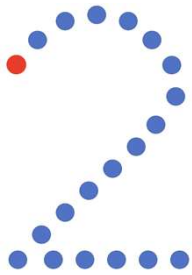
SEMTRON

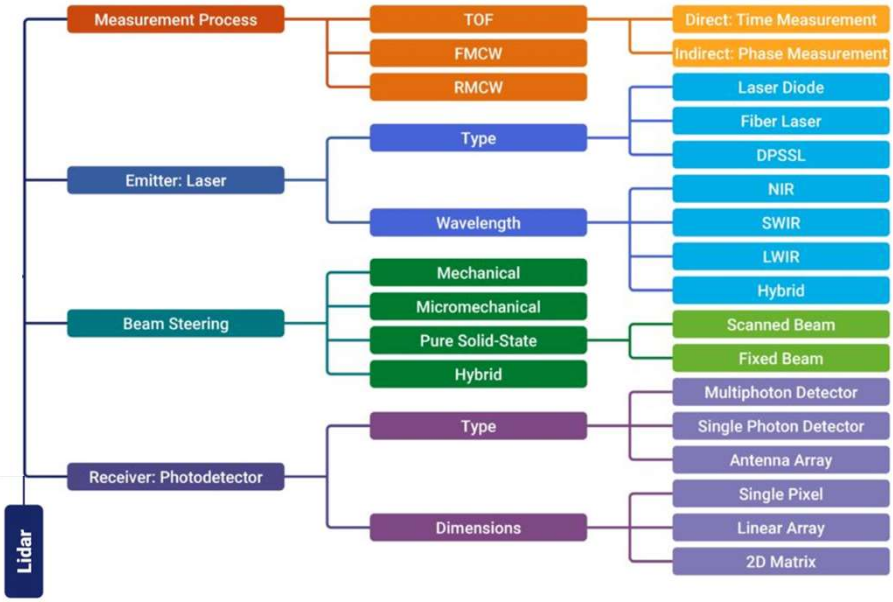
TWO

Robot Sensors

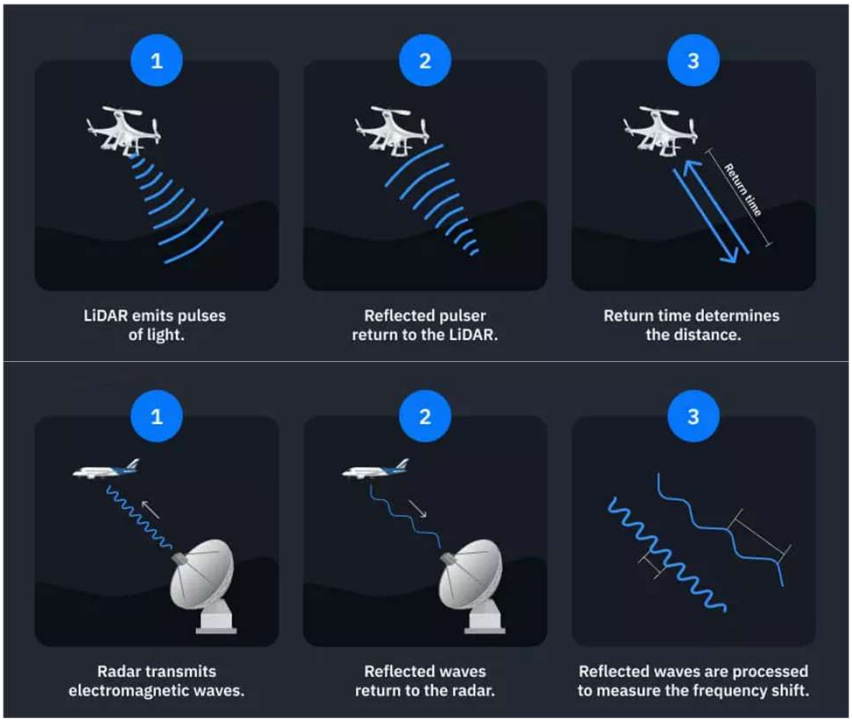


SEMTRON

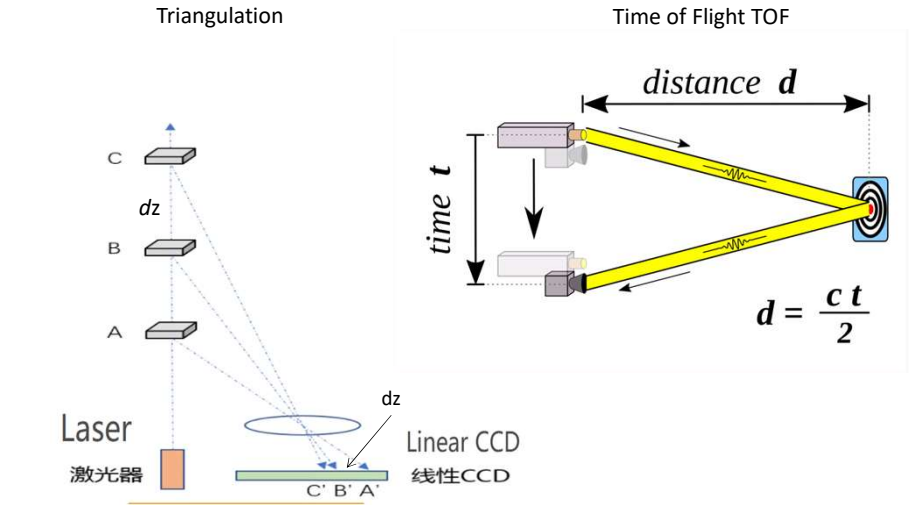
YDLIDAR X3/X3 Pro Lidar TOF 360°



Lidar for advanced driver assistance systems

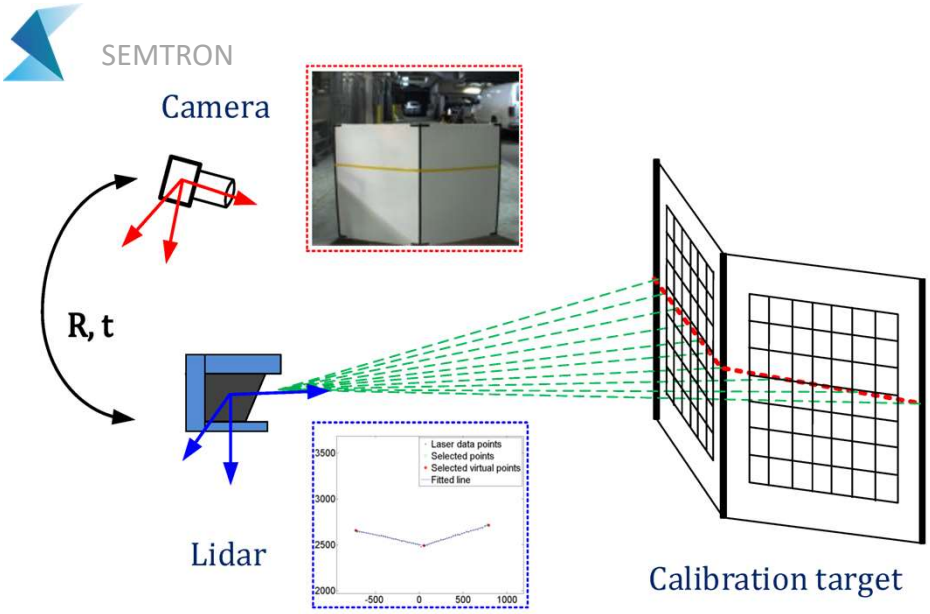


