知能システム論1 (14)

2012.7.11

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

- 分解運動速度制御

- ビジュアルフィードバック、その他センサデータと融合した動作を実現するのに適した制御

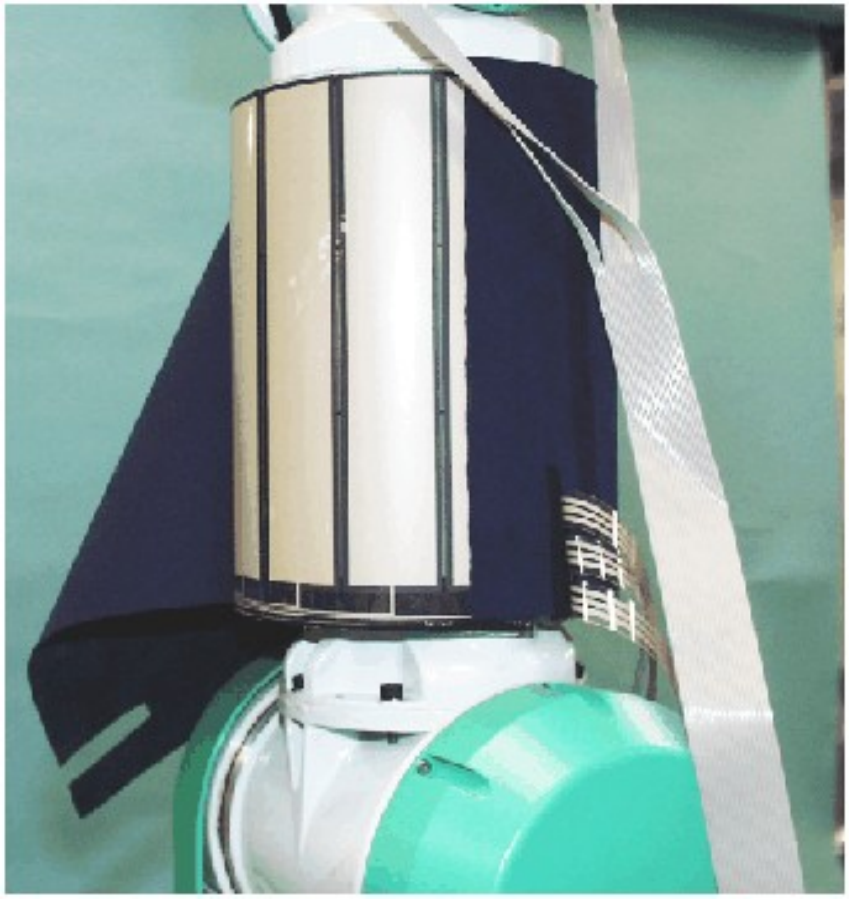
- 逆運動学を明示的には解かない。
- 目標と現状とを少なくするための速度を目標作業空間で求める。
- ヤコビアンを利用して関節パラメタの速度に変換する。
- 関節をその速度で駆動する。

- 分解運動速度制御(実例)

