Robot Perception

EEL 4930/5934: Autonomous Robots

Spring 2023

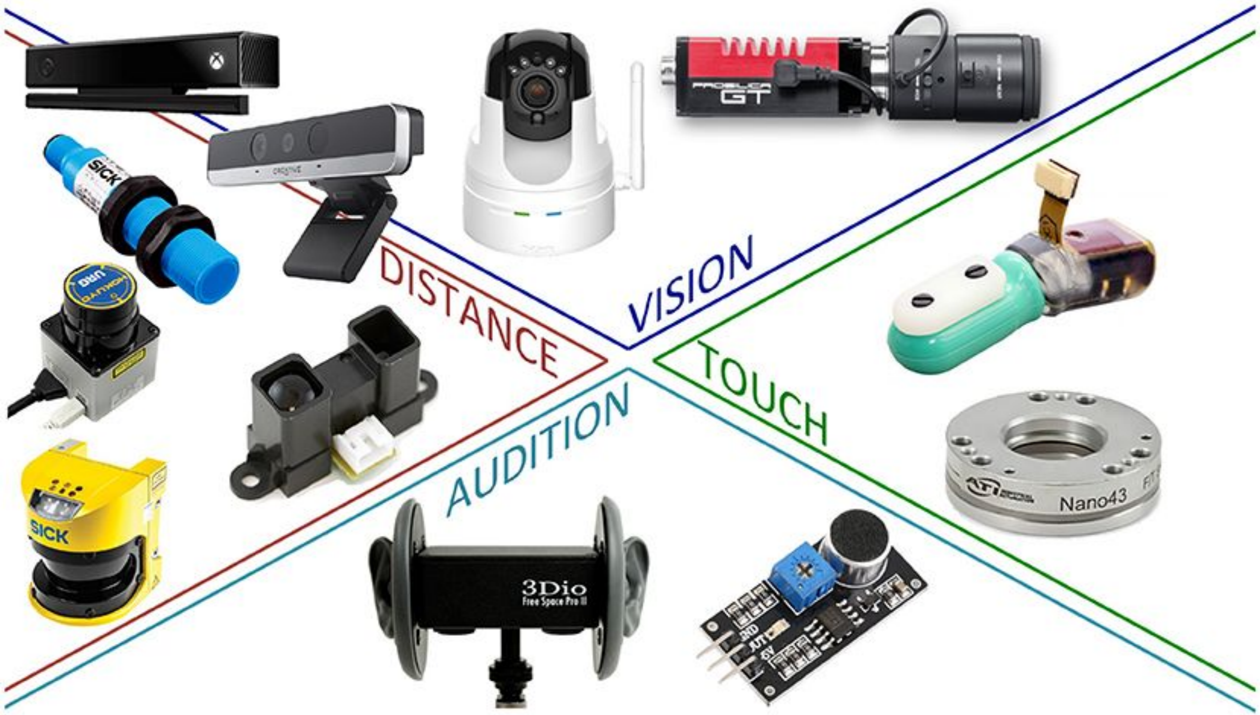
Md Jahidul Islam

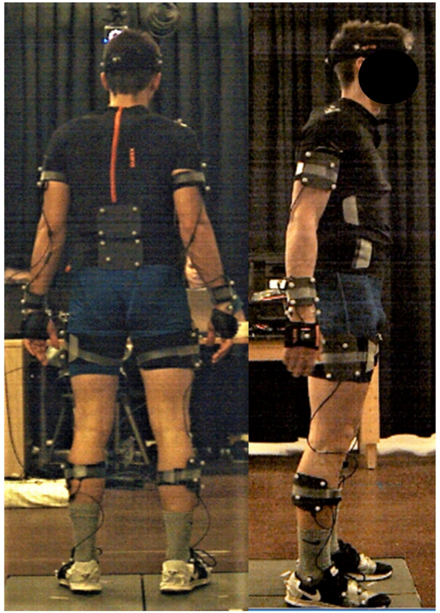
Lecture 6

ECE | Florida
Electrical & Computer Engineering

UF | UNIVERSITY of FLORIDA

Sensors




[redacted]

© Mathew et.al.

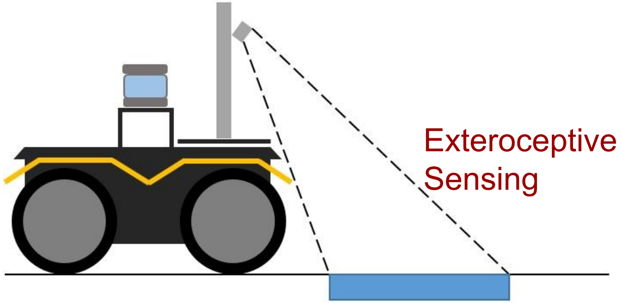
Exteroceptive and Proprioceptive Sensors

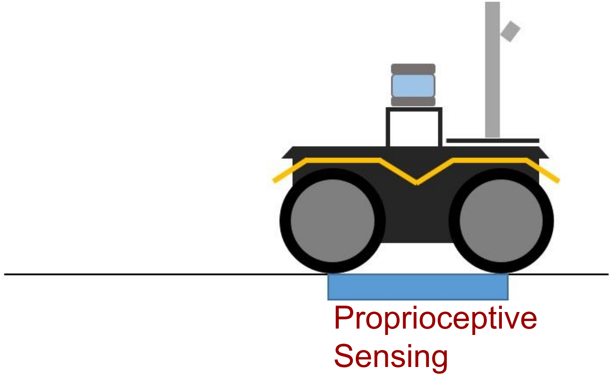
⇒ **Exteroceptive sensing**

- External information from the environment
- Example:
  - ○ Tactile sensors
  - ○ Vision sensors: cameras
  - ○ Proximity sensors: LiDAR, radar, ultrasonic sensors, stereo cameras

⇒ **Proprioceptive sensing**

- Internal information about the robot: state, motion, joint angles, etc.
- Example:
  - ○ Position and velocity: encoders
  - ○ Location: GPS
  - ○ Attitude: Inertial measurement units (IMU), accelerometers, force sensors



