触覚と3次元視覚を備えたロボットアームのためのプログラミング教材

○澤崎 悠太　升谷 保博 （大阪電気通信大学）

Teaching materials for programming robot arms with tactile sensor and 3D vision

○ Yuta Sawasaki　Yasuhiro Masutani

(Osaka Electro-Communication University)

Abstract : The authors are developing a teaching materials that enable novices to learn programming of robot arms equiped with tactile sensors and 3D vision. It modularizes sensors and actuators as RTC. Students learn how to combine the RTCs and how to program controller RTC. The robot arm is assumed to be composed of ROBOTIS Dynamixel actuators. The sensors are assumed to be a depth camera and tactile sensors. It reuses the general-purpose RTC for Dynamixel actuator and the RTC that outputs 3D point cloud from depth camera, which are made public on GitHub by the authors' laboratory. FingerVision is introduced as a tactile sensor and new RTCs are developed for this sensor. Intermediate RTCs are also developed to make it easier for novices to use inputs from the sensors and command to the arm.

1. はじめに

近年，製造業以外でも，運送業や農業，サービス業などでロボットの活用が盛んになってきている．製造業以外の場面でロボットアームを利用することを考えると，単純な教示再生ではなく，いくつものセンサで対象や環境を認識し，それに基づいて動作することが求められており，そのようなシステムの開発ができる人材の育成が必要である．

人材の育成に当たっては，ロボットを構成する個々の要素技術の習得とともに，システムインテグレーションの考え方を身に付けることが重要である．それには，実際にロボットを動かす実習が欠かせない．著者らは大学の情報系の学科に属しており，限られた授業時間の中で，ロボットアームのシステムインテグレーションのソフトウェア面を効果的に学習させる教材を検討している．

そのような教材では，システムがモジュール化している方が扱いやすい．また，実験機器が接続された PC と学生の PC が分かれている方が運用や管理がしやすいので，それらをネットワークで接続した場合の透過性が重要である．そこで，教材の基盤として，RT ミドルウェア [1] を利用する．

ロボットアームの制御のために利用するセンサには様々なものがあるが，種類の異なる複数のセンサの役割や使い分けを学ぶことは重要である．そこで，本稿では，RT ミドルウェアを基盤として，局所的なセンサである触覚センサと大域的なセンサである 3 次元視覚の両方を利用するロボットアームのソフトウェア教材を提案する．

以下の節では，第 2 節で問題設定について述べる．第 3 節で使用するハードウェアを紹介し，第 4 節では使用するソフトウェアについて述べる．第 5 節ではシステムを用いた課題を提案する．

2. 問題設定

教材の対象として想定しているのは，大学の情報系学科の低学年の学生であり，基本的なプログラミングについては学習済みであるが，ロボティクスや画像処理やセンサ技術については学んでいないとする．

時間数が限られた学生実験の授業で利用することを考え，学習する内容を絞り，センサの情報に応じてロボットにタスクを遂行させるプログラムの作成を主題とする．その一方で，ロボットの多数の要素を経験的に知り，より深い学習への動機付けとすることも狙う．

提案する教材を利用する学習の目標は以下の通りである．

- センサ情報に応じた場合分け，状態遷移，エラー処理を経験する．
- 性質の異なる複数のセンサ情報の利用を経験する．
- テストを繰り返してプログラムを開発することを身に付ける
- ロボット用のミドルウェアの位置付けやモジュール化の効用を知る．

提案する教材で利用するロボットアームは，4 関節とする．一般に手先の位置姿勢を任意に設定するには 6 関節以上が必要であるが，6 関節にすると教材が大掛かりになり価格も高くなるので，関節数を減らすことにした．アームが 4 関節であり，手先の動きが制限されるので，この教材でアームが扱う対象は，机の上に垂直に置かれた円柱に限定する．ただし，円柱の位置・寸法・色・重量・変形のしやすさは多様であるとし，これらに対応することを課題とする．

