エージェントシステム第３講義

ソフトウェアエンジニアリング（終）

動作計画

2011-05-18

出杏光　魯仙
であんこう　ろせん

内容

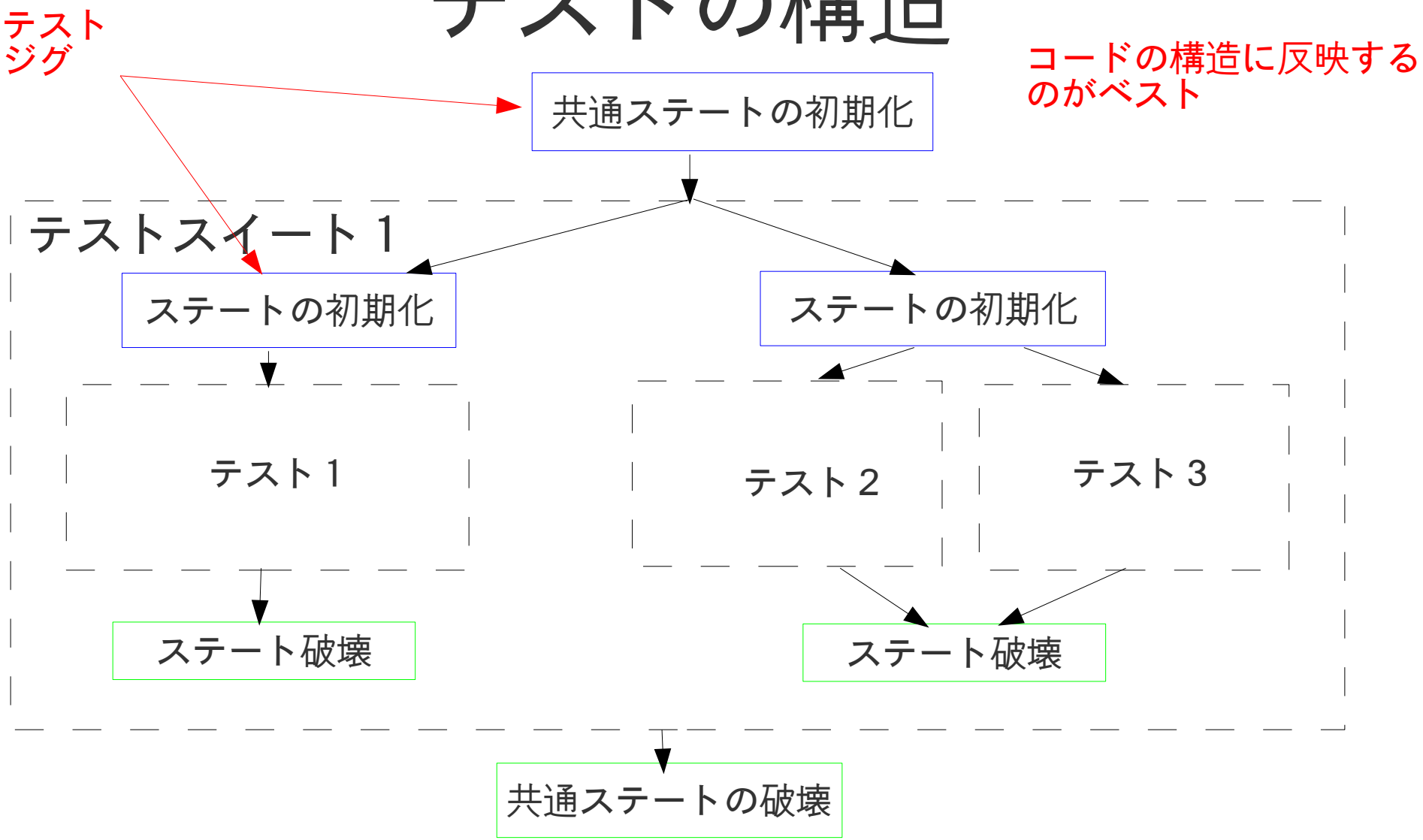
- ソフトウェアエンジニアリング
  - Continuous Integrationの復習
  - ドキュメント
  - リリース
- 動作計画
  - 紹介
  - アルゴリズム

誰でも
使用、信頼、拡張出来る
ソフトウェアの実現
が
現在の技術と
将来のソフトウェア
の柱になる

丈夫なソフトウェア目標

- 使用してもらえるために
  - **リリース**版を頻繁に出す
  - 変更された分を明確にする
  - **ドキュメント**：APIとチュートリアル
- 信頼してもらえるために
  - 書いてある機能が１００％動く保証が出来る**テスト**
  - **ドキュメント**：テスト方法
- 拡張してもらえるために
  - API：アプリケーションプログラミングインタフェース
  - **ドキュメント**：開発者むけ・メモ

テストの構造


テスト
ジグ
コードの構造に反映する
のがベスト
共通ステートの初期化
テストスイート１
ステートの初期化
ステートの初期化
テスト１
テスト２
テスト３
ステート破壊
ステート破壊
共通ステートの破壊

テスト

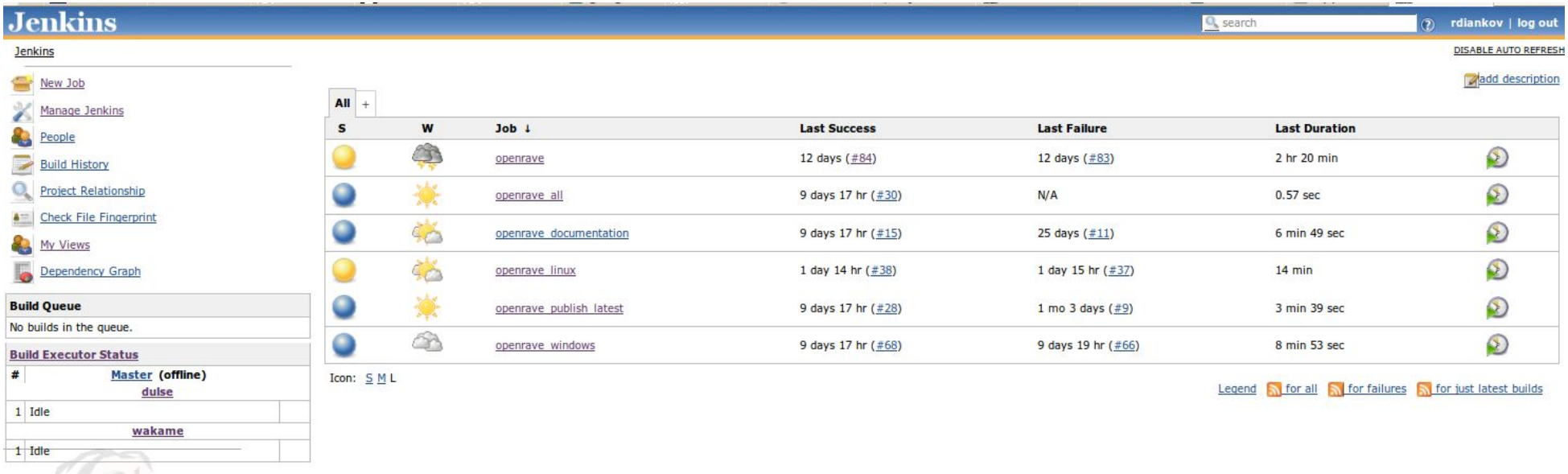
- 例外条件の再現
- 独立性
  - エラー発生の時に、原因に速く絞れる仕組み
- 再現性
  - 失敗がまれな時でもいつも失敗させる
- 再利用性
  - 10個の環境があれば一つのテストを10回実行する
- OSに依存しないように

Continuous Integration

品質管理の連続的なプロセス

Jenkins

- チェックアウト、ビルド、実行、テストのスクリプトをジョブで管理する
- PCクラスターの登録、パラレルに走らせる
- エラー発生時にメール送信
- テスト結果を履歴に保存する



ドキュメントステム

ドキュメントステム概要

- 目的
  - 「このソフトに何が入っているか、どうやって使えばよいか」を自動的に答えられるシステム
- 問題
  - コードの頻繁更新・変化
  - 変化に伴ってドキュメントの書く場所を理解する

ドキュメント方法：論文

- 利点
  - 概念・ニーズ・比較・結果が伝わる
  - 新規性が伝わりやすい
- 欠点
  - 時代遅れになってしまう
  - 詳細を書かないことになっている
    - コードの使い方、便利ツールの関係
  - 少数人で書く
  - 論文の結果を再現出来る方法を書かない

例：Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R, Ng A: An open-source robot operating system (ROS). In: ICRA Workshop on Open Source Software in Robotics, 2009.

例：COLLADA https://collada.org/mediawiki
ROS http://ros.org

ドキュメント方法：Wiki

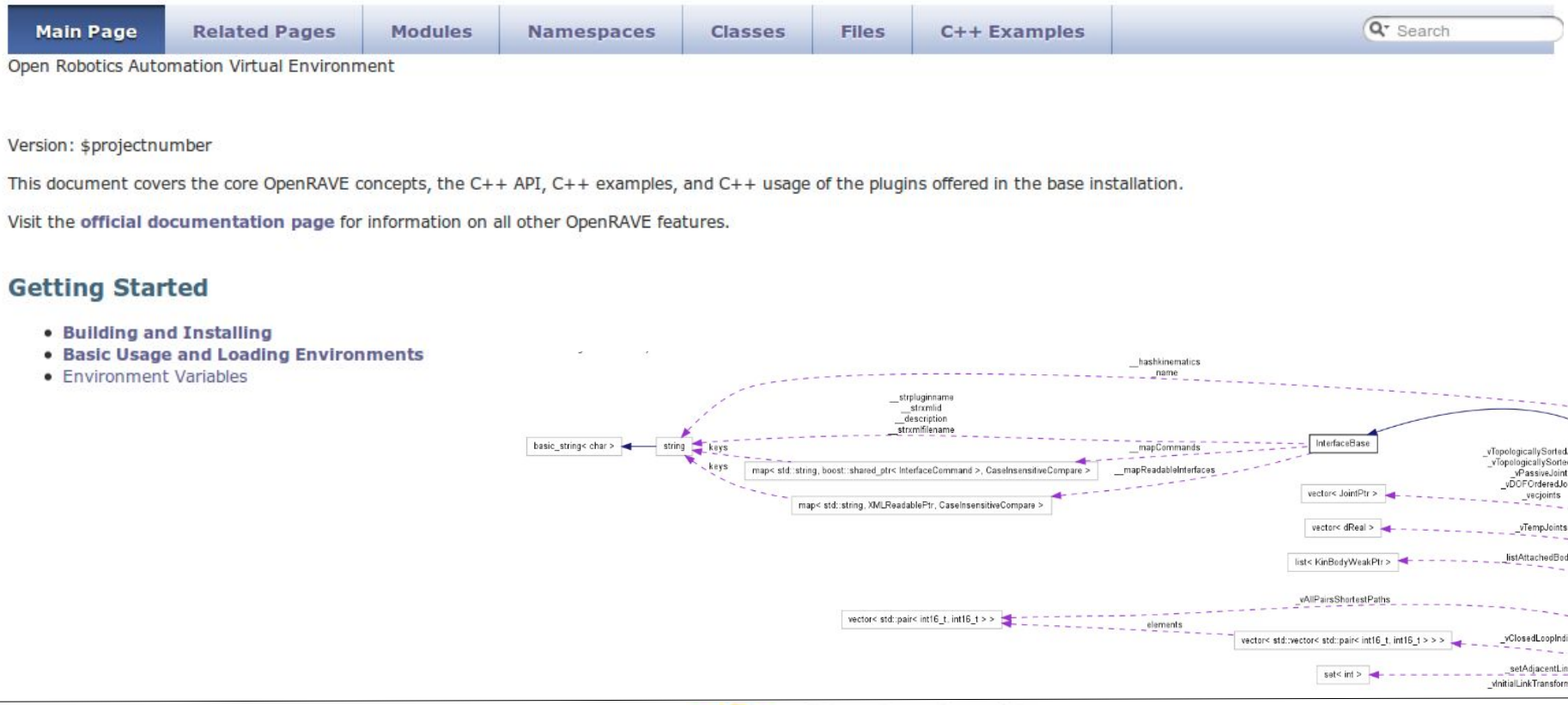
- 利点
  - 誰にでも更新出来るのでコミュニティーが現れる
  - 情報が整理しやすい
    - コンテンツ・ページの作成、検索ツール
  - 世界のニーズとともに変化出来る
- 欠点
  - コードとの同期が失われる
  - Wikiが大きくなって古い情報を削除のが難しい
  - 全データを保存しにくい
    - WikiはDBで管理されているからログインが必要

