Direct off-line robot programming via a common CAD package

備考

著者

Pedro Neto, Nuno Mendes

掲載

Robotics and Autonomous Systems

Abst

本論文では、一般的な3次元CADパッケージ上で動作するCAD図面から、直感的かつ直接的にオフラインでロボットをプログラミングすることに焦点を当てています。CAD図面上でロボットの動きを表現するための最適な方法、図面からそのような動きのデータを自動的に抽出する方法、 仮想 (CADモデル) から実環境へのデータのマッピング、ロボットの経路やプログラムの自動生成プロセスなどを探求しています。本研究の目的は、CADとロボット工学の基礎知識があれば誰でも利用できる、新しいCADベースのロボットプログラミングシステムを提示することである。 さまざまな操作タスクを用いた実験により，提案したアプローチの有効性と汎用性が示された.

1. はじめに

従来のティーチペンダントを用いたロボットプログラミングは、専門知識を必要とする面倒な作業であり、時間もかかる。多くの企業、特に中小企業では、ロボットやその他の自動システムの設定やプログラミングには時間がかかり、その分野の知識を持った作業者が必要なため、設備にロボットやその他の自動システムを使用していない[1]。それにもかかわらず，ほとんどの産業用ロボットは，いまだに従来のティーチングプロセスでプログラムされている．そのため，ロボットのプログラミングに

は，より直感的で新しいアプローチが求められている．実際、ティーチ ペンダントは直感的に使用することができず[25]、この問題に対する解決策を提示している著者もいる。これには，ロボットのティーチングプロセスに衝突回避や自動経路計画のメカニズムを導入することが考えられる[3,4]。オフラインロボットプログラミング (OLP) は、リードスルー方式 (2項参照) に比べて長所と短所があり、数年前から人気が高まっている[6-8]。本論文では、人間同士のコミュニケーション方法にヒントを得て、ロボットユーザーが、ロボット特有の言語から高い抽象度で、直感的にロボットと対話するための方法論を模索・研究しています。実際、人間はいくつかの異なる方法で教えることができます。例えば、絵を使って教えることができます。例えば、CADで描いた図面をもとに、人間が別の人間に何かを説明している光景はよく見られます。実際、CADデータは80年代からロボットのプログラミングにある程度の信頼性を持って使用されています (2.1項参照)。 本論文では、CADベースの OLP のための新しいシステムを紹介します (図1)。ロボットプログラムは、市販のOLPやCAMソフトウェアではなく、一般に入手可能な3次元CADパッケージ上で動作する3次元CAD図面から直接生成される。目的は、与えられたロボットセルの3次元CADモデル上のロボット経路のグラフィカルな記述から、ロボット動作シーケンス (プログラム) を自動的に生成することである。**CADとロボットプログラミングの手法を統一的に扱うことは、プラ ットフォームの汎用性と自律性の向上につながる可能性があり、言い換えれば、製品設計とロボットプログラミングをシームレスに統合することができます。 この研究では、ロボットの動きを CAD 図面で表現するための最適な方法、図面からそのような動きのデータを自動的に抽出する方、仮想 (CADモデル) から実環境へのデータのマッピング、ロボットの経路やプログラムの自動生成プロセスを検討します。** CADとロボット工学の基礎知識があれば、誰でも利用できるCAD ベースのOLPシステムを作ることが大きな目標です。現在のCADパッケージはかなり普及しており、比較的簡単に使用でき、価格も手頃なので、新しいロボットユーザーへの扉を開き、ひいては企業における既存のロボットの数を増やすことに貢献することができます。読者がこれまでの 研究を再現したり改良したりできるように、実行コード付きのアルゴリズムがいくつか紹介されています。また、さまざまな操作タスクに関する実験により、提案されたアプローチの有効性と汎用性が示されています。

2. オフラインでのロボットプログラミング

OLPは「完全自動」のプログラミングプロセスではなく、実際のロボットのシナリオをシミュレートするコンピュータソフトウェアを用いて、ロボットコードの手動編集やロボットプログラムの定義を行うこともあります。 OLPの主な利点は以下の通りです。

- ロボットの生産を停止／妨害することなく、ロボットをプログラミングできる (図2)。ロボットは設置前にプログラムされ、新しいタスクのために再プログラムされている間も生産を続けることができる[6]。つまり，ロボットのプログラミングは，ロボットの生産と並行して行うことができる (生産休憩が短縮される).