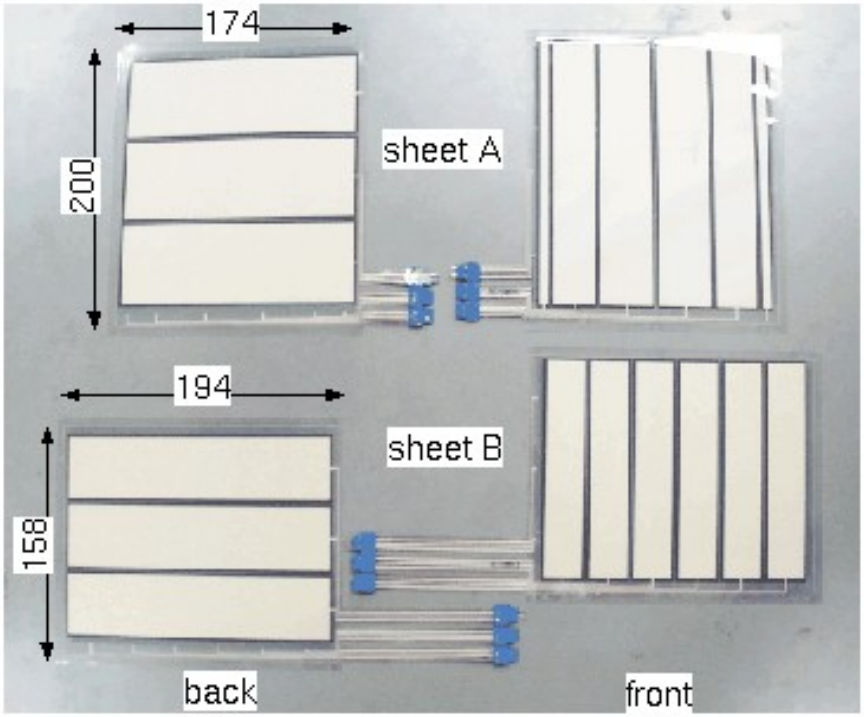
- 7自由度アーム
- 作業動作6自由度と触覚情報(72)との融合

video: avoid

- 全身触覚センサ

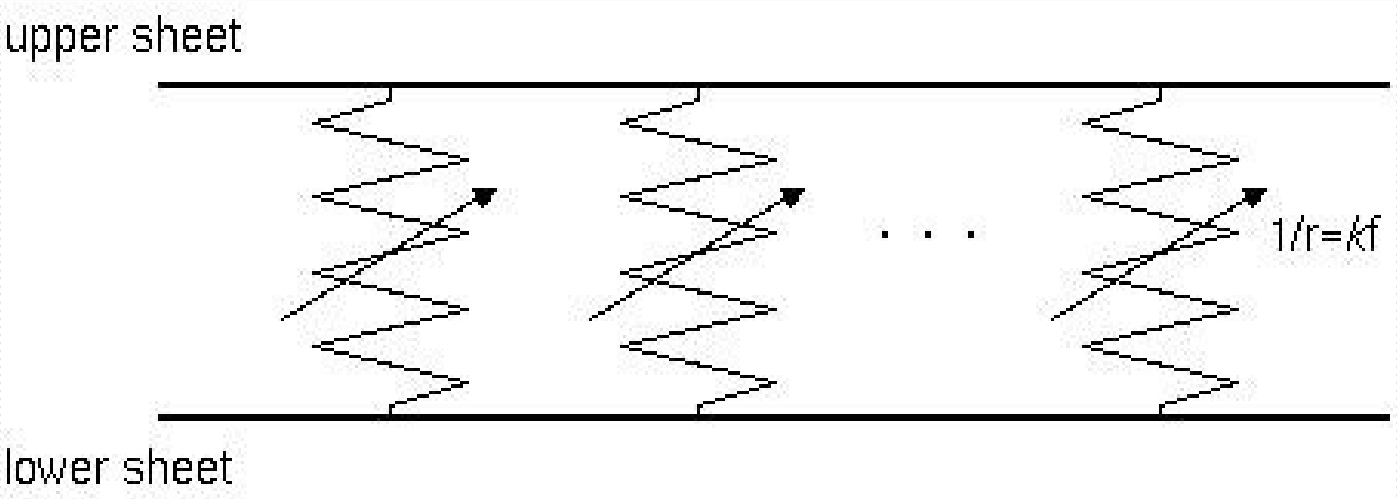


センサの装着状態



センサのシート

- センサシートの等価回路



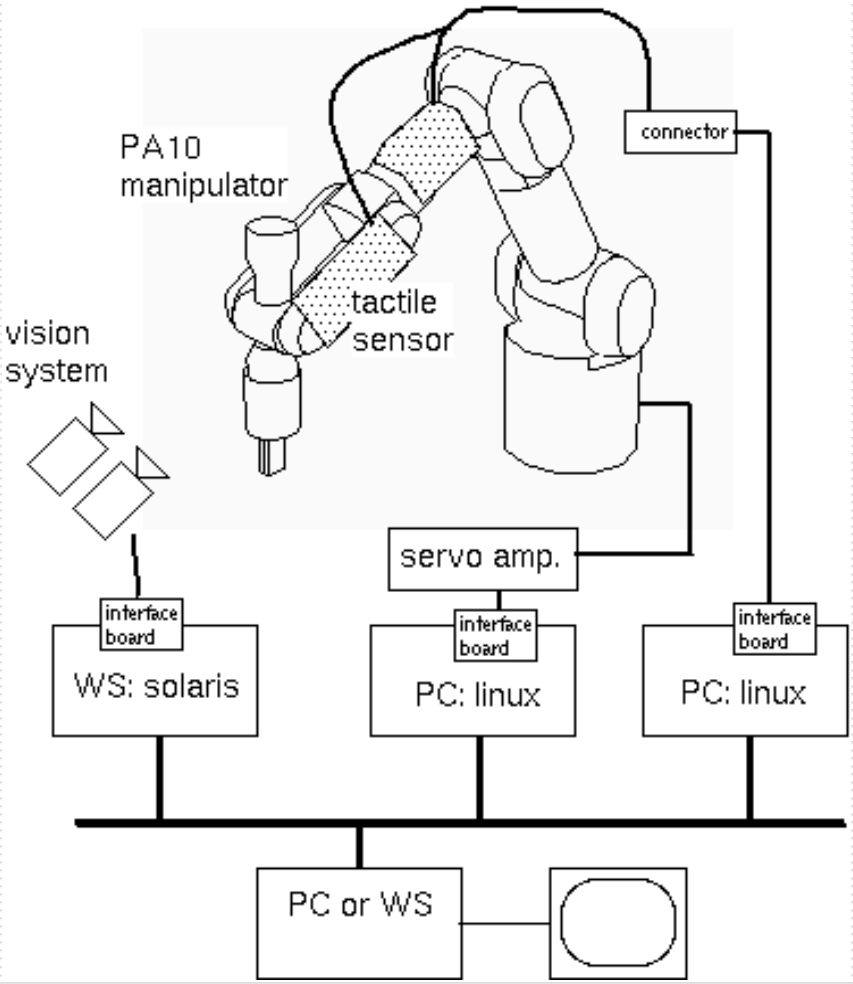
$F = \Sigma f$　　F: total force　　f: partial force