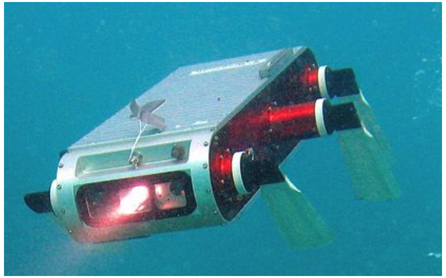


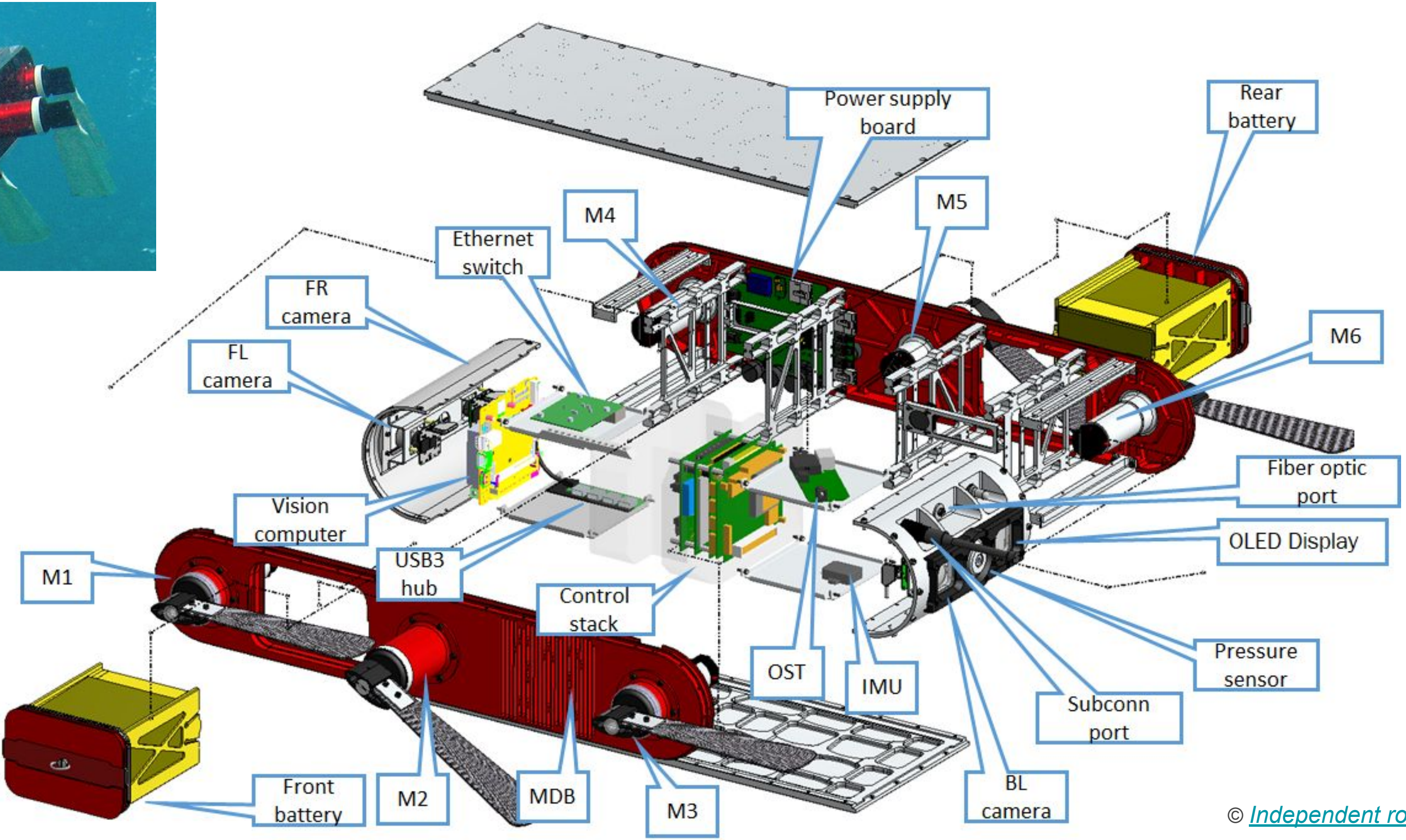
AUV Perception: Robot Convoying

https://youtu.be/Em7V-vBApHc

© *Shkurti et.al.*

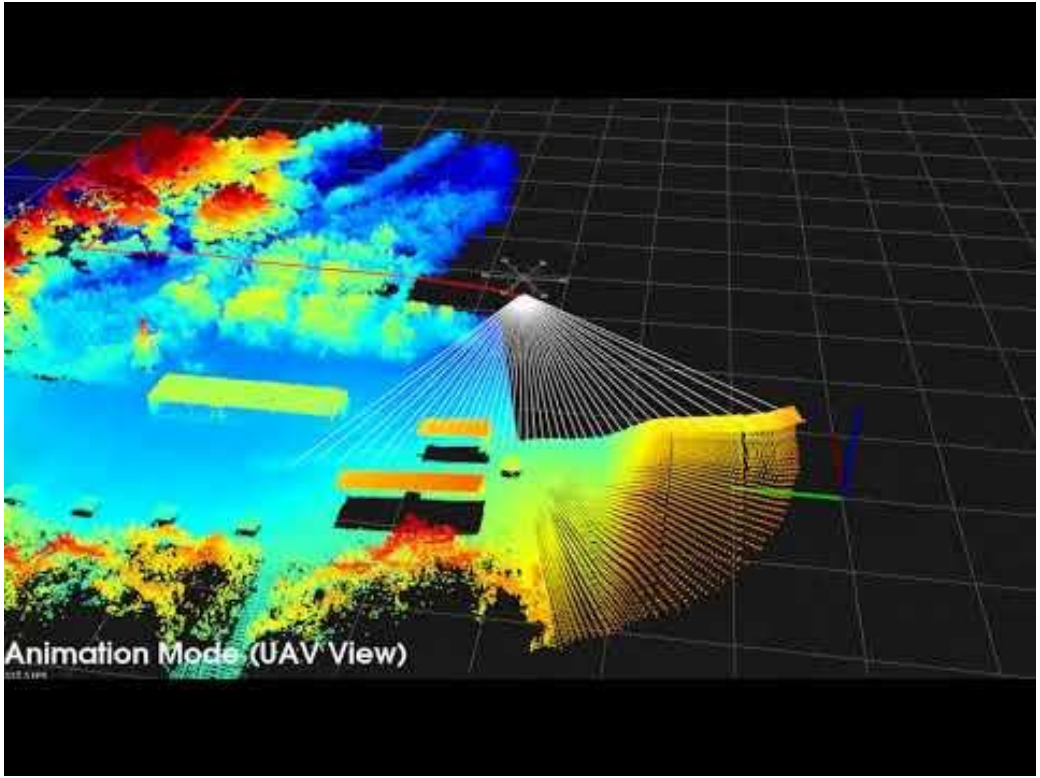
Aqua AUV Components





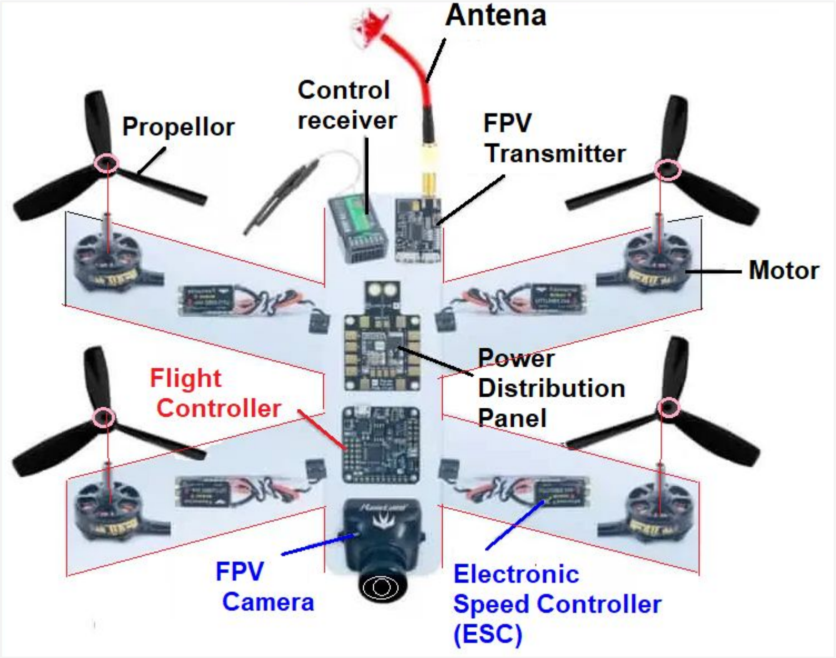
© *Independent robotics*

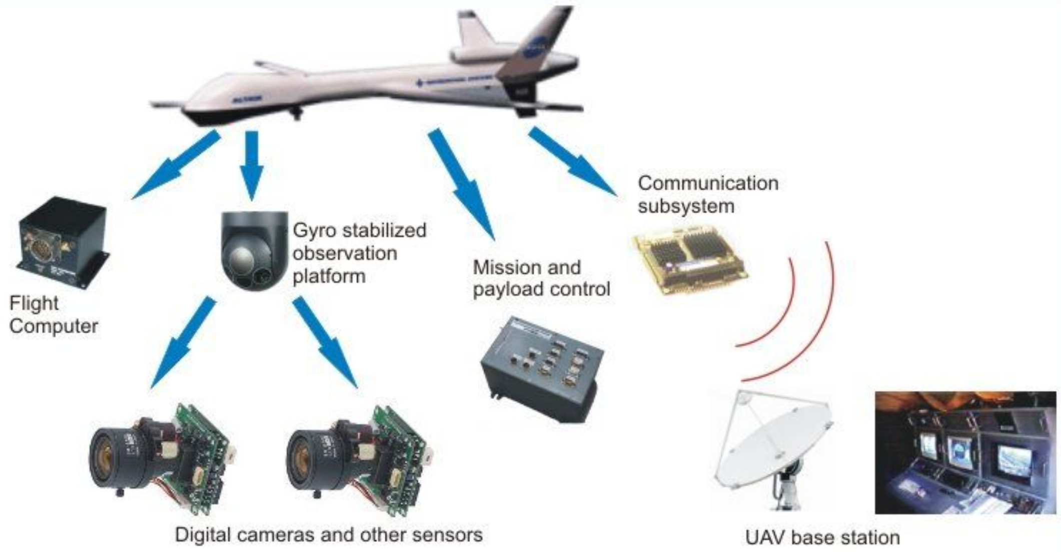
UAV Perception: 3D Mapping



https://youtu.be/cbczfgH1x0s

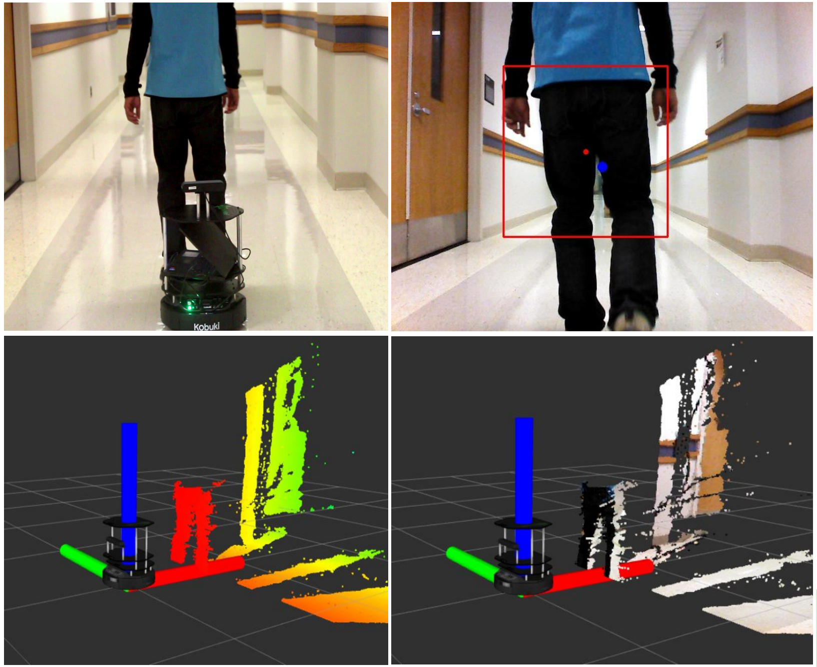
UAV Components



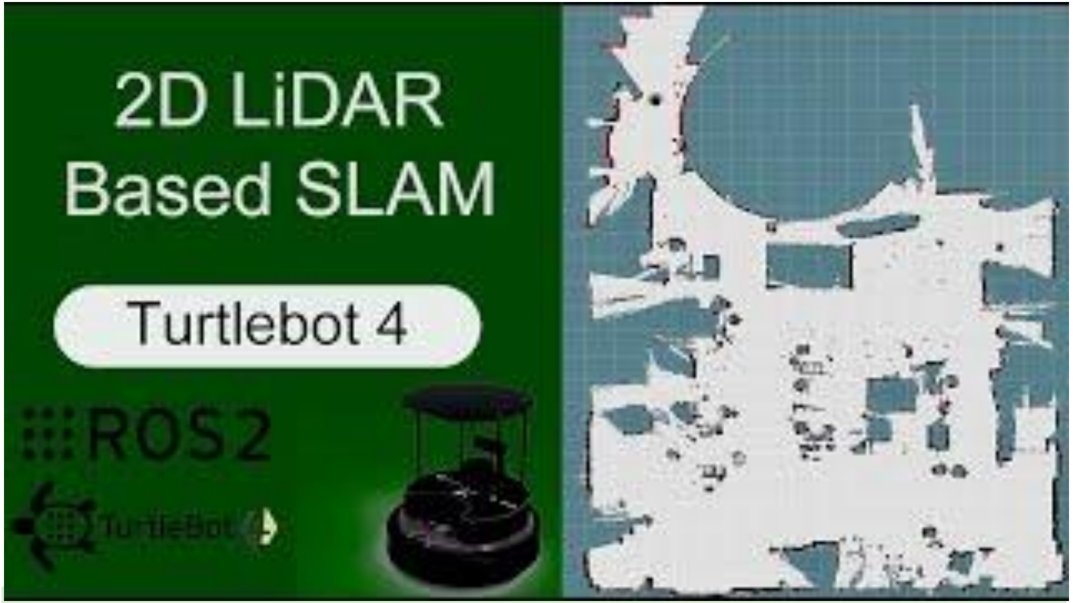

Communication subsystem
Gyro stabilized observation platform
Mission and payload control
Flight Computer
Digital cameras and other sensors
UAV base station

© Pastor et.al.

UGV Perception



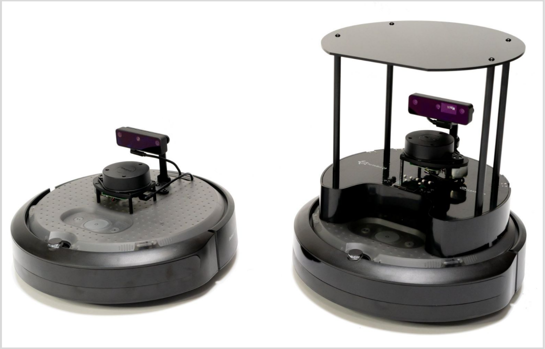
Person following

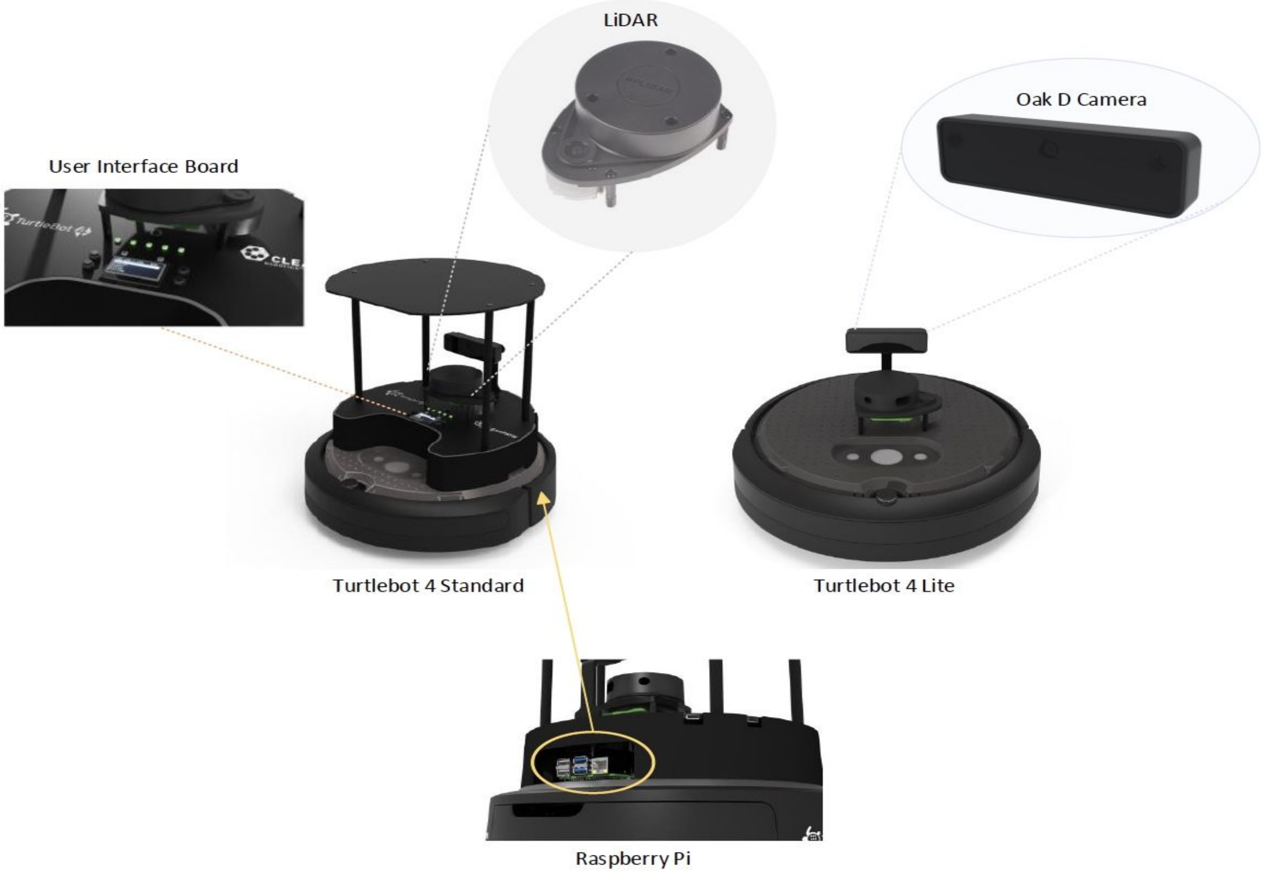


https://youtu.be/vwuKs8F0axE

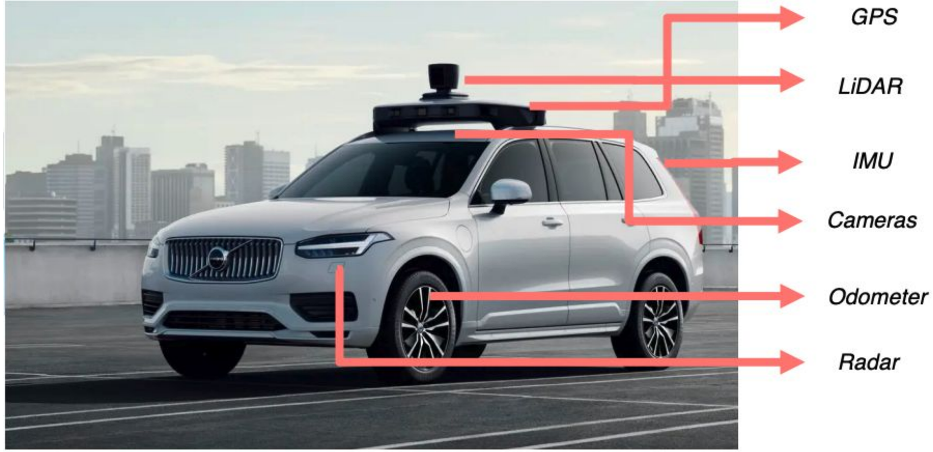
2D Mapping (SLAM)

TurtleBot-4 Components





SDC: Self Driving Cars!



