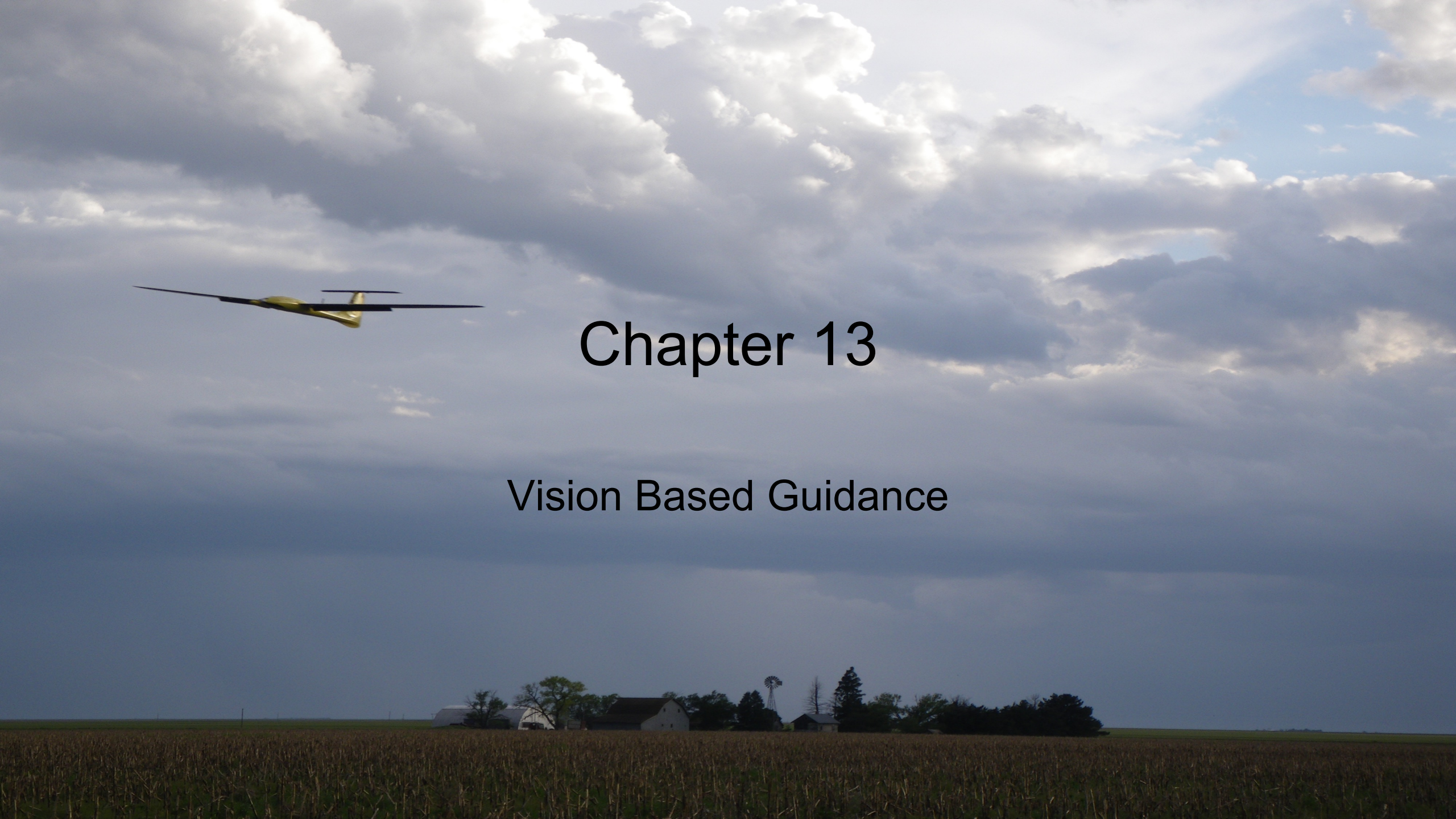