- プログラミングの作業は、ワークショップ (工場の現場) にいるロボットオペレーター から、オフィスにいるエンジニア／プログラマーに移されます。作業の安全性を高める。プログラミングプロセスの間、ユーザーはロボットの作業エリアに入ることはありません。
- ロボットプログラムは、シミュレーションツールを使ってテストすることができます。これは、実際のロボットの動作を予測し、作業プロセスを最適化するために非常に重要です。

その一方で、いくつかのデメリットも指摘できます。

- ソフトウェアや作業者のトレーニングなどの初期投資が比較的高い。この投資は難しいほとんどの中小企業にとっては正当化できない。
- ロボットのキャリブレーションに伴う誤差。ロボットのキャリブレーションには、非常に高価な計測用ハードウェア、ソフトウェア、技術的知識が必要です。
- タスクのキャリブレーションを行うには、経験豊富なオペレーターが必要です。タスクのキャリブレーションが不十分だと、ロボットの動作に大きな誤差が生じる可能性があります。
- オフラインで作成したロボットプログラムが正しく動作するかどうかを検証するためには、実際のロボットでテストする必要があります。その際、キャリブレーションのミスはロボットのクラッシュにつながります。
- 事前にプロセス情報が必要です。
- OLPの手法は、ロボットセルの正確なモデリングに依存します。

OLPに特化したソフトウェアパッケージは、通常、OLPソフトウェアまたはCAR (Computer Aided Robotics) ソフトウェアと呼ばれます。OLPパッケージの中には、異なるメーカーのロボットでも 動作するものがあります(ジェネリックOLPパッケージ)。代表的な汎用OLPパッケージとしては、Dassault Systèmes社のDelmia、Technomatix Technologies社のRobCAD、Jabez Technologies社のRobotmasterが挙げられる。これらのソフトウェアパッケージは、ロボットマニピュレータとそれに付随する機器をグラフィカルに表現し、プログラムを生成し、その結果、与えられたロボットタスクをシミュレートすることができる一連のモデリングおよびシミュレーションツールを提供している[7,8]。一方、ほとんどのロボットメーカーは、独自のOLPソフトウェアを持っています。例えば、KUKA社のKUKA.Sim、ABB Robotics社のRobotStudio、Motoman社のMotoSimなどである。OLPソフトウェアの初期バージョンは、ロボットの運動学の単純なワイヤーフレームモデルに基づいていた。しかし、近年のロボットシミュレーション技術は、コンピュータ技術やグラフィックアニメーション技術の進歩に伴ってか、リアルさが増し、人気が高まっている。現在のOLPパッケージは、より強力なグラフィック、モジュール化 (塗装や溶接などの特定のプロセスのためのモジュール)、標準化 (標準的な CADフォーマットを取り込む機能など) されています。これらの機能にはコストがかかります。

OLPソフトウェアのライセンスには数千ユーロかかることもあり、ほとんどの中小企業にとっては正当化しにくい投資です。OLPソフトウェアの利点は、既存のシステムのいくつかの制限によって抑えられています。実際、これらのシステムは直感的に使用することができず、ロボットの周囲の環境が

先験的に分かっていて、よくモデル化されている状況でのみ適用することができる[9]。さらに、高い絶対位置精度は、正しくキャリブレ ーションされたロボットによってのみ達成される[10-12]．ロボットのポーズ (位置と向き) を手動で教示する場合、再現性は重要な要素であり、位置決め精度は重要ではない[12]。一方，OLPでは，ロボットの経路が与えられた座標系に対して仮想空間で定義されるため，位置決め精度が極めて重要な要素となる．産業用ロボットの位置決め精度は、メーカー、年代、ロボットの種類によって異なる。誤差の大きさは、10分の1ミリ程度のものから数センチのものまであります。適切なキャリブレーションを行うことで、この誤差を1ミリ 以下に抑えることができます。国際規格である ISO9283 では、正しいキャリブレーションの手順が推奨されています。ロボットのキャリブレーショ ンには、さまざまなハードウェアや技術が適用されています。例えば、ROSYシステムは、キャリブレーションボールとデジタルカメラを使用して、運動誤差とその結果としての補正値 (補正パラメータ) を計算します[10]。また、別の研究では、ABB IRB 1600 産業用ロボットの精度が、29パラメータの校正モデルを用いてどのように改善されたかを示している[11]。計測にはレーザートラッカーを使用しています。ほとんどのロボットメーカーは、ロボットの校正サービスを提供しています。

