Introduction

Robot FI Tool to inject faults into any robot please follow this repository

Package Structure

this section briefly explains the structure of the package and explains customizationa options and some important folders.

- src
  - robot_fi_tool: this package containes the fault injection module
    - config: configuration files
    - launch: launch files
    - msg: custom message
    - src: source code
    - utils: code for FI experiment

Quick Start

Requirements

- `sudo apt-get install qt5-default`
- `pip3 install -r requirements.txt`

Install

- add required robot modules and its ROS support packages

Controller setup

we can use a moveit C++ robot controller this guide shows how to setup any robot controller to work with robot fault injector pakcage

- remap `joint_states` topic to `joint_states_fake`

```
<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
<node name="joint_state_publisher" type="joint_state_publisher" pkg="joint_state_publisher">
  <rosparam param="source_list">[franka_state_controller/joint_states, franka_gripper/joint_states] </rosparam>
  <remap from="joint_states" to="joint_states_fake"/>
  <param name="rate" value="30"/>
</node>
```

- Create publisher and subsriber use the same topic as shown in the below figure.

```
ros::Publisher goal_pub = nh.advertise<std_msgs::Bool>("goal_state", 1000);
ros::Publisher pose_state_pub = nh.advertise<std_msgs::Int32>("pose_state", 1000);
ros::Publisher iterations = nh.advertise<std_msgs::Int32>("iterations", 1000);
```

- publish iterations: since the robot perform multiple iterations of a similar movement

```
float pick_x[7] = {0.23, 0.376, 0.23, 0.232, 0.375, 0.22, 0.375};
float pick_y[7] = {0.28, 0.278, 0.417, 0.1, 0.1, 0, 0.45};
float place_x[7] = {0.225, 0.376, 0.232, 0.232, 0.375, 0.3, 0.375};
float place_y[7] = {-0.302, -0.290, -0.42, -0.1, -0.1, 0, -0.45};
```

- publish status message and state of the robot state here means the pose of the robot

```
status.data = true;
goal_pub.publish(status);
state.data = 1;
pose_state_pub.publish(state);
ROS_WARN("pickhover start at:%.8f",ros::Time::now().toSec());
ROS_WARN("--------------------");
hoverPose(group_arm, pick_x[i], pick_y[i], goal_pub);
ROS_WARN("pickhover end at:%.8f",ros::Time::now().toSec());
ROS_WARN("--------------------");
status.data = false;
goal_pub.publish(status);
```

- sleep for 5 seconds befor starting the moveit call

```
ros::Duration(5, 0).sleep();
```

start the simulation

once the robot controller is setup launch the robot simulation.

build the robot simulation package and robot fi module

`catkin build -j4 -DCMAKE_BUILD_TYPE=Release -DFranka_DIR:PATH=~/libfranka/build`

Start simulation

- `source devel/setup.bash`
- `roslaunch panda_gazebo put_robot_in_world.launch`

Start FI Module and person sim

- `roslaunch robot_fi_tool fault_module.launch`
- to inject fault using GUI: `rosrun robot_fi_module fib_gui_v2.py`

or

Start random FI Process

- random planning and execution: `roslaunch robot_fi_tool rand_fault_injector.launch`
- random real-time: `roslaunch robot_fi_tool real_time_rand_fault_injection.launch`

FI Experiment

- `bash pipiline.sh`

this starts random fault injection experiment and saves the bag files into data_v4