Kinova Jaco Arm Control via ROS

Robot-Era Project

Johannes Liebrecht Sebastian Rockel

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

19. November 2012

Outline

1. Kinova Jaco Arm Overview
2. Kinova Jaco Arm Library
3. Jaco Arm ROS Stack
4. Open Motion Planning Library
5. Demo

Outline

1. Kinova Jaco Arm Overview
2. Kinova Jaco Arm Library
3. Jaco Arm ROS Stack
4. Open Motion Planning Library
5. Demo

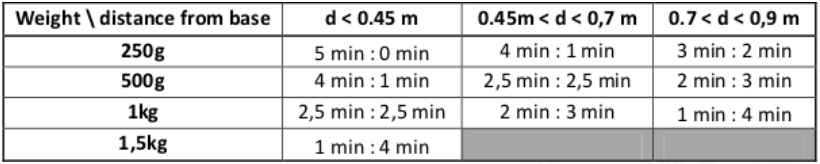
Kinova Jaco Arm

- 6 degrees of freedom
- carbon fiber structure
- total weight: 5Kg
- reach : 90cm

Kinova Jaco Arm

- Maximum Load : 1.5kg at mid-range/1.0kg at end-range
- Maximum linear arm speed : 15cm/sec
- 3 fingers or 2 fingers utilization
- Finger force limited to 7N

Jaco Arm Rest Time



Weight \ distance from base	d < 0.45 m	0.45m < d < 0,7 m	0.7 < d < 0,9 m
250g	5 min : 0 min	4 min : 1 min	3 min : 2 min
500g	4 min : 1 min	2,5 min : 2,5 min	2 min : 3 min
1kg	2,5 min : 2,5 min	2 min : 3 min	1 min : 4 min
1,5kg	1 min : 4 min		

Working with Kinova Jaco Arm

- be aware of collusion with camera head
- home position/ retract position

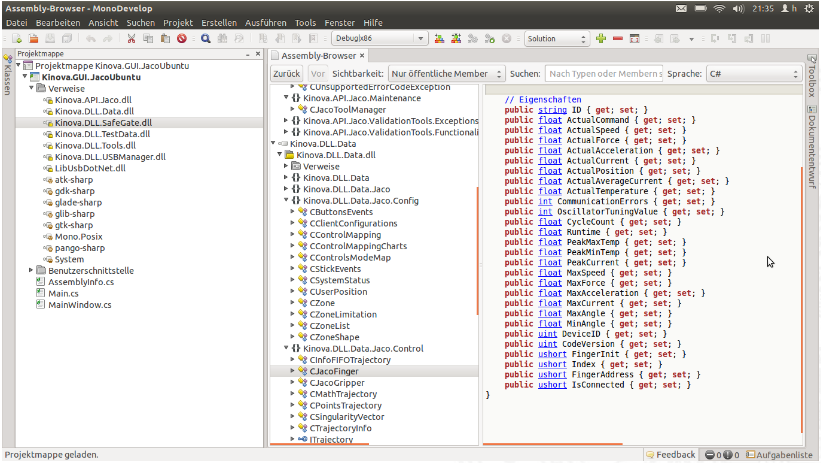
Outline

1. Kinova Jaco Arm Overview
2. Kinova Jaco Arm Library
3. Jaco Arm ROS Stack
4. Open Motion Planning Library
5. Demo

Jaco Arm Library

- powerful library(C#)
- documentation (`Jaco_API` Programming Guide)
- view dynamic-link library with MonoDevelop

