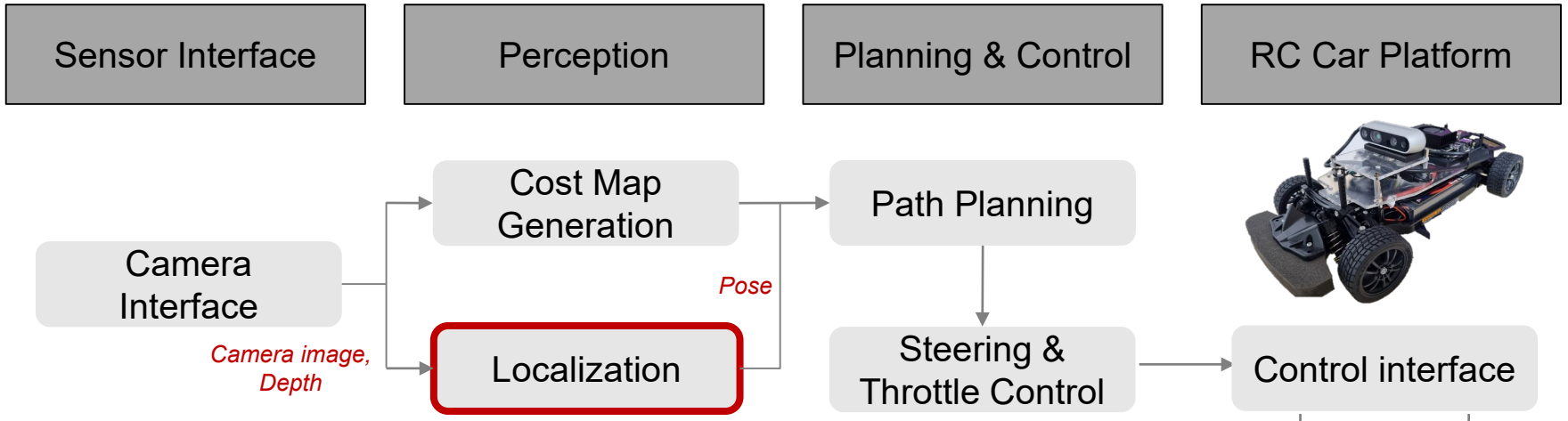
EE405A
Visual SLAM

(TA) Gyuree Kang
School of Electrical Engineering
KAIST

November. 10th, 2023

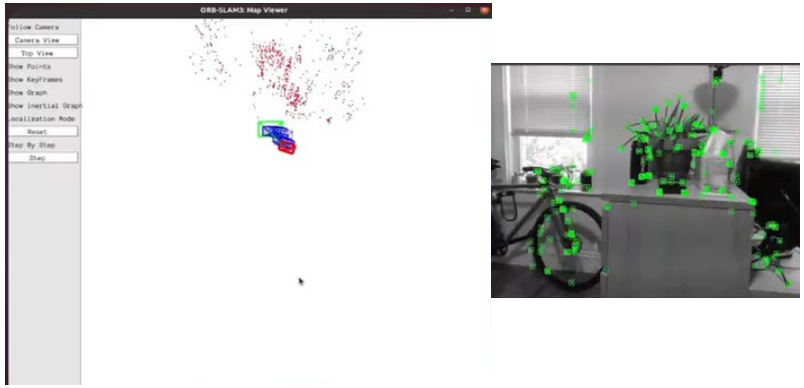
Localization

❖ Perception – Localization



➢ **Localization**

- ❑ Localization is a module to know where the ego robot is located in the environment.
- ❑ Visual localization often incorporates Visual SLAM, the robot simultaneously builds a map of the environment while estimating the ego's current position within that map.
- ❑ The vehicle's position w.r.t. track environment can be used for the path planner or controller.



An example of the vision-based navigation (ORB SLAM3)

Localization

❖ **Where am I now? (ways to do localization)**

Do I have a map already?

yes

Localization

Monte-Carlo Localization(MCL)

**use existing map to know where I am*

no

Do I know where I am?

yes

Map Building Problem

**using my location, build map around me*

no

SLAM

ORB-SLAM, LSD-SLAM, etc. (next slide)

- Apart from SLAM, which considers map building as a part of its tasks, visual odometry is another popular topic.
- Recently, visual+inertial navigation gains a lot of attention.

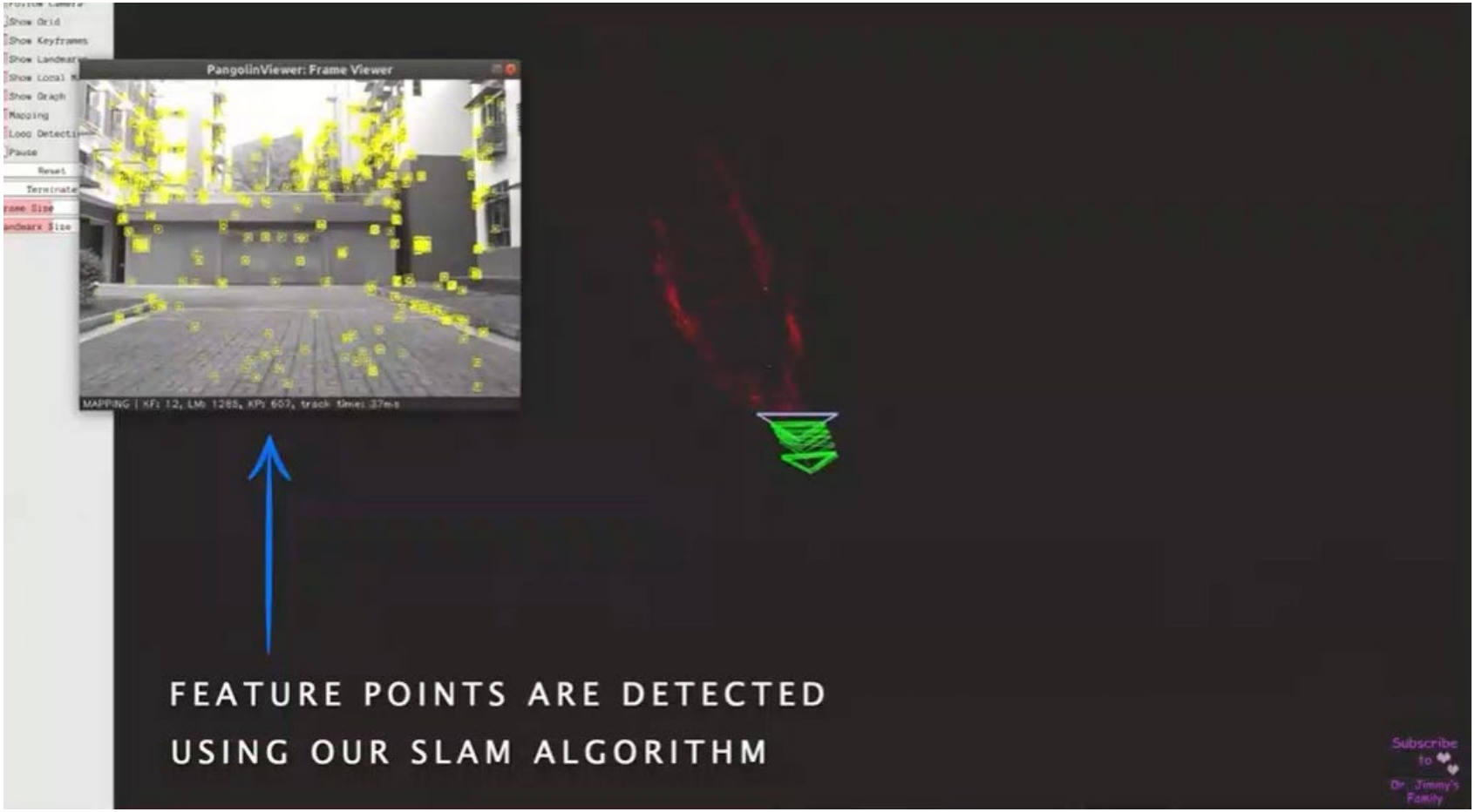
SLAM

❖ What is SLAM?

- ❑ Simultaneous Localization and Mapping
  - If map exists: localization
  - If position is known: surrounding map can be created.
  - If none of these exists: localization can be done using map being built
- ❑ As the sensor system moves, error accumulates. (no external markers are available)→loop closure
- ❑ Loop Closure
  - as localization and mapping are not fully available, a place visited twice can be used to correct the error.
  - If loop closure is not performed, it reduces to odometry problem.