SEMTRON



Principle of LiDAR range finder

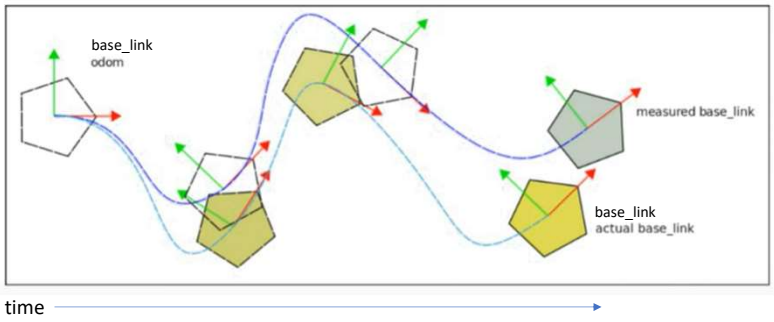
SEMTRON



Calibration of LiDAR odometry

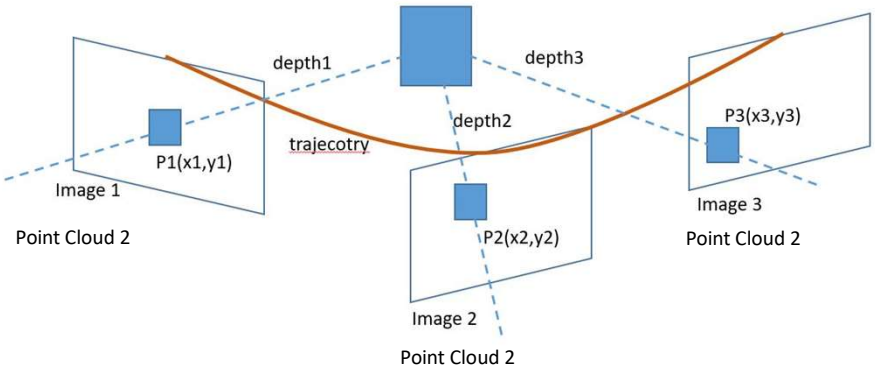
SEMTRON

Odometry is the use of data from motion (odometry) sensors to estimate change in position over time. It is used to estimate robots