsprache programmiert, so gestaltet sich das Deployment sehr effizient. Voraussetzung ist jedoch, dass eine umfangreiche und automatisierte Absicherung der erstellten SW-Bausteine bereits während der Entwicklung stattfindet. Die vorgenannten Aspekte sind integraler Bestandteil des Industrial DevOps-Workflows, sodass diesbezügliche keine Mehraufwände entstehen. Auslegung und der Einsatz individueller Roboter gestalten sich wirtschaftlich effizient.

Wie beschrieben, kann neben der Optimierung des Roboters auch die der Zelle bzw. Anlage einbezogen werden. Aufgrund der Komplexität des so entstehenden Optimierungsproblems ist dies ohne umfangreiches Expertenwissen kaum möglich –

hier spielen KI-Agenten eine wichtige Rolle: Erhalten sie Zugriff auf einen ersten Entwurf einer nicht-optimierten Zelle, so können sie Hypothesen aufstellen, in der Simulation testen und ggf. modifizieren. Im Ergebnis steht dann eine z. B. hinsichtlich Betriebskosten oder Taktzeit optimierte Gesamtanlage.

5. Unternehmensinformationen und Kontaktdaten

Weitere Informationen zu unserer Softwareplattform und wie wir Sie in der industriellen Automatisierung unterstützen können, finden Sie auf unserer Homepage www.vorausrobotik.com. Unsere Dokumentation mit vielen Beispielen gibt es hier: https://docs.vorausrobotik.com/.

Unsere Kontaktdaten

voraus robotik GmbH | Carl-Buderus-Str. 7 | 30455 Hannover | Deutschland
vorausrobotik.com

Ansprechperson

Tobias Ortmaier | tobias.ortmaier@vorausrobotik.com