$1/R = \Sigma\, 1/r$　　R: total resistance　　r: patrtial resistance

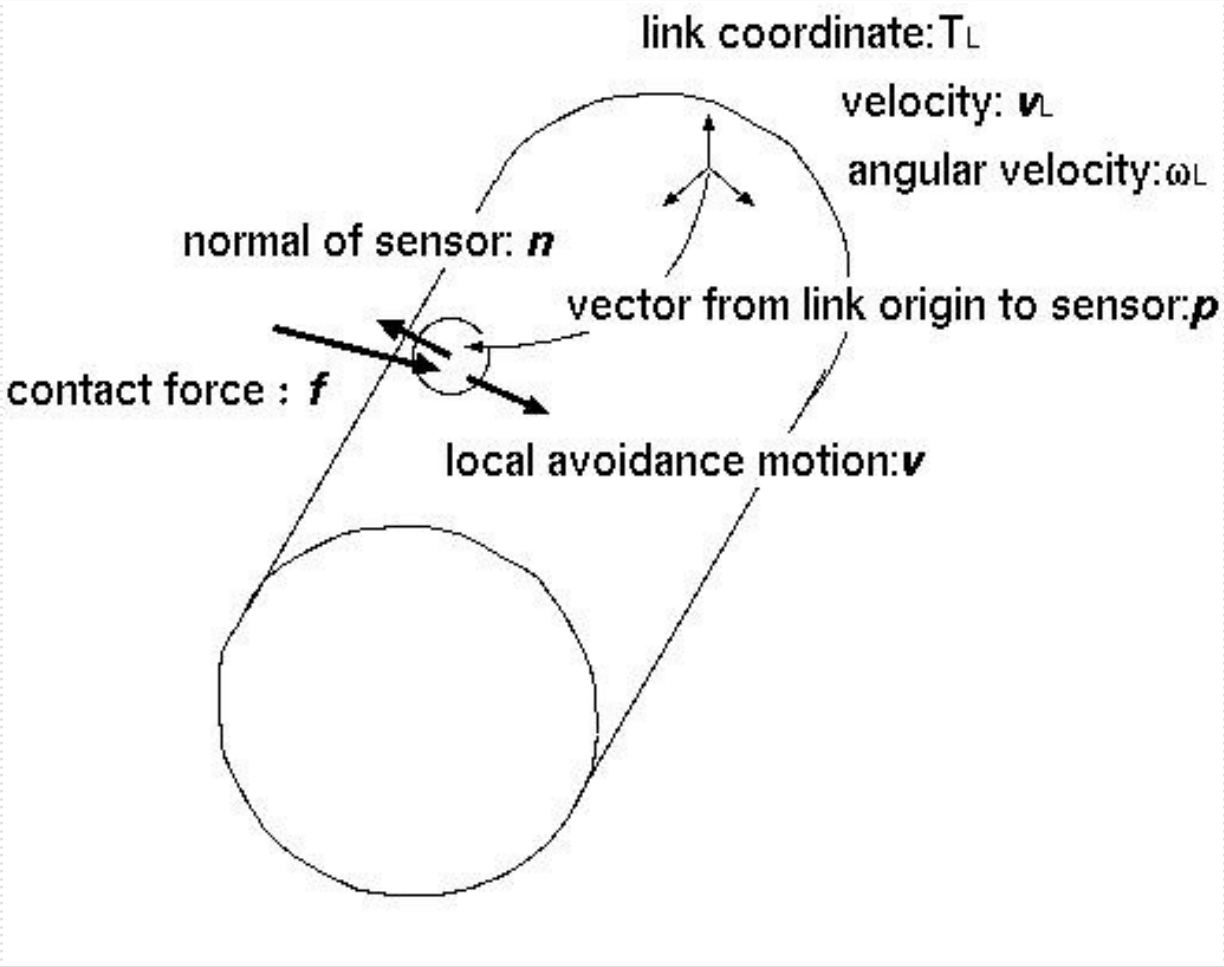
Relation between R and F
$1/R = kF$

Relation between r and f
$1/r = kf$

- システム構成


PA10
manipulator
connector
tactile
sensor
vision
system
servo amp.
interface
board
WS: solaris
interface
board
PC: linux
interface
board
PC: linux
PC or WS

- 接触情報の利用方法


link coordinate:T_L
velocity: v_L
angular velocity:ω_L
normal of sensor: n
vector from link origin to sensor:p
contact force : f
local avoidance motion:v

