trinity

ONLINE TRAJECTORY GENERATION FOR INDUSTRIAL ROBOT WITH 3D CAMERA

INTEGRATION TUTORIAL

www.trinityrobotics.eu

Hardware requirements

- Workstation PC
- GPU at least NVIDIA GeForce GTX 1060 or Quadro P5000, AMD Radeon Vega 56
- Memory at least 8 Gb
- Free disk space at least 1Gb
- USB 3.0 port
- Intel Realsense D435 camera
- Kuka Industrial robot compatible with ROS

Software requirements

- Ubuntu 18.04 or 20.04 OS
- ROS Melodic or Noetic distribution

Pre Work

- Ubuntu 18.04 or 20.04 OS installed
- ROS Melodic or Noetic distribution installed
- Camera connected to PC usb

Software requirements

- Ubuntu 18.04 or 20.04 OS
- ROS Melodic or Noetic distribution

Software requirements

- Ubuntu 18.04 or 20.04 OS
- ROS Melodic or Noetic distribution

Choosing the right camera

- Choosing the camera depends on environment and application requirements
  - Triggering, IP classification, connectivity, working principle, lighting…
- In this case, we need a high-resolution RGB+D image for object detection and trajectory planning
- → Intel Realsense D435i
  - Stereo camera with IR projector for enhanced accuracy

Positioning the camera

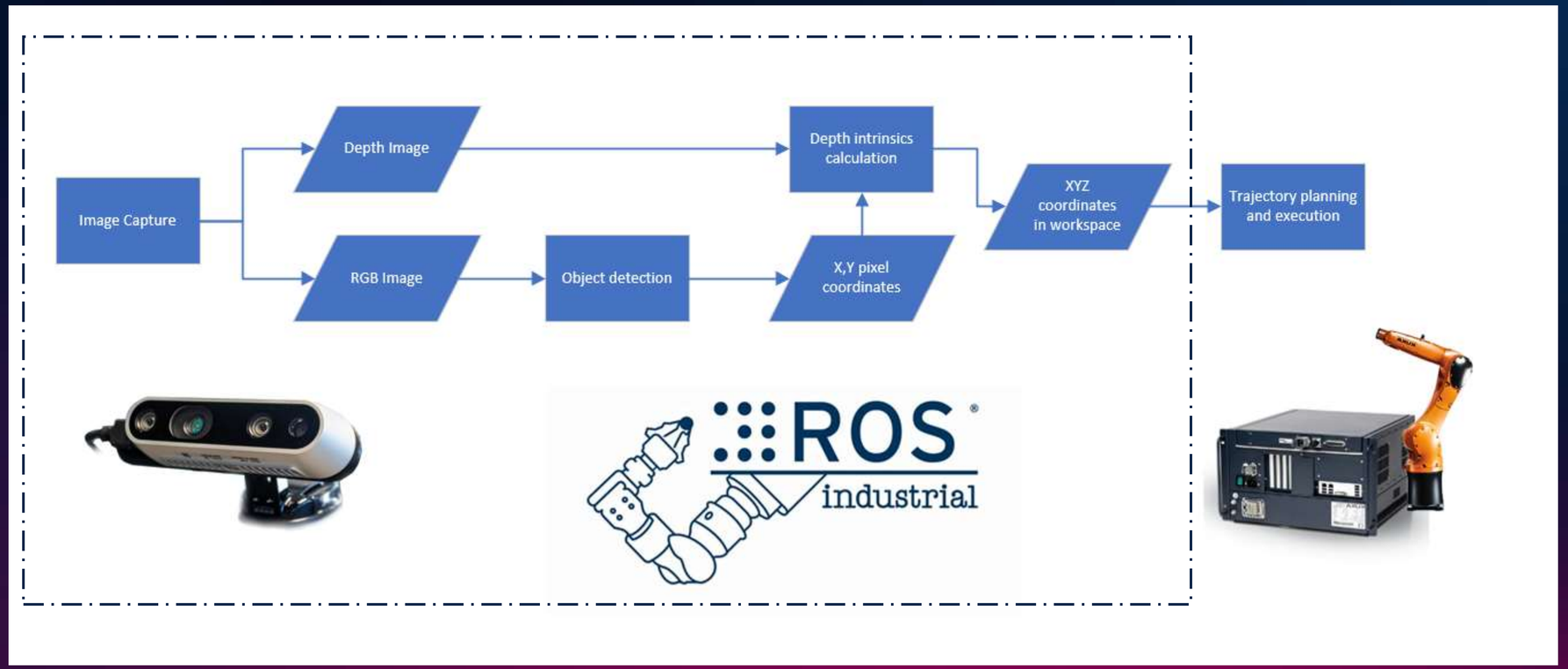
- Things to consider:
  - Good view of robot working area
  - Avoiding disturbance
    - Shaking, temperature, dust/fog/smoke, electromagnetic interference
  - Lighting
  - Connecting the device
  - Angle between camera and working area
    - Perpendicular is recommended to avoid positioning errors