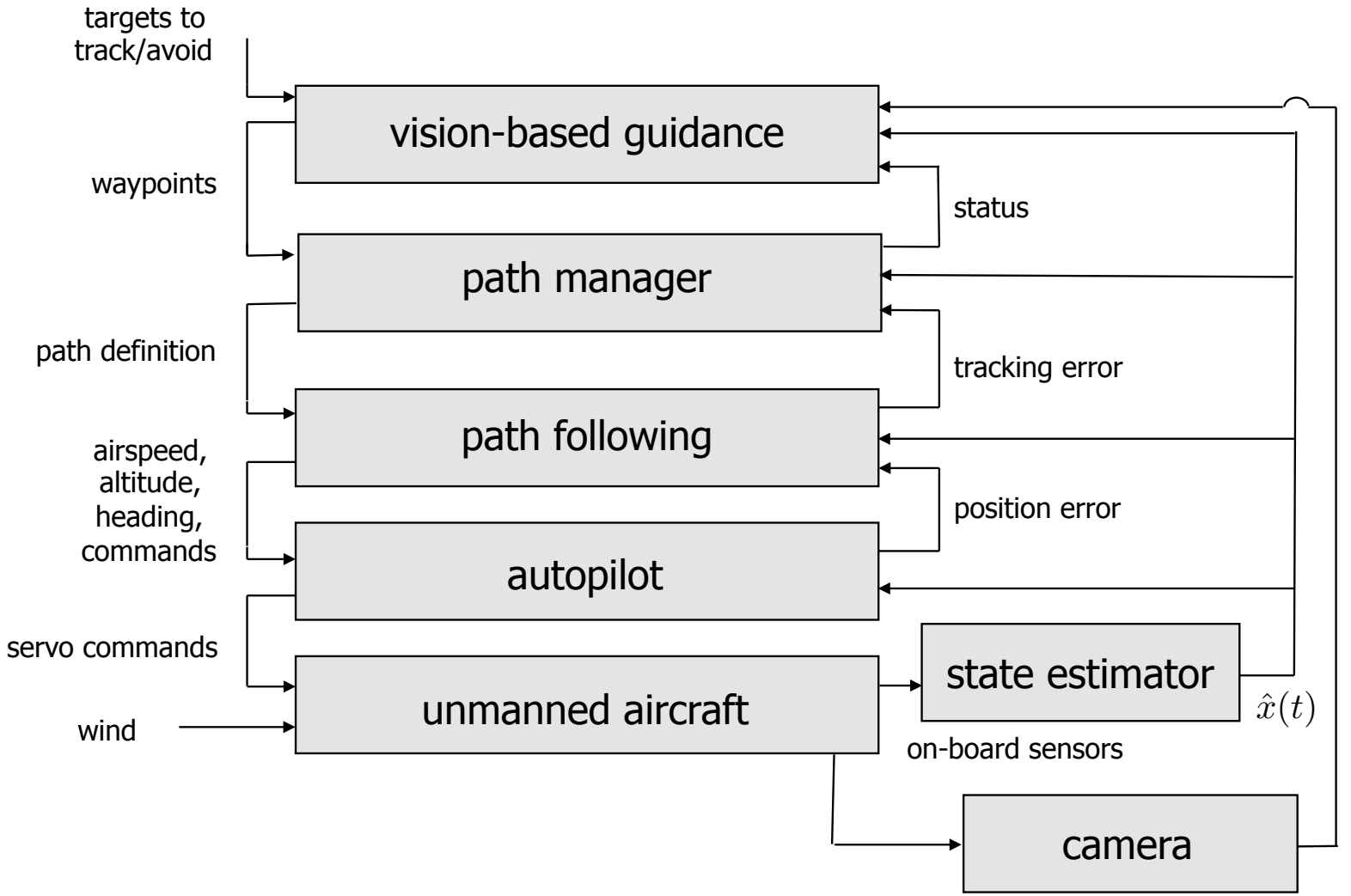