- 分解運動速度制御の適用

センサ出力　$z = \boldsymbol{n} \cdot \boldsymbol{f}$

回避運動生成　$\boldsymbol{n} \cdot \boldsymbol{v} = k(\boldsymbol{n} \cdot \boldsymbol{f})$

リンク速度との関係　$\boldsymbol{v} = \boldsymbol{v}_{link} + \boldsymbol{\omega}_{link} \times \boldsymbol{p}$

回避速度と関節角速度のヤコビアン　$k\,z = n \cdot (\boldsymbol{v}_{link} + \boldsymbol{\omega}_{link} \times \boldsymbol{p}) = J_{avoid}\,\boldsymbol{u}$

作業動作とヤコビアン　$\begin{pmatrix} \boldsymbol{v}_{hand} \\ \boldsymbol{\omega}_{hand} \end{pmatrix} = J_{task}\,\boldsymbol{u}$

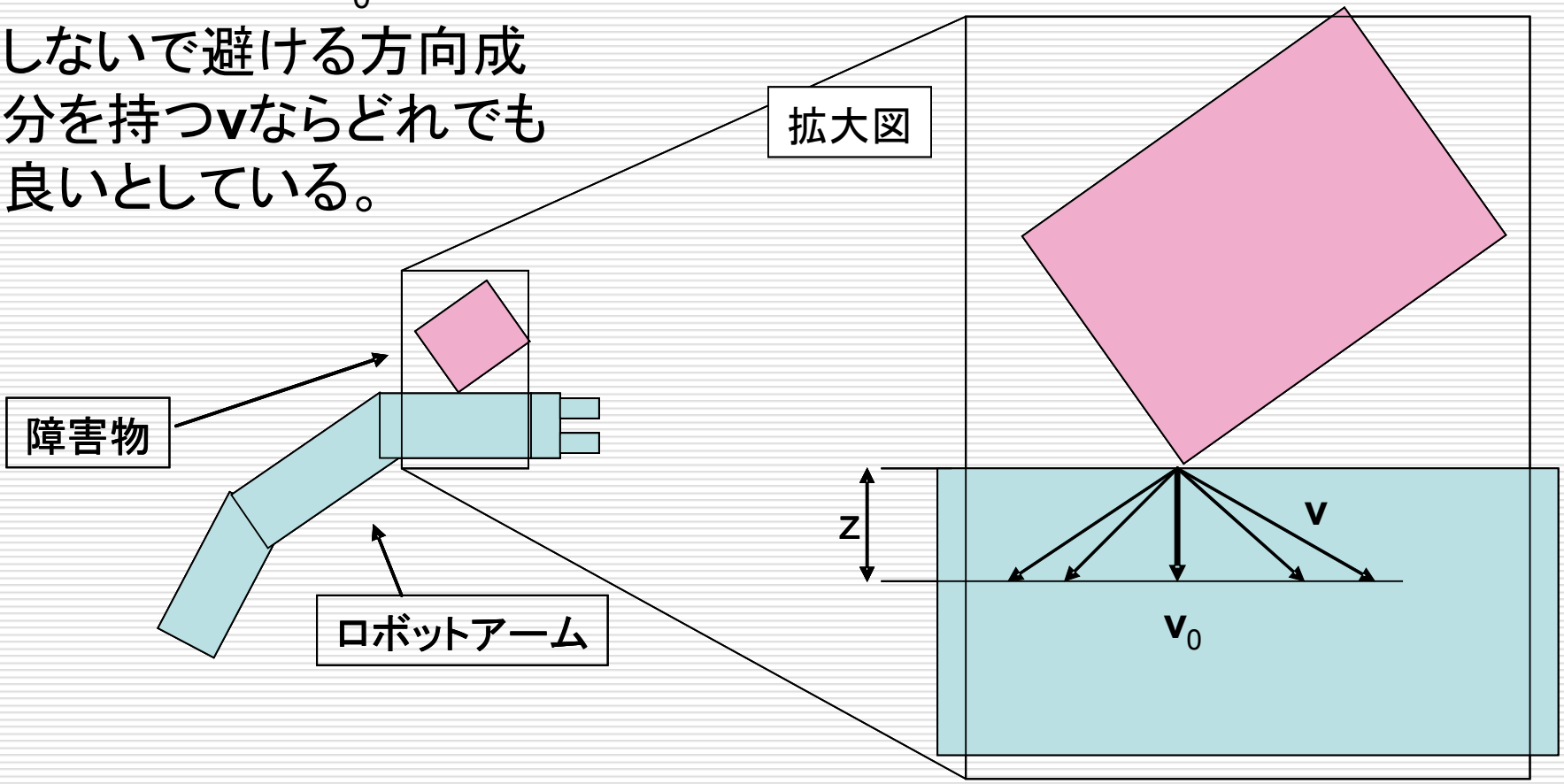
作業と回避の融合　$\begin{pmatrix} \boldsymbol{v}_{hand} \\ \boldsymbol{\omega}_{hand} \\ . \\ k\,z_i \\ . \end{pmatrix} = \begin{pmatrix} J_{task} \\ . \\ J_{\text{avoid-i}} \\ . \end{pmatrix} \boldsymbol{u} \Rightarrow \boldsymbol{u} = \begin{pmatrix} J_{task} \\ . \\ J_{\text{avoid-i}} \\ . \end{pmatrix}^{+} \begin{pmatrix} \boldsymbol{v}_{hand} \\ \boldsymbol{\omega}_{hand} \\ . \\ k\,z_i \\ . \end{pmatrix}$

