RRM

Priama kinematika

Marek Čornák
marek.cornak@stuba.sk

Michal Dobiš
michal.dobis@stuba.sk

Jakub Ivan
jakub.ivan@stuba.sk

Obsah

1. Priama kinematika v C++
2. ROS TF
3. Launch file

Aplikovanie DH parametrov

Pre každý riadok tabuľky DH parametrov vytvoríme príslušnú homogénnu transformáciu:

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_n^0 = A_1 A_2 \dots A_n$$

Priama kinematika v C++

- Budeme programovať 2D polia
- naprogramujeme si násobenie matíc a ďalšie maticové operácie ?!

Eigen

C++ knižnica pre maticové operácie a prácu s transformáciami
Základy sme pre vás pripravili v balíku **eigen_example**
Ako pridať Eigen do vášho programu?

CmakeList.txt
*find_package(**Eigen3** REQUIRED)*
*include_directories(**${EIGEN3_INCLUDE_DIRS}**)*

Eigen3 je takzvaná “header-only” knižnica - nemusíte ju linkovať

Eigen

Pozor *include_directories(**${EIGEN3_INCLUDE_DIRS}**)* vloží Eigen do všetkých *targetov* (node). Ak chcete špecifikovať je ideálne vložiť do *target_include_directories* napr. pre logger_node ako:

target_include_directories(logger_node PUBLIC
$<BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include>
$<INSTALL_INTERFACE:include/${PROJECT_NAME}>
***${EIGEN3_INCLUDE_DIRS}**)*

Knižnicu do svojho zdrojového kódu vložíme ako
#include <Eigen/Geometry>

Eigen vektor

Inicializácia vektora po prvkoch

```
Eigen::VectorXd v1(3);
v1 << 1, 2, 3;
```

Inicializácia nulového vektora

```
Eigen::VectorXd v2 = Eigen::VectorXd::Zero(3);
```

Špecifický počet prvkov

```
Eigen::Vector3d v3; Eigen::Vector4d v4;
```

Eigen matica

Inicializácia 4x4 matice po prvkoch

```
Eigen::MatrixXd m1(3,3);
m1 <<  1, 2, 3, 4, 5, 6, 7, 8, 9;
```

Vytvorenie identickej matice

```
Eigen::MatrixXd m2 = Eigen::MatrixXd::Identity(3,3);
Eigen::Matrix4d m3 = Eigen::Matrix4d::Identity();
```

Vytvorenie nulovej matice

```
Eigen::MatrixXd m4 = Eigen::MatrixXd::Zero(3,3);
Eigen::Matrix4d m5 = Eigen::Matrix4d::Zero();
```

Eigen základné maticové operácie

Súčet

```
auto m = m1 + m2;
```

Násobenie

```
auto m = m1 * v1;
```

Inverzia matice

```
auto m = m1.inverse();
```

Skalárny súčin

```
auto m = v1.dot(v2);
```

Vektorový súčin

```
auto m = v1.cross(v2);
```

Eigen - example

Example balík je dostupný na dokumentovom serveri

Po kompilácii ho môžete pustiť príkazom:

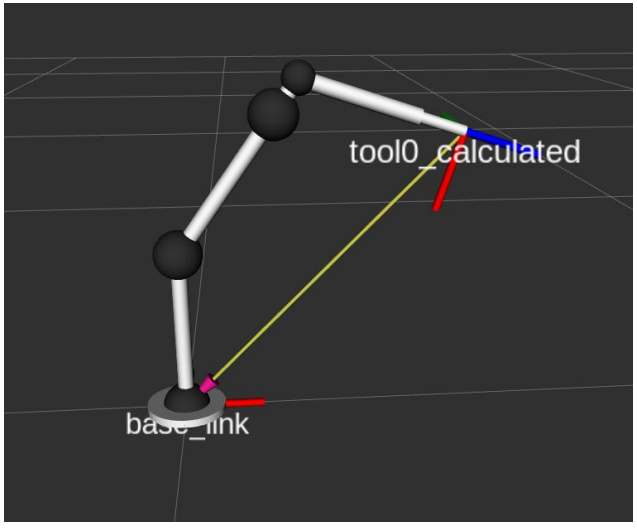
ros2 run eigen_example eigen_example

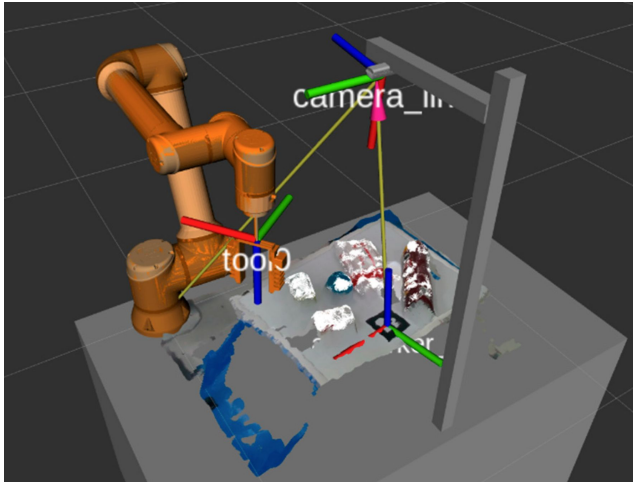
TF broadcaster

- Vysiela transformácie
- Kinematicke štruktúry robota
- Data so snímačov vztiahnuté na transformácie