GPS

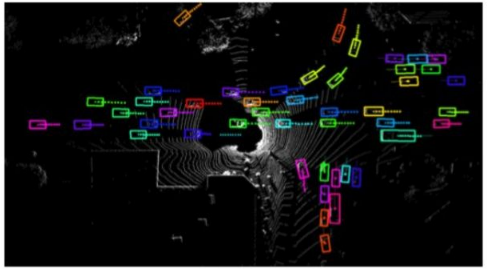
LiDAR

IMU

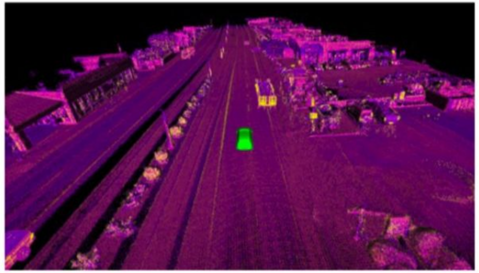
Cameras

Odometer

Radar



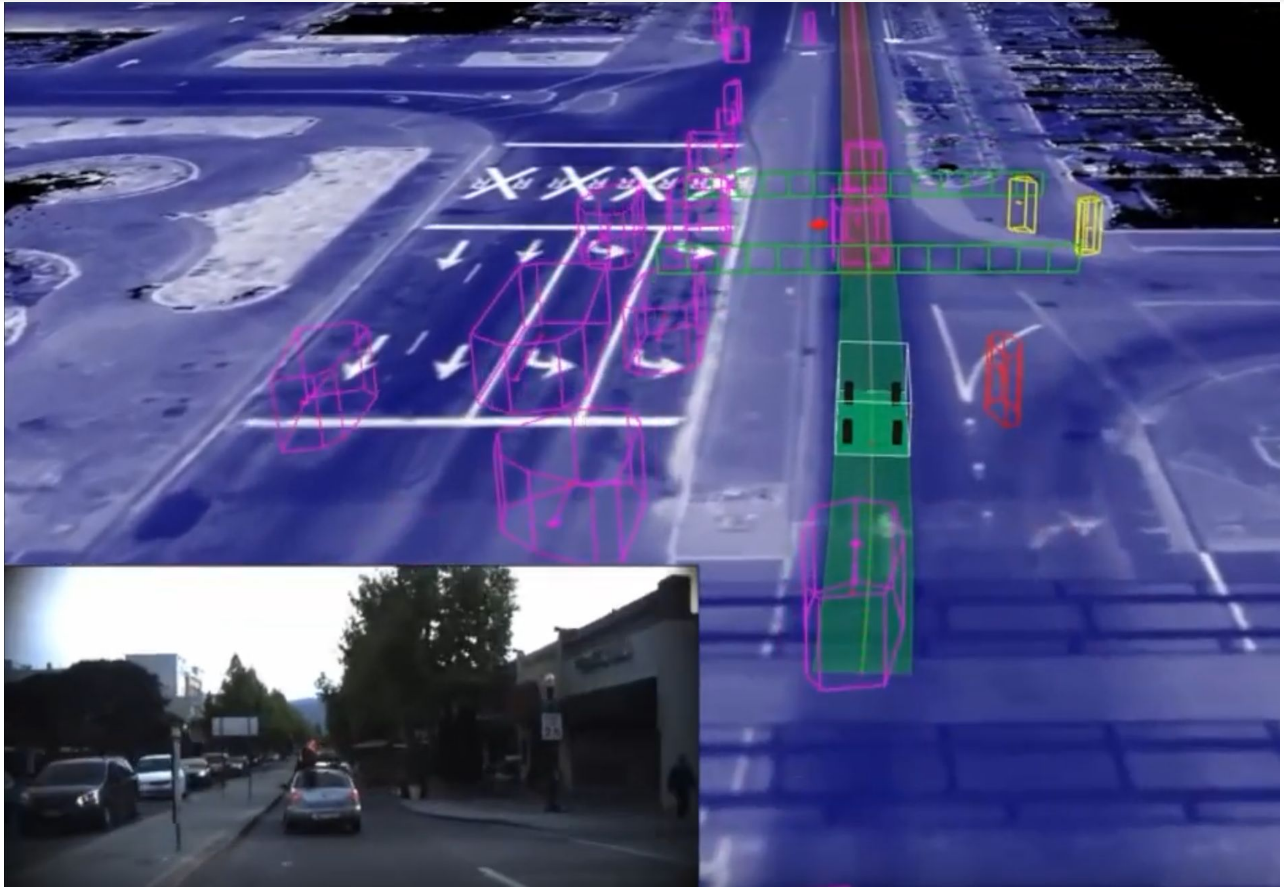
Perceive other objects



Localize itself

Perceive surrounding scene

SDC: Visual Perception



SDC: Learning Visual Perception

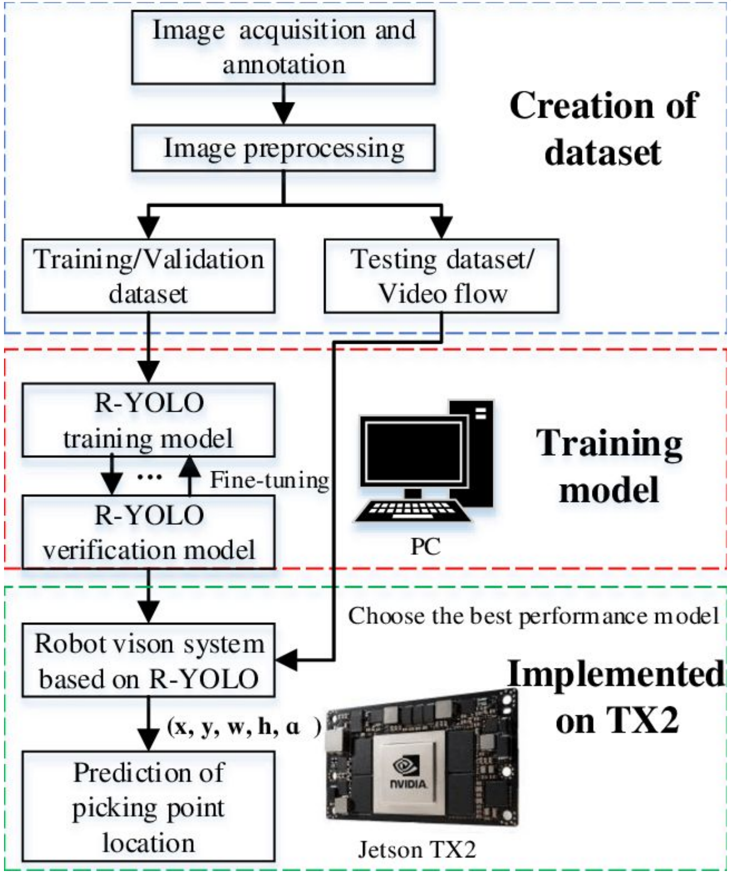
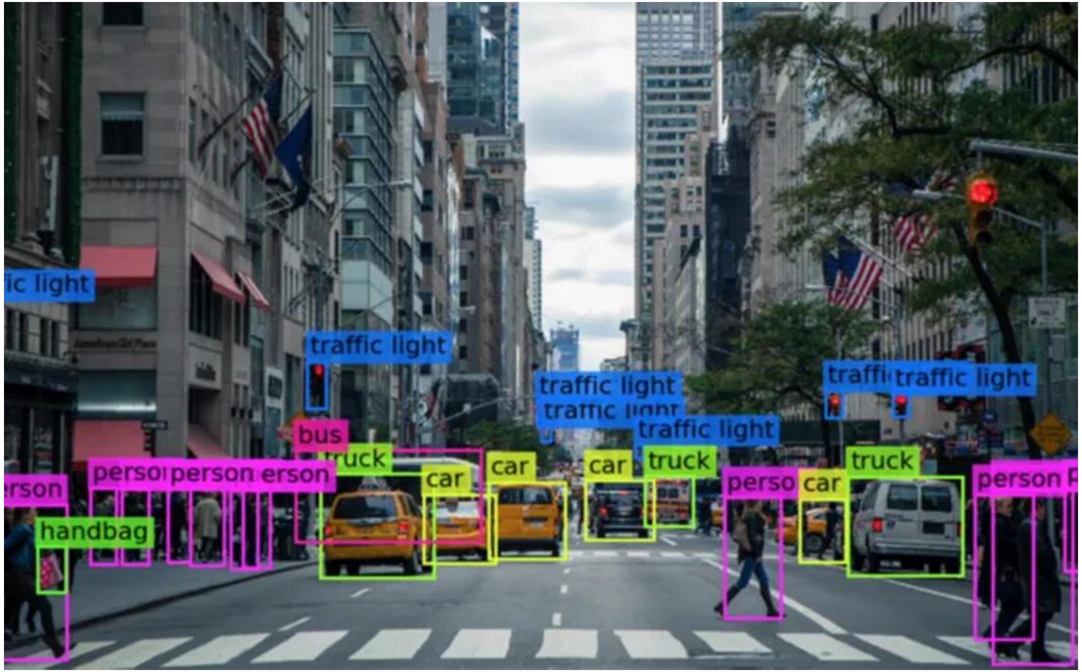
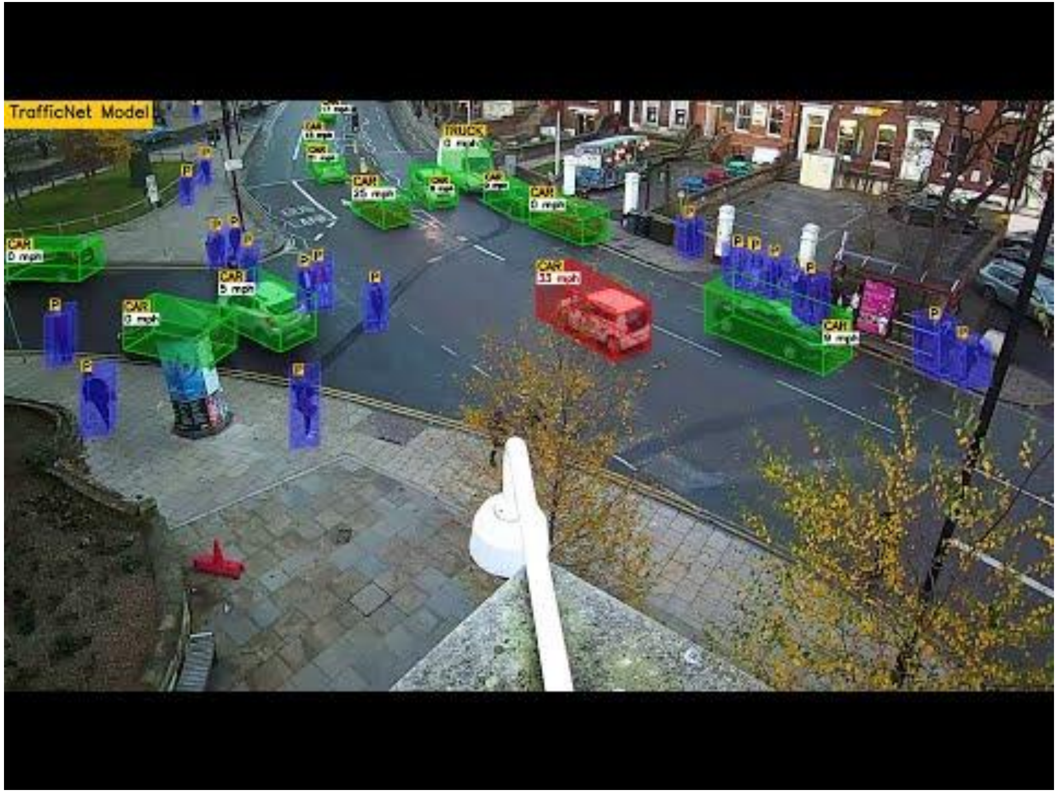


Image acquisition and annotation
Image preprocessing
Creation of dataset
Training/Validation dataset
Testing dataset/ Video flow
R-YOLO training model
...
Fine-tuning
R-YOLO verification model
PC
Training model
Choose the best performance model
Robot vison system based on R-YOLO
(x, y, w, h, α)
Implemented on TX2
Prediction of picking point location
Jetson TX2

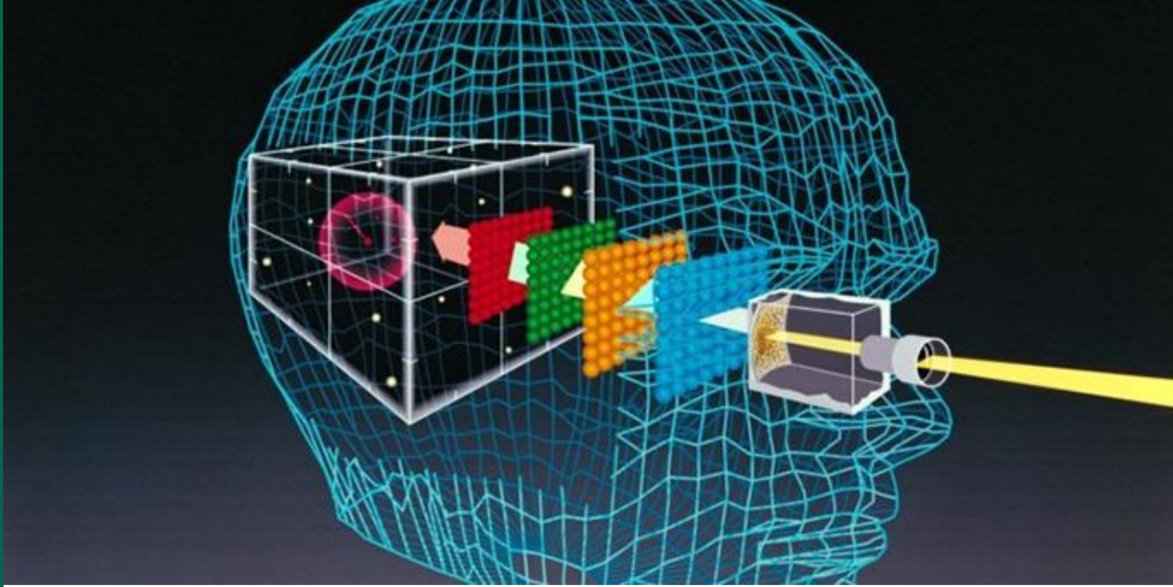


Machine Perception: Traffic Monitoring



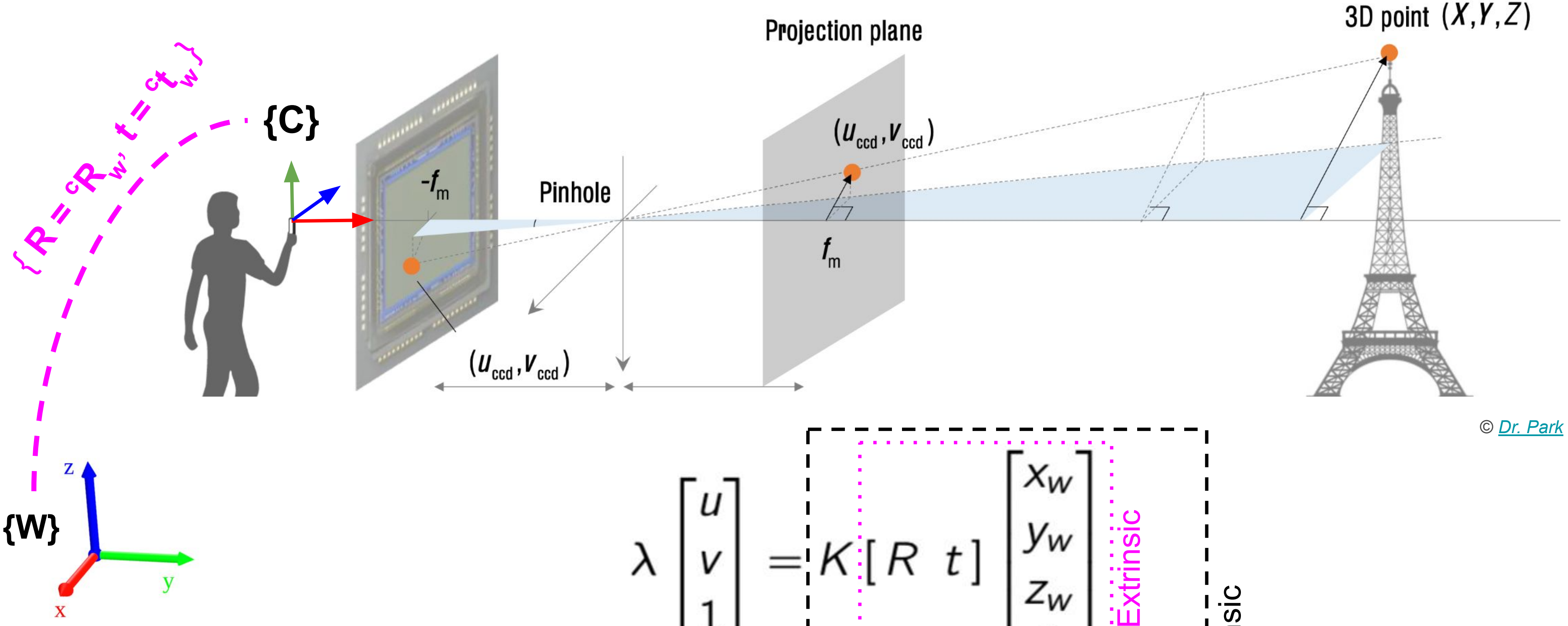
https://youtu.be/FdiQ_EGbZe0

Visual Perception In Robotics: A Hands-on Introduction



- **Camera model:**
  - Intrinsic and extrinsic parameters
  - Projection matrix
- Perspective transformation
  - Homography estimation
  - Camera calibration
- Stereo geometry
  - Stereo camera configuration
  - Disparity and depth-map

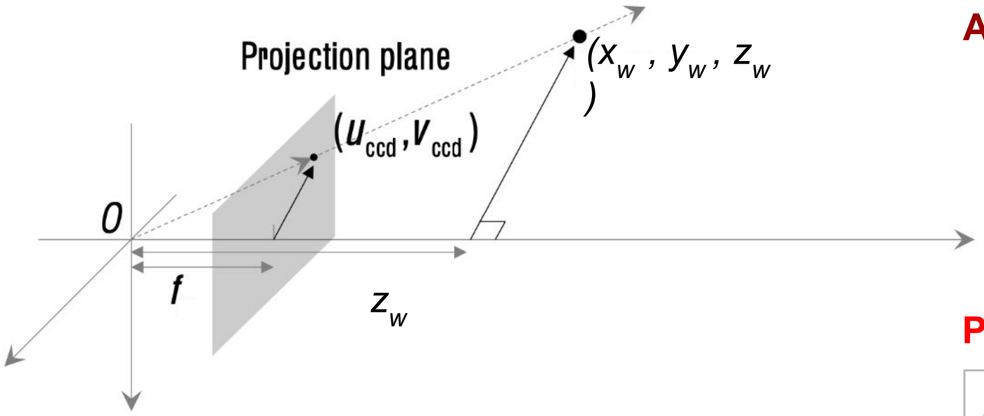
Pinhole Camera Model

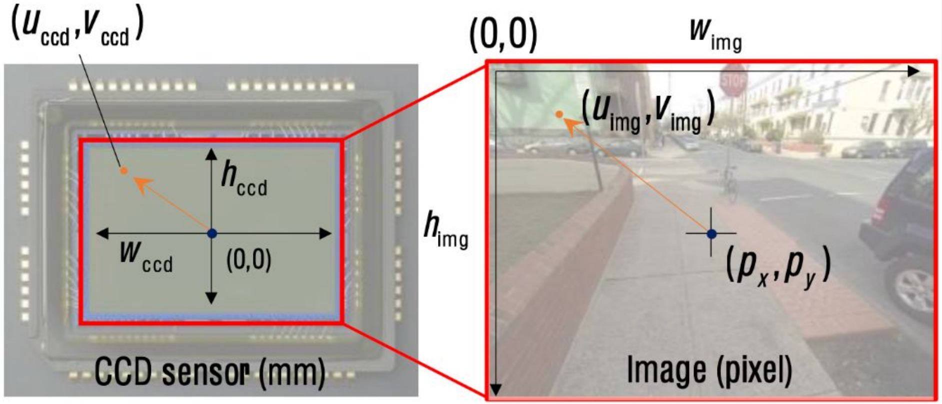


© Dr. Park

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{K \underbrace{[R \;\; t] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}}_{\text{Extrinsic}}}_{\text{Intrinsic}}$$

CCD ↔ Image Pixels





Assume origin at camera center {C} = {W}

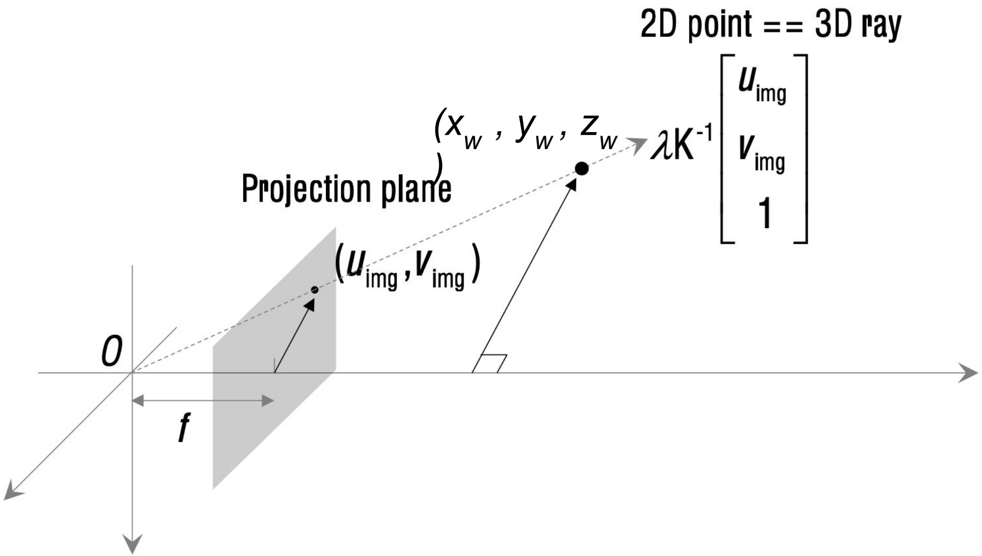
$$u_{ccd} = f\,\frac{x_w}{z_w} \qquad v_{ccd} = f\,\frac{y_w}{z_w}$$

