Summary: EEL 4930/5934: Autonomous Robots

Spring 2023

Department of ECE, University of Florida

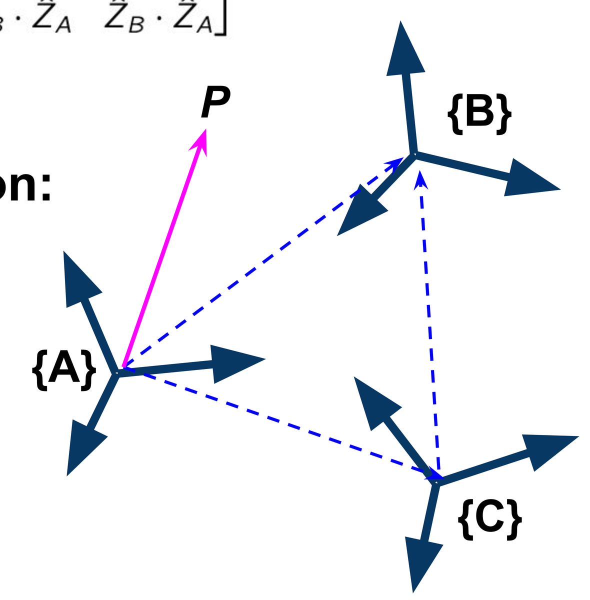
RoboPI: Robot Perception & Intellgence Laboratory

Spatial Transformation

- **Elementary rotations: X/Y/Z**

$$R_x(\theta)=\begin{bmatrix}1&0&0\\0&\cos\theta&-\sin\theta\\0&\sin\theta&\cos\theta\end{bmatrix}\quad R_y(\theta)=\begin{bmatrix}\cos\theta&0&\sin\theta\\0&1&0\\-\sin\theta&0&\cos\theta\end{bmatrix}\quad R_z(\theta)=\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}$$

- **Properties of a rotation matrix:**
  - $^A_BR=[{}^A\hat{X}_B\ {}^A\hat{Y}_B\ {}^A\hat{Z}_B]=\begin{bmatrix}\hat{X}_B\cdot\hat{X}_A&\hat{Y}_B\cdot\hat{X}_A&\hat{Z}_B\cdot\hat{X}_A\\\hat{X}_B\cdot\hat{Y}_A&\hat{Y}_B\cdot\hat{Y}_A&\hat{Z}_B\cdot\hat{Y}_A\\\hat{X}_B\cdot\hat{Z}_A&\hat{Y}_B\cdot\hat{Z}_A&\hat{Z}_B\cdot\hat{Z}_A\end{bmatrix}$
  - **Det(R) = 1**
  - **$R^T = R^{-1}$ and $R^TR = I_{3x3}$**
- **Homogeneous representation:**
  - $^AP={}^A_BR\,{}^BP+{}^AP_{Borg}$
  - $^AP={}^A_BT\,{}^BP$
  - $\begin{bmatrix}^AP\\1\end{bmatrix}=\underbrace{\begin{bmatrix}^A_BR&^AP_{Borg}\\0\ 0\ 0&1\end{bmatrix}}_{^A_BT}\begin{bmatrix}^BP\\1\end{bmatrix}$
  - Pure translation: R = I_{3x3}
  - Pure rotation: $^AP_{Borg}=0_{3x1}$



- **Compound transforms**
  - $^A_CT={}^A_BT\cdot{}^B_CT$

$$^A_CT=\begin{bmatrix}^A_BR\cdot{}^B_CR&^A_BR\cdot{}^BP_{Corg}+{}^AP_{Borg}\\0\ 0\ 0&1\end{bmatrix}$$

- **Inverse transforms**
  - $^B_AT\equiv\{^B_AR,\ ^BP_{Aorg}\}={}^A_BT^{-1}$
  - $^B_AT=\begin{bmatrix}^A_BR^T&-^A_BR^T\cdot{}^AP_{Borg}\\0\ 0\ 0&1\end{bmatrix}$
- **R(X-Y-Z) fixed angle**

$$^A_BR_{XYZ}(\gamma,\beta,\alpha)=R_Z(\alpha)\cdot R_Y(\beta)\cdot R_X(\gamma)$$

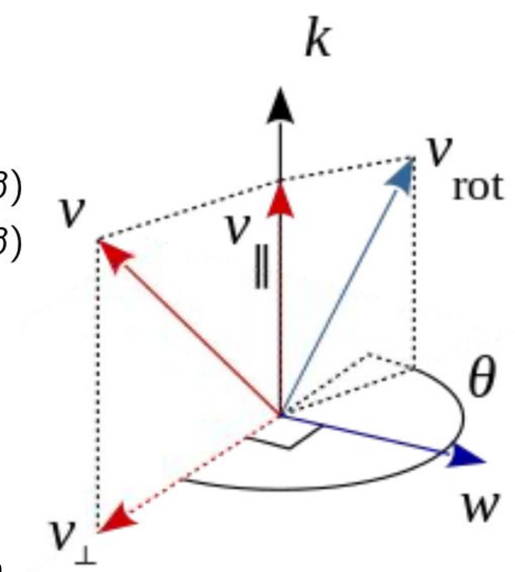
$$=\begin{bmatrix}c\alpha c\beta&c\alpha s\beta s\gamma-s\alpha c\gamma&c\alpha s\beta c\gamma+s\alpha s\gamma\\s\alpha c\beta&s\alpha s\beta s\gamma+c\alpha c\gamma&s\alpha s\beta c\gamma-c\alpha s\gamma\\-s\beta&c\beta s\gamma&c\beta c\gamma\end{bmatrix}$$

- **General solution:**

$\beta=Atan2(-r_{31},\sqrt{r_{11}^2+r_{21}^2})$

$\alpha=Atan2(\psi\ r_{21},\psi\ r_{11}),\quad\psi=sign(c\beta)$

$\gamma=Atan2(\psi\ r_{32},\psi\ r_{33}),\quad\psi=sign(c\beta)$



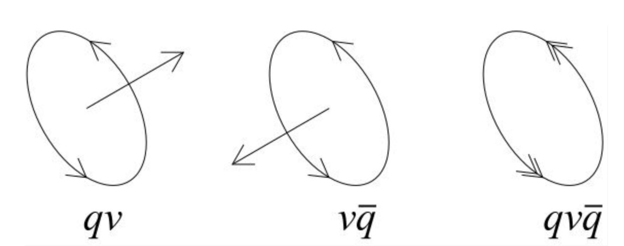
- **Rodrigues' rotation formula**

$$v_{rot}=v\cos\theta+(1-\cos\theta)(k\cdot v)k+(k\times v)\sin\theta$$

$$R(k,\theta)=\begin{bmatrix}k_x^2(1-c\theta)+c\theta&k_xk_y(1-c\theta)-k_zs\theta&k_xk_z(1-c\theta)+k_ys\theta\\k_xk_y(1-c\theta)+k_zs\theta&k_y^2(1-c\theta)+c\theta&k_yk_z(1-c\theta)-k_xs\theta\\k_xk_z(1-c\theta)-k_ys\theta&k_yk_z(1-c\theta)+k_xs\theta&k_z^2(1-c\theta)+c\theta\end{bmatrix}$$

Quaternions

- **Unit quaternion:**
  - $\mathbf{q}=\begin{bmatrix}\bar{q}\\q_4\end{bmatrix}=\begin{bmatrix}q_1\\q_2\\q_3\\q_4\end{bmatrix};\quad|q|=1$
  - $\mathbf{q}=q_1i+q_2j+q_3k+q_4$
  - $i^2=j^2=k^2=ijk=-1$