ドキュメント方法：コード挿入

- コードの中にコメントとして挿入し、ツールでHTML、PDF、LATEX、XML等を生成する
  - ツール：Sphinx、Doxygen
- 利点
  - 管理しやすい
  - コードとの同期がとれている
  - コードの変更時に側のドキュメントも更新する
  - ドキュメントされていないものに警告が出る
- ツールによって出力の柔軟性が変わる
  - Sphinxの柔軟性がWikiレベル以上

Doxygen：言語・ツール

- C++ドキュメント作成・管理
- 継承と依存関係図
- ソースコードからのリンク作成



Doxygen関数例

```
/** \brief Sets the velocity of the base link and each of the joints.

    Computes internally what the correponding velocities of each of the links should be in order to
    achieve consistent results with the joint velocities. Sends the velocities to the physics engine.
    Velocities correspond to the link's coordinate system origin.
    \param[in] linearvel linear velocity of base link
    \param[in] angularvel angular velocity rotation_axis*theta_dot
    \param[in] vDOFVelocities - velocities of each of the degrees of freeom
    \param checklimits if true, will excplicitly check the joint velocity limits before setting the values.
*/
virtual bool SetDOFVelocities(const std::vector<dReal>& vDOFVelocities, const Vector& linearvel,
                              const Vector& angularvel,bool checklimits = false);
```

bool SetDOFVelocities (const std::vector< dReal > & vDOFVelocities,
const Vector & linearvel,
const Vector & angularvel,
bool checklimits = false
) [virtual]

Sets the velocity of the base link and each of the joints.

Computes internally what the correponding velocities of each of the links should be in order to achieve consistent results with the joint velocities. Sends the velocities to the physics engine. Velocities correspond to the link's coordinate system origin.

Parameters:

[in] linearvel	linear velocity of base link
[in] angularvel	angular velocity rotation_axis*theta_dot
[in] vDOFVelocities	- velocities of each of the degrees of freeom checklimits if true, will excplicitly check the joint velocity limits before setting the values.

Definition at line **2194** of file **KinBody.cpp**.

Doxygenの多言語サポート

```
/** \en \brief Computes the minimal chain of joints that are between two links in the order of linkindex1 to linkindex2

    Passive joints are also used in the computation of the chain and can be returned.
    Note that a passive joint has a joint index and dof index of -1.
    \param[in] linkindex1 the link index to start the search
    \param[in] linkindex2 the link index where the search ends
    \param[out] vjoints the joints to fill that describe the chain
    \return true if the two links are connected (vjoints will be filled), false if the links are separate

    \ja \brief 2つのリンクを繋ぐ関節の最短経路を計算する.

    受動的な関節は，位置関係が固定されているリンクを見つけるために調べられている
    受動的な関節も返される可能があるから，注意する必要があります.
    \param[in] linkindex1 始点リンクインデックス
    \param[in] linkindex2 終点リンクインデックス
    \param[out] vjoints  関節の経路
    \return 経路が存在している場合，trueを返す.
*/
virtual bool GetChain(int linkindex1, int linkindex2, std::vector<JointPtr>& vjoints) const;
```

bool GetChain (int linkindex1,
int linkindex2,
std::vector< JointPtr > & vjoints
) const [virtual]

2つのリンクを繋ぐ関節の最短経路を計算する.

受動的な関節は，位置関係が固定されているリンクを見つけるために調べられている．受動的な関節も返される可能があるから，注意する必要があります.

引数:

[in] linkindex1 始点リンクインデックス
[in] linkindex2 終点リンクインデックス
[out] vjoints 関節の経路

戻り値:

経路が存在している場合，trueを返す.

KinBody.cpp の 2774 行で定義されています。

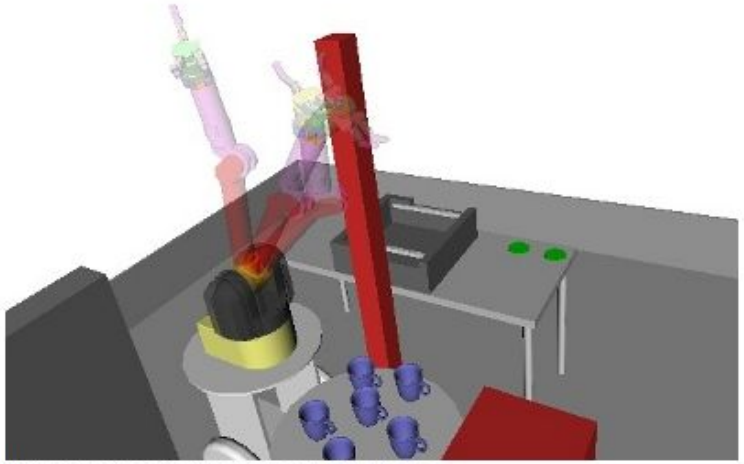
Doxygenサンプル管理

Main Page | Related Pages | Modules | Namespaces | Classes | Files | Examples

Search

ortrajectory.cpp

Author:
Rosen Diankov



Robot moving in random configurations.

Shows how to send a cubicaly interpolated trajectory to the robot controller. The actual trajectory consists of two points: the current configuration and the target configuration.

```
TrajectoryBasePtr traj = penv->CreateTrajectory(probot->GetDOF());
probot->GetDOFValues(q); // get current values
traj->AddPoint(TrajectoryBase::TPOINT(q,probot->GetTransform(),0.0f));
q[RaveRandomInt()%probot->GetDOF()] += RaveRandomFloat()-0.5; // move a random axis
traj->AddPoint(TrajectoryBase::TPOINT(q,probot->GetTransform(),2.0f));
traj->CalcTrajTiming(probot,TrajectoryBase::CUBIC,false,false); // initialize the trajectory structures
```

The demo also adds a collision check at the target point to make sure robot is going to a collision free configuration.

```
{
    RobotBase::RobotStateSaver saver(probot); // add a state saver so robot is not moved permenantly
    probot->SetDOFValues(q);
    if( penv->CheckCollision(RobotBaseConstPtr(probot)) ) {
        continue; // robot in collision at final point, so reject
    }
}
```

Python Bindingへの移転

- Breathe: http://michaeljones.github.com/breathe/
  - DoxygenのドキュメントをXMLでPythonに移転出来る
- Pythonで入力すれば

```
import openravepy
help openravepy.KinBody.SetDOFVelocities
```

- PythonのヘルプとしてDoxygenが出ます

```
Help on method SetDOFVelocities:

SetDOFVelocities(...) unbound openravepy.openravepy_int.KinBody method
    SetDOFVelocities( (KinBody)arg1, (object)dofvelocities) -> bool :


        bool  **SetDOFVelocities**\(const std::vector<  dReal  > & vDOFVelocities, bool checklimits = false )

            Sets the velocity of the joints.

            *Parameters*
             ``vDOFVelocity`` -
              - velocities of each of the degrees of freeom  checklimits if true, will excplicitly check the jo
int velocity limits before setting the values. Copies the current velocity of the base link and calls SetDOFVel
ocities(linearvel,angularvel,vDOFVelocities)


    SetDOFVelocities( (KinBody)arg1, (object)dofvelocities, (object)linear, (object)angular) -> bool :


        bool  **SetDOFVelocities**\(const std::vector<  dReal  > & vDOFVelocities, const  Vector  & linearvel,
const  Vector  & angularvel, bool checklimits = false )
```

reStructuredText：言語
Sphinx：コンパイラー役

- Pythonの標準ドキュメント言語
  - http://docutils.sourceforge.net/rst.html
- Doxygenと一緒にコードに挿入出来ます
- Doxygenより柔軟性がはるかに高い
- Sphinxで非常に綺麗なドキュメントが作成可能

例：OpenRAVE　http://openrave.org
　　matplotlib http://matplotlib.sourceforge.net/

Sphinx関数例

```
@docstring.dedent_interpd
def plot(self, *args, **kwargs):
    """
    Plot lines and/or markers to the
    :class:`~matplotlib.axes.Axes`.  *args* is a variable length
    argument, allowing for multiple *x*, *y* pairs with an
    optional format string.  For example, each of the following is
    legal::

        plot(x, y)        # plot x and y using default line style and color
        plot(x, y, 'bo')  # plot x and y using blue circle markers
        plot(y)           # plot y using x as index array 0..N-1
        plot(y, 'r+')     # ditto, but with red plusses

    If *x* and/or *y* is 2-dimensional, then the corresponding columns
    will be plotted.

    An arbitrary number of *x*, *y*, *fmt* groups can be
    specified, as in::

        a.plot(x1, y1, 'g^', x2, y2, 'g-')

    Return value is a list of lines that were added.

    The following format string characters are accepted to control
    the line style or marker:

    ===============    ============================
     character          description
    ===============    ============================
     ``'-'``            solid line style
     ``'--'``           dashed line style
```

matplotlib.pyplot.**plot**(**args*, ***kwargs*)

Plot lines and/or markers to the **Axes**. *args* is a vari
with an optional format string. For example, each

```
plot(x, y)          # plot x and y using default l
plot(x, y, 'bo')    # plot x and y using blue circl
plot(y)             # plot y using x as index arra
plot(y, 'r+')       # ditto, but with red plusses
```

If *x* and/or *y* is 2-dimensional, then the correspond

An arbitrary number of *x*, *y*, *fmt* groups can be spe

```
a.plot(x1, y1, 'g^', x2, y2, 'g-')
```

Return value is a list of lines that were added.