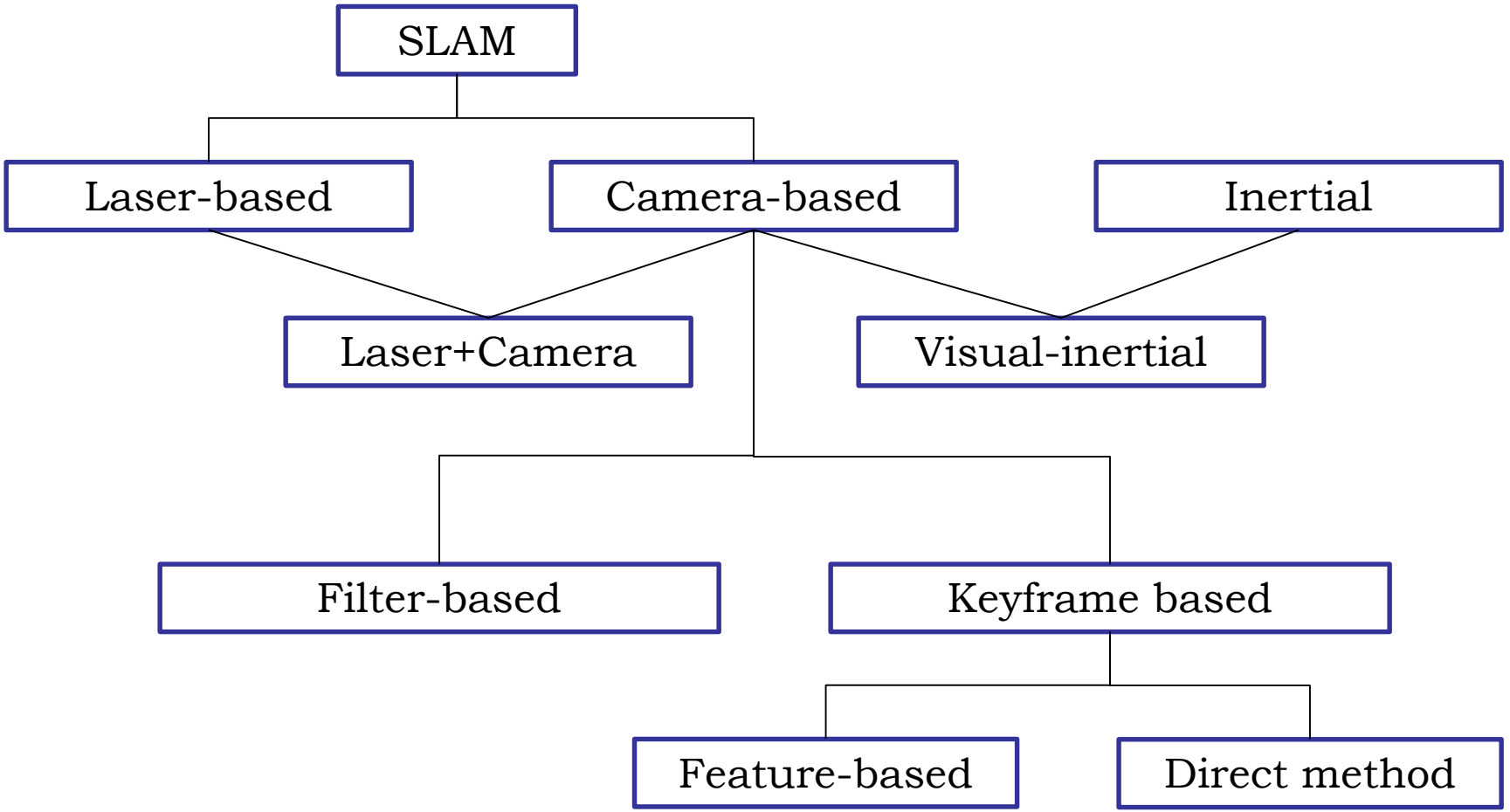
SLAM (Loop closure)