- **Rotation around a unit quaternion:**
  - $\mathbf{q}=\begin{bmatrix}\bar{k}\ s(\theta/2)\\c(\theta/2)\end{bmatrix}=\begin{bmatrix}k_x\ s(\theta/2)\\k_y\ s(\theta/2)\\k_z\ s(\theta/2)\end{bmatrix};\quad|q|=|k|=1$
  - $\mathbf{q}^{-1}=\frac{\mathbf{q}^*}{|q|}=\mathbf{q}^*=-q_1i-q_2j-q_3k+q_4=\begin{bmatrix}-q_1\\-q_2\\-q_3\\q_4\end{bmatrix}$

Quaternion Properties

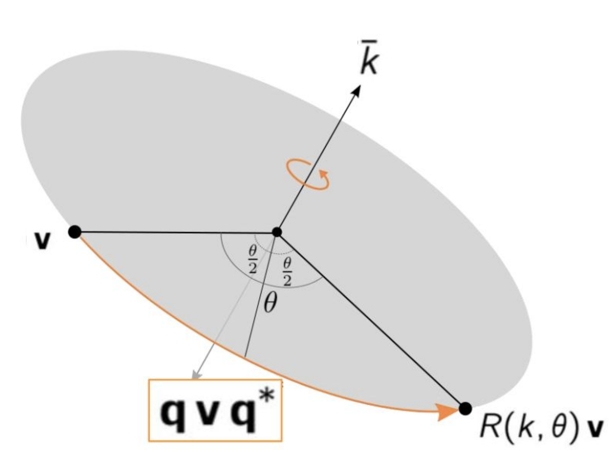
$qq^*=q^*q$

$|pq|=\sqrt{pq(pq)^*}$

$pq\neq qp$

$(op)q=o(pq)$

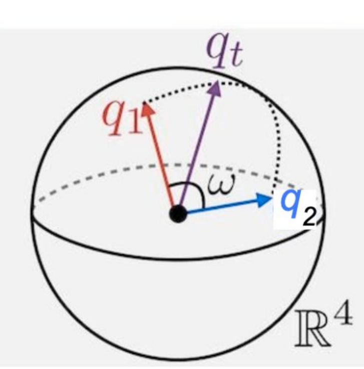
$o(p+q)=op+oq$



Quaternion SLERP: Spherical linear interpolation

Spherical Linear Interpolation (SLERP):

$$Slerp(q_1,q_2;\mu)=\frac{\sin(1-\mu)\theta}{\sin\theta}q_1+\frac{\sin\mu\theta}{\sin\theta}q_2$$




```
T1={R1,t1}
T2={R2,t2}
    ↓
T(μ) = ?
```

```
t_lerp = μ * t1 + (1-μ) * t2
q1 = quaternion_from_rotation(R1)
q2 = quaternion_from_rotation(R2)
q_slerp = quaternion_slerp(q1, q2, μ)
R_slerp = rotation_from_quaternion(q)
Tμ = {R_slerp, t_lerp}
```

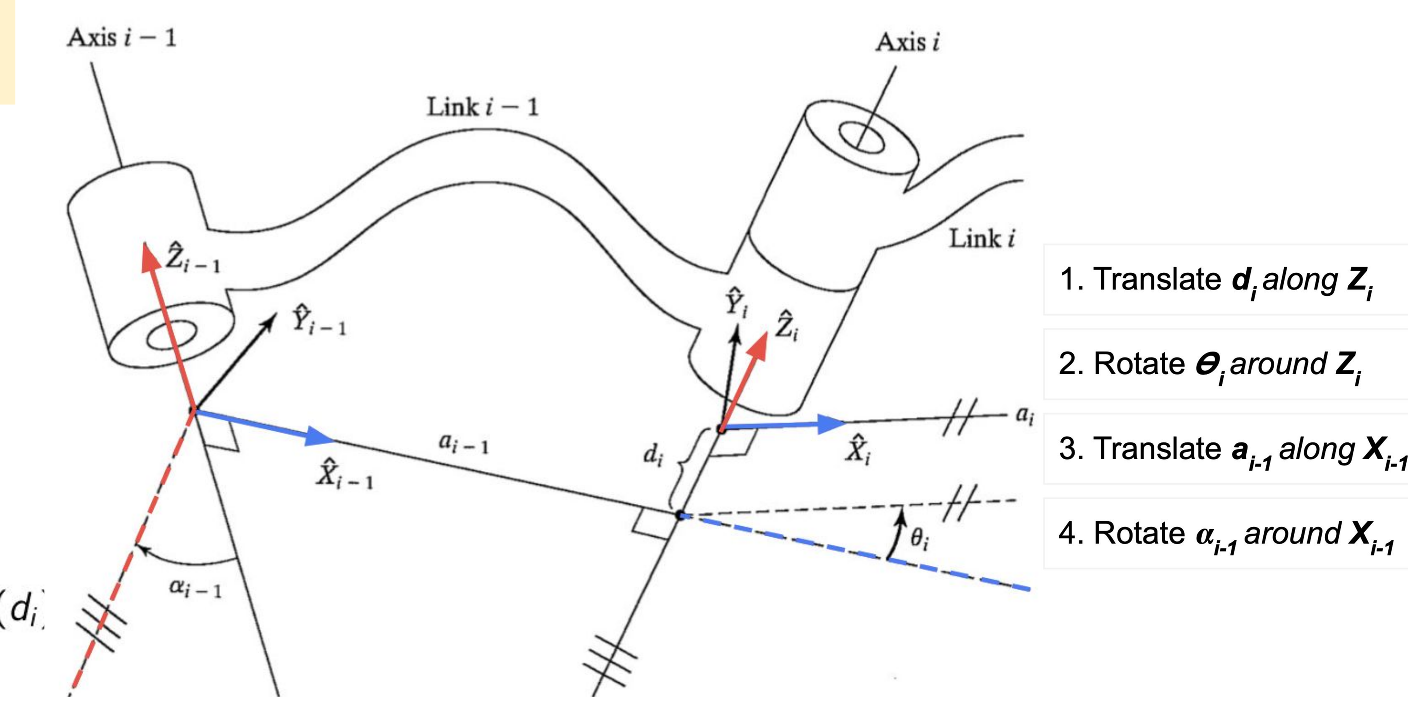
$$R(k,\theta)\mathbf{v}=\mathbf{q}\begin{bmatrix}\mathbf{v}\\0\end{bmatrix}\mathbf{q}^*=\mathbf{q}\begin{bmatrix}\mathbf{v}\\0\end{bmatrix}\mathbf{q}^{-1}=\mathbf{q}\begin{bmatrix}v_x\\v_y\\v_z\\0\end{bmatrix}\mathbf{q}^*$$

$$R(k,\theta)=\begin{bmatrix}k_x^2(1-c\theta)+c\theta&k_xk_y(1-c\theta)-k_zs\theta&k_xk_z(1-c\theta)+k_ys\theta\\k_xk_y(1-c\theta)+k_zs\theta&k_y^2(1-c\theta)+c\theta&k_yk_z(1-c\theta)-k_xs\theta\\k_xk_z(1-c\theta)-k_ys\theta&k_yk_z(1-c\theta)+k_xs\theta&k_z^2(1-c\theta)+c\theta\end{bmatrix}$$

$$=\begin{bmatrix}1-2q_2^2-2q_3^2&2(q_1q_2-q_3q_4)&2(q_1q_3+q_2q_4)\\2(q_1q_2+q_3q_4)&1-2q_1^2-2q_3^2&2(q_2q_3-q_1q_4)\\2(q_1q_3-q_2q_4)&2(q_2q_3+q_1q_4)&1-2q_1^2-2q_2^2\end{bmatrix}$$

