Cinemática Direta 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

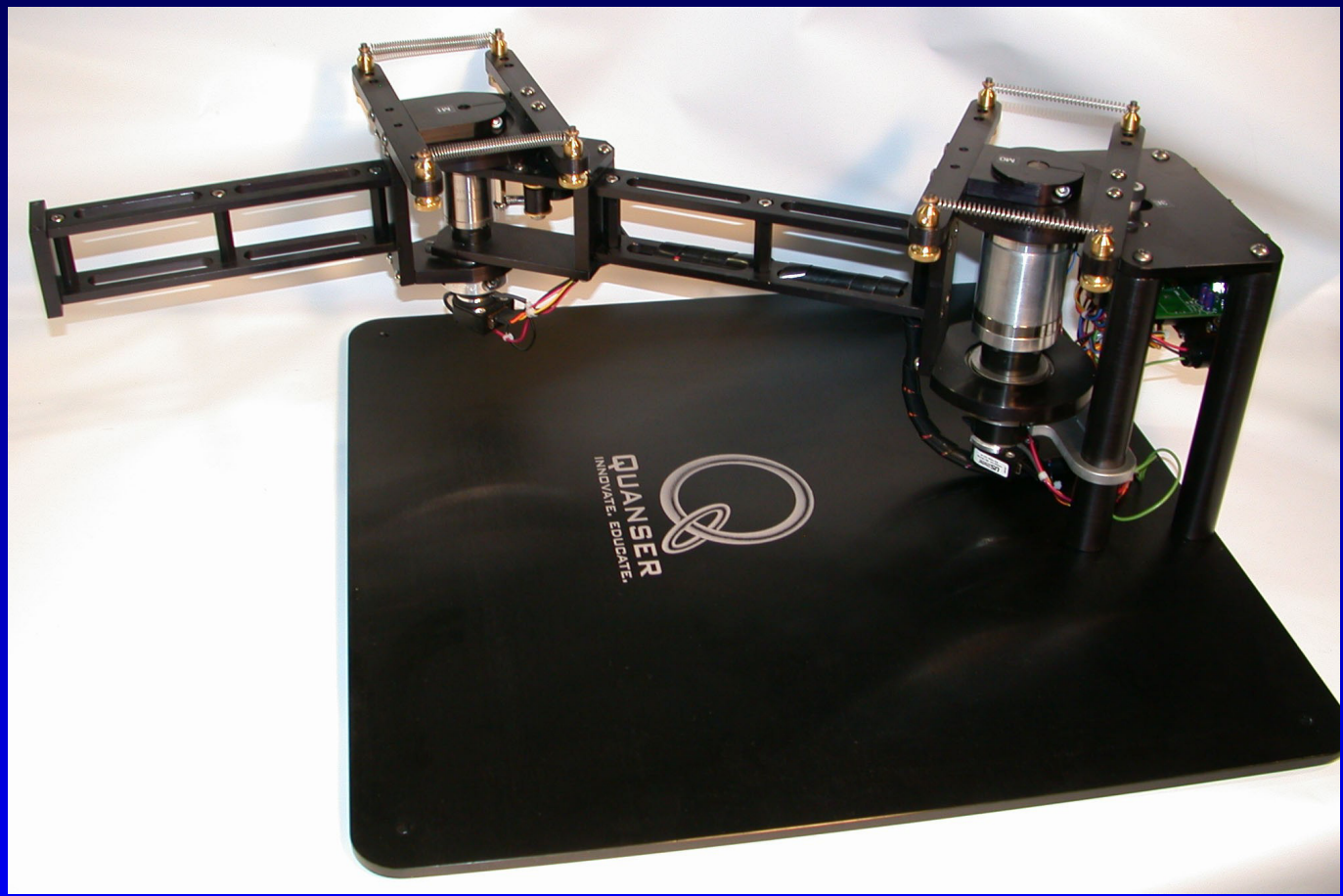
UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Cinemática Direta no ROS

- Usando a biblioteca KDL (*Kinematics and Dynamics Library*)
  - Simples se considerado apenas o robô
  - Para vários objetos tem-se que montar a cadeia cinemática ”na mão“
  - Não depende da infraestrutura de tópicos
  - *Real-time ready*
  - Só acessível por programa
- Através do tópico `/tf`
  - Consulta-se diretamente a transformação de interesse
  - Depende da infraestrutura de tópicos
  - Acessível por programa ou linha de comando