2.1. CADによるロボットプログラミング

近年、CAD技術は経済的にも魅力的で、簡単に扱えるようになりました。今日、世界中の何百万もの中小企業が、製品の設計やモデル作成にCAD技術を利用しています。とはいえ、CAD業界は今後、重大な技術的課題に直面しなければなりません[13]。

80年代にはすでに，CADはロボット工学の発展に役立つ技術と見なされていた[14]．それ以来，CADを利用したロボットの計画とプログラミングの分野では，さまざまな研究が行われています．長年にわたり，一部の研究者は，CAD技術の機能をロボット工学分野に拡張しようと模索してきた．今日では、CADの図面/ファイルから情報を抽出して、さまざまな用途のロボットパス/プログラムを生成することが可能です[15,18]。

ロボットと人間の間のインターフェースとして、CADを用いた一連の研究が行われています。スプレー塗装やコーティングのプロセスには，様々なソリューションが提案されている．Chenらは，スプレー塗装のための CAD ベースのロボット経路計画のレビューを行っている[19]．自動車製造でよく見られる複合面のスプレー塗装のプロセスのために，CADガイド付きのロボットパスジェネレータが提案されている[20]．ArikanとBalkanは、曲面のスプレー塗装プロセスに対処するCADベースのロボットシステムを提案している (OLPとシミュレーション) [21]し、Chenらは、自由形状の表面のスプレー塗装のためのCADベースの自動ロボット軌道計画システムを提案している[22]。CADベースのロボティクスの分野における重要な研究では，靴のアウトソールとアッパーのためのロボット接着剤スプレーシステム用の3Dロボット作業パスを生成する方法が紹介されている[23]．ロボットとCADの汎用性を活かした新しいプロセスの例として、金属板のいわゆるインクリメンタル・フォーミング・プロセスが挙げられる。コストのかかるフォームを使用せずに、金属板を剛体フレームにクランプし、ロボ

ットが高周波振動スタンプを備えたツールを金属表面上に誘導することで、所定の3D輪郭を作り出す。ロボットの軌道は、CADモデルから特定の材料モデルに基づいて計算されます。プロトタイプパネルやカスタマイズされた自動車のパネルは，この方法で経済的に製造することができる[24]. Pulkkinenらは、金属プロファイルをロボットで加工し、ワークピースの2次元幾何学的表現しか利用できないアプリケーションのためのロボットプログラミングコンセプトを発表している[25]。

永田らは、ロボットの経路をCAD/CAMソフトウェアで生成するロボットサンディングプラットフォームを提案している[26]。ロボット言語を使わずに産業用ロボットがCLデータに沿って移動できるロボット CAD/CAM システムが永田らによって発表されている[27,28]。最近の研究では、ラピッドプロトタイピングアプリケーションのためのCAMソフトウェアからのロボットパス生成について議論されている[29]。Feng-yun and Tianshengは、CAD システムのポストプロセッサから CL データが生成される研磨プロセスのためのロボットパスジェネレータを発表している[30]。その他の先行研究では、CAD図面から切削データを抽出するラピッドプロトタイピング用のロボットシステムの開発が報告されている[31,32]。また、CADデータからバリ取りパスを生成するCADベースのシステムがMurphyらによって提案されている[33]。SallinenとServioは、CAMソフトウェアから製造パスを生成し、ロボットベ ースのシステムを使用してプロトタイプの鋳物を製造する方法を提案している[34]。異なる種類のアプリケーションでは，ロボットのナビゲーション目的で，すなわち大規模な経路計画のためにCAD図面が使用されている[35]. これまで見てきたように、CAD、CAM、VRMLベースのOLPの分野では様々な研究が行われています。**しかし、これまでの研究では、生のCADデータを使用し、市販のCADパッケージと直接インターフェースすることで、直感的で低コストのOLPソリューションを実現する有効なソリューションはありませんでした。**
この分野の研究は大きな成果を上げており、そのうちのいくつかはすでに産業界で実装されていますが、特定の産業プロセス（溶接、塗装など）に限定されています。様々なアプローチが提示されていますが、費用対効果の高い標準的なソリューションはまだ確立されていません。