Forward Kinematics: Manipulators

- **DH notation:** four parameters
  - Link twist: $\alpha_{i-1}\equiv angle(Z_{i-1},\ Z_i)_{X_{i-1}}$
  - Link length: $a_{i-1}\equiv distance(Z_{i-1},\ Z_i)_{X_{i-1}}$
  - Link offset: $d_i\equiv distance(X_{i-1},\ X_i)_{Z_i}$
  - Joint angle: $\theta_i\equiv angle(X_{i-1},\ X_i)_{Z_i}$



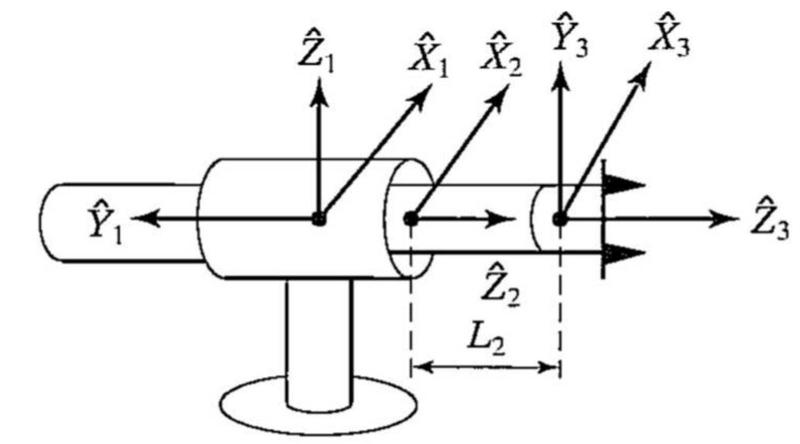
- **Transformation: from {i} to {i-1}**

$$^{i-1}_iT=Rotate_{X_{i-1}}(\alpha_{i-1})\times Translate_{X_{i-1}}(a_{i-1})\times Rot_{Z_i}(\theta_i)\times Translate_{Z_i}(d_i)$$

$$=\begin{bmatrix}C\theta_i&-S\theta_i&0&a_{i-1}\\C\alpha_{i-1}\cdot S\theta_i&C\alpha_{i-1}\cdot C\theta_i&-S\alpha_{i-1}&-d_i\cdot S\alpha_{i-1}\\S\alpha_{i-1}\cdot S\theta_i&S\alpha_{i-1}\cdot C\theta_i&C\alpha_{i-1}&d_i\cdot C\alpha_{i-1}\\0&0&0&1\end{bmatrix}$$

$\theta_1, \theta_1, \ldots, \theta_n$ → **Forward Kinematics** → ^j_iT

- **Example**



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$\pi/2$	0	d_2	0
3	0	0	L_2	θ_3

Say: $\theta_1=\pi/3$, $\theta_3=\pi/6$, $d_2=0.5$, $L_2=1$

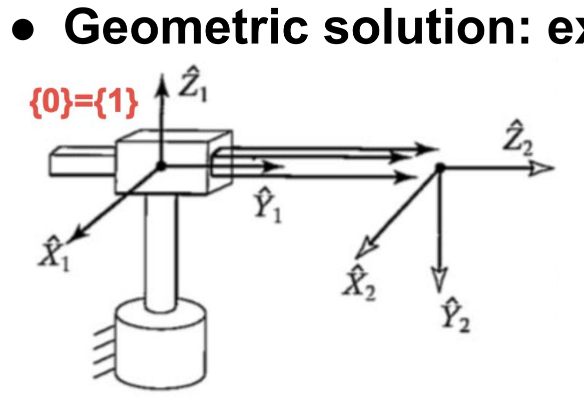
$$^0_1T=\begin{bmatrix}c\theta_1&-s\theta_1&0&0\\s\theta_1&c\theta_1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\xrightarrow{(\theta_1=\frac{\pi}{3})}\begin{bmatrix}0.5&-0.866&0&0\\0.866&0.5&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$^1_2T=\begin{bmatrix}1&0&0&0\\0&0&-1&-d_2\\0&1&0&0\\0&0&0&1\end{bmatrix}\xrightarrow{(d_2=0.5)}\begin{bmatrix}1&0&0&0\\0&0&-1&-0.5\\0&1&0&0\\0&0&0&1\end{bmatrix}$$

$$^2_3T=\begin{bmatrix}c\theta_3&-s\theta_3&0&0\\s\theta_3&c\theta_3&0&0\\0&0&1&L_2\\0&0&0&1\end{bmatrix}\xrightarrow{(\theta_3=\frac{\pi}{6}),L_2=1}\begin{bmatrix}0.866&-0.5&0&0\\0.5&0.866&0&0\\0&0&1&1\\0&0&0&1\end{bmatrix}$$

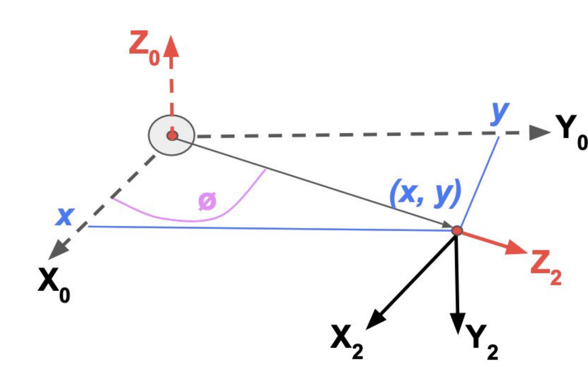
Inverse Kinematics: Manipulators

- **Geometric solution: example #1**



^j_iT → **Inverse Kinematics** → $\theta_1, \theta_1, \ldots, \theta_n$

$$\sin\phi=\frac{y}{\sqrt{x^2+y^2}}\qquad\cos\phi=\frac{x}{\sqrt{x^2+y^2}}$$



- **Geometric solution: example #2**

$x=L_1c_1+L_2c_{12}=L_1c_1+L_2c_1c_2-L_2s_1s_2=(L_1+L_2c_2)c_1-(L_2s_2)s_1$

$y=L_1s_1+L_2s_{12}=L_1s_1+L_2s_1c_2+L_2c_1s_2=(L_1+L_2c_2)s_1+(L_2s_2)c_1$

$x=k_1c_1-k_2s_1$, $y=k_1s_1+k_2c_1$

$k_1=r\cos\gamma$, $k_2=r\sin\gamma$, $r=\sqrt{k_1^2+k_2^2}$

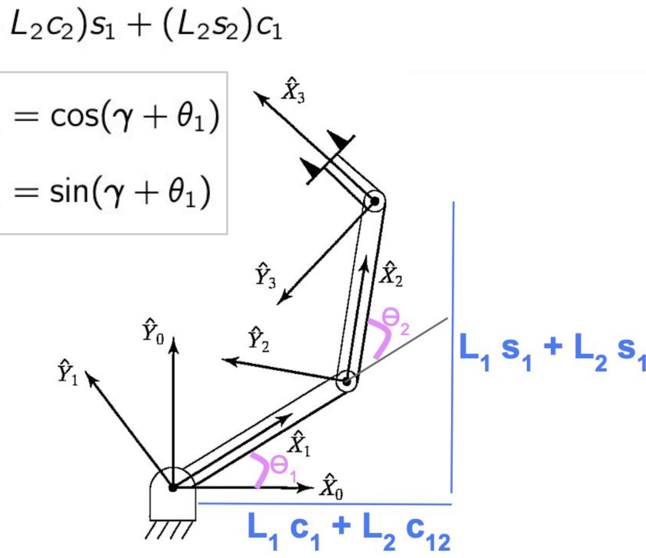
$\frac{x}{r}=c\gamma c_1-s\gamma s_1=\cos(\gamma+\theta_1)$, $\frac{y}{r}=c\gamma s_1+s\gamma c_1=\sin(\gamma+\theta_1)$

