知能システム論1 (6)

2012.5.23

情報システム学研究科
情報メディアシステム学専攻
知能システム学講座
末廣尚士

8. 作業ロボットのモデル化 - 理想(仮想)ロボットアーム

ロボットアームとは何をするものか

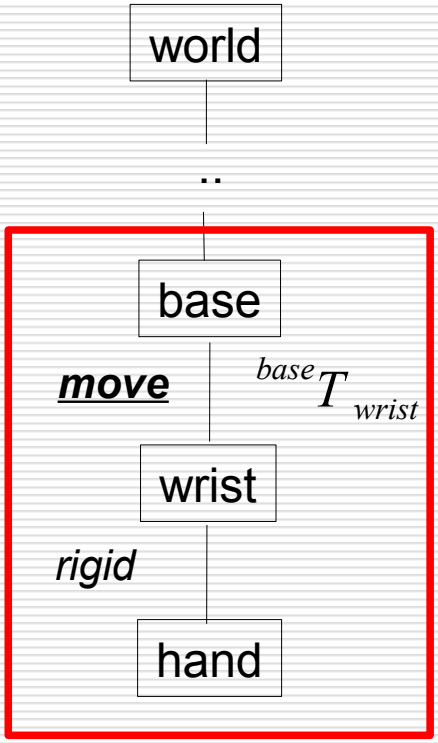
- いろいろなものを「操作」する
- ここでは「対象の位置・姿勢の操作」に注目
- さらに座標系を動かす機能として抽象化する

- 本質は、手先を自由な位置・姿勢に持っていけるということ
- ただし、手先の位置・姿勢の基準をワールドにするのではなく、アームのベースを導入することでよりリアルな表現にする

- 理想(仮想)ロボットアームの構造

baseとwristをもっている。
- ハンドツールを変えられるように

- まずbaseがどこかに配置される。
  - world, table, 移動体などの上
- 動作命令で、wristをハンドに対して自由に位置決めできる。
- 別途handを付けられる。
  - handは開閉などの命令を受け付ける。



- 理想アームによる位置決め(1)

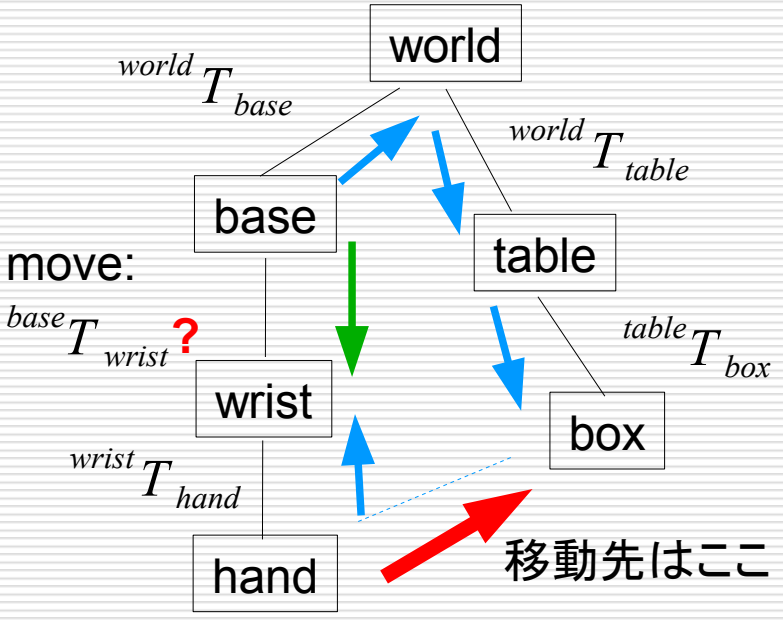
handをboxの位置に持っていく

- move命令には ${}^{base}T_{wrist}$ が必要。どう求めるか。
- **boxとhandが一致しているとして**、変換を逆にたどる。

$${}^{base}T_{world}\ {}^{world}T_{table}\ {}^{table}T_{box}\ {}^{hand}T_{wrist}$$

これはつまり、

$$\left({}^{world}T_{base}\right)^{-1}\ {}^{world}T_{table}\ {}^{table}T_{box}\left({}^{wrist}T_{hand}\right)^{-1}$$