3. CADベースのアプローチ

3.1. CADパッケージ

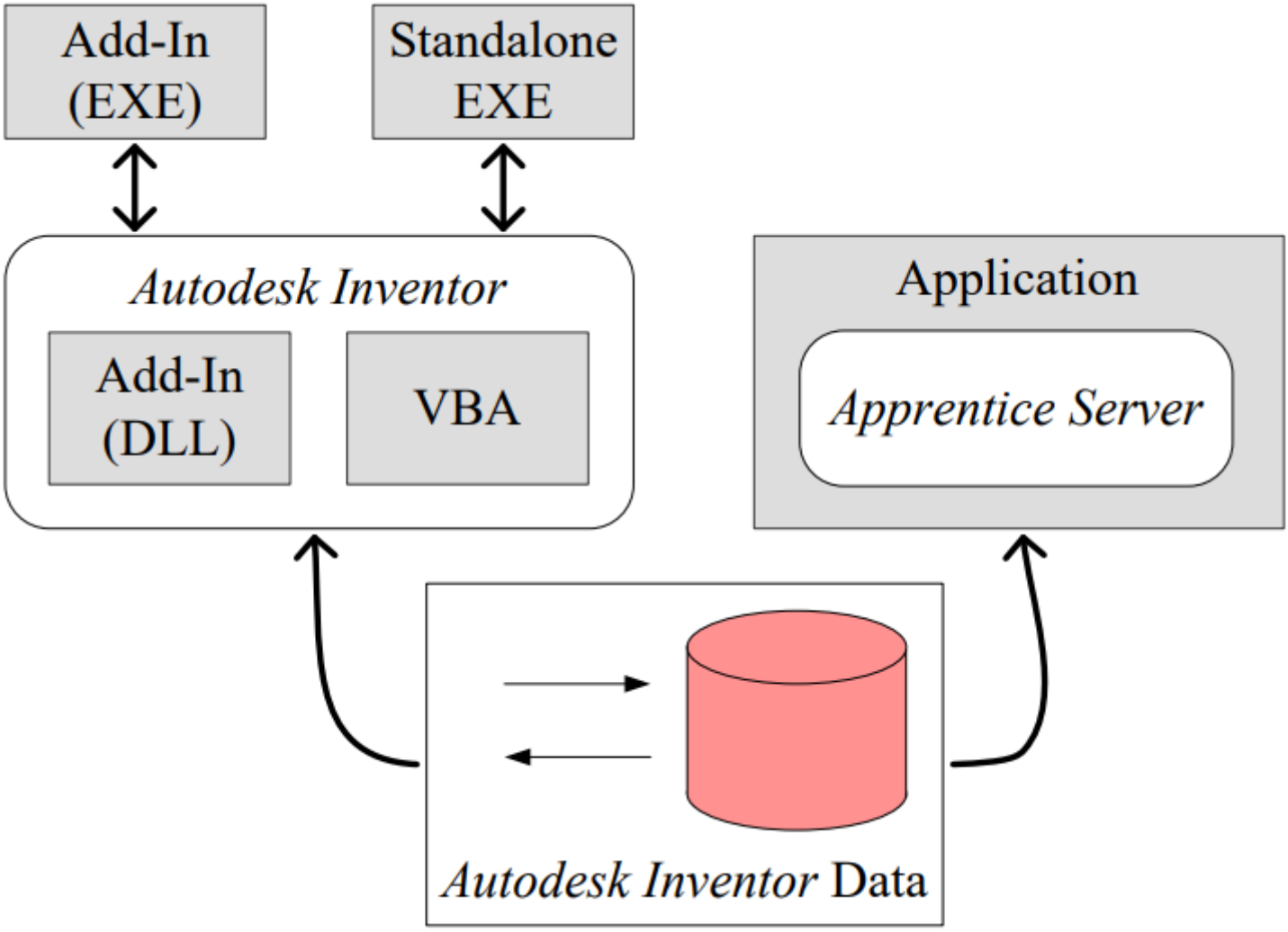
CAD技術は、経済的にも魅力的で扱いやすくなり、今日では世界中で何百万もの企業が製品の設計やモデル作成にCADを使用しています。CADパッケージの価格が下がる一方で、ユーザーインターフェースの改善や簡素化、ユーザー指向の機能、標準製品の自動設計など、機能や性能が向上してきました。現在では、ほとんどのCADパッケージが幅広い関連機能 (統合モジュールまたはスタンドアロンソリューション) を提供しており、効果的な設計プロセスに役立つだけでなく、機械的シミュレーションや動的プロセスの物理的シミュレーションなどの他のタスクにも役立ちます。ロボットのプログラミングとシミュレーションは、CADパッケージが統合できるもう一つの機能と見なされています。

提案されたソリューションのインターフェースとして，現在最も一般的な3次元CADパッケージの1つであるAutodesk Inventorが選ばれた.

