Cinemática Inversa no ROS 2

Walter Fetter Lages

fetter@ece.ufrgs.br

Universidade Federal do Rio Grande do Sul

Escola de Engenharia

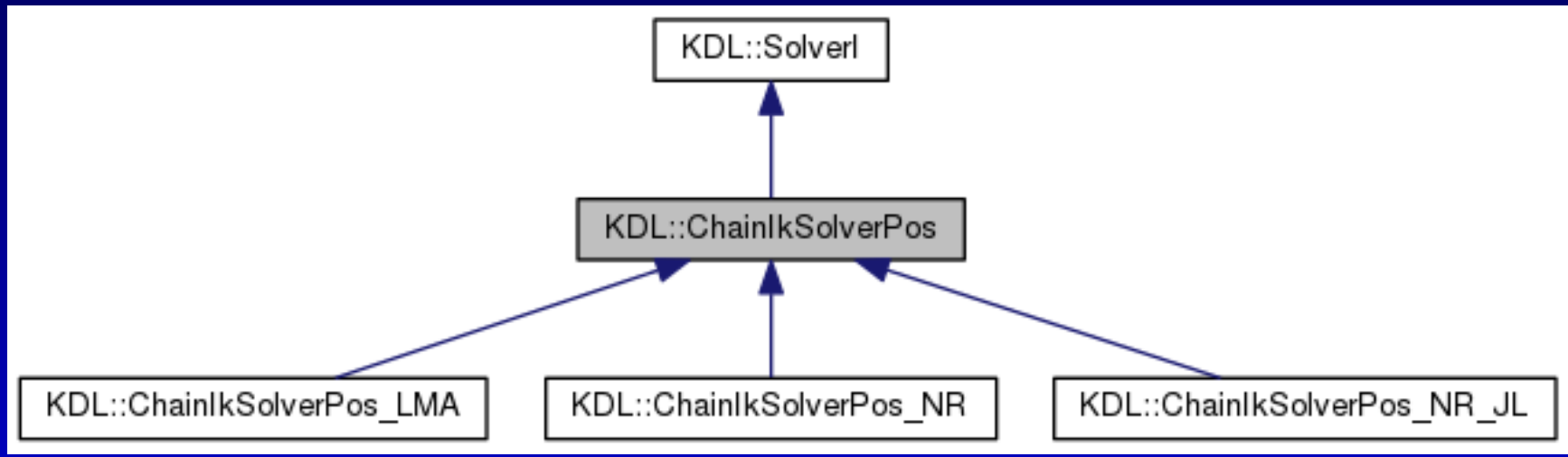
Departamento de Sistemas Elétricos de Automação e Energia

ENG10052 Laboratório de Robótica

Copyright (c) Walter Fetter Lages – p. 1

Cinemática Inversa no ROS

- Implementado no ROS utilizando a biblioteca KDL (*Kinematics and Dynamics Library*) do projeto OROCOS



Copyright (c) Walter Fetter Lages – p. 2

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Métodos Numéricos

- Método de Newton-Raphson
  - sem limites nas juntas:
    `KDL::ChainIkSolverPos_NR`
  - com limites nas juntas:
    `KDL::ChainIkSolverPos_NR_JL`
- Método de Levenberg-Marquardt (mínimos quadrados amortecido)
  - `KDL::ChainIKSolverPos_LMA`

Copyright (c) Walter Fetter Lages – p. 3

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Método de Newton-Raphson

- Usa a inversa do jacobiano

$$q[k+1] = q[k] + J^{-1}(q[k])(X - \text{fk}(q[k]))$$

- Quando J não é quadrada é usada a inversa generalizaa $J^{\dagger}$
  - Calculada por decomposição em valores singulares
    - Seja $J \in \mathbb{R}^{m \times n}$ tal que $J = USV^T$
    - $U \in \mathbb{R}^{m \times m}$, $S \in \mathbb{R}^{m \times n}$ é uma matriz diagonal e $V \in \mathbb{R}^{n \times n}$
    - A pseudoinversa é $J^{\dagger} = VS^{\dagger}U^T$
    - $S^{\dagger}$ é computada subtituindo cada elemento diferente de zero da diagonal de S pelo seu recíproco e transpondo a matrix resultante

