

自主プロジェクトレポート

Mini Mobile Manipulator ロボット

利光泰徳

2018年2月1日



東京大学工学部機械情報工学科 3 年 03-170294
レポジトリ : <https://github.com/Yasu31/Mini-Mobile-Manipulator>

目次

1	序説	2
1.1	モバイルマニピュレータ	2
1.2	目標	3
2	仕様	4
2.1	スペック	4
2.2	使用部品	4
3	製作過程、設計	4
3.1	概要の決定	4
3.2	CAD データの作成	5
3.3	動物園	5
3.4	デザインの決定、製作	6
3.5	Arduino-Raspberry Pi 間の通信	8
3.6	ロボットを記述する URDF の作成、ROS への対応	8
4	動作結果	9
5	考察、感想	10
6	その後	10

1 序説

この自主プロジェクトでは、モバイルマニピュレータ型ロボットを自作して、ROS を使って動かしてみようとした。まずは、モバイルマニピュレータの概要と、自主プロジェクトの目標について述べる。

1.1 モバイルマニピュレータ

モバイルマニピュレータは、移動式台車の上にロボットアームが載った構造をしたロボットのことで、動ける範囲が広いというロボットカーの利点と、器用にものを扱えるというロボットアームの利点を併せ持つ。しかし、工業においては、それぞれの技術が独自に発達を遂げていて、二つを組み合わせたモバイルマニピュレータ型のロボットはまだ少ないと言われている。^{*1}

市場に出ているものの例として、図 1 にある、KUKA 社製の youBot がある [1]。5 自由度のアーム、全方位に動けるタイヤを持っている。台車の大きさは $0.6m \times 0.4m$ で、 $0.5kg$ の物を持ち上げることができる。ROS に対応していて、価格は 260 万円ほどだが、現在は生産を中止している。

^{*1} https://en.wikipedia.org/wiki/Mobile_manipulator



図 1 KUKA youBot[1]

1.2 目標

今回の自主プロでは、ROS に対応したモバイルマニピュレータ型ロボットを作り、動作させることを目標とした。ハードウェアの製作や ROS との連携を、作りながら学んでいくことも狙いだった。

上述した KUKA youBot は 260 万円と、とても趣味で手が届く価格ではないので、比較的手軽な価格で「つくれてあそべる」ようなロボットを作りたかった。ロボットが普及する上でも、価格がネックの一つとなっている。より求めやすい価格となることで、使用される場面が広がり、それによって開発が更に進む、という好循環が期待されている [3]。

大型のロボットだと価格が高くなってしまうのもあり、文庫本程度のサイズのロボットを作ることとしたので、Mini Mobile Manipulator と命名した（以降、MMM^{*2}）。

小型にしたことで、実現できる動作もあると考えた。例えば、階段のステップなどに掴まって、自分自身を持ち上げることで段差を超えていく、という動作だ。これは大きなロボットでは、自らを持ち上げるにはあまりに大きなトルクが必要だったり、危険性が高かったりと実現が難しいが、MMM のような小さなロボットならば可能と考えた。使ったサーボのトルクは 6kgcm で、最終的にロボットの重さは 700g 程度だったので、重心が先端から 8.5cm 程度の位置にあるようにして持ち上げれば、この動作は実現可能といえる。

^{*2} 図 5 で見えるように、前後の半円状のパーツには、名前にちなんでアルファベットの M があしらわれている。

2 仕様

2.1 スペック

アーム	7自由度(マニピュレータ部6+グリッパー1) 0.5kg可搬 MoveIt!(動作計画ソフト)で制御可能
カメラ	アーム先端部 rostopic /raspicam_node/image に出力 ARマーカーを認識可能
台車	秒速約30cm rostopic /turtlesim/cmd_vel で制御
材料費	~40,000円
重さ	700g
大きさ	幅110mm、奥行き190mm、アーム長340mm
構造	MDFボード
電圧	6~7.4V(アームと車輪モーター) 5V(計算器類)
レポジトリ	https://github.com/Yasu31/Mini-Mobile-Manipulator プログラムやレーザーカットのデータなど、全て載せてあります