Exemplo

- Será usado o pacote `q2d_description`
- Já usado e instalado anteriormente



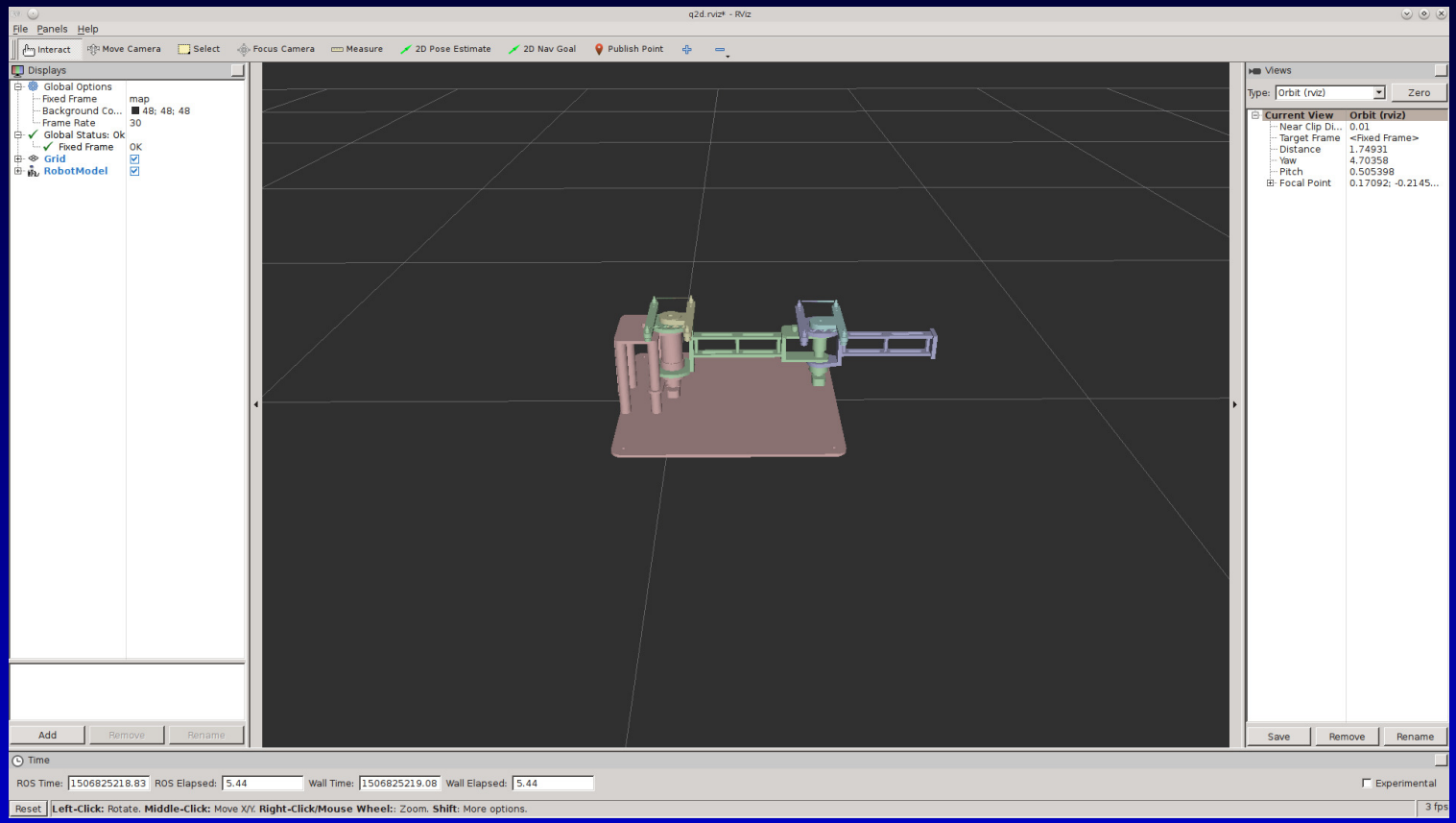
Instalação do Pacote

- 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_ws
colcon build ––symlink–install
source ~/colcon_ws/install/setup.bash
```

Visualização no RViz

ros2 launch q2d_description display.launch.xml gui:=**true**



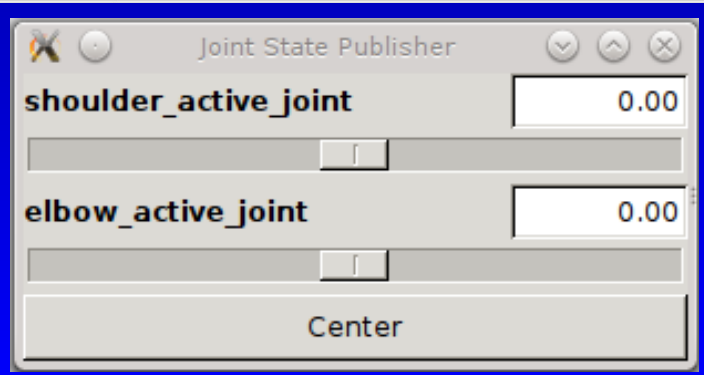
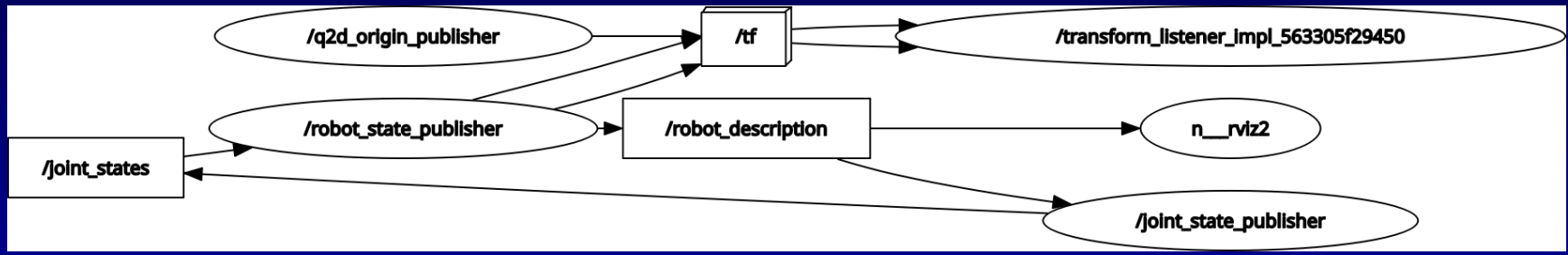


Gráfico de Computação

- Em outro terminal:

```
source ~/colcon_ws/install/setup.bash
rqt_graph &
```



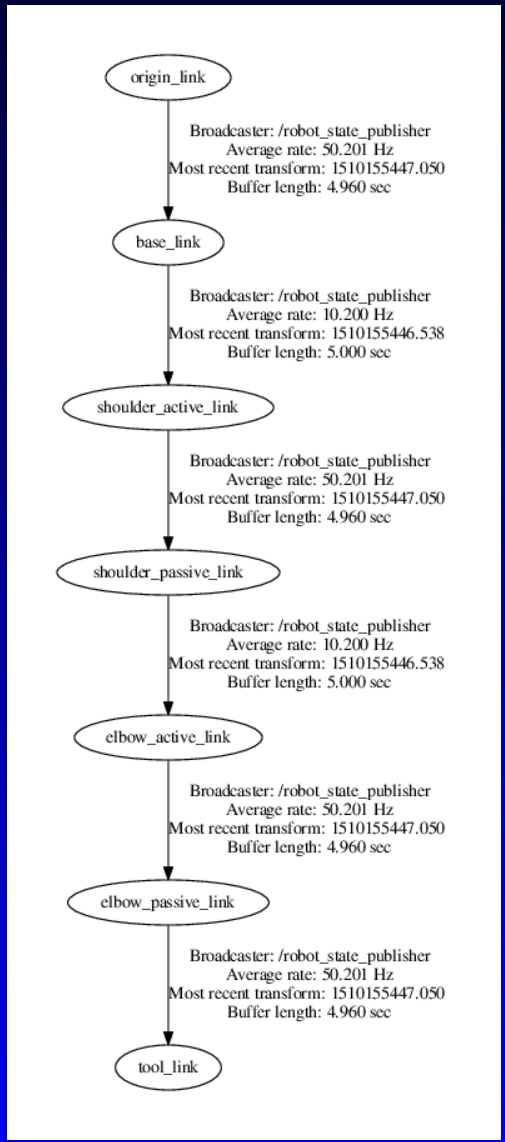
Árvore de Transformações

- `view_frames`
  - Gera uma descrição em `.pdf` da árvore de transformações

---

ros2 run tf2_tools view_frames

Árvore de Transformações



Consulta a uma Transformação

- `tf2_echo`
- Descrição da garra em relação à base:

---

ros2 run tf2_ros tf2_echo base_link tool_link

At time 1674760629.131840800

– Translation: [0.610, 0.000, 0.148]

– Rotation: in Quaternion [0.000, 0.000, 0.000, 1.000]

– Rotation: in RPY (radian) [0.000, –0.000, 0.000]

– Rotation: in RPY (degree) [0.000, –0.000, 0.000]

– Matrix:

1.000 0.000 0.000 0.610

0.000 1.000 0.000 0.000

0.000 0.000 1.000 0.148

0.000 0.000 0.000 1.000

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Consulta à `/tf` em C++

- Transformações podem ser obtidas de `/tf`
  - Existem funções para consultar diretamente as transformações, sem precissar assinar explicitamente o tópico
- Neste exemplo será convertida para o tipo `KDL::Frame`
  - Classe para representar transformação homogênea definida na KDL
  - Atributos são a matriz de rotação e vetor de posição
- Nodo `tf2_kdl` que mostra o *frame* na tela

Pacote eng10026_tf2