The following format string characters are accepted

character	description
'-'	solid line style
'--'	dashed line style

実行例からギャラリーの自動作成

OpenRAVE search

home | examples | robots | plugins | database generators | wiki | contents » modules | index

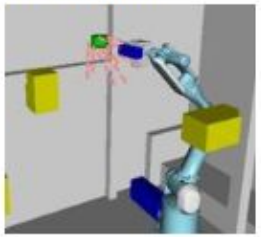
Examples

Getting Started Tutorials

C++ Examples Page

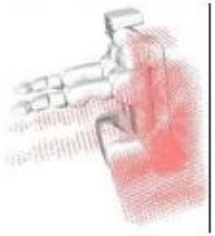
Python Examples

calibrationviews



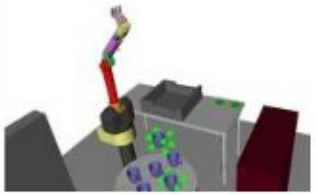
Calibrates a camera attached on a robot by moving it around a pattern.

checkconvexdecomposition



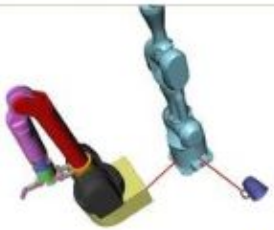
Builds the convex decomposition of the robot and plots all points inside its volume.

checkvisibility



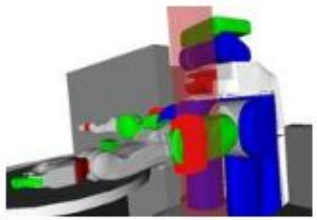
Computes the visibilty extents of a camera and an object.

collision



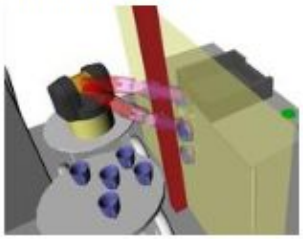
Check collision calls, use collision reports, and do distance queries.

collision2



Plot collision contacts.

constraintplanning



Shows how to use simple gradient-based jacobians to constrain the motion of the robot while planning.

Questions? Suggestions?

Join the openrave-users mailing list

your@email

your name

Digest: No Subscribe

Trac: Report bugs/request features

Reference

Core C++ API

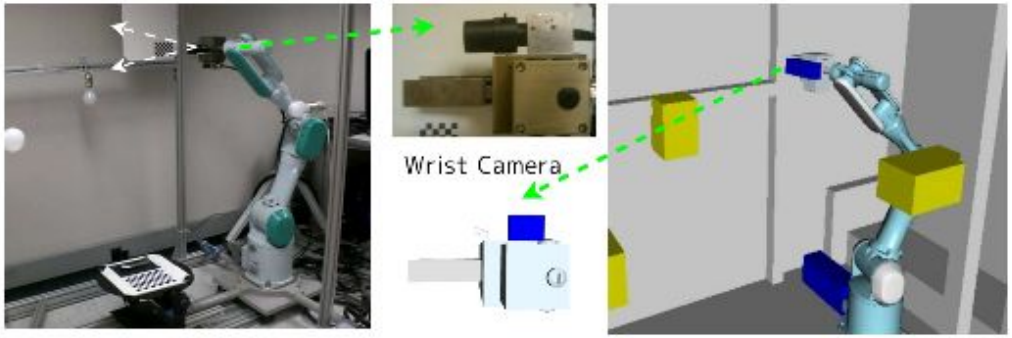
Python API

Developers Guide

実行例ページ

calibrationviews Module

Calibrates a camera attached on a robot by moving it around a pattern.



Running the Example:

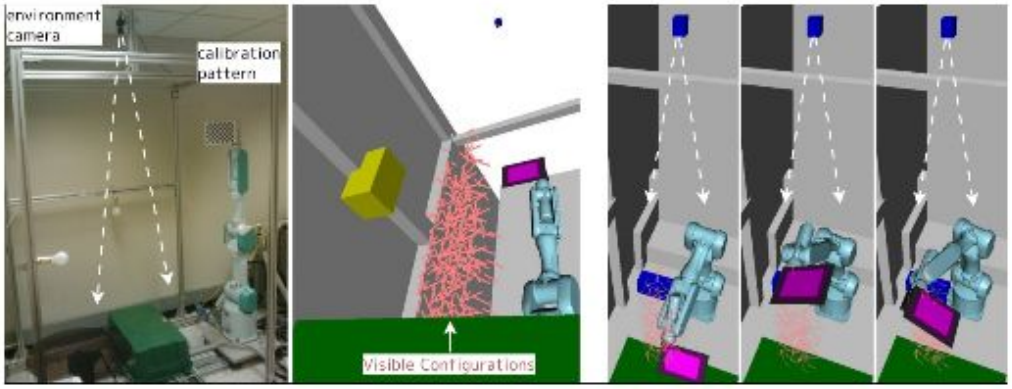
```
openrave.py --example calibrationviews
```

Description

The pattern is attached to the robot gripper and robot uses moves it to gather data. Uses visibilitymodel to determine which robot configurations make the pattern fully visible inside the camera view.

It is also possible to calibrate an environment camera with this exapmle using:

```
openrave.py --example calibrationviews --scene=data/pa10calib_envcamera.env.xml --sensorrobot=ceiling
```



Command-line

```
Usage: openrave.py [options]

Views a calibration pattern from multiple locations.

Options:
  -h, --help            show this help message and exit
  --scene=SCENE         Scene file to load (default=data/pa10calib.env.xml)
  --sensorname=SENSORNAME
                        Name of the sensor whose views to generate (default is
                        first sensor on robot)
  --sensorrobot=SENSORROBOT
                        Name of the robot the sensor is attached to
                        (default=none)
  --norandomize         If set, will not randomize the bodies and robot
                        position in the scene.
  --novisibility        If set, will not perform any visibility searching.
  --posedist=POSEDIST   An average distance between gathered poses. The
                        smaller the value, the more poses robot will gather
                        close to each other

  OpenRAVE Environment Options:
    --loadplugin=_LOADPLUGINS
                        List all plugins and the interfaces they provide.
    --collision=_COLLISION
                        Default collision checker to use
    --physics=_PHYSICS  physics engine to use (default=none)
    --viewer=_VIEWER    viewer to use (default=qtcoin)
    --server=_SERVER    server to use (default=None).
    --serverport=_SERVERPORT
                        port to load server on (default=4765).
    -l _LEVEL, --level=_LEVEL
                        Debug level, one of
                        (fatal,error,warn,info,debug,verbose)
```

Main Code

```
def main(env,options):
    "Main example code."
    env.Load(options.scene)
    robot = env.GetRobots()[0]
    sensorrobot = None if options.sensorrobot is None else env.GetRobot(options.sensorrobot)
    env.UpdatePublishedBodies()
    time.sleep(0.1) # give time for environment to update
    self = CalibrationViews(robot,sensorname=options.sensorname,sensorrobot=sensorrobot,randomize=options.randomize

    attachedsensor = self.vmodel.attachedsensor
    if attachedsensor.GetSensor() is not None and attachedsensor.GetSensor().Supports(Sensor.Type.Camera):
        attachedsensor.GetSensor().Configure(Sensor.ConfigureCommand.PowerOn)
        attachedsensor.GetSensor().Configure(Sensor.ConfigureCommand.RenderDataOn)

    while True:
        print 'computing all locations, might take more than a minute...'
        self.computeAndMoveToObservations(usevisibility=options.usevisibility,posedist=options.posedist)
```

Class Definitions

class openravepy.examples.calibrationviews.**CalibrationViews**(*robot*, *sensorname=None*, *sensorrobot=None*, *target=None*, *maxvelmult=None*, *randomize=False*)[source]

computeAndMoveToObservations(*waitcond=None*, *maxobservations=inf*, *posedist=0.050000000000000003*, *usevisibility=True*, ***kwargs*)[source]

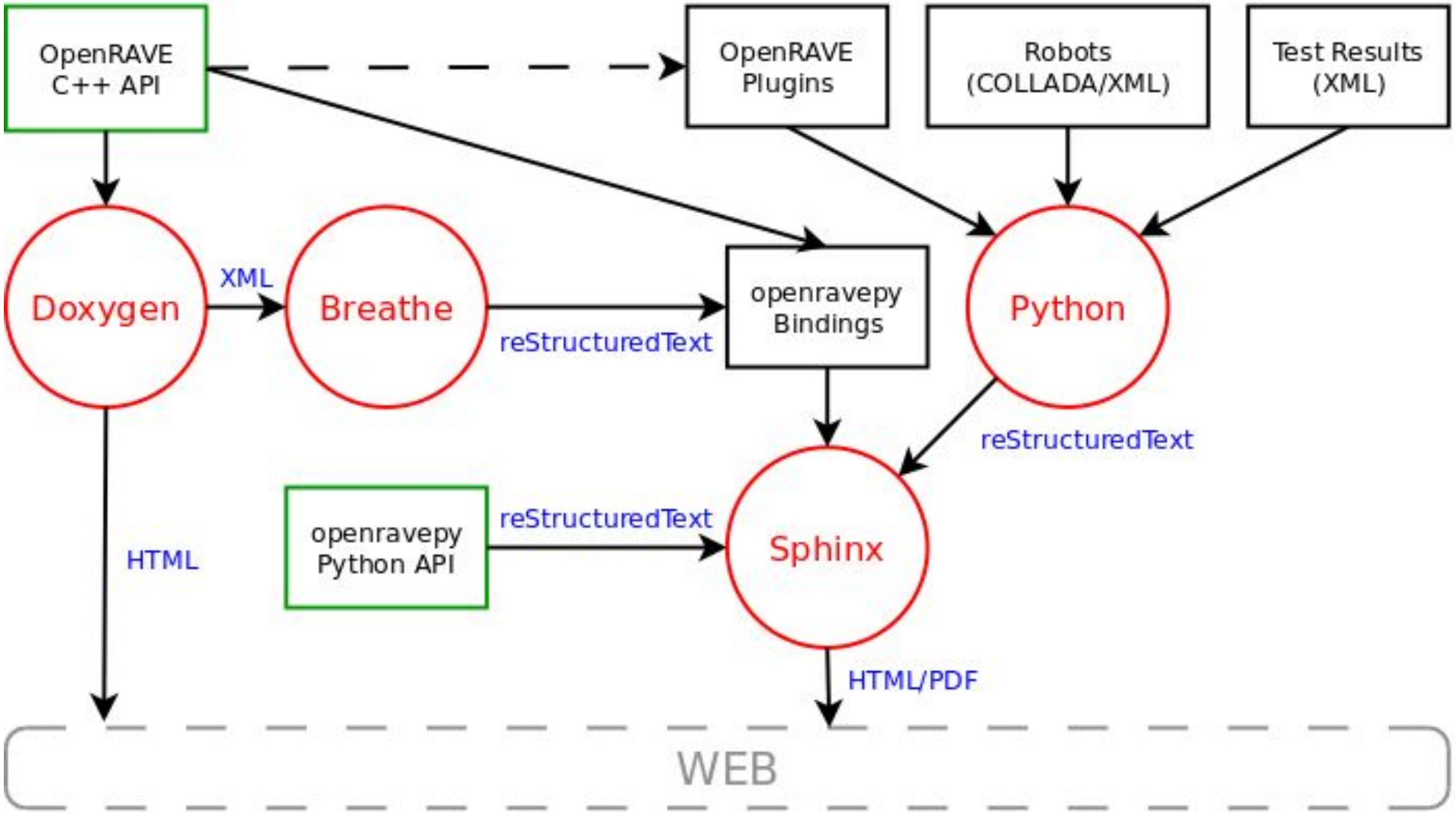
Computes several configuration for the robot to move. If usevisibility is True, will use the visibility model of the pattern to gather data. Otherwise, given that the pattern is currently detected in the camera, move the robot around the local neighborhood. This does not rely on the visibiliy information of the pattern and does not create a pattern

