優先度付き逆運動学による動作生成
リファレンスマニュアル

平成 32 年 1 月 27 日

平岡直樹
hiraoka@jsk.t.u-tokyo.ac.jp

目 次

1 優先度付き逆運動学の基礎

1.1 優先度付き逆運動学について

hogehoge

1.2 章の構成

第 2 章では，クラスを説明する．

2 優先度付き逆運動学

prioritized-inverse-kinematics *tasks variables &key (regular 1.000000e-06) (regular-rel 1.0) (regular-max 0.1) (task0 nil) (stop 50) (min-loop 0) (revert-if-fail t) (debug-view :no-message) (qp-solver #'solve-osqp-qp) (qp-args (list :solve-qp-mode :sqp-with-hotstart))* [function]

- tasks : (list (list task1-1 task1-2 ...)
  (list task2-1 task2-2 ...)
  (list task3-1 task3-2 ...)
  ...)
  各 *task* は *inverse-kinematics-task* クラスである．*prioritized-inverse-kinematics* は次の問題を解く．priority 1 のタスクが最も優先度が高い．

$$
\begin{aligned}
\text{priority1} &: \min_{\boldsymbol{x}} \ \text{task1-1} + \text{task1-2} + ... + \boldsymbol{x}^{\mathrm{T}}\mathbf{W}\boldsymbol{x} \\
\text{priority2} &: \min_{\boldsymbol{x}} \ \text{task2-1} + \text{task2-2} + ... + \boldsymbol{x}^{\mathrm{T}}\mathbf{W}\boldsymbol{x} \\
\text{priority3} &: \min_{\boldsymbol{x}} \ \text{task3-1} + \text{task3-2} + ... + \boldsymbol{x}^{\mathrm{T}}\mathbf{W}\boldsymbol{x} \\
&\vdots
\end{aligned}
$$

- variables : (list variable1 variable2 ...)
  探索変数のリストである．各 *variable* は *joint* クラスまたは *inverse-kinematics-variable* クラスである．*joint* クラスが与えられた場合，その *joint-angle* が探索変数となる．同一の探索変数を意味する *variable* を *2* 回以上与えてはならない．
- *regular, regular-rel, regular-max : weight or (list weight1 weight2 weight3 ...)*
  各 priority の QP において解を一意に定めるための正則化重みの大きさである．各 *weight* は *float* クラスである．*regular* として *float* が与えられた場合は全 priority 共通でその重みを用い，*list* が与えられた場合は各 priority ごとに指定された重みを用いる．w = regular，w_r = regular-rel，w_{max} = regular-max とし，各 priority の現在のエラーの 2 乗和を e とおくと，重みは $\min(w + w_r e, w_{max})$ となる．

- task0 : (list task0-1 task0-2 ...)
  各 *task* は *inverse-kinematics-task* クラスである．*task0* が与えられた場合，*prioritized-inverse-kinematics* は次の問題を解く．

$$\begin{aligned} \text{priority0} &: \min\ \text{task0-1} + \text{task0-2} + ... \\ \text{priority1} &: \min\ \text{task1-1} + \text{task1-2} + ... \\ \text{priority2} &: \min\ \text{task2-1} + \text{task2-2} + ... \\ \text{priority3} &: \min\ \text{task3-1} + \text{task3-2} + ... \\ &\vdots \end{aligned}$$

  ただし，priority 0 の各タスクの最適値を同時に満たす解が必ず存在すると仮定し，priority 0 では QP を解かないことで高速化を図る．したがって，*task0* としてこの仮定が成り立たないタスクを与えるべきでない．

- *stop*
  stop 回の反復計算後，直ちに終了する．*stop* は *interger* クラスである．

- *min-loop*
  min-loop 回の反復計算後から，各反復終了時に全 *task* の終了条件を満たすなら直ちに終了する．*min-loop* は *interger* クラスである．

- *revert-if-fail : t or nil*
  t ならば，終了時にある task の終了条件を満たして無い場合に，初期状態に戻してから *prioritized-inverse-kinematics* から返る