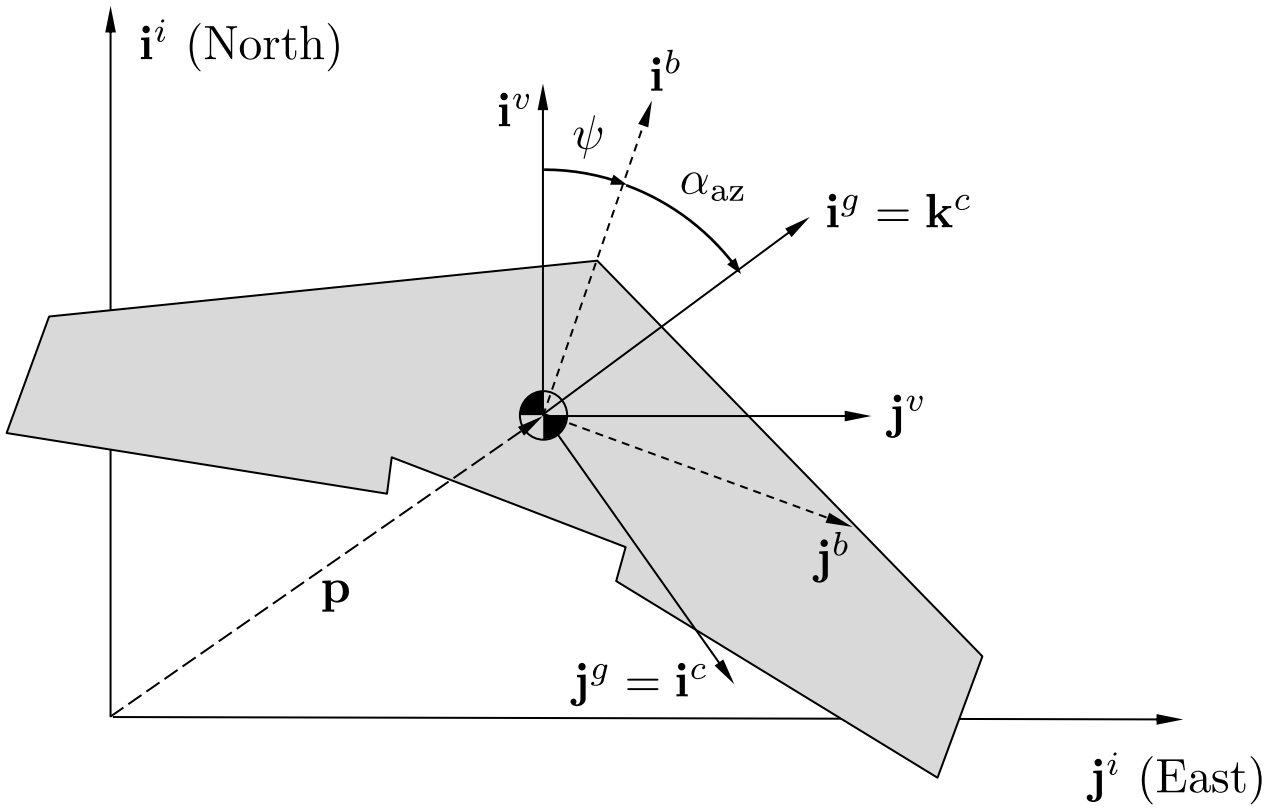
Chapter 13

Vision Based Guidance

Architecture w/ Camera



Gimbal and Camera Reference Frames



Gimbal-1 frame: $\mathcal{F}^{g1} = (\mathbf{i}^{g1}, \mathbf{j}^{g1}, \mathbf{k}^{g1})$
Gimbal frame: $\mathcal{F}^{g} = (\mathbf{i}^{g}, \mathbf{j}^{g}, \mathbf{k}^{g})$
Camera frame: $\mathcal{F}^{c} = (\mathbf{i}^{c}, \mathbf{j}^{c}, \mathbf{k}^{c})$

Gimbal-1 frame:
Rotate body frame about $\mathbf{k}^b$ axis
by gimbal azimuth angle, α_{az}

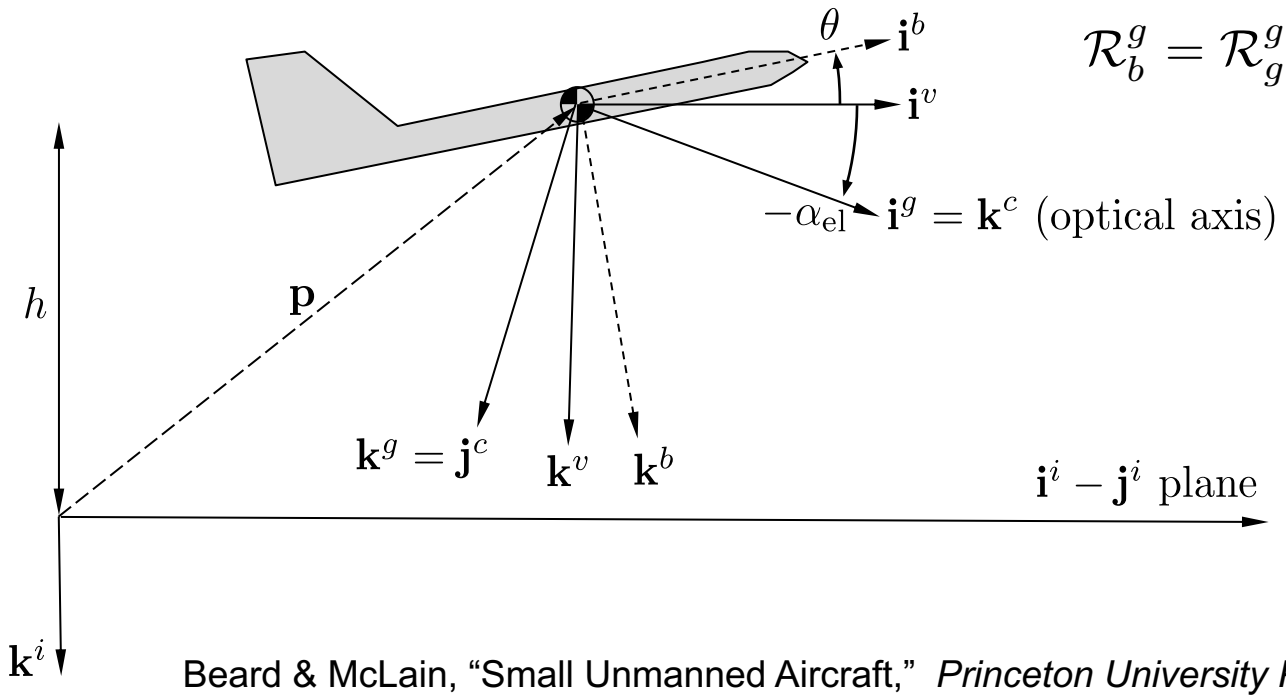
$$\mathcal{R}_b^{g1}(\alpha_{\text{az}}) \triangleq \begin{pmatrix} \cos\alpha_{\text{az}} & \sin\alpha_{\text{az}} & 0 \\ -\sin\alpha_{\text{az}} & \cos\alpha_{\text{az}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Gimbal Reference Frames