computeLocalPoses(*maxcosangle=0.5*, *maxposedist=0.14000000000000000*

実行例のreStructuredText

```
Calibrates a camera attached on a robot by moving it around a pattern.

.. examplepre-block:: calibrationviews

Description
-----------

The pattern is attached to the robot gripper and robot uses moves it to gather data.
Uses :mod:`.visibilitymodel` to determine which robot configurations make the pattern fully
visible inside the camera view.

It is also possible to calibrate an environment camera with this exapmle using:

.. code-block:: bash

  openrave.py --example calibrationviews --scene=data/pa10calib_envcamera.env.xml --
sensorrobot=ceilingcamera

.. image:: ../../images/examples/calibrationviews_envcamera.jpg
  :width: 640

Calibration
-----------

Although this example does not contain calibration code, the frames of reference are the
following:

.. image:: ../../images/examples/calibrationviews_frames.jpg
  :width: 640


**T_pattern^world** and **T_camera^link** are unknown, while **T_pattern^camera** and
**T_link^world** are known.

.. examplepost-block:: calibrationviews
```

OpenRAVEドキュメント構造



ドキュメントシステム立ち上げのお勧め

- reStructuredTextで本も論文も書く
- コード挿入法で
  - コアのAPIとチュートリアルは
  - 本とシステム論文の出力
- Wiki法で
  - 他のシステムとの連携
  - 第三者のプロジェクト事例
  - ニュース

例：OpenRAVE http://openrave.org

リリースシステム

リリースシステム概要

- ユーザにソースからビルドすると問題が起こる
- リリースパッケージの理想
  - 依存関係の解決
  - コンポーネント化
    - ユーザによって必要なものが異なる
  - 一つのコマンドで作成とアップロードを行う
  - ターゲットOSに対応する
  - ビルドシステムでインストールの設定を行える
- Ubuntu, Windows, RPMパッケージの作成
  - CMake

CMake/CPackでパッケージ作成

- CPackを使用し、OS対応の作成が出来る
  - cpack -G NSIS
  - cpack -G DEB
  - cpack -G TGZ
- CMakeLists.txt
  - set(CPACK_XXX)で設定を行う
  - コンポーネント化
    - install(FILES myfile DESTINATION bin COMPONENT base)
  - 依存関係
    - set(CPACK_DEBIAN_PACKAGE_DEPENDS libboost-dev)

Debianパッケージ：Ubuntu用

https://wiki.ubuntu.com/PackagingGuide/Complete

- Launchpadビルドファームの利用
  - ソースコードのSSH転送で自動的にパッケージがコンパイルされ、世界に使用出来るようになります

```
sudo add-apt-repository ppa:openrave/release
sudo apt-get update
sudo apt-get install openrave
```

- Debianソースパッケージ、バイナリーパッケージ
- debianフォルダーで設定を行う。主なファイル
  - control → パッケージの依存関係・一般情報
    - ビルドの依存関係
  - rules →　ビルドのプロセス
    - cmakeの実行
    - dpkg-*プログラムでシンボル・依存管理

CMakeからDebianパッケージの生成

- DebSourcePPA.cmakeを使用する
  - https://openrave.svn.sourceforge.net/svnroot/openrave/trunk/modules-cmake/DebSourcePPA.cmake

```
# CMake settings for Debian package installation
set(CPACK_COMPONENTS_ALL fitting-base fitting-devel)
install(TARGETS fitting DESTINATION lib COMPONENT fitting-base)
install(TARGETS fitting_node DESTINATION bin COMPONENT fitting-base)
install(FILES includes/fitting.h DESTINATION include/opencv_fitting COMPONENT fitting-devel)

set(CPACK_DEBIAN_PACKAGE_NAME opencv_fitting)
set(CPACK_DEBIAN_DISTRIBUTION_RELEASES karmic lucid maverick natty)
set(CPACK_PACKAGE_DESCRIPTION_SUMMARY "my opencv fitting package")
set(CPACK_PACKAGE_VERSION "0.1")
set(CPACK_PACKAGE_INSTALL_DIRECTORY "opencv_fitting")
set(DPUT_HOST "ppa:user/package")

set(CPACK_DEBIAN_DISTRIBUTION_NAME ubuntu)
set(CPACK_DEBIAN_BUILD_DEPENDS_UBUNTU debhelper cmake
                                      ros-diamondback-common-msgs
                                      ros-diamondback-vision-opencv)
set(CPACK_DEBIAN_PACKAGE_DEPENDS ${CPACK_COMPONENTS_ALL})
set(CPACK_DEBIAN_PACKAGE_PRIORITY optional)
set(CPACK_DEBIAN_PACKAGE_SECTION devel)
set(CPACK_DEBIAN_PACKAGE_SOURCE_COPY svn export --force)

set(CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake-modules")
include(CPack)
include(DebSourcePPA)
```

Windowsパッケージ

- Nullsoft Scriptable Install System
  - http://nsis.sourceforge.net/
- Windows Registryの登録・調査
- 依存プログラムをネットからダウンロード
- 環境変数の設定

```
outFile "myinstaller.exe"
# インストール先
InstallDir "$PROGRAMFILES\\MyProgram"

# インストールのコンポーネント
section
  setOutPath $INSTDIR
  file test.txt
  writeUninstaller $INSTDIR\uninstaller.exe
sectionEnd

# アンインストール設定
section "Uninstall"
delete $INSTDIR\uninstaller.exe
delete $INSTDIR\test.txt
sectionEnd
```

ROSパッケージ

http://www.ros.org/wiki/release/Setup

- ROSの依存関係の管理が違う
  - 機能単位はROSパッケージ
  - インストール単位はROSスタック
- WillowGarageのサーバーへリリースする

```
rosrun release create.py stack_name stack_version distro_name
```

  - ビルドファーム
    - http://build.willowgarage.com/
  - Ubuntuパッケージが出来上がり、ROSの正式なレポジトリーに入る
- 注意：Ubuntu/Launchpadに収束しつつある

リリースシステム立ち上げのお勧め

- CMakeで純粋なUbuntu/Windowsパッケージ作成
- ROSを使用しているシステム
  - コアの部分：ROSに依存しないのがベスト
    - 正式リリースし
  - 拡張機能：ROSとの連携
    - ROSスタックとしてWillowGarageにリリース
- 頻繁なリリース
  - テストシステムの速い段階の立ち上げが必須

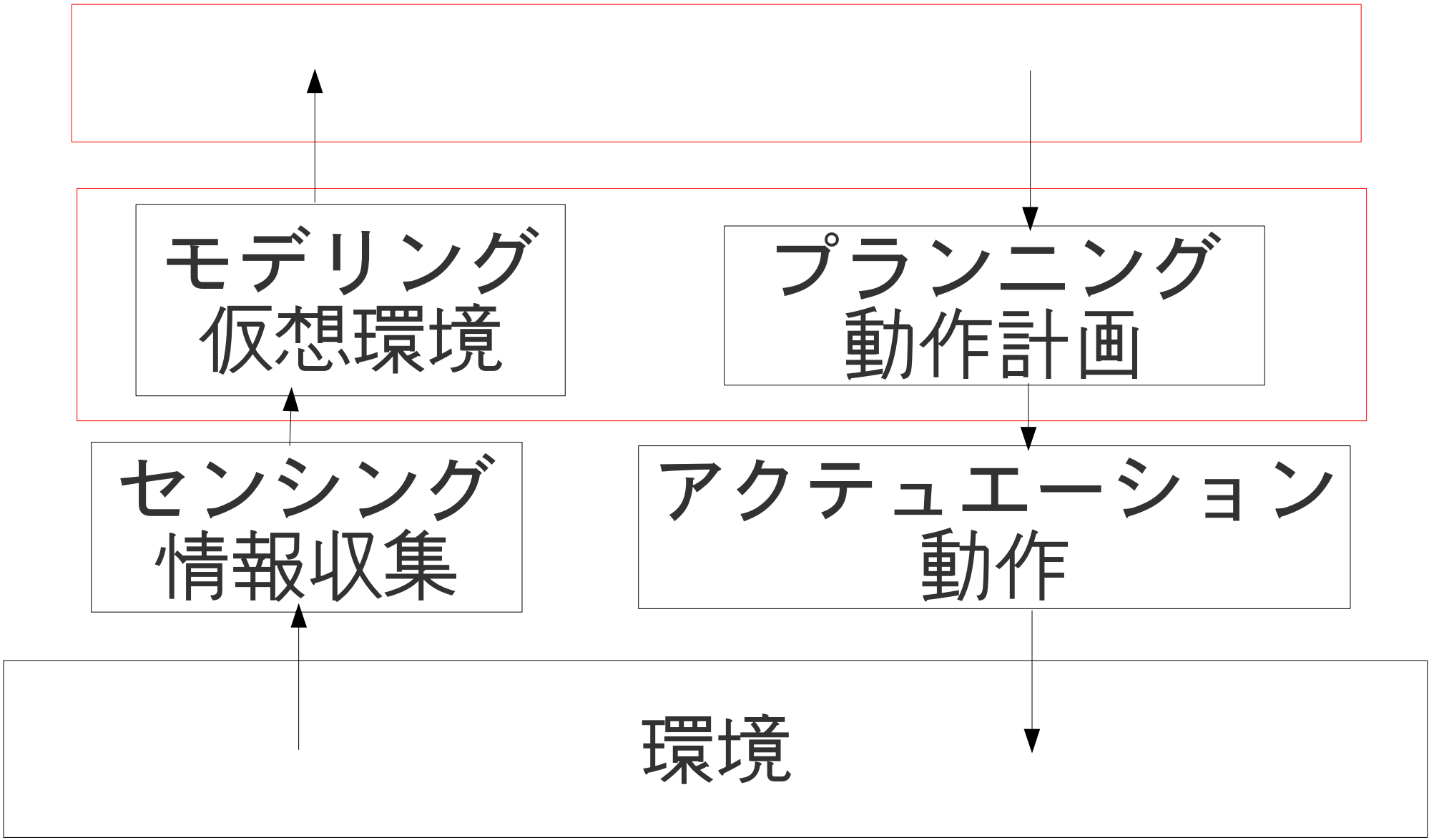
開発の管理

- sourceforge, googlecode, launchpad
- オープンなソース管理ツール
  - subversion, git, mercurial, bazaar
- オープンなプロジェクト管理ツール
  - trac, redmine