Autodesk Inventorは，設計，ビジュアライゼーション，シミュレーション，ユーザーフレンドリーなインタ ーフェイスなど，最新のCADパッケージの機能をすべて備えており，カスタマイズ用の完全なアプリケーション・プログラミング・インターフェイス (API) を提供しているため，開発者はCADベースのアプリケーションをカスタマイズすることができます[36]．ファイル・フォーマットに関しては、すべての標準フォーマットの他に、Autodesk Inventorには、シングル・パート・モデル・ファイル（iptファイル）とアセンブリ・モデル・ファイル（iamファイル）を定義するための独自のファイル・フォーマットがあります。

3.2. CAD図面からのデータ抽出

提案するCADベースのOLPプラットフォームのベースとなるのは、Autodesk Inventor上で動作するCAD図面からロボットの動作データを自動的に抽出する機能である。そのために、Autodesk Inventor APIが使用されています。Autodesk Inventor APIは、Autodesk Inventorの機能を、Microsoft社のAutomationと呼ばれる技術を用いて、オブジェクト指向で公開しています。このようにして、開発者は Visual Basic (VB)、Visual C#、Visual C++ などの現行のプログラミング言語を使用して Autodesk Inventor と対話することができます。このAPIにより、開発者は、ユーザがAutodesk Inventorを対話的に使用する際に実行できる操作と同じタイプ (種類) の操作を行うソフトウェア・インタフェースを作成することができます。要約すると、APIは、Autodesk Inventorのリソースに基づいてソフトウェア・インタフェースを構築するために使用できるルーチンのセットを提供します。



Autodesk Inventor の API にアクセスします。

Autodesk Inventor API にアクセスするには、さまざまな方法があります (図3)。白いボックスはAPIが提供するコンポーネント (Autodesk InventorとApprentice Server)、灰色のボックスは開発者が書いたプログラムを表しています。あるボックスが別のボックスを囲んでいる場合、囲んでいるボックスが、囲んでいるボックスと同じプロセスで実行されていることを示している。したがって、「プロセス内」のプログラムは、プロセスの外で実行されているプログラムよりも大幅に速く実行されます。本論文では、APIとそれに続くInventorのデータにアクセスするためのスタンドアロン・アプリケーションを提案します。この選択は、メイン・アプリケーションに、CADとのインタラクション・プロセスだけでなく、他のタスクのための他のソフトウェア・コンポーネント (例えば、ロボット・コミュニケーション) も統合する必要があったためです。

オートデスクが提供するAPIには、ロボット工学での利用を想定した機能が多数用意されている。例えば、スタンドアロンのアプリケーションで、Autodesk Inventorを可視モードで開き (図4)、Inventorのドキュメントを開くことができます (図5)。ドキュメントのプロパティにも簡単にアクセスできます (図5)。

Algorithm 1. Opening *Autodesk Inventor* (coded in VB).

```
' Get object.
Private oApp As Inventor.Application
Try
    oApp =
    System.Runtime.InteropServices.Marshal.GetActiveObject("Inventor.Application")
Catch ex As Exception
    MessageBox.Show("A problem occurs")
End Try
' Open Inventor (oApp).
Dim InventorAppType As Type = System.Type.GetTypeFromProgID("Inventor.Application")
oApp = System.Activator.CreateInstance(InventorAppType)
' Make Inventor visible.
oApp.Visible = True
```

図4: Autodesk Inventorを開く(VBでコーディングされています)。

Algorithm 2. Opening an *Autodesk Inventor* document (coded in VB).

```
' Open an Inventor document (InventorDoc).
Private oApp As Inventor.Application
Private InventorDoc As Inventor.Document
InventorDoc = oApp.Documents.Open("document name")
' Properties of the document.
Dim oPropsets As PropertySets
oPropsets = InventorDoc.PropertySets
```

図5: Inventorのドキュメントを開き、そのプロパティを抽出する (VBでコード化).

CAD図面から抽出できるデータは非常に多くあります。問題は、目的(OLP) を達成するためにどのようなデータが必要かということです。**実際には、CADからロボットのモーションを取得する必要があります。** すなわち、既知の座標系 (デカルト空間) に対するロボットのエンドエフェクタの姿勢を表す一連の目標点です。このように、Autodesk Inventor APIの能力を考慮すると、あるロボットセルを表す適切なCAD図面から、3D空間におけるオブジェクトの位置と方向を抽出する必要があることがわかりました。