Gimbal frame:
Rotate gimbal-1 frame about $\mathbf{j}^{g1}$ axis
by gimbal elevation angle, α_{el}

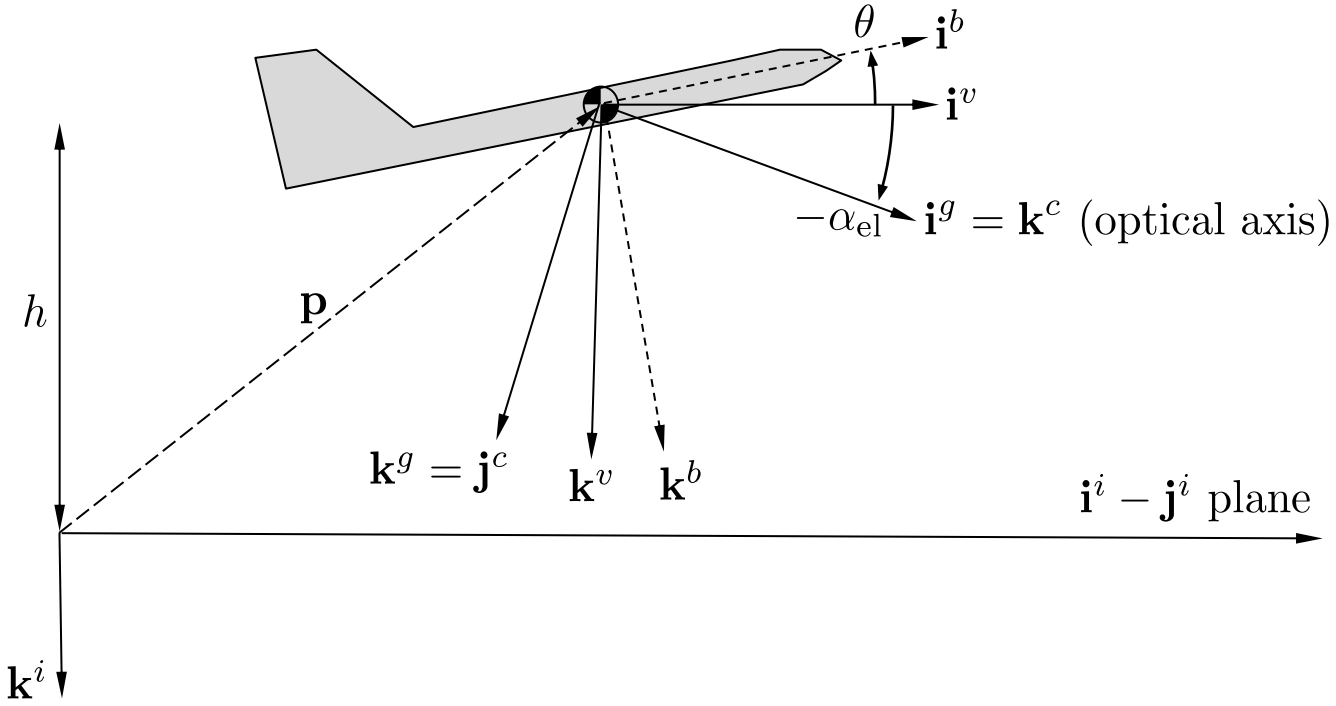
$$\mathcal{R}^g_{g1}(\alpha_{\text{el}}) \triangleq \begin{pmatrix} \cos\alpha_{\text{el}} & 0 & -\sin\alpha_{\text{el}} \\ 0 & 1 & 0 \\ \sin\alpha_{\text{el}} & 0 & \cos\alpha_{\text{el}} \end{pmatrix}$$

