Jacobiano no ROS 2

Walter Fetter Lages

fetter@ece.ufrgs.br

Universidade Federal do Rio Grande do Sul

Escola de Engenharia

Departamento de Sistemas Elétricos de Automação e Energia

ENG10052 Laboratório de Robótica

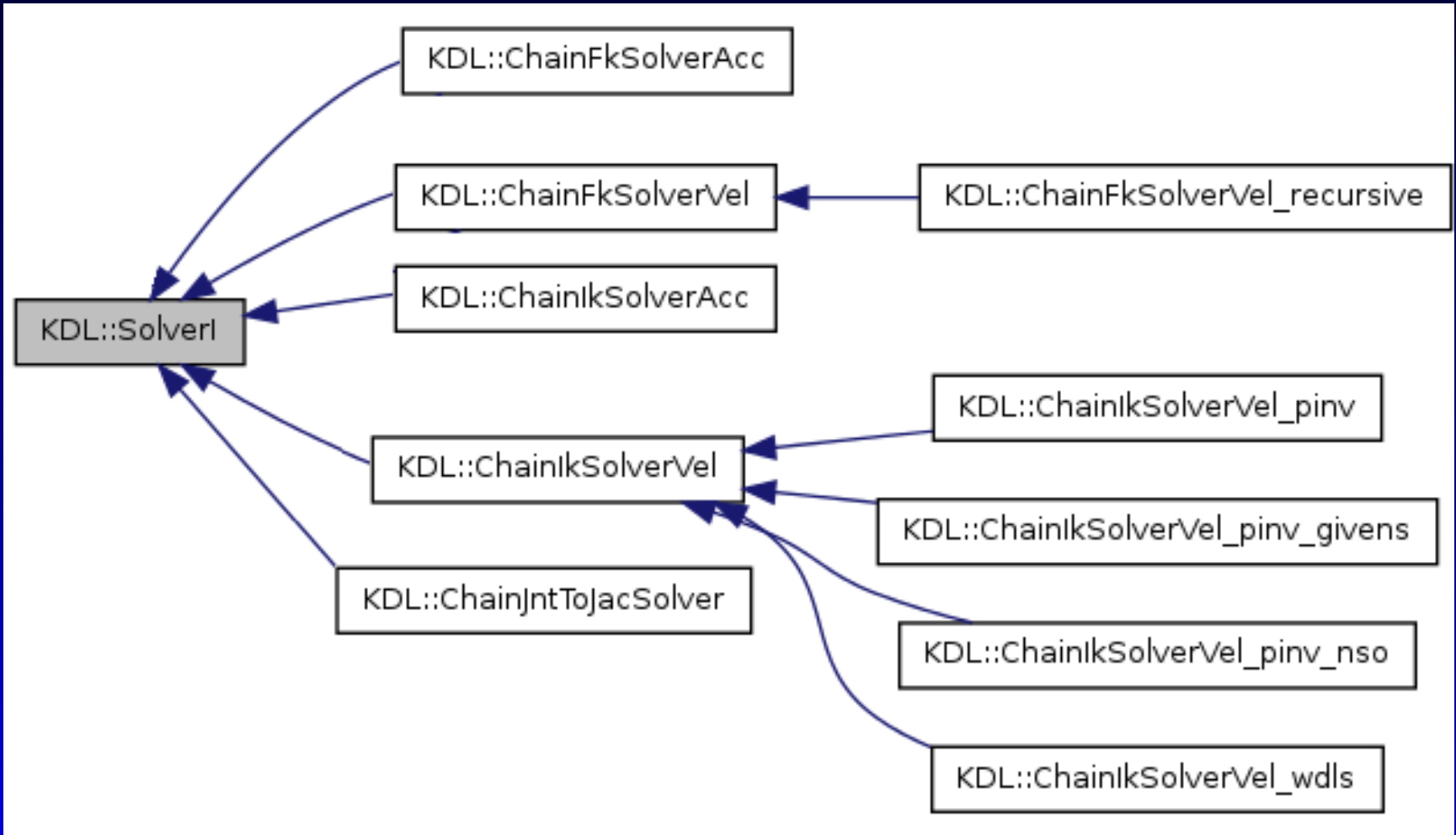
UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Jacobiano no ROS

- Implementado na biblioteca KDL (*Kinematics and Dynamics Library*)
- Implementa classes virtuais para cálculo do Jacobiano, Jacobiano inverso e suas Derivadas
- A maioria calcula diretamente a velocidade e/ou aceleração e não os Jacobianos em si
- Inversas calculadas numericamente
- Várias implementações, com algoritmos diferentes