❖ **Loop closure**



SLAM

❖ **Development of SLAM**

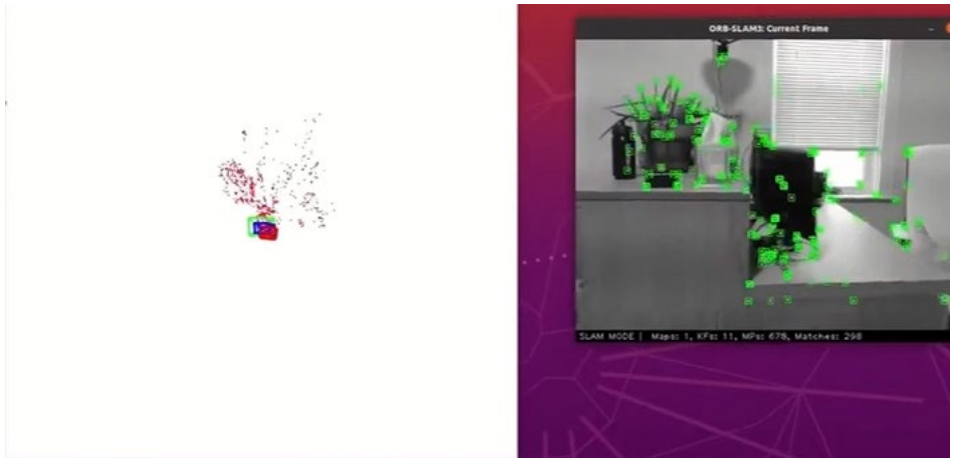


SLAM

❖ **LiDAR SLAM (BLIO)**

https://youtu.be/h_PsFVhh-As?si=L_mWsS353aO1MYPQ

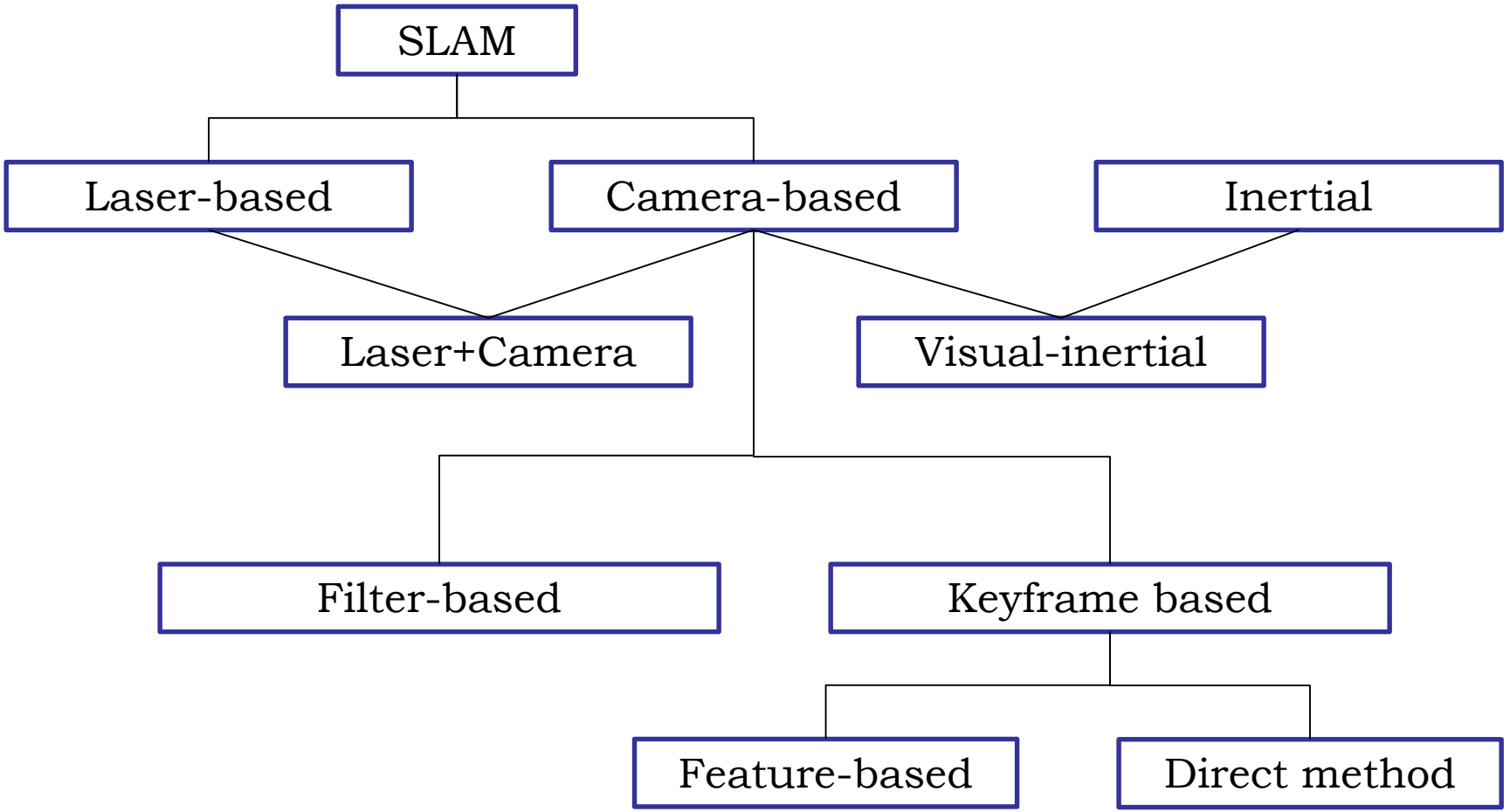
❖ **Camera SLAM (ORB SLAM3)**



https://youtu.be/VOHloE1mnos?si=jSsB_YmEOKq44gAV

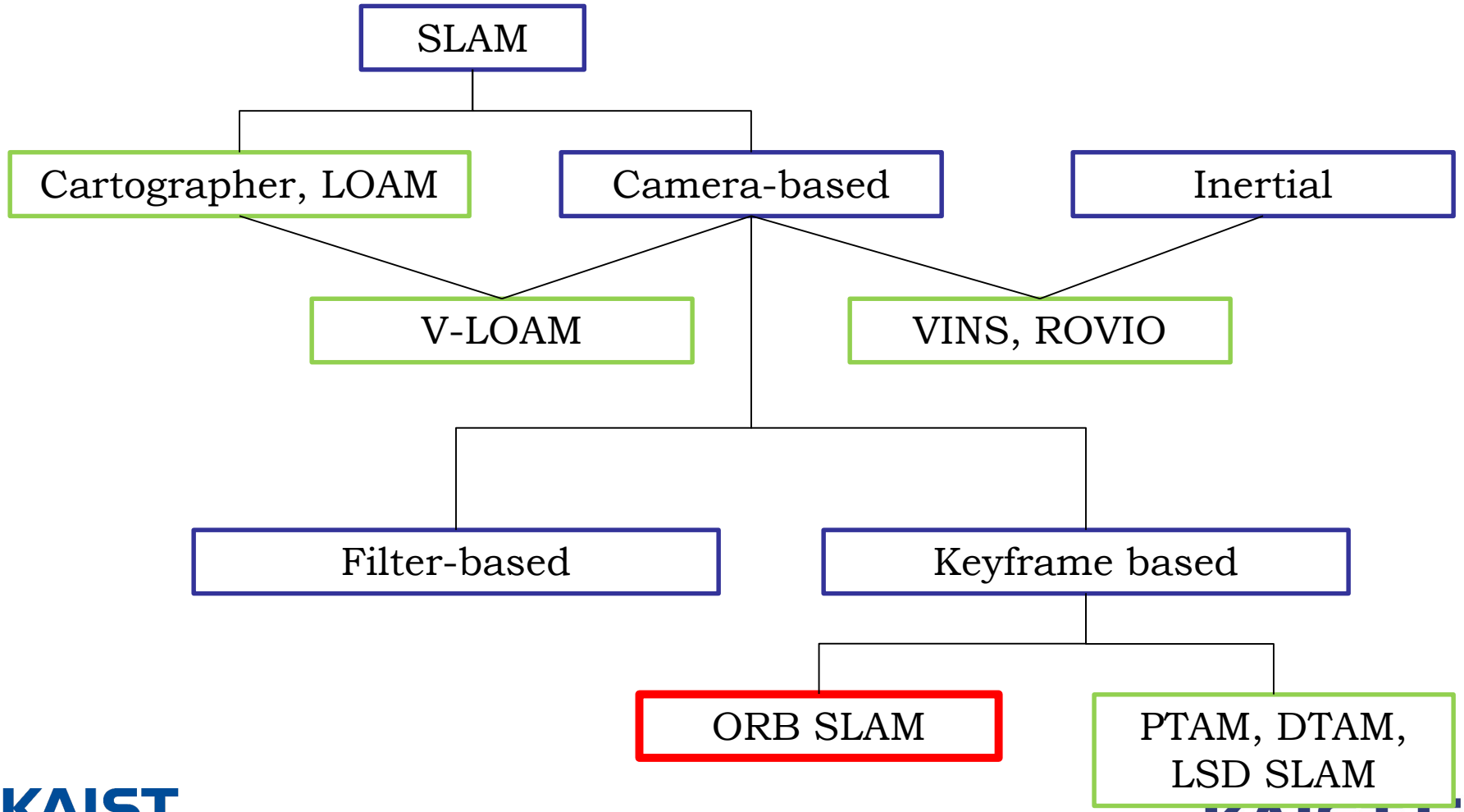
SLAM

❖ **Development of SLAM**



SLAM

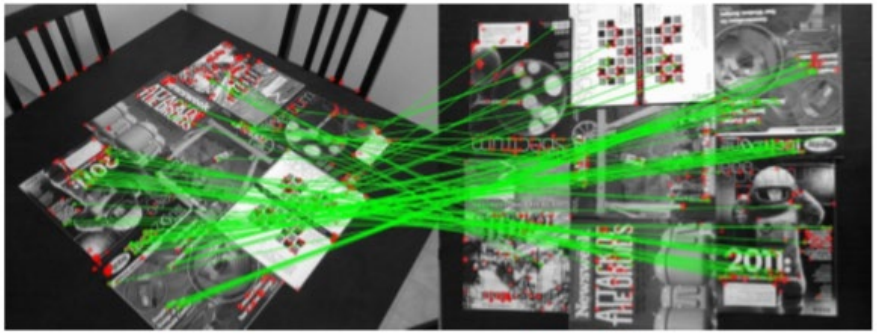
❖ **Development of SLAM**

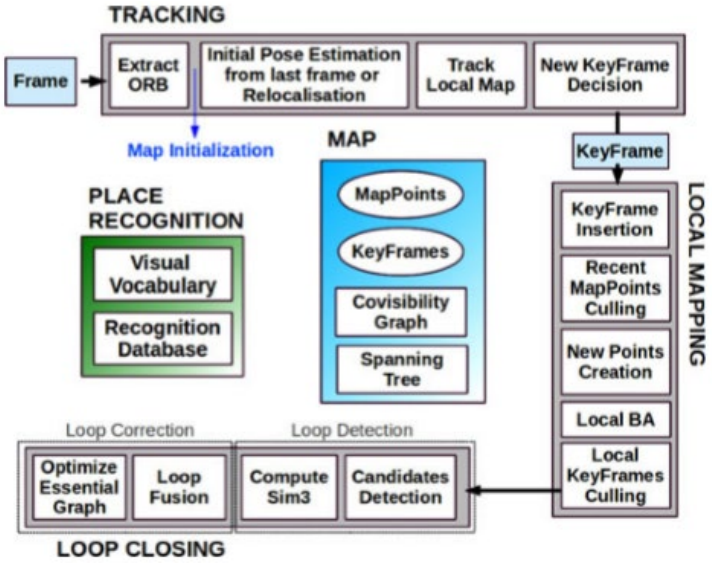


ORB SLAM

ORB SLAM

- What is ORB SLAM?
  - ORB SLAM is a feature based visual SLAM
  - uses ORB* (Oriented FAST and Rotated BRIEF) keypoint detector.
  - 3 Major tasks:
    - Tracking
      - ORB extraction
      - Initial pose estimation
    - Local Mapping
      - Keyframe insertion
      - New map point creation
    - Loop Closing
      - Loop detection/fusion
      - Essential graph optimization





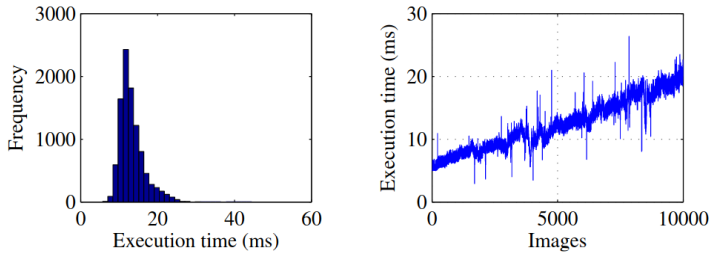
*ORB (Oriented FAST and Rotated BRIEF)

*BRIEF: Binary Robust Independent Elementary Features

**ORB-SLAM: A Versatile and Accurate Monocular SLAM System, Mur-Artal et al, IEEE TRO, Oct 2015.*

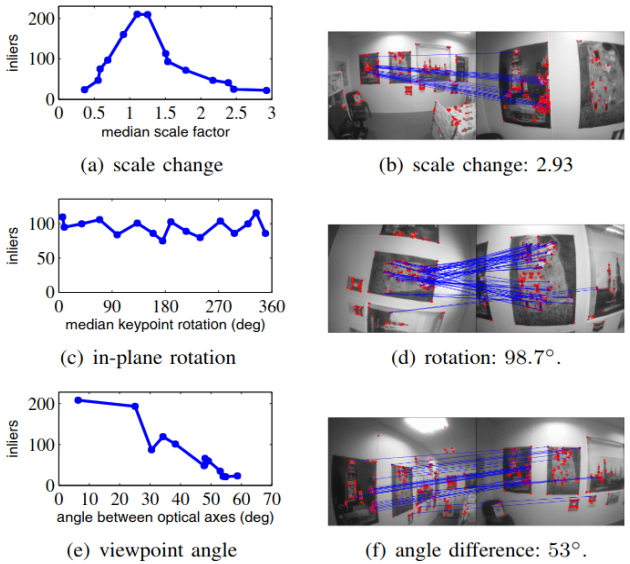
ORB SLAM

- ORB features: Oriented FAST and rotated BRIEF
  - Fast for extraction (~ 13.3 ms)
  - This excludes SIFT (~300 ms), SURF (~ 300 ms).
- Rotation invariance => Good invariance to viewpoint
  - Good performance for place recognition
    - Robust to viewpoint (rotation and scale) and good invariant to illumination change, camera auto-gain and auto-exposure.



