README.md

Setting Up Robotics Kit Environment

1. Requirements & Dependency

Supported Hardware Platforms

Platform	Supported Devices	Supported EVM
J721E	TDA4VM	TDA4VMXEVM

J7 Processor SDK RTOS

The Robotics SDK requires the pre-built package of J721E Processor SDK RTOS 7.3.0. The pre-built package contains Processor SDK Linux and libraries that are necessary for setting up the Robotics development environment.

Ubuntu PC

A Ubuntu (18.04 recommended) PC is required. For RViz visualization of ROS topics published from the J7 target, it is assumed that ROS (Melodic recommended) is installed on the PC. For ROS installation steps, please refer to this ROS wiki page.

USB Stereo Camera [Optional]

The Robotics SDK provides a OpenCV-based ROS driver for ZED stereo camera. All the demo applications can be tried out with a live stereo camera as well as a ROSBAG file provided. For configuration of a stereo camera, please see `drivers/zed_capture/README.md`.

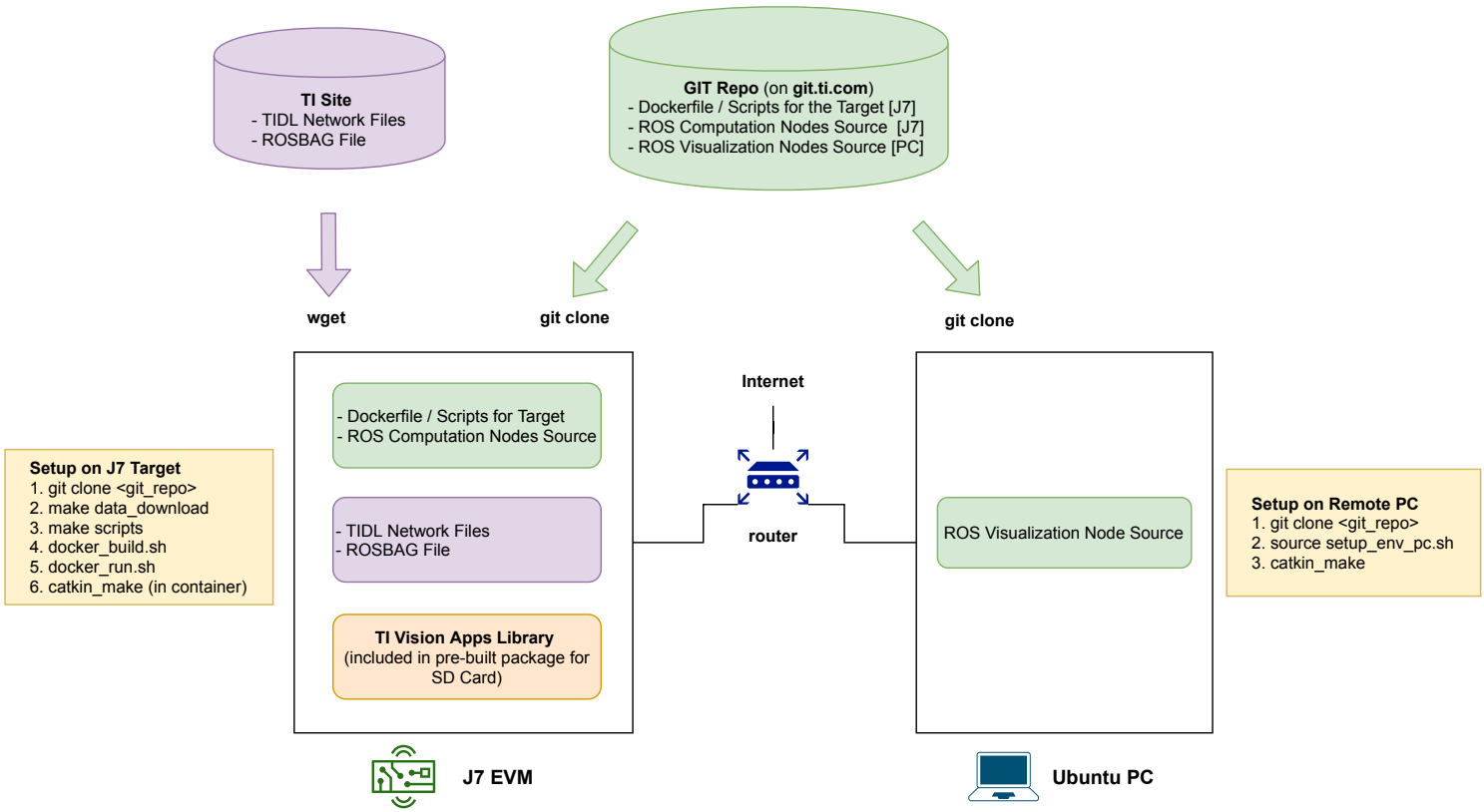

Figure 1. Robotics Kit: Setup and Installation Steps

2. Set Up the J7 Target and Development Environment

Figure 1 shows the hardware setup and high-level installation steps on the J7 target and the Ubuntu PC.

Build SD Card

1. From Ubuntu PC, download Processor SDK RTOS 7.3.0 pre-built package
2. Install the pre-built package to a SD card (minimum 32GB high-performance) by referring to the instruction on this page

Connect Remotely to the J7 Target

1. To find the IP address assigned to J7 EVM, use a serial port communications program (for example, `sudo minicom -D /dev/ttyUSBx` where `/dev/ttyUSBx` is the device for the UART serial port), log in with `root` account, and run `ifconfig`.
2. From a terminal on the PC, open a SSH session to connect remotely to the J7 target:

   ```
   user@pc:~$ ssh root@<J7_IP_address>
   ```

   Note: It is recommended to use a *static* IP for the J7 EVM to make ROS network setting easy.

Clone Git Repository

1. Set up the project directory and the catkin workspace:

   ```
   root@j7-evm:~# WORK_DIR=$HOME/j7ros_home
   root@j7-evm:~# CATKIN_WS=$WORK_DIR/catkin_ws

   root@j7-evm:~# mkdir -p $CATKIN_WS/src
   root@j7-evm:~# cd $CATKIN_WS/src
   ```

2. Clone the project GIT repository:

   ```
   git clone https://git.ti.com/git/processor-sdk-vision/jacinto_ros_perception.git
   ```

Download TIDL Model & ROSBAG File

1. For convenience, set up a soft-link:

   ```
   root@j7-evm:~/j7ros_home/catkin_ws/src# cd $WORK_DIR
   root@j7-evm:~/j7ros_home# ln -s $CATKIN_WS/src/jacinto_ros_perception/docker/Makefile
   ```

2. To download data files, run the following in `$WORK_DIR`:

   ```
   root@j7-evm:~/j7ros_home# make data_download
   ```

   Two tarballs (for deep-learning model artifacts, and a ROSBAG file) are downloaded and uncompressed under `$WORK_DIR/data`. If preferred, each tarball can be downloaded individually with `make model_download` and `make rosbag_download`, respectively.

Set Up Docker Environment

1. Following this link, check that Docker and network work correctly on the J7 host Linux.
2. To generate bash scripts for building and running a Docker image for the Robotics Kit:

```
root@j7-evm:~/j7ros_home# make scripts
```

Make sure that two bash scripts, `docker_build.sh` and `docker_run.sh`, are generated.

Note: The default Dockerfile installs `ros-perception` ROS package group in the Docker image. In case you want to minimize the size of Docker image by installing only minimal dependency required to build and run the ROS nodes under `$CATKIN_WS/src`, use the scripts generated with the following:

```
root@j7-evm:~/j7ros_home# make scripts_rosdep
```

3. To build the Docker image, in `$WORK_DIR` run:

```
root@j7-evm:~/j7ros_home# ./docker_build.sh
```

It will take several minutes to build the Docker image. The Docker image built can be listed with `docker images`.

3. Set Up the Ubuntu PC for Visualization

Open another terminal on Ubuntu PC to set up environment for RViz visualization.