- 特徴

- 作業動作と接触情報に基づく回避動作との融合
- 多数の接触情報を同時に処理できる
- 特異値分解
  - 過剰拘束、冗長性を自然な形で有効利用
- 作業動作と回避動作の配分を自由に選択できる

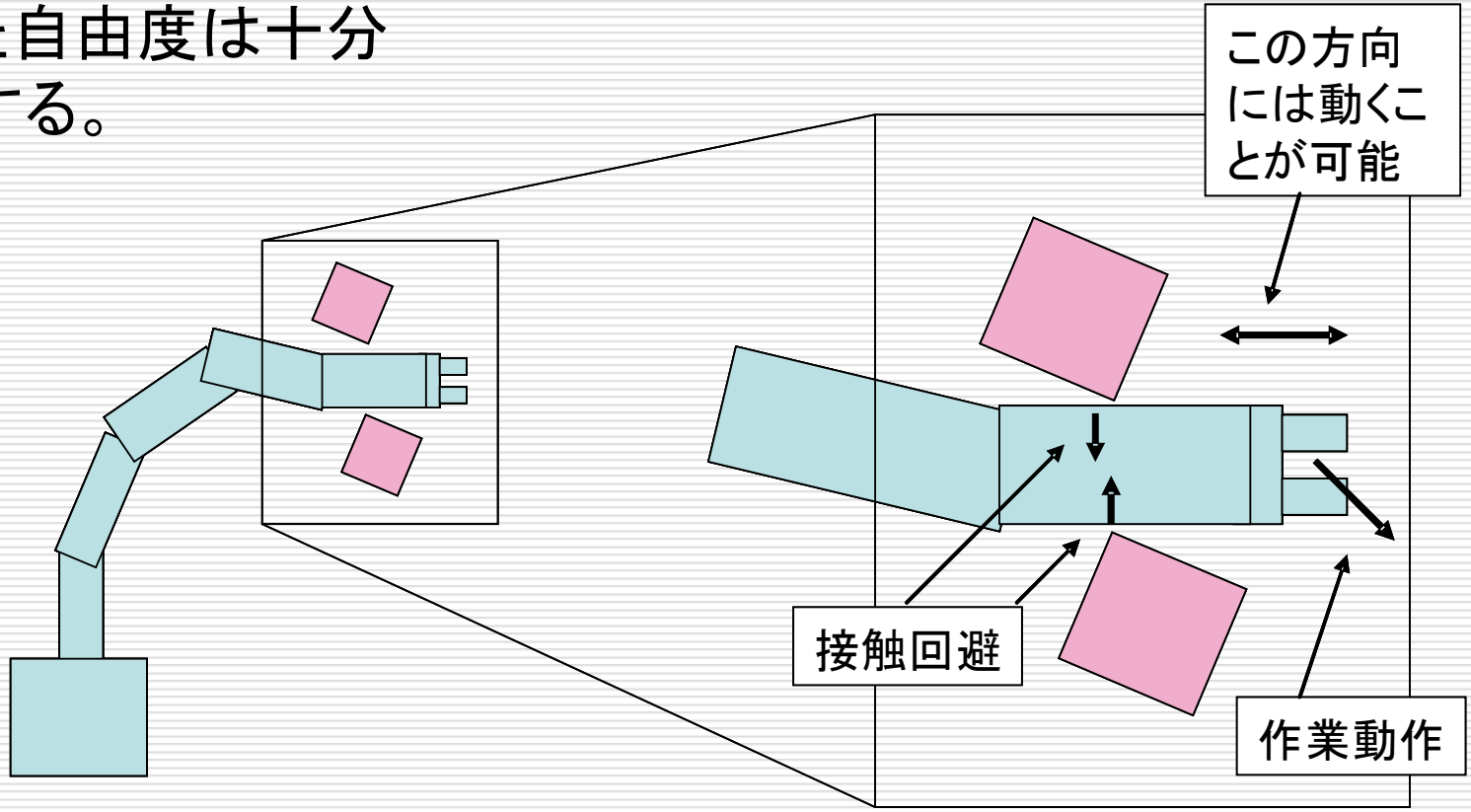
- 回避運動生成の特徴

避ける方向を$\mathbf{v}_0$と固定しないで避ける方向成分を持つvならどれでも良いとしている。