```
eng10026_tf2/
├─CMakeLists.txt
├─launch/
│ ├─display.launch.xml ......Não usado aqui
│ ├─display_abc.launch.xml  Não usado aqui
│ ├─publish_abc.launch.xml  Não usado aqui
│ └─publish_abcd.launch.xml  .. Não usado
│   aqui
├─package.xml
├─config/
│ └─eng10026_tf2.rviz
└─src/
  └─tf2_kdl.cpp
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Criação do Pacote

```
source ~/colcon_ws/install/setup.bash
cd ~/colcon_ws/src
ros2 pkg create --build-type ament_cmake --dependencies rclcpp
    tf2_ros orocos_kdl geometry_msgs tf2_geometry_msgs --node-
    name tf2_kdl eng10026_tf2
```

- `package.xml` deve ser editado para configurar os detalhes de documentação e incluir dependências
- Editar `CMakeLists.txt` para descomentar e ajustar as *tags* para compilação e instalação do pacote

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

package.xml

- Editar o arquivo `package.xml` para preenchimento dos meta-dados do pacote
  - Descrição
  - Mantenedor
  - Licença
  - Dependências
  - Exportações

---

cd eng10026_tf2

kate package.xml

CMakeLists.txt

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

```
add_executable(tf2_kdl
  src/tf2_kdl.cpp)

ament_target_dependencies(tf2_kdl
  rclcpp tf2_ros orocos_kdl geometry_msgs)

install(TARGETS tf2_kdl
  DESTINATION lib/${PROJECT_NAME})

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

tf2_kdl.cpp

```
#include <rclcpp/rclcpp.hpp>
#include <geometry_msgs/msg/transform_stamped.hpp>
#include <tf2_ros/buffer.h>
#include <tf2_ros/transform_listener.h>
#include <kdl/frames.hpp>
#include <kdl/frames_io.hpp>
```

tf2_kdl.cpp

```
int main(int argc,char* argv[])
{
  rclcpp::init(argc,argv);
  rclcpp::Node tf2Kdl("tk2_kdl");

  if(argc != 3)
  {
        RCLCPP_ERROR_STREAM(tf2Kdl.get_logger(),"Usage:
    tf2_kdl base_frame frame id\n");
        return -1;
  }

  tf2_ros::Buffer tfBuffer(tf2Kdl.get_clock());
  tf2_ros::TransformListener tfListener(tfBuffer);
```

tf2_kdl.cpp

```
rclcpp::Rate loop(10);
while(rclcpp::ok())
{
  geometry_msgs::msg::TransformStamped tfStamped;
  try
  {
    tfStamped=tfBuffer.lookupTransform(argv[1],argv[2],
      tf2::timeFromSec(0.0),tf2::durationFromSec(5.0)); //time–out
   is optional
  }
  catch(tf2::TransformException &ex)
  {
    RCLCPP_WARN_STREAM(tf2Kdl.get_logger(),ex.what());
    continue;
  }
```

tf2_kdl.cpp

```
KDL::Frame frame(
  KDL::Rotation::Quaternion(tfStamped.transform.rotation.x,
    tfStamped.transform.rotation.y,
    tfStamped.transform.rotation.z,
    tfStamped.transform.rotation.w),
  KDL::Vector(tfStamped.transform.translation.x,
    tfStamped.transform.translation.y,
    tfStamped.transform.translation.z));

RCLCPP_INFO_STREAM(tf2Kdl.get_logger(),"Frame:\n" <<
 frame << std::endl);
```

tf2_kdl.cpp