position relative to a starting location. This method is sensitive to errors due to the integration of velocity measurements over time to give position estimates.



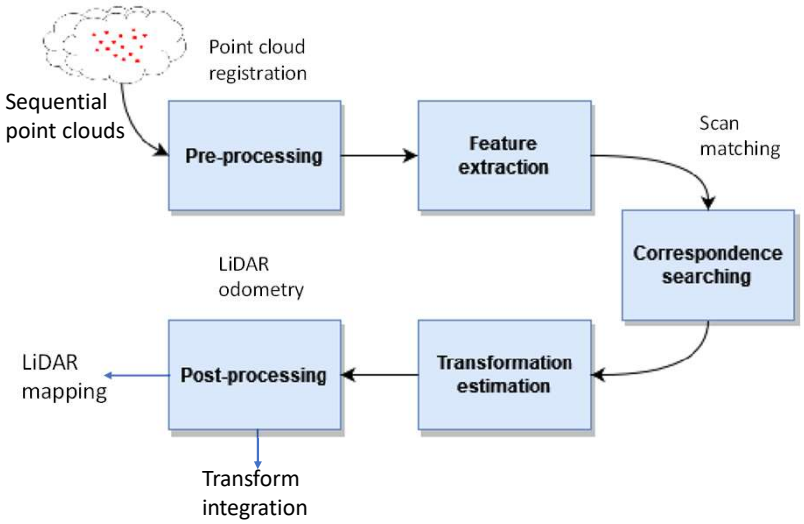
Odometry 里程計

SEMTRON



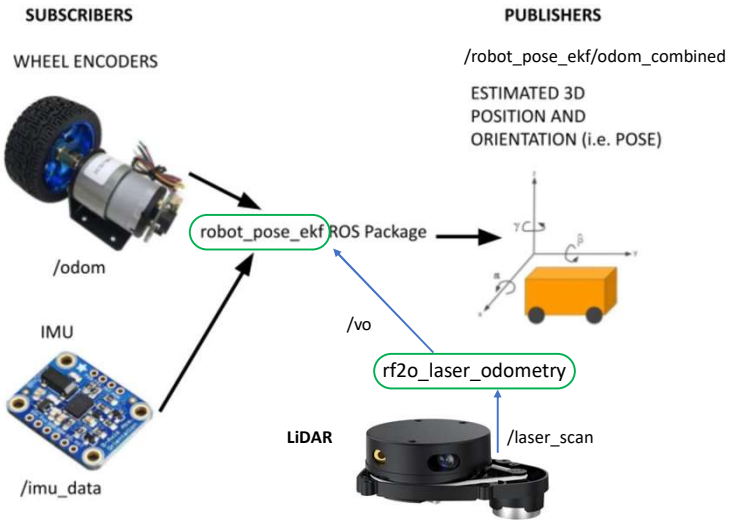
Egomotion estimations in visual odometry