Tutoriál

http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20broadcaster%20%28C%2B%2B%29





TF broadcaster

- Pridať závislosť na catkin balík **tf2, tf2_ros** do CMakeList.txt a package.xml
- Hlavičkový súbor

```
#include <tf2_ros/transform_broadcaster.h>
```

- Objekt TransformBroadcaster - vhodné deklarovať ako **private**

```
tf2_ros::TransformBroadcaster tf_broadcaster_;
```

- Pozor je potrebné inicializovať v konštruktore

```
constructor() : tf_broadcaster_(this)
```

- Publikovanie

```
tf_broadcaster_.sendTransform(transform);
```

TF transform

Hlavička správy (stamp, parent & child link):

```
geometry_msgs::msg::TransformStamped transform;
transform.header.stamp = this->get_clock()->now();
transform.header.frame_id = "base_link";
transform.child_frame_id = "tool0_calculated";
```

Poloha

```
transform.transform.translation.x = 0.0;
transform.transform.translation.y = 0.0;
transform.transform.translation.z = 0.5;
```

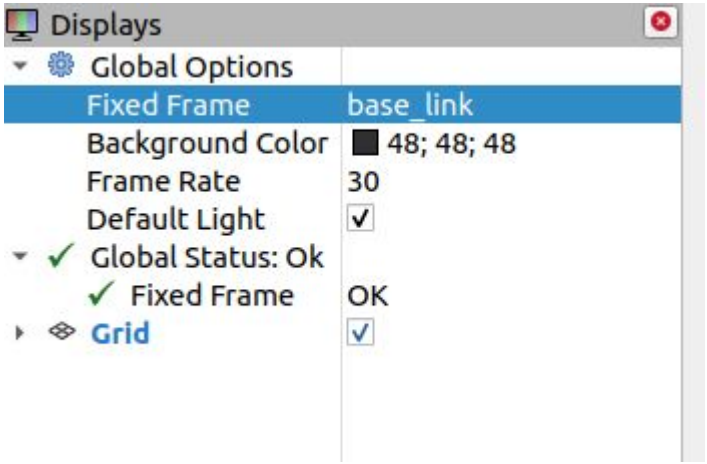
Rotácia

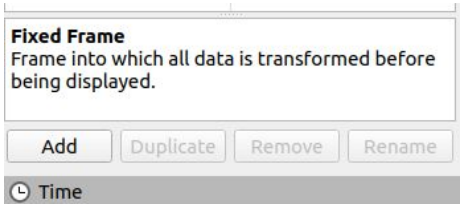
```
#include <tf2/LinearMath/Quaternion.h>


tf2::Quaternion q;

q.setRPY(0, 0, 1.57);
transform.transform.rotation.x = q.x();
transform.transform.rotation.y = q.y();
transform.transform.rotation.z = q.z();
transform.transform.rotation.w = q.w();
```