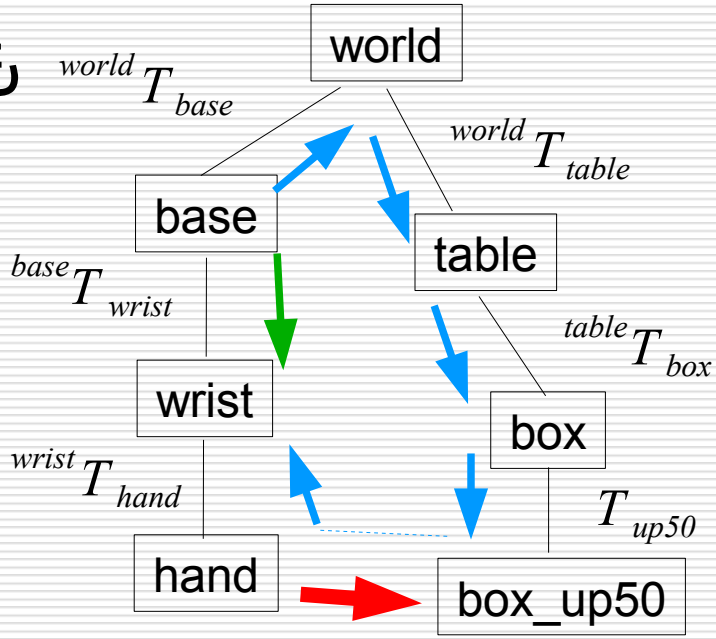


- 理想アームによる位置決め(2)

handをboxの上50mm(単位系?)に持っていく

- 上50の変換は、どこから見ても

$$T_{up50} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 50 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



したがって、

$${}^{base}T_{wrist} = \left({}^{world}T_{base}\right)^{-1} {}^{world}T_{table} \; {}^{table}T_{box} \; T_{up50} \left({}^{wrist}T_{hand}\right)^{-1}$$

- 動作レベルのロボット言語のイメージ

```
# 環境モデルの定義
# アームの情報:
# ハンドの情報:
# 箱の情報:

arm.move(hand, box*up50)
arm.hand.open(100)
arm.move(hand. box)
arm.hand.close(40)
box.unfix()
box.affix(hand)
arm.move(box, place_a*up50)
.
```

$^{world}T_{base}$ $^{wrist}T_{hand}$

$^{world}T_{table}$ $^{table}T_{box}$

- たとえばカメラで物体を認識する場合は、キャリブレーションで $^{world}T_{camera}$ を事前に求めておいて認識命令で $^{camera}T_{box}$ を計測することになる。