SEMTRON



LiDAR-based odometry pipeline

SEMTRON



Odometry in ROS

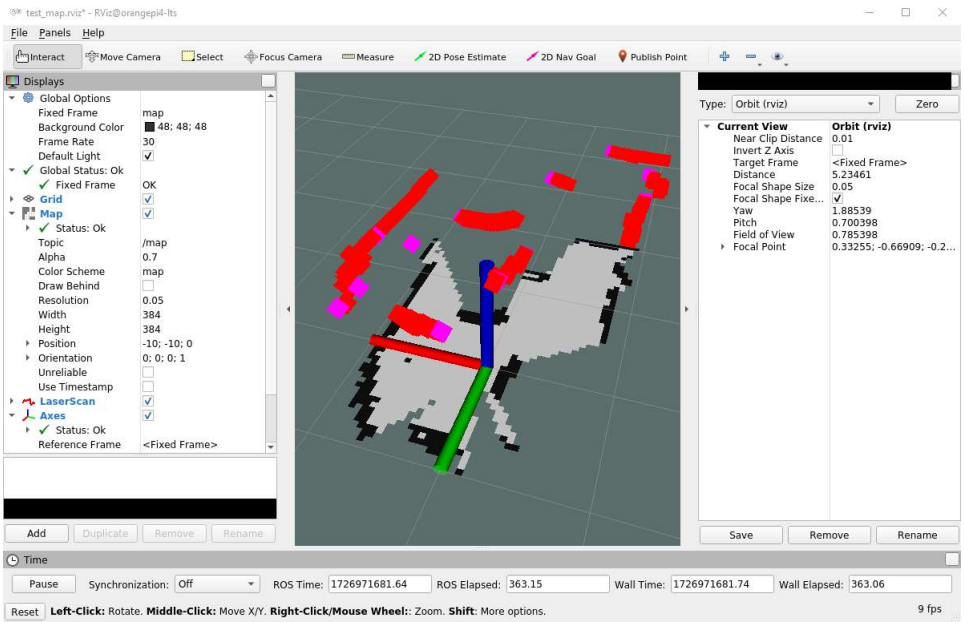
SEMTRON

```
fi
export LIDAR_TYPE=x3     #x3,4ros
echo -e "LIDAR_TYPE: \033[32m$LIDAR_TYPE\033[0m"
source /opt/ros/melodic/setup.bash
source ~/ydlidar_ws/devel/setup.bash --extend
```

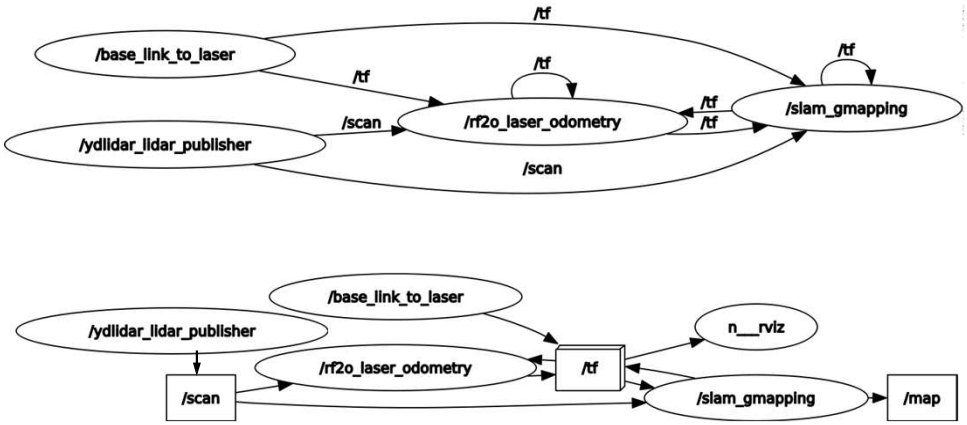
$ echo “export LIDAR_TYPE=x3" >> ~/.bashrc

$ sudo apt install ros-noetic-slam-gmapping
$ scp @:rf2o_laser_odometry ~/ros_test/src/rf2o_laser_odometry
$ cd ros_test && catkin_make
$ roslaunch ydlidar_ros_driver test_gmapping.launch