MonoDevelop

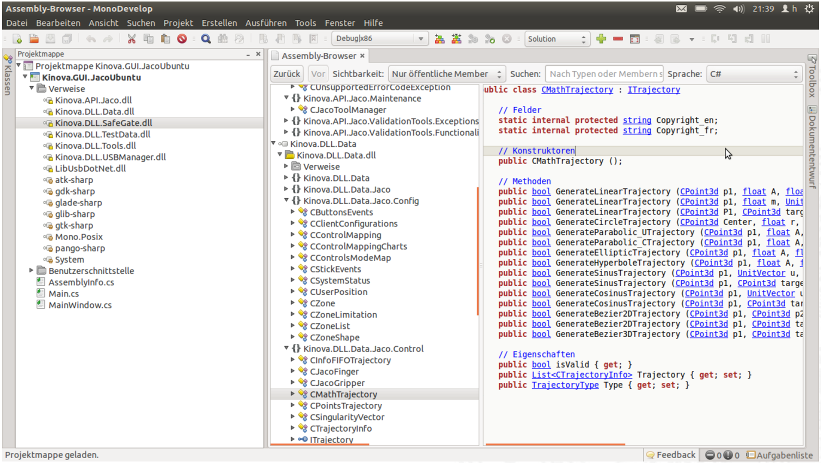


```
    // Eigenschaften
    public string ID { get; set; }
    public float ActualCommand { get; set; }
    public float ActualSpeed { get; set; }
    public float ActualForce { get; set; }
    public float ActualAcceleration { get; set; }
    public float ActualCurrent { get; set; }
    public float ActualPosition { get; set; }
    public float ActualAverageCurrent { get; set; }
    public float ActualTemperature { get; set; }
    public int CommunicationErrors { get; set; }
    public int OscillatorTuningValue { get; set; }
    public float CycleCount { get; set; }
    public float Runtime { get; set; }
    public float PeakMaxTemp { get; set; }
    public float PeakMinTemp { get; set; }
    public float PeakCurrent { get; set; }
    public float MaxSpeed { get; set; }
    public float MaxForce { get; set; }
    public float MaxAcceleration { get; set; }
    public float MaxCurrent { get; set; }
    public float MaxAngle { get; set; }
    public float MinAngle { get; set; }
    public uint DeviceID { get; set; }
    public uint CodeVersion { get; set; }
    public ushort FingerInit { get; set; }
    public ushort Index { get; set; }
    public ushort FingerAddress { get; set; }
    public ushort IsConnected { get; set; }
}
```

Jaco Arm Library

- acceleration, velocity, force, joint temperature,....
- protection zones
- trajectories

Jaco Arm Library



```
ublic class CMathTrajectory : ITrajectory

   // Felder
   static internal protected string Copyright_en;
   static internal protected string Copyright_fr;

   // Konstruktoren
   public CMathTrajectory ();

   // Methoden
   public bool GenerateLinearTrajectory (CPoint3d p1, float A, floa
   public bool GenerateLinearTrajectory (CPoint3d p1, float m, Unit
   public bool GenerateLinearTrajectory (CPoint3d P1, CPoint3d targ
   public bool GenerateCircleTrajectory (CPoint3d Center, float r,
   public bool GenerateParabolic_UTrajectory (CPoint3d p1, float A,
   public bool GenerateParabolic_CTrajectory (CPoint3d p1, float A,
   public bool GenerateEllipticTrajectory (CPoint3d p1, float A, fl
   public bool GenerateHyperboleTrajectory (CPoint3d p1, float A, f
   public bool GenerateSinusTrajectory (CPoint3d p1, UnitVector u,
   public bool GenerateSinusTrajectory (CPoint3d p1, CPoint3d targe
   public bool GenerateCosinusTrajectory (CPoint3d p1, UnitVector u
   public bool GenerateCosinusTrajectory (CPoint3d p1, CPoint3d tar
   public bool GenerateBezier2DTrajectory (CPoint3d p1, CPoint3d p2
   public bool GenerateBezier2DTrajectory (CPoint3d p1, CPoint3d ta
   public bool GenerateBezier3DTrajectory (CPoint3d p1, CPoint3d ta

   // Eigenschaften
   public bool isValid { get; }
   public List<CTrajectoryInfo> Trajectory { get; set; }
   public TrajectoryType Type { get; set; }
```

Projektmappe geladen.

Outline

1. Kinova Jaco Arm Overview
2. Kinova Jaco Arm Library
3. Jaco Arm ROS Stack
4. Open Motion Planning Library
5. Demo

Kinova Jaco Arm ROS Stack

- unofficial ROS stack from Kinova
- no existing documentation
- C# code into C++ code

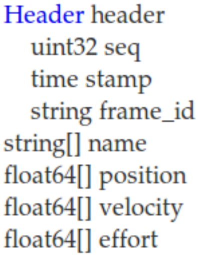
Kinova Jaco Arm ROS Stack

- Jaco Node
- Jaco State Publisher(`robot_state_publisher`)