エージェントシステムの実現化

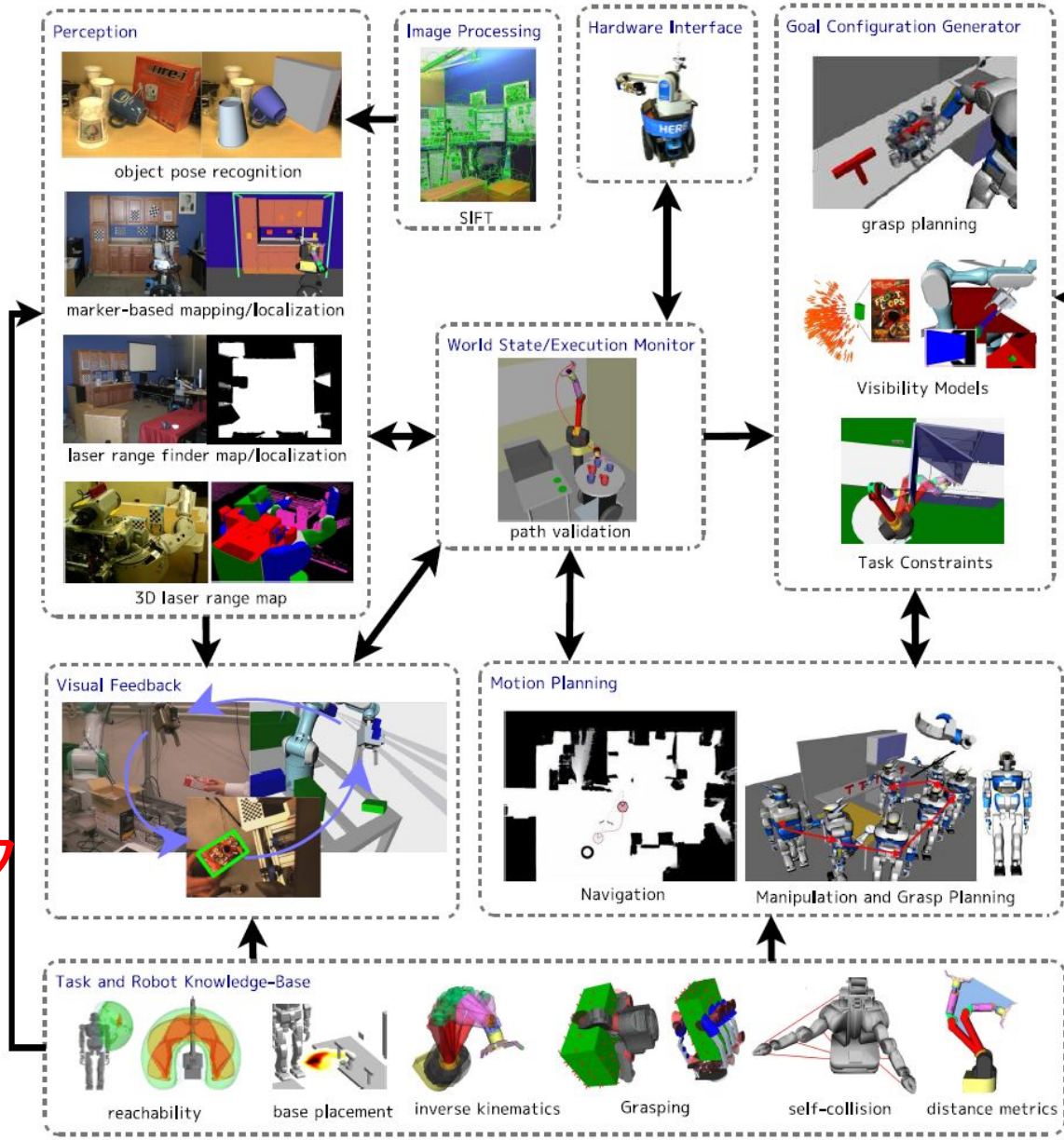
ロボットシステム

エージェントシステムの基本像



ロボットシステムの基本像

認識：
対象物体
障害物



目標状態
生成

制御・センサ
フィードバック

動作計画

知識ベース

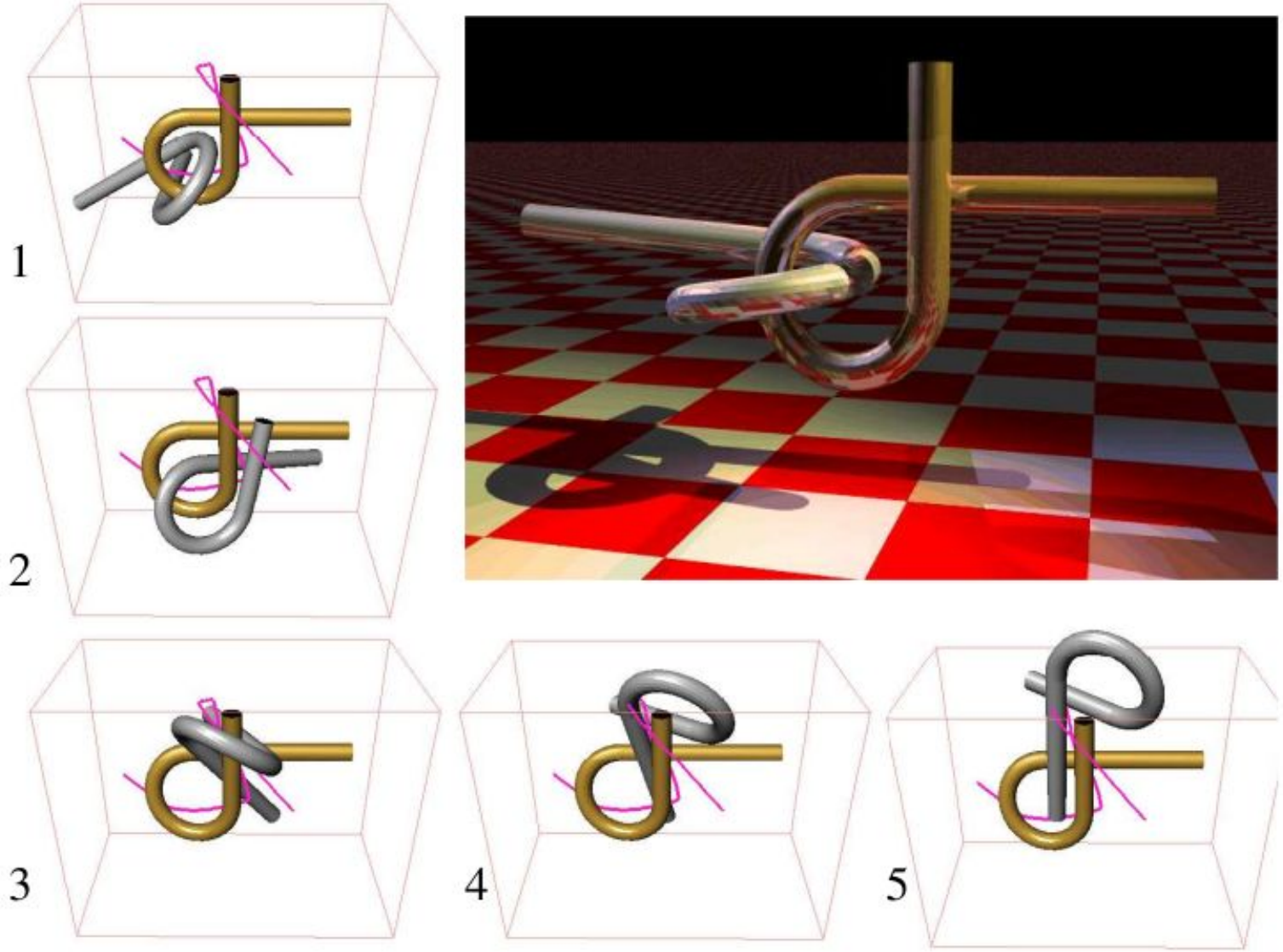
動作計画アルゴリズム

images:
Steven LaValle. Planning Algorithms. Cambridge University, 2006

動作計画は？

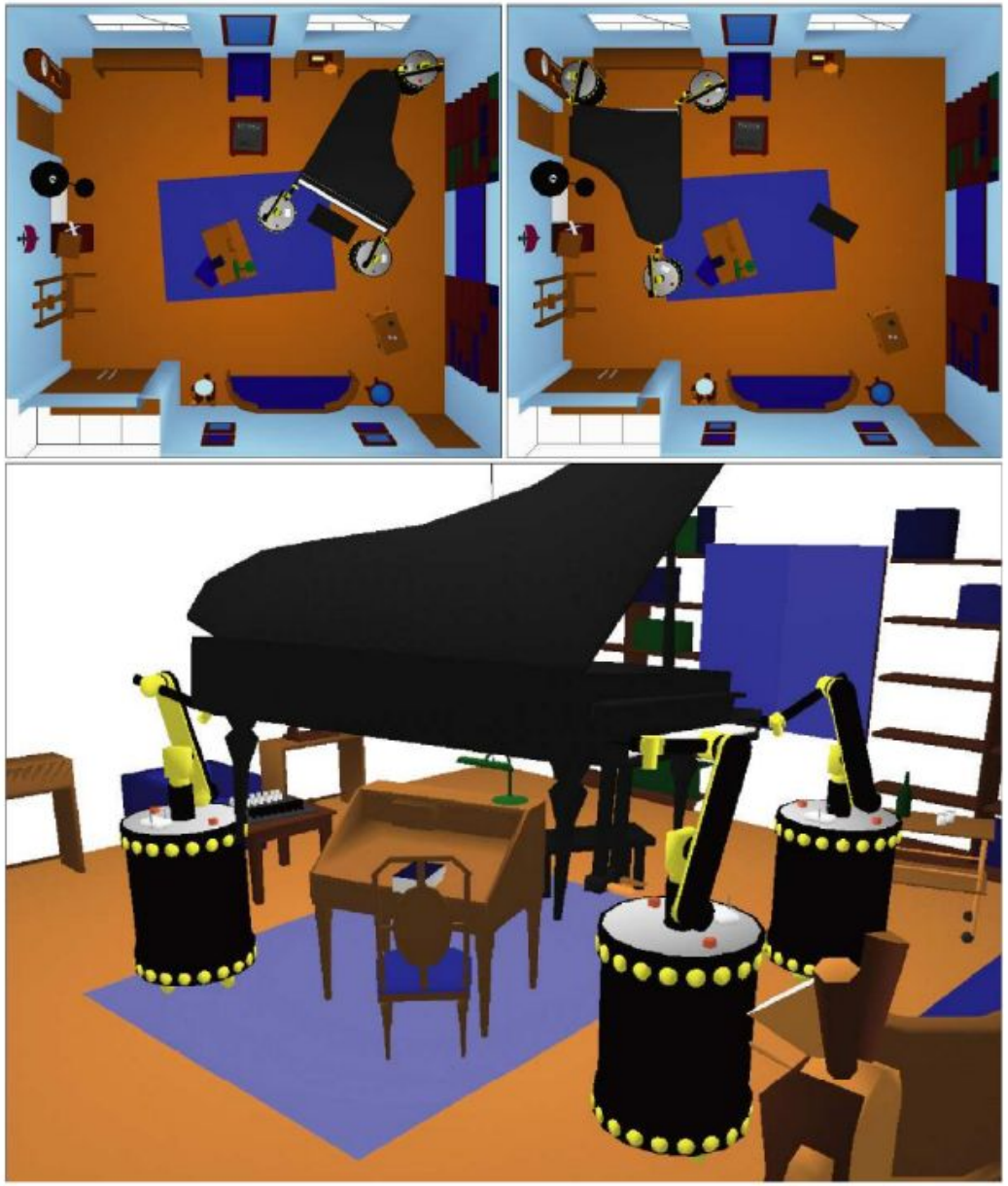
- 目的を果たすために動作単位で計画する
- 動作単位
  - 関節・モーター角度
  - 速度
  - 力応用