Range flow 2D odom

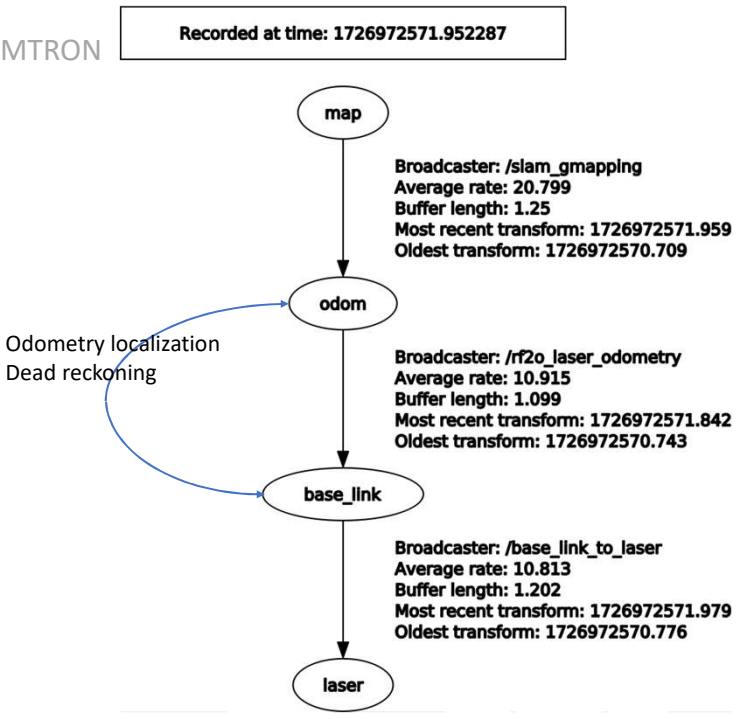


SEMTRON

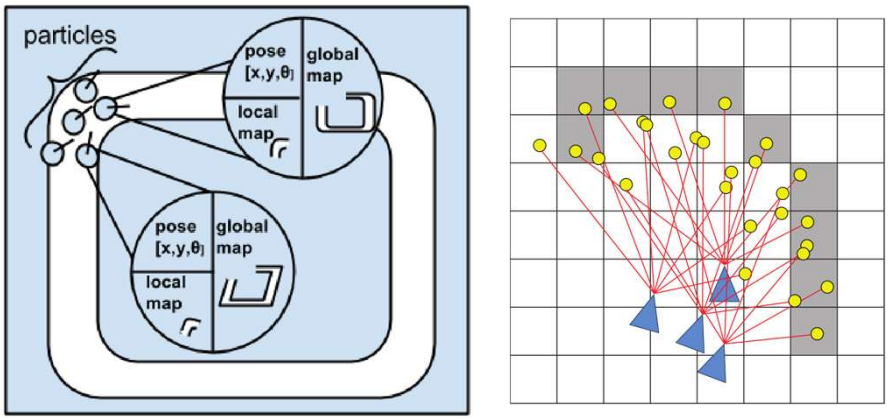


Range flow 2D odom

SEMTRON



SEMTRON

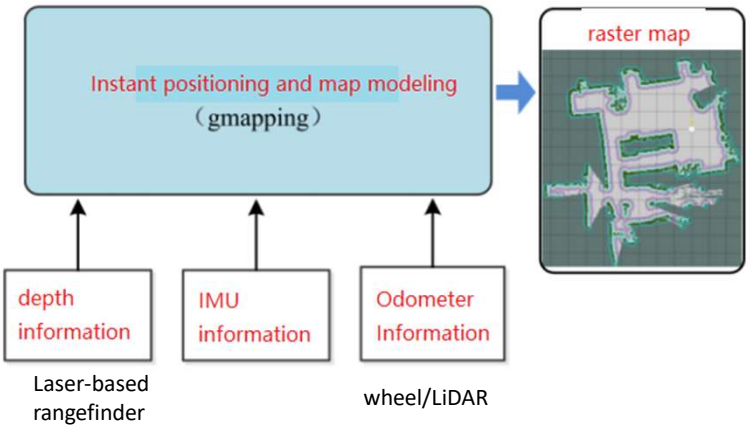


Grid-based scan-to-scan matching

SEMTRON

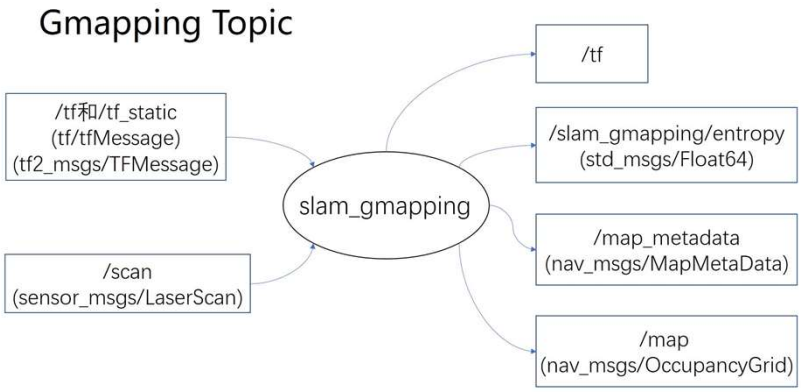
Grid-based SLAM with Rao-Blackwellized Particle Filters, RBPF by Adaptive Proposals and Selective Resampling

SEMTRON



GMapping SLAM

SEMTRON



GMapping SLAM

Parameters	Default value	Description
Group (1): Coordinate system parameters		
base_frame	Base link frame	The coordinate system of the mobile robot platform
odom_frame	Odometry frame	
map_frame	Map frame	
transform_publish_period		The publishing rate of data among frames
Group (2): OGM parameters		
xmin	-100 m	The initial map dimensions represented a rectangular region defined by two opposite corners
ymin	-100 m	
xmax	100 m	
ymax	100 m	
delta	0.05 m	The length of one side of the grid cell in meters
occ_thresh	0.25	The threshold is used to determine whether a cell is an obstacle or not.
Group (3): Measurement Update Parameters		
linearUpdate	1.0 m	The translational and rotational threshold of measurement update
angularUpdate	0.5 rad	
temporalUpdate	-1.0	The time threshold of the Measurement update

Group (4): Motion model parameters		
srr	0.1	Covariance of sampling motion noise