Pixel projection

$$u_{img} = u_{ccd}\,\frac{w_{img}}{w_{ccd}} + p_x = u_{ccd}\,m_x + p_x = m_x\,f\,\frac{x_w}{z_w} + p_x$$

$$v_{img} = v_{ccd}\,\frac{h_{img}}{h_{ccd}} + p_y = v_{ccd}\,m_y + p_y = m_y\,f\,\frac{y_w}{z_w} + p_y$$

K: Intrinsic Matrix



Intrinsic Parameters

- m_x, m_y
- f
- p_x, p_y
- *s (skew) is often considered*
  - *In K[0, 1] position*
  - *Analog cameras*

Assume origin at camera center {C} = {W}

$$\lambda \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = K \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \underbrace{\begin{bmatrix} m_x f & 0 & p_x \\ 0 & m_y f & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{K}} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}$$

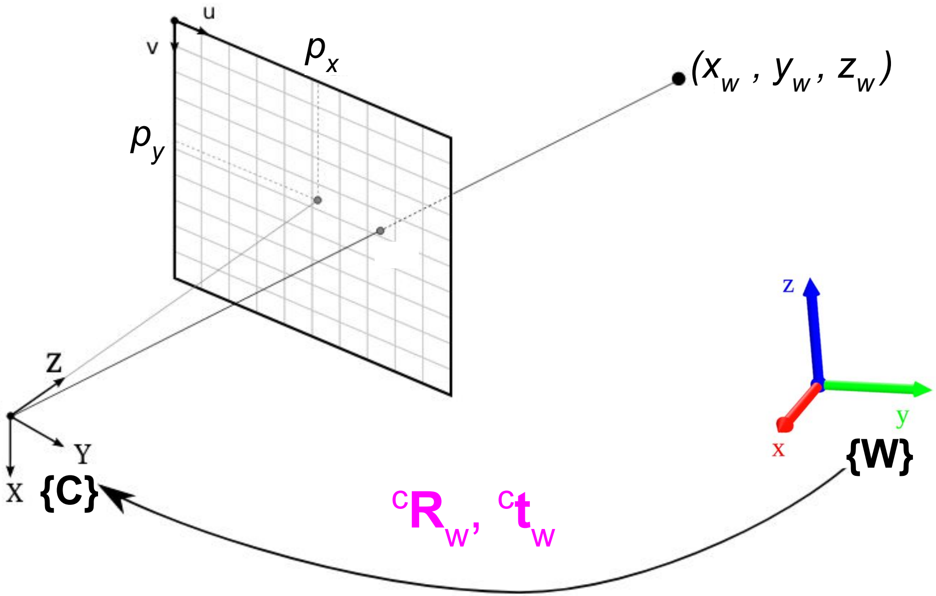
P: Projection Matrix

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K\,[\,R \;\; t\,] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = P \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_0 \\ r_{21} & r_{22} & r_{23} & t_1 \\ r_{31} & r_{32} & r_{33} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$\boldsymbol{P = K\,[R \;\; t]}$$



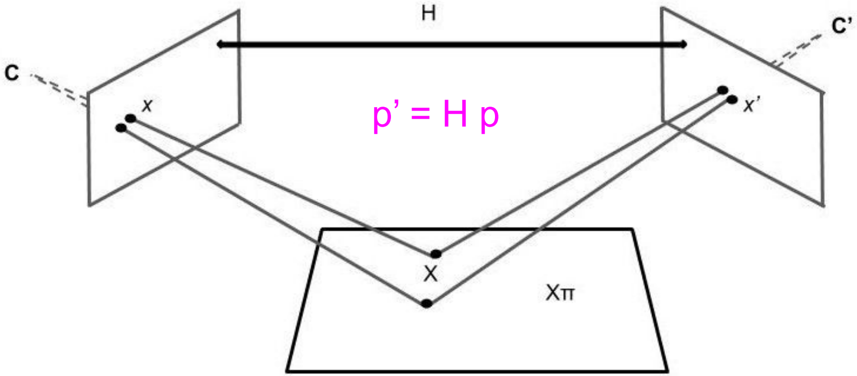
Perspective Transformation: Homography

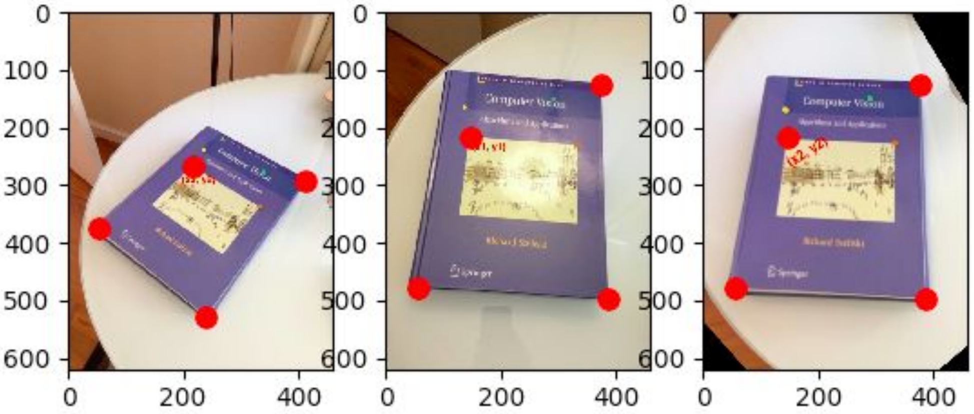
Homography:

Transformation between two planes (up to a scale factor)

Transformation cases:

- Pure camera rotation: fixed camera center
- Same planar surface viewed by two cameras



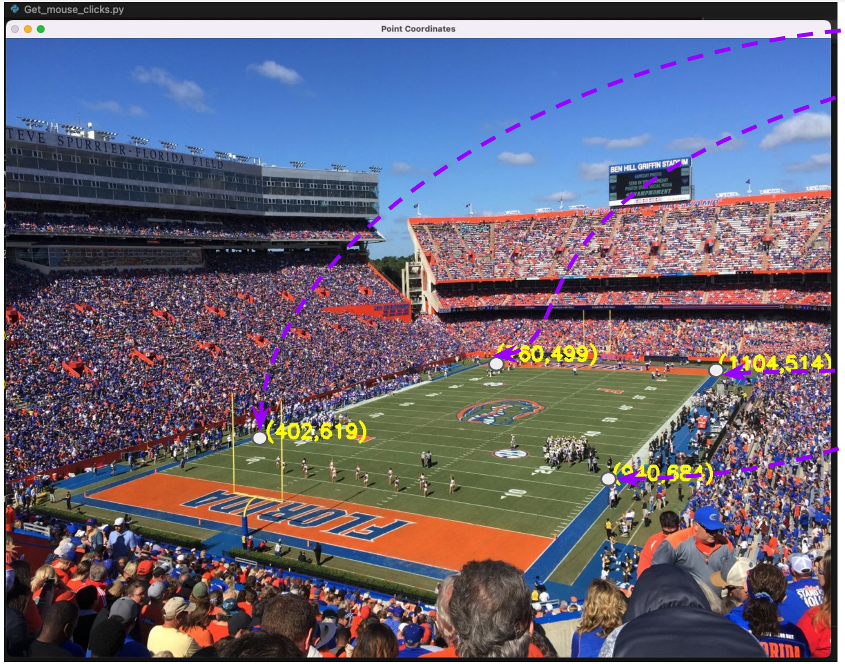


$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x' = \frac{h_{11}\,x + h_{12}\,y + h_{13}}{h_{31}\,x + h_{32}\,y + h_{33}}$$

$$y' = \frac{h_{21}\,x + h_{22}\,y + h_{23}}{h_{31}\,x + h_{32}\,y + h_{33}}$$

H: Homography Estimation



$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xx' & -yx' & -x' \\ 0 & 0 & 0 & x & y & 1 & -xy' & -yy' & -y' \end{bmatrix} \begin{bmatrix} h11 \\ h12 \\ h13 \\ h21 \\ h22 \\ h23 \\ h31 \\ h32 \\ h33 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Need at-least **4 points** = **8 equations**

h_{33}* = *1 (up to scale)

When n >> 4 points are available

- Solve using SVD
- Use RANSAC algorithm
- Why SVD and RANSAC?

$$\underbrace{\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}}_{x',\, y'} = \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{x,\, y}$$

Solving for H: Use SVD

$$
\begin{array}{lcccc}
 & 2N \times 8 & 8 \times 1 & & 2N \times 1 \\
\begin{array}{l} \textbf{Point 1} \\ \\ \textbf{Point 2} \\ \\ \textbf{Point 3} \\ \\ \textbf{Point 4} \\ \\ \end{array} &
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\
0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\
x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\
0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\
x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\
0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4
\end{bmatrix} &
\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} & = &
\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} \\
\textbf{additional points} & \vdots & & & \vdots \\
 & \boldsymbol{A} & \boldsymbol{h} & & \boldsymbol{= b}
\end{array}
$$

A is rank deficient, the least square solution (use normal equation)

$$h = (A^TA)^{-1}(A^Tb)$$

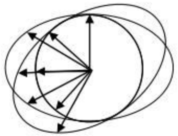
We can also use SVD to solve this ***Ah = 0***

- **A** is **2Nx9** and **h** is **9x1** (see next slide)

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^{T}_{n\times n} \qquad m \geq n$$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \qquad \begin{array}{c} \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0 \\ U^TU = I \\ V^TV = I \end{array}$$

$$A = U_1\sigma_1V_1^T + U_2\sigma_2V_2^T + \cdots + U_n\sigma_nV_n^T$$

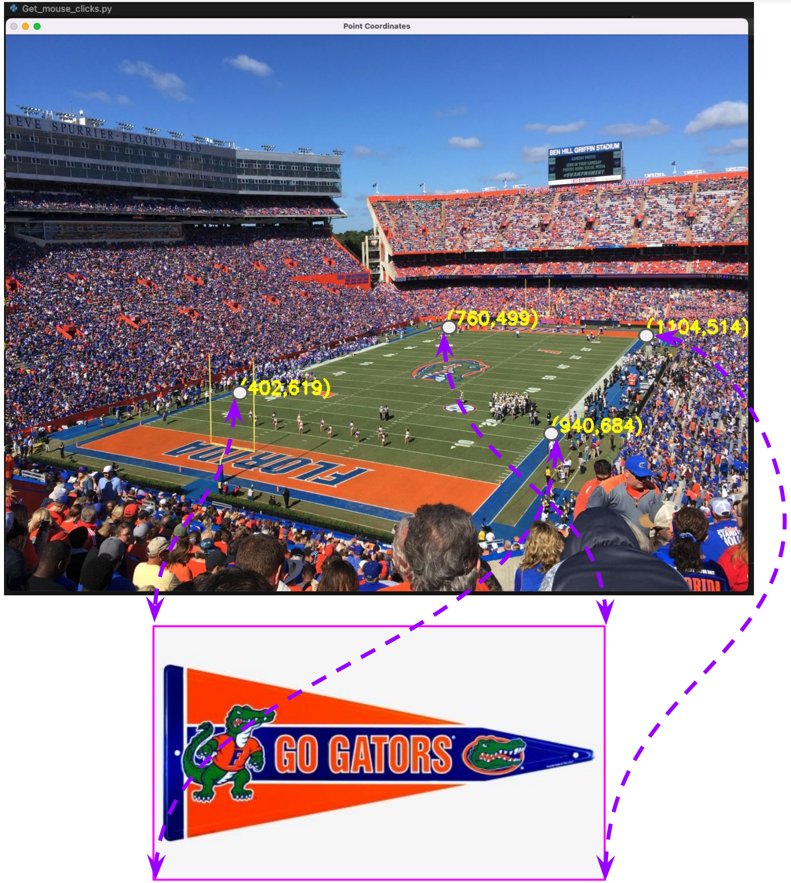


$$U\Sigma V^T\mathbf{x}$$

SVD Pointers

- UCSD lecture UConn lecture
- U-Illinois lecture ROS-OpenCV

Homography Transformation: HH3-A

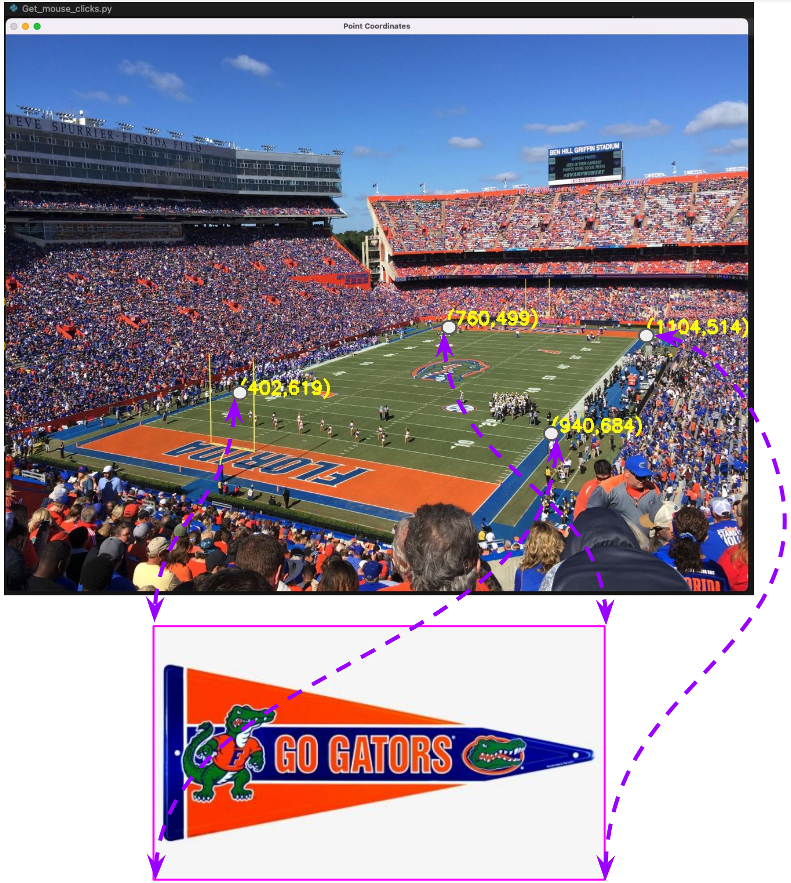


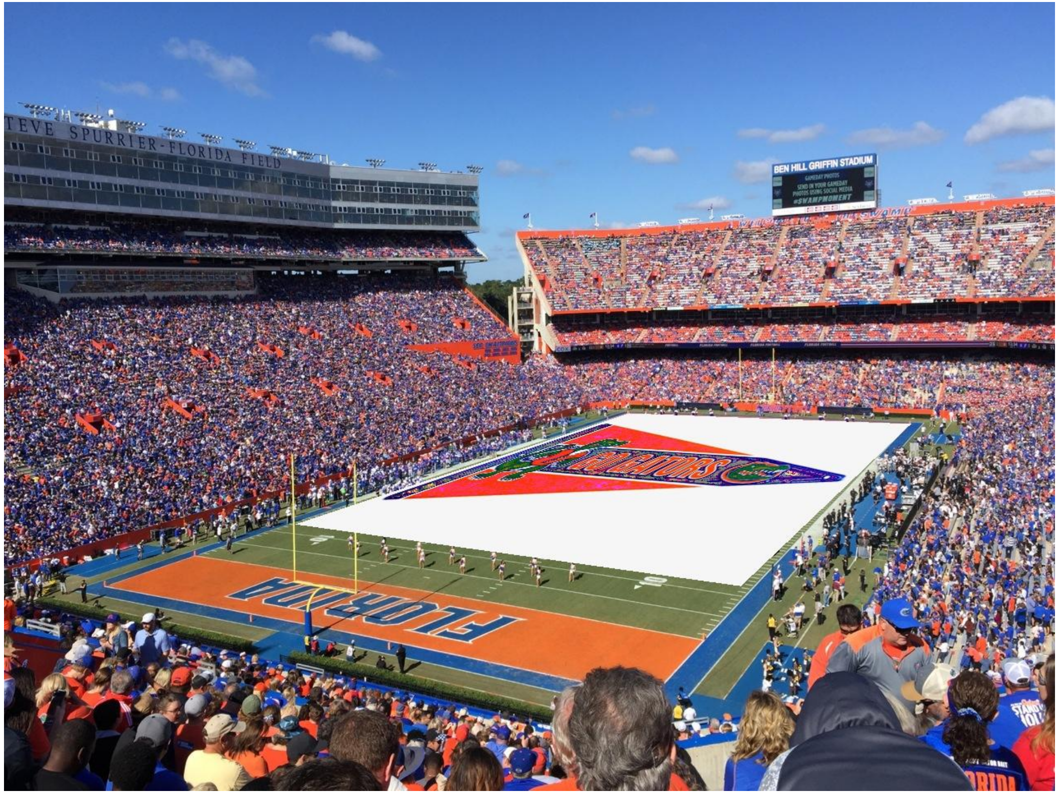
```
U: [(0, 0), (w, h), (w, 0), (0, h)]
V: find the points in the 'destination' image

Formulate the rank-deficient A matrix
 (see previous slide)

# SVD  composition
u, s, v = np.linalg.svd(A)

# solution: right-most singular vector into a 3x3
H = np.reshape(v[8], (3, 3))

#normalize and now we have H
H = (1/H.item(8)) * H

# warp the logo image into the destination image
imw, imh = im_dest.shape[1], im_dest.shape[0]
mask = cv2.warpPerspective(im_logo, H, (imw, imh))

# use mask to generate the final image (bonus +5)
… …
```

We are solving ***Ah = 0***

Making sure that ***H[3,3] = 1***

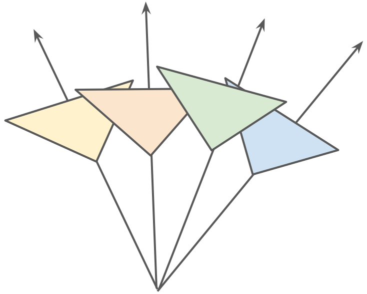
Homography Transformation: HH3-A





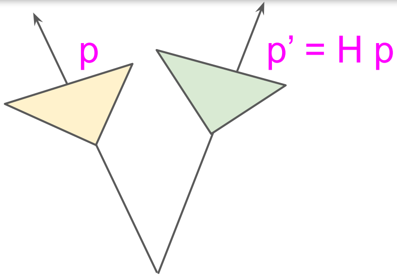
Homography: Pure Rotation

360 panorama



Self study!

