Evolutionary Inverse Kinematics for ROS and MoveIt

Philipp Ruppel

May 30, 2017

Master Thesis

ROS, MoveIt

Robot Operating System

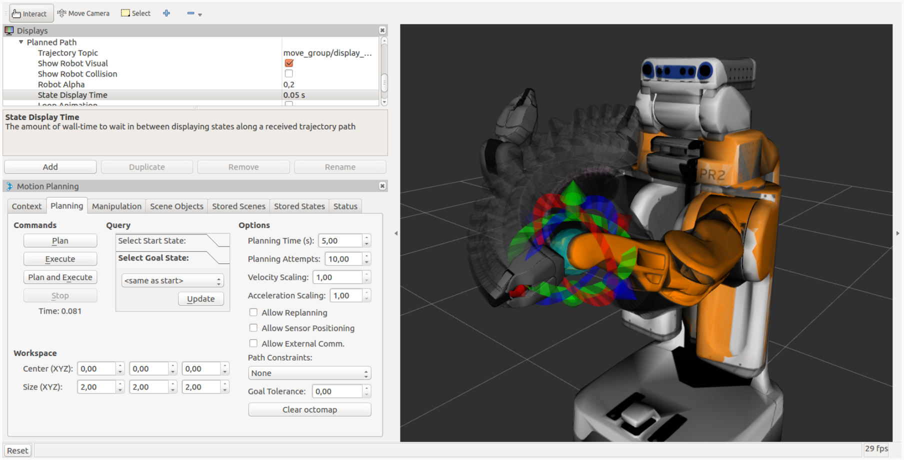
Motion planning framework

- Forward & inverse kinematics
- Perception
- Motion planning

MoveIt

video

MoveIt



Planned Path

Trajectory Topic	move_group/display_...
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	0,2
State Display Time	0.05 s

State Display Time
The amount of wall-time to wait in between displaying states along a received trajectory path

Time: 0.081

Select Start State:
Select Goal State:
<same as start>

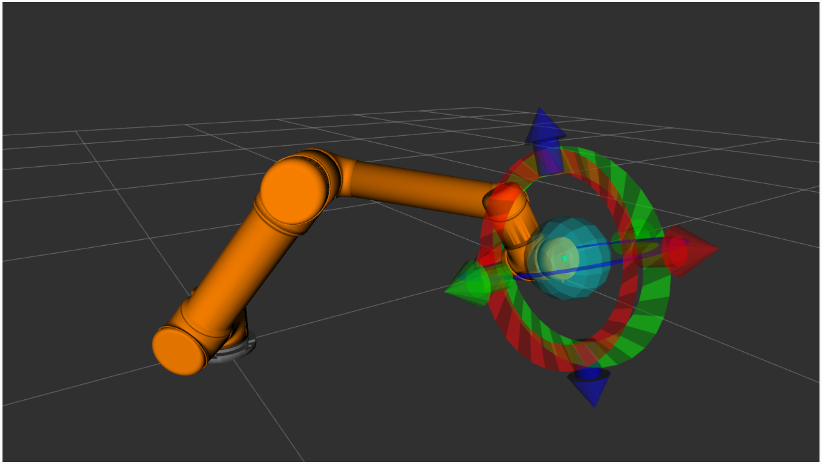
Planning Time (s): 5,00
Planning Attempts: 10,00
Velocity Scaling: 1,00
Acceleration Scaling: 1,00
☐ Allow Replanning
☐ Allow Sensor Positioning
☐ Allow External Comm.
Path Constraints: None
Goal Tolerance: 0,00