str	0.1	
str	0.2	
stt	0.2	
Group (5.1): Scan matching and LiDAR parameters		
maxRange	-	LiDAR maximum distance range
maxUrange	-	The usable LiDAR range
throttle_scans	1	The number of throttle scans
sigma	0.05	the Standard deviation (σ) of the score and likelihood functions of a single beam of the LiDAR
lsigma	0.075	
kernelSize	1	The size of the kernel's search window for matching scans
lstep	0.05 m	Linear and Angular displacement step increment size
astep	0.05 m	
iterations	1	The number of refinement steps used in the scan matches
lskip	0	The number of beams skipped in each scan

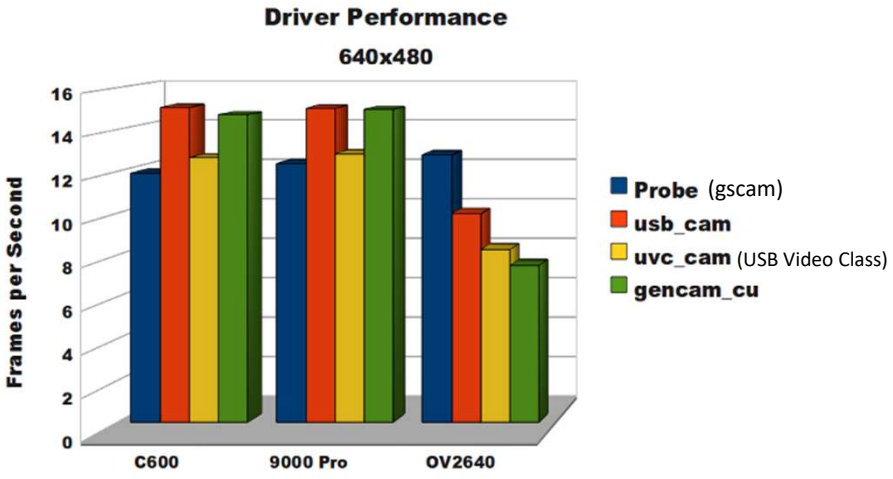
Parameters	Default value	Description
Group (5.2): Likelihood sampling parameters		
llsamplerange	0.01	Translation sampling range of scan matching
llsamplestep	0.01	Translation sampling step of scan matching
lasamplerange	0.005	Rotational sampling range of scan matching
lasamplestep	0.005	Rotational sampling step of scan matching
Group (6): PF parameters		
particles	30	the number of particles
map_update_interval	5.0	the map update period
minimumScore	0	The lowest score that a scan-matching algorithm must achieve for the outcome of the scan to be considered valid
ogain	3.0	the gain of weight normalizer function for smoothing the resampling step
resampleThreshold	0.5	Resampling threshold