$\gamma+\theta_1=Atan2(\frac{y}{r},\frac{x}{r})=Atan2(y,x)$

$\theta_1=Atan2(y,x)-Atan2(k_2,k_1)$

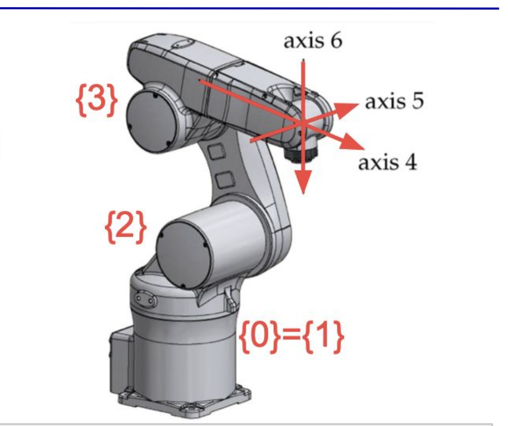
$\phi=\theta_1+\theta_2+\theta_3=Atan2(s_{123},c_{123})$

$\theta_3=Atan2(s_{123},c_{123})-\theta_1-\theta_2$



Algebraic solution: Pieper robot

6R manipulator: Axis 4, 5, 6 intersect



$^0_6P={}^0_5P={}^0_4P$

$$^0_4T=\begin{bmatrix}^0_4R&^0P_{4org}\\0\ 0\ 0&1\end{bmatrix}$$

$$^0P_{4org}=\begin{bmatrix}x\\y\\z\\1\end{bmatrix}={}^0_1T\cdot{}^1_2T\cdot{}^2_3T\cdot{}^3P_{4org}={}^0_1T\cdot{}^1_2T\cdot\begin{bmatrix}f_1\\f_2\\f_3\\1\end{bmatrix}={}^0_1T\cdot\begin{bmatrix}g_1\\g_2\\g_3\\1\end{bmatrix}=\begin{bmatrix}c_1g_1-c_1g_2\\s_1g_1+c_1g_2\\g_3\\1\end{bmatrix}$$

$f_1=a_3c_3+d_4s\alpha_3s_3+a_2$

$f_2=a_3c\alpha_2s_3-d_4s\alpha_3c\alpha_2c_3-d_4s\alpha_2c\alpha_3-d_3s\alpha_2$

$f_3=a_3s\alpha_2s_3-d_4s\alpha_3s\alpha_2c_3+d_4c\alpha_2c\alpha_3+d_3c\alpha_2$

$g_1=c_2f_1-s_2f_2+a_1$

$g_2=s_2c\alpha_1f_1+c_2c\alpha_1f_2-s\alpha_1f_3-d_2s\alpha_1$

$g_3=s_2s\alpha_1f_1+c_2s\alpha_1f_2+c\alpha_1f_3+d_2c\alpha_1$

$r=x^2+y^2+z^2=g_1^2+g_2^2+g_3^2$

$\quad=f_1^2+f_2^2+f_3^2+a_1^2+d_2^2+2d_2f_3+2a_1(c_2f_1-s_2f_2)$

$r=2a_1(k_1c_2+k_2s_2)+k_3$

$z=g_3=s\alpha_1(k_1s_2-k_2c_2)+k_4$

$k_1=f_1\qquad k_2=-f_2$

$k_3=f_1^2+f_2^2+f_3^2+a_1^2+d_2^2+2d_2f_3$

$k_4=f_3c\alpha_1+d_2c\alpha_1$

⇒ Solve for Θ_3
⇒ With Θ_3 solve for Θ_2
⇒ Then solve for Θ_1

⇒ Finally, solve for Θ_4, Θ_5, Θ_6

Summary: EEL 4930/5934: Autonomous Robots

Spring 2023

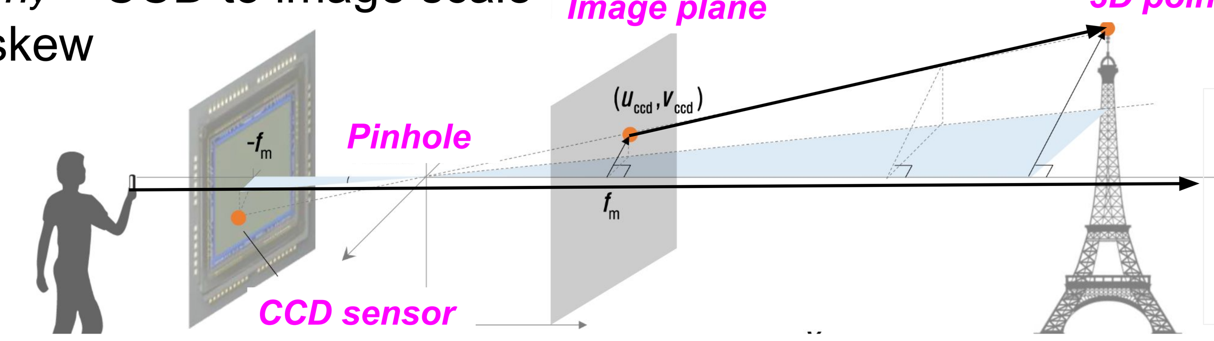
Department of ECE, University of Florida

RoboPI: Robot Perception & Intellgence Laboratory

Visual perception

- **Intrinsic matrix: K**
  - f = focal length
  - p_x, p_y = imw/2, imh/2
  - m_x, m_y = CCD to image scale
  - s = skew

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

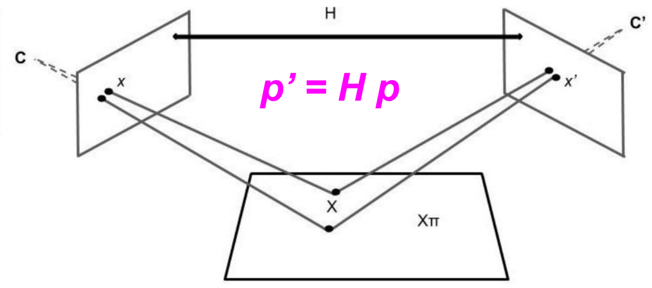


- **Perspective transformation: homography**

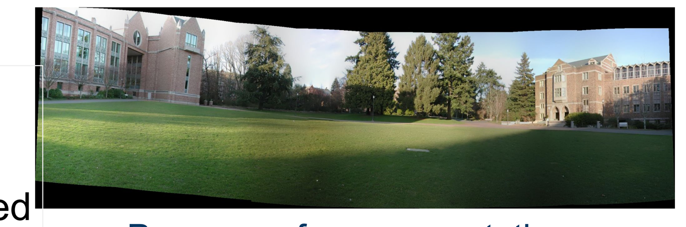
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

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



Homography transformation
- Pure camera rotation
- Same planar surface viewed by two cameras



Panorama from pure rotation

- **Projection matrix: P**
  - $\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K [R \;\; t] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = P \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$
  - $\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_0 \\ r_{21} & r_{22} & r_{23} & t_1 \\ r_{31} & r_{32} & r_{33} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$
  - $\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$

P = K [R t]

Camera calibration

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

↓

Capture the images of the checkerboard from different viewpoints.

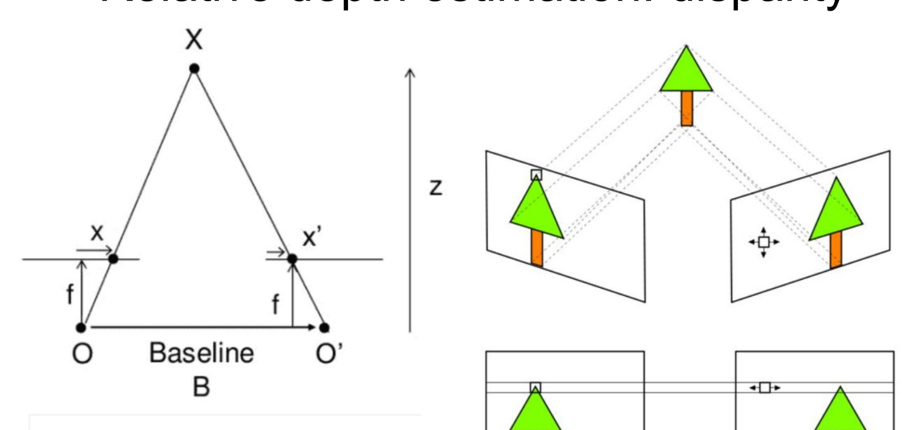
↓

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.