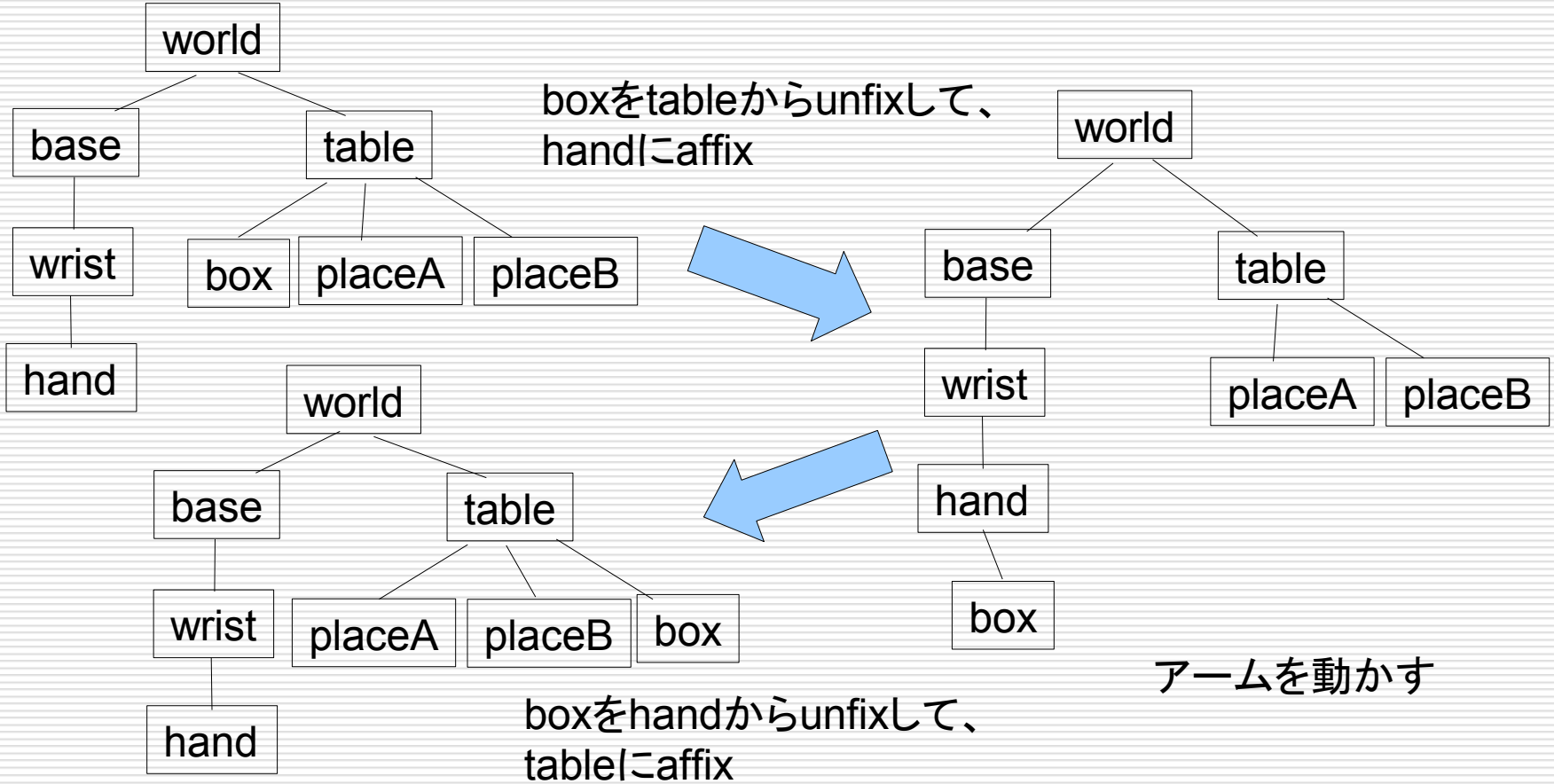
- 座標系ベースロボット言語の機能

- 理想アームは、${}^{base}T_{wrist}$ で動く。
- 「ハンドを箱の位置に動かす」と言いたい。
  - アームの情報: ${}^{world}T_{base}$
  - ハンドの情報: ${}^{wrist}T_{hand}$
  - 箱の情報: ${}^{world}T_{table}\,{}^{table}T_{box}$
  - $${}^{base}T_{wrist} = \left({}^{world}T_{base}\right)^{-1}\,{}^{world}T_{table}\,{}^{table}T_{box}\left({}^{wrist}T_{hand}\right)^{-1}$$

    を自動的に計算する。
- 動かした箱をまたつかめるようにaffix, unfixで座標系を管理する。

- 座標系の連鎖とその変化

boxをplaceAからplaceBへ持っていく



- ロボットの座標系表現

- 理想ロボット：ArmWithHand
  - 構造は関係なし。
  - move：手先や持った物を動かす。
- ハンド：Hand
  - open, close, width

以下はまた後で

- アームのリンクの計算：Link
  - set_joint
- シリアルリンクアーム：LinkedArm

- ArmWithHand

```
class ArmWithHand :
    def __init__(self, base_pos=None,hand=None,init_pos=None) :
        self.base = CoordinateObject()
        if base_pos :
            self.base.set_pos(base_pos)
        if not init_pos :
            init_pos=FRAME()
        self.init_pos=init_pos
        self.wrist = CoordinateObject()
        self.wrist.affix(self.base,init_pos,'rigid')
        self.tools={}
        self.hand=hand
        if hand :
            self.tools['hand']=hand
            hand.affix(self.wrist,hand.rel_trans,'rigid')
```