1. 位置
   位置データは、CAD図面から様々な方法で取得することができます。例えば、図6のワークポイ

ントの位置データ (CAD図面内に配置可能なポイント) を取得することができます。他の状況では、位置データは、CAD図面上の仮想ロボットパスを表す様々な線のそれぞれを特徴づける点から得られます (図7)。例えば、ロボットの経路がCAD図面上のスプラインの形状を想定している場合、APIはそのような形状を定義するために必要なすべてのポイントを提供します。これらのデータはすべて、ロボットセルのCADアセンブリモデルの原点に関連して定義されます。

2. 姿勢
   APIは、CADアセンブリモデルで表現された各部品モデルの変換マトリクス (または同次変換) に関する情報を提供します (図8)。変換行列には、回転行列と、それが参照する部品モデルの原点の位置が含まれており、いずれもロボットセルのCADアセンブリモデルの原点に対するものです。

3.7. ロボットプログラムの生成

機械をプログラムするための新しい、より直感的な方法を求めて、機械コードを生成する技術が登場しました。ここ数十年の間に、いくつかのコード生成技術が開発された。最も顕著な例は、市販のCAD/CAMシステムを使用してCNC加工用の信頼性の高いCLデータを生成することです[41]。CNCのツールパスも、標準的なCADのフォルマントから生成することができます[42-44]。しかし，これらのコード生成システムには，異なる状況から一般化する能力や，見知らぬ問題に対応する能力など，いくつかの欠点がある傾向がある．コード生成のアルゴリズムを練る際、キーワードは「一般化」であり、決して「特定化」ではない。アルゴリズムは、プロセスの幅広いバリエーションをカバーするために準備されなければならない。プロセスのバリエーション の数が限られていてよく知られている特定のアプリケーションでは、この種のアルゴリズムは許容できるパフォーマンスを示します[45]。

ロボットコントローラに特化した言語は、ここ数年でわずかな進歩しか見られない。一部の著者は，タスクに沿ってロボットプログラムを汎用化できると同時に，必要に応じてカスタマ イズできる方法論の作成に注目している[46]．操作は、ロボット操作のタイプやワークピースの形状などの観点からカスタマイズすることができる。本質的に、これにより、プログラマーの経験やプロセス知識を取り入れて、過去の類似作業から利益を得ることができる[47]。したがって，関連する製品やタスクのためのロボットプログラムを作成する時間を短縮し，専門家ではない人でもロボットプログラムを作成することができるようになる。これらのシステムは、コンピュ ータサイエンスの世界でよく知られているマクロやスクリプトと同じ論理に従っている。また，ロボットプログラミング言語の翻訳者も問題となっており[48]，特定のロボット言語を使用せずに動作するロボットシステムの開発も行われている[27-28].

本稿では、CAD図面から抽出した情報を用いてロボットプログラムを自動生成するアルゴリズムを提案する。ロボットコードを生成するプロセスの適用方法は、対象となるロボットのタスク によって異なる。しかし、すべてのロボットプログラムには共通する点がある。それは、ロボットは通常、マニピュレーションのタスクを実行するので、ロボットプログラムを生成するプロセスはアプリケーショ

ンによって大きな違いはなく、図17に示すような、掴む、動かす、置くといった共通のタスクを含んでいるということである。

ロボットプログラムの自動生成は、ロボットコマンドをテキストファイルに一行ずつ書き込んでいくことに他ならない。本論文では、このプロセスを、CAD図面からデータを抽出し、そのデ ータを解釈し、最終的にロボットプログラムを生成するソフトウェアインターフェース (3.8節) によって管理している。ロボットプログラムを生成するプロセスは、2つの異なるフェーズに分かれている。

1. ロボットのエンドエフェクタの姿勢、フレーム、ツール、定数の定義とパラメータ化。図18のアルゴリズムは、ツールモデルやパスラインからデータを取得し、ロボットコードを生成するプロセスをまとめたものである。以下の式は、ロボットプログラムにおけるロボットポーズの一般的な定義を表している。

$$P = x, y, z$$

この段階では、ロボットの姿勢に加えて、特定のプロセスおよびロボットのパラメータ (座標系、ツールなど) が指定される。この情報は、ロボットのホームポジション、作業サイクル数、接近距離など、ソフトウェアインターフェースに導入されたパラメータから得られる。

2. プログラムの本体。ロボットプログラムには、リニア、ジョイント、サーキュラー、スプラインなど、主にロボットの動作命令が含まれています。これらの動作命令は、CAD 図面やソフトウェアのインターフェースで設定された動作の種類を尊重しています。例えば、パスのセグメントが直線として描かれている場合、生成されるコードには、ロボットのエンドエフェクタをそのパスセグメント内で直線的に移動させるロボット命令が含まれます。また、この段階では、他の機械と通信するためのIOコマンドの生成や、接近距離の定義など、各ロボットタスクに関連する特殊な状況にも対応しなければなりません。