- Take 8-10 images in pure rotation
- Create panorama using homography!
- See cs.brown.edu/courses/cs129/results/final/yunmiao/



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Points ***p*** in left camera correspond to points ***p'*** in right camera

- Pure rotation is related by homography ***p' = Hp***
- How to find **H**?

For a 3D point **X**

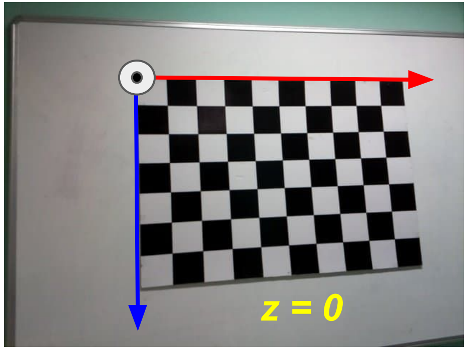
- Assuming first camera at origin, $\boldsymbol{p = K\,[I^{3x3}\ 0^{3x1}]\,X = KX}$
- Assuming the right camera is transformed by ***R, t***

  $\boldsymbol{p' = K\,[R\quad t=0^{3x1}]\,X = K\,R\,X\ = K\,R\,K^{-1}\,p\ = Hp}$
- *Thus* $\boldsymbol{H = K\,R\,K^{-1}} \leftrightarrow \boldsymbol{R = K^{-1}H\,K}$

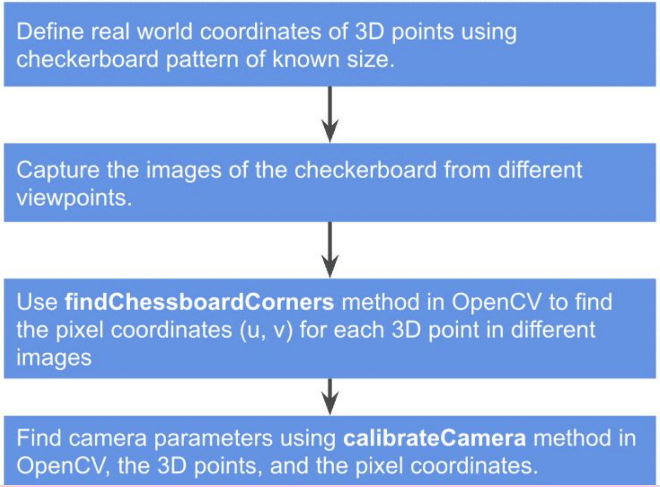
*We can recover camera parameters from **H***

Camera Calibration

$$\underbrace{\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\textbf{2D Point}} = \underbrace{\begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{K}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_0 \\ r_{21} & r_{22} & r_{23} & t_1 \\ r_{31} & r_{32} & r_{33} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{R} \qquad \boldsymbol{t}} \underbrace{\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}}_{\textbf{3D Point}}$$



$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{11} & p_{12} & p_{13} & p_{14} \\ p_{11} & p_{12} & p_{13} & p_{14} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$



Define real world coordinates of 3D points using checkerboard pattern of known size.

Capture the images of the checkerboard from different viewpoints.

Use **findChessboardCorners** method in OpenCV to find the pixel coordinates (u, v) for each 3D point in different images

Find camera parameters using **calibrateCamera** method in OpenCV, the 3D points, and the pixel coordinates.

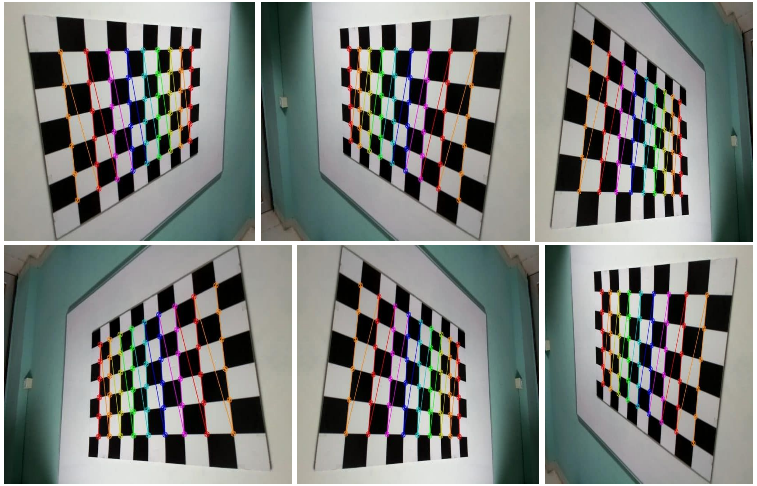
- Find known 2D-3D point pairs
- Find the projection matrix **P**: p_{11} to p_{34}
- Find the extrinsic parameters: **R** and **t**
- Find the intrinsic matrix **K**

See LearnOpenCV tutorial

Finding P = K [R t]

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$u = \frac{p_{11}x + p_{12}y + p_{13}z + p_{14}}{p_{31}x + p_{32}y + p_{33}z + p_{34}}, \quad v = \frac{p_{21}x + p_{22}y + p_{23}z + p_{24}}{p_{31}x + p_{32}y + p_{33}z + p_{34}}$$



$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -x_1u & -y_1u & -z_1u \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -x_1v & -y_1v & -z_1v \\ & & & & & & & \cdots & & & \\ & & & & & & & \cdots & & & \\ x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -x_nu & -y_nu & -z_nu \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -x_nv & -y_nv & -z_nv \end{bmatrix} \begin{bmatrix} p11 \\ p12 \\ p13 \\ p14 \\ p21 \\ p22 \\ h23 \\ p24 \\ p31 \\ p32 \\ p33 \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}$$

p34 = 1, use SVD to solve ***P***

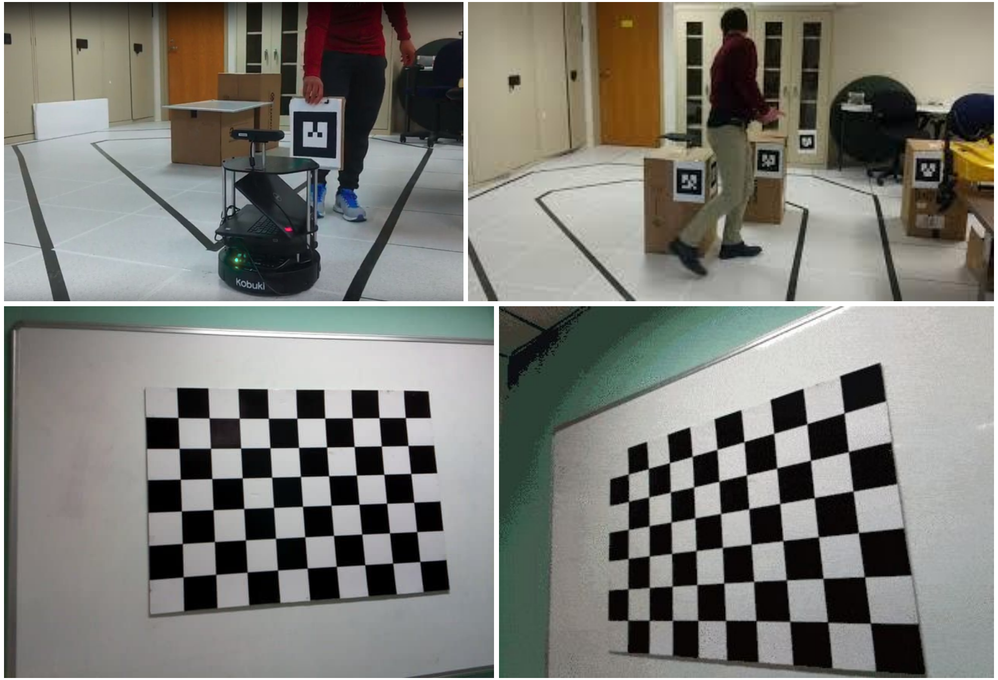
Explore the following pointers:

- How to get ***P*** efficiently by taking advantage of *z = 0?*
- How to find ***R, t,*** and ***K*** given the projection matrix ***P***?

Libraries:

- ROS, OpenCV
- CalTech Matlab code

Finding K: HH3-B



Find the intrinsic calibration matrix of either:

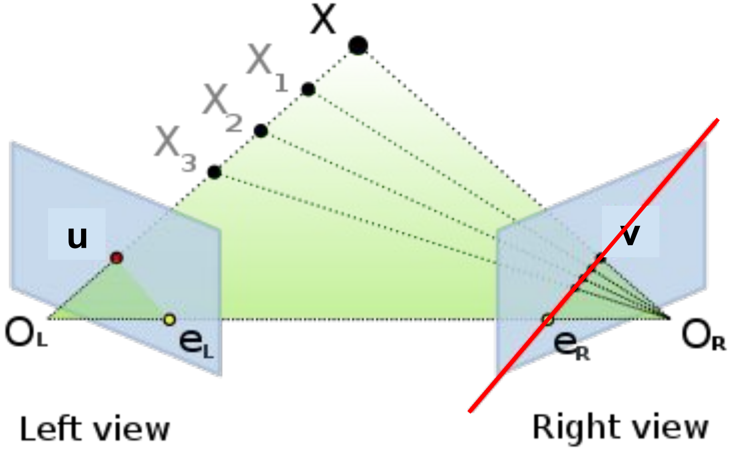
- Your TurtleBot-4 or
- Your cellphone camera

Process:

- Print a checkerboard and place it on a wall
- Use any of the following libraries:
  - ROS, OpenCV
  - CalTech Matlab code

  to calibrate your camera
- Report the **K**
- Check if the $\boldsymbol{f}$, $\boldsymbol{p_x}$, $\boldsymbol{p_y}$ are correct!

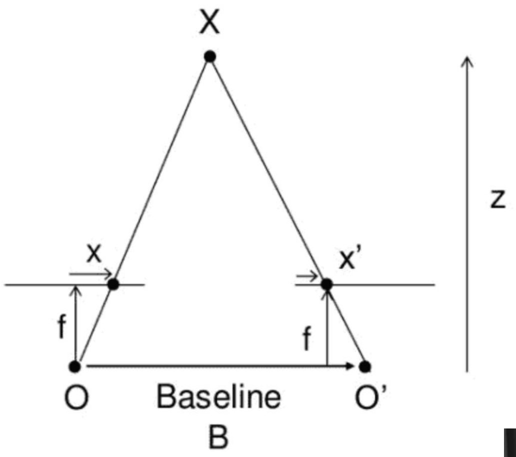
Stereo Cameras



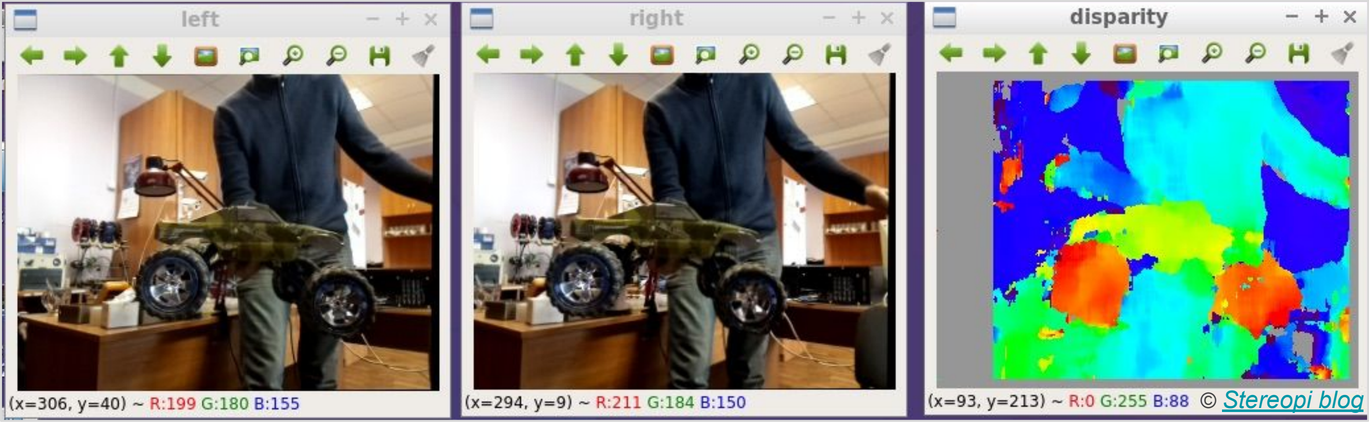
Stereo: two camera lenses are offset by a ‘baseline’

- Simulates human binocular vision: left and right views
- Relative depth perception
- Epipolar geometry: two-view case