- *debug-view*
  t ならば反復計算中に状態を描画しデバッグメッセージを表示する．*:no-message* ならば反復計算中に状態を描画する．*nil* ならば何もしない．

- *qp-solver*
  QP のソルバを指定する．*qp-solver* は *function* クラスである．

- *qp-args*
  prioritized-inverse-kinematics 内で，*(apply qp-solver ... qp-args)* の形で *qp-solver* が呼ばれる．

inverse-kinematics-task [class]

:super **propertied-object**
:slots (a $\mathbf{A}$)
(b $\boldsymbol{b}$)
(c $\mathbf{C}$)
(dl $\mathbf{d}_l$)
(du $\mathbf{d}_u$)
(wa $\mathbf{W}_A$)
(wc $\mathbf{W}_C$)
(asparce $\mathbf{A}_{sparce}$, used for sparce matrix calculation)
(csparce $\mathbf{C}_{sparce}$, used for sparce matrix calculation)
(equality-rows size of row of $\mathbf{A}$)
(inequality-rows size of row of $\mathbf{C}$)
(cols size of column of $\mathbf{A}$ (constant))

各タスクを表すクラス.
prioritized-inverse-kinematics 中で次の問題に変換される.

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{\omega}} \quad & (\mathbf{A}\boldsymbol{x}-\boldsymbol{b})^{\mathrm{T}}\mathbf{W}_A(\mathbf{A}\boldsymbol{x}-\boldsymbol{b})+\boldsymbol{\omega}^{\mathrm{T}}\mathbf{W}_C\boldsymbol{\omega} \\ \text{subject to} \quad & \boldsymbol{d}_l \leqq \mathbf{C}\boldsymbol{x}+\boldsymbol{\omega} \leqq \boldsymbol{d}_u \end{aligned}$$

:init [method]
Initialize instance

:initialize *variable-length variables* [method]
called at the start of prioritized-inverse-kinematics

:update [method]
called at the start of each iteration.
update A, b , Wa, Asparce, equality-rows, C, dl, du , Wc, Csparce, inequality-rows

:is-satisfied [method]
終了判定に用いる.

:draw [method]
debug view

:debug [method]
debug message

:a [method]
return $\mathbf{A}$

:b [method]
return $\boldsymbol{b}$

:wa [method]
return $\mathbf{W}_A$

:asparce [method]
return $\mathbf{A}_{sparce}$

:c [method]
return $\mathbf{C}$

:dl [method]
return $\boldsymbol{d}_l$

:du [method]
return $\boldsymbol{d}_u$

:wc [method]
return $\mathbf{W}_C$

:csparce [method]
return $\mathbf{C}_{sparce}$

minmax-angle-task [class]

:super **inverse-kinematics-task**
:slots (j , target joint)
(max-angle)
(min-angle)
(target-variable , list of inverse-kinematics-variable corresponding to target joint)
(check 終了判定を行うか否か)
(check-margin 終了判定時のマージン)

joint の角度上下限制約を表現するクラス.
joint の角度上下限を $\boldsymbol{\theta}_{max}$，$\boldsymbol{\theta}_{min}$，現在の角度を $\boldsymbol{\theta}$ とすると，*prioritized-inverse-kinematics* 中で次の問題に変換される.

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{\omega}} \quad & \boldsymbol{\omega}^{\mathrm{T}} W \boldsymbol{\omega} \\ \text{subject to} \quad & \boldsymbol{\theta}_{min} \leqq \boldsymbol{\theta} + \boldsymbol{x} + \boldsymbol{\omega} \leqq \boldsymbol{\theta}_{max} \end{aligned} \tag{2.1}$$

$\boldsymbol{\theta}$ の単位は m，rad である.