2.2 使用部品

- 近藤科学 KXR-A5 アーム部のサーボと接続パーツ
- ICS 変換基盤(アームと Arduino 間の通信用)
- Raspberry Pi 3 model B
- Arduino nano^{*3}
- MDF ボード
- TA7291P モータードライバ
- タミヤ ミニモーター多段ギヤボックス
- タミヤ スリックタイヤ(直径31mm)
- キャスター

3 製作過程、設計

3.1 概要の決定

始めの数日間で、ロボットの概要を決めて、どのように作っていくか方向性をつけた。モバイルマニピュレータとして成り立たせるために、ロボットアームと車輪はもちろんのこと、周囲の様子が分かるようにカメ

^{*3} aitendo で販売していた互換機を使用。本家の半額で購入できるが、別途 USB ドライバをインストールする必要がある。

ラも取り付けることとした。カメラはアームの先端に取り付けることを考えた。この設計によって、アームを伸ばして回せば、周囲を見渡すことができる。アームが動いたら、カメラに映った物がどこにあるのか分からなくなる、と思うかもしれないが、アームのサーボモーターの現在の角度を求めることができるので、それをもとに計算すればよい。順運動学を計算すればアームの先端の位置が分かるので、ロボットのベースからアームの先端（カメラの位置）へのベクトルと、カメラから物体へのベクトル（これは、AR マーカーを認識するライブラリなどを使えば求められる）を足せば、認識された物体の位置が分かる。

ロボットアームには、近藤科学社製のサーボと部品を使用した。KXR-A5 アーム型ロボット^{*4}のアーム部分を構成する部品、およびそれを制御するのに必要な Arduino 用ボード (ICS 変換基盤^{*5}) などを購入した。近藤科学の製品を使ったのは、JSK での少人数ゼミで既に使っていた、ある程度使い勝手が分かっていたことと、比較的安価^{*6}で比較的強力だったからである。アームだけで約 25,000 円かかり、やや高価な自主プロとなるが、うまく動く既製品を使うことでハードウェアを早めに完成させ、実際に動かすための頭脳であるソフトウェアにすぐに取り掛かることを目指した。

ボディは、アームやホイールや電子部品を取り付けられるような構造を CAD で設計して、3D プリントすることを考えた。その上で、それぞれの部品の 3D データが必要となった。

3.2 CAD データの作成

次に、一週間ほどかけて、CAD データを作っていました。ボディの設計のためだけではなく、後ほどアームの逆運動学を解くことを考えても、アームの各部品を細かくモデリングして、アームの 3D データを作成する必要があった。サーボモーターの CAD データは公開されていたものの、それらを繋ぐ部品については、寸法を測りながら大まかな形状を再現していました。CAD ソフトは Autodesk Fusion 360(図 2) を使用した。これは、付属の iOS アプリもあり、(編集はできないが) デザインを iPad で回しながら見ることができた。これは、デザインの妥当性を評価する上で役立った。また、各部品の重さを設定すれば、重心の位置が分かる。アームが重いため、アームを伸ばすとロボット自体が倒れてしまう心配があったので、CAD で重心の位置を考慮しながら設計することができた。前輪がなるべく前方に配置されているのも、この結果である。タイヤを駆動するためのタミヤのギヤボックスもモデリングし、Raspberry Pi についてはウェブ上の CAD モデルをダウンロードして使った。

3.3 動物園

ロボットのデザインを親しみやすいものにするために、動物に基づいて設計しようとした。また、ロボットの行動についても、動物の振る舞いからヒントが得られることを期待して、上野動物園を訪れた。ロボットと同じように、長い首（や鼻）を持つ動物として、ダチョウなどの鳥類、蛇、象などが考えられるため、これらの動物の行動を重点的に見た。