$$\mathcal{R}^g_b = \mathcal{R}^g_{g1}\mathcal{R}^{g1}_b = \begin{pmatrix} \cos\alpha_{\text{el}}\cos\alpha_{\text{az}} & \cos\alpha_{\text{el}}\sin\alpha_{\text{az}} & -\sin\alpha_{\text{el}} \\ -\sin\alpha_{\text{az}} & \cos\alpha_{\text{az}} & 0 \\ \sin\alpha_{\text{el}}\cos\alpha_{\text{az}} & \sin\alpha_{\text{el}}\sin\alpha_{\text{az}} & \cos\alpha_{\text{el}} \end{pmatrix}$$



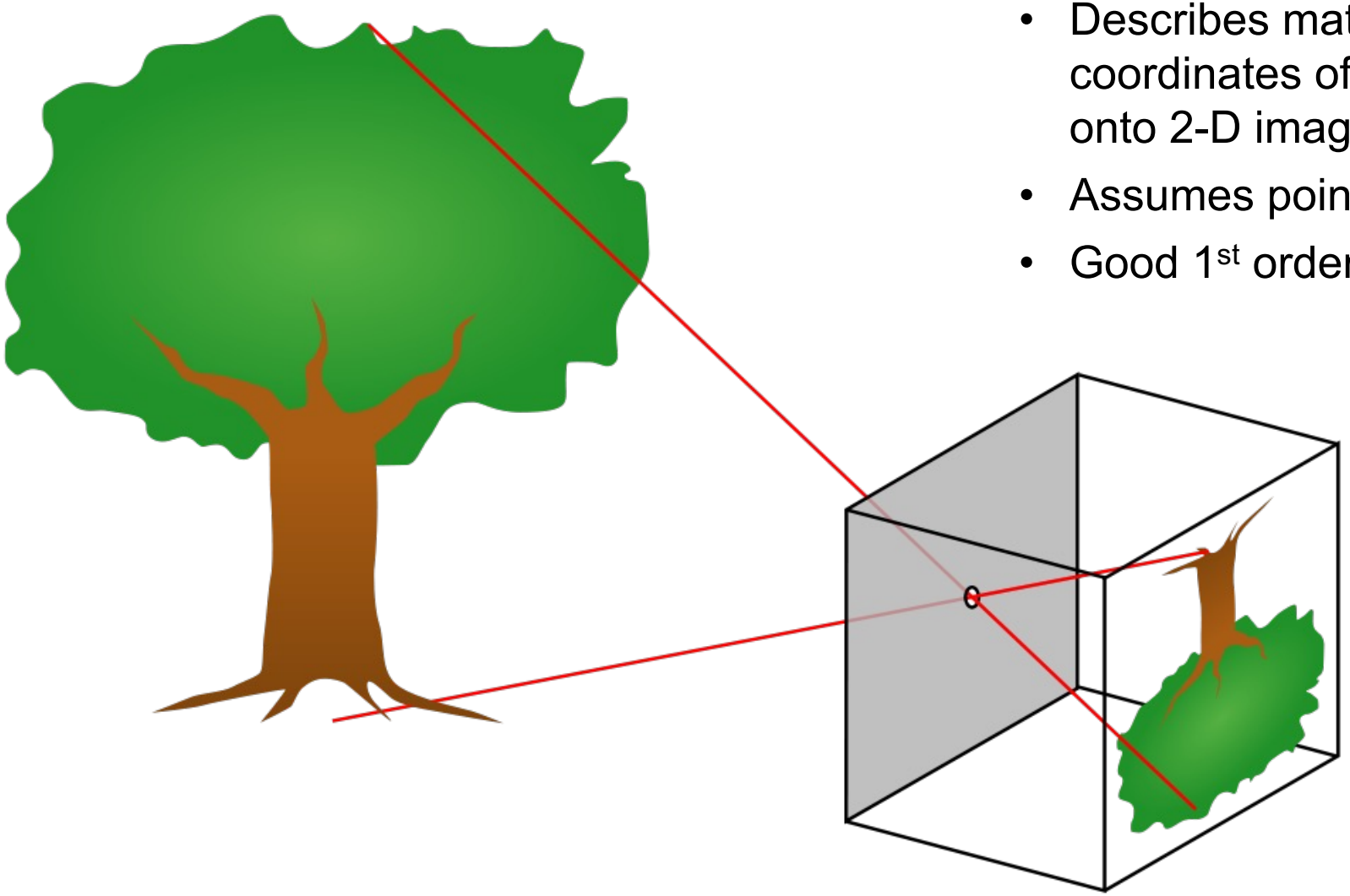
Camera Reference Frame

Camera frame:
$\mathbf{i}^c$ points to the right in the image
$\mathbf{j}^c$ points down in the image
$\mathbf{k}^c$ points along the optical axis



$$\mathcal{R}_g^c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Pinhole Camera Model



- Describes mathematical relationship between coordinates of 3-D point and its projection onto 2-D image plane
- Assumes point aperture, no lenses
- Good 1st order approximation

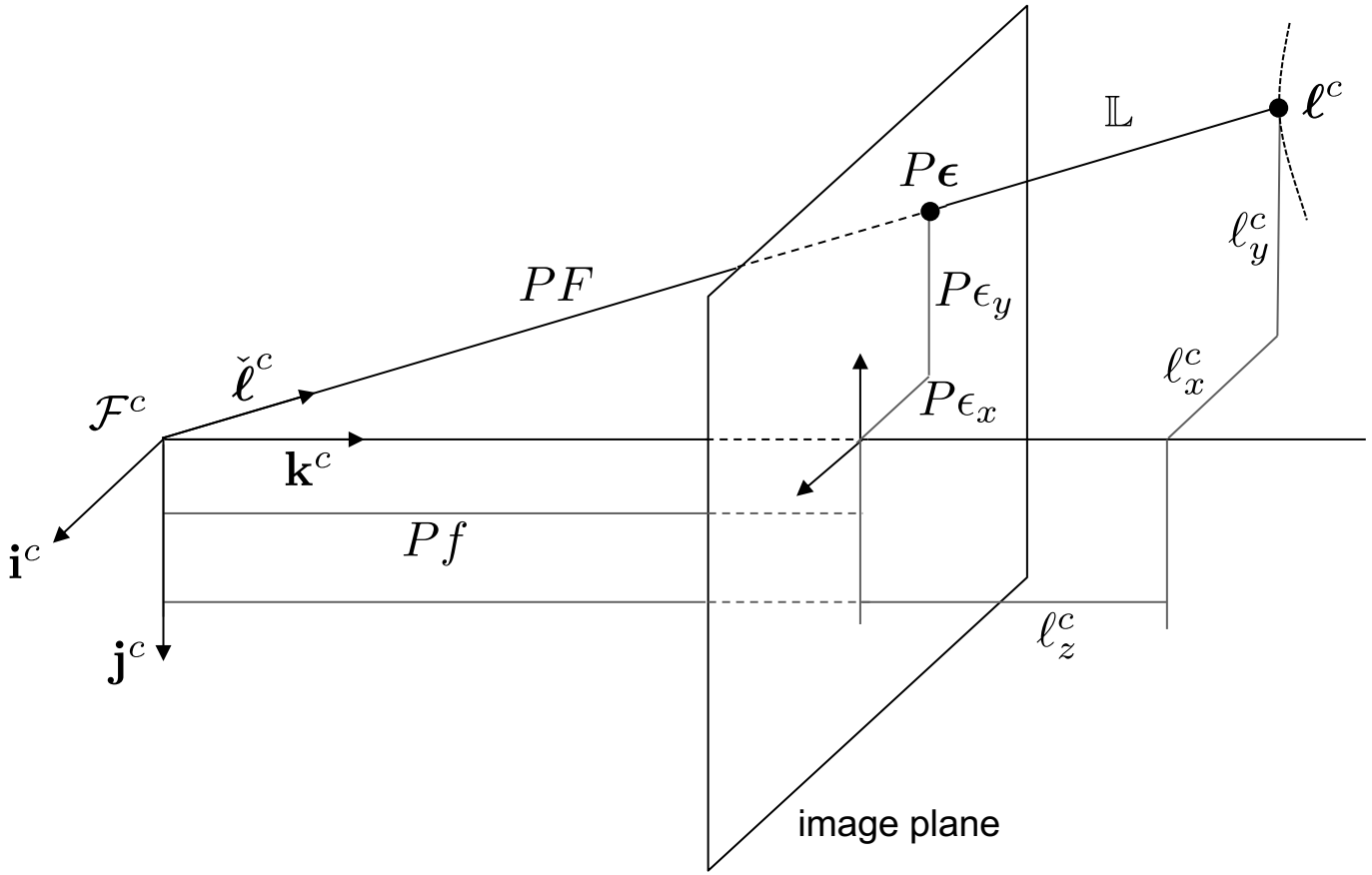
Pinhole Camera Model

Goal:

From pixel location (ϵ_x, ϵ_y) in image plane and camera parameters, determine location of object in world $\boldsymbol{\ell}^c$

Approach:

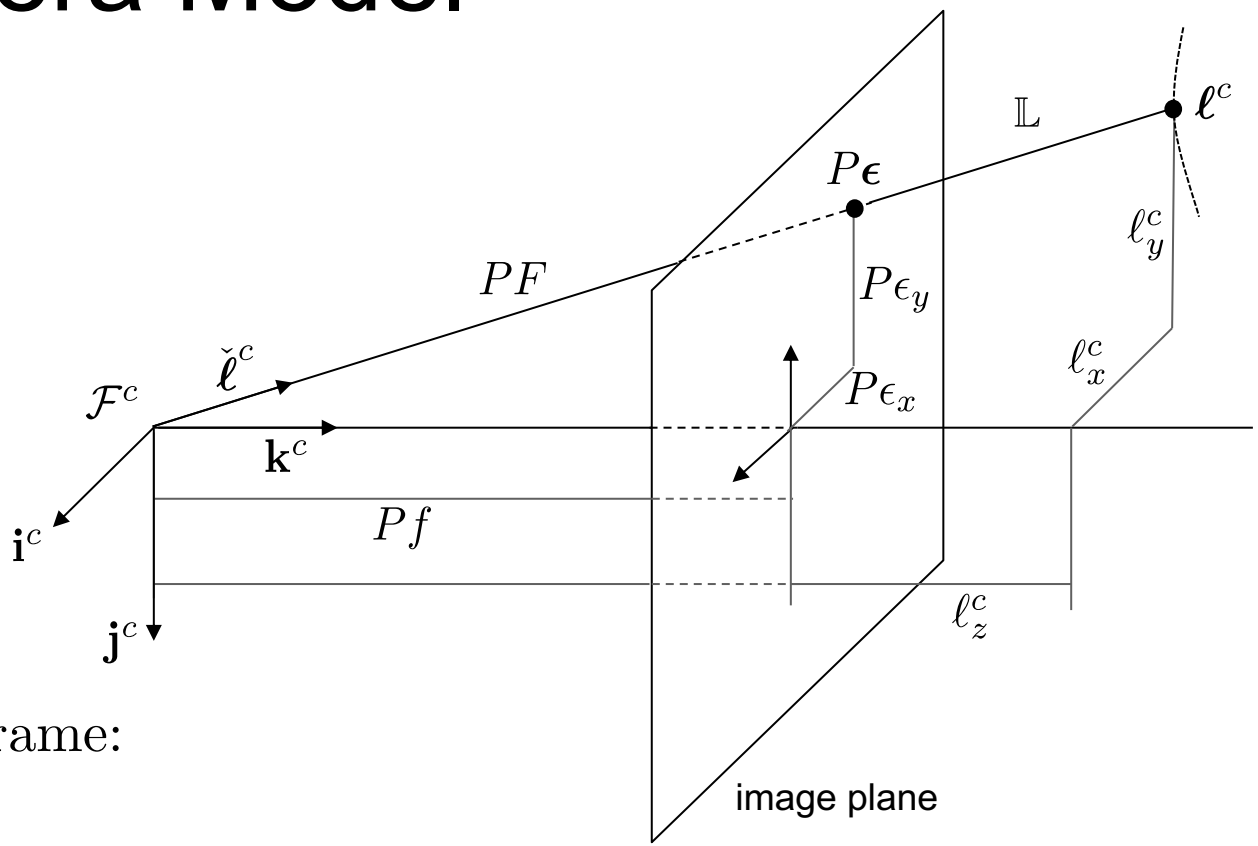
Use similar triangles geometry



Camera Model

f: focal length in pixels
P: converts pixels to meters
M: width of square pixel array
v: field of view
$\boldsymbol{\ell}^c$: 3-D location of point

$$f = \frac{M}{2\tan\left(\frac{v}{2}\right)}$$



Location of projection of object in camera frame:
$(P\epsilon_x, P\epsilon_y, Pf)$
Pixel location (in pixels): ϵ_x, ϵ_y