Copyright (c) Walter Fetter Lages – p. 4

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

KDL::ChainIkSolverPos_NR

```
#include <chainiksolverpos_nr.hpp>

class ChainIkSolverPos_NR : public ChainIkSolverPos
{
  public:
  ChainIkSolverPos_NR(const Chain &chain,
    ChainFkSolverPos &fksolver,
    ChainIkSolverVel &iksolver,
    unsigned int maxiter=100,double eps=1e-6);

  ~ChainIkSolverPos_NR();
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

KDL::ChainIkSolverPos_NR

```
  virtual int CartToJnt(const JntArray &q_init,
    const Frame &p_in, JntArray &q_out);

  virtual const char* strError(const int error) const;

  virtual void updateInternalDataStructures();
};
```

Copyright (c) Walter Fetter Lages – p. 6

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

KDL::ChainIkSolverPos_NR_JL

```
#include <chainiksolverpos_nr_jl.hpp>

class ChainIkSolverPos_NR_JL : public ChainIkSolverPos
{
  public:
  ChainIkSolverPos_NR_JL(const Chain &chain,
    const JntArray &q_min,const JntArray &q_max,
    ChainFkSolverPos &fksolver,
    ChainIkSolverVel &iksolver,
    unsigned int maxiter=100,double eps=1e-6);

  ChainIkSolverPos_NR_JL(const Chain &chain,
    ChainFkSolverPos &fksolver,
    ChainIkSolverVel &iksolver,
    unsigned int maxiter=100,double eps=1e-6);
```

Copyright (c) Walter Fetter Lages – p. 7

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

KDL::ChainIkSolverPos_NR_JL

```
~ChainIkSolverPos_NR_JL();

virtual int CartToJnt(const JntArray &q_init,
  const Frame &p_in,JntArray &q_out);

int setJointLimits(const JntArray &q_min,
  const JntArray &q_max);

virtual void updateInternalDataStructures();

const char* strError(const int error) const;
};
```

- O construtor sem os limites das juntas, considera os limites para um `double`

Copyright (c) Walter Fetter Lages – p. 8

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Algoritmo de Levenberg-Marquardt

- Problema de otimização:

$$q = \arg\min_q \left(X - \text{fk}(q)\right)^T \left(X - \text{fk}(q)\right)$$

- A cada iteração $\text{fk}(q)$ é substituido por $\text{fk}(q+\delta)$
- δ é determinado pela linearização de $\text{fk}(q+\delta)$:

$$\text{fk}(q+\delta) \approx \text{fk}(q) + J(q)\delta$$

- O custo a minimizar é:

$$S(q+\delta) \approx \left(X - \text{fk}(q) - J(q)\delta\right)^T \left(X - \text{fk}(q) - J(q)\delta\right)$$

Copyright (c) Walter Fetter Lages – p. 9

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Minimização do Custo

- Derivado em relação a δ e igualado a zero resulta:

$$-2J^T(q)\left(X - \text{fk}(q) - J(q)\delta\right) = 0$$
$$\left(J^T(q)J(q)\right)\delta = J^T(q)\left(X - \text{fk}(q)\right)$$

- sistema linear de onde δ pode ser determinado
- Contribuição de Levenberg: versão amortecida:

$$\left(J^T(q)J(q) + \lambda I\right)\delta = J^T(q)\left(X - \text{fk}(q)\right)$$

- Marquardt: substituir I por $\text{diag}\left(J^T(q)J(q)\right)$:

$$\left(J^T(q)J(q) + \lambda\,\text{diag}\left(J^T(q)J(q)\right)\right)\delta = J^T(q)\left(X - \text{fk}(q)\right)$$

Copyright (c) Walter Fetter Lages – p. 10

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

KDL::ChainIKSolverPos_LMA