結局、あまり直接的にデザインに関係してくる発見はなかったが、一つ取り入れたのは、フラミンゴのような首の長い鳥が、首を引っ込めているときの S 字のカーブである。これは、ロボットのポーズの姿勢として設定し、”bird”という名前をつけた。

^{*4} <http://kondo-robot.com/product/03134>

^{*5} <http://kondo-robot.com/product/03121>

^{*6} KRS-3301 サーボモーターの本体だけなら約 2000 円

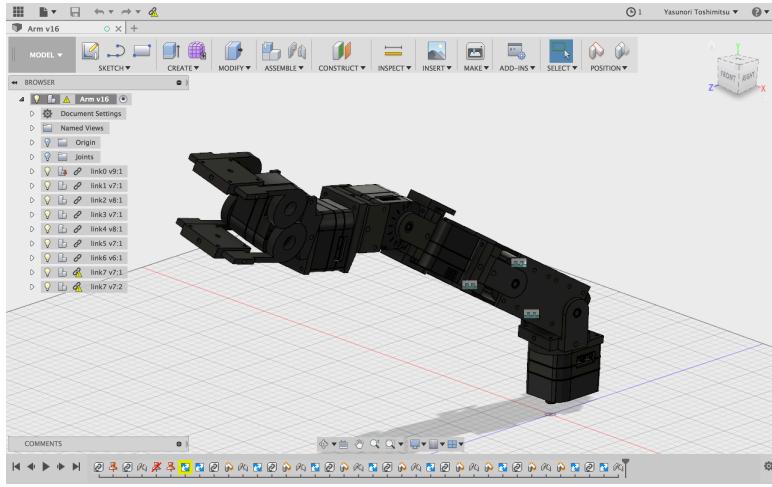


図2 Autodesk Fusion 360



図3 フラミンゴ

3.4 デザインの決定、製作

年末までにハードウェアを完成させることを目指していた。

このロボットには、搭載するボードが2つある。一つはRaspberry Piで、もう一つはArduinoやサーボとの通信モジュールを含んだユニバーサル基盤である。それらをどう配置しようか悩んでいたが、山型に斜めに配置することで、高さと幅が大きくなりすぎるので防ぎつつ、容易に部品にアクセスできるようなデザインにした。また、このデザインなら3Dプリンタよりもレーザーカッターで作った方がやりやすい、との友人の助言も受けて、レーザーカッターで作れるように、設計しなおした(図4)。3Dプリンタだと一体作るのに数時間かかるが、レーザーカッターなら数分で切り終わり、ちょっとした修正も容易なので、とてもよかったです。実際製作した時も、CADで想定できていなかった干渉や間違いがあったが、すぐに修正できた。図5にレーザーカットしている様子を載せた。この日は、それまでCADでしか見ていなかったものを、たった1日で組み上げることができて、とても面白かった。