Stereo Camera: Baseline and Disparity



$$\text{Disparity } x - x' = \frac{Bf}{Z}$$

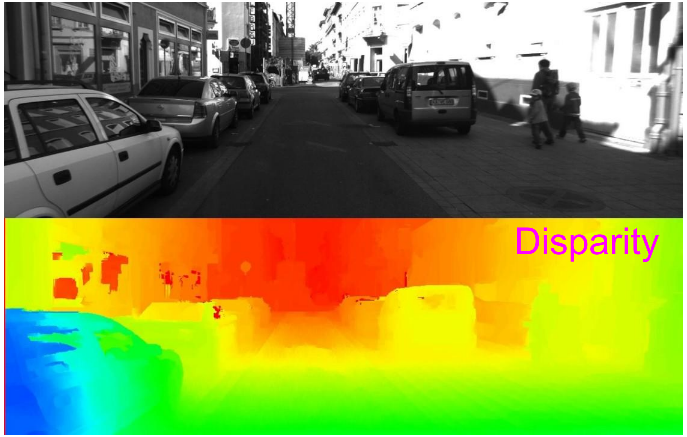


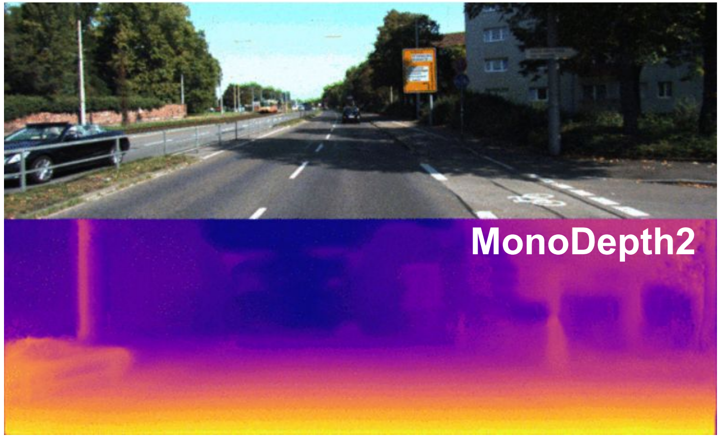
(x=306, y=40) ~ R:199 G:180 B:155

(x=294, y=9) ~ R:211 G:184 B:150

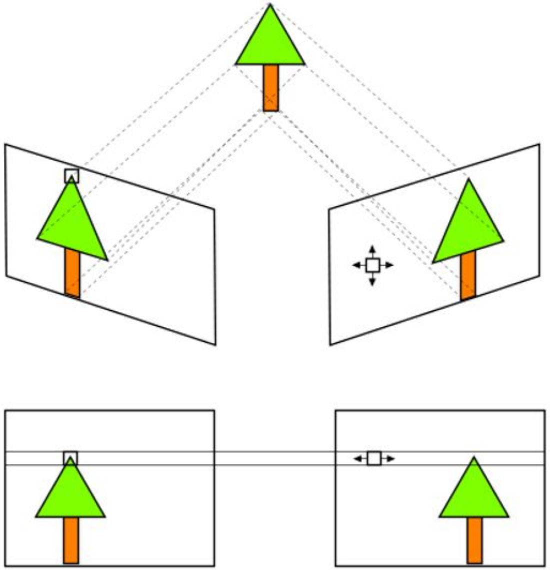
(x=93, y=213) ~ R:0 G:255 B:88

© Stereopi blog



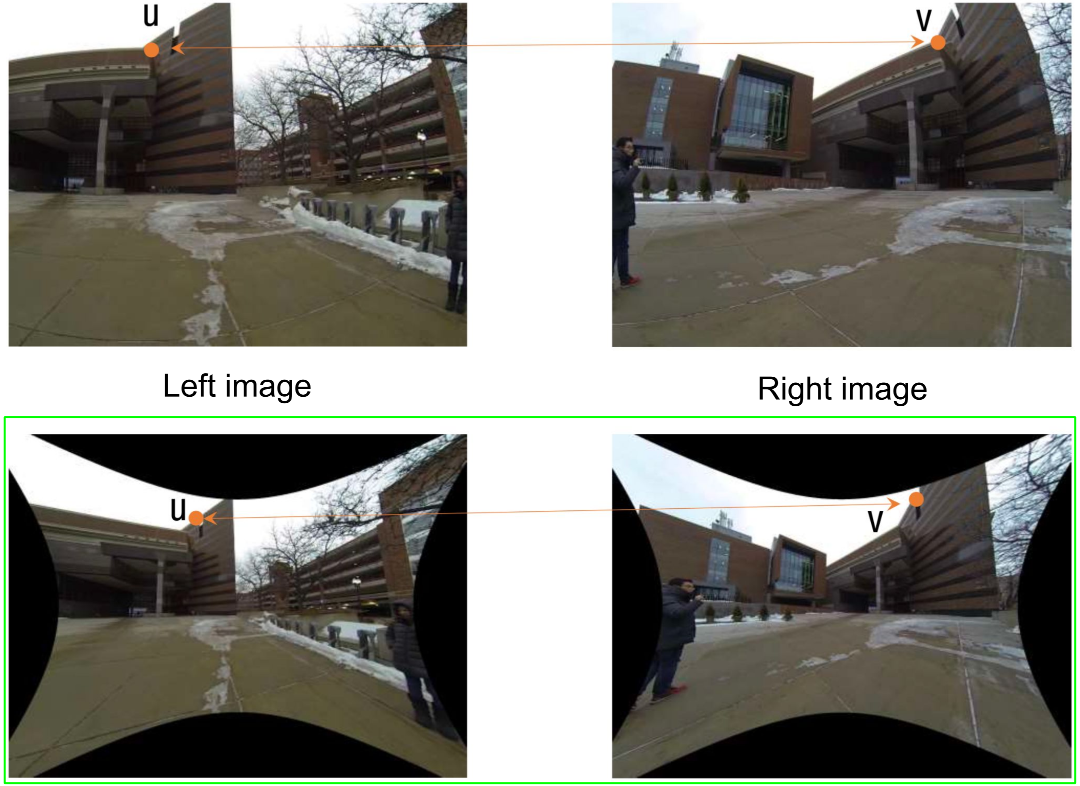


Stereo Rectification



Original images

Rectified images



Left image

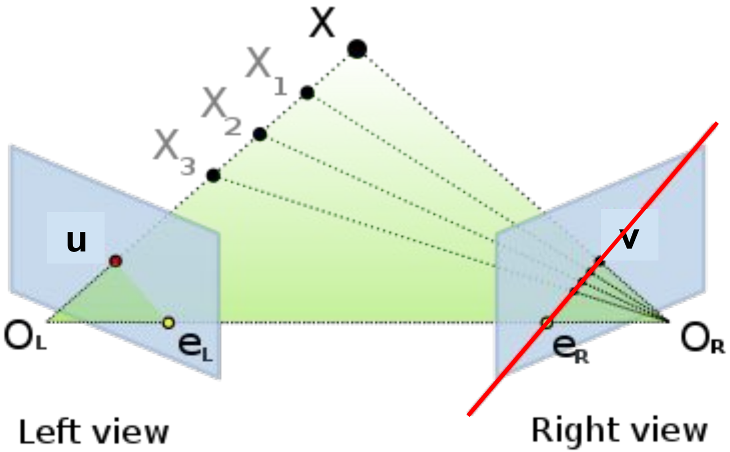
Right image

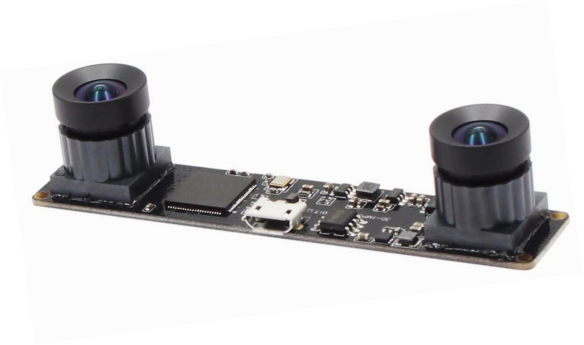
Rectify: make images upright (by homography)

- Horizontal displacement
- Makes feature matching easy!
- Epipolar lines become horizontal

© *Dr. Park*

Epipolar Geometry



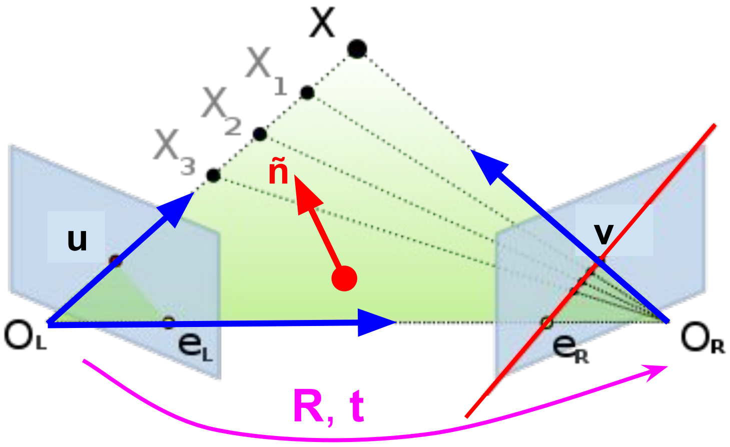


- **Camera centers: O_L, O_R**
- **Baseline:**
  - The line connecting the optical centers $B=O_LO_R$
- **Epipoles: e_L, e_R**
  - Intersection of image planes with the baseline
- **Epipolar plane: $O_L – O_R – X$**
  - Plane connecting the optical centers and a 3D point
- **Epipolar lines:**
  - Lines defined by the intersection of the epipolar plane and the two image planes

A pixel in the left image **u**

- Can correspond to X, X1, X2,... (any 3D point in OX line)
- Gets projected into the right epipolar line

Epipolar Geometry: Two-View



$$P_L = \mathbf{K}_\mathrm{L}\,[\mathbf{I} \quad \mathbf{0}] \quad \text{and} \quad P_R = \mathbf{K}_\mathrm{R}\,[\mathbf{R} \quad \mathbf{t}]$$

$$e_L = K_L\,[\mathbf{I} \quad \mathbf{0}] \begin{bmatrix} -R^T t \\ 1 \end{bmatrix} = -K_L R^T t$$

$$e_R = K_R\,[R \quad t] \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} = K_R t$$

Let’s formulate the epipolar constraint!

$\overline{O_L X} = K_L^{-1} u,$ and $\overline{O_L O_R} = -R^T t.$

$\overline{O_R X} = K_R^{-1} v,$ its projection in the left camera is: $-R^T t + R^T K_R^{-1} v$

The surface normal to the epipolar place: $\tilde{n} = R^T (t \times K_R^{-1} v)$

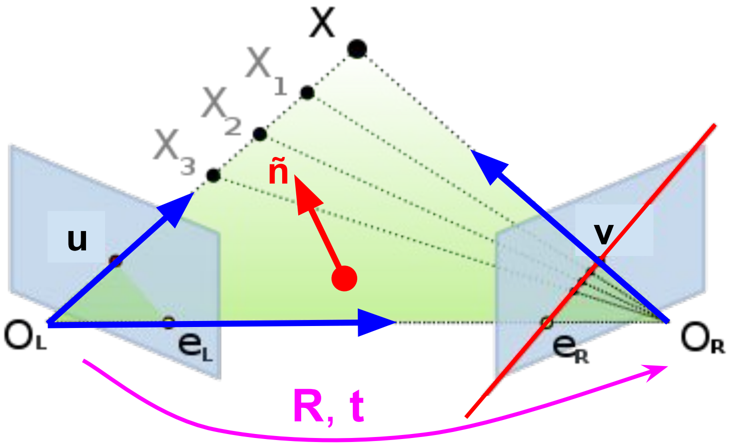
$$\tilde{n} = \overline{O_L O_R} \times (-R^T t + R\, K_R^{-1} v) = -R^T t \times (-R^T t + R^T\, K_R^{-1} v)$$

Proof:

$$\begin{aligned} \tilde{n} &= \overline{O_L O_R} \times (-R^T t + R\, K_R^{-1} v) \\ &= [-R^T t]_\times (-R^T t + R^T\, K_R^{-1} v) \\ &= [-R^T t]_\times R^T K_R^{-1} v \\ &= [R^T t]_\times R^T K_R^{-1} v \quad (\text{because } [a]_\times = [-a]_\times) \\ &= R^T ([t]_\times K_R^{-1} v) \quad (\text{because } R(a \times b) = Ra \times Rb) \end{aligned}$$

Epipolar constraint: $\overline{O_L X}^T \tilde{n} = u^T K_L^{-T} \tilde{n} = 0$

Epipolar Constraint



$$P_L = \mathbf{K}_L \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad P_R = \mathbf{K}_R \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$$

$$e_L = K_L \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} -R^T t \\ 1 \end{bmatrix} = -K_L R^T t$$

$$e_R = K_R \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} = K_R t$$

Epipolar constraint: $\overline{O_L X}^T \tilde{n} = u^T K_L^{-T} \tilde{n} = 0$

$$\rightarrow (K_L^{-1} u)^T R^T (t \times K_R^{-1} v) = 0$$

$$\rightarrow u^T \underbrace{K_L^{-T} R^T [t]_\times K_R^{-1}}_{F^T} v = 0$$

$$\rightarrow v^T F u = 0$$

$$F = K_R^{-T} \underbrace{[t]_\times R}_{E} K_L^{-1} = K_R^{-T} E K_L^{-1}$$

$$E = [t]_\times R$$

Essential Matrix: **E = t x R**

Fundamental Matrix: **$F = K_R^{-T}\, E\, K_L^{-1} \equiv K^{-T}\, E\, K^{-1}$**

- Relates **u** (left image point) and **v** (right image point)
- The constraint: **$v^T F u = 0$**

F: Fundamental Matrix

$\mathbf{v^T F u = 0}$ where $\mathbf{F} = K^{-T}\ \mathbf{E}\ K^{-1}$

Fundamental Matrix:

- **u** in the left image represent a line: **Fu=0** in right image
  - Which is why $\mathbf{v^T (Fu) = 0}$ makes sense!
  - It is the epipolar line **L = Fu**
  - The actual match **v** can be anywhere in this line
- The right epipole is also on this line
  - Therefore $\mathbf{e^T_R (Fu) = 0}$
- Similarly, **v** in the right image
  - Represent a line: $\mathbf{F^T v=0}$ in the left image
  - Left epipole satisfies $\mathbf{e^T_L (F^T v) = 0}$

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Properties

- Rank(F)=2
- DOF = 7

  // 9 - 1 (scale) -1 (rank) = 7

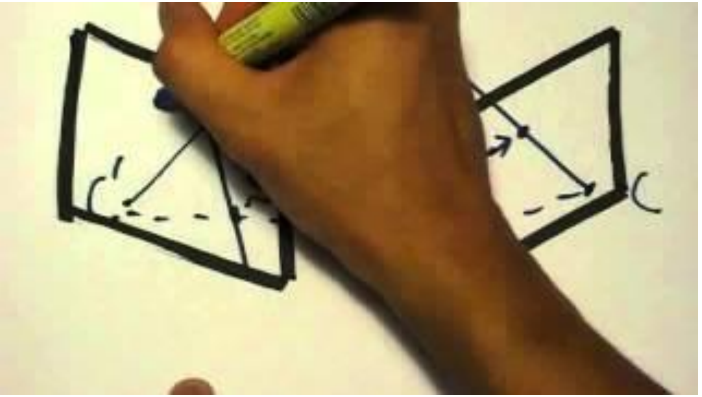
$$v^T F u = \begin{bmatrix} v^x & v^y & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u^x \\ u^y \\ 1 \end{bmatrix} = 0$$

Computing F: 8-point Algorithm

$\mathbf{v^T F u = 0}$ where $\mathbf{F} = K^{-T}\,\mathbf{E}\,K^{-1}$

$$v^T F u = \begin{bmatrix} v^x & v^y & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u^x \\ u^y \\ 1 \end{bmatrix} = 0 \longrightarrow$$

$$\begin{bmatrix} u_1^x v_1^x & u_1^y v_1^x & v_1^x & u_1^x v_1^y & u_1^y v_1^y & v_1^y & u_1^x & u_1^y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_m^x v_m^x & u_m^y v_m^x & v_m^x & u_m^x v_m^y & u_m^y v_m^y & v_m^y & u_m^x & u_m^y & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0_{m\times 1}$$

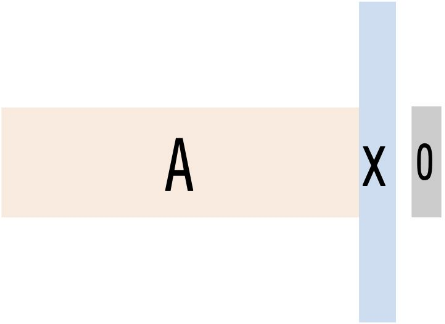


https://youtu.be/GMil9tpwE_Q

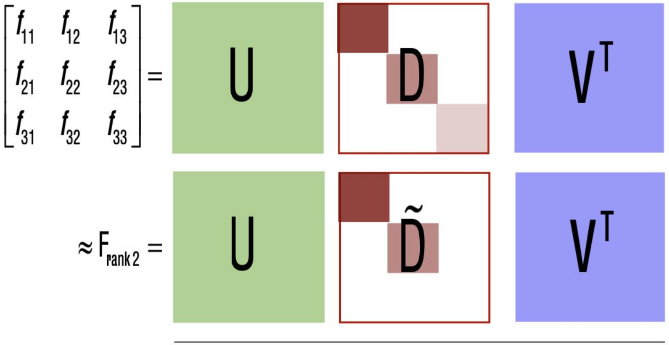
8-point Algorithm

- Match 8 feature points
  - Get $\mathbf{u_1{:}u_8}$ and $\mathbf{v_1{:}v_8}$
- Solve **Af = 0**
  - Use SVD! (see this)

Computing F: 8-point Algorithm



The solution is not necessarily satisfy rank 2 constraint.


f_{11} f_{12} f_{13}
f_{21} f_{22} f_{23}
f_{31} f_{32} f_{33}
=
U
D
V^T
$\approx F_{\text{rank 2}}$ =
U
$\tilde{D}$
V^T