:init *_joint &key ((:w _w) 1.0)* [method]
((:max-angle _max-angle) (send _joint :max-angle))
((:min-angle _min-angle) (send _joint :min-angle))
((:check _check) t)
((:check-margin _check-margin) 0.0)

- joint
  joint クラス. 上下限を考える対象の関節である.
- *W*
  float クラスの場合 **W** はその値を対角成分に並べた行列になる
  vector クラスの場合 **W** は各値を対角成分に並べた行列になる
  matrix クラスの場合 **W** としてそのまま使用される.
- *max-angle, min-angle*
  float または *float-vector* クラス．関節上下限を表す.
- *check : t or nil*
  終了判定を行うか否か
- *check-margin float* クラス．終了判定時にこの値以下の侵犯を許容する．単位は m,rad

:initialize *variable-length variables* [method]
探索変数 *variables* のうち，該当 *joint* に影響を与える成分を調べ，記録する．*0* または *1* 個の *variables* が発見されなければならない.

:update [method]
現在の関節角度に応じて $\boldsymbol{d}_l$，$\boldsymbol{d}_u$ を更新する

:is-satisfied [method]
関節角度上下限を満足しているかどうかを判定

:debug [method]
現在の関節角度と上下限を表示

move-target-task [class]

:super **inverse-kinematics-task**
:slots (target-coords)
(move-target)
(translation-axis ,represented in translation-coords)
(translation-coords)
(wtrans $\mathbf{W}_{trans}$, represented in translation-coords)
(rows-trans size of row of $\boldsymbol{e}_{trans}$)
(rotation-axis ,represented in rotation-coords)
(rotation-coords)
(wrot $\mathbf{W}_{rot}$, represented in rotation-coords)
(rows-rot size of row of $\boldsymbol{e}_{rot}$)
(target-coords-variables , target-coords に影響を与える variable のリスト)
(move-target-variables , move-target に影響を与える variable のリスト)
(check 終了判定を行うか否か)
(thre 終了判定時の並進許容誤差)
(rthre 終了判定時の回転許容誤差)
(b-raw , min-max 適用前の $\boldsymbol{b}$)
(p-limit 一回の反復計算で動く並進ノルムの大きさの上限 [m])
(r-limit 一回の反復計算で動く回転ノルムの大きさの上限 [rad])
(tmp-v0)
(tmp-v1)
(tmp-v2)
(tmp-v3)
(tmp-v3a)
(tmp-v3b)
(tmp-m66)
(tmp-m33)

2つの *coordinates* を一致させるタスクを表すクラス．*move-target* と *target-coords* を一致させる．*move-target* と *target-coords* はどちらも動いてよい．
move-target と *target-coords* の並進誤差を *translation-coords* の座標系で表現し，*translation-axis* によって抽出された成分を $\boldsymbol{e}_{trans}$ と表す．*move-target* と *target-coords* の回転誤差を *rotation-coords* の座標系で表現し，*rotation-axis* によって抽出された成分を $\boldsymbol{e}_{rot}$ と表す．$\boldsymbol{e}_{trans}$，$\boldsymbol{e}_{rot}$ のヤコビアンをそれぞれ $\mathbf{J}_{trans}$，$\mathbf{J}_{rot}$ とすると，*prioritized-inverse-kinematics* 中で次の問題に変換される．

$$\min_{\boldsymbol{x}} \quad (\mathbf{J}_{trans}\boldsymbol{x}-\boldsymbol{e}_{trans})^{\mathrm{T}}\mathbf{W}_{trans}(\mathbf{J}_{trans}\boldsymbol{x}-\boldsymbol{e}_{trans})+(\mathbf{J}_{rot}\boldsymbol{x}-\boldsymbol{e}_{rot})^{\mathrm{T}}\mathbf{W}_{rot}(\mathbf{J}_{rot}\boldsymbol{x}-\boldsymbol{e}_{rot})$$

$\boldsymbol{e}_{trans}$ の単位は m，$\boldsymbol{e}_{rot}$ の単位は rad である．

:init *_target-coords _move-target &key ((:translation-axis _translation-axis) t) ((:rotation-axis _rotation-axis) t) ((:transform-coords _translation-coords) (make-coords)) ((:rotation-coords _rotation-coords) (make-coords)) ((:wtrans _wtrans) 1.0) ((:wrot _wrot) 1.0) ((:check _check) t) ((:thre _thre) 0.001)* [method]