(a) ORB extraction (histogram) (b) Loop detection

Fig. 2. Execution times for 10K images from NewCollege. Loop detection times do not include geometrical check.



(a) scale change (b) scale change: 2.93

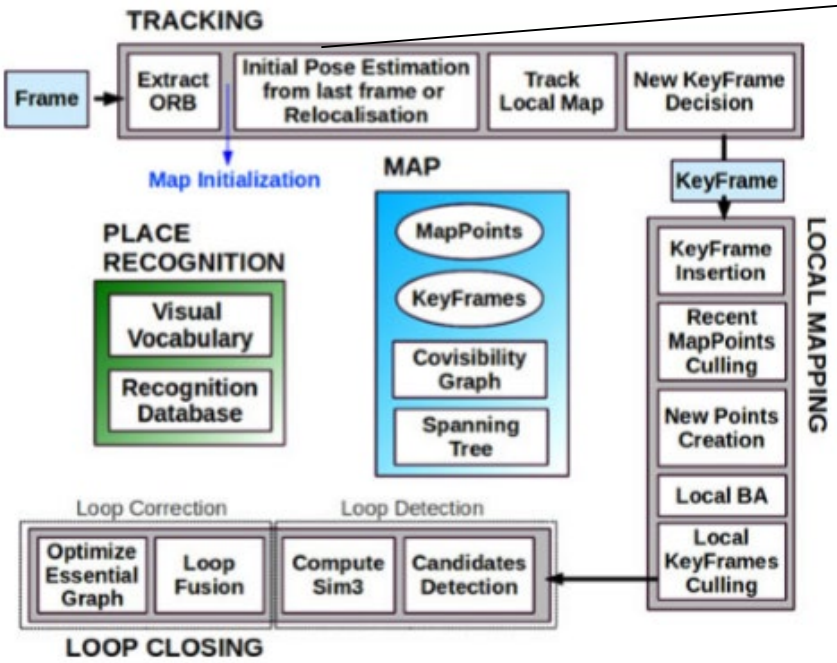
(c) in-plane rotation (d) rotation: 98.7°.

(e) viewpoint angle (f) angle difference: 53°.

Fig. 8. On the left: PnP inliers at each relocalisation. On the right: an example for each experiment. Blue lines are the inlier correspondences.

ORB SLAM

- 3 Processes in ORB SLAM



Tracking thread

- Localization of camera with every frame by extracting ORB.
- Initial pose is estimated.If tracking is lost, global search is performed using BoW
- Project maps into current map point projection and optimize.
- Check to insert as a keyframe or not (depending on accuracy, timing, etc)

Local Mapping thread

- Map points are added (redundants are culled)
- Using the matched points in local map, camera pose is optimized.
- Local BA is performed to achieve optimal reconstruction
- Redundant keyframes are deleted(culling)

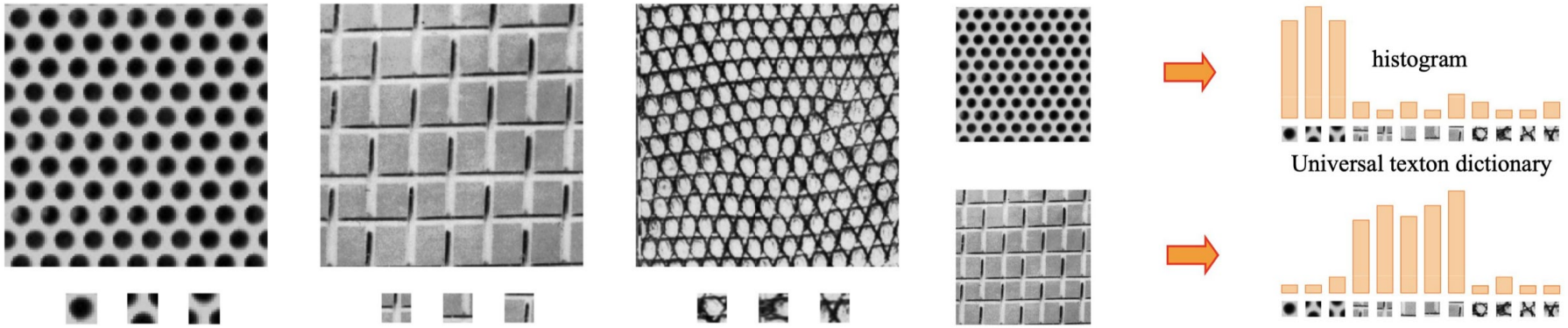
Loop Detection/Correction thread

- Loop closing searches for loops with every new keyframe
- When loop is detected, overlapping points are fused and pose graph optimization is performed for global consistency using “Essential Graph”

ORB-SLAM: A Versatile and Accurate Monocular SLAM System, Mur-Artal et al, IEEE TRO, Oct 2015.

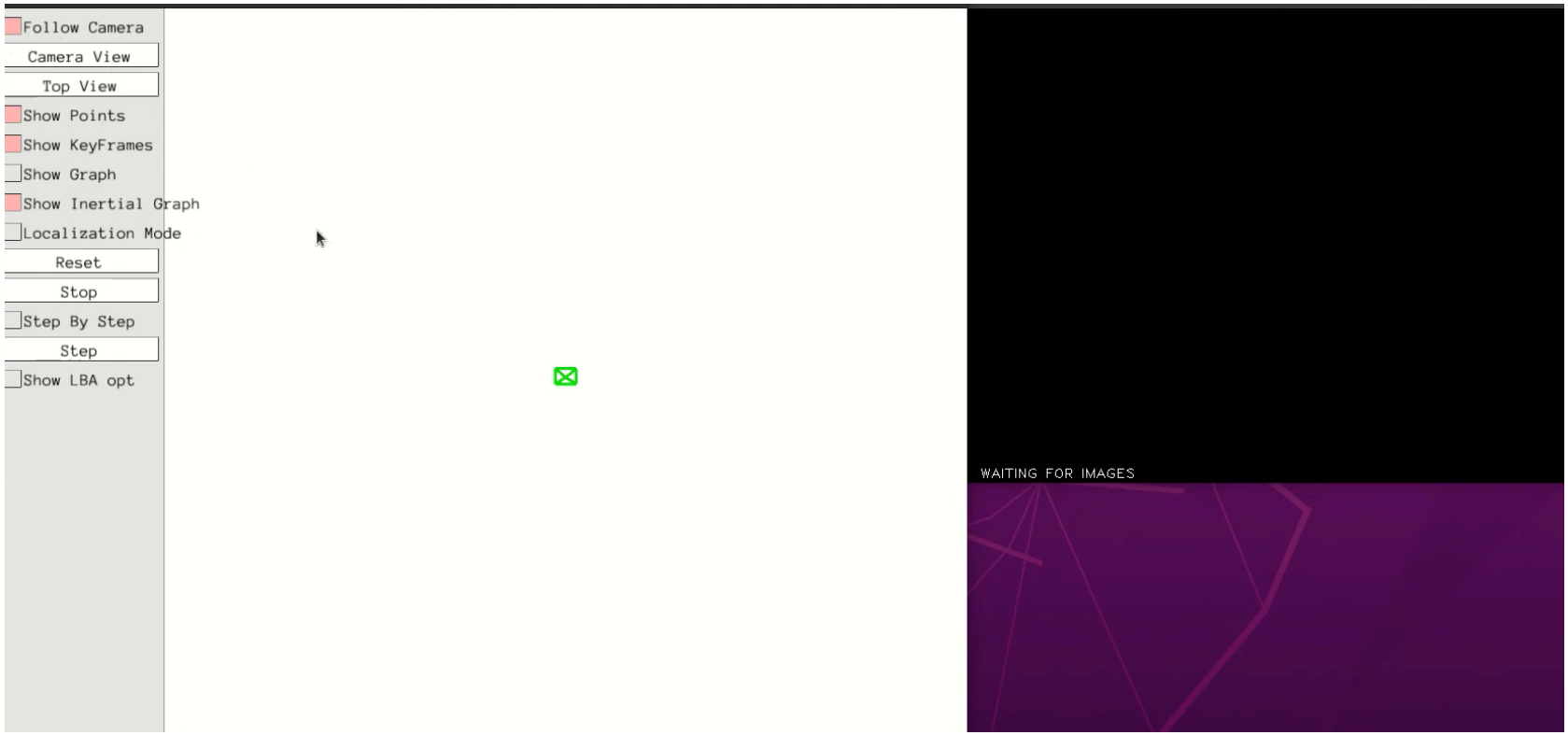
ORB SLAM

- Bags of Words
  - Representing an image using a bag of local features (visual words)
  - If the images are general enough, the same vocabulary can be used for different environments.
  - The vocabulary is created offline with the ORB descriptors extracted from a large set of images.
  - Very widely used these days.
- Used for loop detection and relocalization
- Used for searching matches for triangulating new points