Workspace
Center (XYZ): 0,00 0,00 0,00
Size (XYZ): 2,00 2,00 2,00

29 fps

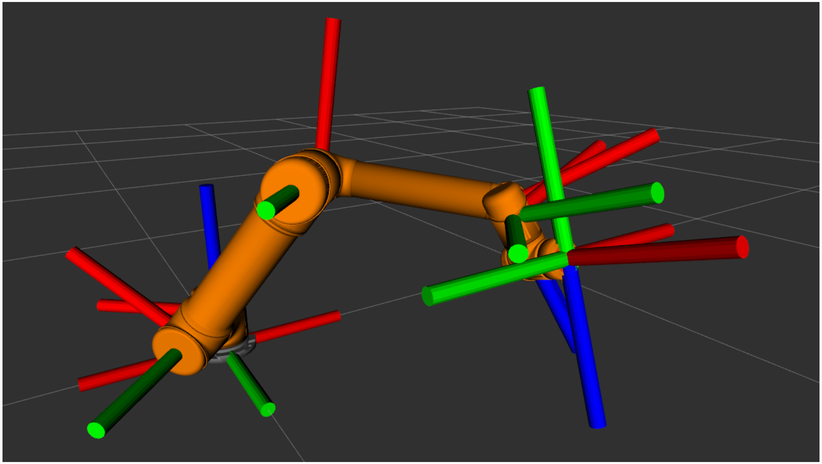
Inverse Kinematics

End effect pose $\rightarrow$ joint angles



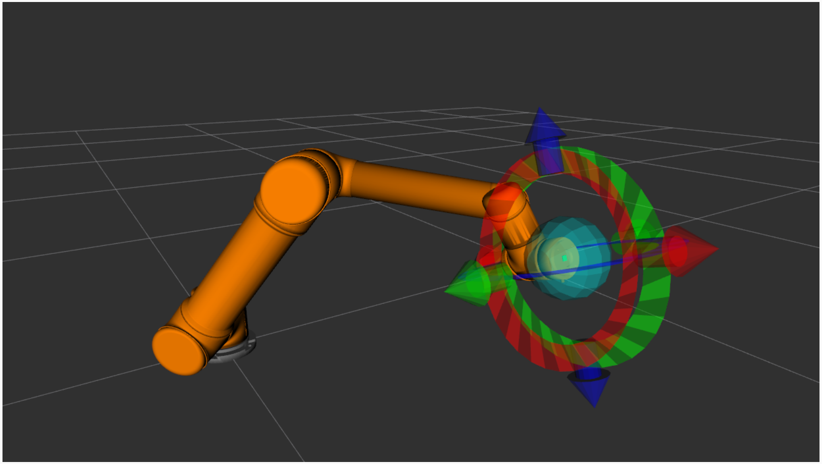
Forward Kinematics

Joint angles $\rightarrow$ end effect pose



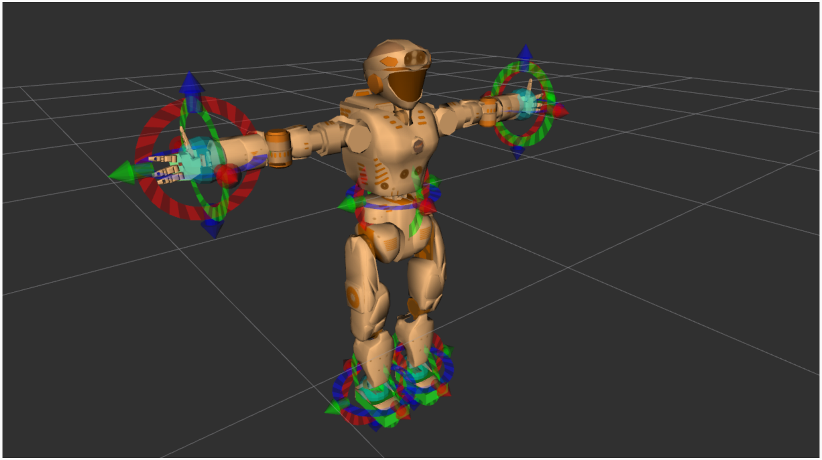
Inverse Kinematics

End effect pose $\rightarrow$ joint angles

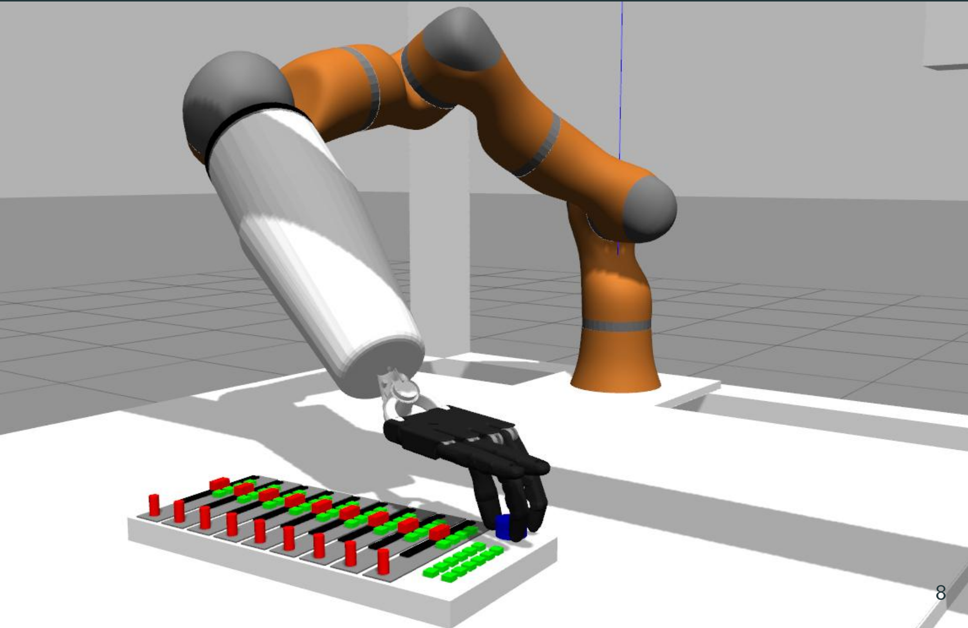


Inverse Kinematics

Multi-Goal IK (multiple eef, or generalized: arbitrary constraints on robot pose)



Inverse Kinematics



IK Solvers in MoveIt

- IK Plugin interface (kinematics::KinematicsBase)
- Analytic
    - Specialized solvers
    - ikfast
- Numeric
    - KDL
    - TRAC-IK
- Currently single-goal only!