衝突せずに引き離す経路



物運び

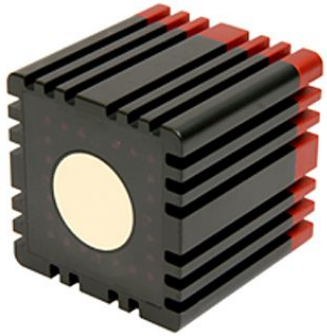
- ロボットによって運ばれる物
  - ロボットの制約条件
  - ３D障害物
- 状態
  - 位置姿勢
  - 関節の値



Maxon Motor

10万円以上

Swiss Ranger SR4000



～５０万円

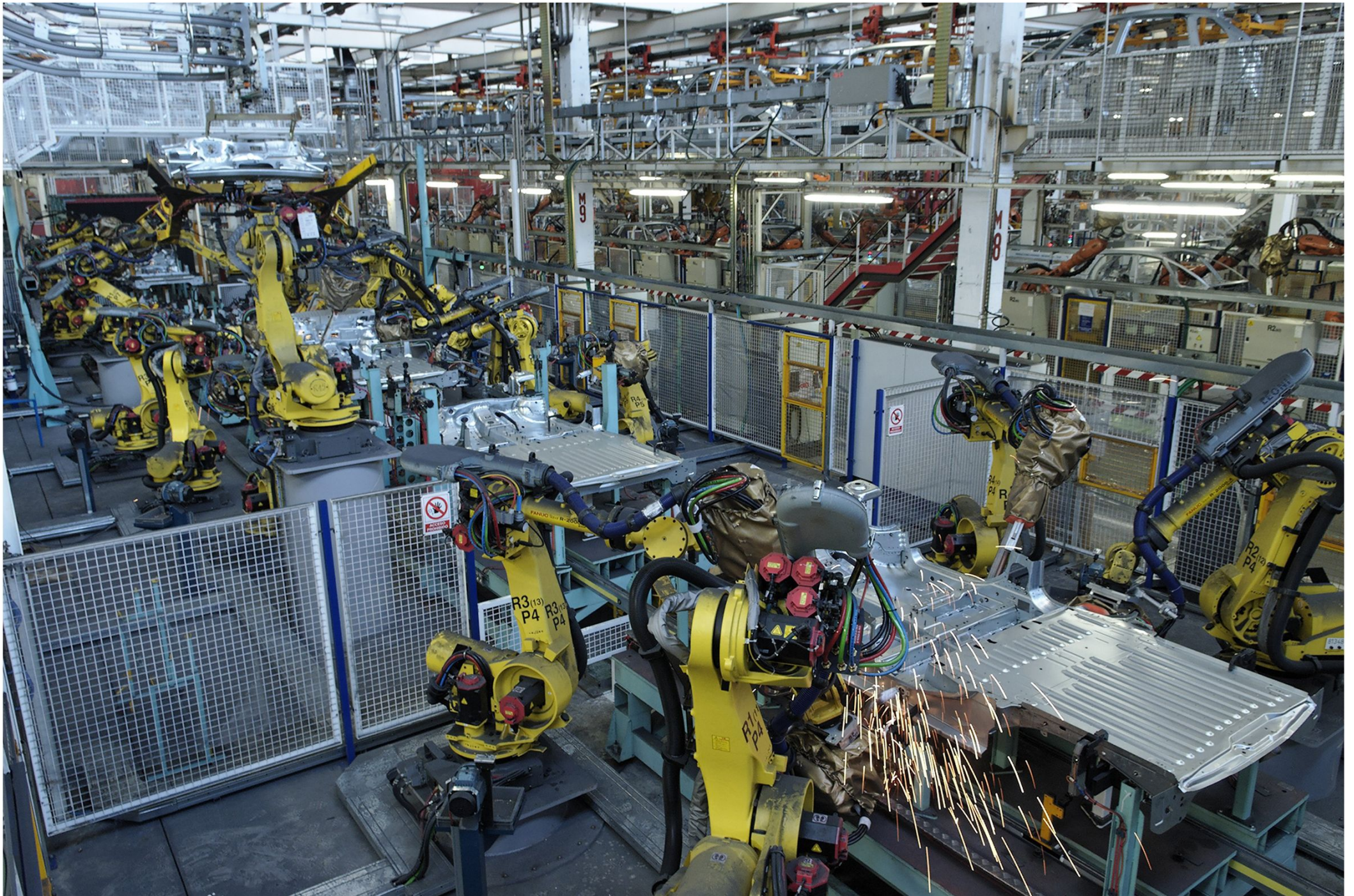
製造コスト

量産化の
ギャップ

試行錯誤
職人の技
検証テスト

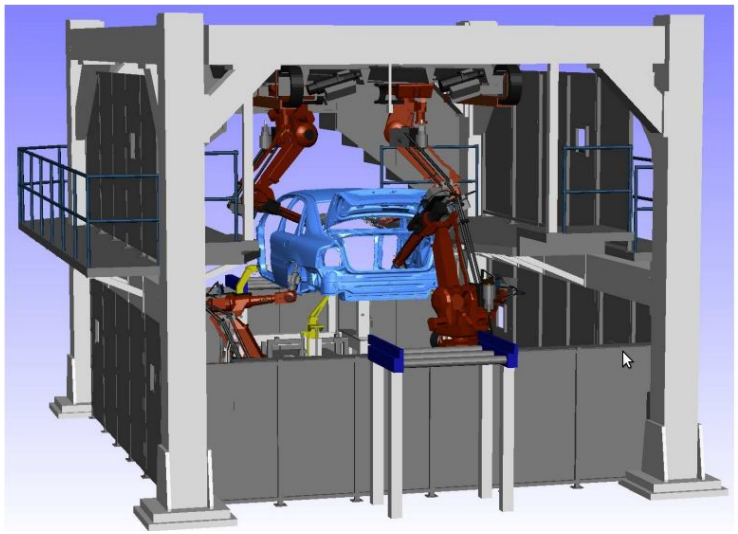
５０万円以下

動作計画の現実：1台～５０万円以下

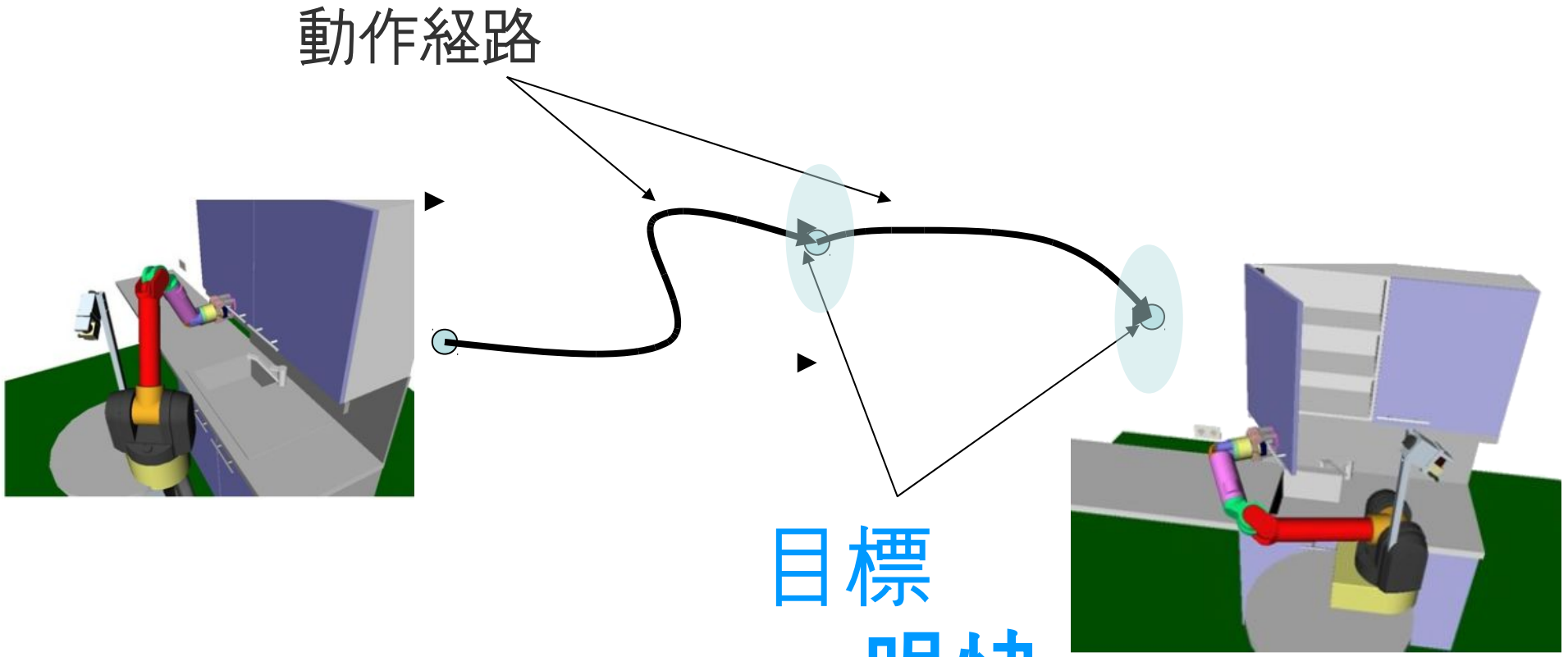


産業用ロボット

- 動作計画
  - 切削
  - 溶接点
  - 組み立て
  - 検査
- 設計



計画の二つの段階



動作計画 → 経路探索

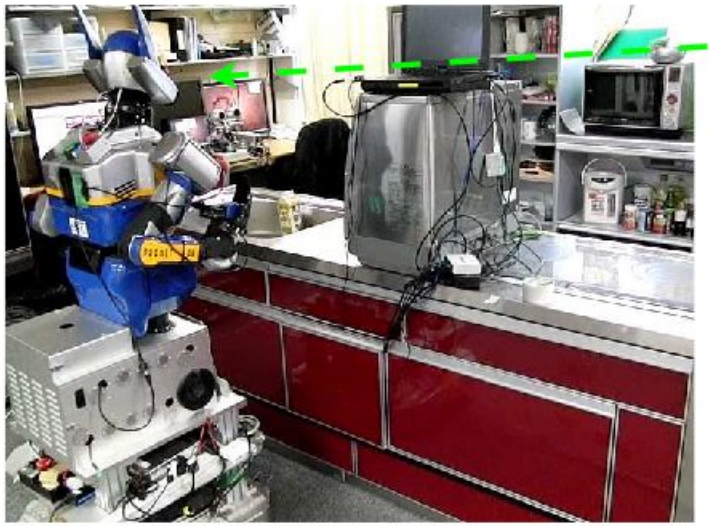
A*

ランダムサンプリング

Rapidly-exploring Random Tree (RRT)