GMapping parameters

SEMTRON

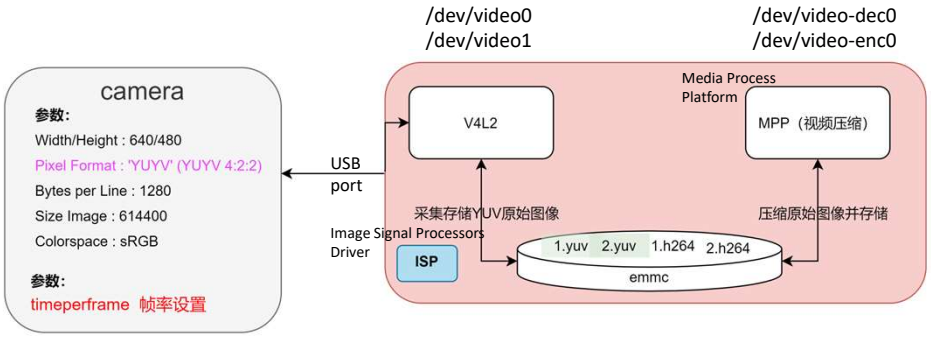
Microdia KD-USB Camera 1280/720

SEMTRON



ROS USB camera driver

SEMTRON



Format Video Capture:
Crop Capability Video Capture:
Selection Video Capture:
Streaming Parameters Video Capture:

ROS USB camera driver

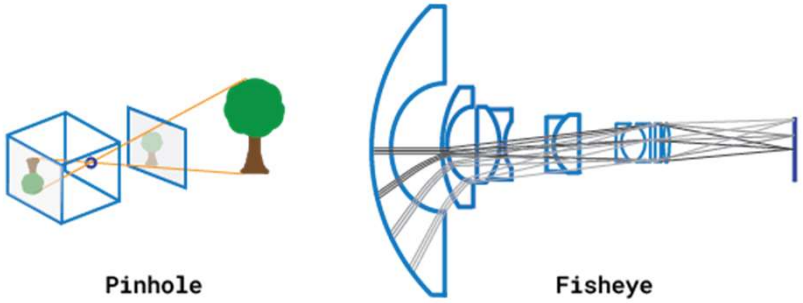
SEMTRON

```
$ v4l2-ctl -d /dev/video0 --all
$ v4l2-ctl --list-ctrls --device /dev/video0
$ v4l2-ctl -d 0 --list-formats-ext

$ roslaunch usb_cam usb_cam-test.launch
$ rosrun rqt_image_view rqt_image_view

$ rosrun camera_calibration cameracalibrator.py --size 8x6 --square 0.108
image:=/usb_cam/image_raw camera:=/usb_cam
```