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Hierarquia de Classes



UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Classes Virtuais

`KDL::ChainFkSolverVel:` Velocidade da garra a partir da velocidade das juntas

`KDL::ChainIkSolverVel:` Velocidade das juntas a partir da velocidade da garra

`KDL::ChainFkSolverAcc:` Aceleração da garra a partir da aceleração das juntas

`KDL::ChainIkSolverAcc:` Aceleração das juntas a partir da aceleração da garra

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Classes para o Jacobiano

`KDL::ChainJntToJacSolver`: Jacobiano

```
int JntToJac(const JntArray &q_in,Jacobian &jac,int
    segmentNR=−1)
int setLockedJoints(const std::vector<bool> locked_joints)
```

`KDL::ChainFkSolverVel_recursive`: Velocidade da garra a partir da velocidade das juntas, calculada recursivamente

```
int JntToCart(const JntArrayVel &q_in,FrameVel &out,int
    segmentNr=−1)
int JntToCart(const JntArrayVel &q_in,std::vector<FrameVel>
    &out,int segmentNr=−1)
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Classes para o Jacobiano Inverso

`KDL::ChainIkSolverVel_pinv`:
Pseudo-inversa calculada por SVD

`KDL::ChainIkSolverVel_pinv_givens`:
Pseudo-inversa calculada usando rotação de Givens

`KDL::ChainIkSolverVel_pinv_nso`:
Pseudo-inversa generalizada, com minimização do erro quadrático para robôs redundantes

`KDL::ChainIkSolverVel_wdls`: Mínimos quadrados amortecidos, considerando ponderações no espaço das juntas e no espaço cartesiano

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Funções para o Jacobiano Inverso

```
int CartToJnt(const JntArray &q_in,const Twist &v_in,JntArray &
      qdot_out)
int CartToJnt(const JntArray &q_init,const FrameVel &v_in,
      JntArrayVel &q_out)
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Classes para a Derivada do Jacobiano

`KDL::ChainJntToJacDotSolver:` Derivada do Jacobiano

```
int JntToJacDot(const KDL::JntArrayVel &q_in,KDL::Jacobian
    &jdot,int seg_nr=-1)
void setBodyFixedRepresentation(void)
void setHybridRepresentation(void)
void setInternialRepresentation(void)
int setLockedJoints(const std::vector<bool> locked_joints)
```

`KDL::ChainFkSolverAcc:` Aceleração da garra a partir da aceleração das juntas

```
int JntToCart(const JntArrayAcc &q_in,FrameAcc &out,int
    segmentNr=-1)
int JntToCart(const JntArrayAcc &q_in,std::vector<FrameAcc>
    &out,int segmentNr=-1)
```

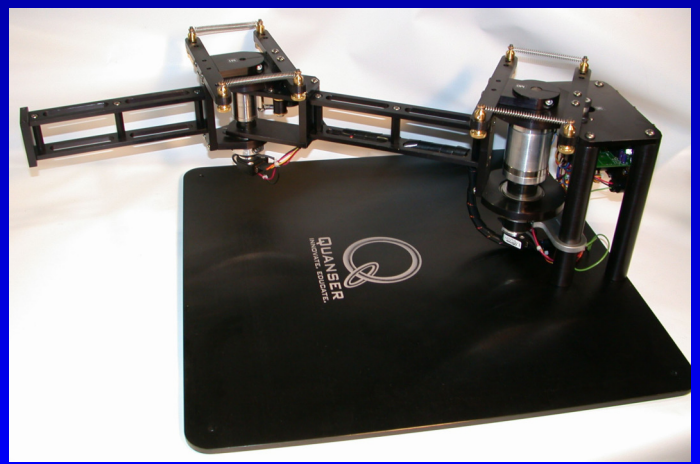
Mapeamento Inverso da Aceleração

`KDL::ChainIkSolverAcc:` Aceleração das juntas a partir da aceleração da garra

```
int CartToJnt(const JntArray &q_in,const JntArray &qdot_in,
    const Twist a_in,JntArray &qdotdot_out)
int CartTojnt(const JntArray &q_init,const FrameAcc &a_in,
    JntArrayAcc &q_out)
int CartToJnt(const JntArray &q_in,const Twist &v_in,const
    Twist &a_in,JntArray &qdot_out,JntArray &qdotdot_out)
int CartTojnt(const JntArray &q_init,const Frame &p_in,const
    JntArray &qdot_in,const Twist &a_in, JntArray &q_out,
    JntArray &qdotdot_out)
```