SY0010/19/0000 - 0926

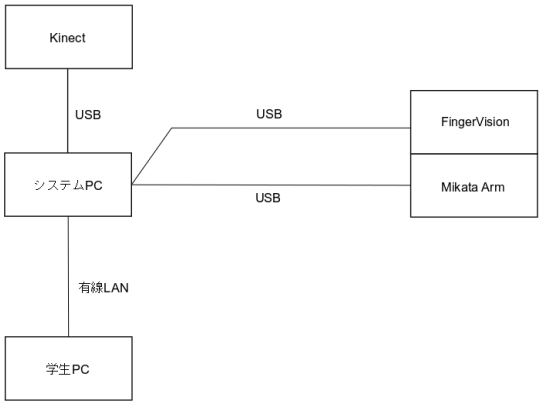

Fig. 1: System diagram of the teaching materials

Fig. 2: Setup of the teaching materials

3. ハードウェア

開発している教材のハードウェアの構成を **Fig.1** に，実際の写真を **Fig.2** に示す．

ロボットアームとして柴田らが開発した Mikata Arm[2] を用いる．触覚センサとして，山口らが開発した FingerVision[3] を用いる．3 次元視覚として深度画像センサである Microsost 社の Xbox One Kinect[4] を用いる．アームやセンサが接続されている PC を「システム PC」と呼ぶことにする．一方，学生がプログラム開発を行う PC は別であり，「学生 PC」と呼ぶことにする．2 種類の PC はネットワークで接続されている．

ロボットアームの Mikata Arm は，全長は 526.75[mm]

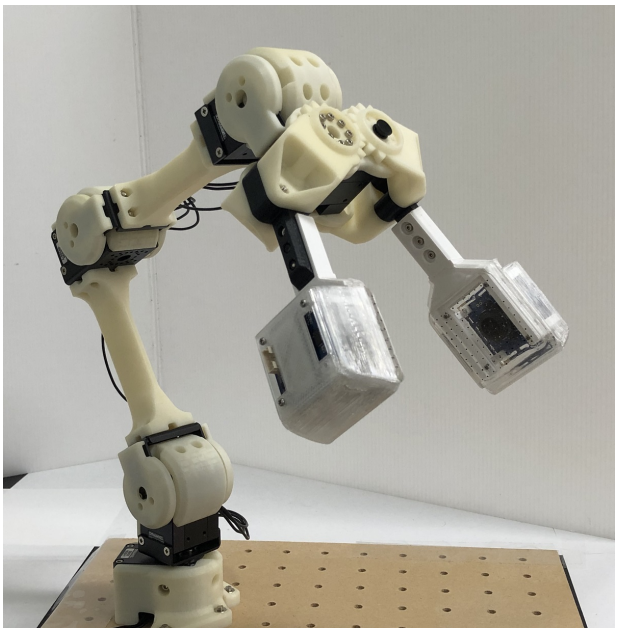

Fig. 3: FingerVision and Mikata Arm

で，関節 4 軸，グリッパ 1 軸で構成されている．アクチュエータには，ROBOTIS 社の Dynamixel XM430-W350-T を 5 個使っており，フレームやグリッパの部品の大部分を 3D プリンタで造形している．各アクチェータは USB を介して PC と双方向に通信している．

触覚センサ FingerVision は，**Fig.3** に示すように Mikata Arm の左右のグリッパとして 2 個利用している．**Fig.4** に示すようにセンサ面はビーズが埋め込まれた透明なシリコンゴムであり，シリコンゴムの部分の寸法は 45[mm] × 45[mm] である．

センサ面の後方に魚眼レンズ付きのカメラ ELP-USBFHD01M-L180 を設置しており，センサ面のビーズやシリコンゴムを通して映る近接物体を撮影する．画像の解像度は 320 × 240 であり，USB を介して PC に取り込む．

深度画像センサ Xbox One Kinect は，PC と USB 3.0 で接続し，そこから RGB 画像と深度画像を得ている．

教材として用意した RTC には，処理の負荷の大きいものがあり，全ての RTC を 1 台の PC で実行するには無理がある，そこで処理能力の高い PC をシステム PC として用意し，アームやセンサを接続する．このようにすれば，ハードウェアを扱うためのドライバやライブラリを学生 PC にインストールしなくて済み，授業の準備の面でも有利である．

