How to Set Up TI OpenVX + ROS Environment

Requirements & Dependency

Supported Hardware Platforms

Platform	Supported Devices	Supported EVM
J721E	TDA4VM	TDA4VMXEVM

J7 Processor SDK RTOS

This TI OpenVX + ROS development framework works with J721E Processor SDK RTOS 7.1.0.

1. Download Pre-built Package
2. Install to a SD card by referring to the instruction on this page

Ubuntu PC

A Ubuntu (18.04 recommended) PC is required. For RViz visualization of input/output topics published from the J7, it is assumed that ROS (Melodic recommended) is installed on the Ubuntu PC.

Once finding the IP address assigned to J7 EVM (using a serial port communications program, for example, `minicom`), connect to J7 Linux with SSH:

```
ssh root@<J7_IP_address>
```

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

Setup

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

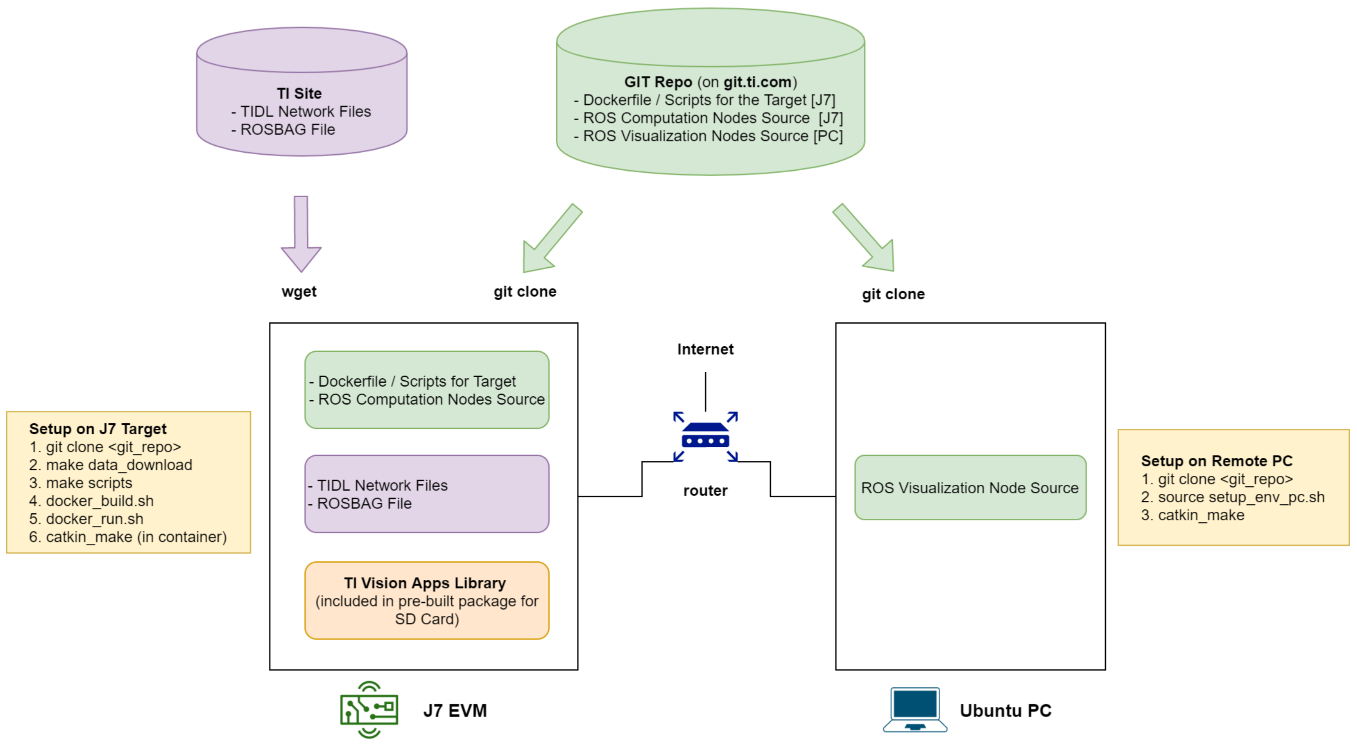

Figure 1. TI OpenVX + ROS Framework: Setup and Installation

Clone Git Repository

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

```
WORK_DIR=$HOME/j7ros_home
CATKIN_WS=$WORK_DIR/catkin_ws

mkdir -p $CATKIN_WS/src
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 following soft-links:
   ```
   cd $WORK_DIR
   ln -s $CATKIN_WS/src/jacinto_ros_perception/docker/Makefile
   ```
2. To download data files, run the following in `$WORK_DIR`:

   ```
   make data_download
   ```

   Two tarballs (TIDL network model files, and a ROSBAG file) are downloaded and uncompressed under `$WORK_DIR/data`. Each tarball can be downloaded individually with `make tidl_net_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 project:

   ```
   make scripts
   ```

   Make sure that two bash scripts named `docker_build.sh` and `docker_run.sh` are generated.
3. To build the Docker image, at `$WORK_DIR` run:

   ```
   ./docker_build.sh
   ```

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

Note: The Docker image that is built using `Makefile` provided in the GIT repository will include minimal number of ROS packages on which the ROS package(s) under `$CATKIN_WS/src` have dependency. In case when more ROS package(s) are added under the folder, it is required to re-build a Docker image using the `Makefile`.

Set Up Remote PC for Visualization

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

1. Clone GIT repository:

   ```
   CATKIN_WS=$HOME/j7ros_home/catkin_ws
   mkdir -p $CATKIN_WS/src
   cd $CATKIN_WS/src
   git clone https://git.ti.com/git/processor-sdk-vision/jacinto_ros_perception.git
   ```
2. Build ROS nodes:

   ```
   cd $CATKIN_WS
   catkin_make
   ```
3. ROS network setting: For convenience, set up a soft-link:

   ```
   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 **J7 terminal**.

   To set up the PC environment, run the following:

   ```
   source setup_env_pc.sh
   ```

After launching ROS nodes on the J7, we can check the all the ROS topics by running "`rostopic list`".

Build Demo ROS Applications

1. To run the docker image:

```
./docker_run.sh
```

2. To build ROS applications, inside the Docker container:
```
cd $CATKIN_WS
catkin_make
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:

   ```
   ./docker_run.sh roslaunch ti_sde bag_sde.launch
   ```

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

   ```
   roslaunch ti_sde bag_sde.launch
   ```

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

   ```
   roslaunch ti_sde rviz.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:

   ```
   ./docker_run.sh roslaunch ti_semseg_cnn bag_semseg_cnn.launch
   ```

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

   ```
   roslaunch ti_semseg_cnn bag_semseg_cnn.launch
   ```

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

   ```
   roslaunch ti_semseg_cnn rviz.launch
   ```

Run Stereo Vision and CNN Semantic Segmentation Together

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

   ```
   ./docker_run.sh roslaunch ti_sde bag_sde_semseg.launch
   ```

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

   ```
   roslaunch ti_sde bag_sde_semseg.launch
   ```

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

   ```
   roslaunch ti_sde rviz_sde_semseg.launch
   ```