

Обзор библиотеки `ros tf`

Денис Шепелев

Как работает `tf`

Задачи и
структура `tf`
`Listener` и
`Broadcaster`

Работа с `tf`

`Data`
`Broadcaster`
`Listener`

Инструменты
командной строки

Инструменты
командной
строки

Возможные
проблемы

Возможные
проблемы

Задачи и структура tf

Обзор
библиотеки `ros tf`

Денис Шепелев

Как работает tf

**Задачи и
структура tf**
Listener и
Broadcaster

Работа с tf

Data
Broadcaster
Listener

Инструменты
командной строки

Инструменты
командной
строки

Возможные
проблемы

Возможные
проблемы

Основные задачи tf

- ▶ обеспечить надежную и эффективную работу в распределенной системе
- ▶ простоту изменения и получения данных

Задачи и структура tf

Обзор
библиотеки `ros tf`

Денис Шепелев

Как работает tf

Задачи и
структура tf
Listener и
Broadcaster

Работа с tf

Data
Broadcaster
Listener

Инструменты
командной строки

Инструменты
командной
строки

Возможные
проблемы

Возможные
проблемы

- ▶ Основная структура - граф
- ▶ Вершина - система координат (frame)
- ▶ Ребро - преобразование (transform) между соединенными этим ребром фреймами

Никаких явных ограничений на граф нет, однако рекомендуется

- ▶ чтобы граф был связный
- ▶ и ациклический

Задачи и структура tf

Обзор
библиотеки `ros tf`

Денис Шепелев

Как работает `tf`

Задачи и
структура `tf`
`Listener` и
`Broadcaster`

Работа с `tf`

`Data`
`Broadcaster`
`Listener`

Инструменты
командной строки

Инструменты
командной
строки

Возможные
проблемы

Возможные
проблемы

Типичный запрос в `tf` выглядит следующим образом:

- ▶ Получить преобразование `transform` между `parent` и `child` фреймами в момент времени `t`

Поэтому для каждого ребра в хронологическом порядке хранится история его изменений в течение некоторого промежутка времени.

Основные компоненты для работы с tf

- ▶ Broadcaster - периодически отправлять данные в tf
- ▶ Listener - собирает данные, которые приходят в tf, и работает с запросами

Следует отметить, что Listener разрешает запросы используя интерполяцию (SLERP = spherical linear interpolation). Поэтому

- ▶ необходимо обеспечить достаточную быструю частоту публикаций данных
- ▶ с учетом пропускной способности сети

Благодаря интерполяции система может работать асинхронно, вдобавок повышая устойчивость в случае потери данных.

Механизм разрешения запроса

- ▶ Listener гуляет по графу, формируя остовное дерево содержащее child и parent
- ▶ Если parent не найдено, будет получено соответствующее исключение
- ▶ Если остовное дерево сформировано, то вдоль ребер, лежащих на пути между child и parent, ищется преобразование между целевыми фреймами
- ▶ На каждом ребре ищется интерполяция в момент времени, заданный в запросе, используя 2 ближайших по времени преобразования. Если все они существуют, то между фреймами a и c , между которыми в графе лежит фрейм b , преобразование может быть найдено по формуле

$$T_a^c = T_a^b * T_b^c$$

Основные структуры, которыми оперирует `tf`

- ▶ `tf::Quaternion`
- ▶ `tf::Vector3`
- ▶ `tf::Point`
- ▶ `tf::Pose`
- ▶ `tf::Transform`
- ▶ `tf::Stamped <T>`
- ▶ `tf::StampedTransform`

Стоит отдельно отметить полезную функцию
`tf::Quaternion createQuaternionFromRPY(double roll, double pitch, double yaw)`

Для того чтобы отправить преобразование достаточно сделать следующее

- ▶ `tf::TransformBroadcaster()`
- ▶ `void sendTransform(const StampedTransform & transform)` - отправляет преобразование

Как работает `tf`

Задачи и
структура `tf`
`Listener` и
`Broadcaster`

Работа с `tf`

`Data`
`Broadcaster`
`Listener`

Инструменты
командной строки

Инструменты
командной
строки

Возможные
проблемы

Возможные
проблемы

И наконец, для того чтобы можно было посылать запросы системе используется

- ▶ `tf::TransformListener`
- ▶ `bool canTransform(...)` - проверяет, может ли преобразование быть найдено
- ▶ `bool waitForTransform(...)` - блокирует исполнение, на некоторый промежуток времени, пока преобразование не будет найдено, или не закончится время
- ▶ `void lookupTransform(...)` - находит преобразование, в случае неудачи выбрасывает исключение

Как работает `tf`

Задачи и
структура `tf`
`Listener` и
`Broadcaster`

Работа с `tf`

`Data`
`Broadcaster`
`Listener`

Инструменты
командной строки

Инструменты
командной
строки

Возможные
проблемы

Возможные
проблемы

Также `Listener` предоставляет возможность преобразовывать данные между фреймами, такие как `Quaternion`, `Vector3`, `Pose` и т.д.

- ▶ `void tf::TransformListener::transformDATATYPE(...)`

Некоторые методы выбрасывают исключения, которые наследуются от `tf::TransformException`

- ▶ `tf::ConnectivityException` - если 2 фрейма не принадлежат одному связанному графу
- ▶ `tf::ExtrapolationException` - если между 2 фреймами одно и более устаревших связей
- ▶ `tf::InvalidArgument` - при неправильных аргументах
- ▶ `tf::LookupException` - если в запросе один из фреймов не существует

Основные инструменты командной строки

- ▶ `view_frames` - визуализирует в pdf структуру графа `tf`
- ▶ `tf_monitor` - мониторит преобразования между фреймами (всеми, либо 2 заданными)
- ▶ `tf_echo` - выводит преобразование между 2 заданными фреймами
- ▶ `roswtf` - клевая тулза которая позволяет найти много разных проблем в `ros`, в том числе и `tf`, например, выведет предупреждение о том, что граф не связный
- ▶ `static_transform_publisher` - позволяет публиковать статическое преобразование

Возможные проблемы при работе с tf

- ▶ **Проблема** - Некоторые ноды ros могут публиковать в tf противоречивую информацию (циклы в графе, несвязный граф).
Решение - Искать баг с помощью инструментов командной строки, в самом начале разработки внимательно следить и регулировать кто что публикует.
- ▶ **Проблема** - tf ругается на устаревшие данные.
Решение - Возможно вы запустили rosbag play и забыли прописать `--clock` или `/use_sim_time = true` в rosbagm.
- ▶ **Проблема** - Сообщения приходят быстрее чем, обновляется информация в tf.
Решение - Использовать `waitForTransform` или `tf::MessageFilter`



TF

<http://wiki.ros.org/tf>