提案したアルゴリズムは、Motomanロボットコントローラ用のロボットプログラム (INFORM 言語) を生成することができます (図19)。しかし、すべてのロボットプログラムは同じ原理に基づいているので、提案したアルゴリズムは他のプログラミング言語のコードを生成するために 適応することができます。

3.8. Software interface

開発されたソフトウェア・インターフェースは、ユーザー、CADパッケージ、およびロボットの間のリンクを作るものである。提案されたソフトウェア・インターフェースの機能性と全体的なアーキテクチャを以下に模式的に示す (図20)。このソフトウェア・インターフェースは、Microsoft Windows オペレーティング・システム（XP以上）、およびAutodesk Inventor をホストできる処理能力とグラフィック能力を備えた産業用コンピュータやパーソナル・ コンピュータで動作します (図21)。主にVB.NETで作成されています。

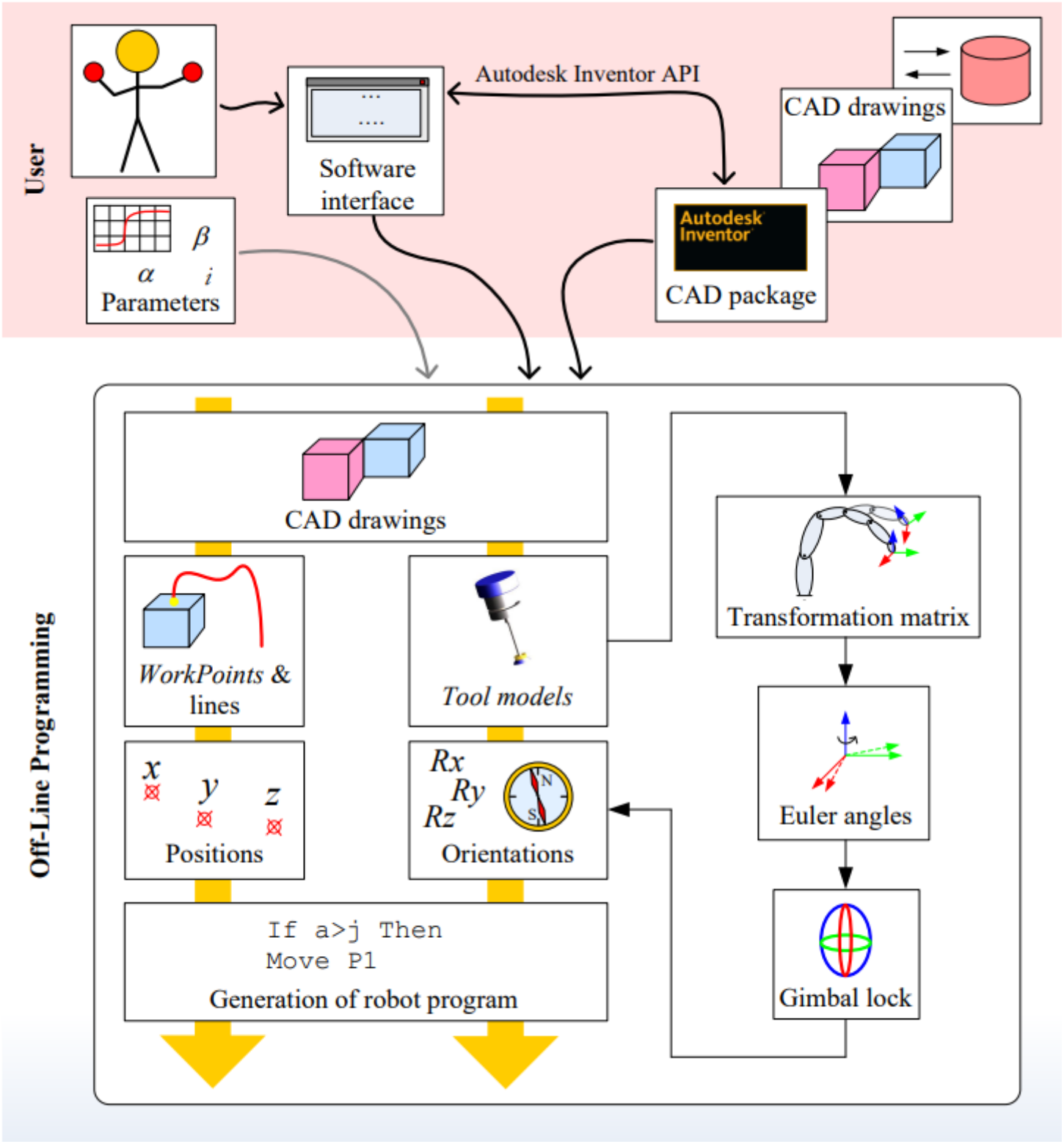


図20: 機能性とアーキテクチャ

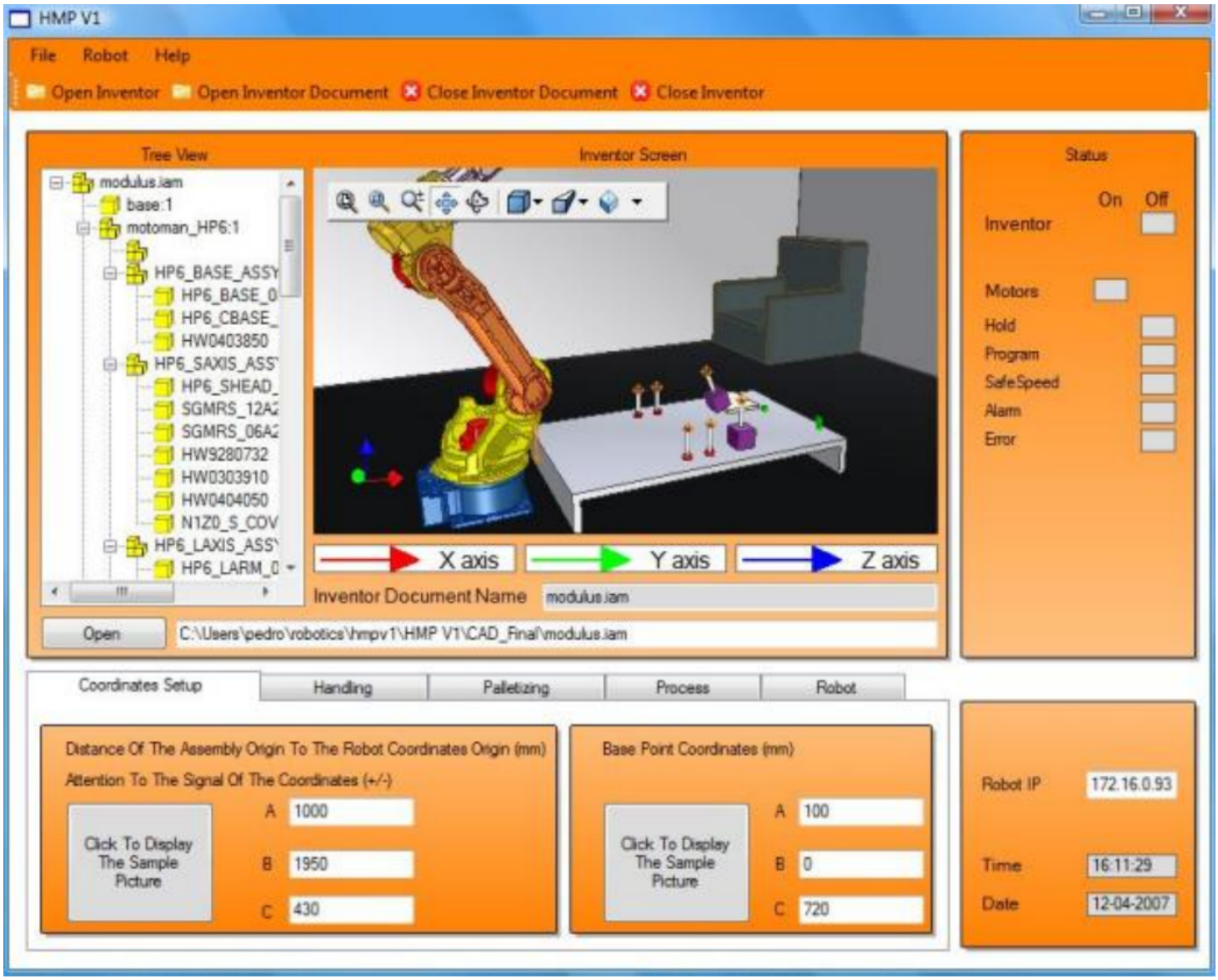


図21: 開発したソフトウェアのグラフィカル・ユーザー・インターフェース