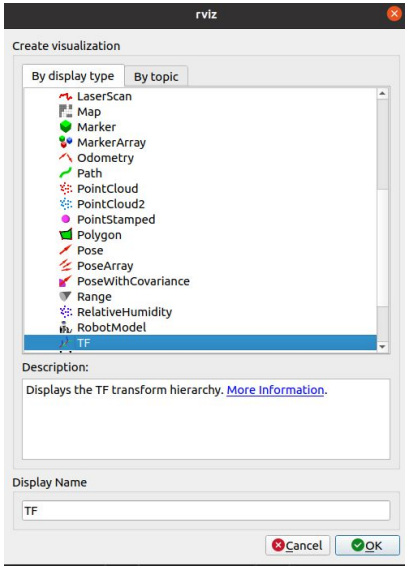
Vizualizácia v Rviz2

Vyberte Fixed Frame *base_link*

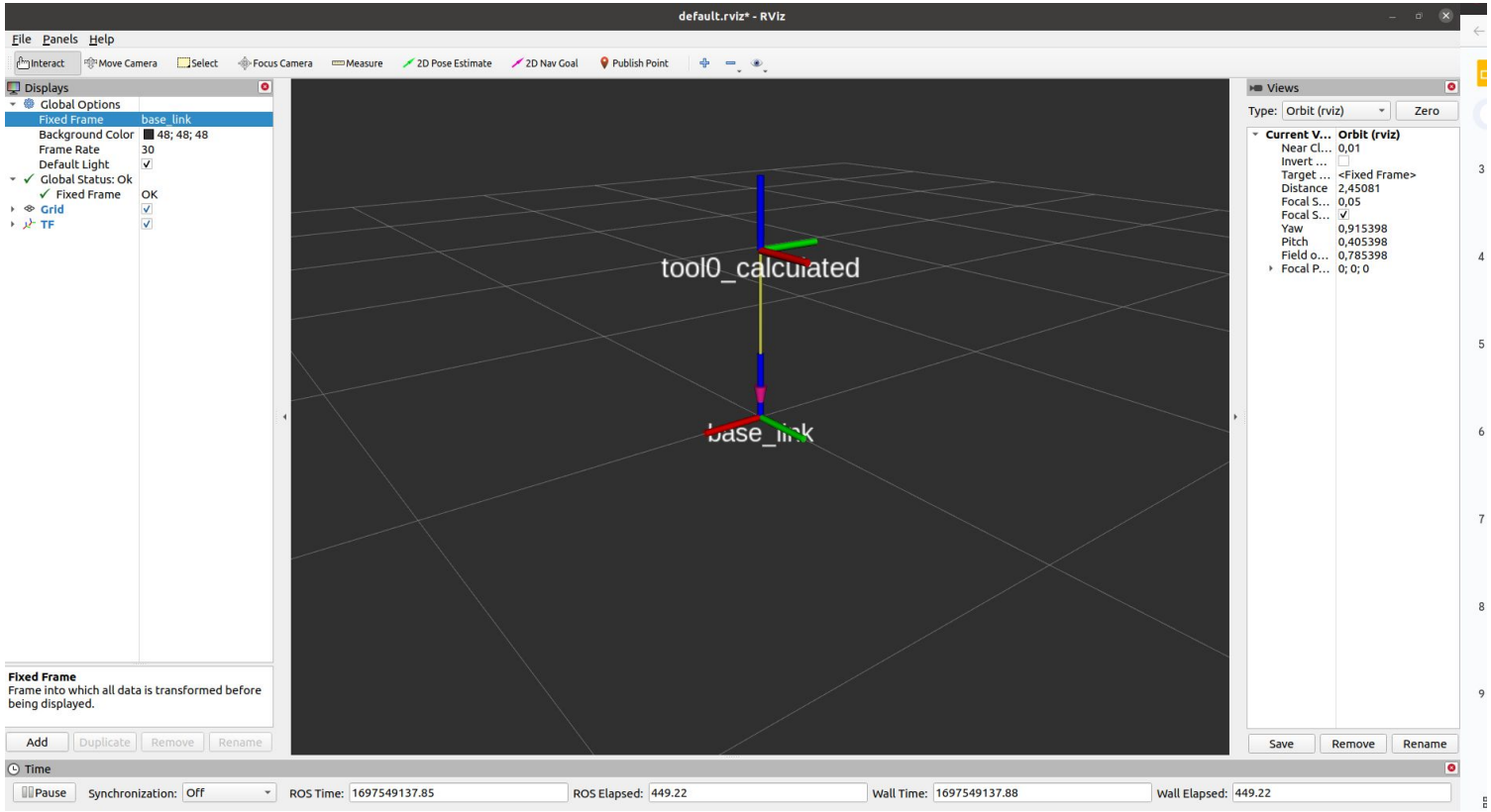
a tlačidlom *Add* pridajte *TF*







Vizualizácia v Rviz



Konverzia Eigen do TF

```
  Eigen::Matrix4d matrix = Eigen::Matrix4d::Identity();
transform.transform.translation.x = matrix(0, 3);
transform.transform.translation.y = matrix(1, 3);
transform.transform.translation.z = matrix(2, 3);
Eigen::Quaterniond q(matrix.block<3, 3>(0, 0));
transform.transform.rotation.x = q.x();
transform.transform.rotation.y = q.y();
transform.transform.rotation.z = q.z();
transform.transform.rotation.w = q.w();
```

ROS launch

- Súbor príkazov ako štartovať aplikáciu
- Umožňuje štartovať viac node naraz
- Umožňuje jednoduché načítanie parametrov
- podpora v jazyku **Python** alebo **XML**
- názov súboru dodržiava konvenciu *nazov.launch.xml* alebo *nazov.launch.py*

</node> - definuje node, ktorá sa bude púšťať
</launch> - vkladá iný *launch* súbor
</param> - definuje parameter
</arg> - argument daného súboru

https://docs.ros.org/en/foxy/How-To-Guides/Launch-file-different-formats.html

ROS launch xml

```
<launch>
    <node pkg="rrm_sim" exec="robot_sim_node" name="rrm_sim" output="screen"/>
    <include file="$(find-pkg-share rrm_simple_robot_model)/launch/robot_state_publisher.launch.xml"/>
</launch>
```

ROS launch

Nezabudnite zadať príkaz na inštaláciu v CMakeList.txt

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