Jaco_Node Subscriber Topics

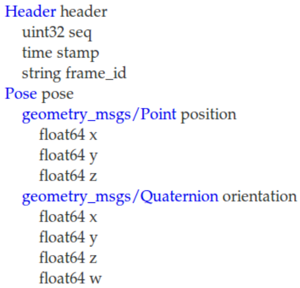
- /jaco_node/cur_goal(geometry_msgs/PoseStamped)
- /hand_pose(geometry_msgs/PoseStamped)
- /joint_states(sensor_msgs/JointState)

Jaco_Node Subscriber Messages



```
Header header
  uint32 seq
  time stamp
  string frame_id
string[] name
float64[] position
float64[] velocity
float64[] effort
```

sensor_msgs/JointState



```
Header header
  uint32 seq
  time stamp
  string frame_id
Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

geometry_msgs/PoseStamped

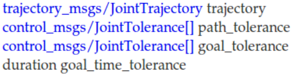
Jaco_Node Publisher Topics/Messages

- `/hand_goal(geometry_msgs/PoseStamped)`
- `/cmd_abs_finger(jaco_node/FingerPose)`
- `/cmd_abs_joint(jaco_node/JointPose)`
- `/cmd_rel_cart(geometry_msgs/Twist)`

Jaco_Node Publishers/Subscribers

- `/.../follow_joint_trajectory/result`
- `/.../follow_joint_trajectory/feedback`
- `/.../follow_joint_trajectory/goal`
- `/.../follow_joint_trajectory/status`
- `/.../follow_joint_trajectory/cancel`

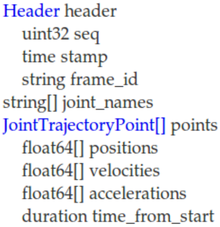
Jaco_Node Joint Trajectory Messages



```
trajectory_msgs/JointTrajectory trajectory
control_msgs/JointTolerance[] path_tolerance
control_msgs/JointTolerance[] goal_tolerance
duration goal_time_tolerance
```

control_msgs/

FollowJointTrajectoryGoal.msg



```
Header header
    uint32 seq
    time stamp
    string frame_id
string[] joint_names
JointTrajectoryPoint[] points
    float64[] positions
    float64[] velocities
    float64[] accelerations
    duration time_from_start
```

trajectory_msgs/JointTrajectory.msg

Kinova Jaco Arm ROS Stack

- fixing bugs(open his hand, `hand_pose`)
- integrate more functionality

Outline

1. Kinova Jaco Arm Overview
2. Kinova Jaco Arm Library
3. Jaco Arm ROS Stack
4. Open Motion Planning Library
5. Demo

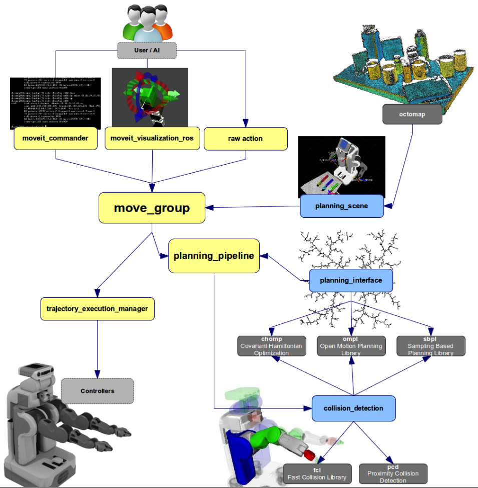
The Open Motion Planning Library (OMPL)

- library of sampling-based motion planning algorithms
- integrated in ROS arm navigation stack (used on the PR2)
- integrated in (new) MoveIt! project
- includes state-of-the-art motion planning algorithms
- no collision checking
- demo videos at `http://ompl.kavrakilab.org/gallery.html`
- tutorials on how to integrate OMPL at `http://www.ros.org/wiki/ompl_ros_interface/Tutorials`

MoveIt! - A Planning Framework

- includes kinematics, dynamics, collision checking, constraints evaluation, visualization ..
- centered around planning and execution motion plans for different robots
- Tools include: specification of motion plans, configuration and debugging tools, visualization, benchmarking
- Overview at `http://moveit.ros.org`

MoveIt! - A Planning Framework



Outline

1. Kinova Jaco Arm Overview
2. Kinova Jaco Arm Library
3. Jaco Arm ROS Stack
4. Open Motion Planning Library
5. Demo

Demo