1. Clone the GIT repository:

```
user@pc:~$ CATKIN_WS=$HOME/j7ros_home/catkin_ws
user@pc:~$ mkdir -p $CATKIN_WS/src
user@pc:~$ cd $CATKIN_WS/src
git clone https://git.ti.com/git/processor-sdk-vision/jacinto_ros_perception.git
```

2. Build the ROS nodes for visualization:

```
user@pc:~/j7ros_home/catkin_ws/src$ cd $CATKIN_WS
user@pc:~/j7ros_home/catkin_ws$ catkin_make
```

3. ROS network setting: For convenience, set up a soft-link:

```
user@pc:~/j7ros_home/catkin_ws$ ln -s src/jacinto_ros_perception/setup_env_pc.sh
```

Update the following lines in `setup_env_pc.sh`:

```
PC_IP_ADDR=<PC_IP_address>
J7_IP_ADDR=<J7_IP_address>
```

`<J7_IP_address>` can be found by running `make ip_show` on a J7 terminal.

To set up the PC environment, run the following:

```
user@pc:~/j7ros_home/catkin_ws$ source setup_env_pc.sh
```

4. Build and Run Demo Applications

Build the ROS Applications

1. To run the Docker image:

```
root@j7-evm:~/j7ros_home# ./docker_run.sh
```

2. To build the ROS applications, inside the Docker container:

```
root@j7-docker:~/j7ros_home/catkin_ws$ catkin_make
root@j7-docker:~/j7ros_home/catkin_ws$ source devel/setup.bash
```

Run Stereo Vision Application

1. **[J7]** To launch `ti_sde` node with playing back a ROSBAG file, run the following in `$WORK_DIR` on the J7 host Linux:

```
root@j7-evm:~/j7ros_home$ ./docker_run.sh roslaunch ti_sde bag_sde.launch
```

   Alternatively, you can run the following `roslaunch` command **inside** the Docker container:

```
root@j7-docker:~/j7ros_home/catkin_ws$ roslaunch ti_sde bag_sde.launch
```

   To process the image stream from a ZED stereo camera, replace the launch file with `zed_sde.launch` in the above.

2. **[PC]** For visualization, on the PC:

```
user@pc:~/j7ros_home/catkin_ws$ roslaunch ti_sde rviz.launch
```

Run Stereo Vision Application with Point-Cloud Enabled

1. **[J7]** To launch `ti_sde` node with playing back a ROSBAG file, run the following in `$WORK_DIR` on the J7 host Linux:

```
root@j7-evm:~/j7ros_home$ ./docker_run.sh roslaunch ti_sde bag_sde_pcl.launch
```

   Alternatively, you can run the following `roslaunch` command **inside** the Docker container:

```
root@j7-docker:~/j7ros_home/catkin_ws$ roslaunch ti_sde bag_sde_pcl.launch
```

   To process the image stream from a ZED stereo camera, replace the launch file with `zed_sde.launch` in the above.

2. **[PC]** For visualization, on the PC (Make sure `source setup_env_pc.sh` ahead):

```
user@pc:~/j7ros_home/catkin_ws$ roslaunch ti_sde rviz_pcl.launch
```

Run CNN Semantic Segmentation Application

1. **[J7]** To launch `ti_semseg_cnn` node with playing back a ROSBAG file, run the following in `$WORK_DIR` on the J7 host Linux:

```
root@j7-evm:~/j7ros_home$ ./docker_run.sh roslaunch ti_semseg_cnn bag_semseg_cnn.launch
```

   Alternatively, you can run the following `roslaunch` command **inside** the Docker container:

```
root@j7-docker:~/j7ros_home/catkin_ws$ roslaunch ti_semseg_cnn bag_semseg_cnn.launch
```

To process the image stream from a ZED stereo camera, replace the launch file with `zed_semseg_cnn.launch` in the above.

2. **[PC]** For visualization, on the PC:

```
user@pc:~/j7ros_home/catkin_ws$ roslaunch ti_semseg_cnn rviz.launch
```

Run 3D Obstacle Detection Application

1. **[J7]** To launch `ti_estop` node with playing back a ROSBAG file, run the following in `$WORK_DIR` on the J7 host Linux:

```
root@j7-evm:~/j7ros_home$ ./docker_run.sh roslaunch ti_estop bag_estop.launch
```

Alternatively, you can run the following `roslaunch` command **inside** the Docker container:

```
root@j7-docker:~/j7ros_home/catkin_ws$ roslaunch ti_estop bag_estop.launch
```

To process the image stream from a ZED stereo camera, replace the launch file with `zed_estop.launch` in the above.

2. **[PC]** For visualization, on the PC:

```
user@pc:~/j7ros_home/catkin_ws$ roslaunch ti_estop rviz.launch
```