http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration

SEMTRON

$$\text{Horiz FoV} = 2\tan^{-1}\frac{sensor_{width}}{2EFL}$$



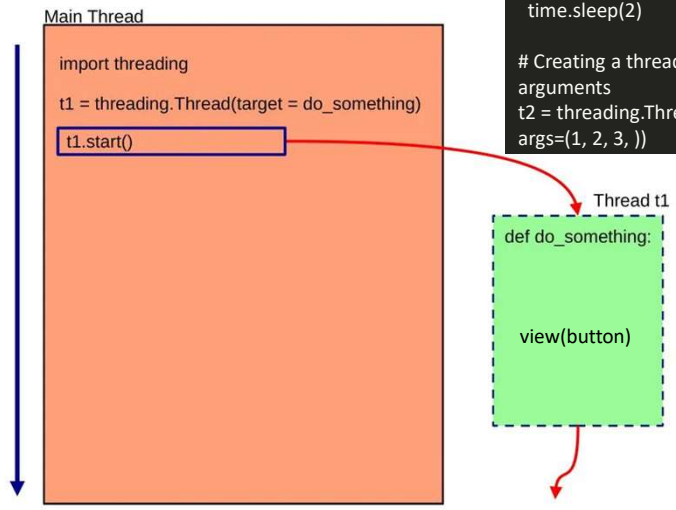
Camera horizontal FOV

SEMTRON

```
Streaming Parameters Video Capture:
     Capabilities     : timeperframe
     Frames per second: 30.000 (30/1)
     Read buffers     : 0
                brightness 0x00980900 (int)    : min=-64 max=64 step=1 default=0 value=0
                contrast 0x00980901 (int)    : min=0 max=95 step=1 default=4 value=4
                saturation 0x00980902 (int)    : min=0 max=100 step=1 default=72 value=72
                hue 0x00980903 (int)    : min=-2000 max=2000 step=100 default=0 value=0
  white_balance_temperature_auto 0x0098090c (bool)   : default=1 value=1
                gamma 0x00980910 (int)    : min=100 max=300 step=1 default=100 value=100
  power_line_frequency 0x00980918 (menu)   : min=0 max=2 default=1 value=1
                        0: Disabled
                        1: 50 Hz
                        2: 60 Hz
  white_balance_temperature 0x0098091a (int)    : min=2800 max=6500 step=1 default=4600 value=4600 flags=inactive
  sharpness 0x0098091b (int)    : min=1 max=7 step=1 default=2 value=2
  backlight_compensation 0x0098091c (int)    : min=0 max=3 step=1 default=0 value=0
  exposure_auto 0x009a0901 (menu)   : min=0 max=3 default=3 value=3
                        1: Manual Mode
                        3: Aperture Priority Mode
  exposure_absolute 0x009a0902 (int)    : min=78 max=10000 step=1 default=312 value=312 flags=inactive
  focus_absolute 0x009a090a (int)    : min=0 max=15 step=1 default=0 value=0 flags=inactive
  focus_auto 0x009a090c (bool)   : default=1 value=1
```

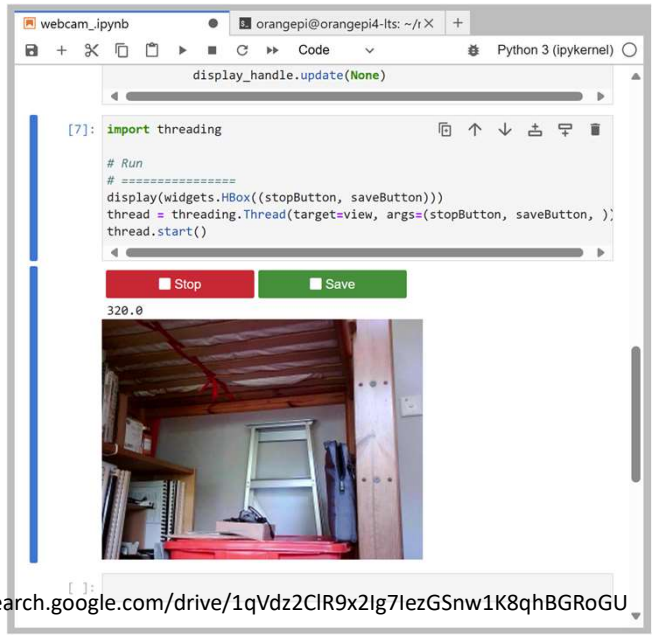
SEMTRON

```
# You can also pass multiple arguments to your function
def do_something(myarg1, myarg2, myarg3):
  print(myarg1, myarg2, myarg3)
  time.sleep(2)

# Creating a thread and passing multiple arguments
t2 = threading.Thread(target=do_something, args=(1, 2, 3, ))
```



Programming Python threading

SEMTRON



https://colab.research.google.com/drive/1qVdz2ClR9x2Ig7IezGSnw1K8qhBGRoGU