```
    rclcpp::spin_some(tf2Kdl.get_node_base_interface());
    loop.sleep();
  }
  return 0;
}
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Compilar

cd ~/colcon_ws

colcon build –-symlink–install

source ~/colcon_ws/install/setup.bash

- `colcon build` compila o pacote
- Deve ser sempre executado no diretório `~/colcon_ws`
- É recomendável reconfigurar o *workspace* após a compilação de pacotes

Execução

ros2 run eng10026_tf2 tf2_kdl base_link tool_link

[INFO] [1674789113.254013503] [tf2_kdl]: Frame:
[[1, 0, 0;
0, 1, 0;
0, 0, 1]
[0.61, 0, 0.1477]]

- Movendo os *sliders* para mover o robô a transformação homogênea modifica-se

Instalação do Pacote

- Clonar e compilar o repositório `eng10026_tf2`

```
cd ~/colcon_ws/src
git clone –b $ROS_DISTRO http://git.ece.ufrgs.br/eng10026/
    eng10026_tf2
cd ~/colcon_ws
colcon build ––symlink–install
source $HOME/colcon_ws/install/setup.bash
```

- Pacotes em código fonte devem ser colocados em `~/colcon_ws/src`
- `colcon build` compila o(s) pacote(s)
  - Deve ser sempre executado no diretório `~/colcon_ws`

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Exercício

- Criar um arquivo de *launch* que lance o robô no Rviz, a interface gráfica para movimentar o robô no Rviz e o nodo `tf2_kdl`

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Modelo Cinemático Usando a KDL

- Pode ser obtido direto da descrição em URDF
- Neste exemplo, será otido como um objeto do tipo `KDL::Frame`
  - Classe para representar transformação homogênea definida na KDL
  - Atributos são a matriz de rotação e vetor de posição
- Documentação em `<http://docs.orocos.org>`
- Será criado o nodo `fwdkin_kdl` que mostra o *frame* na tela
- Incluído no pacote `eng10026_fwdkin`

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ChainFkSolverPos_recursive

```
class ChainFkSolverPos_recursive:public ChainFkSolverPos
{
  public:
  ChainFkSolverPos_recursive(const Chain &chain);
  ~ChainFkSolverPos_recursive();

  int JntToCart(const JntArray &q_in,Frame &p_out,int segmentNr
      =-1);
  int JntToCart(const JntArray &q_in,std::vector<Frame> &p_out,int
      segmentNr=-1);
  void updateInternalDataStructures();
}
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Pacote `eng10026_fwdkin`

- Criar o pacote:

```
cd ~/colcon_ws/src
ros2 pkg create --build-type ament_cmake --dependencies
    rclcpp std_msgs sensor_msgs urdf kdl_parser orocos_kdl --
    node-name fwdkin_kdl eng10026_fwdkin
```

- `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` e `target_link_libraries`:

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

ament_target_dependencies(fwdkin_kdl
  rclcpp std_msgs sensor_msgs urdf kdl_parser orocos_kdl)

install(TARGETS fwdkin_kdl
  DESTINATION lib/${PROJECT_NAME})
```

Inclusão no Meta-Pacote

- O pacote `eng10026_fwdkin` será incluido no meta-pacote `eng10026`
- Editar o arquivo `package.xml` do pacote `eng10026` e incluir

<run_depend>eng10026_fwdkin**</run_depend>**

---

fwdkin_kdl.cpp

```
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/joint_state.hpp>
#include <std_msgs/msg/string.hpp>
#include <kdl/chainfksolverpos_recursive.hpp>
#include <kdl/frames.hpp>
#include <kdl/frames_io.hpp>
#include <kdl_parser/kdl_parser.hpp>

class FwdKinKdl: public rclcpp::Node
{
  public:
  FwdKinKdl(const std::string name);
  ~FwdKinKdl(void);
  KDL::Frame getFrame(void) {return frame_;};
```

fwdkin_kdl.cpp

```
  private:
  rclcpp::Subscription<sensor_msgs::msg::JointState>::SharedPtr
      jointStatesSubscriber_;

  KDL::Frame frame_;
  KDL::JntArray jointPositions_;
  std::string robotDescription_;
  KDL::Chain chain_;
  KDL::ChainFkSolverPos_recursive *fwdKinSolver_;

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

fwdkin_kdl.cpp