```
#include <chainiksolverpos_lma.hpp>

class ChainIkSolverPos_LMA : public KDL::ChainIkSolverPos
{
  public:
  ChainIkSolverPos_LMA(const KDL::Chain &_chain,
    const Eigen::Matrix<double,6,1>& _L,
    double _eps=1E-5,int _maxiter=500,
    double _eps_joints=1E-15);

  ChainIkSolverPos_LMA(const KDL::Chain &_chain,
    double _eps=1E-5,int _maxiter=500,
    double _eps_joints=1E-15);
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

KDL::ChainIKSolverPos_LMA

```
virtual int CartToJnt(const KDL::JntArray &q_init,
  const KDL::Frame &T_base_goal,
  KDL::JntArray &q_out);

virtual ~ChainIkSolverPos_LMA();

void updateInternalDataStructures();

virtual const char* strError(const int error) const;
};
```

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

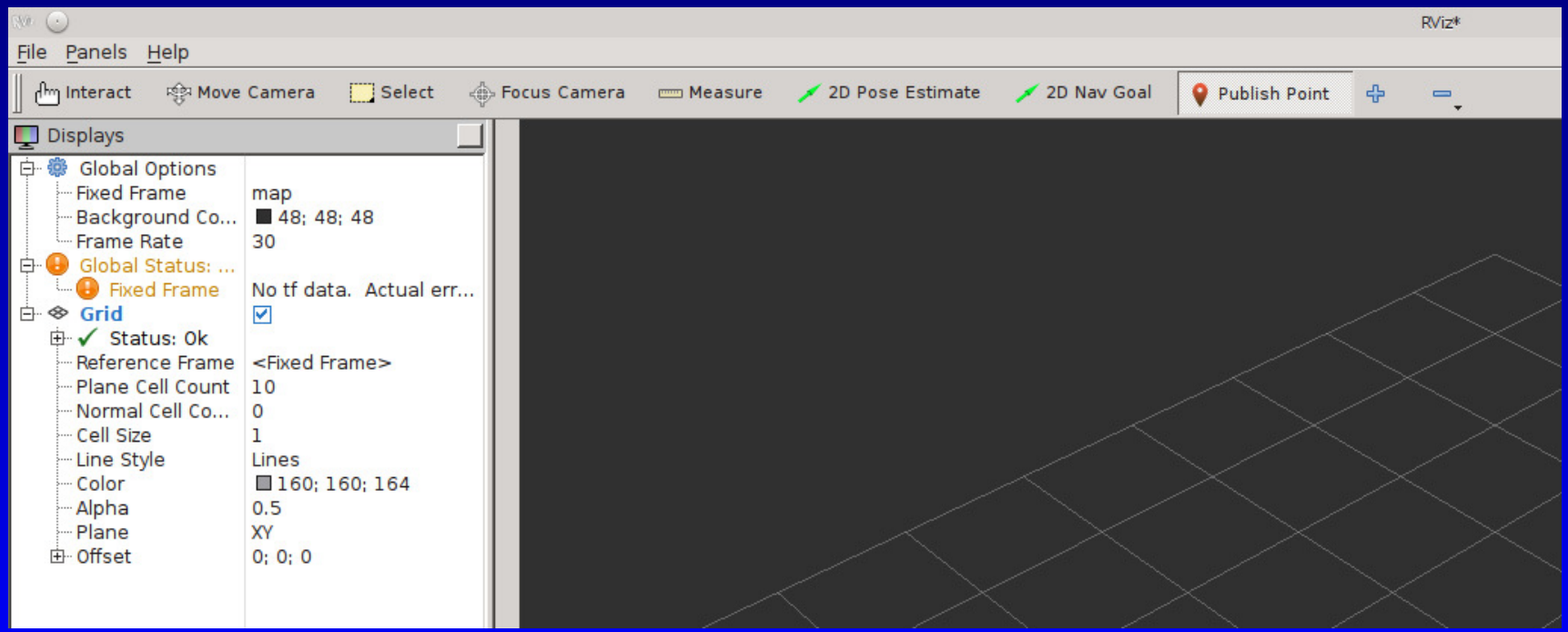
Exemplos

- Nodo para obter as variáveis de junta para a posicionar a flange do robô em um ponto especificado através de um *click* no RViz
- Com as coordenadas cartesianas do ponto desejado pode-se usar o modelo cinemático inverso para calcular as variáveis de junta correspondentes