4. ソフトウェア

システム PC と学生 PC のどちらも Windows 10 64bit を用いており，RT ミドルウェアは，OpenRTM-aist 1.2.0 を利用している．

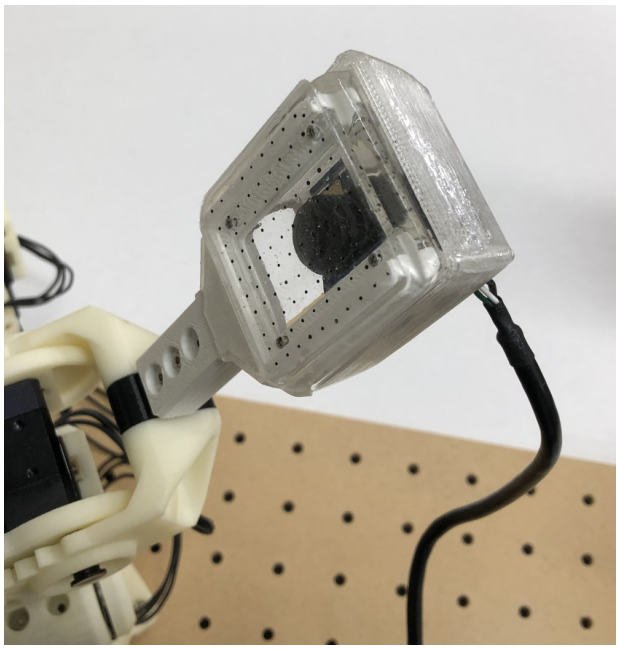

Fig. 4: FingerVision Sensor

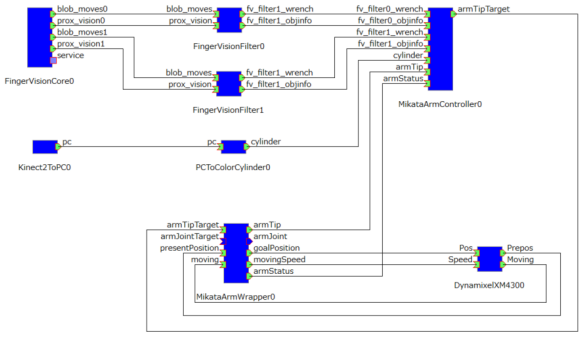

Fig. 5: System diagram of RT Components

本節では，この教材で使っている RT コンポーネント（以下 RTC）について説明する．用意した RTC を全て接続した場合のダイアグラムを **Fig.5** に示す．この中で，MikataArmController だけが学生 PC で動作しており，それ以外の RTC はシステム PC で動作している．

アームと 2 種類のセンサのそれぞれに，ハードウェアとやり取りする RTC と中継役の RTC を用意している．中継役の RTC は，ハードウェア担当の RTC と学生がプログラムする RTC の間に位置し，初学者でもハードウェアが扱いやすいようにする役割を担う．

4.1 アーム用 RTC

アーム用の RTC は，アームのアクチェータとやり取りする Dynamixel [5] と中継役の MikataArmWrapper [6] である．

Dynamixel は，ROBOTIS 社のアクチュエータ Dynamixel の同一機種を複数個利用するための RTC で，Dynamixel のプロトコルのバージョン 1 と 2 の両方に対応しており，AX12, XL320, XM430 で利用実績がある．これは，筆者らの研究室で以前から公開しているものを再利用した．入力ポートは，Pos（目標位置）と Speed（動作速度），出力ポートは，Prepos（現在位置）と Moving（動作フラグ）である．また，コンフィギュレーションによって，通信速度，プロトコルのバージョン，Dynamixel のモデル，アクチュエータの数を設定できる．この RTC は C++で記述しており，ROBOTIS 社が公開している DynamixelSDK3.2.0[7] を利用している．