ORB SLAM

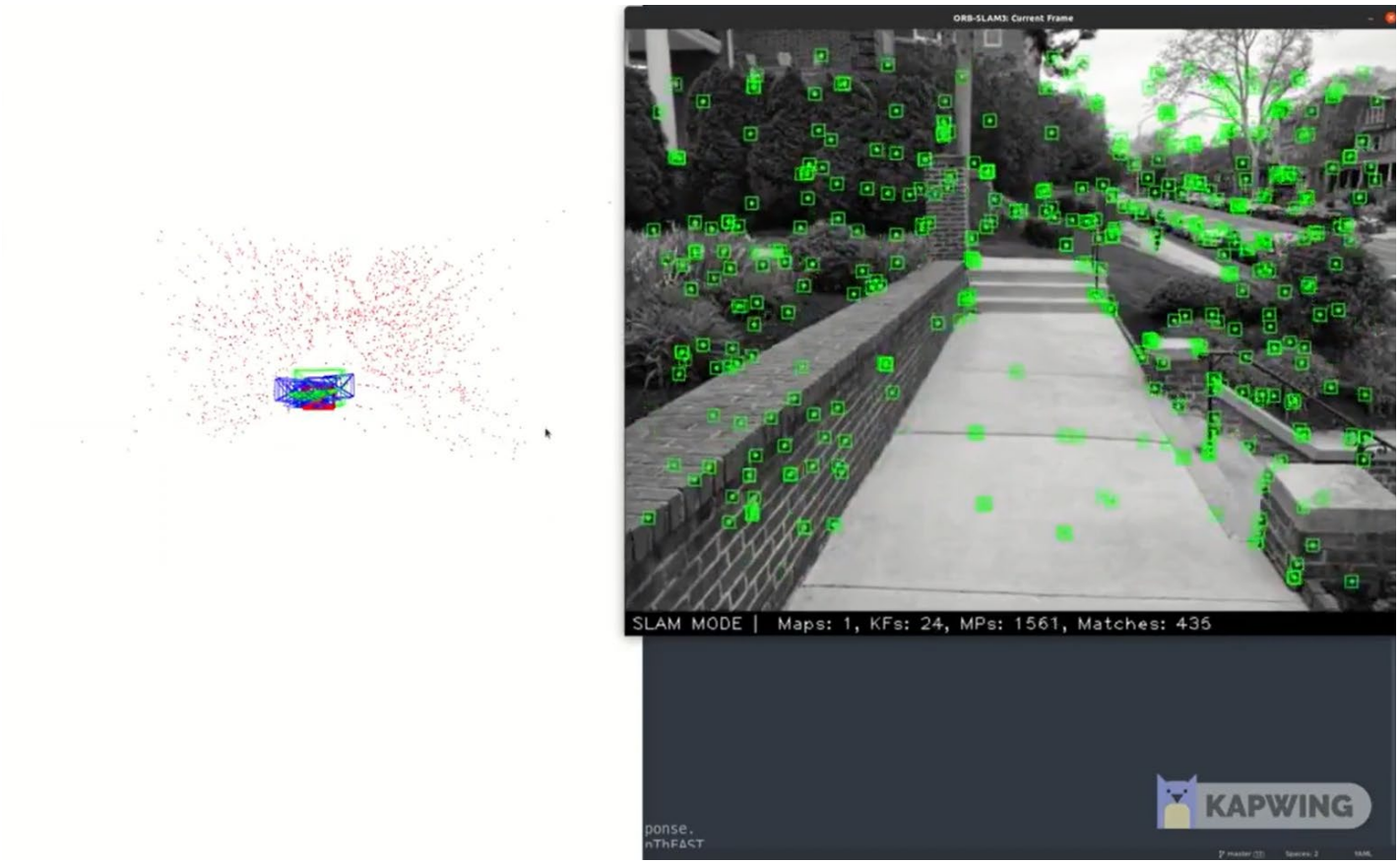
❖ Result (Indoor)



https://youtu.be/tYuVzymr1qc?si=hF6eUAlefeT5zbx_

ORB SLAM

- Result (Outdoor)



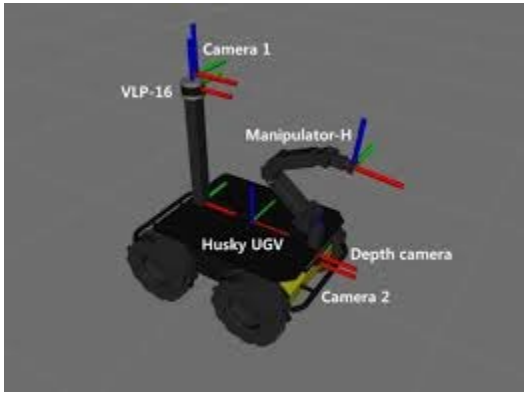
https://youtu.be/UVPDS3sU8xg?si=kiK1Zo_rpfjdt7Gv

Frame Transformation (TF)

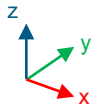
Frame Transformation

❖ **What is tf (Transformation)?**

- ❑ tf is a package that lets the user **keep track of multiple coordinate frames over time**.
- ❑ tf maintains **the relationship between coordinate frames** in a **tree structure** buffered in time, and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time.



Frames in a mobile robot with multiple sensors.



A frame
(=coordinate)

Frame Transformation

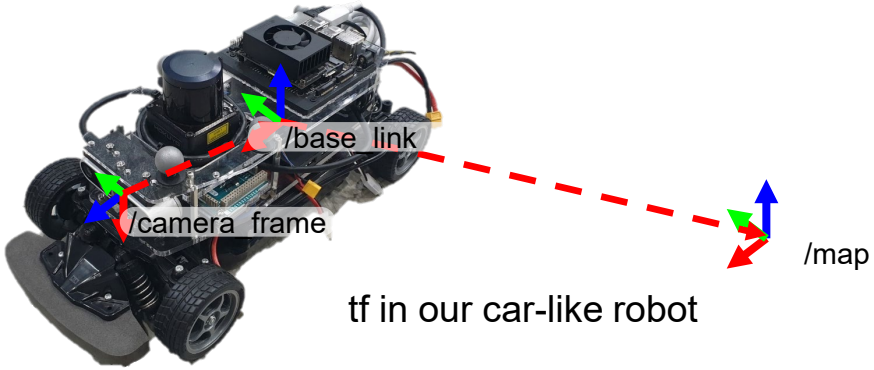
❖ **TFs in Car-Like Robot**

❑ TF for perception: calculating **relative position and orientation of detected objects (obstacles) on the sensor frame** with respect to the base of our robot.

$$X_{base\_link}^{obstacle} = \boldsymbol{T}_{\boldsymbol{base\_link}}^{\boldsymbol{sensor}} X_{sensor}^{obstacle}$$

❑ TF for localization: estimating **where the base of our robot is currently located** based on the map(world) frame.

$$X_{base\_link}^{goal} = X_{map}^{goal} T_{base\_link}^{map}, \text{ or } X_{map}^{target} = X_{base\_link}^{target} T_{map}^{base\_link}$$



tf in our car-like robot

TF for Perception

❖ TF for Perception

- ❑ Since the sensor data is measured with respect to the sensor frame, **the value of the measurement varies according to the transformation** between the sensor and base frames (which is the control output frame).
- ❑ TF can **define the relationship between the coordinate frames** in the robot system, enabling the **conversion of measurement values into the base coordinate frame.**

$$X_{base\_link}^{obstacle} = \boldsymbol{T}_{\boldsymbol{base\_link}}^{\boldsymbol{sensor}} X_{sensor}^{obstacle}$$

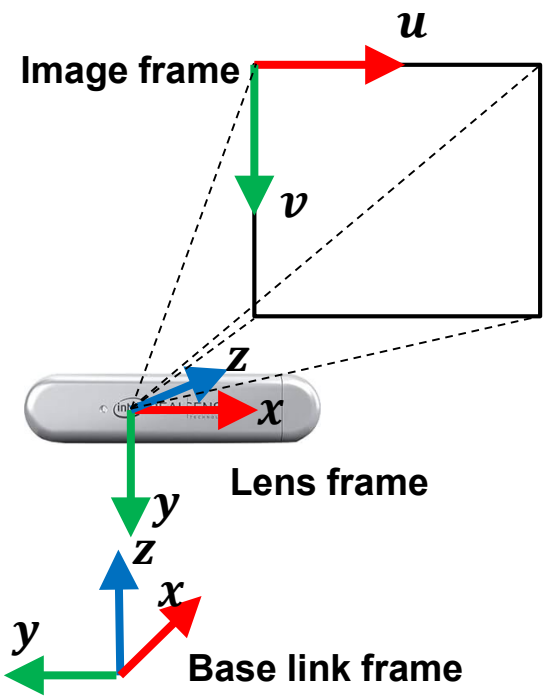
TF for Perception

- Image frame (2D) (v, u)
- Camera lens frame (3D) (x, y, z)