Analytic Solution

- Manually solve equations
- Very fast at runtime
- Only practical for simple cases
- Solver is specific to a certain robot and goal configuration
- New robot or different goal types?
  - write new solver

ikfast

- Generates code automatically
- (Generated solver is specific to a certain robot and goal configuration)
- Only kinematic chains, only some goal types
- Long compilation time
- Doesn't always work

"for 6D ik, there are still mechanisms it cannot solve" (http://openrave.org/docs/0.8.2/openravepy/ikfast/)

6D = Translation (3D) + Orientation (3D)

KDL / Pseudo-Inverse Jacobian

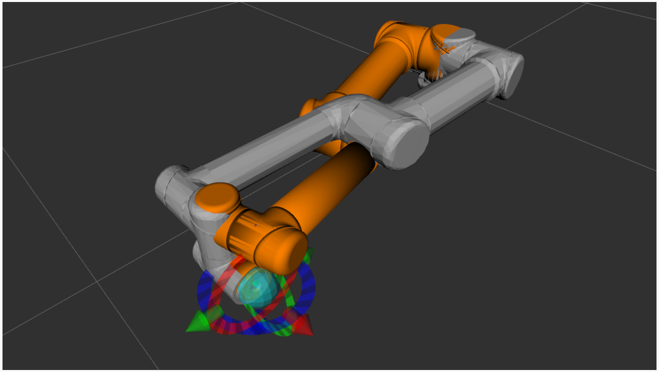
- Pseudo-inverse jacobian
- Minimize tip frame offset

$$\begin{array}{c} \\ j1 \\ j2 \\ j3 \\ \\ \end{array}\begin{array}{c} \begin{array}{cccc} \quad X \quad & \quad Y \quad & \quad Z \quad & \quad \end{array} \\ \begin{pmatrix} \delta x/\delta j_1 & \delta y/\delta j_1 & \delta z/\delta j_1 & \dots \\ \delta x/\delta j_2 & \delta y/\delta j_2 & \delta z/\delta j_2 & \dots \\ \delta x/\delta j_3 & \delta y/\delta j_3 & \delta z/\delta j_3 & \dots \\ \dots & \dots & \dots & \ddots \end{pmatrix} \end{array}$$

$$\Delta J = -\Delta P * J^{-1}$$

Pseudo-Inverse Jacobian - Issues

- Joint limits
- Possibly degenerate pseudo-inverse
- Local minima



TRAC-IK

- Random restarts if stuck to mitigate local minima
- In parallel: sequential quadratic programming
- Faster than KDL

Inverse Kinematics, MoveIt

Limitations of built-in IK solvers:

- Only single-goal
- Only pose goal or position goal
- Analytic or gradient-based
- Hard to extend

Extensibility & Flexibility:

- Arbitrary cost function
- Minimize cost function

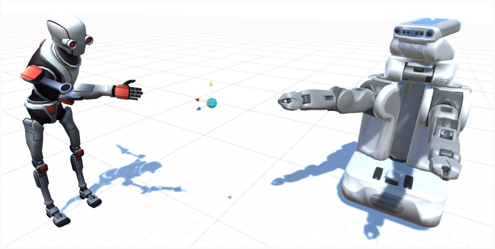
Cost Function

Cost Function

- Sum of errors (eg. square distances)
- Goals: Position, Orientation, Pose, LookAt, MaxDistance, MinDistance, Touch, Line, AvoidJointLimits, MinimalDisplacement, JointVariable, CenterJoints, JointFunction, Balance,

BioIK

BioIK, Sebastian



- Multi-goal, arbitrary cost function
- Biologically Inspired
- Hybrid: Genetic Algorithm
  $+$ Particle Swarm Optimization
- C#, Unity, Game Development

BioIK, Sebastian

Genetic algorithm

- Population, individuals, genes
- Genome = joint values
- Mutation, recombination, selection

Particle swarm optimization

- Momentum

BioIK for ROS - Requirements

- C++
- MoveIt
- High Accuracy (10^{-5}, $10\mu m$, $0.0018deg$)
- Fail if unreachable (optional, default)
- Performance (default timeout: 5ms)
- Compatibility: Joint Types, Mimic Joints, etc.

Modified Evolution

Genome and Mutation

Genome

- Genes = Joint positions (eg. joint angles)
- Only active variables
- **double**[] genes;

Mutation

- $r * 2^{-s}$

  r: random number gene

  s: random scale per individual, 2^{-s} fast bit-shift

  Small mutations most likely $\Rightarrow$ high accuracy
- Clipping at joint limits

Genome and Mutation

NOT used:

- **int** [] genes;
- Mutation: mutate random bits
- Hamming walls: single mutated bit = 2 to 3 (20 to 21), but not 7 to 8 (0111 to 1000)
- Gray code: conversion overhead
- Different mutation scales per gene
- Floating point numbers & exponential term in mutation function = similar distribution

Fitness & Selection

- IK goals = fitness function
- Compute fitness for each individual
- Select fittest individuals
- Genotype: active joint values (eg. angles)
- Phenotype: end effector link frames (and active joint values)
- Forward kinematics
  = genotype-phenotype mapping
  = most computation time
  $\rightarrow$ focus performance optimization on genotype phenotype mapping / forward kinematics

Secondary Objectives

- Primary and secondary objectives
- eg. primary: position goal, secondary: minimal displacement
- Pre-selection by secondary objectives
  Random number of survivers

Wipeouts

- Small mutations most likely
- Might temporarily be stuck at local optimum
- Wipeout if no improvement for a few generations
- Random wipeouts
- Wipeout = random reset of all genes
- Same for all individuals, to prevent premature convergence

Islands / Parallelization

- Multiple islands
- Independent evolution
- One thread per island

Multiple Species

- Problem: wipeout might accidentally destroy best solution
- Two competing species per island
- Only less fit species wiped out

Initialization

- Initial guess from MoveIt

Termination

- Timeout
- Good fitness
- Specialized pose comparison for compatibility

Particle Swarm Optimization

- Per-gene momentum

-
  ```
  struct Individual
  {
      aligned_vector<double> genes;
      aligned_vector<double> momentum;
  ```
- genes += momentum
- momentum randomly doubled or erased

