Vision I : Classic Methods

Zirui Zang and the F1TENTH Team
Contact: rahulm@seas.upenn.edu

Portions of these slides are from Penn Course MEAM620, CIS680

Vision Module Overview

Lecture I : Classic Methods

- Vision Hardware
- Accessing Camera on Linux
- Camera Model & Distance Estimation
- Useful OpenCV Functions
- Visual SLAM

Lecture II : DL Methods

- Deep Learning Basics
- Object Detection w/ Image - YOLO (2015)
- Object Detection w/ Pointcloud - Pointpillars (2018)
- Transformer for Vision (2020)
- Network Deployment

Vision Module Overview

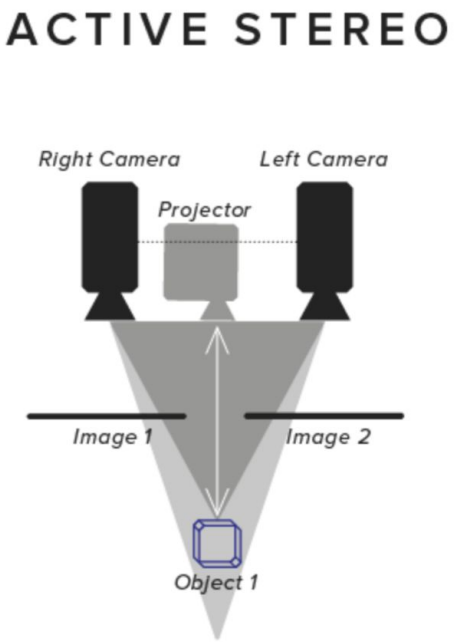
Lecture 1 : Classic Methods

- Vision Hardware
- Accessing Camera on Linux
- Camera Model & Distance Estimation
- Useful OpenCV Functions
- Visual SLAM

Vision Hardware

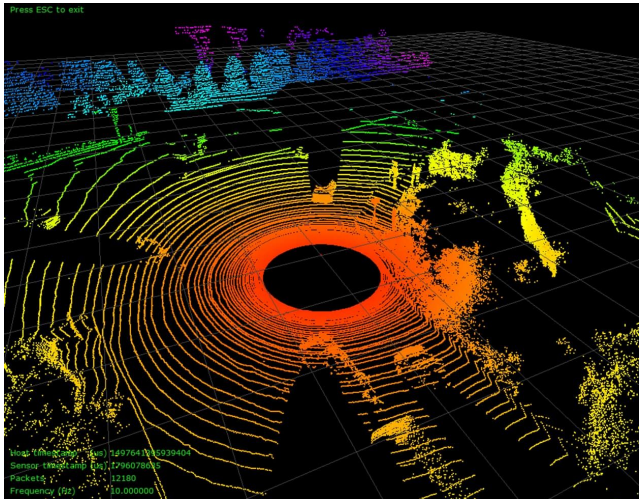
Vision Sensors - Cameras

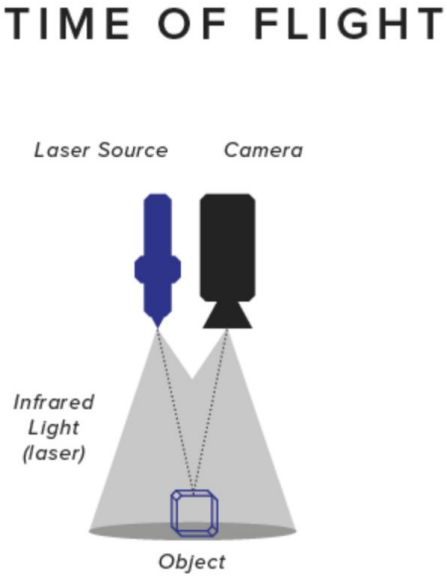
- RGB Camera
- Event-based Camera
- Depth Camera
  - Stereo Camera
  - Structural Light Camera
  - Active Stereo (Combining both)



Vision Sensors - Lidar

- Outputs: xyz coordinates + intensity
- Advantage:
  - Reliable, Accurate
- Disadvantage:
  - Expensive, Not usable in rain

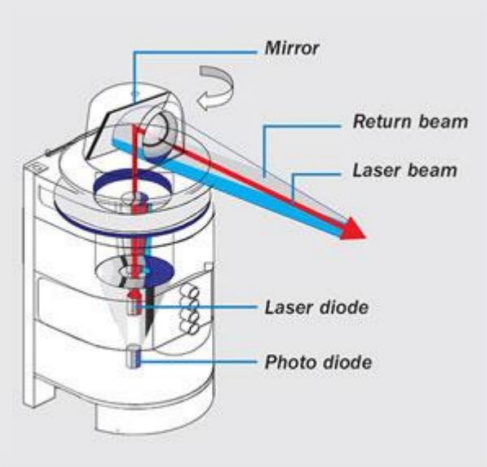


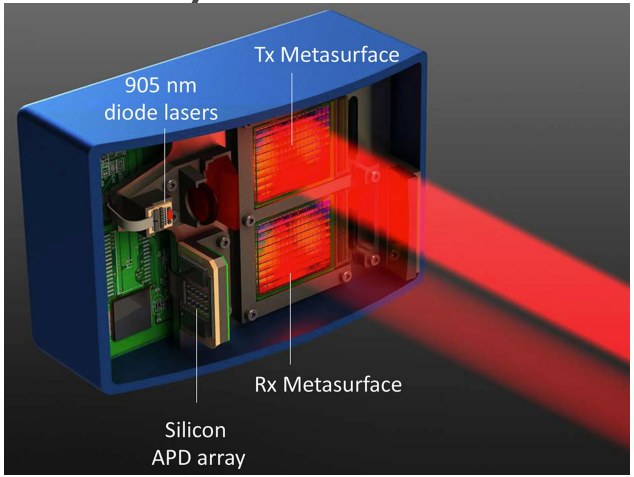


$$\text{Distance} = c(t_{\text{return}} - t_{\text{emit}})/2$$

Vision Sensors - Lidar types

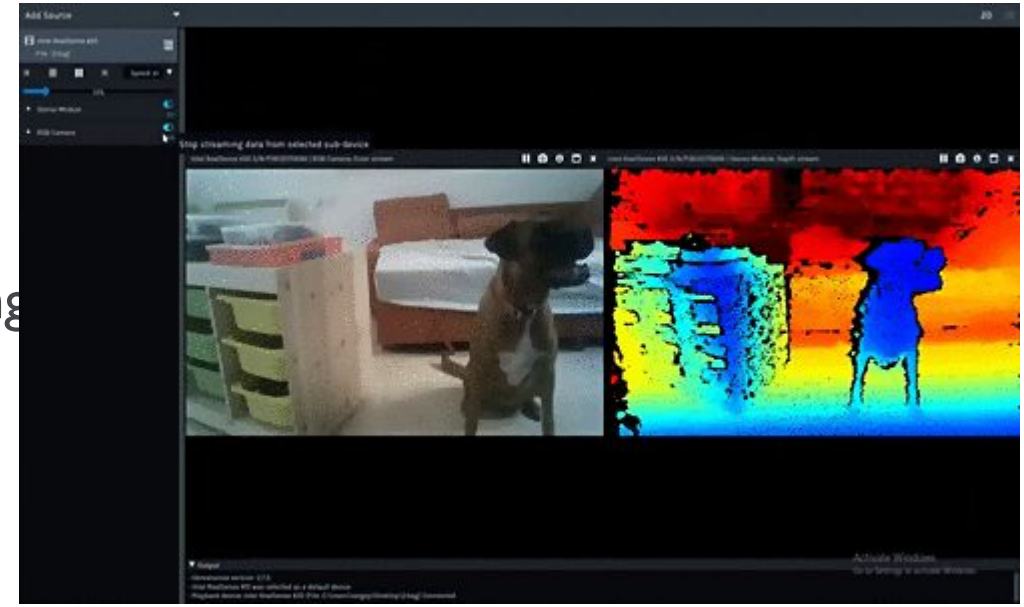
- Mechanical Lidar vs. Solid State Lidar
  - Moving parts, Scanning speed, Beam density





Intel Realsense D435i

- Active depth up to 3m.
- Output image, depth map and point cloud.
- Supports most libraries, including OpenCV and ROS2.



Let's read its spec.

And see why do they provide these spec.