- Tfs for camera

$$X_{base\_link}^{obstacle} = T_{base\_link}^{camera} T_{camera}^{lens} X_{lens}^{obstacle}$$

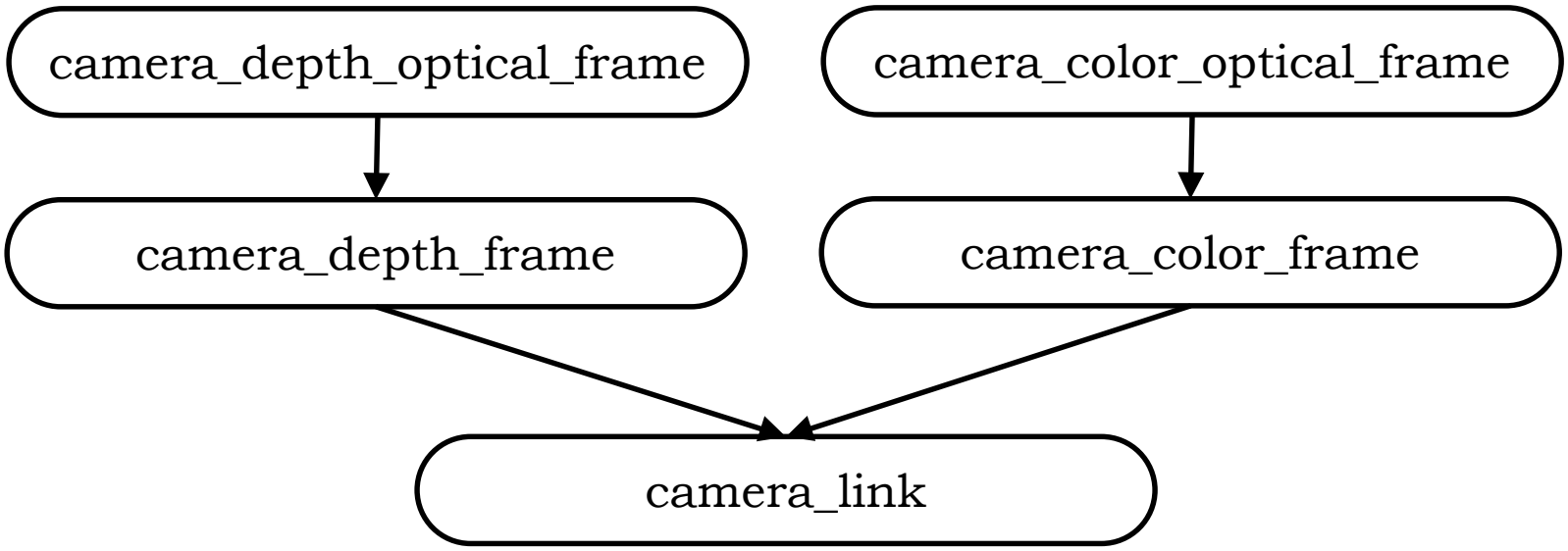
  - $\mathbf{X}_{\mathbf{lens}}^{\mathbf{obstacle}}$: Lens frame ↔ Obstacle (object)
    - **Target object pose estimation**
    - Depth estimation
  - $\mathbf{T}_{\mathbf{camera}}^{\mathbf{lens}}$: Camera base frame ↔ Lens frame
    - **Known value** by measurement
    - For realsense you can get it using "**rostopic echo /tf_static**" (list of static tfs)
  - $\mathbf{T}_{\mathbf{base\_link}}^{\mathbf{camera}}$**:** Base frame ↔ Camera base frame
    - **Value you have to measure**



Robot perception frames

TF for Perception

- Realsense TF tree ($\mathbf{T}_{\mathbf{camera}}^{\mathbf{lens}}$: Camera base frame ↔ Lens frame)



Realsense camera frames

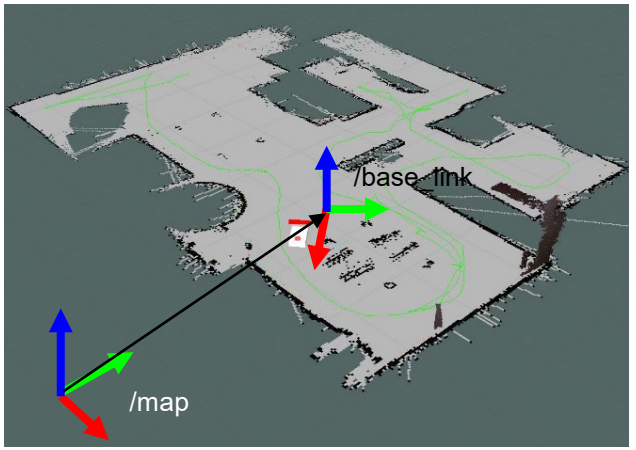
TF for Localization

- Localization is a transformation between **fixed global frame** of a map and **dynamic local body frame** of our robot.
- TF for Localization
  - To give a goal point in the map frame as a control/planning point in robot's base frame

$$X_{base\_link}^{goal} = X_{map}^{goal} T_{base\_link}^{map}$$

  - To save a target (object) pose on the map

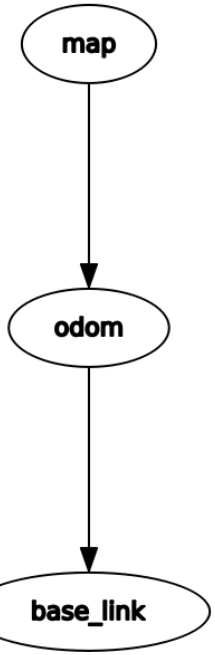
$$X_{map}^{target} = X_{base\_link}^{target} T_{map}^{base\_link}$$



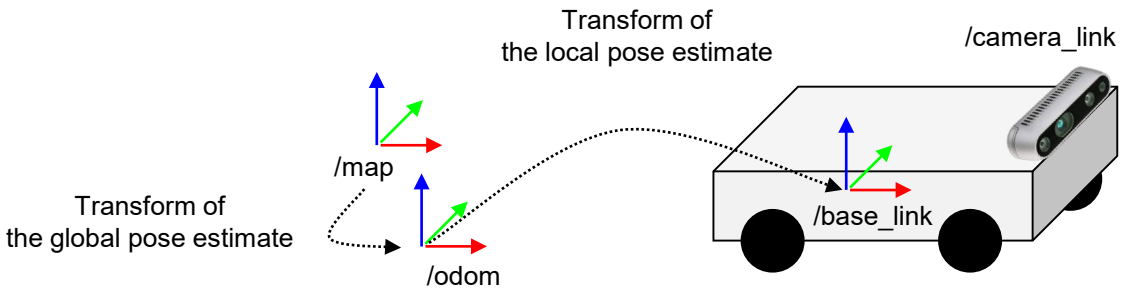
Localization within global map

TF for Localization

- TF for Localization
  - Global & local pose estimates with coordinate frames
    - There are two kinds of pose estimates: global and local pose estimates.
    - **Global pose estimate** is more accurate in the long run but discontinuous (/map → /odom).
    - **Local pose estimate** is continuous but drifts over time (/odom → /base_link).
    - These two pose estimates affect different transforms between three coordinate frames: '/map', '/odom', and '/base_link'.



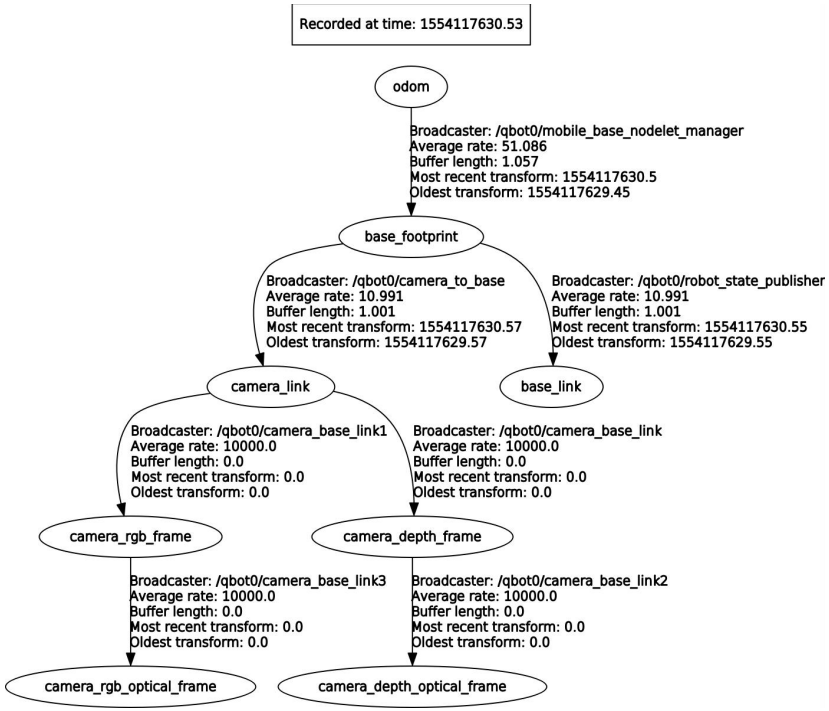
Tf tree



Global & local pose estimation

ROS TF tips

- Ways to check TFs
  - Static TFs
    - "rostopic echo /tf_static"
  - Dynamic TFs
    - "rostopic echo /tf"
  - Visualize all TFs as tf tree
    - "rosrun rqt_tf_tree rqt_tf_tree"
  - Visualize all TFs in RVIZ
    - "rivz" → Click "Add"
      → In "By display type" click "TF"
- Way to add TFs
  - Broadcast TF
    - C++ tutorial: http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20broadcaster%20%28C%2B%2B%29
    - Python tutorial: http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20broadcaster%20%28Python%29
  - Publish TF
    - static_transform_publisher: http://wiki.ros.org/tf#static_transform_publisher