- **Stereo geometry**
  - Two cameras offset by a 'baseline'
  - Relative depth estimation: disparity

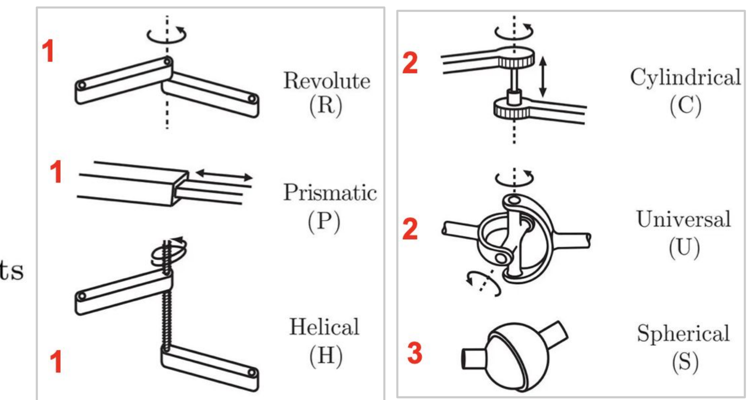


Disparity $x - x' = \frac{Bf}{Z}$

Robot locomotion

- **DOF: degrees of freedom**

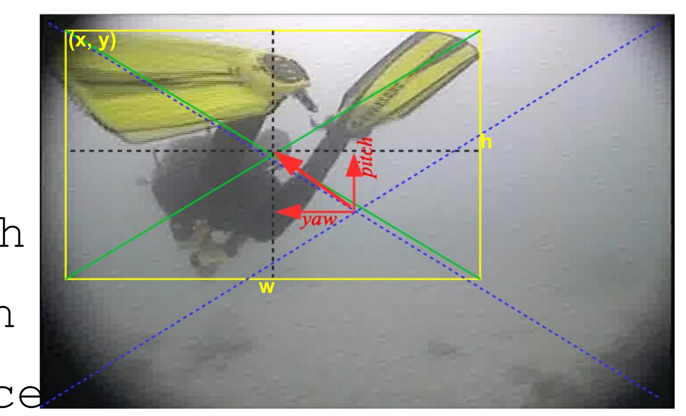
$$\text{dof} = \underbrace{m(N-1)}_{\text{rigid body freedoms}} - \underbrace{\sum_{i=1}^{J} c_i}_{\text{joint constraints}}$$



- **Motion gaits**
  - Sliding gaits, crawling gaits, swimming gaits
  - Legged (walking) gaits, wheel (rolling) gaits

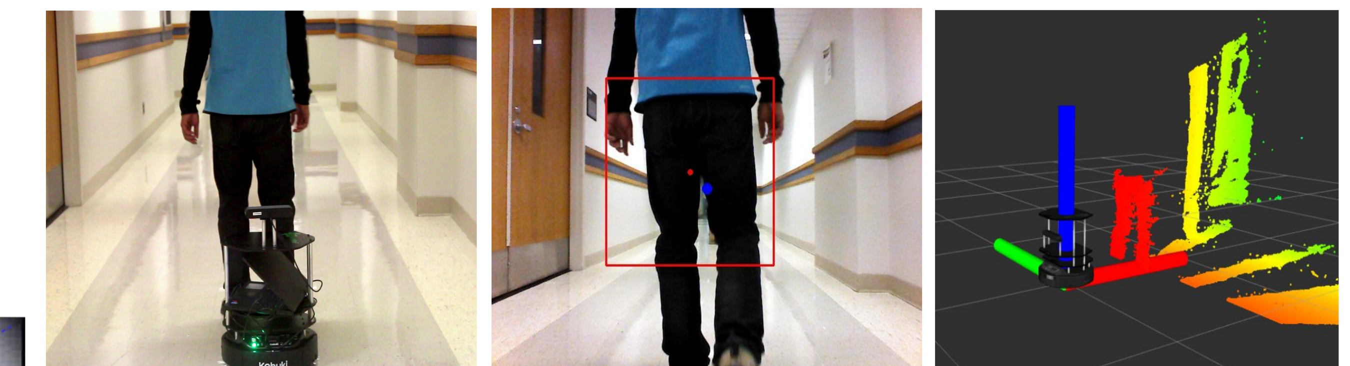
Yaw-Pitch controller for diver following by AUVs

```
xc, yc = (x+w/2), (y+h/2)
x0, y0 = im_w/2, im_h/2
offset_yaw = (xc-x0)/im_w
yaw_angle ∝ offset_yaw
offset_pitch = (yc-y0)/im_h
pitch_angle ∝ offset_pitch
velocity_forward ∝ distance
```



Person following process by UGVs (TurtleBots)

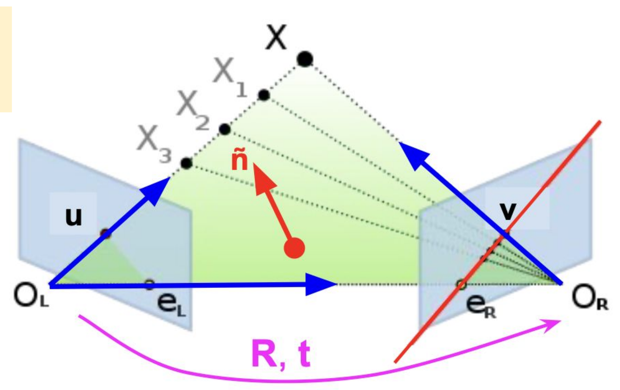
- With 3D bounding box (BBox):
  - Given a bounding box **(x, y, w, h)**
  - Get the center **(xc, yc)** of BBox
  - Calculate **offset_x** from center w.r.2 the image width
  - Calculate angular offset **theta**
  - Rotate with **angular.z**
  - Get the depth value **d[xc, yc]**
  - Move with **linear.x** (maintain a safe distance *d0*)



- With point clouds
  - Get the center of the point cloud
    - Closest point cloud usually works
  - Human leg detectors (by laser scanners)
  - Person detectors (by ML/DL methods)

Structure from motion (SfM)

- **Epipolar geometry:**
  - Camera centers: O_L, O_R
  - Baseline: line connecting the optical centers B=$O_L O_R$
  - 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 3D point
  - Epipolar lines: lines defined by the intersection of epipolar plane and image planes