Memetics

- Evolution + local search
- Tried solvers from CppNumericalSolvers - slower
- Quadratic optimization
- Approximate local fitness landscape by parabola along gradient
- Find extremum & update if better

- Evolution for *n* generations
- Quadratic optimization, up to *m* steps

Extrapolated Forward Kinematics

Forward Kinematics

Most computation time spent on computing forward kinematics

- Compute joint transforms
  - Trigonometry, sin/cos
- Concatenate frames
  - Quaternion or matrix operations

Forward Kinematics

MoveIt / KDL

- Rotation matrices (3x3 matrix, 3 basis vectors)
- Concatenation:
  27 multiplications, 18 additions

BioIK

- Quaternions (4 elements)
- Concatenation:
  16 multiplications, 12 additions

Forward Kinematics

Extrapolated Forward Kinematics

- Compute exact forward kinematics only every n steps
- Extrapolate forward kinematics for most mutations
- Rotation: linear, $x = a + b * dj$
- Position: quadratic, $x = a + b * dj + c * dj^2$

Extrapolated Forward Kinematics

Extrapolation

- No trigonometry
- Only active joints, no static or inactive joints, no link frames
- No expensive quaternion-quaternion or matrix-matrix multiplications
  - only addition and multiplication by scalar
- Simple control flow (single loop)
  - Low risk of branch misprediction
  - Pipelining

Extrapolated Forward Kinematics

- FK: analytic derivatives, similar to pseudo-inverse jacobian methods
- Also 2nd order / quadratic
- IK goals: evolution
- Arbitrary fitness function
- Less problems at joint limits & with local minima
- No explicit matrix inversions

SIMD

SIMD

- SIMD (Single Instruction Multiple Data)
- load/store/add/mul/sub/... multiple values at the same time
- Vectors
- SIMD registers, SIMD instructions
- SSE, AVX, FMA
- Aligned memory access
- Programming:
    - SIMD intrinsics (eg. _mm_add_pd(a, b))
    - auto-vectorization (eg. #pragma omp simd)
- Function multiversioning

SIMD Mutation

Mutation

- SIMD auto-vectorization

```
#pragma omp simd
    aligned(genes_min:32),
    aligned(genes_max:32),
    ...
for(size_t gene_index = 0; ...)
{
    ...
```

SIMD Forward kinematics

- Forward kinematics
- Quaternion operations: bad for SIMD - shuffle
- Can't be efficiently vectorized
- Instruction latencies

- - ```
    double r_x =
      (p_w * q_x + p_x * q_w) +
      (p_y * q_z - p_z * q_y);
    double r_y =
      (p_w * q_y - p_x * q_z) +
      (p_y * q_w + p_z * q_x);
    double r_z =
      (p_w * q_z + p_x * q_y) -
    ```
    ```

SIMD Forward kinematics

- Forward kinematics
- Extrapolation = ideal for SIMD

    -
      ```
      px += delta_pos.x() * delta_joint;
      py += delta_pos.y() * delta_joint;
      pz += delta_pos.z() * delta_joint;
      ```

    -
      ```
      __m256d p = _mm256_load_pd(
            tip_frame_ptr);
          [...]
      p = _mm256_fmadd_pd(ff,
          _mm256_load_pd(joint_delta_ptr), p);
          [...]
      _mm256_store_pd(tip_mutation_ptr, p);
      ```

Neural Networks

Neural Networks

- Library: FANN
  ("Fast Artificial Neural Network", C++)

- Input: goal
- Output: joint values
- ... inaccurate!

Neural Networks

- Learn relative movements instead!
- Input:
  - normalized relative goal pose (x,y,z, rx,ry,rz)
  - current joint values
- Output: joint offsets
- Multiple iterations
- Fully connected [n, 50, m] network

PR2 Arm	Neural Network	5ms	10^{-5}	12%
PR2 Arm	Neural Network	100ms	10^{-5}	29%
PR2 Arm	KDL Network	5ms	10^{-5}	55%