```
FwdKinKdl::FwdKinKdl(const std::string name): Node(name),
      jointPositions_(0)
{
  jointStatesSubscriber_=create_subscription<sensor_msgs::msg::
      JointState>("joint_states",100,std::bind(&FwdKinKdl::
      jointStatesCB,this,std::placeholders::_1));

  rclcpp::QoS qos(rclcpp::KeepLast(1));
  qos.transient_local();
  auto robotDescriptionSubscriber=create_subscription<std_msgs::
      msg::String>("robot_description",qos,std::bind(&FwdKinKdl::
      robotDescriptionCB,this,std::placeholders::_1));
```

fwdkin_kdl.cpp

```
while(robotDescription_.empty())
{
  RCLCPP_WARN_STREAM_SKIPFIRST_THROTTLE(
   get_logger(),*get_clock(),1000,"Waiting for robot model on /
   robot_description.");
  rclcpp::spin_some(get_node_base_interface());
}

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

fwdkin_kdl.cpp

```
  if (!tree.getChain("origin_link","tool_link",chain_))
    RCLCPP_ERROR_STREAM(get_logger(),"Failed to get chain
     from KDL tree.");

  // The solver does not copy the chain!
  fwdKinSolver_=new KDL::ChainFkSolverPos_recursive(chain_);

  int nJoints=chain_.getNrOfJoints();
  jointPositions_.resize(nJoints);
}

FwdKinKdl::~FwdKinKdl(void)
{
  delete fwdKinSolver_;
}
```

fwdkin_kdl.cpp

```
void FwdKinKdl::jointStatesCB(const sensor_msgs::msg::JointState::
      SharedPtr jointStates)
{
  for(unsigned int i=0;i < jointPositions_.rows();i++)
    jointPositions_(i)=jointStates->position[i];

  if(fwdKinSolver_->JntToCart(jointPositions_,frame_) < 0)
    RCLCPP_ERROR_STREAM(get_logger(),"Failed to compute
     forward kinematics.");
}

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

fwdkin_kdl.cpp

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

  FwdKinKdl fwdKinKdl("fwdkin_kdl");
  rclcpp::Rate loop(10);
  while(rclcpp::ok())
  {
    RCLCPP_INFO_STREAM(fwdKinKdl.get_logger(), "Frame:\n"
     << fwdKinKdl.getFrame() << std::endl);
    rclcpp::spin_some(fwdKinKdl.get_node_base_interface());
    loop.sleep();
  }
  return 0;
}
```

Instalação do Pacote

- Clonar e compilar o repositório `eng10026_fwdkin`

```
cd ~/colcon_ws/src
git clone –b $ROS_DISTRO  http://git.ece.ufrgs.br/eng10026/
      eng10026_fwdkin
cd ..
colcon build ––symlink–install
source $HOME/catkin_ws/devel/setup.bash
```

Execução

- Para gerar o `/joint_states`:

ros2 launch q2d_description display.launch.xml gui:=**true**

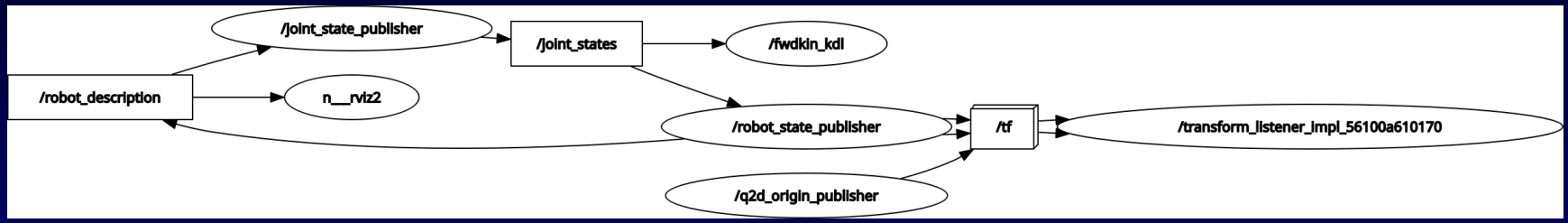
- Em outra janela:

ros2 run eng10026_fwdkin fwdkin_kdl

[INFO] [1674790321.004181630] [fwdkin_kdl]: Frame:
[[1, 0, 0;
0, 1, 0;
0, 0, 1]
[0.61, 0, 0.1477]]

- Movendo os *sliders* para mover o robô a transformação homogênea modifica-se correspondentemente

Gráfico de Computação



UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Exercício

- Criar um arquivo de *launch* que lance o robô no Rviz, a interface gráfica para movimentar o robô no Rviz e o nodo `fwdkin_kdl`