	D415	D435	D455
Use Environment	Indoor/Outdoor		
Depth Technology	Active infrared (IR) stereo		
IR Projector & Left/Right Camera Type	Standard	Wide	
Shutter Type	Rolling		Global
Image Sensor Module	OV2740 (OV02740-H34A-Z)[34]		OV9782 (OV09782-GA4A)[35]
Image Sensor Technology	PureCel® HDR[34]		OmniPixel®3-GS[35]
Image Sensor Size	1⁄6 inch, 3855µm × 2919µm[34]		1⁄4 inch, 3896µm × 2453µm[35]
Image Sensor Pixel Size	1.4µm × 1.4µm[34]		3µm × 3µm[35]
Vision Processor Board	RealSense Vision Processor D4		
Depth Sensor Module	RealSense Module D415	RealSense Module D430 + RGB Camera	RealSense Module D450
Depth Field of View for HD	H:65°±2 V:40°±1 D:72°±2	H:87°±3 V:58°±1 D:95°±3	
Depth Field of View for VGA	H:50°±2 V:40°±1 D:61°±2	H:75°±3 V:62°±1 D:89°±3	
Depth Resolution and Framerate	Up to 1280px × 720px @ 90fps		
Minimum Depth Distance at Maximum Resolution	45cm / 17.7"	28cm / 11.0"	52cm / 20.5"
Depth Accuracy	<2% at 2.0m / 2.2yd		<2% at 4.0m / 4.4yd
Ideal and Maximum Range	0.5m to 3m / 0.6yd to 3.3yd	0.3m to 3m / 0.3yd to 3.3yd	0.6m to 6m / 0.3yd to 6.6yd
RGB Resolution, Framerate and Aspect Ratio	1920px × 1080px @ 30fps (16:9)		1280px × 800px @ 30fps (8:5)
RGB Field of View	H:69.4° V:42.5° D:77.0°		H:91.2° V:65.5° D:100.6°
RGB Lens Distortion	≤1.5%		
Device Dimensions	99mm × 20mm × 23mm	90mm × 25mm × 25mm	124mm × 26mm × 29mm
Connector	USB Type-C 3.1 Gen 1		

Camera Spec - Shutter Type

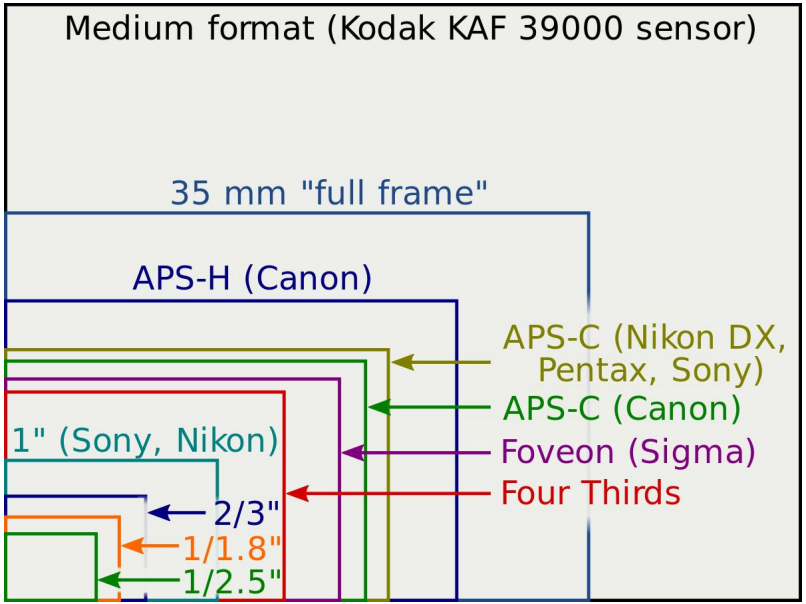
- Shutter Type
  - Rolling: Cheaper
  - Global: better for high speed objects

Rolling : Line by line pixel capture

	D415	D435	D455
Use Environment	Indoor/Outdoor		
Depth Technology	Active infrared (IR) stereo		
IR Projector & Left/Right Camera Type	Standard	Wide	
Shutter Type	Rolling		Global
Image Sensor Module	OV2740 (OV02740-H34A-Z)[34]		OV9782 (OV09782-GA4A)[35]
Image Sensor Technology	PureCel® HDR[34]		OmniPixel®3-GS[35]
Image Sensor Size	1⁄6 inch, 3855µm × 2919µm[34]		1⁄4 inch, 3896µm × 2453µm[35]
Image Sensor Pixel Size	1.4µm × 1.4µm[34]		3µm × 3µm[35]
Vision Processor Board	RealSense Vision Processor D4		
Depth Sensor Module	RealSense Module D415	RealSense Module D430 + RGB Camera	RealSense Module D450
Depth Field of View for HD	H:65°±2 V:40°±1 D:72°±2	H:87°±3 V:58°±1 D:95°±3	
Depth Field of View for VGA	H:50°±2 V:40°±1 D:61°±2	H:75°±3 V:62°±1 D:89°±3	
Depth Resolution and Framerate	Up to 1280px × 720px @ 90fps		
Minimum Depth Distance at Maximum Resolution	45cm / 17.7"	28cm / 11.0"	52cm / 20.5"
Depth Accuracy	<2% at 2.0m / 2.2yd		<2% at 4.0m / 4.4yd
Ideal and Maximum Range	0.5m to 3m / 0.6yd to 3.3yd	0.3m to 3m / 0.3yd to 3.3yd	0.6m to 6m / 0.3yd to 6.6yd
RGB Resolution, Framerate and Aspect Ratio	1920px × 1080px @ 30fps (16:9)		1280px × 800px @ 30fps (8:5)
RGB Field of View	H:69.4° V:42.5° D:77.0°		H:91.2° V:65.5° D:100.6°
RGB Lens Distortion	≤1.5%		
Device Dimensions	99mm × 20mm × 23mm	90mm × 25mm × 25mm	124mm × 26mm × 29mm
Connector	USB Type-C 3.1 Gen 1		

Camera Spec - Sensor Size & Pixel Size

- Sensor Size & Pixel Size
  - Bigger is better, more light gathering, more information.

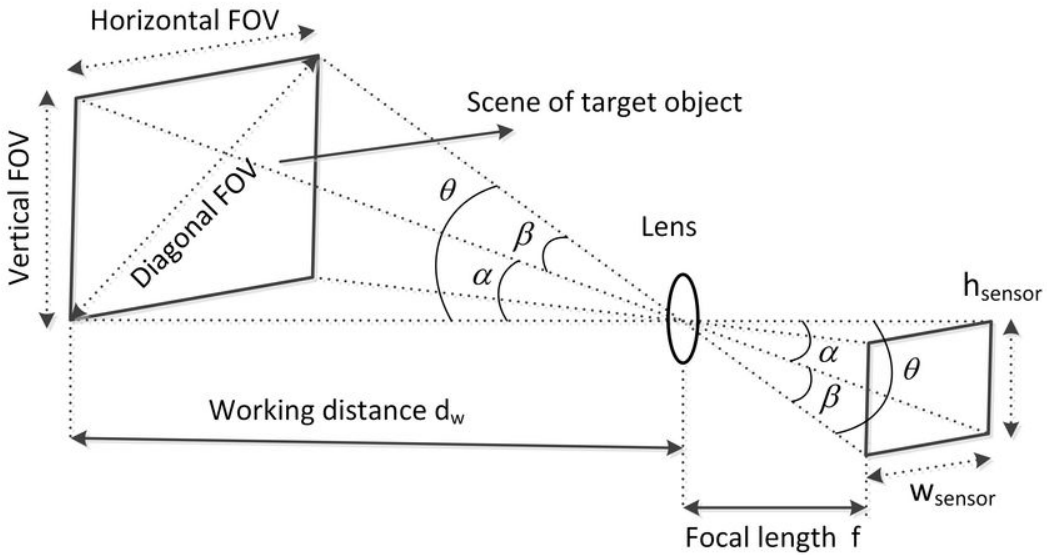


Common Sensor Formats

	D415	D435	D455
Use Environment	Indoor/Outdoor		
Depth Technology	Active infrared (IR) stereo		
IR Projector & Left/Right Camera Type	Standard	Wide	
Shutter Type	Rolling		Global
Image Sensor Module	OV2740 (OV02740-H34A-Z)[34]		OV9782 (OV09782-GA4A)[35]
Image Sensor Technology	PureCel® HDR[34]		OmniPixel®3-GS[35]
Image Sensor Size	1⁄6 inch, 3855µm × 2919µm[34]		1⁄4 inch, 3896µm × 2453µm[35]
Image Sensor Pixel Size	1.4µm × 1.4µm[34]		3µm × 3µm[35]
Vision Processor Board	RealSense Vision Processor D4		
Depth Sensor Module	RealSense Module D415	RealSense Module D430 + RGB Camera	RealSense Module D450
Depth Field of View for HD	H:65°±2 V:40°±1 D:72°±2	H:87°±3 V:58°±1 D:95°±3	
Depth Field of View for VGA	H:50°±2 V:40°±1 D:61°±2	H:75°±3 V:62°±1 D:89°±3	
Depth Resolution and Framerate	Up to 1280px × 720px @ 90fps		
Minimum Depth Distance at Maximum Resolution	45cm / 17.7"	28cm / 11.0"	52cm / 20.5"
Depth Accuracy	<2% at 2.0m / 2.2yd		<2% at 4.0m / 4.4yd
Ideal and Maximum Range	0.5m to 3m / 0.6yd to 3.3yd	0.3m to 3m / 0.3yd to 3.3yd	0.6m to 6m / 0.3yd to 6.6yd
RGB Resolution, Framerate and Aspect Ratio	1920px × 1080px @ 30fps (16:9)		1280px × 800px @ 30fps (8:5)
RGB Field of View	H:69.4° V:42.5° D:77.0°		H:91.2° V:65.5° D:100.6°
RGB Lens Distortion	≤1.5%		
Device Dimensions	99mm × 20mm × 23mm	90mm × 25mm × 25mm	124mm × 26mm × 29mm
Connector	USB Type-C 3.1 Gen 1		