SVD Cleanup

8-point Algorithm

- Match 8 feature points
  - Get $\mathbf{u}_1$:$\mathbf{u}_8$ and $\mathbf{v}_1$:$\mathbf{v}_8$
- Solve **Af = 0**
  - Use SVD! (see this)

```
f = SolveHomogeneousEq(A);

F = [f(1:3)'; f(4:6)'; f(7:9)'];

[u d v] = svd(F);
F1 = F;
d(3,3) = 0;
F = u*d*v';
```

SVD Cleanup

Computing F: 8-point + RANSAC

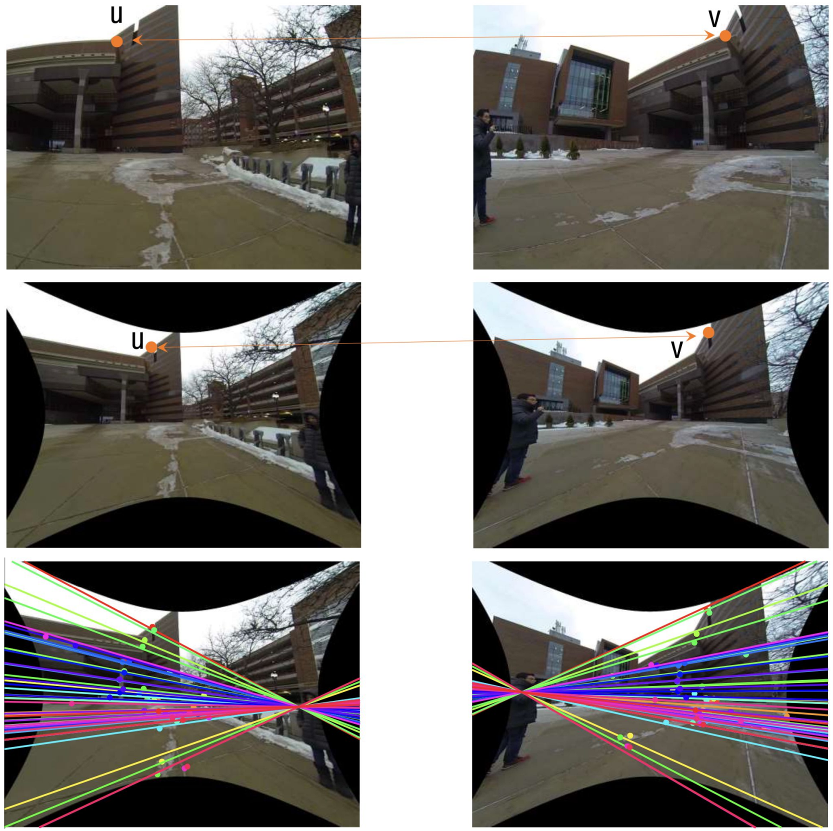
8-point + RANSAC

1. Perform 2D feature matching, eg: SIFT / FAST / ORB
2. Randomly choose 8 feature points
3. Solve **F** using 8-point algorithm
    a. Error term $\boldsymbol{\varepsilon}$: $|\mathbf{v}^T\mathbf{F}\mathbf{u}|$
    b. If $\boldsymbol{\varepsilon}$ is acceptable ($\boldsymbol{\varepsilon}$ < *threshold*)

        Return **F**
4. Otherwise save the best **F** (minimum $\boldsymbol{\varepsilon}$)
5. Repeat steps 2-4

See the implementation of

```
cv2.findFundamentalMat(U, V, cv2.FM_RANSAC)
```



© Dr. Park

E: Essential Matrix

$$\mathbf{F} = K^{-T}\,\mathbf{E}\,K^{-1} \Rightarrow \mathbf{E} = K^{T}\,\mathbf{F}\,K = \underline{\mathbf{t} \times \mathbf{R}}$$

Essential Matrix:

- Represents the same relationship (uncalibrated camera)
  - Also, `t = ± nullspace(Eᵀ)` // can you prove it?
- How to get camera pose (**R**, **t**) from **E**
  - SVD!

$$E = UDV^T = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \boldsymbol{v}_3^T \end{bmatrix}$$

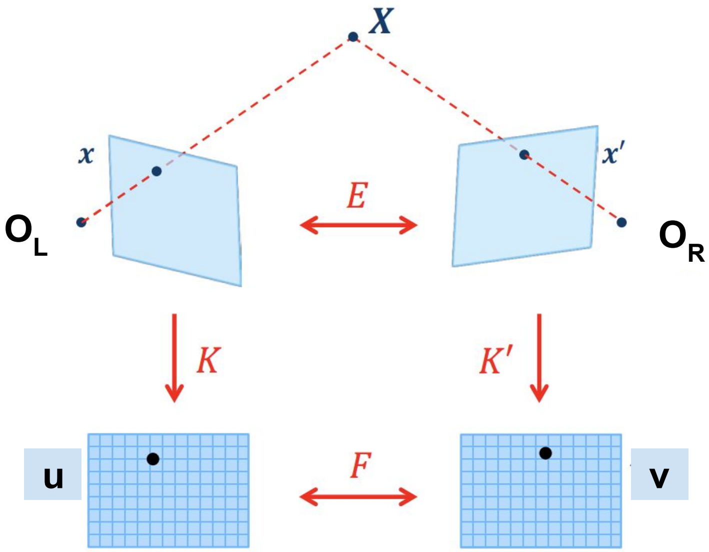
  - Then,

$$R \in \{UWV^T, UW^TV^T\}$$
$$\boldsymbol{t} = \pm\lambda \boldsymbol{u}_3; \quad \lambda \in \mathbb{R}\backslash 0$$

  - Where

$$W = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

See Proof



© Dr. Thomas

Camera Pose: R and t from E

Process

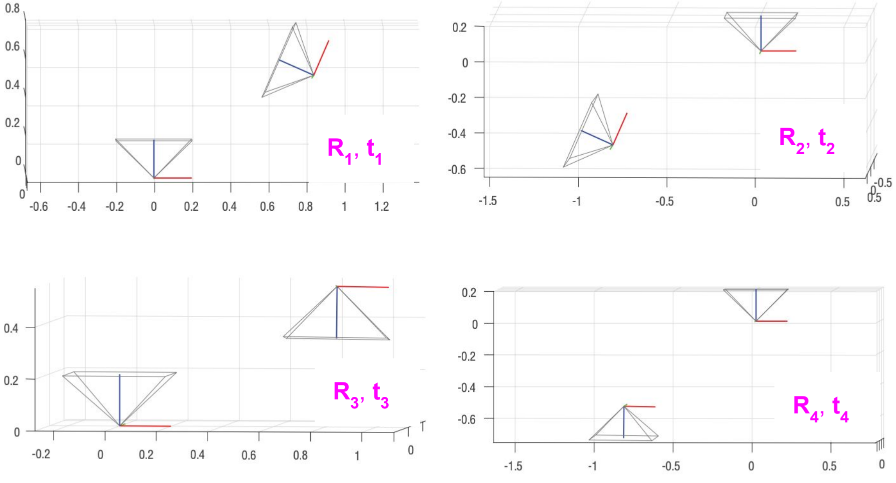
- SVD. $E = UDV^T = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \boldsymbol{v}_3^T \end{bmatrix}$

- Then, $R \in \{UWV^T, UW^TV^T\}$
  $\boldsymbol{t} = \pm\lambda\boldsymbol{u}_3;\ \ \lambda \in \mathbb{R}\backslash 0$

- Where
  $$W = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Get four solutions

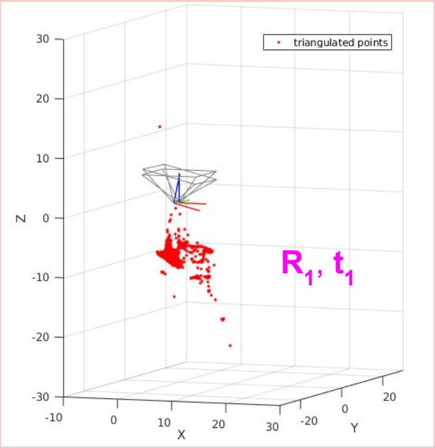
$$\begin{bmatrix} UWV^T & \boldsymbol{u}_3 \end{bmatrix}$$
$$\begin{bmatrix} UWV^T & -\boldsymbol{u}_3 \end{bmatrix}$$
$$\begin{bmatrix} UW^TV^T & \boldsymbol{u}_3 \end{bmatrix}$$
$$\begin{bmatrix} UW^TV^T & -\boldsymbol{u}_3 \end{bmatrix}$$

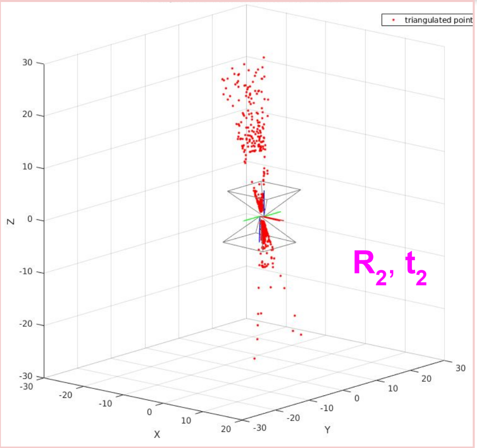


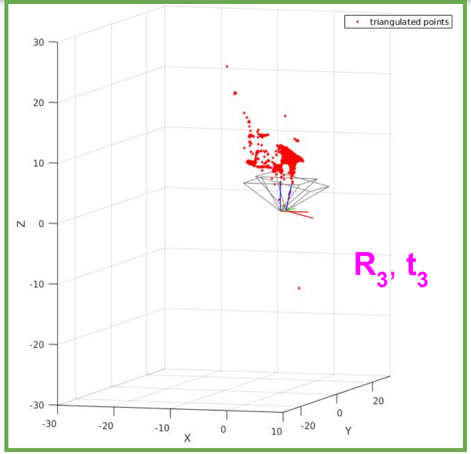
How to choose the right **R**, **t**?

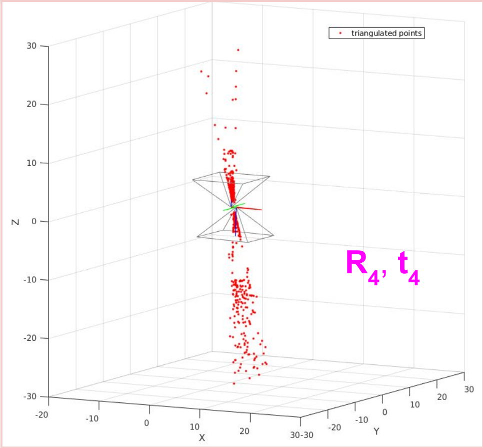
All 3D points should be in front of both cameras

Cheriality Condition: Finding the right **R** and **t**









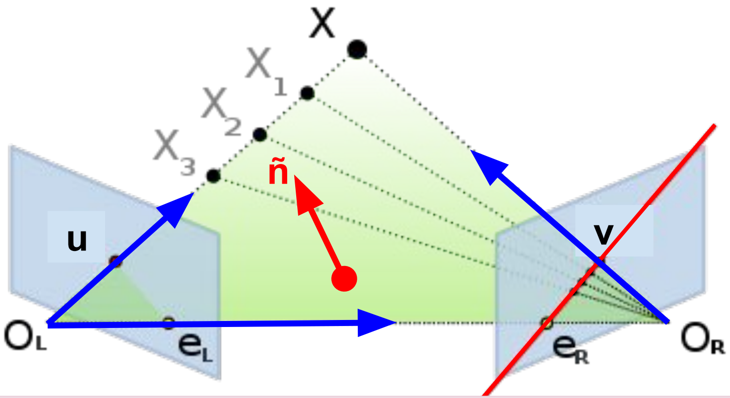
- **Pose options:** (R_1, t_1), (R_2, t_2), (R_3, t_3), (R_4, t_4)
- **Triangulate 3D points:**
  - Standard linear/nonlinear algorithms
  - Count the number of points in front of both cameras
- **Select the pose**
  - With the most number of points (in +*Z* direction)

Cheriality Condition:

A 3D point **x** is in front (+*Z* side) of the camera (R = [r1, r2, r3], t) if

$$\mathbf{r3^T(x - C) = r3^T(x + R^Tt) > 0}$$

Triangulation: u (2D) to X (3D)



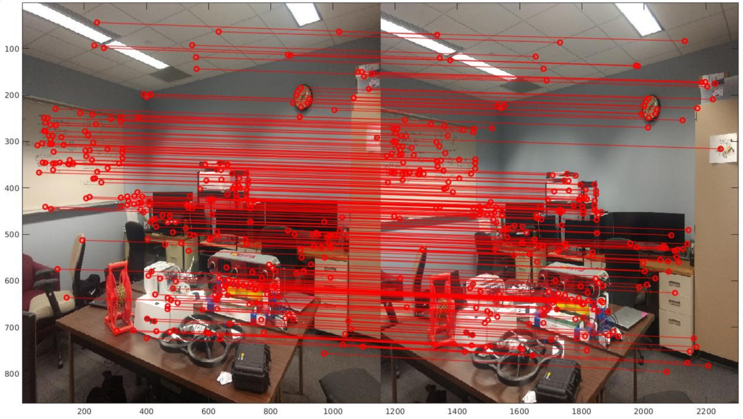
$$P_L = [\mathbf{I} \quad \mathbf{0}] \quad \text{and} \quad P_R = [\mathbf{R} \quad \mathbf{t}]$$

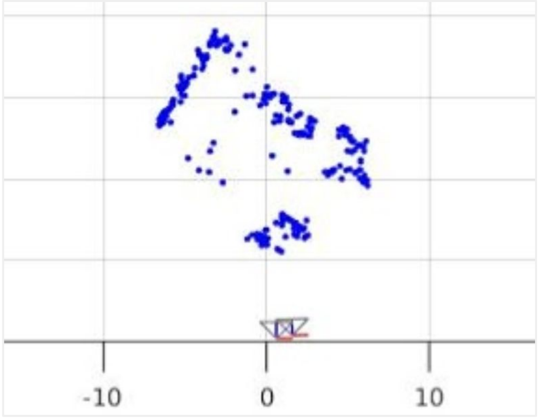
$$\begin{bmatrix}\mathbf{u}\\1\end{bmatrix} \times P_L \begin{bmatrix}\mathbf{X}\\1\end{bmatrix} = 0$$

$$\begin{bmatrix}\mathbf{v}\\1\end{bmatrix} \times P_R \begin{bmatrix}\mathbf{X}\\1\end{bmatrix} = 0$$

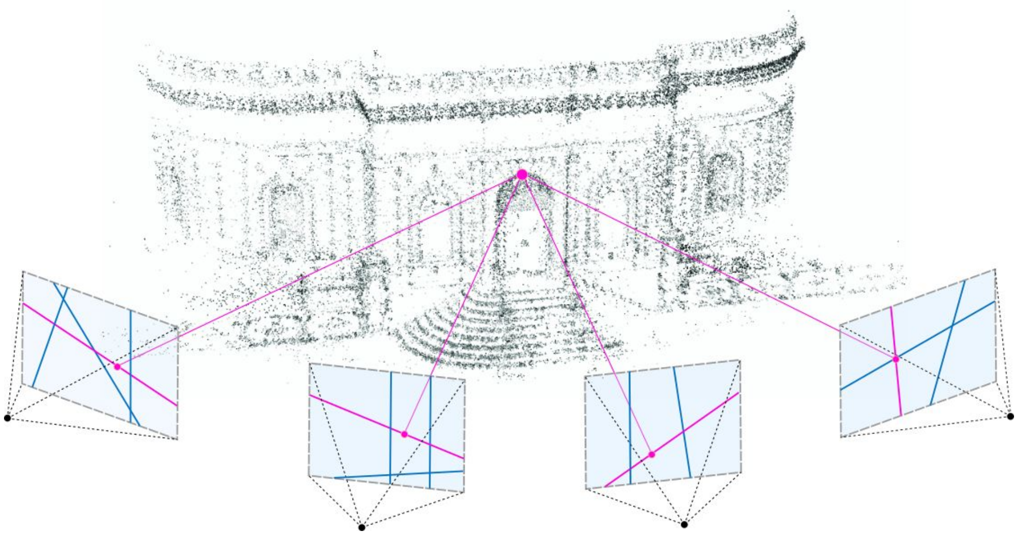
$$\begin{matrix}\begin{bmatrix}\mathbf{u}\\1\end{bmatrix}_\times P_L \\ \begin{bmatrix}\mathbf{v}\\1\end{bmatrix}_\times P_R\end{matrix} \begin{bmatrix}\mathbf{X}\\1\end{bmatrix} = 0$$