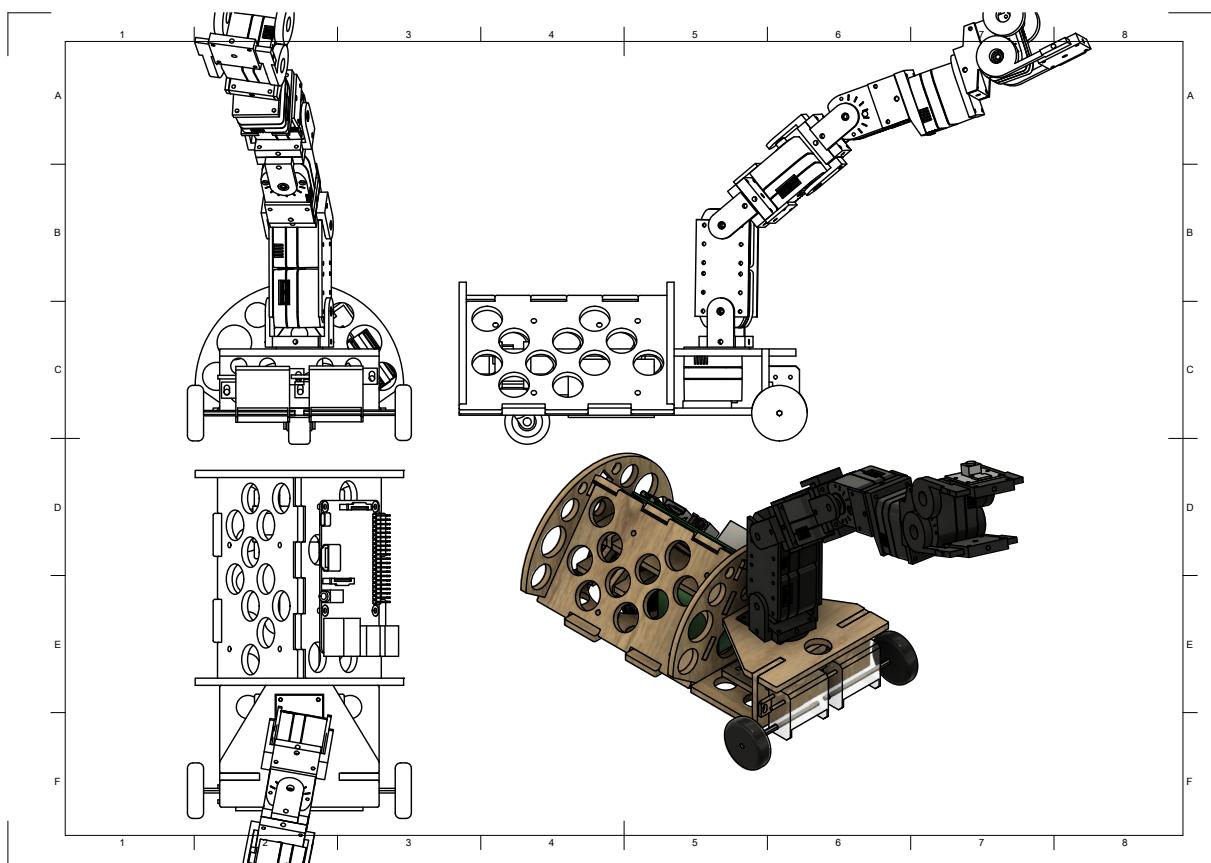


図4 最終的なデザイン。軽量化のため、肉抜きしてある。



図5 レーザーカットしている様子

3.5 Arduino-Raspberry Pi 間の通信

ソフトウェア面での第一の課題として、これらの間の通信に取り組んだ。普通なら、USB ケーブルで繋いでシリアル通信すればいいが、近藤科学の ICS 変換基盤は、Arduino のシリアル通信ポートを占有してしまい、通常のシリアル通信ができなくなる、という問題があった^{*7}。このため、Arduino と Raspberry Pi は I2C 通信することとした。I2C のためのライブラリが、バイト単位で処理するものしか見つからなかつたため、16bit 整数を、二つのバイトに変換してシリアルポートに送ることとした。今回決めた通信プロトコルは、以下のようなである。

3.5.1 Arduino → Raspberry Pi

7 つのサーボのポジションデータと、IMU の値を、34byte のデータとして返す。

0-1 bytes : 1 個目のサーボのポジション。 14.21° なら、1421 と変換される (100 倍した整数値になる)。

⋮

12-13 bytes : 7 個目のサーボのポジション。

14-15 bytes : IMU のデータ 1 個目 (加速度の x 値)

⋮

24-25 bytes : IMU のデータ 6 個目 (ジャイロの z 値)