Experiments

PR2 KDL

video

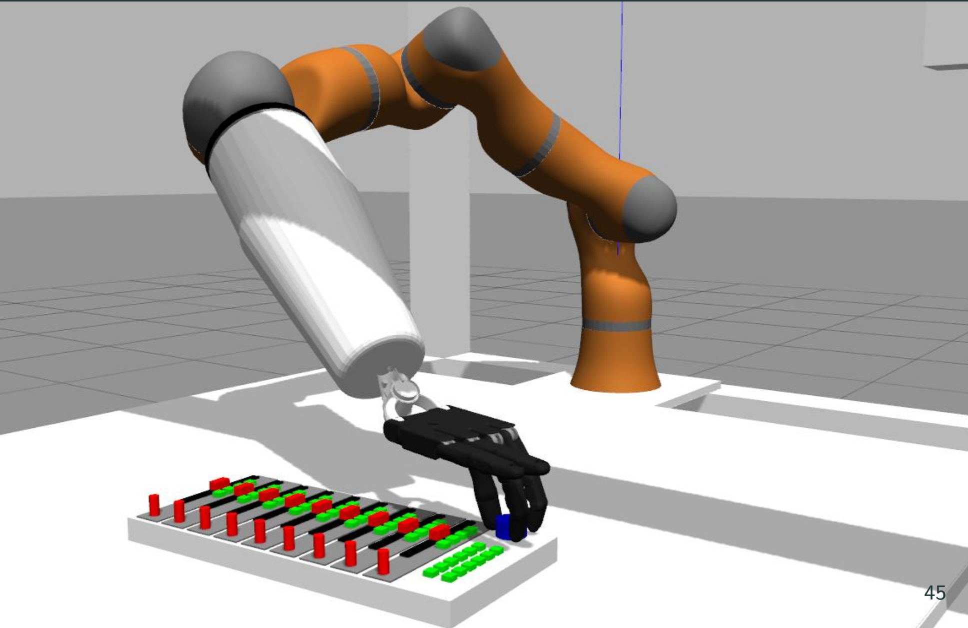
PR2 BioIK Minimal Displacement

video

PR2 BioIK Multi-Goal

video

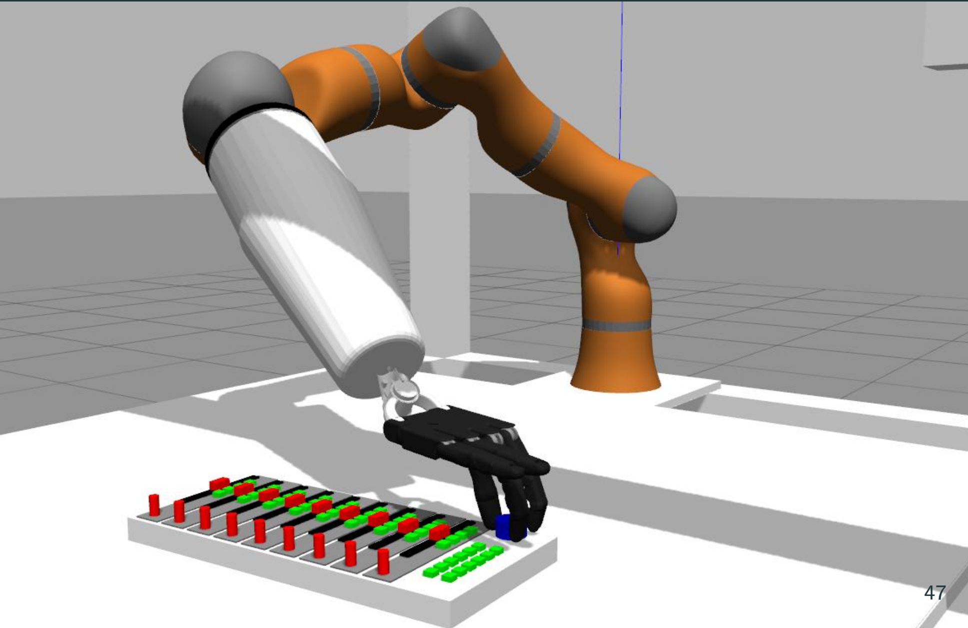
Shadow Hand



Shadow Hand BioIK

video

Shadow Hand Mixer Test



Shadow Hand Mixer Test

IK Goals

- Finger tips (Line, Touch, LookAt)
- Palm horizontal (LookAt)
- Coupled joint pair (J1, J2)
- MinimalDisplacement (secondary), CenterJoints (secondary)