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

$v^T F u = 0$

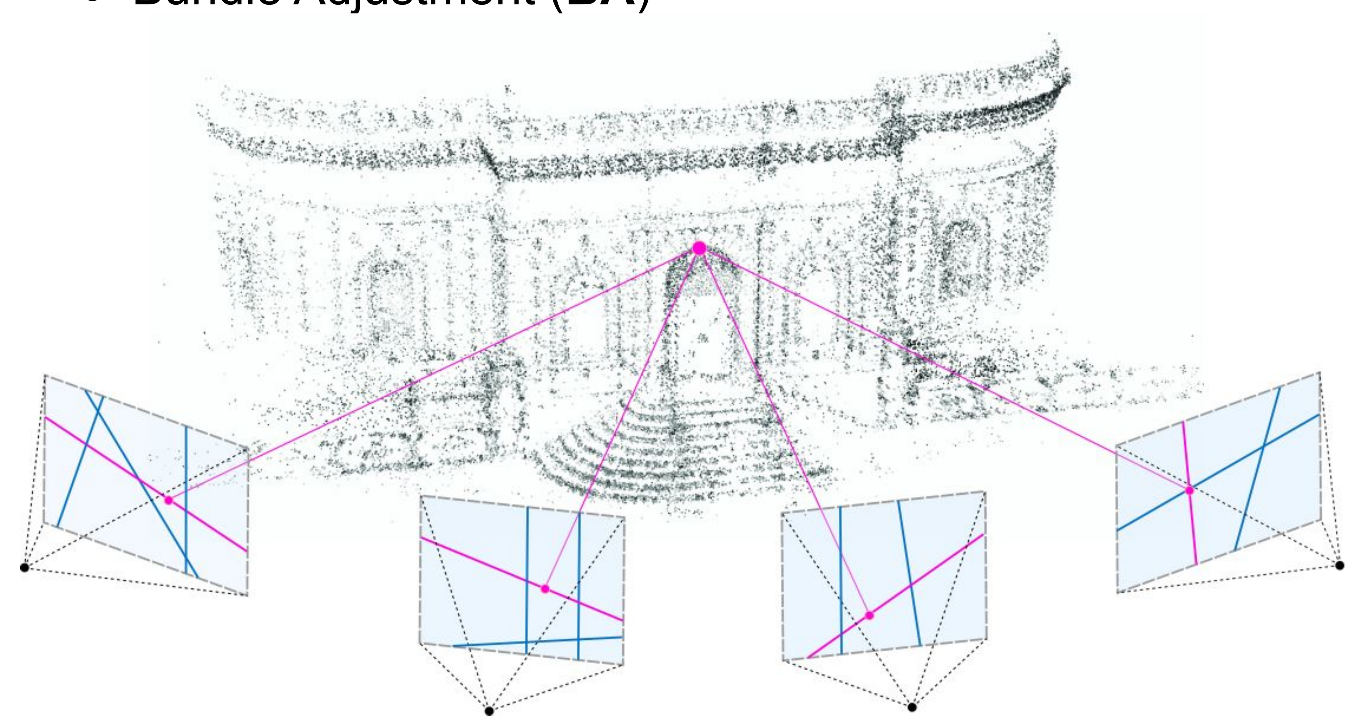
$F = K^{-T} E K^{-1} \;\Rightarrow\; E = K^T F K = \underline{t \times R}$

- **Fundamental matrix:**
  - $F = K_R^{-T} E K_L^{-1} \equiv K^{-T} E K^{-1}$
  - **u** in the left image represent a line: **Fu=0** in right image
    - It is the epipolar line **L = Fu**
  - The right epipole is also on this line
    - Therefore $e_R^T (Fu) = 0$
  - Similarly, **v** in the right image
    - Represent a line: $F^T v = 0$ in left image
    - Left epipole satisfies $e_L^T (F^T v) = 0$

- **Essentia matrix:**
  - $E = K^T F K = \underline{t \times R}$
  - How to get camera pose (R, t) from E
    - SVD: $E = UDV^T = [u_1 \;\; u_2 \;\; u_3] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \end{bmatrix}$
    - Then, $R \in \{UWV^T, UW^TV^T\}$; $t = \pm\lambda u_3$; $\lambda \in \mathbb{R}\backslash 0$
    - Where $W = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
    - Get four solutions: $[UWV^T \;\; u_3]$, $[UWV^T \;\; -u_3]$, $[UW^TV^T \;\; u_3]$, $[UW^TV^T \;\; -u_3]$

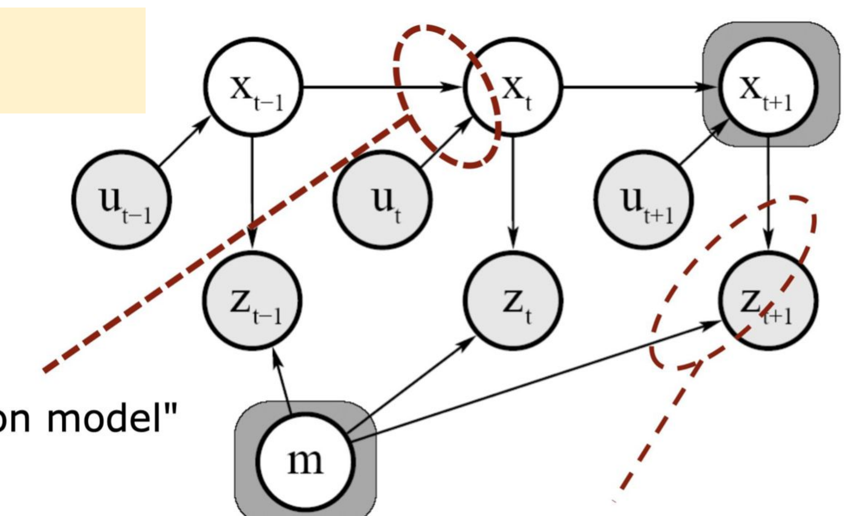
SfM pipeline: 3D structures from 2D image sequences

- 2D feature detection in images: SIFT, ORB, FAST, etc
- Feature matching across viewpoints
  - KNN + ratio test
- Estimating **F** from matched features: *(u, v)* pairs
  - 8-point algorithm + RANSAC
- Estimating **E** from **F**: **E** = K^T **F** K
- Finding **R**, **t** from **E**: triangulation + Cheriality condition
- Finding projection matrices: P_L, P_R
- Triangulating all 3D points
- **PnP** and nonlinear refinement
- Bundle Adjustment (**BA**)



Robot localization: SLAM

- **SLAM: Simultaneous Localization and M**
  - Given:
    - The robots controls $u_{1:T} = \{u_1, u_2, u_3 \ldots, u_T\}$
    - The measurements observations $z_{1:T} = \{z_1, z_2, z_3 \ldots, z_T\}$
  - Wanted:
    - The environment map m
    - The robot pose $x_{0:T} = \{x_0, x_1, x_2 \ldots, x_T\}$



- **Full SLAM vs Online SLAM**
  - Full SLAM estimates the entire state $p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$
  - Online SLAM estimates the current state $p(x_t, m \mid z_{1:t}, u_{1:t})$

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

distribution · path · map · given · observations · controls

Summary: EEL 4930/5934: Autonomous Robots

Spring 2023

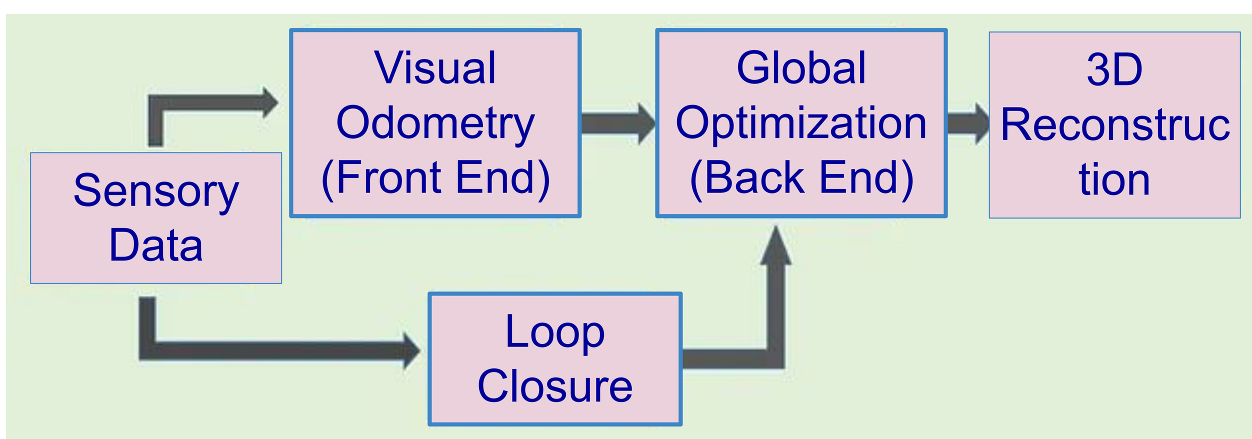
Department of ECE, University of Florida