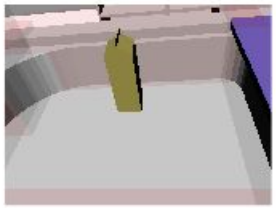
移動マニピュレーション例：記述

対象物体

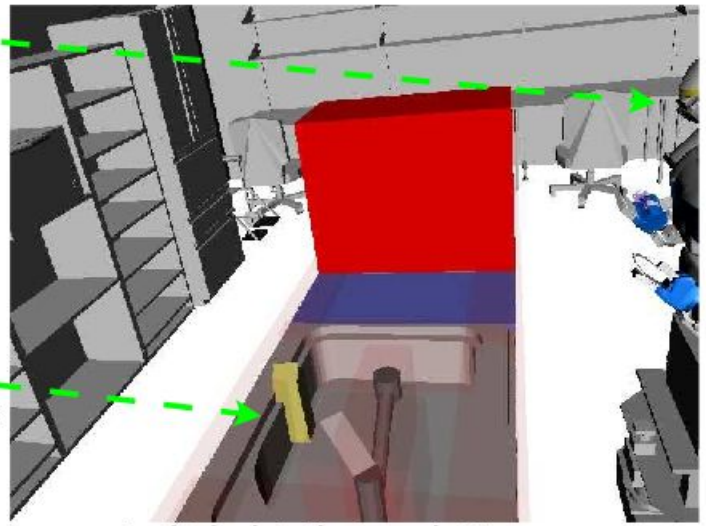


Initial Real Scene

Head Camera

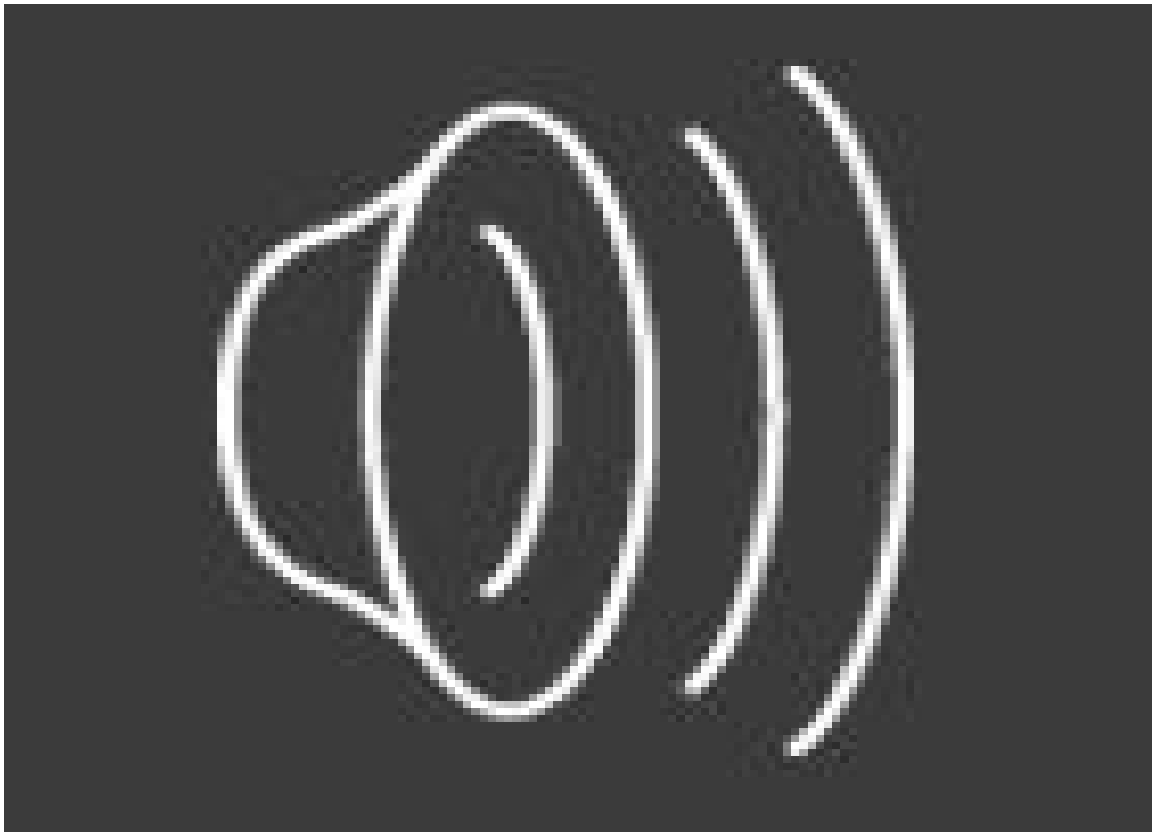


Half of Object is
Colliding with Sink

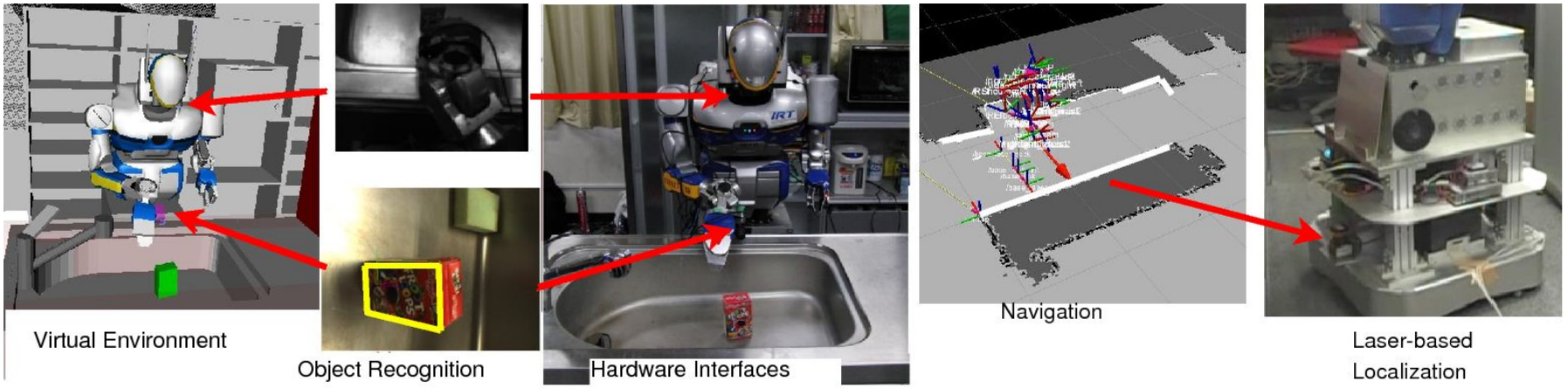


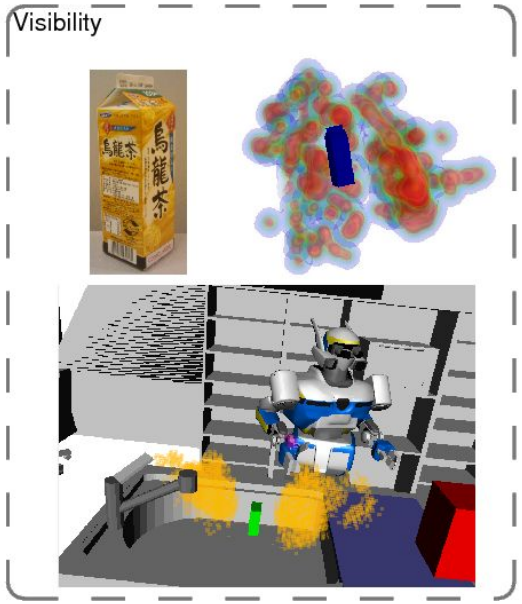
Initial Virtual Scene

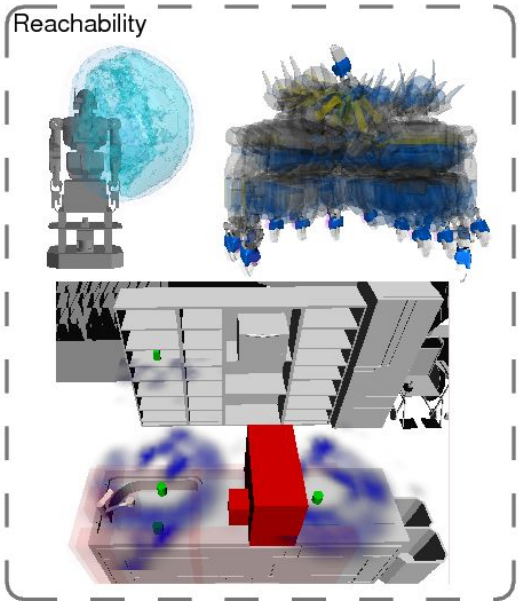
移動マニピュレーション：実行

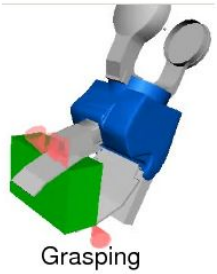


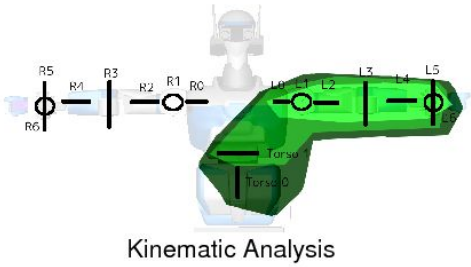
移動マニピュレーション：要素技術

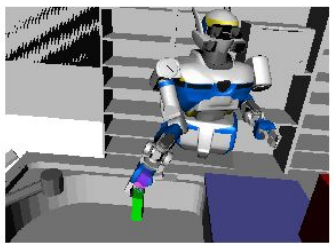


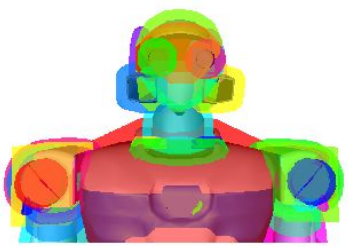