MikataArmWrapper は，Mikata Arm の 5 個のアクチェータ XM430 とやり取りする RTC Dynamixel を使いやすくする RTC である．MikataArmWrapper では，上位のコンポーネントから受け取った角度指令をアクチュエータ内部表現の角度に変換し，各軸が同じ時間をかけて指令前の角度から目標角度へ動くための各アクチュエータの角度を計算処理してから Dynamixel のコンポーネントへ出力する．

入力ポートは，armTipTarget（手先の目標位置），armJointTarget（関節の目標位置），presentPosition（Dynamixel から現在位置），moving（Dynamixel から受け取る動作フラグ），出力ポートは，armTip（手先の現在地）armJoint（関節の現在位置）goalPosition（Dynamixel へ送る目標位置）movingSpeed（Dynamixel へ送る動作速度）armStatus（アームの状態）である．この RTC は C++で記述している．

4.2 3 次元センサ用 RTC

3 次元視覚用の RTC は，深度画像センサである Xbox One Kinect（Kinect v2）[8] とやり取りする Kinect2ToPC[9] と，点群の中から円柱を抽出する PCToClolorCylinder [10] である．

Kinect2ToPC は，Kinect から色画像と深度画像を読み取り，汎用的な色付き点群のデータ型である RTC:PCL[11] の PointCloud 型（PointCloudTypes::PointCoud）を出力する．この RTC は C++で記述されており，Kinect for Windows SDK v2.0 と PointCloudLiblary 1.8.1 を利用している．

筆者らの研究室では，RTC:PCL の PointCloud 型を出力する Kinect for Windows(Kinect v1) 用 RTC[12] や RealSense D400 系列用 RTC[13] も公開しており，それらを Kinect2ToPC と入れ替えることが可能である．

PCToColorCylinder は，RTC:PCL の PointCloud 型のデータを受け取り，本教材の内容に合わせて，色付きの点群の中から複数円柱の情報を抽出する．出力ポートは

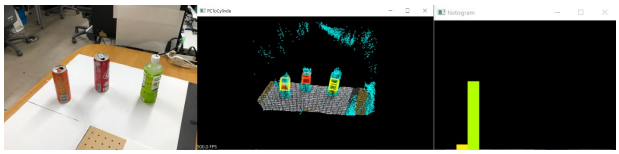

Fig. 6: Cylinder extraction screen, histogram and actual environment

cylinder のデータ型は TimedDoubleSeq であり，認識した円柱の数× 5 個の要素を有する．各要素の意味は，それぞれの円柱に対して順に，円柱の中心の x 座標 [m], y 座標 [m], z 座標 [m]，円柱の半径 [m]，円柱と側面に属する点群で支配的な色相（0〜360[deg]）である．なお，円柱の座標値はセンサ座標系で表されている．

以上の抽出処理には，PointCloudLiblary と OpenCV の機能を利用している．円柱は平面の上に配置されていることを前提としており，最初に点群の中から平面を推定し，その平面に近い点群を削除する．次に，その平面の法線の方向と軸の方向が一致する円柱を推定し，円柱側面に近い点群を抽出する．抽出された点群の色相のヒストグラムを求め，最も頻度の高い色相を得る．以上を円柱が見つからなくなるまで繰り返す．

PCToColorCylinder の動作画面と環境を **Fig.6** に示す．

4.3 触覚センサ用 RTC

触覚センサ用の RTC は，FingerVisionCore[14] と FingerVisionFilter[15] の二つである．FingerVisionCore は，カメラ画像を処理してマーカや近接物体の動きを検出する．FingerVisionFilter は，FingerVisionCore の出力した情報を受け取り扱いやすい形式に変換する．それぞれの RTC は，山口が公開している ROS のノード fv_core_node と fv_filter1 [16] を移植をしたものである．

FingerVisionCore は，設定ファイルの記述によって FingerVision の 1 台または 2 台のカメラから USB 経由で画像を受け取る．それぞれのカメラに対して，マーカの動きと近接視覚の処理結果を出力するため 4 個の出力ポートを有する．出力ポートのデータ型のメンバは，オリジナルの ROS トピックスのメンバの名前や型にできるだけ合わせている．