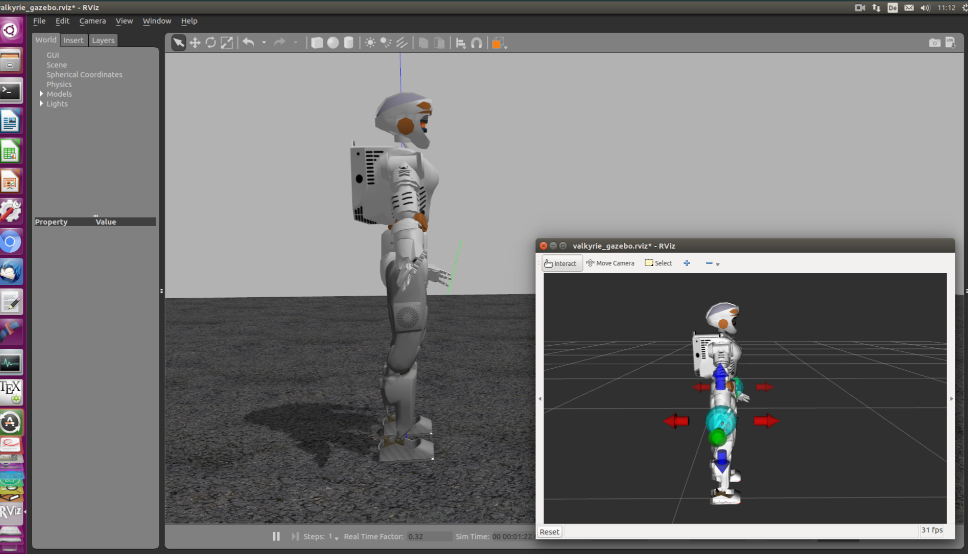
Shadow Hand Mixer Test

video

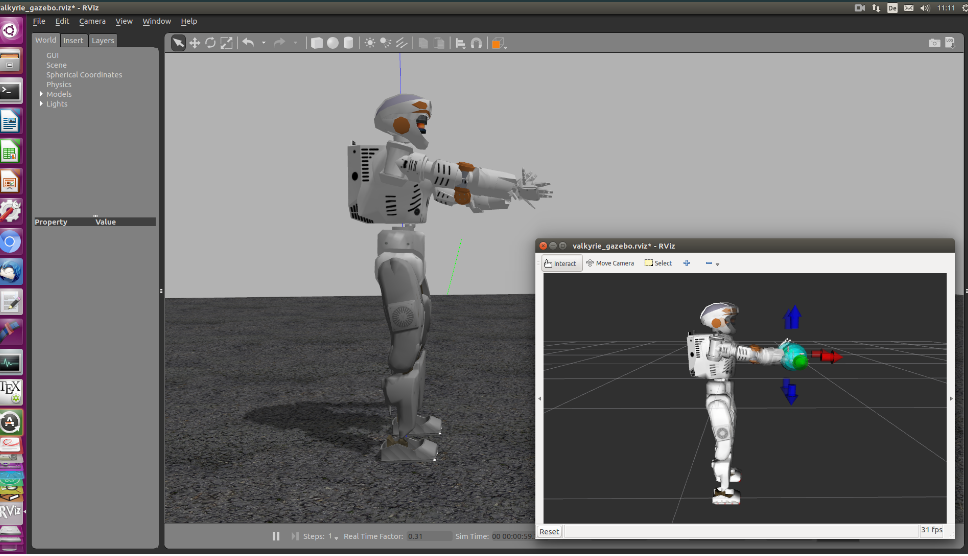
Valkyrie Balancing Test

- Hands (PositionGoal, interactive)
- Feet (PoseGoal)
- Foot reflexes
- Body upright (PositionGoal, OrientationGoal, OrientationGoal)
- BalanceGoal

Valkyrie Balancing Test



Valkyrie Balancing Test



alkyrie_gazebo.rviz* - RViz
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.31 Sim Time: 00 00:00:59
valkyrie_gazebo.rviz* - RViz
Interact Move Camera Select
Reset
31 fps

Valkyrie Balancing Test

video

Benchmark

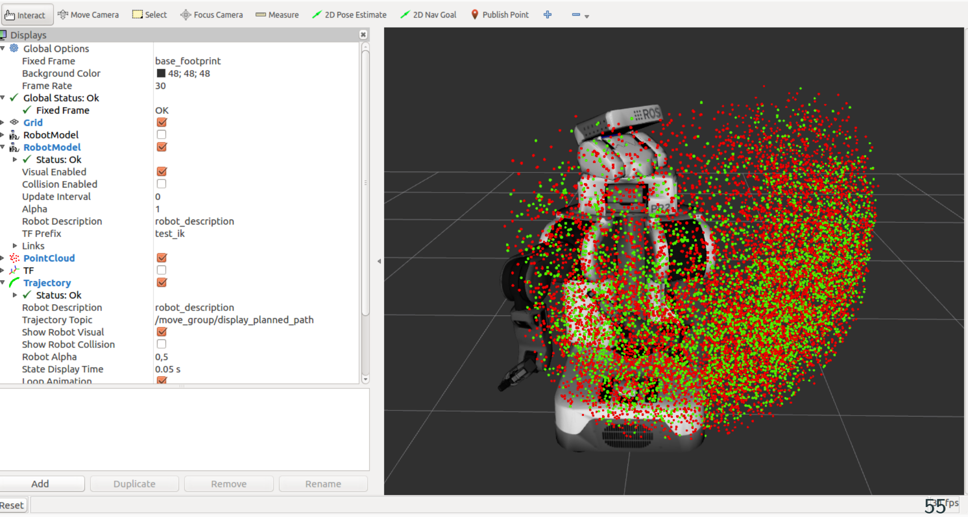
Benchmark

- Generate random joint values
- Compute forward kinematics
- Compute inverse kinematics
- Compare results

- Timeout (5ms)
- Max error (10^{-5})
- 10000 samples
- Success rate
- Average solve time

IK Test - PR2 - KDL

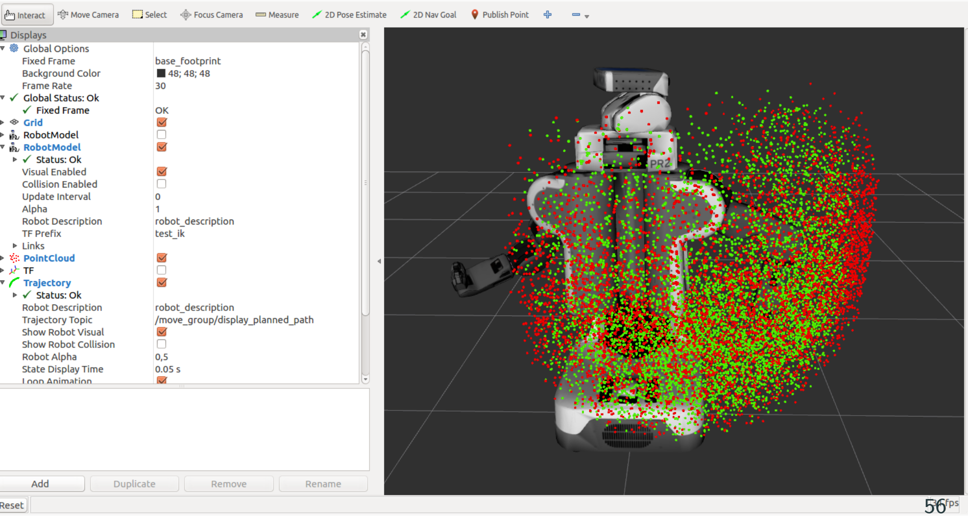
- 53.44% success rate



Global Options
Fixed Frame: base_footprint
Background Color: ■ 48; 48; 48
Frame Rate: 30
✓ Global Status: Ok
✓ Fixed Frame: OK
Grid: ✓
RobotModel: ☐
RobotModel: ✓
✓ Status: Ok
Visual Enabled: ✓
Collision Enabled: ☐
Update Interval: 0
Alpha: 1
Robot Description: robot_description
TF Prefix: test_ik
Links
PointCloud: ✓
TF: ☐
Trajectory: ✓
✓ Status: Ok
Robot Description: robot_description
Trajectory Topic: /move_group/display_planned_path
Show Robot Visual: ✓
Show Robot Collision: ☐
Robot Alpha: 0,5
State Display Time: 0.05 s
Loop Animation: ✓