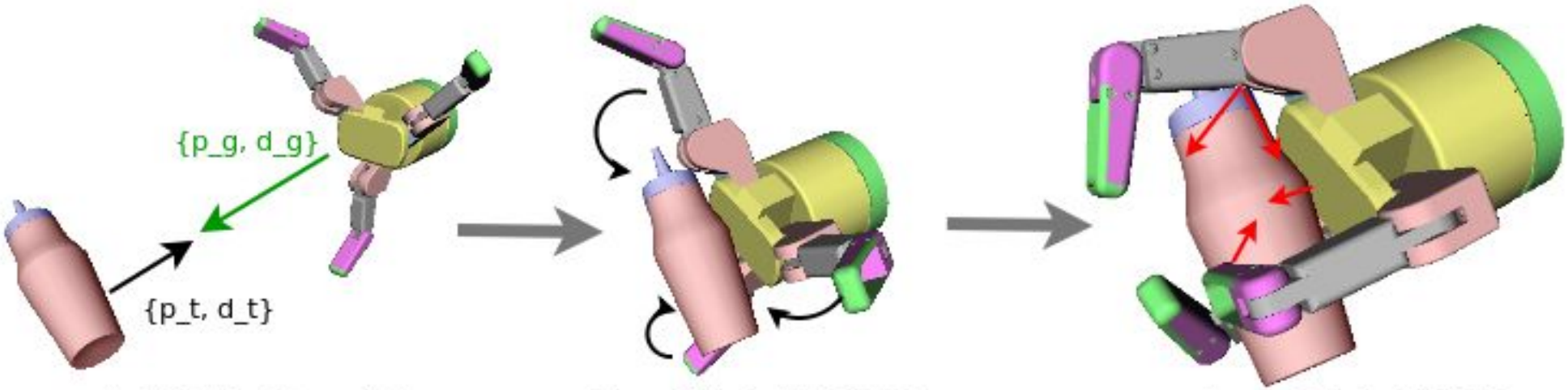








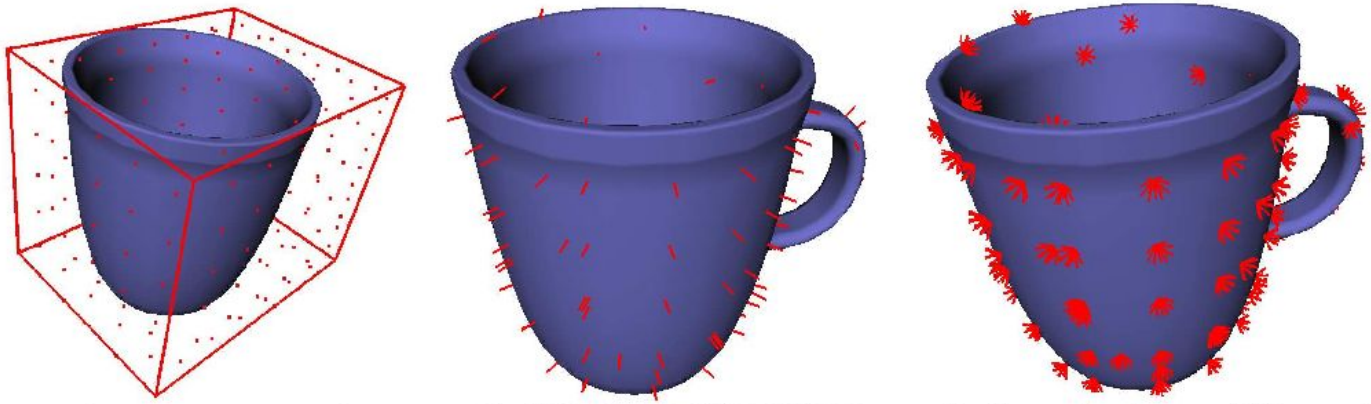
安定な把持



1．対象物体に迫る　2．指を閉じる　3．安定性を計算

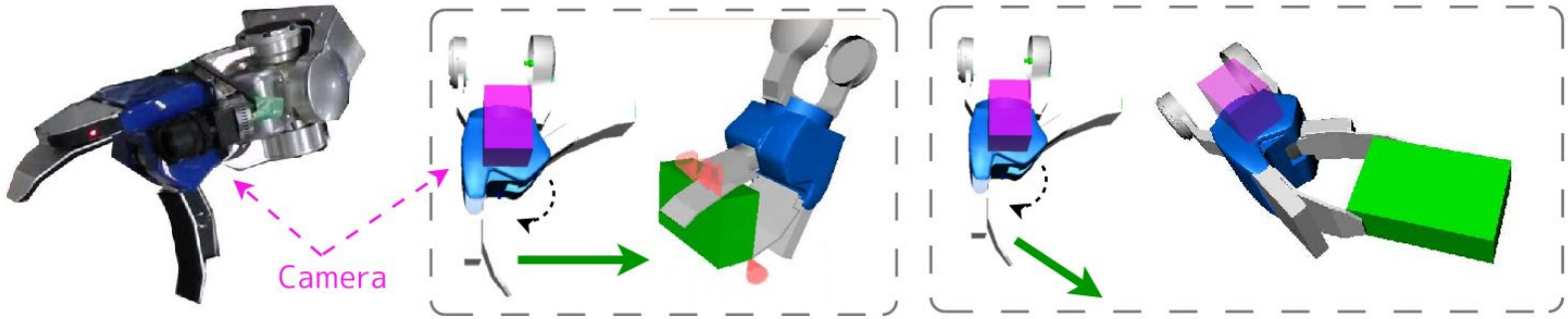
把持空間のパラメター化：８自由度

物体表面上
サンプリング
（4DOF）



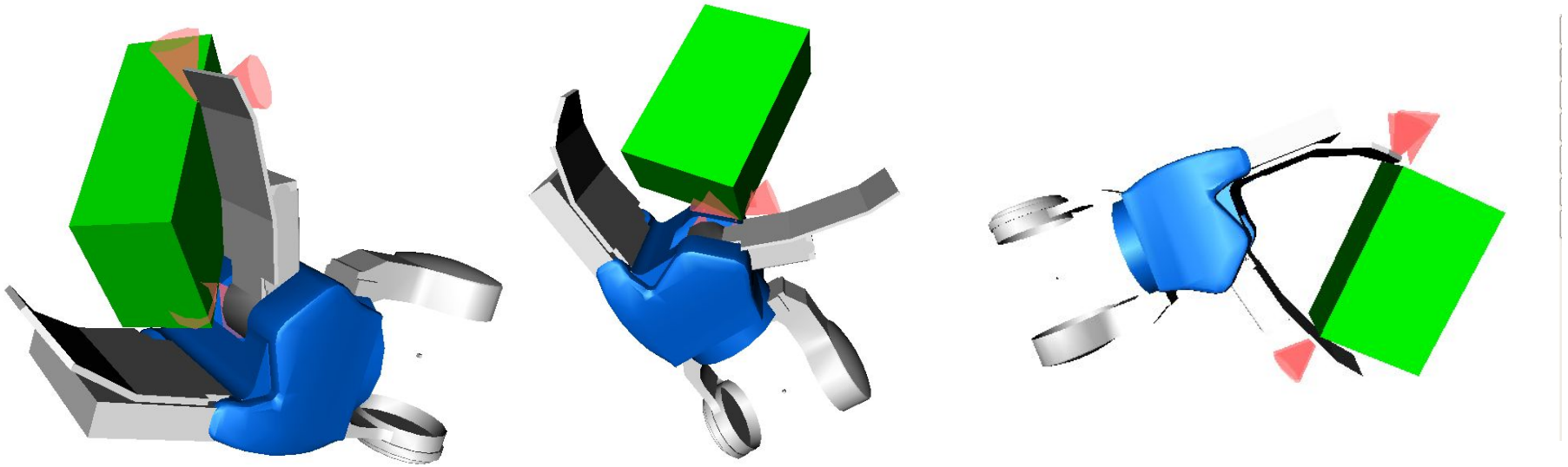
ロボット
表面上

（4DOF）



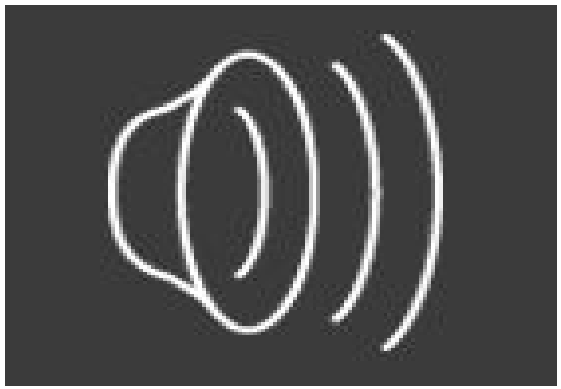
Stand-off（1DOF）

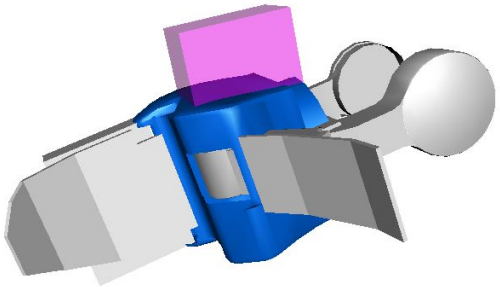
もろい把持の検出

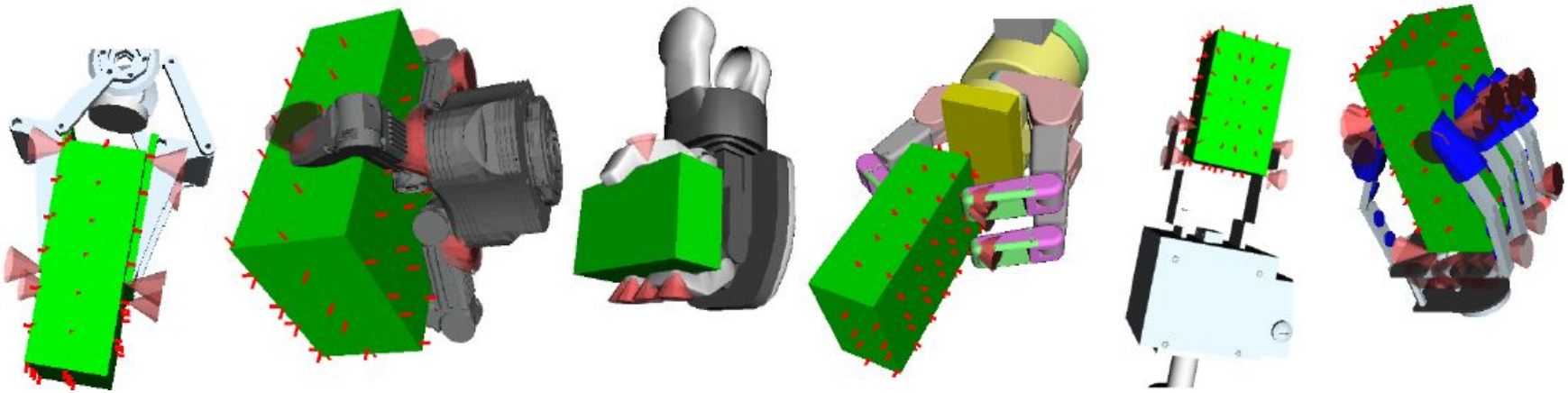


- 繰り返しの計算
  - ノイズを加え、統計を計算
  - 姿勢の分布で判定する

把持セット







アナウンス

- 第３宿題の締切り：22日23:59時
  - テスト：半分の点数
- 授業のビデオ録画の解像度公開？