- blob_moves0 （データ型： BlobMoves）
- prox_vision0 （データ型： ProxVision）
- blob_moves1 （データ型： BlobMoves）
- prox_vision1 （データ型： ProxVision）

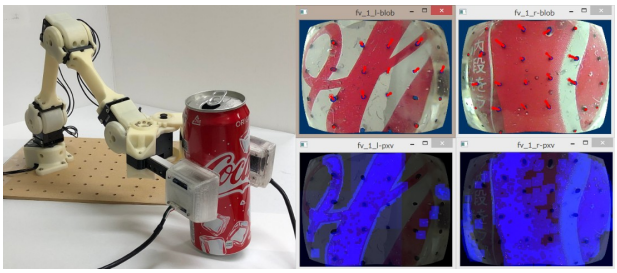

Fig. 7: FingerVision grabbed the cylinder

入力や画像処理の設定は YAML ファイルに記述するが，その仕様はオリジナルの ROS のノードと同一である．オリジナルの ROS パラメータは，RTC のコンフィギュレーションとして，ROS サービスは，RTC のサービスポートとして実装している．独自のデータ型とサービスポートの定義するために FingerVision.idl を作成した．この RTC は C++で記述しており，画像処理には OpenCV を利用している．円柱を把持している状況の FingerVision の動作画面を **Fig.7** に示す．

FingerVisionFilter は，FingerVisionCore が出力した 1 台分のカメラのマーカの動きと近接視覚の情報をより扱いやすい形式に変換する．このポートの出力のデータ型もオリジナルに合わせており，上述の IDL ファイル内で定義している．

- fv_filter1_wrench （データ型： Filter1Wrench）
- fv_filter1_objinfo （データ型： Filter1ObjInfo）

Filter1Wrench が力の分布の情報，Filter1ObjInfo が近接物体の情報を表している．これらの中には多数の情報が含まれているが，初学者の教材として使いやすいのは，力の平均値，近接物体の動きや面積の変化である．この RTC は，オリジナルの ROS ノードと同じく Python で記述している．

4.4 学習用 RTC

学習用 RTC は，MikataArmController[17] と名付けた．この RTC は，FingerVisionFilter と PCToColorCylinder から得られたセンサ情報を活用して，MikataArmWrapper とやり取りして，アームに作業を遂行させる．

PCToColorCylinder から得られる円柱の座標はセンサ座標系で表せており，ロボット座標系に変換する必要がある．学生に座標変換を扱わせるのは無理があると考え，それは教材側で用意する．あらかじめ，別プログラムで座標変換のパラメータを推定して，それをファイルに保存しておく．この RTC では，そのファイルを読み込み，ロボット座標系へ変換後の座標値を提供する．

提案する教材では，学生にはこの RTC のプログラムを作らせる．学生の理解度や授業の目的によっていくつかの選択肢を検討している．

- RTC Builder で RTC のテンプレートを作るところから始め，RTC のクラスのメンバ関数 onActivated() や onExecute() の中身を直接記述させる．
- あらかじめ RTC を作っておき，onActivated(), onExecute() から呼び出される関数を用意しその中身を記述させる．データポートの入出力は隠蔽し，そのための関数を提供する．作成する関数の処理のタイミングは RTC と同期させる．
- あらかじめ RTC と，その RTC とは非同期に動作するスレッドを作っておく．その別スレッドで実行される関数の中身を記述させる．センサやアームに対する入出力ができるようにスレッド間でデータをやり取りする関数を提供する．

5. 課題案

前節までで述べた教材を開発しながら，それを活用する課題についても検討を進めている．最終的に，触覚センサと 3 次元視覚の両方を使って複数の円柱の把持と運搬ができるように，段階的な課題を設定する．検討中の課題案を以下に示す．