Exemplo

- O tópico `/tf` publica as transformações entre os vários sistemas de coordenadas
- O tópico `/joint_state` publica posição, velocidade, aceleração e esforço nas juntas
- Será criado um nodo para publicar a velocidade da garra
- Será usado o pacote `q2d_description`
  - Já usado e instalado anteriormente



Instalação do Pacote

- Clonar e compilar o repositório `q2d`

```
cd ~/colcon_ws/src
git clone -b $ROS_DISTRO http://git.ece.ufrgs.br/q2d
touch q2d/q2d_bringup/COLCON_IGNORE
touch q2d/q2d_teleop/COLCON_IGNORE
cd .. ; colcon build --symlink-install
source ~/colcon_ws/install/setup.bash
```

- Atualizar, se já instalado

```
cd ~/colcon_ws/src/q2d
git pull
cd .. ; colcon build --symlink-install
source ~/colcon_ws/install/setup.bash
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Velocidades Usando a KDL

- Neste exemplo, a velocidade da garra será obtida como objeto do tipo `KDL::FrameVel`
  - Classe para representar velocidades cartesianas definida na KDL
- Será criado o nodo `twist_publisher` que publicará o *twist* de velocidade da garra
- Será criado no pacote `eng10026_jacobian` para conter o nodo

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Pacote `eng10026_jacobian`

- Criar o pacote:

---

cd ~/colcon_ws/src

ros2 pkg create −−build−type ament_cmake −−dependencies roscpp geometry_msgs sensor_msgs std_msgs tf2_kdl kdl_parser orocos_kdl −−node−name twist_publisher eng10026_jacobian

---

- `package.xml` deve ser editado para configurar os detalhes de documentação e incluir dependências

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

CMakeLists.txt

- Editar `CMakeLists.txt` para descomentar e ajustar as *tags*:

```
add_executable(twist_publisher src/twist_publisher.cpp)

ament_target_dependencies(twist_publisher
  roscpp geometry_msgs sensor_msgs std_msgs tf2_kdl
    kdl_parser orocos_kdl)

install(TARGETS twist_publisher
  DESTINATION lib/${PROJECT_NAME})

install(DIRECTORY launch
  DESTINATION share/${PROJECT_NAME})
```

Inclusão no Meta-Pacote

- O pacote `eng10026_jacobian` será incluido no meta-pacote `eng10026`
- Editar o arquivo `package.xml` do pacote `eng10026` e incluir

<run_depend>eng10026_jacobian</run_depend>

---

twist_publisher.cpp

```
#include <rclcpp/rclcpp.hpp>
#include <geometry_msgs/msg/twist_stamped.hpp>
#include <sensor_msgs/msg/joint_state.hpp>
#include <std_msgs/msg/string.hpp>
#include <kdl/chainfksolvervel_recursive.hpp>
#include <tf2_kdl/tf2_kdl.h>
#include <kdl_parser/kdl_parser.hpp>

class TwistPublisher: public rclcpp::Node
{
  public:
  TwistPublisher(const char *name,const char *frameId,const char *
      childFrameId);
  ~TwistPublisher(void);
  void publish(void) const;
```

twist_publisher.cpp

```
private:
rclcpp::Subscription<sensor_msgs::msg::JointState>::SharedPtr
    jointStatesSubscriber_;
rclcpp::Publisher<geometry_msgs::msg::TwistStamped>::SharedPtr
    twistPublisher_;

std::string frameId_;

std::string robotDescription_;
KDL::Chain chain_;
KDL::JntArrayVel jointPosVel_;
KDL::ChainFkSolverVel_recursive *fkSolverVel_;
```

twist_publisher.cpp