26-34 bytes : 予備。Arduino 側にボタンを設置したりした時にその値を返すのに使おうと思った。

3.5.2 Raspberry Pi → Arduino

サーボやタイヤを動かす指令を送る。2 つの整数値を、4 byte のデータにして送る。一つめの整数値は noun と呼び、何に対して動作させたいのか決める。二つ目の整数値は verb と呼び、noun で選ばれた動作対象を、具体的にどのように変更するか決める。^{*8}

$noun = 0, 1, \dots, 6$: verb は、noun 番目のサーボの指令角度。Arduino から Raspberry Pi への通信の時と同様、角度を 100 倍した整数値。

$noun = 10, 11, \dots, 16$: verb が 0 なら、(noun-10) 番目のサーボが脱力状態になり、1 なら力が入った状態になる。

$noun = 20, 21$: タイヤのスピードを verb = -255 - 255 で指定。noun=20 が右、21 が左。

3.6 ロボットを記述する URDF の作成、ROS への対応

次に、ロボットのリンク機構を表した URDF(Unified Robot Description Format) ファイルを作った。Fusion 360 から各リンクの 3D メッシュを出力して、それらの相対関係を調べながら組み立てた。

それから、アームの各サーボの現在値を、rostopic /joint_states に出力するようなパブリッシャを書いた。これで、rviz 上で、ロボットの現在の姿勢を反映して動くモデルが完成した。MoveIt!で動かすためには、action server というものを立てる必要があったが、これについてのまとまった情報がなかなか見つからず、手

^{*7} 自主プロジェクトが終わってから約一週間後、シリアルポートを占有しないように改良された、ICS Library for Arduino ver.2 が公開された。素晴らしいタイミング…

^{*8} この noun-verb ペアでのやりとりは、Apollo Guidance Computer の DSKY interface にインスピアされて思いついた。