Camera Spec - FOV

- Field of View (FOV)
  - FOV = arctan(sensor_size/focal_length * 2) * 2



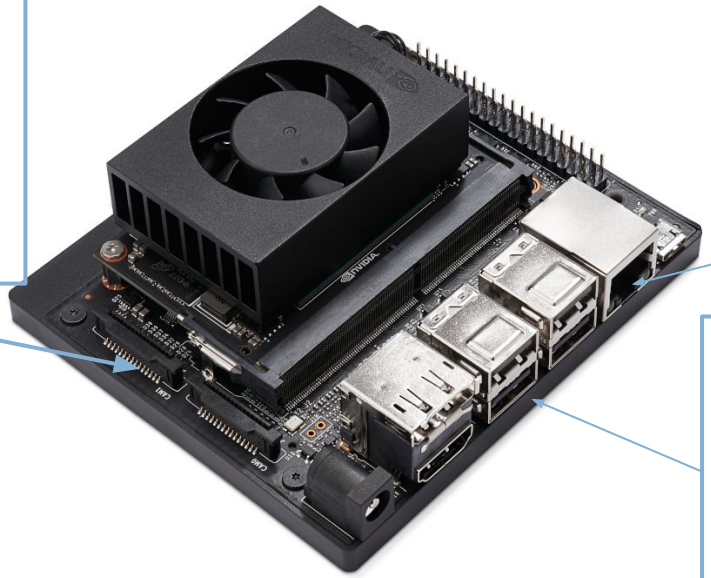
	D415	D435	D455
Use Environment	Indoor/Outdoor		
Depth Technology	Active infrared (IR) stereo		
IR Projector & Left/Right Camera Type	Standard	Wide	
Shutter Type	Rolling		Global
Image Sensor Module	OV2740 (OV02740-H34A-Z)[34]		OV9782 (OV09782-GA4A)[35]
Image Sensor Technology	PureCel® HDR[34]		OmniPixel®3-GS[35]
Image Sensor Size	1⁄6 inch, 3855µm × 2919µm[34]		1⁄4 inch, 3896µm × 2453µm[35]
Image Sensor Pixel Size	1.4µm × 1.4µm[34]		3µm × 3µm[35]
Vision Processor Board	RealSense Vision Processor D4		
Depth Sensor Module	RealSense Module D415	RealSense Module D430 + RGB Camera	RealSense Module D450
Depth Field of View for HD	H:65°±2 V:40°±1 D:72°±2	H:87°±3 V:58°±1 D:95°±3	
Depth Field of View for VGA	H:50°±2 V:40°±1 D:61°±2	H:75°±3 V:62°±1 D:89°±3	
Depth Resolution and Framerate	Up to 1280px × 720px @ 90fps		
Minimum Depth Distance at Maximum Resolution	45cm / 17.7"	28cm / 11.0"	52cm / 20.5"
Depth Accuracy	<2% at 2.0m / 2.2yd		<2% at 4.0m / 4.4yd
Ideal and Maximum Range	0.5m to 3m / 0.6yd to 3.3yd	0.3m to 3m / 0.3yd to 3.3yd	0.6m to 6m / 0.3yd to 6.6yd
RGB Resolution, Framerate and Aspect Ratio	1920px × 1080px @ 30fps (16:9)		1280px × 800px @ 30fps (8:5)
RGB Field of View	H:69.4° V:42.5° D:77.0°		H:91.2° V:65.5° D:100.6°
RGB Lens Distortion	≤1.5%		
Device Dimensions	99mm × 20mm × 23mm	90mm × 25mm × 25mm	124mm × 26mm × 29mm
Connector	USB Type-C 3.1 Gen 1		

Camera Spec - Connector

- Connector
  - Normal webcam is USB2.0, but why does Realsense use USB3.1?

Camera Serial Interface (CSI) Connector
- 10 Gbps / 4 lanes
- Dedicated bandwidth
- Direct connection to Image Signal Processor (ISP)

CSI Camera

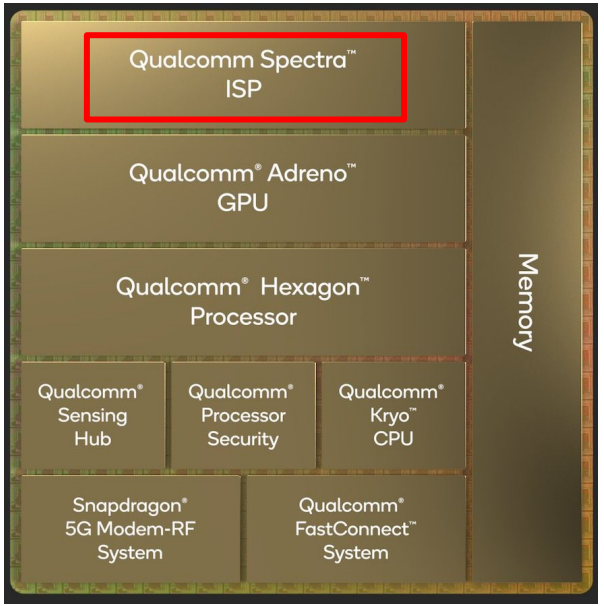


Ethernet
- Able to use Ethernet protocol
- Usually only 1Gbps

USB
- Easy to use
- Shared bandwidth of 5Gbps (USB 3.1)
- No direct connection to Image Signal Processor (ISP)

Image Signal Processor (ISP)

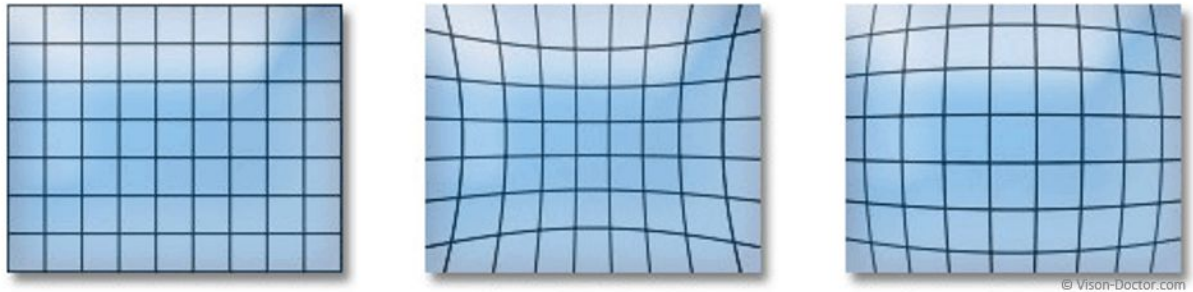
- A specialized chip for image processing. A more and more important part of a SoC chip.
- ISP can handle image denoise, resize, transform, filter, auto-exposure, auto-whitebalance, etc. faster than CPU or GPU.
- As for our Realsense, there is an onboard SoC.



new Snapdragon 8 Gen 1 chipset offers 18-bit triple-ISP design

	D415	D435	D455
Use Environment	Indoor/Outdoor		
Depth Technology	Active infrared (IR) stereo		
IR Projector & Left/Right Camera Type	Standard	Wide	
Shutter Type	Rolling		Global
Image Sensor Module	OV2740 (OV02740-H34A-Z)[34]		OV9782 (OV09782-GA4A)[35]
Image Sensor Technology	PureCel® HDR[34]		OmniPixel®3-GS[35]
Image Sensor Size	1⁄6 inch, 3855µm × 2919µm[34]		1⁄4 inch, 3896µm × 2453µm[35]
Image Sensor Pixel Size	1.4µm × 1.4µm[34]		3µm × 3µm[35]
Vision Processor Board	RealSense Vision Processor D4		
Depth Sensor Module	RealSense Module D415	RealSense Module D430 + RGB Camera	RealSense Module D450
Depth Field of View for HD	H:65°±2 V:40°±1 D:72°±2	H:87°±3 V:58°±1 D:95°±3	
Depth Field of View for VGA	H:50°±2 V:40°±1 D:61°±2	H:75°±3 V:62°±1 D:89°±3	
Depth Resolution and Framerate	Up to 1280px × 720px @ 90fps		
Minimum Depth Distance at Maximum Resolution	45cm / 17.7"	28cm / 11.0"	52cm / 20.5"
Depth Accuracy	<2% at 2.0m / 2.2yd		<2% at 4.0m / 4.4yd
Ideal and Maximum Range	0.5m to 3m / 0.6yd to 3.3yd	0.3m to 3m / 0.3yd to 3.3yd	0.6m to 6m / 0.3yd to 6.6yd
RGB Resolution, Framerate and Aspect Ratio	1920px × 1080px @ 30fps (16:9)		1280px × 800px @ 30fps (8:5)
RGB Field of View	H:69.4° V:42.5° D:77.0°		H:91.2° V:65.5° D:100.6°
RGB Lens Distortion	≤1.5%		
Device Dimensions	99mm × 20mm × 23mm	90mm × 25mm × 25mm	124mm × 26mm × 29mm
Connector	USB Type-C 3.1 Gen 1		