Copyright (c) Walter Fetter Lages – p. 13

Exemplo Rviz

- O Rviz publica no tópico `/clicked_point` as coordenadas do *click* do mouse
- Ferramenta Publish Point
  - Só funciona com pontos do *grid*
  - Reduzir o tamanho da célula para 0.001
  - Aumentar o número de células para 1000



Copyright (c) Walter Fetter Lages – p. 14

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Tópico /clicked_point

- Verificar o tipo da mensagem

rviz2 &

ros2 topic info /clicked_point

- Verificar a descrição da mensagem

ros2 interface show geometry_msgs/msg/PointStamped

Copyright (c) Walter Fetter Lages – p. 15

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

geometry_msgs/msg/PointStamped

---

std_msgs/Header header

    builtin_interfaces/Time stamp

        int32 sec

        uint32 nanosec

    string frame_id

Point point

    float64 x

    float64 y

    float64 z

Copyright (c) Walter Fetter Lages – p. 16

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Verificação com `rostopic`

- Pode-se verificar o que é publicado nos tópicos com o comando

---

ros2 topic echo /clicked_point

- Será criado um pacote para assinar o tópico `/clicked_point`, computar o modelo cinemático inverso e publicar as referências para os controladores de junta
- Se não for usado o simulador Gazebo, e desejar-se apenas vizualizar o robô é possível publicar diretamente as variáveis de junta no tópico `/joint_state`

Copyright (c) Walter Fetter Lages – p. 17

Criação do Pacote

- Criar o pacote:

```
cd ~/colcon_ws/src/q2d
ros2 pkg create --build-type ament_cmake --dependencies
    rclcpp geometry_msgs sensor_msgs std_msgs kdl_parser
    urdf orocos_kdl --node-name q2d_teleop_rviz
    q2d_teleop
```

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

Copyright (c) Walter Fetter Lages – p. 18

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

CMakeLists.txt

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

```
add_executable(q2d_teleop_rviz src/q2d_teleop_rviz.cpp)

ament_target_dependencies(q2d_teleop_rviz
  rclcpp geometry_msgs sensor_msgs std_msgs kdl_parser urdf
    orocos_kdl)

install(TARGETS q2d_teleop_rviz
  DESTINATION lib/${PROJECT_NAME})

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

Copyright (c) Walter Fetter Lages – p. 19

Código Fonte do Nodo

- Editar o arquivo `q2d_teleop_rviz.cpp` com o código fonte no diretório `src`
- Compilar com os comandos:

---

cd ~/colcon_ws

colcon build −−symlink−install

Copyright (c) Walter Fetter Lages – p. 20

Código Fonte do Nodo

```
#include <rclcpp/rclcpp.hpp>
#include <geometry_msgs/msg/point_stamped.hpp>
#include <std_msgs/msg/float64.hpp>
#include <std_msgs/msg/string.hpp>
#include <kdl/chainiksolverpos_lma.hpp>
#include <kdl_parser/kdl_parser.hpp>

#define sqr(x) ((x)*(x))

class Q2dTeleop: public rclcpp::Node
{
  public:
  Q2dTeleop(void);
  ~Q2dTeleop(void);
  void publish(void);
```

Copyright (c) Walter Fetter Lages – p. 21

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Código Fonte do Nodo