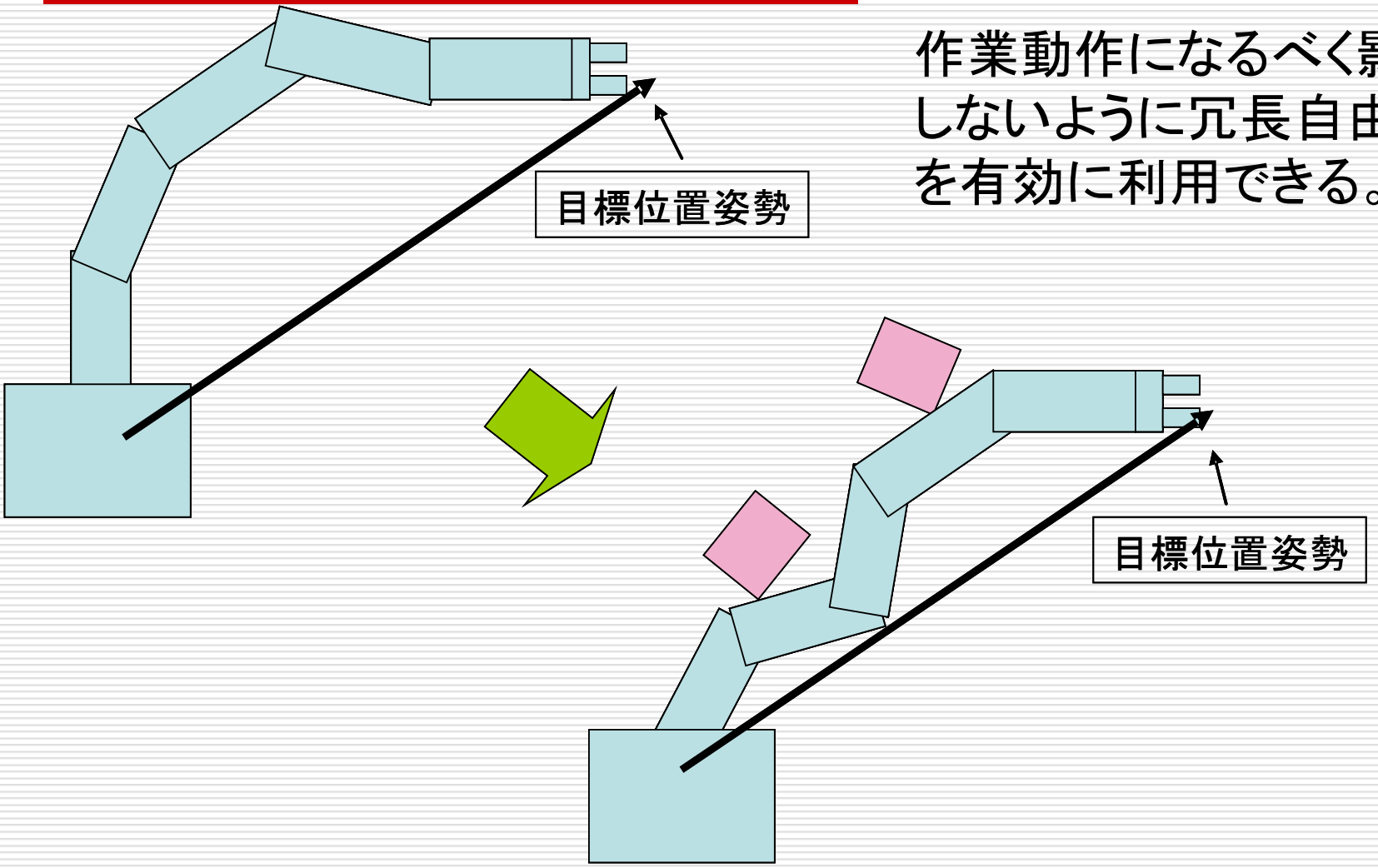


- 過剰拘束の処理

過剰な拘束があっても
残された自由度は十分
に利用する。

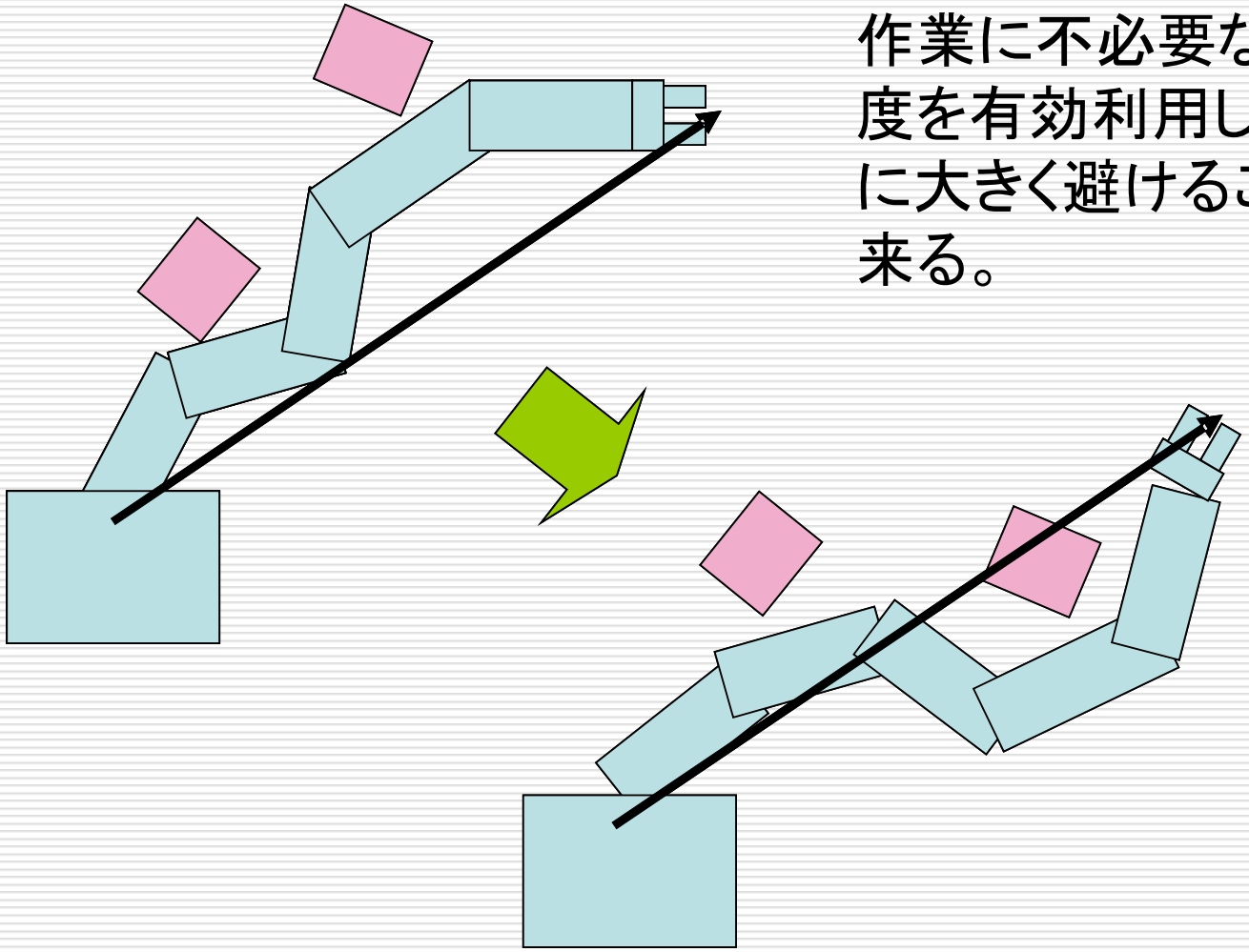


- 冗長自由度の利用



作業動作になるべく影響しないように冗長自由度を有効に利用できる。

- 避けるための自由度の追加