Distortions

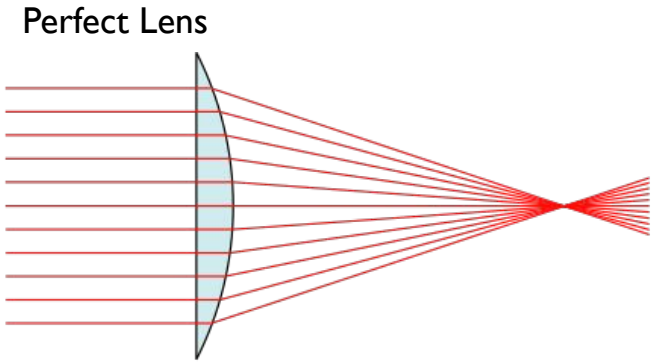


undistorted | pincushion distortion | barrel distortion

Usually correctable with software

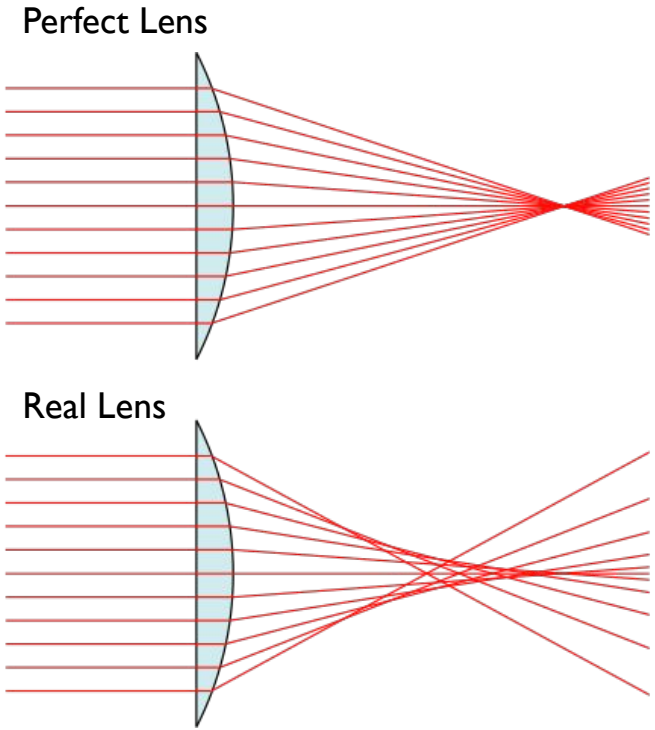
Lens Aberrations

- Lens are usually made with different elements to form a lens system. Why?

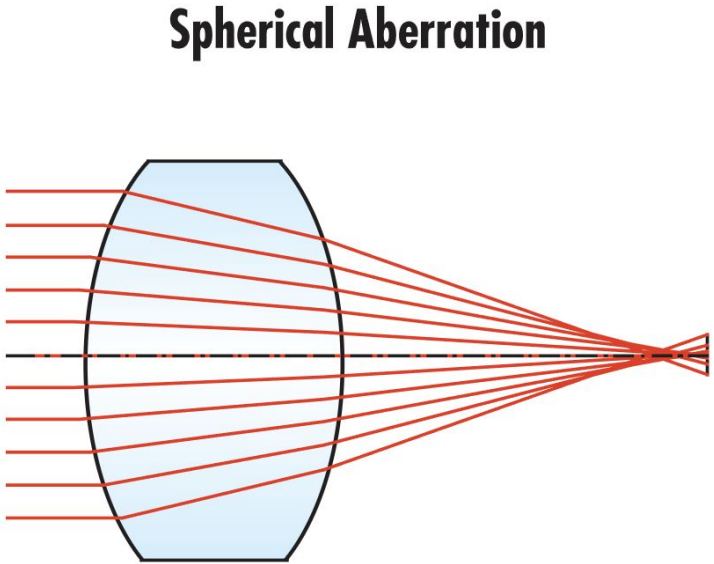


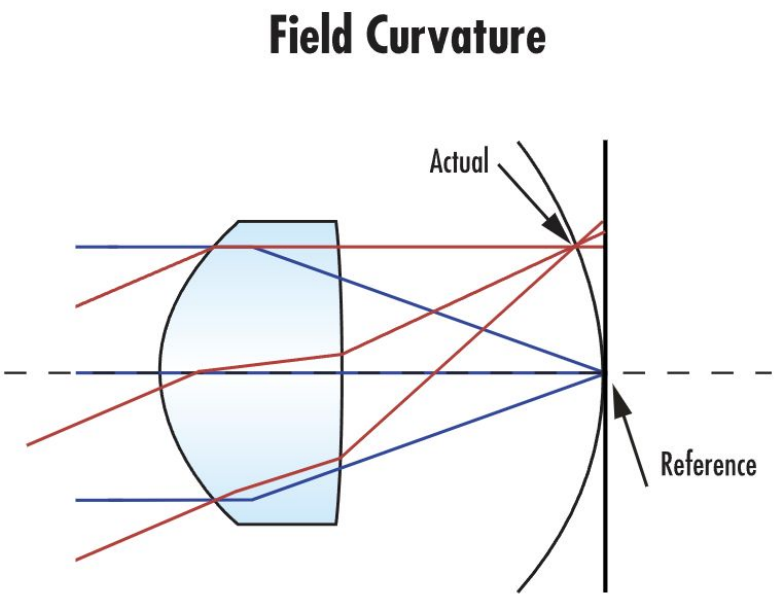
Lens Aberrations

- Lens are usually made with different elements to form a lens system. Why?
- Because lens have aberrations (causes blurry images)
- On-vehicle cameras usually have lens made with molded plastic, which makes them more susceptible to aberrations.



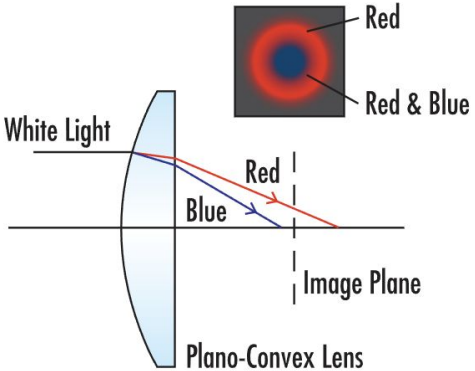
Spherical Aberrations & Field Curvature

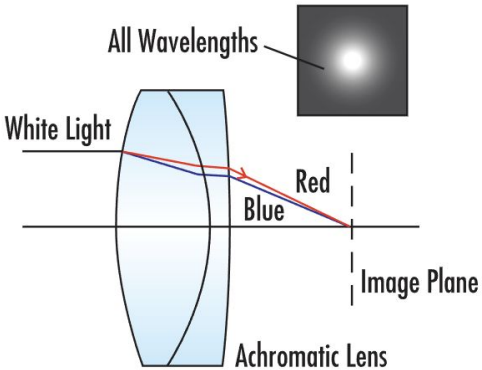




Center of image is clear,
corner of image is blurry.

Chromatic Aberration (Color shift)





(a) Full image

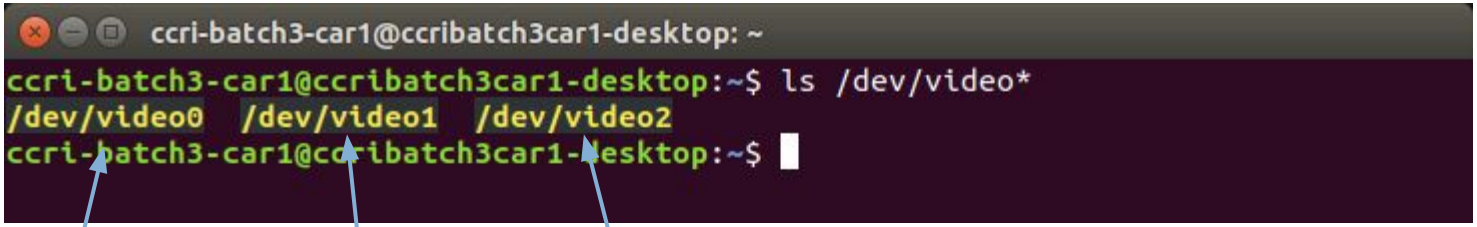
(b) Uncorrected

(c) Corrected

Accessing Camera on Linux

ls /dev/video*

- In Linux, every device is mounted as a file. So are cameras.
- After you connect your camera, you can use `ls /dev/video*` to see if the system has recognized your camera.
- For example, if the Realsense D435i is recognized by Jetson NX:



16-bit Depth Map

IR Image

RGB Image

v4l2src

Exclamation marks are signs to pass into a pipeline

- Video4Linux (v4l2) is an API to access camera on Linux.
- v4l2src can be used to capture video from v4l2 devices.
- You can give the pipeline to an OpenCV VideoCapture object.

```
import cv2
import time

cap = cv2.VideoCapture("v4l2src device=/dev/video2
extra-controls=\"c,exposure_auto=3\" ! video/x-raw, width=960,
height=540 ! videoconvert ! video/x-raw,format=BGR ! appsink")

time_old = time.time()
if cap.isOpened():
    cv2.namedWindow("demo", cv2.WINDOW_AUTOSIZE)
    while True:
        time_now = time.time()
        ret_val, img = cap.read()
        print(1/(time_now - time_old), 'Hz')
        time_old = time_now
        cv2.imshow('demo',img)
        cv2.waitKey(1)
else:
    print("camera open failed")

cv2.destroyAllWindows()
```

v4l2-ctl -d /dev/video2 --list-ctrls-menus

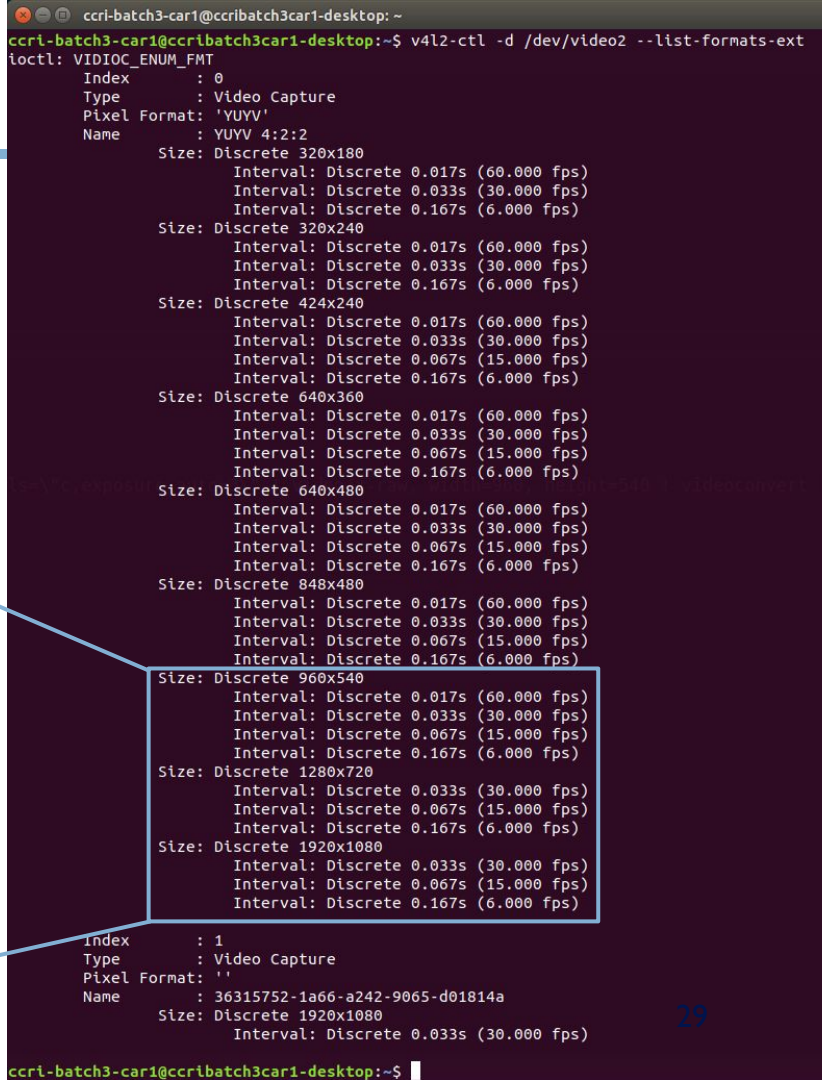
- `v4l2-ctl` can be used to control the camera on Linux.
- OpenCV also has an API that can control the camera. Most cameras will support either (if not both) v4l2 or OpenCV.

```
ccri-batch3-car1@ccribatch3car1-desktop:~$ v4l2-ctl -d /dev/video2 --list-ctrls-menus
                     brightness 0x00980900 (int)    : min=-64 max=64 step=1 default=0 value=0
                       contrast 0x00980901 (int)    : min=0 max=100 step=1 default=50 value=50
                     saturation 0x00980902 (int)    : min=0 max=100 step=1 default=64 value=64
                            hue 0x00980903 (int)    : min=-180 max=180 step=1 default=0 value=0
 white_balance_temperature_auto 0x0098090c (bool)   : default=1 value=1
                          gamma 0x00980910 (int)    : min=100 max=500 step=1 default=300 value=300
                           gain 0x00980913 (int)    : min=0 max=128 step=1 default=64 value=64
          power_line_frequency 0x00980918 (menu)   : min=0 max=2 default=3 value=3
                                0: Disabled
                                1: 50 Hz
                                2: 60 Hz
     white_balance_temperature 0x0098091a (int)    : min=2800 max=6500 step=10 default=4600 value=4600 flags=inactive
                     sharpness 0x0098091b (int)    : min=0 max=100 step=1 default=50 value=50
         backlight_compensation 0x0098091c (int)    : min=0 max=1 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=1 max=10000 step=1 default=166 value=166 flags=inactive
         exposure_auto_priority 0x009a0903 (bool)   : default=0 value=0
```

v4l2-ctl -d /dev/video2 --list-formats-ext

- `--list-formats-ext` can show all operating modes for the camera.
- Note that if we want the Realsense camera to operate at 60Hz, it can only support 960x540.

```
  Size: Discrete 960x540
          Interval: Discrete 0.017s (60.000 fps)
          Interval: Discrete 0.033s (30.000 fps)
          Interval: Discrete 0.067s (15.000 fps)
          Interval: Discrete 0.167s (6.000 fps)
  Size: Discrete 1280x720
          Interval: Discrete 0.033s (30.000 fps)
          Interval: Discrete 0.067s (15.000 fps)
          Interval: Discrete 0.167s (6.000 fps)
  Size: Discrete 1920x1080
          Interval: Discrete 0.033s (30.000 fps)
          Interval: Discrete 0.067s (15.000 fps)
          Interval: Discrete 0.167s (6.000 fps)
```



Camera Model & Calibration

Image - Camera - World

- Goal: We need to find transformations between:

Image pixel coordinates — Camera coordinates — World coordinates

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \qquad \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

- We will get:

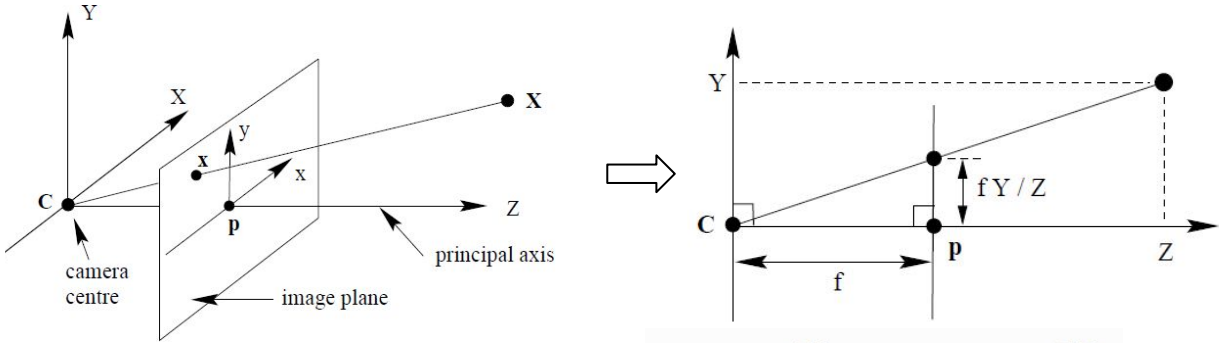
Camera Intrinsics

Camera Extrinsics

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = K \begin{pmatrix} {}^{c}R_w & {}^{c}T_w \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Scale factor

Camera Intrinsics



- From trigonometry we can find: $x = f\frac{X}{Z}, \qquad y = f\frac{Y}{Z}.$
- Write it into matrix form and add a image center shift term, because image center may not be on the principal axis:

Focal length (pixels)

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = K \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

Image Center Shift (pixels)