生成

- ArmWithHand

```
def move(self, smthng, smwhr, wrt=None) :
    if isinstance(smthng, str) :
       smthng = self.tools[smthng]
    if smthng.ancestor(self.wrist) :
       if isinstance(smwhr,CoordinateObject) :
          target=smwhr.where(self.base)
       elif isinstance(smwhr, FRAME) :
          if wrt :
             target = wrt.where(self.base) * smwhr
          else :
             target = (- self.base.where()) * smwhr
       else :
          print "error: wrong place expression"
          return False
       self.wrist.set_trans(target*(- smthng.where(self.wrist)))
    else :
      print "error: cannot move unfixed object"
      return False
```

動作

- Hand

```
class Hand(PartsObject) :
    def __init__(self, trans=None) :
        if not trans :
            trans=FRAME()
        PartsObject.__init__(self)
        if trans :
            self.set_pos(trans)
        self.width=0.0
    def open(self, width) :
        self.widht = width
        print "open"
    def close(self, width) :
        self.widht = width
        print "close"
    def width(self) :
        return self.width
```

生成

指開

指閉

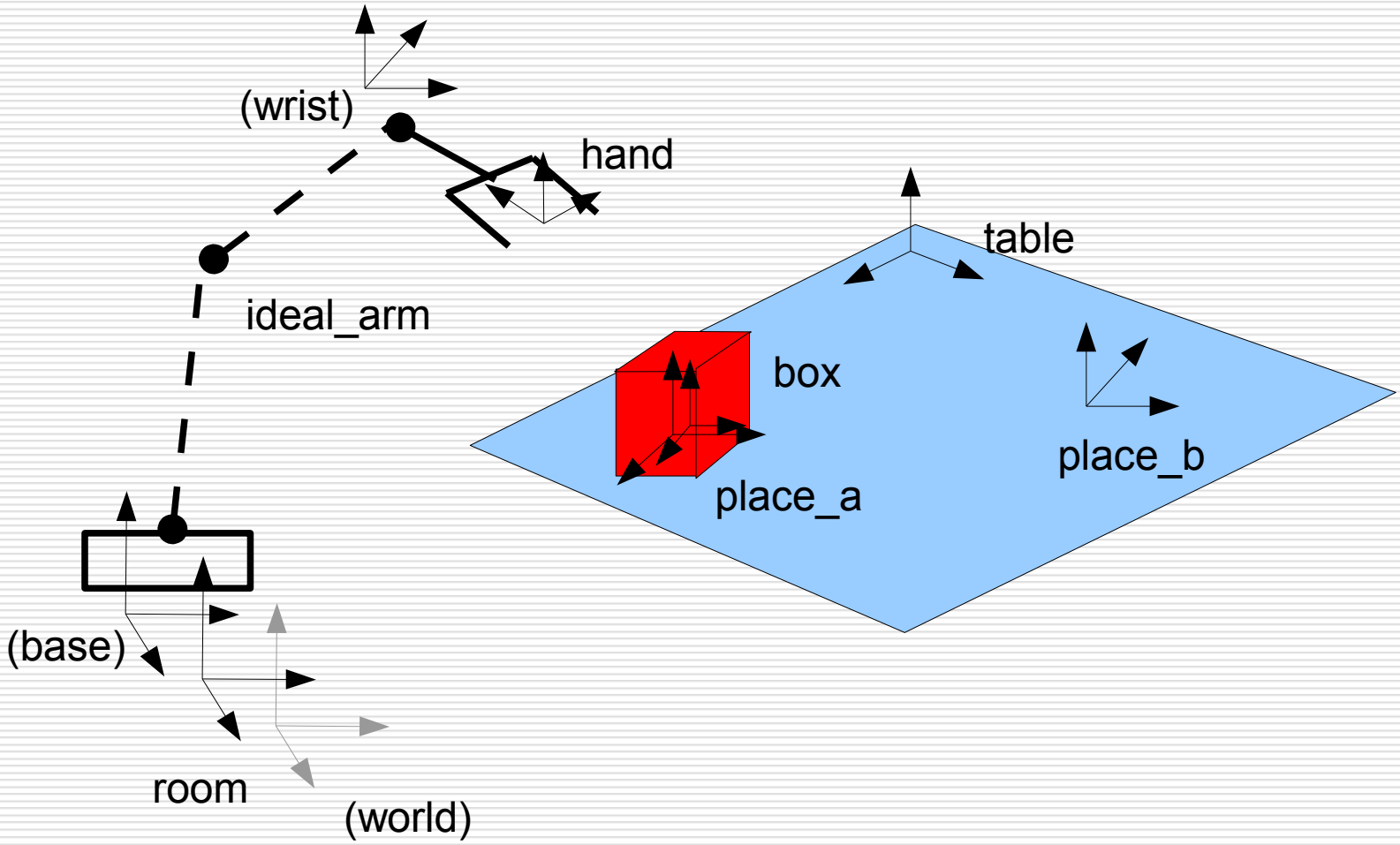
指幅の問い合わせ

- 環境構築:例題8-1

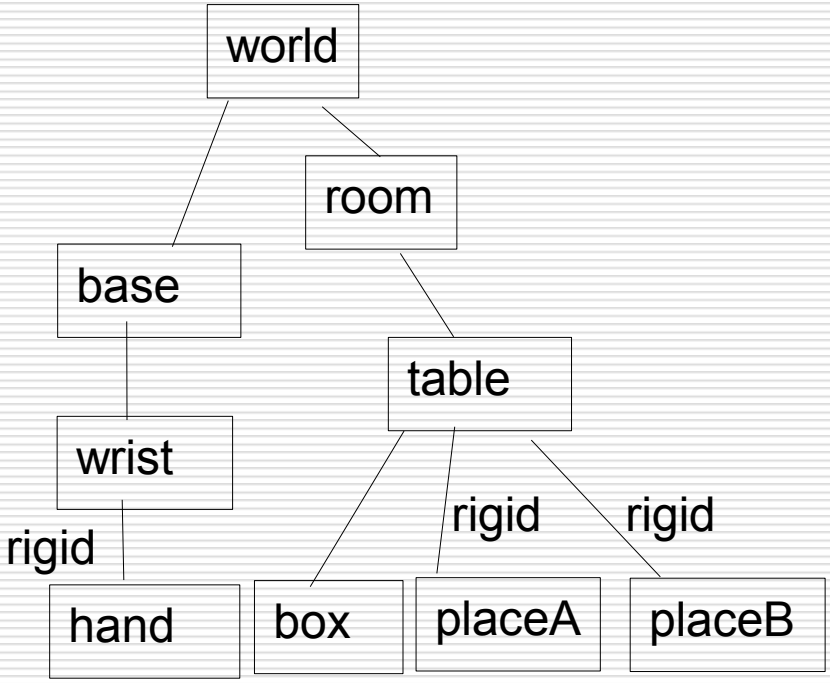
以下のロボット作業環境を生成せよ

- room: ワールドと一致した座標系
- table: room内、高さ0.5、z回りに30度
- place_a: tabel固定、x方向に0.1
- place_b: table固定、x方向に-0.1、z回りに -30度
- box: table上、place_aに一致
- hand: Hand()のインスタンス
- ideal_arm: ArmWithHand, handを持つ

- 例題8-1で構築される環境



- 例題8-1で構築される座標系連鎖



- 座標系連鎖の計算:例題8-2

- place_a、place_bのワールド座標系からの座標変換を求めよ。それを位置x、y、z、α、β、γで表示せよ。
- place_bからみたplace_aへの座標変換(place_a wrt place_b)を求めよ。

- tableをz回りに90度回転させよ。
- place_a、place_bのワールド座標系からの座標変換を求めよ。それを位置x、y、z、α、β、γで表示せよ。
- place_bからみたplace_aへの座標変換を求めよ。

- 作業例(座標管理なし)1:例題8-3

以下のロボット作業を座標系管理(affix, unfix)を使わずに実行せよ。

- (まず上0.2の座標変換up200を生成する)
- handをboxの上0.2に移動する
- handを0.1開く
- handをboxに移動する
- handを0に閉じる

- 作業例(座標管理なし)2:例題8-4

以下のロボット作業の続きを実行せよ

- boxをplace_bに持っていく。
- handを0.1開く
- handをplace_bの上0.2に移動する

- 作業例(座標管理あり):例題8-5

以下のロボット作業をaffix, unfixを用いて座標系を管理しながら実行せよ。

- handをboxの上0.2に移動する
- handを0.1開く
- handをboxに移動する
- handを0に閉じる
- boxをplace_bに持っていく。
- handを0.1開く
- handをplace_bの上0.2に移動する

- 次回予告

- VPythonをつかった3D表示