```
private:
rclcpp::Subscription<geometry_msgs::msg::PointStamped>::
    SharedPtr clickSub_;
rclcpp::Publisher<std_msgs::msg::Float64>::SharedPtr
    shoulderCmdPub_;
rclcpp::Publisher<std_msgs::msg::Float64>::SharedPtr
    elbowCmdPub_;

KDL::Frame goal_;

std::string robotDescription_;
KDL::Chain chain_;
KDL::ChainIkSolverPos_LMA *ikSolverPos_;
KDL::JntArray q_;
```

Copyright (c) Walter Fetter Lages – p. 22

Código Fonte do Nodo

```
  void clickCB(const geometry_msgs::msg::PointStamped::SharedPtr
      click);
  void robotDescriptionCB(const std_msgs::msg::String::SharedPtr
      robotDescription);
};
```

Copyright (c) Walter Fetter Lages – p. 23

Código Fonte do Nodo

```
Q2dTeleop::Q2dTeleop(void): Node("Q2d_teleop_rviz"), q_(2)
{
  using std::placeholders::_1;
  clickSub_=create_subscription<geometry_msgs::msg::PointStamped
    >("clicked_point",100,std::bind(&Q2dTeleop::clickCB,this,_1));
  shoulderCmdPub_=create_publisher<std_msgs::msg::Float64>("
    shoulder_controller/command",100);
  elbowCmdPub_=create_publisher<std_msgs::msg::Float64>("
    elbow_controller/command",100);

  rclcpp::QoS qos(rclcpp::KeepLast(1));
  qos.transient_local();
  auto robotDescriptionSubscriber_=create_subscription<std_msgs::
    msg::String>("robot_description",qos,std::bind(&Q2dTeleop::
    robotDescriptionCB,this,_1));
```

Copyright (c) Walter Fetter Lages – p. 24

Código Fonte do Nodo

```
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.");

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

Copyright (c) Walter Fetter Lages – p. 25

Código Fonte do Nodo

```
  q_.resize(chain_.getNrOfJoints());

  goal_.Identity();
  goal_.p.x(0.61);
  goal_.p.z(0.1477);

  Eigen::Matrix<double,6,1> L;
  L << 1.0 , 1.0 , 1.0, 0.01, 0.01, 0.01;

  // A copy of chain_ is not created inside!
  ikSolverPos_=new KDL::ChainIkSolverPos_LMA(chain_,L);
  ikSolverPos_->display_information=false;
}
```

Código Fonte do Nodo

```
Q2dTeleop::~Q2dTeleop(void)
{
  delete ikSolverPos_;
}

void Q2dTeleop::clickCB(const geometry_msgs::msg::PointStamped::
      SharedPtr click)
{
  goal_.p.x(click->point.x);
  goal_.p.y(click->point.y);
  goal_.p.z(0.1477);
  goal_.M.RotZ(atan2(click->point.y,click->point.x));

  publish();
}
```

Copyright (c) Walter Fetter Lages – p. 27

Código Fonte do Nodo

```
void Q2dTeleop::robotDescriptionCB(const std_msgs::msg::String::
        SharedPtr robotDescription)
{
        robotDescription_=robotDescription->data;
}
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Código Fonte do Nodo

```
#define KDL_IK
void Q2dTeleop::publish(void)
{
#ifdef KDL_IK
   KDL::JntArray q_out=q_;
   int error=0;
   if((error=ikSolverPos_->CartToJnt(q_,goal_,q_out)) < 0)
      RCLCPP_ERROR_STREAM(get_logger(),"Failed to compute
       invere kinematics: (" << error << ") "
            << ikSolverPos_->strError(error));
   q_=q_out;
#else
```

Código Fonte do Nodo