Camera Extrinsics

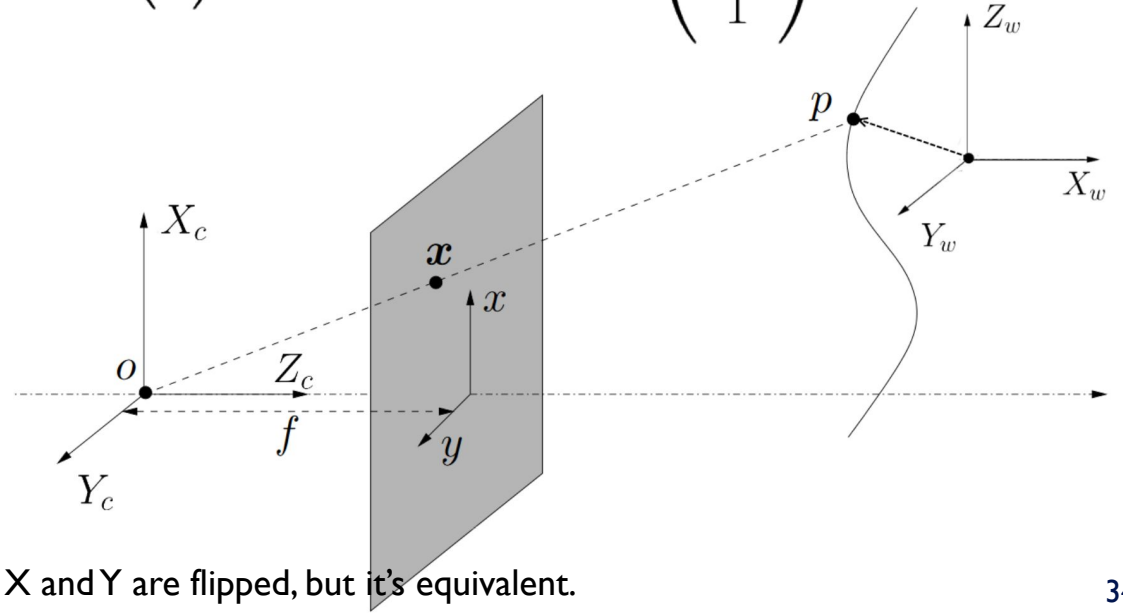
- Camera Extrinsics is a homogeneous transform from camera frame to world frame, both are 3D systems.
- It will be consist of a rotation and a translation.

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} {}^cR_w & {}^cT_w \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Intrinsics & Extrinsics

- But how do we calculate them for an actual camera?

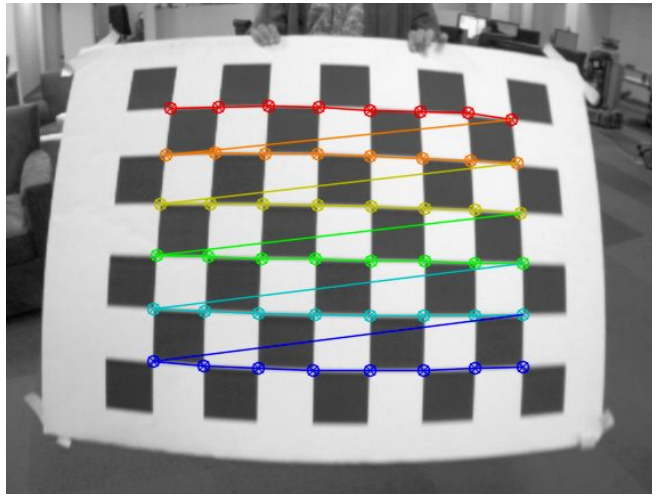
$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$



Here X and Y are flipped, but it's equivalent.

Get Intrinsics - Camera Calibration

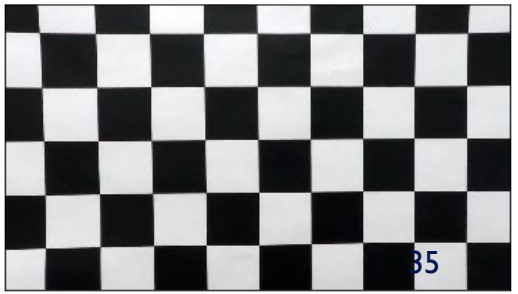
- We can use OpenCV functions `findChessboardCorners` and `calibrateCamera` to do the calibration.
- It will not only calculate the camera matrix K, but also a distortion matrix to correct the image of any distortion.
- Tutorial, OpenCV documentation.



$$\begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Camera Matrix

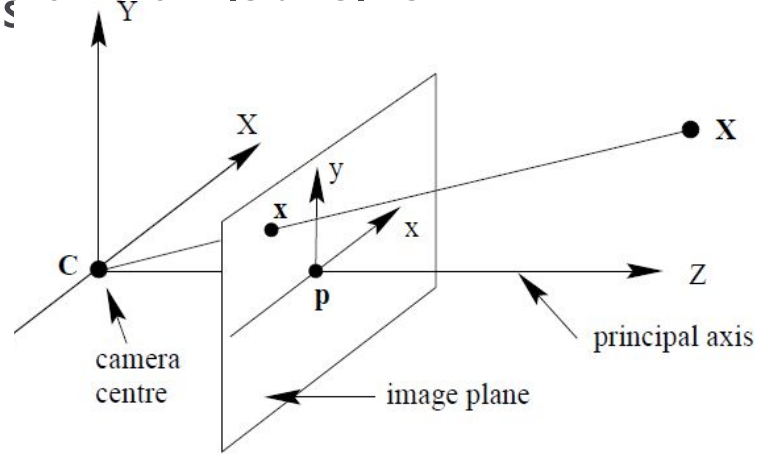
Original image



Distortion corrected image

Get Extrinsics - Computing a Transformation

- The extrinsics can be calculated with:
  - Find enough corresponding points in camera and world frames;
  - Solve the linear equation for the homography;
  - Separate out rotation and translation us[illegible] decomposition.
- However, we don't race in the world frame. We race in the car's frame.
- We can directly work with the camera frame.



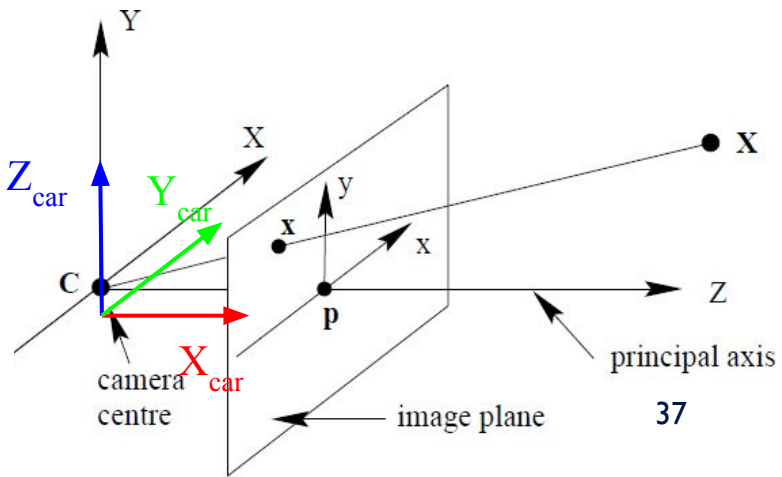
Camera - Image Frame

Distance Estimation

- We can get the camera matrix from calibration.
- If we mount the camera on the car without any roll-pitch-yaw, then axes in the camera and car frames are overlapping.
- Then if we look at the third line, we can see $\lambda = Z_{cam}$

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{pmatrix} = \begin{pmatrix} Y_{car} \\ Z_{car} + H_{mount} \\ X_{car} \end{pmatrix}$$



Distance Estimation

- Hence

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{pmatrix} = \begin{pmatrix} Y_{car} \\ Z_{car} + H_{mount} \\ X_{car} \end{pmatrix}$$

$$\lambda = Z_{cam}$$

$$\Rightarrow$$

$$x = f_x \frac{Y_{car}}{X_{car}} + x_0$$

$$y = f_y \frac{Z_{car} + H_{mount}}{X_{car}} + y_0$$

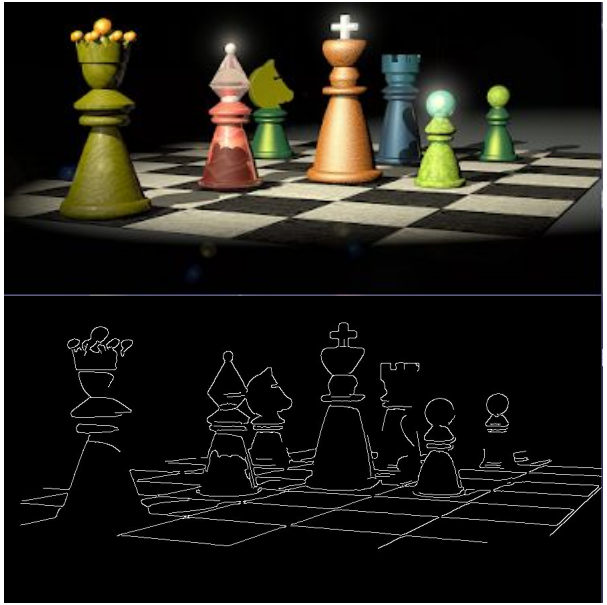
- If the object is on the ground, i.e. $Z_{car} = 0$ then we only need one pair of corresponding points to find H_{mount} .
- What if there are rotations in camera mounting?

We can add more corresponding points to solve for any missing variable.