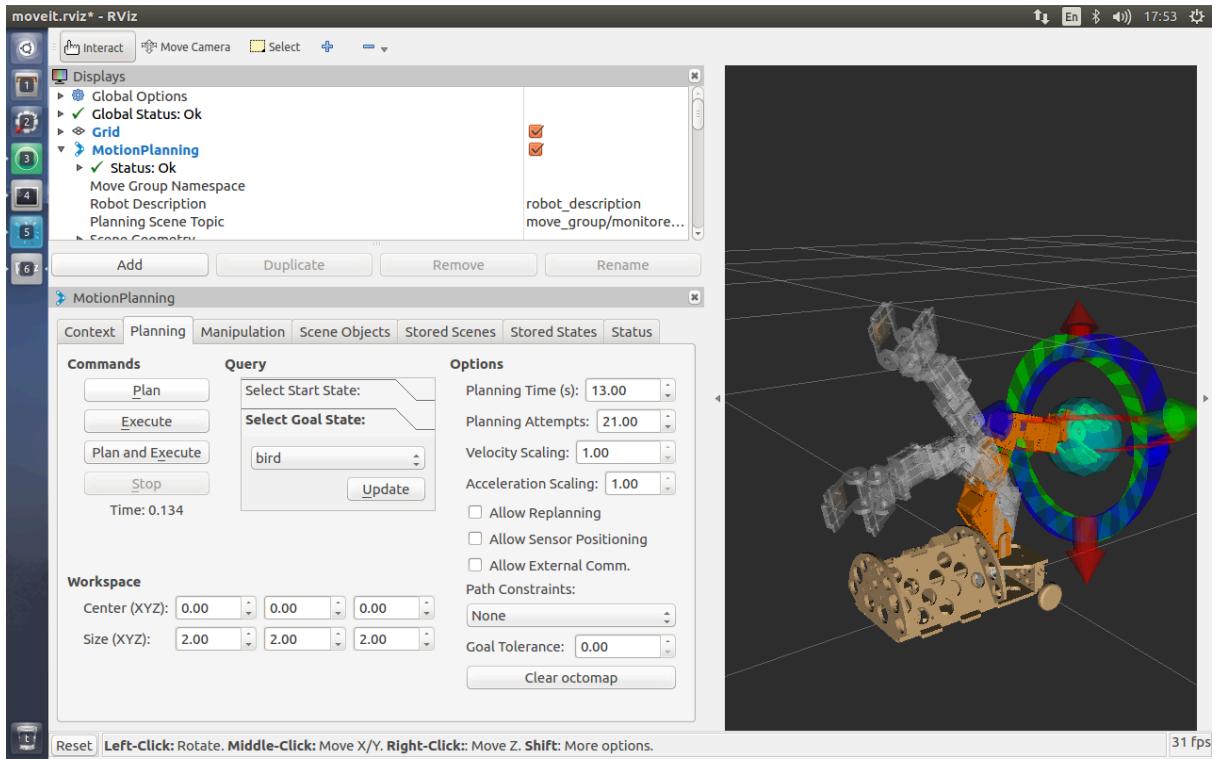


図 6 MoveIt!の画面

探りで作っていった。最終的には、少々強引な方法で MoveIt!からのコマンドを取得して、アームを動かすことができた。

4 動作結果

ひととおり、ROS を通じて動くロボットができた。MoveIt!で動作指令を送ると、その姿勢に動いてくれる。図 6 は、アームが後ろを向いている状態から、bird という、前方に向いた姿勢まで動くための動作計画をしている最中の画面である。MoveIt!は逆運動学ライブラリを含んでいて、Python インターフェースもあるため、MoveIt!に対応していれば比較的簡単に、コードからロボットの動作を指定することができる。

また、ROS のチュートリアルでよく使われる turtlesim の turtle_teleop_key から前後回転の指令を送ることで、タイヤが周り、ラジコンのように動かすこともできる。こちらも、rostopic の出力を Arduino に送っているだけなので、コード内から、ロボットを動かすことができる。

また、アームの先端についたカメラからの画像を取り、AR マーカーを認識させることにも成功した。Raspberry Pi のカメラから rostopic に書き出すのは raspicam_node という catkin パッケージ^{*9}、その画像からマーカーを認識するのは AR_track_alvar^{*10}を用いた。

YouTube に、デモのビデオを載せてある^{*11}。

^{*9} https://github.com/UbiquityRobotics/raspicam_node

^{*10} http://wiki.ros.org/ar_track_alvar

^{*11} <https://www.youtube.com/watch?v=AJrDMAA22wg> (Mini Mobile Manipulator で検索すれば出てくるはず)

5 考察、感想

「ROS で動くロボットを作る」という当初の目標は達成できたため、満足している。しかし、自主プロが終わった時点では、ROS 経由でコントロールできるものの、ロボット用の動作ソフトは作る時間がなかった。そのため、ロボットというよりは遠隔操縦できる操り人形のような状態だった。これからは、このロボットを使って、制御法や動作計画を学びながら実装していってみたい。製作中、MMM にさせてみたいことをどんどん思いついたので、これからはそれらのアイデアも実践していってみたい。

このプロジェクトを通して、「ロボットを作る」という一連の過程を簡易なりにも体験できたため、多くのことを学ぶことができた。レーザーカッターや ROS など、今までに使ったことのないツールが多かったが、実際に学んでみると案外すぐに作れる場合がほとんどだった。難しそうと決めつけてしまい、新しいツールを始めることに抵抗感を感じることが多かったが、これを機に様々なツールに触れてみようと思った。

6 その後

稲葉先生の少人数ゼミ「トランスフォーマ・ロボットを作ろう」の最終発表が 1 月 31 日にあり、何かしらのデモを発表することとなっていたため、ゼミで提供された人型ロボット KXR と、自主プロジェクトで作った MMM ロボット同士で協調行動するデモを作ろうとした。自主プロジェクトの範疇ではないので詳細な説明は省くが、Brooks が提唱した subsumption architecture[2] で動作を設計した。MMM ロボットの方は図 7 のようなアーキテクチャとなっている。KXR ロボットに貼られた AR マーカーを見つけると、近づき、十分近づくと手を伸ばし、KXR が持っているものを受け取り、しまう。しかし、カメラが壊れてしまったことで、全ての実装をすることが不可能となってしまった。その状態でも、手動で何かを持たせれば、しまってくれる。一部が壊れても全てが動作しなくなるわけではない、という subsumption architecture の特長を、思いがけず活用できた^{*12}。