```
// Algebric inverse kinematics
    const double l1=0.343;
    const double l2=0.267;
    double c1=(sqr(goal_.p.x())+sqr(goal_.p.y())−sqr(l1)−sqr(l2))/2/
    l1/l2;
    if(c1 >= 0.0 && c1 <= 1.0)
    {
        double s1a=sqrt(1−sqr(c1));
        double s1b=−s1a;
        double q1a=atan2(s1a,c1);
        double q1b=atan2(s1b,c1);

        double q0a=atan2(goal_.p.y(),goal_.p.x())−atan2(l2∗s1a,l1+
    l2∗c1);
        double q0b=atan2(goal_.p.y(),goal_.p.x())−atan2(l2∗s1b,l1
    +l2∗c1);
```

Copyright (c) Walter Fetter Lages – p. 30

Código Fonte do Nodo

```
            if((fabs(q_(0)−q0a) < fabs(q_(0)−q0b)) || (cos(q0a) > 0.0))
            {
                  q_(0)=q0a;
                  q_(1)=q1a;
            }
            else
            {
                  q_(0)=q0b;
                  q_(1)=q1b;
            }
      }
#endif
```

Copyright (c) Walter Fetter Lages – p. 31

Código Fonte do Nodo

```
  std_msgs::msg::Float64 shoulderCmd;
  std_msgs::msg::Float64 elbowCmd;

  shoulderCmd.data=q_(0);
  elbowCmd.data=q_(1);

  shoulderCmdPub_->publish(shoulderCmd);
  elbowCmdPub_->publish(elbowCmd);
}
```

Copyright (c) Walter Fetter Lages – p. 32

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Código Fonte do Nodo