((:rthre _rthre) (deg2rad 1))
((:p-limit _p-limit) 0.1)
((:r-limit _r-limit) 0.5)

- target-coords, move-target
  coordinate クラス. どちらも動いて良い
- *translation-axis : t :x :y :z :xy :yx :yz :zy :zx :xz nil*
- *rotation-axis : t :x :y :z nil*
- *translation-coords*
  translation-axis 及び *Wtrans* は *translation-coords* 系で表現される
- *rotation-coords*
  rotation-axis 及び *Wrot* は *rotation-coords* 系で表現される
- *Wtrans*
  float クラスの場合 $\mathbf{W}_{trans}$ はその値を対角成分に並べた行列になる
  vector クラスの場合 $\mathbf{W}_{trans}$ は各値を対角成分に並べた行列になる
  matrix クラスの場合 $\mathbf{W}_{trans}$ としてそのまま使用される.
- *Wrot*
  float クラスの場合 $\mathbf{W}_{rot}$ はその値を対角成分に並べた行列になる
  vector クラスの場合 $\mathbf{W}_{rot}$ は各値を対角成分に並べた行列になる
  matrix クラスの場合 $\mathbf{W}_{rot}$ としてそのまま使用される.
- *check : t or nil*
  終了判定を行うか否か
- *thre, rthre*
  float クラスまたは *float-vector* クラス．終了判定時にこの値以下の侵犯を許容する．単位は m,rad
- p-limit, r-limit
  一回の反復計算で動く並進ノルム・回転ノルムの上限．単位は m,rad. 特に回転については変位が大き過ぎると線形近似誤差の影響によって計算が収束しない.

:initialize *variable-length variables* [method]
探索変数 *variables* のうち，*target-coords,move-target* に影響を与える成分を調べ，記録する.

:update [method]
現在の状態近傍で線形近似し $\mathbf{A}$, $\boldsymbol{b}$ を更新する

:is-satisfied [method]
move-target と target-coords が一致しているかどうかを判定

:draw [method]
move-target と target-coords を描画

:debug [method]
現在のエラーを表示する

inverse-kinematics-variable [class]

:super **propertied-object**
:slots (dim , dimension of variable)
(index , index から index + dim - 1] 番目の $\boldsymbol{x}$ の要素が対応する)

(initial-state , prioritized-inverse-kinematics の初期状態)

探索変数を表現するクラス

:init [method]

Initialize instance

:index *&optional idx* [method]

update or return index

:apply-x *x* [method]

計算された $\boldsymbol{x}$ の対応する要素を実際に適用する

:revert [method]

prioritized-inverse-kinematics の初期状態に戻す

:init-form [method]

prioritized-inverse-kinematics の開始時に一回呼ばれる.*inverse-kinematics-task* の*:initialize* より前に呼ばれる

:cleanup-form [method]

prioritized-inverse-kinematics の終了時に *unwind-protect* を用いて一回呼ばれる.

:dim [method]

return dim

joint-variable [class]

:super **inverse-kinematics-variable**
:slots (j , joint)

joint のクラス.

:init *_joint* [method]

joint の *joint-angle* に相当する探索変数. $\boldsymbol{x}$ は *joint* の変位に相当する. $\boldsymbol{x}$ の単位は *degree, rad*.

:apply-x *x* [method]

$\boldsymbol{x}$ の値だけ joint-angle を相対的に更新する.

:revert [method]

prioritized-inverse-kinematics の初期状態に戻す

:joint [method]

return joint

virtual-joint-variable [class]

:super **joint-variable**
:slots (child , virtual joint の子リンク)
(parent , virtual joint の親リンク)

仮想関節の joint-angle に相当する探索変数．$\boldsymbol{x}$ は仮想関節の変位に相当する．$\boldsymbol{x}$ の単位は degree, rad.

:init *_child &key (joint-type 6dof-joint)* [method]
((:parent _parent) (instance bodyset-link :init (make-cascoords :pos (copy-object (send _child :world
joint-args

Initialize instance

:init-form [method]

仮想関節を取り付ける

:cleanup-form [method]

仮想関節を除去する