Installing the camera

- Mounting guidelines may vary between equipment
  - Most manufacturers provide detailed screw pattern drawings, mounting instructions and mounting brackets
- Follow the manufacturer’s general instructions
- Ensure proper alignment

Interfacing ROS with camera

- Software/Hardware infrastructure:



Interfacing ROS with camera

- We will need the following software for our camera:
  - Intel Realsense SDK (OS install)
  - Realsense2-ros (ROS Package)
- We will also need an object detection framework:
  - Find-object (ROS Package)
- Let’s install the software on our PC

Installing camera drivers

- Download and install Intel Realsense SDK:
  - https://www.intelrealsense.com/sdk-2/

- Clone the realsense-ros repository into ROS workspace:
  - *Cd ~/catkin_ws/src*
  - *Git clone https://github.com/IntelRealSense/realsense-ros.git*

Installing camera drivers

- Install missing dependencies:
  - *cd ~/catkin_ws*
  - *Rosdep install --from-paths src --ignore-src -r -y*
- Build the workspace:
  - *Catkin_make*

Verifying camera driver installation

- Start the ROS Master:
  - *roscore*
- Start the realsense data publisher nodelet:
  - *Roslaunch realsense2_camera rs_aligned_depth.launch*
- Run rviz:
  - *Rosrun rviz rviz*

Verifying camera driver installation

- In Rviz:
  - Set ”Fixed frame” to ’camera_link’
  - Add a new display to the scene
  - Select ”DepthCloud” and click OK
  - Open the DepthCloud display menu and select ’/camera/aligned_depth_to_color/image_raw’ as Depth Map Topic
  - The depth map is now visualized in Rviz scene
- Let’s add a color map to this scene. Click ”Color image topic” from DepthCloud display menu
- Select ’/camera/color_image_raw’ topic and click OK
- The depthcloud is textured with the RGB camera data with proper alignment

Installing object detection framework

- Close all running ROS processes
- Get the find-object package:
  - *Cd ~/catkin_ws/src*
  - *Git clone https://github.com/introlab/find-object.git*
- Build the workspace:
  - *Cd ~/catkin_ws*
  - *Catkin_make*

Configuring object detection framework

- Start ROS master and realsense camera nodelet
  - *Roscore & roslaunch realsense2_camera rs_aligned_depth.launch*

- Run ”rostopic list” and write down following topics:
  - RGB image topic (camera/color/image_raw)
  - Depth image topic (camera/aligned_depth_to_color/image_raw)
  - RGB camera info topic (camera/color/camera_info)