参考文献

- [1] R. Bischoff, U. Huggenberger, and E. Prassler. Kuka youbot - a mobile manipulator for research and education. In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.
- [2] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1):3 – 15, 1990. Designing Autonomous Agents.
- [3] M. Quigley, A. Asbeck, and A. Ng. A low-cost compliant 7-dof robotic manipulator. In *2011 IEEE International Conference on Robotics and Automation*, pages 6051–6058, 2011.

^{*12} デモ動画 : <https://www.youtube.com/watch?v=ZHt5yPhyLIU>

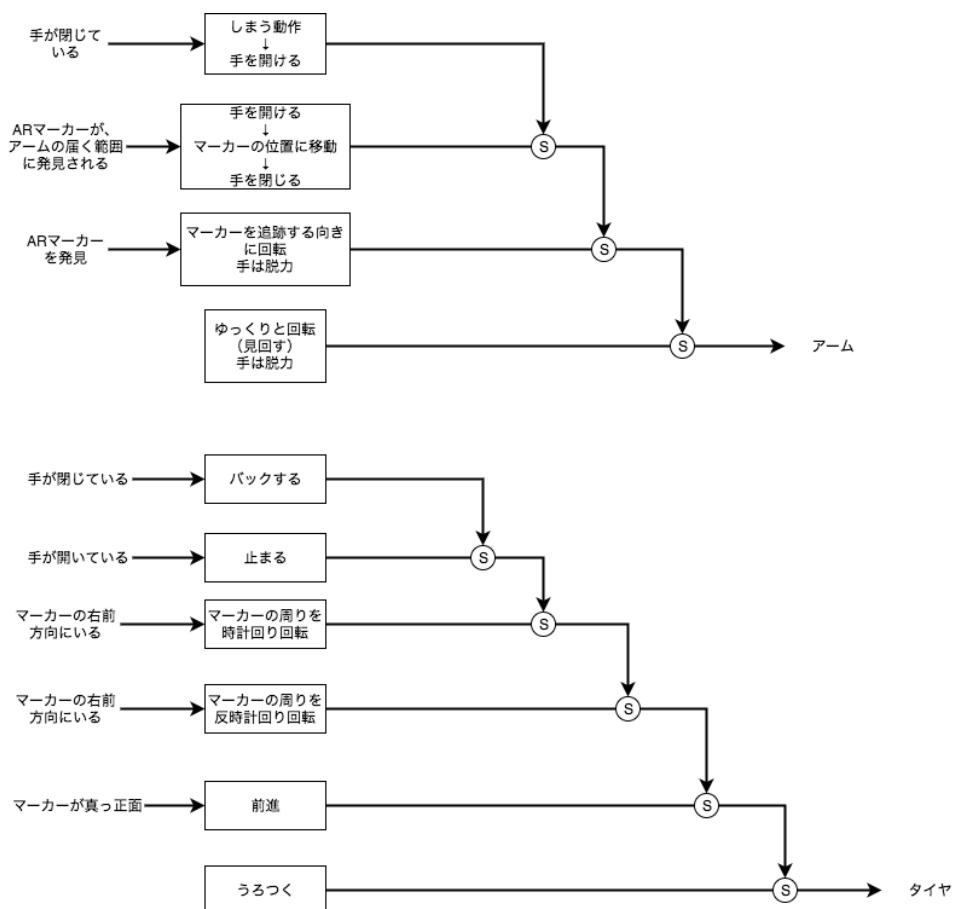


図 7 MMM に実装した subsumption architecture