```
  void jointStatesCB(const sensor_msgs::msg::JointState::SharedPtr
      jointStates);
  void robotDescriptionCB(const std_msgs::msg::String::SharedPtr
      robotDescription);
};
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

twist_publisher.cpp

```
TwistPublisher::TwistPublisher(const char *name,const char *
    frameId,const char *childFrameId)
  :Node(name), jointPosVel_(0)
{
  frameId_=frameId;

  using std::placeholders::_1;
  jointStatesSubscriber_=create_subscription<sensor_msgs::msg::
    JointState>("joint_states",100,std::bind(&TwistPublisher::
    jointStatesCB,this,_1));

  twistPublisher_=create_publisher<geometry_msgs::msg::
    TwistStamped>("twist",100);
```

twist_publisher.cpp

```
rclcpp::QoS qos(rclcpp::KeepLast(1));
qos.transient_local();
auto robotDescriptionSubscriber=create_subscription<std_msgs::
    msg::String>("robot_description",qos,std::bind(&TwistPublisher::
    robotDescriptionCB,this,_1));
while(robotDescription_.empty())
{
   RCLCPP_WARN_SKIPFIRST_THROTTLE(get_logger(),*
    get_clock(),1000,"Waiting for robot model on /robot_description."
    );
   rclcpp::spin_some(get_node_base_interface());
}
```

twist_publisher.cpp

```
  KDL::Tree tree;
  if (!kdl_parser::treeFromString(robotDescription_,tree))
    RCLCPP_ERROR(get_logger(),"Failed to construct KDL tree.");

  if (!tree.getChain(frameId_,childFrameId,chain_))
    RCLCPP_ERROR(get_logger(),"Failed to get chain from KDL
     tree.");

  fkSolverVel_=new KDL::ChainFkSolverVel_recursive(chain_);

  jointPosVel_.resize(chain_.getNrOfJoints());
}
```

twist_publisher.cpp

```
TwistPublisher::~TwistPublisher(void)
{
  delete fkSolverVel_;
}

void TwistPublisher::jointStatesCB(const sensor_msgs::msg::
      JointState::SharedPtr jointStates)
{
  for(unsigned int i=0;i < jointPosVel_.q.rows();i++)
  {
    jointPosVel_.q(i)=jointStates->position[i];
    jointPosVel_.qdot(i)=jointStates->velocity[i];
  }
  publish();
}
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

twist_publisher.cpp

```
void TwistPublisher::robotDescriptionCB(const std_msgs::msg::String
      ::SharedPtr robotDescription)
{
      robotDescription_=robotDescription->data;
}

void TwistPublisher::publish(void) const
{
  KDL::FrameVel frameVel;
  if(fkSolverVel_->JntToCart(jointPosVel_,frameVel) < 0)
    RCLCPP_ERROR(get_logger(),"Failed to compute forward
     kinematics velocity.");

  auto twistMsg=tf2::toMsg(tf2::Stamped<KDL::Twist>(frameVel.
     GetTwist(),tf2::get_now(),frameId_));
  twistPublisher_->publish(twistMsg);
```

twist_publisher.cpp

```
int main(int argc,char* argv[])
{
  rclcpp::init(argc,argv);

  if(argc < 3)
  {
        std::cerr << "Usage: twist_publisher base_frame tool_frame\
    n";
        return -1;
  }

  TwistPublisher twistPublisher("twist_publisher",argv[1],argv[2]);

  rclcpp::spin(twistPublisher.get_node_base_interface());
  return 0;
```

Instalação do Pacote

- Clonar e compilar o repositório `eng10026_jacobian`

```
cd ~/colcon_ws/src
git clone -b $ROS_DISTRO http://git.ece.ufrgs.br/eng10026/
    eng10026_jacobian
cd ..
colcon build --symlink-install
source $HOME/colcon_ws/install/setup.bash
```

display.launch.xml

```
<launch>
  <arg name="use_gui" default="false"/>

  <group>
    <include file="$(find-pkg-share q2d_description)/launch/display.launch.xml">
      <arg name="use_gui" value="false"/>
    </include>
  </group>
  <node if="$(var use_gui)" name="joint_state_publisher" pkg="
       joint_state_publisher_gui" exec="joint_state_publisher_gui">
    <param name="publish_default_velocities" value="true"/>
    <param name="publish_default_efforts" value="true"/>
    <param name="source_list" value="['partial_joint_states']"/>
  </node>
  <node name="twist_publisher" pkg="eng10026_jacobian" exec="twist_publisher" args=
       "origin_link tool_link"/>
</launch>
```

Execução

ros2 launch eng10026_jacobian display.launch.xml use_gui:=**true**

- Publicar a velocidade das juntas

  ros2 topic pub /partial_joint_states sensor_msgs/msg/JointState
  '{name: ["shoulder_active_joint", "elbow_active_joint"],
  velocity: [1.0, 1.0]}'

- Verificar a velocidade da garra

  ros2 topic echo /twist

- Movendo os *sliders* para mover o robô a transformação homogênea e a velocidade da garra modificam-se correspondentemente

Gráfico de Computação