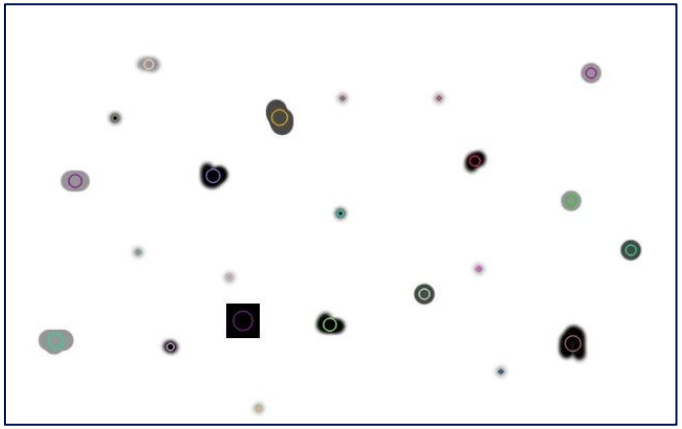
Detection with OpenCV

Useful for doing the vision lab.

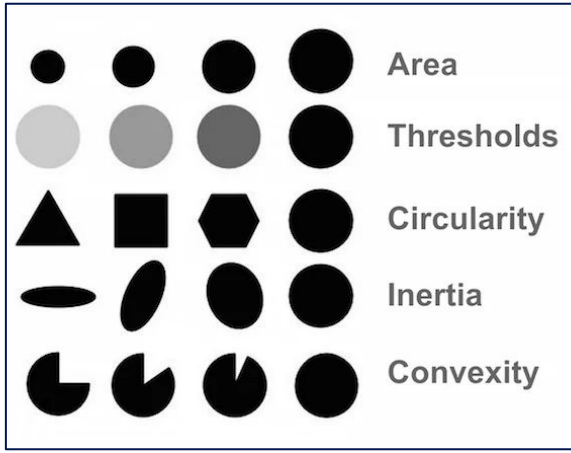
Edge & Blob Detection



`cv2.Canny`



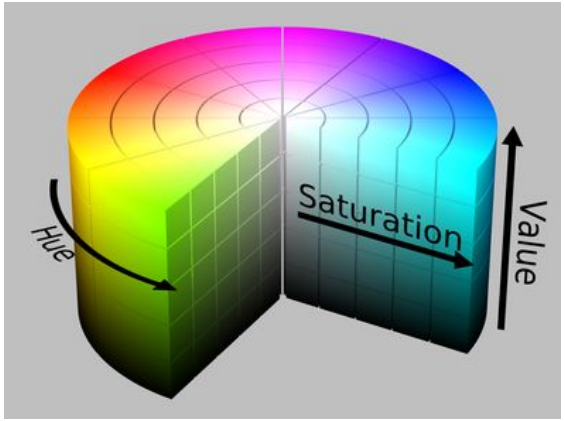
```
detector = cv2.SimpleBlobDetector()
keypoints = detector.detect(img)
```



Parameters for the `SimpleBlobDetector`

HSV color space

- Convert (`red, green, blue`)
- To (`hue, saturation, value`)
- value is also called intensity or brightness.
- HSV is sometimes more useful than RGB when doing color-based filtering.
- We can use to `cv2.cvtColor` convert color space.

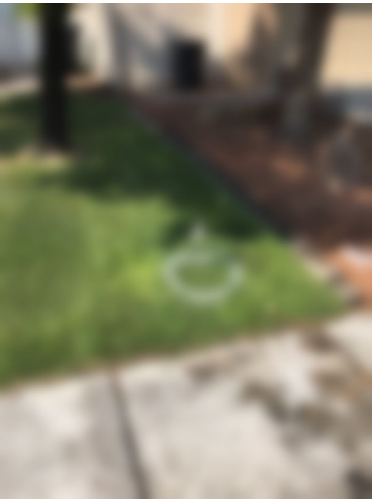


HSV color space

Hue value

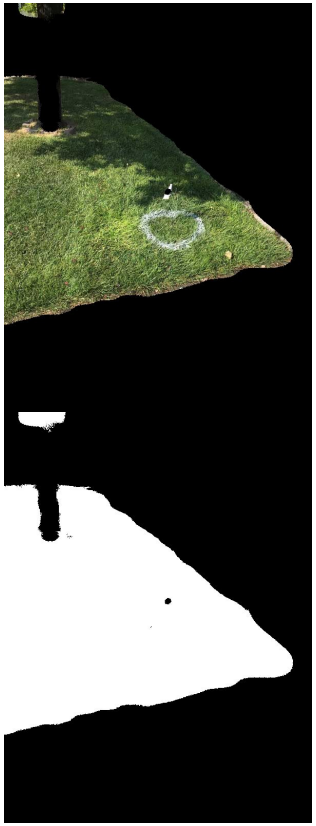
Example: lawn detection

- Suppose you are an engineer in an autonomous lawn mower startup.
- The investor is coming to see the prototype tomorrow. There is no time to train a neural network.
- You are given the task to detect the boundary of a lawn, with classic methods.
- It seems difficult to do it in RGB space. We can try HSV color space.



Used function:

`cv2.GaussianBlur`



Used function:

`cv2.cvtColor`
`cv2.threshold`

Used function:

`cv2.connectedCompon`
`entsWithStats`

Used function:

`cv2.dilate`
`cv2.erode`

Feature Extraction

- To detect an object in an image, we always need to find features.
- What is a feature?
- AB: flat surfaces in some channels.
- CD: certain edges
- EF: certain corners or curves.
- We need special filters to respond to them.

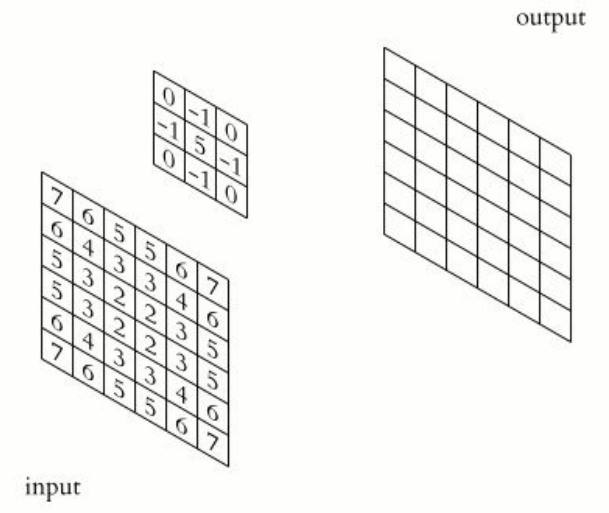
2D Convolution

2D convolution:

$$h = f \otimes g$$

f - the values in a 2D grid that we want to convolve

g - convolutional weights of size MxN



- The weights g is often called a kernel or a filter.
- Kernels with different weights will respond differently to features.

$$g_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad g_2 = \begin{bmatrix} 0.107 & 0.113 & 0.107 \\ 0.113 & 0.119 & 0.113 \\ 0.107 & 0.113 & 0.107 \end{bmatrix} \quad g_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$f =$

$f \otimes g_1$

Unchanged Image

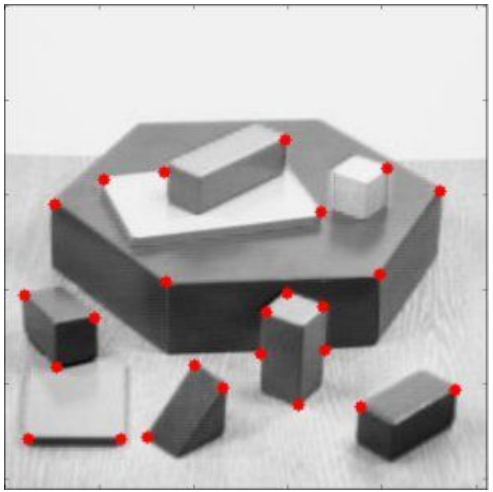
$f \otimes g_2$

Blurred Image

$f \otimes g_3$

Vertical Edges

Some Classical Feature Extraction Methods

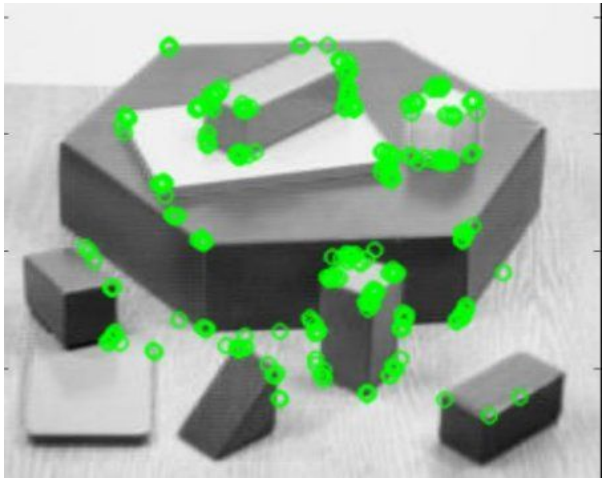


Shi-Tomasi Corner Detector:

```
cv2.goodFeaturesToTrack()
```

SIFT (Scale-Invariant Feature Transform):

```
sift = cv2.SIFT_create()
kp = sift.detect()
```