作業に不必要な自由度を有効利用してさらに大きく避けることも出来る。

- 自由な構造のシリアルアームを作る

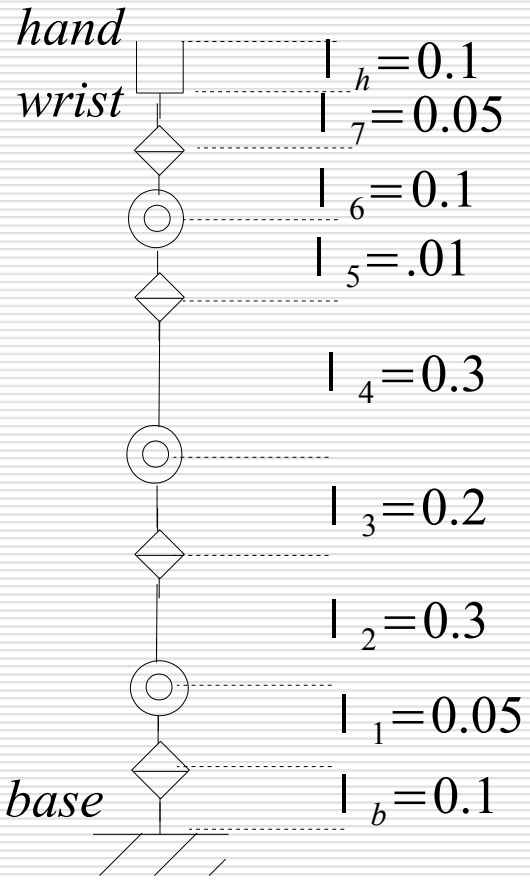
larm_w_hand_arm_sol.py

- arm6dofのうち以下の関数は関節数がいくつでも使える。
  - solve()
  - ready(), park()
  - Move()
- これを使って自由な構造のアームが作れる