```
int main(int argc,char* argv[])
{
  rclcpp::init(argc,argv);
  rclcpp::spin(std::make_shared<Q2dTeleop>());
  rclcpp::shutdown();
  return 0;
}
```

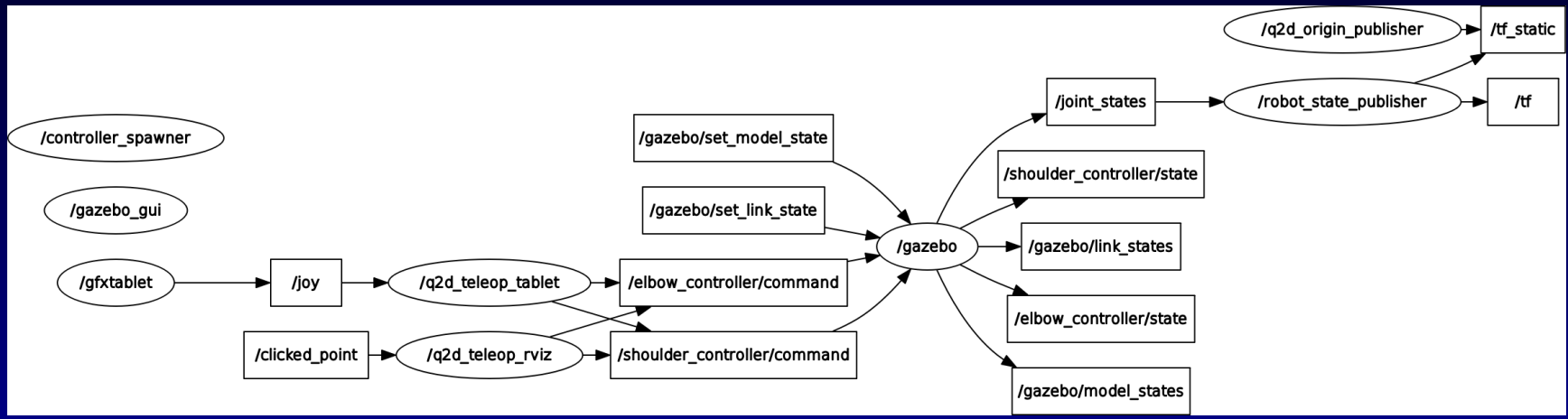
UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

Execução

ros2 launch q2d_teleop gazebo.launch.xml

Copyright (c) Walter Fetter Lages – p. 34

Gráfico de Computação

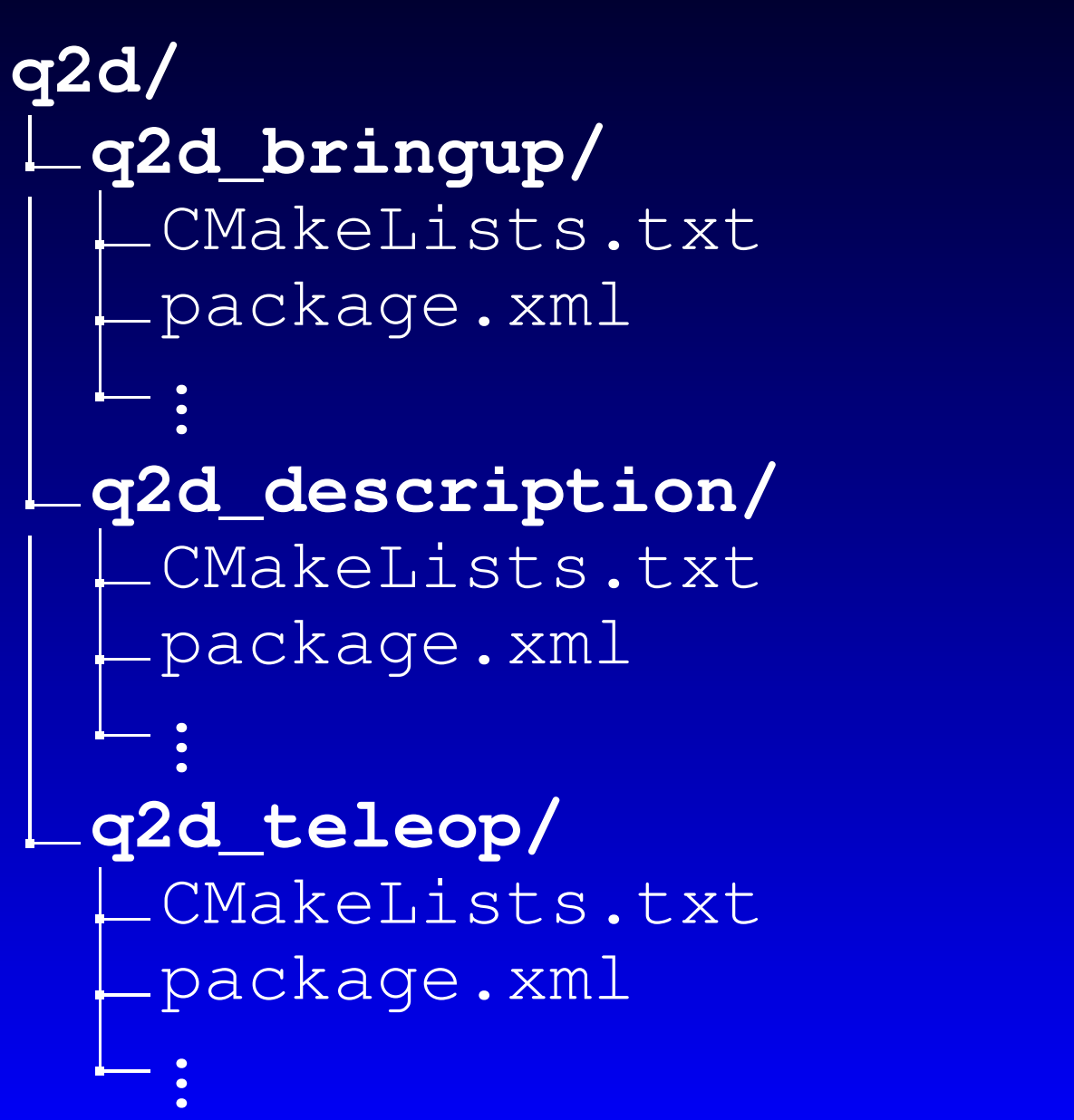


Instalação do Pacote

- Clonar e compilar o repositório `q2d`

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

Repositório q2d



Copyright (c) Walter Fetter Lages – p. 37