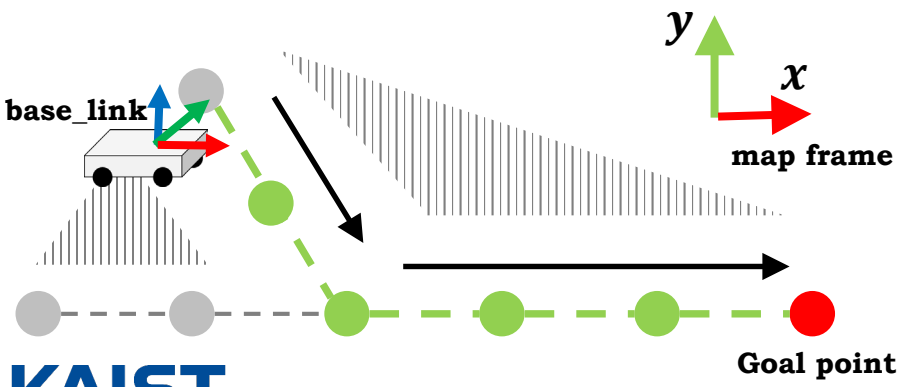


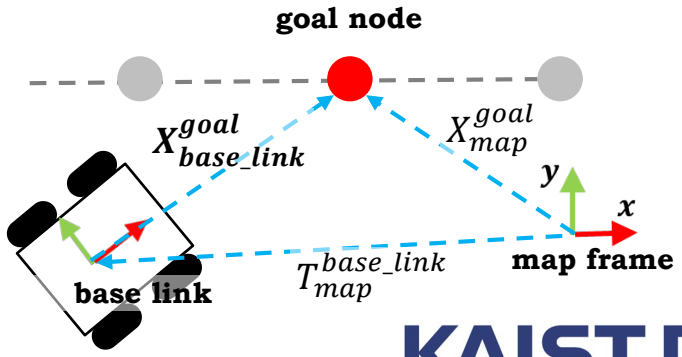
Examples of the tf tree

Global Path

Node-Link Path Planning

- Nodes
  - The points on the map that the robot can visit
  - You can add heuristic value to each nodes to select the goal node
- Links
  - Connection between two nodes
  - A collision free straight path between two nodes
- Path planning with node-links
  - Find the node that is closest to the goal point and set it as **goal node**
  - **Find path to the goal node using links** connecting the nodes
  - Apply different path planning algorithms (A*, dijkstra) to find the closest path to the goal point/node



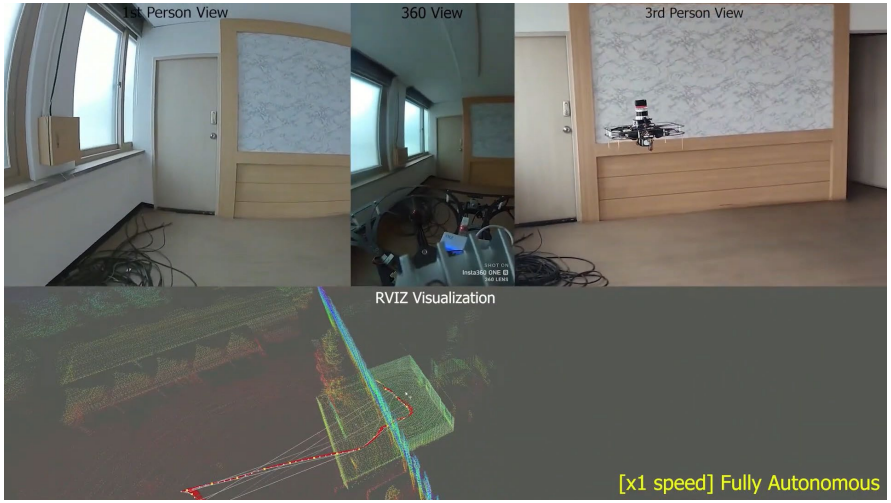


Node-Link Path Generation

- Node-link waypoint generation
  - Trajectory of the robot
    - Stack pose of the robot on the map frame

      $W = \{X_{map,0}^{base\_link}, X_{map,1}^{base\_link}, \cdots, X_{map,T}^{base\_link}\}$
    - Smooth feasible path
  - Sampling feasible nodes

- Heuristic value of the node
  - For selecting the goal node, you can add heuristic value to each node.
    - Location, surrounding environment, etc.



https://www.youtube.com/watch?v=SXe_FJpxv94

Path generation using the trajectory

Visual SLAM Practice (ROS)

ORB SLAM for Mobile robot

- There are 3 options for ORB SLAM input
  - Monocular camera image
  - Stereo camera image
  - Depth camera image

- In this class, we will use **depth camera image input**
  - Less costs less computation compared to stereo
  - Monocular camera input has scale ambiguity* problem

    * Cannot know the scale of the result

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 1. Download ORB-SLAM3 source:

- Download ROS implemented version
- Please install ORB-SLAM3 in ROS Melodic environment
  - Noetic is also possible, but It will be really hard to figure out what is wrong when error is occurred during installation!
- Please DO NOT put this package inside catkin workspace(ex. Catkin_ws, etc)

```
(for melodic) git clone –b jetson451 https://github.com/yongeePark/ORB_SLAM3_ROS_Interface.git
(for noetic)   git clone –b jetson_51 https://github.com/yongeePark/ORB_SLAM3_ROS_Interface.git
```

❑ 2. Install Pangolin

- This is not a ROS package, so do not put this package inside catkin workspace
- Reference(Korean) : https://yongee.tistory.com/30

```
git clone --recursive https://github.com/stevenlovegrove/Pangolin.git
Cd Pangolin
./scripts/install_prerequisites.sh recommended
mkdir build && cd build
cmake .. && make –j4 && sudo make install
ldconfig
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 3.4.1(1) – for ROS Melodic

- Reference : https://yongee.tistory.com/111
- If you are in ROS noetic environment, please install opencv **4.5.5**
  - The instruction of installation of opencv 4.5.5 will be later
- Install dependencies

```
sudo apt-get update
sudo apt-get install -y build-essential cmake pkg-config
sudo apt-get install -y libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev libx264-dev libxine2-dev
sudo apt-get install -y libv4l-dev v4l-utils
sudo apt-get install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
sudo apt-get install -y libqt4-dev
sudo apt-get install -y mesa-utils libgl1-mesa-dri libqt4-opengl-dev
sudo apt-get install -y libgtkgl2.0-dev libgtkglext1-dev
sudo apt-get install -y libatlas-base-dev gfortran libeigen3-dev
sudo apt-get install -y libgtk-3-dev
sudo apt-get install -y python2.7-dev python3-dev python-numpy python3-numpy
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 3.4.1(2)

- Download and unzip source

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/3.4.1.zip
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/3.4.1.zip
unzip opencv.zip
unzip opencv_contrib.zip
```

- Before building please edit a file related to cuda

```
sudo gedit /usr/local/cuda/include/cuda_gl_interop.h
```

- Go to line 61, and comment nearby lines except “#include <GL/gl.h>”
- The result should be same as below

```
//#if defined(__arm__) || defined(__aarch64__)
//#ifndef GL_VERSION
//#error Please include the appropriate gl headers before including cuda_gl_interop.h
//#endif
//#else
#include <GL/gl.h>
//#endif
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 3.4.1(3)

- Open bashrc, and add these lines at the end of the file

```
#gedit ~/.bashrc
export PATH="/usr/local/cuda-10.2/bin:$PATH“
export LD_LIBRARY_PATH="/usr/local/cuda-10.2/lib64:$LD_LIBRARY_PATH"
```