- 好きに作ってみよう

- まず関節構造のデザイン
  - 自由度、関節軸、座標系、寸法
- 次に形状のデザイン
  - リンク原点と形状原点の位置関係に注意

- 7自由度アームの例




```
def make_shape(self) :
    self.bh = 0.1
#
    self.l1h = 0.05
    self.links[0].place_joint(self.base, FRAME(xyzabc=[0,0,self.bh,0,0,0]),5)
#
    self.l2h = 0.3
    self.links[1].place_joint(self.links[0], FRAME(xyzabc=[0,0, self.l1h,0,0,0]),4)
#
    self.l3h=0.2
    self.links[2].place_joint(self.links[1], FRAME(xyzabc=[0,0, self.l2h,0,0,0]),5)
#
    self.l4h = 0.3
    self.links[3].place_joint(self.links[2], FRAME(xyzabc=[0,0, self.l3h,0,0,0]),4)
#
    self.l5h = 0.1
    self.links[4].place_joint(self.links[3], FRAME(xyzabc=[0,0, self.l4h,0,0,0]),5)
#
    self.l6h = 0.1
    self.links[5].place_joint(self.links[4], FRAME(xyzabc=[0,0, self.l5h,0,0,0]),4)
#
    self.l7h = 0.05
    self.links[6].place_joint(self.links[5], FRAME(xyzabc=[0,0, self.l6h,0,0,0]),5)
#
    self.wrist.affix(self.links[6],FRAME(xyzabc=[0,0, self.l7h,0,0,0]))
```

- 7自由度アームの例

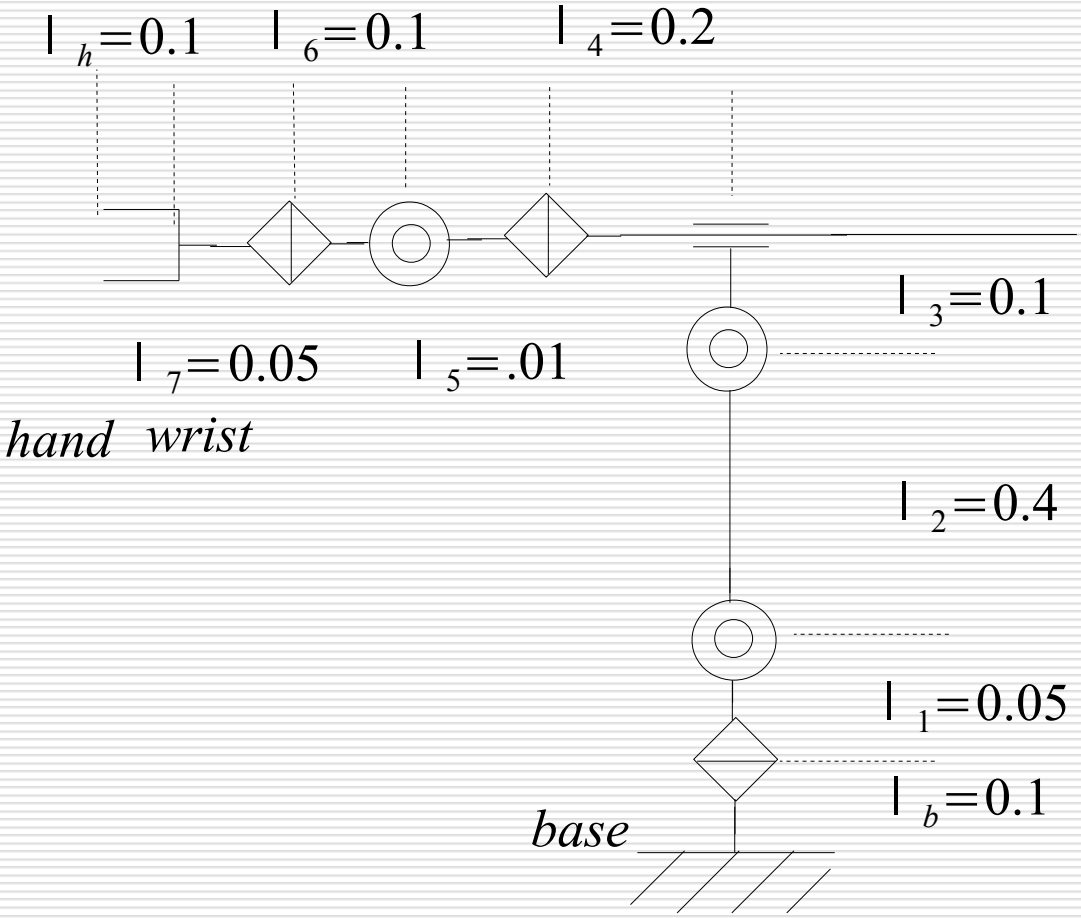
arm7dof.py：7自由度アームの例

- make_shape(), creat_arm()を作る
- ready_angle, park_angleを関節数にあわせることに注意

env.py：汎用環境

- 読み込むアームを替える

- もうひとつの例


l h=0.1
l 6=0.1
l 4=0.2
l 3=0.1
l 7=0.05
l 5=.01
hand
wrist
l 2=0.4
l 1=0.05
l b=0.1
base

- 次回予告

- 作業プログラミングシステムの構築
  - レゴブロック遊び
- 最終課題