IK Test - PR2 - BioIK 1

- 70.34% success rate

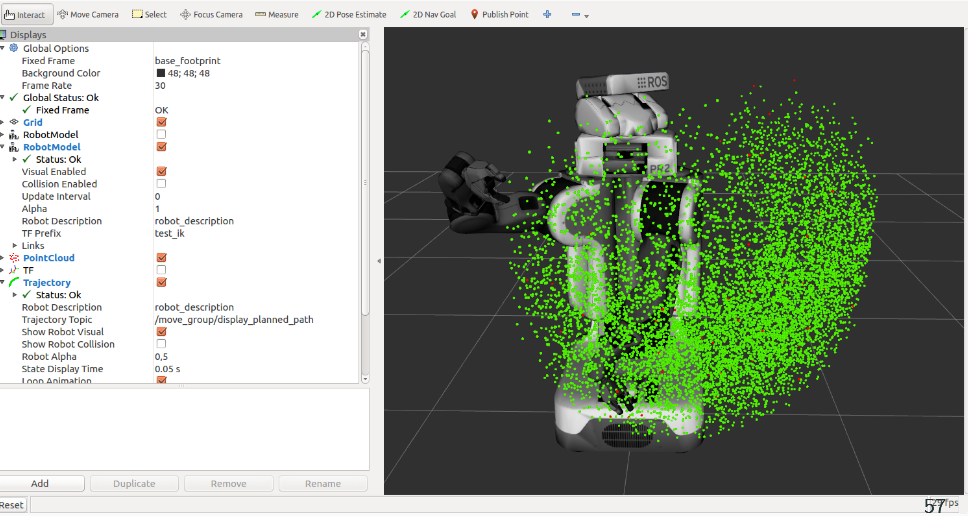


Global Options
Fixed Frame base_footprint
Background Color 48; 48; 48
Frame Rate 30
✓ Global Status: Ok
✓ Fixed Frame OK
Grid ✓
RobotModel ☐
RobotModel ✓
✓ Status: Ok
Visual Enabled ✓
Collision Enabled ☐
Update Interval 0
Alpha 1
Robot Description robot_description
TF Prefix test_ik
Links
PointCloud ✓
TF ☐
Trajectory ✓
✓ Status: Ok
Robot Description robot_description
Trajectory Topic /move_group/display_planned_path
Show Robot Visual ✓
Show Robot Collision ☐
Robot Alpha 0,5
State Display Time 0.05 s
Loop Animation ✓

36 fps

IK Test - PR2 - TRAC IK

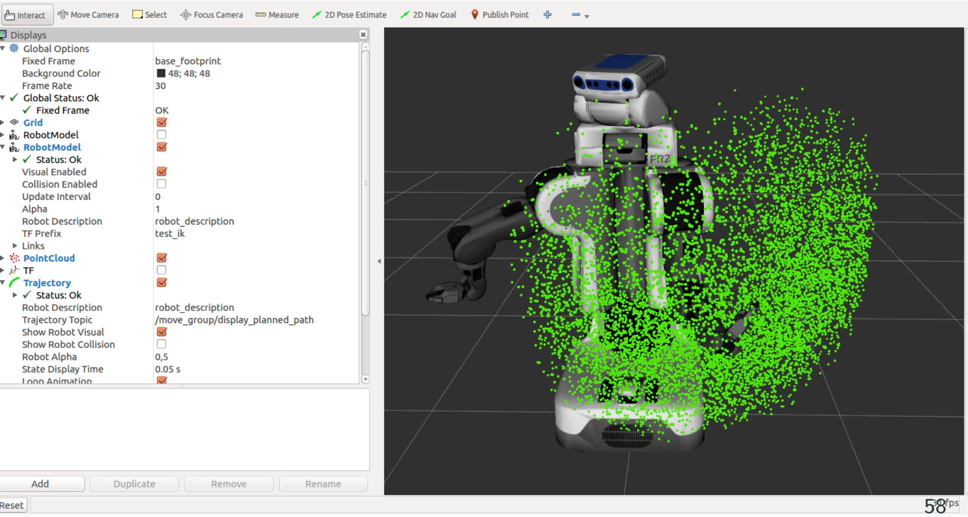
- 99.69% success rate



Global Options	
Fixed Frame	base_footprint
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	☐
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Update Interval	0
Alpha	1
Robot Description	robot_description
TF Prefix	test_ik
Links	
PointCloud	✓
TF	☐
Trajectory	✓
✓ Status: Ok	
Robot Description	robot_description
Trajectory Topic	/move_group/display_planned_path
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	0,5
State Display Time	0.05 s
Loop Animation	✓