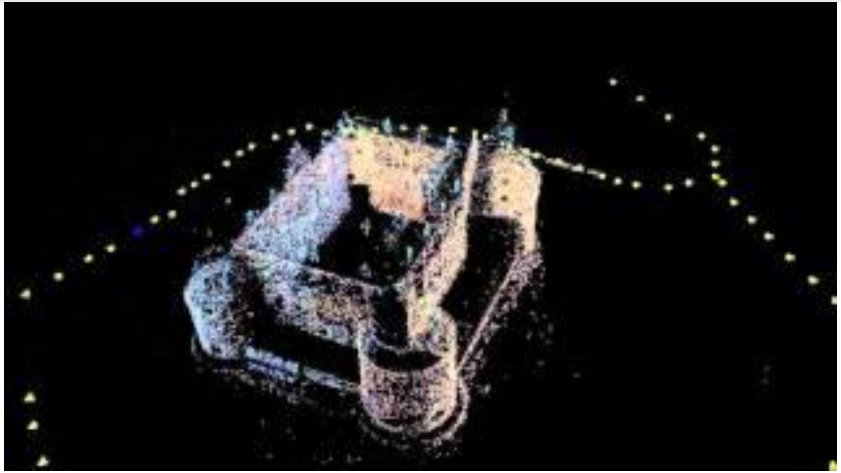
Solve for **X**





SfM: Structure from Motion





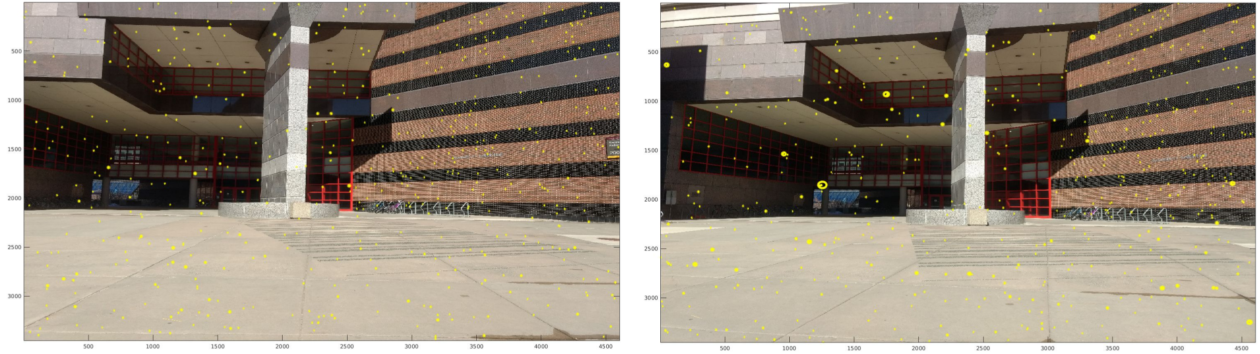
https://youtu.be/i7ierVkXYa8

SfM: Estimation of 3D structures from 2D image sequences.

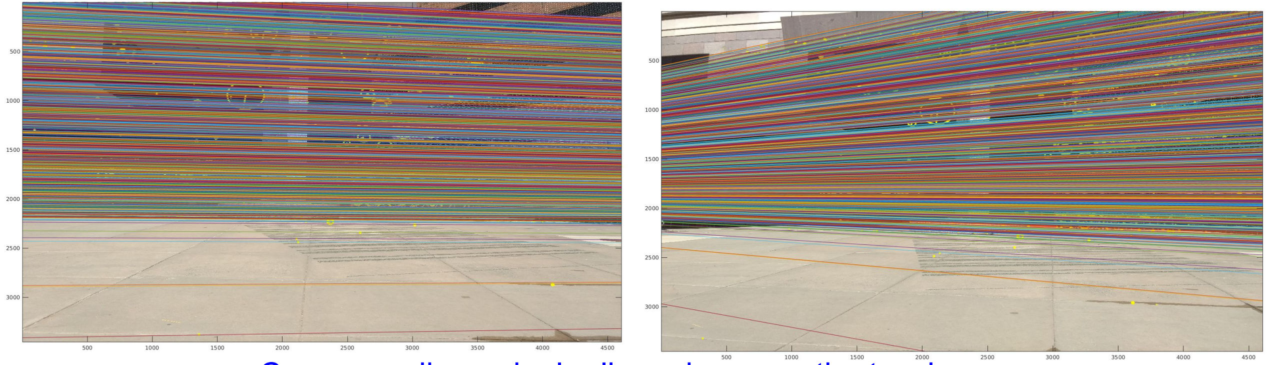
1. 2D feature detection in images: SIFT, ORB, FAST, etc.
2. Feature matching across viewpoints: KNN and ratio test
3. Estimating **F** from matched features: (u, v) pairs
4. Estimating **E** from **F**: **E** = K^T **F** K
5. Finding **R**, **t** from **E**: triangulation and Cheriality condition

6. Finding projection matrices: $\mathbf{P}_L$, $\mathbf{P}_R$
7. Triangulating all 3D points
8. **PnP** and nonlinear refinement
9. Bundle Adjustment (**BA**)

Two-view SfM: HH3-C

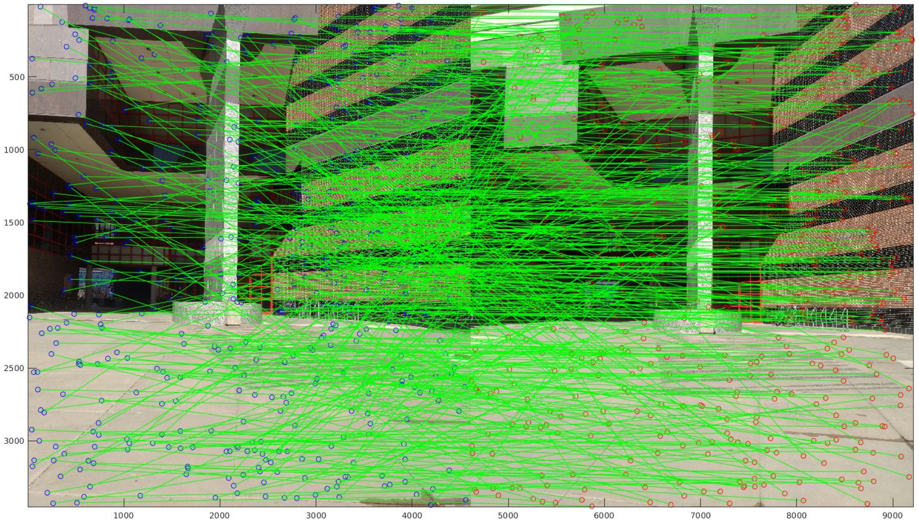


Left and right image taken by your calibrated camera

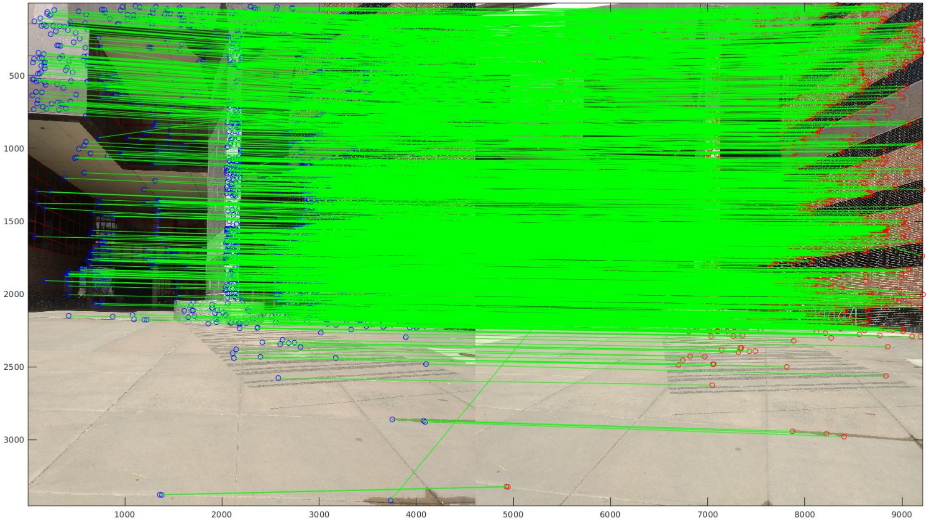


Corresponding epipolar lines drawn on the two images

Two-view SfM: HH3-C

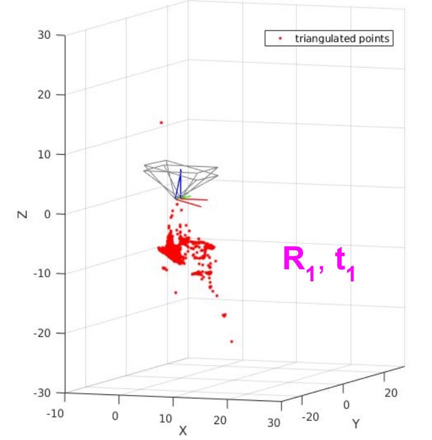


SIFT feature matches before ratio test (bonus part)

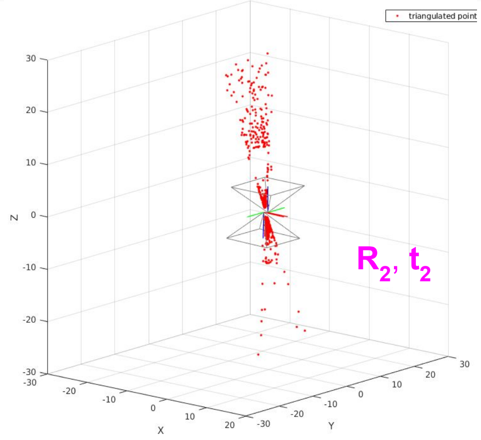


SIFT feature matches after ratio test (bonus part)

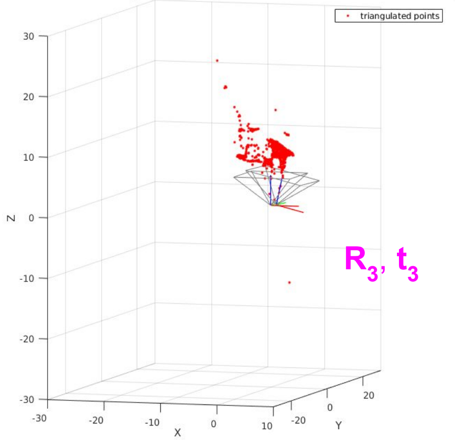
Two-view SfM: HH3-C



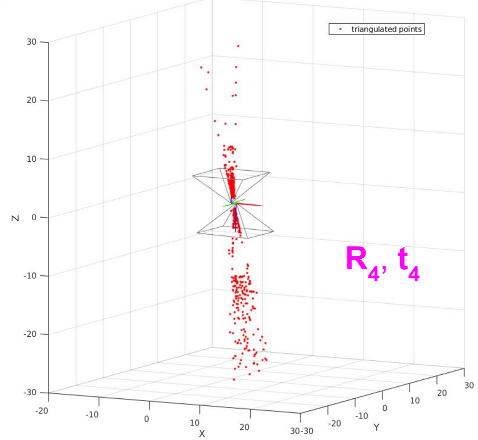
R_1, t_1



R_2, t_2



R_3, t_3



R_4, t_4

Complete the provided SfM pipeline template to do the following:

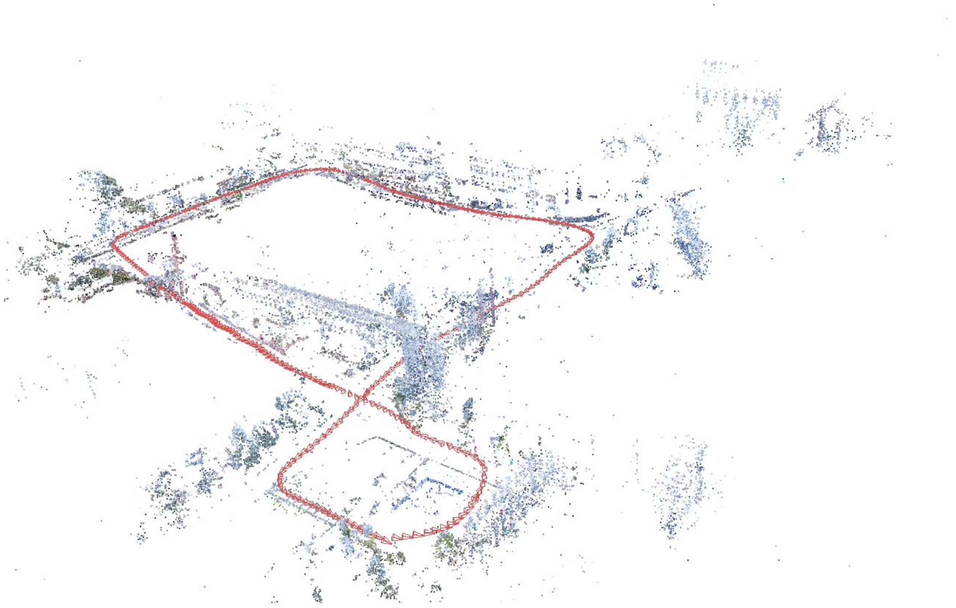
- *Visualizing the four prospective camera poses and corresponding triangulated point cloud*
- *Selection of the correct camera poses and 3D triangulation*
- *Visualization of the reconstructed scene*

Please check the HH3 pdf and blank code template in canvas!

Pointers: SfM and 3D Computer Vision

Recommended course materials

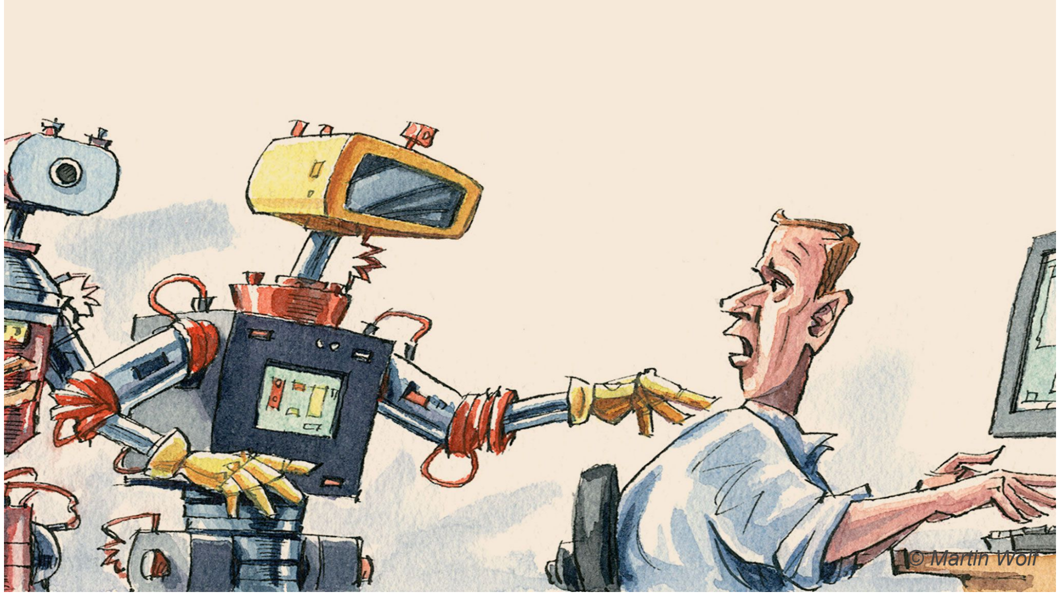
- Prof. Mubarak Shah (UCF):
  - Lecture videos
  - Course materials
- Prof. Hyun Soo Park (UMN):
  - Course materials
- Prof. James Hays (Brown)
  - Course materials
- Other resources
  - SfM by field robots (3D surveys)
  - Bundle adjustment
  - CMSC426 notes



© ANPL

Coming Next…

- Visual odometry
- Motion tracking and filtering by mobile robots
- Active planning and control
- SLAM: Simultaneous Localization and Mapping



© Martin Wolf