RoboPI: Robot Perception & Intellgence Laboratory

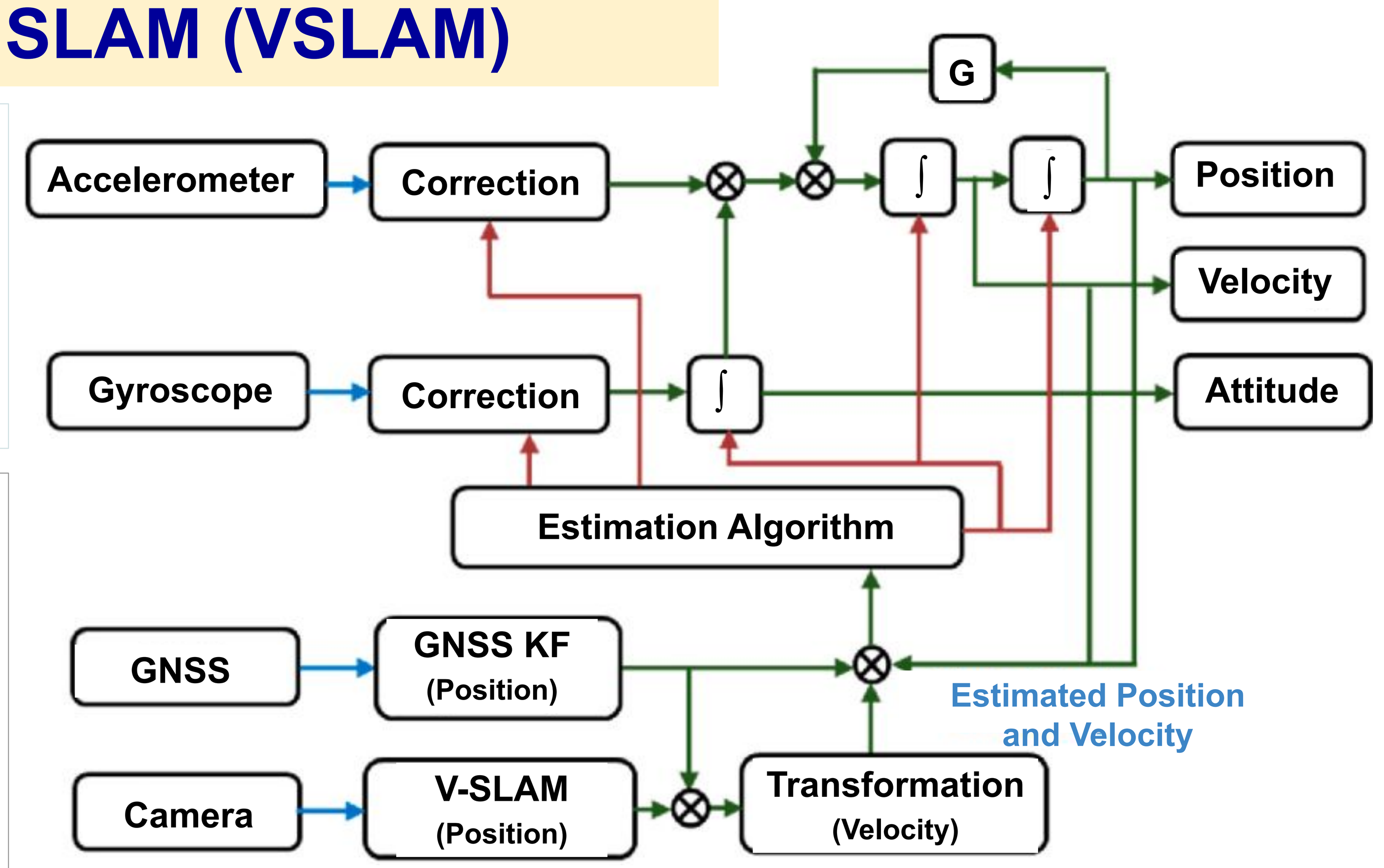
Visual Odometry (VO) and Visual SLAM (VSLAM)

VO: pose recovery from motion of a calibrated camera
VSLAM: VO + place recognition (loop closure) + global optimization for consistency
SfM: Recovers scene structure from unordered cameras at different viewpoints (often uncalibrated cameras)

⇒ VSLAM = VO + loop closure + global optimization



- VO provides only local/relative estimates, and the path is refined online with windowed optimization.
- VSLAM provides a global and consistent estimate
  - The detection of loop closure reduces the drift in both the map and the trajectory estimates
  - By performing **Bundle Adjustment (BA)**



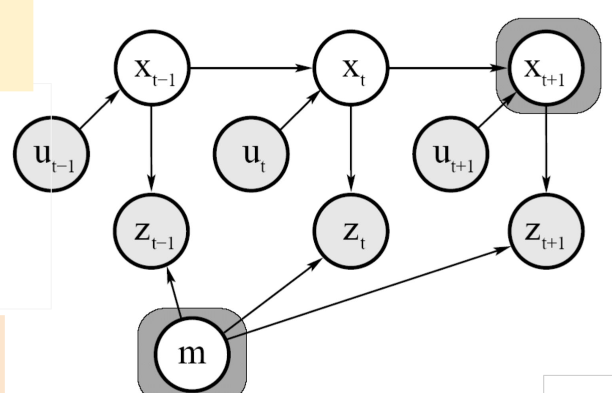
⇒ VIO = VO (pose estimates) + IMU (error correction)

- Uses VO to estimate camera pose from motion
- Inertial measurements from the IMU are used for error corrections associated with rapid motion
- Backbone of **VINS (Visual Inertial Navigation System)**
- Uses synchronized camera and IMU sensory fusion to estimate robot pose in real-time

⇒ Direct / indirect + Sparse / Dense / Semi-dense

State Estimation & Filtering

- x_t = State
- z_t = Measurement
- u_t = Control input
- m = map
- t = time
- $x = \arg\max P(x \mid z)$



KF

State transition: $x_t = \mathbf{F}_t x_{t-1} + \mathbf{B}_t u_t + \epsilon_t$
Observation: $z_t = \mathbf{H}_t x_t + \delta_t$

Predict State: $x_t = \mathbf{F}_t x_{t-1} + \mathbf{B}_t u_t$
Predict covariance: $\mathbf{P}_t = \mathbf{F}_t \mathbf{P}_{t-1} \mathbf{F}_t^{\mathsf{T}} + \mathbf{Q}_t$

Observation residual: $y_t = z_t - \mathbf{H}_t x_{t-1}$
Observation covariance: $\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t-1} \mathbf{H}_t^{\mathsf{T}} + \mathbf{R}_t$
Kalman gain: $\mathbf{K}_t = \mathbf{P}_t \mathbf{H}_{t-1}^{\mathsf{T}} \mathbf{S}_t^{-1}$
Update state: $x_t = x_t + \mathbf{K}_t y_t$
Update covariance: $\mathbf{P}_t = (\mathbf{I}_t - \mathbf{K}_t \mathbf{H}_t)\mathbf{P}_t$

EKF

State transition: $x_t = \mathbf{f}(x_t, u_t) + \epsilon_t$
Observation: $z_t = \mathbf{h}(x_t) + \delta_t$