- Build opencv

```
cd opencv-3.4.1
mkdir build && cd build
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 3.4.1(4)

- Build opencv

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D WITH_TBB=OFF \
-D WITH_IPP=OFF \
-D WITH_1394=OFF \
-D BUILD_WITH_DEBUG_INFO=OFF \
-D BUILD_DOCS=OFF \
-D INSTALL_C_EXAMPLES=ON \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D BUILD_EXAMPLES=OFF \
-D BUILD_TESTS=OFF \
-D BUILD_PERF_TESTS=OFF \
-D WITH_QT=ON \
-D WITH_CUDA=ON \
-D CUDA_ARCH_BIN=7.2 \
-D WITH_CUBLAS=1 \
-D WITH_CUDNN=ON \
-D CUDNN_VERSION='8.0' \
-D ENABLE_FAST_MATH=1 \
-D CUDA_FAST_MATH=1 \
-D WITH_TBB=ON \
-D WITH_OPENGL=ON \
-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-3.4.1/modules \
-D WITH_V4L=ON \
-D WITH_FFMPEG=ON \
-D WITH_XINE=ON \
-D WITH_NVCUVID=OFF \
-D PYTHON3_PACKAGES_PATH=/usr/lib/python3/dist-packages \
-D OPENCV_GENERATE_PKGCONFIG=ON \
-D BUILD_NEW_PYTHON_SUPPORT=ON \../
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 3.4.1(5)

- Build opencv

```
make –j4

rm -rf /usr/include/opencv4/opencv2
sudo make install
sudo ldconfig
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 4.5.5(1) – for ROS Noetic

- Reference : https://yongee.tistory.com/31
- Install dependencies

```
sudo apt update
sudo apt-get install build-essential

sudo apt-get install -y libjpeg-dev libtiff5-dev libpng-dev \ libgtk2.0-dev libvtk6-dev libopenexr-dev libtbb-dev
\ libavcodec-dev libavformat-dev libswscale-dev \ libxvidcore-dev libx264-dev libxine2-dev \ libv4l-dev v4l-utils
\ libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \ mesa-utils libgl1-mesa-dri \ libgtkgl2.0-dev libgtkglext1-dev
\ libatlas-base-dev gfortran libeigen3-dev \ libgtk-3-dev python3-dev python3-numpy
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 4.5.5(2)

- Download and unzip source

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.5.zip
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.5.5.zip
unzip opencv.zip
unzip opencv_contrib.zip
```

- Before building please edit a file related to cuda

```
sudo gedit /usr/local/cuda/include/cuda_gl_interop.h
```

- Go to line 61, and comment nearby lines except “#include <GL/gl.h>”
- The result should be same as below

```
//#if defined(__arm__) || defined(__aarch64__)
//#ifndef GL_VERSION
//#error Please include the appropriate gl headers before including cuda_gl_interop.h
//#endif
//#else
#include <GL/gl.h>
//#endif
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 4.5.5(3)

- Open bashrc, and add these lines at the end of the file
- Please change cuda-10.2 to your actual cuda version

```
#gedit ~/.bashrc
export PATH="/usr/local/cuda-10.2/bin:$PATH“
export LD_LIBRARY_PATH="/usr/local/cuda-10.2/lib64:$LD_LIBRARY_PATH"
```

- Build opencv

```
cd opencv-4.5.5
mkdir build && cd build
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 4.5.5(4)

- Build opencv

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
 -D CMAKE_INSTALL_PREFIX=/usr/local \
 -D WITH_TBB=OFF -D WITH_IPP=OFF -D WITH_1394=OFF \
 -D BUILD_WITH_DEBUG_INFO=OFF \
 -D BUILD_DOCS=OFF -D BUILD_EXAMPLES=OFF \
 -D BUILD_TESTS=OFF -D BUILD_PERF_TESTS=OFF \
 -D WITH_CUDA=ON -D WITH_CUDNN=ON \
 -D OPENCV_DNN_CUDA=ON \
 -D CUDA_FAST_MATH=ON -D CUDA_ARCH_BIN=7.2 \
 -D WITH_CUBLAS=ON -D WITH_CUFFT=ON \
 -D WITH_QT=ON -D WITH_GTK=OFF \
 -D WITH_OPENGL=ON -D WITH_V4L=ON \
 -D WITH_FFMPEG=ON -D WITH_XINE=ON \
 -D BUILD_NEW_PYTHON_SUPPORT=ON \
 -D BUILD_opencv_python3=ON \
 -D INSTALL_C_EXAMPLES=OFF -D INSTALL_PYTHON_EXAMPLES=ON \
 -D OPENCV_GENERATE_PKGCONFIG=ON \
 -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-4.5.5/modules \
 -D OPENCV_PYTHON3_INSTALL_PATH=$(python3 -c "from distutils.sysconfig import get_python_lib; print(get_python_lib())") \
 -D PYTHON3_EXECUTABLE=/usr/bin/python3 \
 -D PYTHON3_INCLUDE_DIR=$(python3 -c "from distutils.sysconfig import get_python_inc; print(get_python_inc())") \
 -D PYTHON3_PACKAGES_PATH=$(python3 -c "from distutils.sysconfig import get_python_lib; print(get_python_lib())") \
 -D OPENCV_ENABLE_NONFREE=ON -D BUILD_EXAMPLES=OFF ..
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 3. Install OpenCV 4.5.5(5)

- Build opencv

```
make –j4

rm -rf /usr/include/opencv4/opencv2
sudo make install
sudo ldconfig
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 4. Install PCL-ROS

```
(for melodic) sudo apt-get install ros-melodic-pcl-ros
(for noetic)    sudo apt-get install ros-noetic-pcl-ros
```

❑ 5. Install Realsense sdk

- Reference : https://yongee.tistory.com/117
- Download source

```
(for melodic)
wget https://github.com/IntelRealSense/librealsense/archive/refs/tags/v2.48.0.zip
unzip v2.48.0.zip

(for noetic)
wget https://github.com/IntelRealSense/librealsense/archive/refs/tags/v2.51.1.zip
unzip v2.51.1.zip
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 5. Install Realsense sdk

- Build and Install

```
sudo apt-get install libssl-dev
cd [librealsense_dir]
mkdir build && cd build

cmake .. \
 -DFORCE_RSUSB_BACKEND=true \
 -DBUILD_WITH_CUDA=true \
-DCMAKE_BUILD_TYPE=release

make -j4
sudo make install
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 6. Build ORB-SLAM3

- Use build script

```
cd [ORB-SLAM3_Package]
./build.sh
```

❑ 7. Build ROS Package

- Add below line to your bashrc file

```
#gedit ~/.bashrc
export ROS_PACKAGE_PATH="[ORB_SLAM3_Install_PATH]/ORB_SLAM3_ROS_Interface/Examples/orb3_ros_interface:$ROS_PACKAGE_PATH"
```

- Go to the orb3_ros_interface directory and build the ros package

```
cd Examples/orb3_ros_interface/
mkdir build && cd build
cmake ..
make
```

ORB SLAM ROS

❖ Installation of ORB-SLAM3

❑ 8. Run ORB-SLAM3

- Please edit pack path in rgbd.launch to your actual path(default : /home/usrgtest)

```
roscd orb3_ros_interface/launch
gedit rgbd.launch
```

- Run ORB-SLAM3

```
roslaunch orb3_ros_interface rgbd.launch
```

❑ If “what(): failed to set power state” Error happens :

- Reference : https://yongee.tistory.com/152
- Add file in udev rules

```
sudo gedit /etc/udev/rules.d/99-realsense-libusb.rules
```

- Go to the below link, and the copy the contents to above file
- https://github.com/IntelRealSense/librealsense/blob/master/config/99-realsense-libusb.rules

Programming Assignment

- Visual SLAM and Global path generation

Due: 11.27

1. Generating map using ORB SLAM3 **(40%)**
   - https://github.com/yongeePark/ORB_SLAM3_ROS_Interface.git
   - Run ORB SLAM3 and generate a map of N5 2354. (It doesn't have to be autonomous driving mode while generating the map.)
   - Submit a YouTube link to the video of the result
   - You can share the result with your teammates
2. Generate node-link waypoints using localization results **(40%)**
   - Generate the node-link waypoints of N5 2354 map.
   - Example code for the node-link generation will be given by next Tuesday.
   - Submit a YouTube link to the video of the trajectory generation on the map. (30%)
   - Write a simple explanation on how you generated the node-link (10%)
   - You cannot share the result with your teammates
     - Recommend you use **rosbag** to record the SLAM (localization) result + sensor data and share it with your team so that each member can work on the node-link generation separately.
3. Discuss what the heuristics you can use for selecting the goal point/node for the exploration **(20%)**
   - Come up with **2 heuristics** that can be used for the exploration.
   - Provide **maximum three sentences or equations for each**.
   - You cannot share your answer with your teammates.

- Submit a PDF/Word file named EE405A_localization_[Student ID]_[Full name].

Weekly Progress Report

- ORB SLAM

Due: 11.20

  - Install ORB SLAM on your onboard computer

  - Run ORB SLAM with Realsense D435 using depth image in N5 2354

Q & A

fingb20@kaist.ac.kr