- 課題 1： ロボットアームの手先が，(x, y, z) = (0.15[m], 0.20[m], 0.15[m]) → (0.15[m], -0.20[m], 0.20[m]) → (0.2[m], 0.0[m], 0.15[m]) の 3 点を巡回させるプログラムを作成せよ．点間の所要時間は 3 秒とする．
  狙い： ロボットアームがどのような動きができるのかを理解させる．
- 課題 2： 深度画像センサから入力される円柱の座標情報を利用してロボットアームで円柱を把持し，座標 (x, y) = (0.15[m], -0.15[m]) もしくは (0.15[m], 0.15[m]) に円柱を置き，初期位置に戻るプログラムを作成せよ．余力のあるものは円柱の色によって置く場所を変更せよ．
  狙い： 深度画像センサからの情報を利用したロボットアーム制御に慣れてもらう．
- 課題 3： 触覚センサから入力される情報を利用して円柱をつぶさないよう把持し，落とさないように座標 (x, y) = (0.15[m], -0.15[m]) に運搬せよ．円柱は比較的つぶれやすいものを座標 (x, y)=(0.15[m], 0.20[m]) に設置する．また，水が入った円柱を (x, y) = (0.2[m], 0.0[m]) に設置する．
  狙い： 触覚センサからの情報をどのように利用すれば円柱を上手く把持できるかを学習してもらう．
- 課題 4： 3〜5 種類のそれぞれ重さ、半径や高さ，色が違う円柱をアームの周辺にその場で配置を決めて並べる．赤色の円柱は (x, y) = (-0.05[m], 0.10[m]) に置き，青色の円柱は (x, y) =(-0.05[m], -0.10[m]) に置け．それ以外の色の円柱は (x, y) =(-0.10[m], 0.0[m]) に設置せよ．また，円柱を落としたりつぶしたりせずに適切に分別せよ．
  狙い： 全てのセンサ情報を活用したロボットアームの制御を習得してもらう．

6. おわりに

本稿では，著者が開発中のロボットアームと複数センサを用いたプログラミング教材を紹介した．

参考文献

[1] 産業技術総合研究所: OpenRTM-aist, http://www.openrtm.org/
[2] 山口明彦, 梁田 善広: Mikata Arm + FingerVision–センサリッチ教育研究用ロボット, 日本機械学会 ROBOMECH2018, 2A1-H16 (2018)
[3] Akihiko Yamaguchi: FingerVision, http://akihikoy.net/p/fv.html (2017)
[4] Kinect センサー情報 — Xbox One のアクセサリー, https://support.xbox.com/ja-JP/xbox-one/accessories/kinect-sensor-info
[5] 澤崎 悠太，升谷 保博: Dynamixel 用 RT コンポーネント, https://github.com/MasutaniLab/Dynamixel
[6] 澤崎 悠太，升谷 保博: MikataArmWrapper, https://github.com/MasutaniLab/MikataArmWrapper
[7] ROBOTIS: Dynamixel SDK (Protocol1.0/2.0), https://github.com/ROBOTIS-GIT/DynamixelSDK
[8] Microsoft: Kinect センサー情報 — Xbox One のアクセサリー, https://support.xbox.com/ja-JP/xbox-one/accessories/kinect-sensor-info
[9] 升谷 保博: Point Cloud Grabber for Microsoft Kinect v2, https://github.com/MasutaniLab/Kinect2ToPC
[10] 澤崎 悠太，升谷 保博: PCToColorCylinder, https://github.com/MasutaniLab/PCToColorCylinder
[11] Geoffrey Biggs: RTC:PCL, https://github.com/gbiggs/rtcpcl
[12] 升谷 保博: Point Cloud Grabber for Microsoft Kinect, https://github.com/MasutaniLab/KinectToPC
[13] 升谷 保博: Point Cloud Grabber for Intel RealSense, https://github.com/MasutaniLab/RealSense2ToPC
[14] 澤崎 悠太，升谷 保博: FingerVision 用 RT コンポーネント, https://github.com/MasutaniLab/FingerVisionCore
[15] 澤崎 悠太，升谷 保博: FingerVision 後処理フィルタ用 RT コンポーネント, https://github.com/MasutaniLab/FingerVisionFilter

[16] Akihiko Yamaguchi: Data processing programs for the vision-based tactile sensor FingerVision, https://github.com/akihikoy/fingervision
[17] 澤崎 悠太，升谷 保博: MikataArm 制御用 RTC, https://github.com/MasutaniLab/MikataArmController