Predict State: $x_t = \mathbf{f}(x_t, u_t)$
Predict covariance:
$$\mathbf{F}_t = \nabla_x \mathbf{f}(x_t, u_t)$$
$$\mathbf{G}_t = \nabla_\epsilon \mathbf{f}(x_t, u_t)$$
$$\mathbf{P}_t = \mathbf{F}_t \mathbf{P}_{t-1} \mathbf{F}_t^{\mathsf{T}} + \mathbf{Q}_t$$

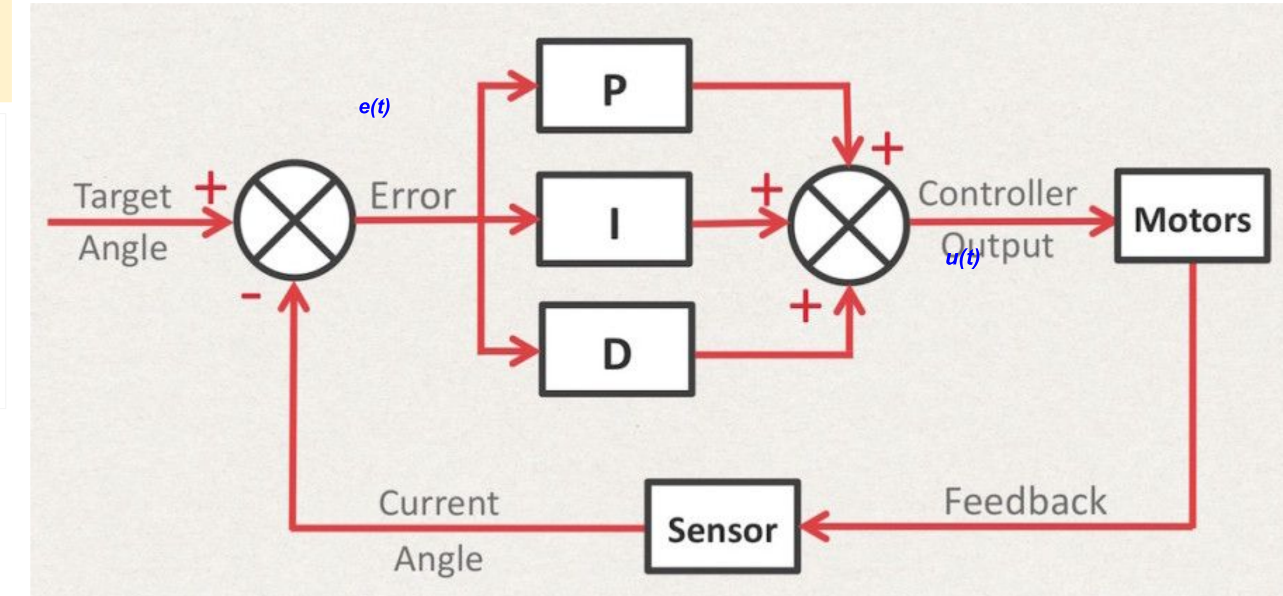
Observation residual: $y_t = z_t - \mathbf{h}(x_t)$
Observation covariance:
$$\mathbf{H}_t = \nabla_x \mathbf{h}(x_t)$$
$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t-1} \mathbf{H}_t^{\mathsf{T}} + \mathbf{R}_t$$
Kalman gain: $\mathbf{K}_t = \mathbf{P}_t \mathbf{H}_{t-1}^{\mathsf{T}} \mathbf{S}_t^{-1}$
Update state: $x_t = x_t + \mathbf{K}_t y_t$
Update covariance: $\mathbf{P}_t = (\mathbf{I}_t - \mathbf{K}_t \mathbf{H}_t)\mathbf{P}_t$

Feedback Control: PID

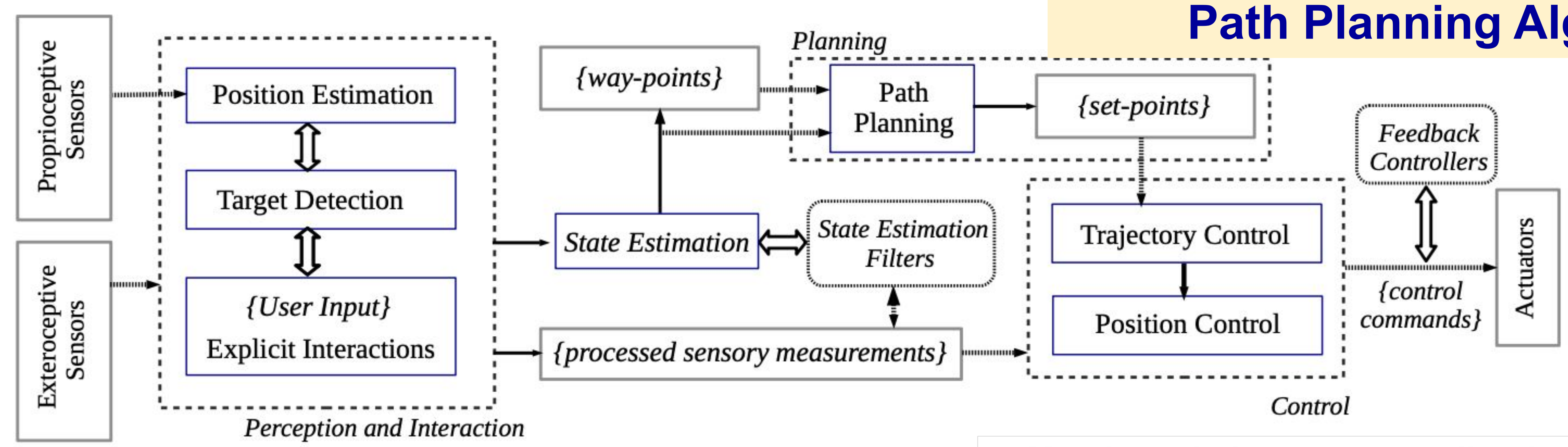
- **Proportional (P)**: compensates for the error difference
- **Derivative (D)**: reacts for the change of error (restricts oscillation)
- **Integral (I)**: responds to the steady-state response

Need to tune K_p, K_i, K_d experimentally

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t)$$



Path Planning Algorithms



⇒ Map-based planners: trea search

- **BFS:** Search a tree, one level at a time
  - Complete (finds solution if there is one)
  - Optimal if cost is increasing with path depth
- **DFS:** Search a tree, keep expanding one child at a time
  - Not complete if infinite depth; Not optimal
- **Dijkstra:** searches the single-source shortest path
  - Optimal and complete, but not always fast
    - Start node is assigned a distance of zero
    - Other node's distance are set to infinity
    - Compute **g(n)**: path cost from the start node to **n**
- **A*:** Uses heuristics to find the "best" node to expand
  - Optimal and complete
    - **g(n)**: path cost from the start node to **n**
    - **h(n)**: cost of the cheapest path from **n** to the goal node
    - Evaluate **n** for expansion based on: **f(n) = g(n) + h(n)**

⇒ Map-based planners: sampling-based algorithms

- **PRM:** Probabilistic road map
  - Learning phase
    - Sample *n* points in configuration space *Cfree*
    - Connect random configurations using a local planner
  - Query phase
    - Connect start and goal configurations with the PRM
    - Use the graph search to find the path
  - Probabilistic completeness
  - Efficient if we need multiple queries on the same graph
- **RRT and RRT*:** Rapidly-exploring Random Trees
  - For each planning problem constructs a new roadmap
  - Aggressively probe and explore the configuration space by expanding incrementally
  - Probabilistic completeness
  - More efficient than PRM if only a single query needed

⇒ Advanced algorithms

- Planning without a map
- Target-centric planners
- Active planners
- Imitation learning
- Learning to plan from demonstrations (LfDs)