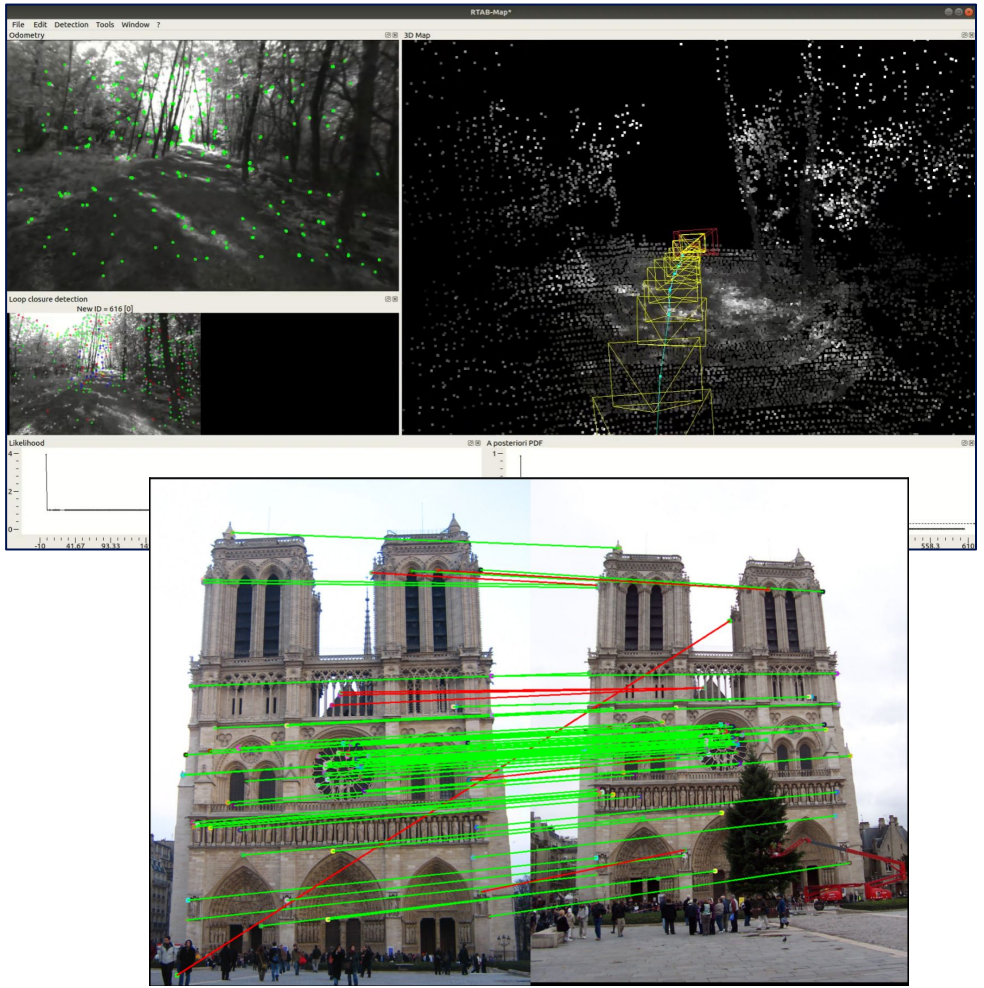
ORB (Oriented FAST and Rotated BRIEF):

```
orb = cv2.ORB_create()
kp = orb.detect()
kp, des = orb.compute()
```

By matching these features between frames, we can track object, classically.

Put it together

- We can find unique features in images and match them between frames.
- And we can find their coordinates in the world frame with camera model.
- Can we use them as landmarks in SLAM?



https://introlab.github.io/rtabmap/

Visual SLAM

Visual SLAM Framework

- Feature Matching: Extract and match visual feature between camera frames.
- Visual Odometry: Calculate the motion between consecutive camera frames.
- Bundle Adjustment: Correct the transformations with historical data.
- Map Building: Build the map based on the measurements and corrections.
- Loop Closure: Detect path returns to a previous point and correct the accumulated drift error.

ORB-SLAM2

Bundle Adjustment

- Optimizing the camera projections by considering all points and cameras.
- Starting from an initial guess.
- Treating camera from different time as different cameras.
- It is a non-linear optimization, including rotation, distortion correction etc.
- Statistically optimal but expensive.

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

⇩ Consider many points

$$\boldsymbol{x}_i = \lambda^{w} \boldsymbol{P} \boldsymbol{X}_i, \text{ for every point } i.$$

⇩ Consider many points and cameras

$$\boldsymbol{x}_{ij} = \lambda_j{}^{w} \boldsymbol{P}_j \boldsymbol{X}_{ij}, \text{ for every point } i \text{ and camera } j.$$

⇩ When the projection is arbitrary

$$\boldsymbol{e}_{ij} = \lambda_j{}^{a} \boldsymbol{P}_j \boldsymbol{X}_{ij} - \boldsymbol{x}_{ij}, \text{ for every point } i \text{ and camera } j.$$

$$\underset{{}^{a}\boldsymbol{P}_j}{\operatorname{argmin}} \sum_{i}^{\text{points}} \sum_{j}^{\text{cameras}} \mathbb{1}_{ij} ||\lambda_j{}^{a} \boldsymbol{P}_j \boldsymbol{X}_{ij} - \boldsymbol{x}_{ij}||^2$$

$\mathbb{1}_{ij}$ is 0 when point i is not seen by camera j

Challenges in Visual SLAM

- Exponentially increasing number of feature points to match.
- Bundle adjustment is especially expensive.
- Can it work on monocular camera without IMU?
- Can the algorithm handle drifts?
- Can it close the loop?

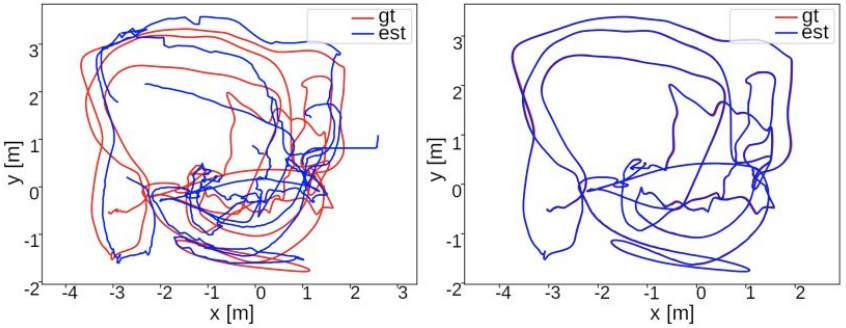


Fig. 4: Pure stereo mode (left) and IMU sensor fusion effect on stereo SLAM (right) for ORB-SLAM3 on EuRoC V203

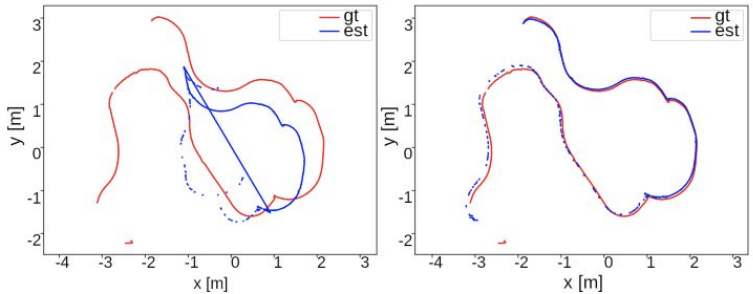
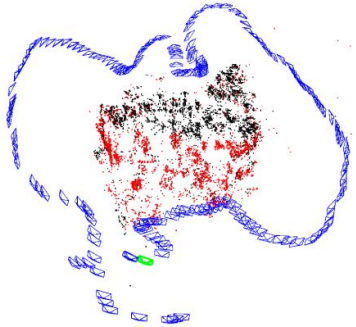


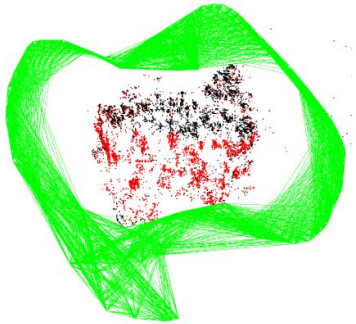
Fig. 5: ORB-SLAM3 (left) and OpenVSLAM (right) on TUM RGB-D pioneer_slam3 sequence. Each image on the

ORB-SLAM

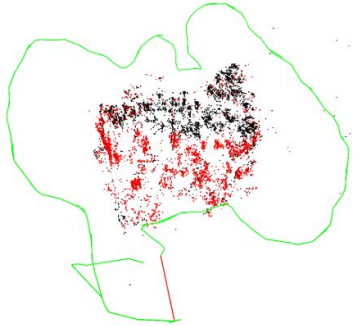
- 3 parallel threads: Tracking, Local Mapping and Loop Closing.
- Selection of keyframes and Culling policies of keyframes and points.
- Separating of local and global information with Covisibility Graph and Essential Graph.
- Use Bags of Words for place recognition in loop closure.



(a) KeyFrames (blue), Current Camera (green), MapPoints (black, red), Current Local MapPoints (red)



(b) Covisibility Graph



(c) Spanning Tree (green) and Loop Closure (red)

(d) Essential Graph

References

- https://medium.com/@dcasadoherraez/introduction-to-visual-slam-chapter-1-introduction-to-slam-a0211654bf0e
- https://arxiv.org/pdf/2107.07589.pdf