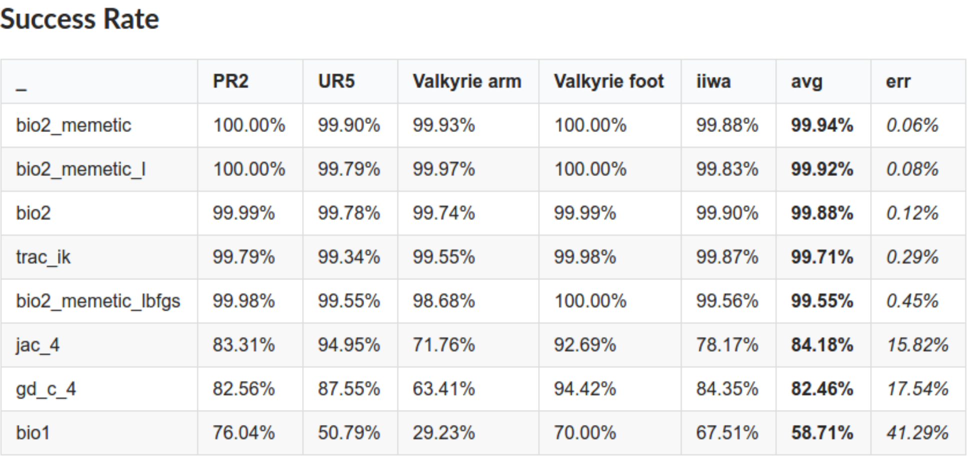
IK Test - PR2 - BioIK 2

- 99.99% success rate



Global Options	
Fixed Frame	base_footprint
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	☐
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Update Interval	0
Alpha	1
Robot Description	robot_description
TF Prefix	test_ik
Links	
PointCloud	✓
TF	☐
Trajectory	✓
✓ Status: Ok	
Robot Description	robot_description
Trajectory Topic	/move_group/display_planned_path
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	0,5
State Display Time	0.05 s
Loop Animation	✓

Benchmark

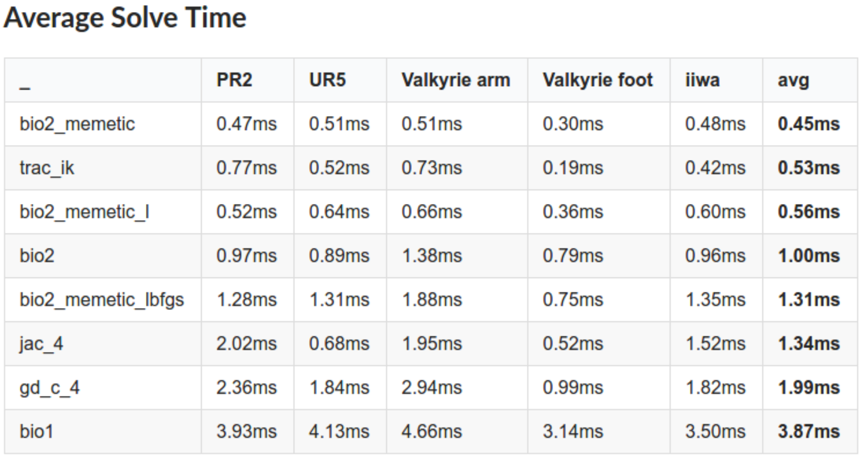


Success Rate

_	PR2	UR5	Valkyrie arm	Valkyrie foot	iiwa	avg	err
bio2_memetic	100.00%	99.90%	99.93%	100.00%	99.88%	**99.94%**	*0.06%*
bio2_memetic_l	100.00%	99.79%	99.97%	100.00%	99.83%	**99.92%**	*0.08%*
bio2	99.99%	99.78%	99.74%	99.99%	99.90%	**99.88%**	*0.12%*
trac_ik	99.79%	99.34%	99.55%	99.98%	99.87%	**99.71%**	*0.29%*
bio2_memetic_lbfgs	99.98%	99.55%	98.68%	100.00%	99.56%	**99.55%**	*0.45%*
jac_4	83.31%	94.95%	71.76%	92.69%	78.17%	**84.18%**	*15.82%*
gd_c_4	82.56%	87.55%	63.41%	94.42%	84.35%	**82.46%**	*17.54%*
bio1	76.04%	50.79%	29.23%	70.00%	67.51%	**58.71%**	*41.29%*

BioIK 99.94%, TRAC_IK 99.71%

Benchmark



Average Solve Time

_	PR2	UR5	Valkyrie arm	Valkyrie foot	iiwa	avg
bio2_memetic	0.47ms	0.51ms	0.51ms	0.30ms	0.48ms	**0.45ms**
trac_ik	0.77ms	0.52ms	0.73ms	0.19ms	0.42ms	**0.53ms**
bio2_memetic_l	0.52ms	0.64ms	0.66ms	0.36ms	0.60ms	**0.56ms**
bio2	0.97ms	0.89ms	1.38ms	0.79ms	0.96ms	**1.00ms**
bio2_memetic_lbfgs	1.28ms	1.31ms	1.88ms	0.75ms	1.35ms	**1.31ms**
jac_4	2.02ms	0.68ms	1.95ms	0.52ms	1.52ms	**1.34ms**
gd_c_4	2.36ms	1.84ms	2.94ms	0.99ms	1.82ms	**1.99ms**
bio1	3.93ms	4.13ms	4.66ms	3.14ms	3.50ms	**3.87ms**

BioIK: 0.45ms, TRAC_IK: 0.53ms

Benchmark

BioIK

- Higher success rate
  (BioIK 99.94%, TRAC_IK 99.71%)
- Lower average solve time
  (BioIK: 0.45ms, TRAC_IK: 0.53ms)