Configuring object detection framework

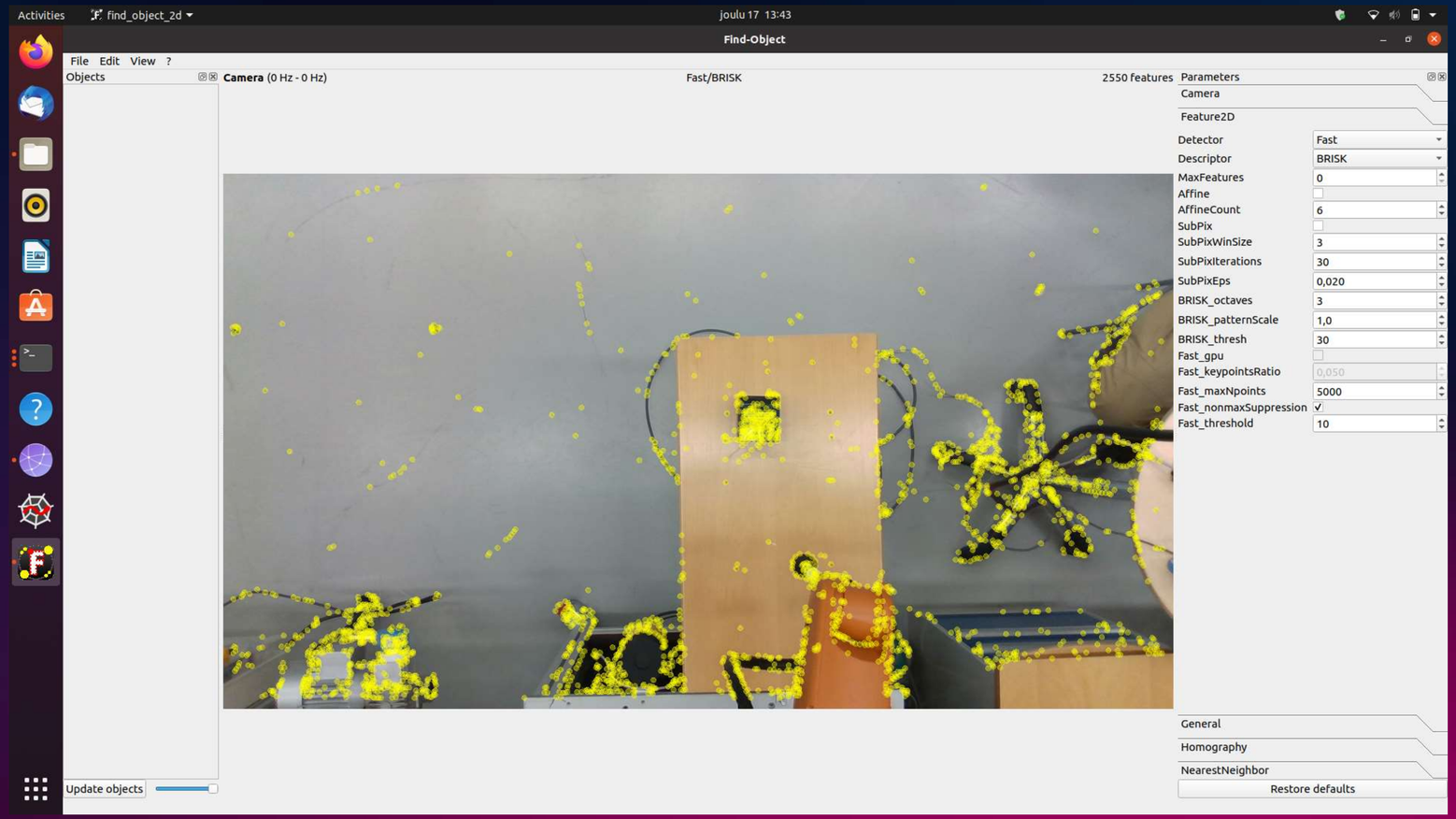
- Open the *'find_object_3d.launch'* file
  - *Gedit ~/catkin_ws/src/find_object/launch/find_object_3d.launch*

- Update the following parameters with topic names mentioned previously:

```
13          <arg name="rgb_topic"          default="camera/rgb/image_rect_color"/>
14          <arg name="depth_topic"        default="camera/depth_registered/image_raw"/>
15          <arg name="camera_info_topic" default="camera/rgb/camera_info"/>
```

Verifying object detection framework

- Launch find_object_3d:
  - *Roslaunch find_object find_object_3d.launch*



Verifying object detection framework

- Select an object from the working area:
  - Click ”Edit → Add objects from scene…”
  - Click ”Take picture”
  - Select the object region from scene and click ”Next”
  - Confirm object area features by clicking ”End”
- The object is instantly trained and will be detected from the working area
- Object ID and coordinates in camera frame are published to /objects topic

Interfacing ROS and robot

- We will need the following software for our robot:
  - Robot driver ROS Package (https://github.com/orgs/ros-industrial/repositories)
  - This training focuses on KUKA Robots (kuka-experimental package)
- We will also need to:
  - Position and orient the camera base frame to robot base frame
  - Interface the robot driver with object detection framework

- Let’s install the drivers

Installing robot drivers

- Get the kuka-experimental package:
  - *Cd ~/catkin_ws/src*
  - *Git clone https://github.com/ros-industrial/kuka_experimental.git*
- Install missing dependencies:
  - *cd ~/catkin_ws*
  - *Rosdep install --from-paths src --ignore-src -r -y*
- Build the workspace:
  - *Cd ..*
  - *Catkin_make*

Configuring robot drivers

- Locate and specify the robot's IP address:
  - *Cd ~/catkin_ws/src/kuka-experimental*
  - *Grep -iR Robot_IP*
  - *Usually found in hardware controller configuration*

Verifying robot driver installation

- Start the ROS Master:
  - *roscore*
- Launch robot driver nodelet:
  - *Roslaunch kuka_kr6_support load_kr6r900sixx.launch*
- Launch robot motion planning nodelet:
  - *Roslaunch kuka_moveit_configuration moveit_rviz.launch*

Verifying robot driver installation

- Test jog the robot using rviz:
  - Jog the robot in a safe direction
  - Click ’Plan’. The robot motion is simulated in rviz
  - Click ’Execute’. The computed path is sent to robot controller.
  - Communication between ROS and robot controller is working

Camera position and orientation

- We will need to specify camera’s position and orientation in robot’s base frame:
  - Jog the robot in a safe direction
  - Click ’Plan’. The robot motion is simulated in rviz
  - Click ’Execute’. The computed path is sent to robot controller.
  - Communication between ROS and robot controller is working

trinity