Distance from origin of camera frame to object image:

$$F = \sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}$$

Camera Model

By similar triangles:

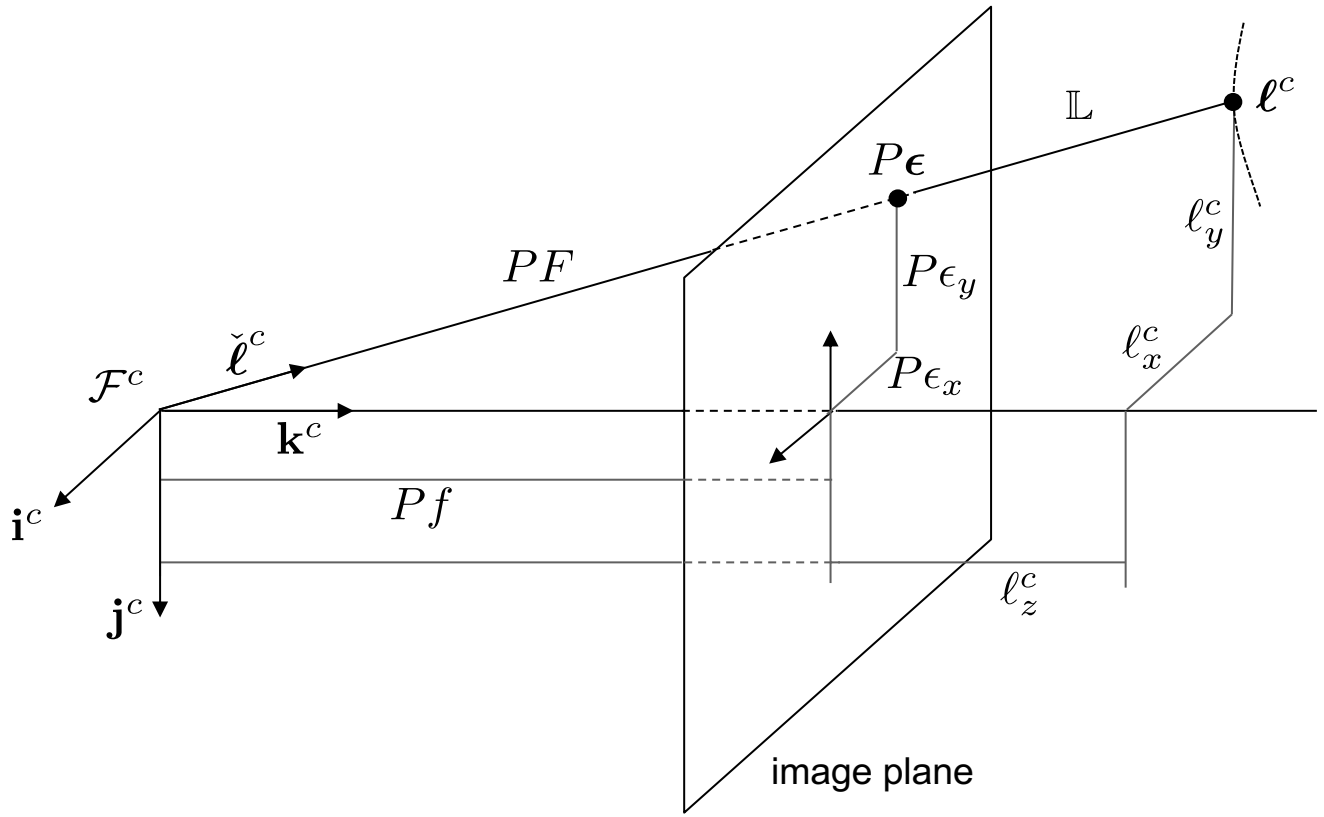
$$\frac{\ell_x^c}{\mathbb{L}} = \frac{P\epsilon_x}{PF} = \frac{\epsilon_x}{F}$$

Similarly:

$$\ell_y^c/\mathbb{L} = \epsilon_y/F \quad \text{and} \quad \ell_z^c/\mathbb{L} = f/F$$

Combining:

$$\boldsymbol{\ell}^c = \begin{pmatrix} \ell_x^c \\ \ell_y^c \\ \ell_z^c \end{pmatrix} = \frac{\mathbb{L}}{F} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}$$



Problem: $\mathbb{L}$ is unknown...

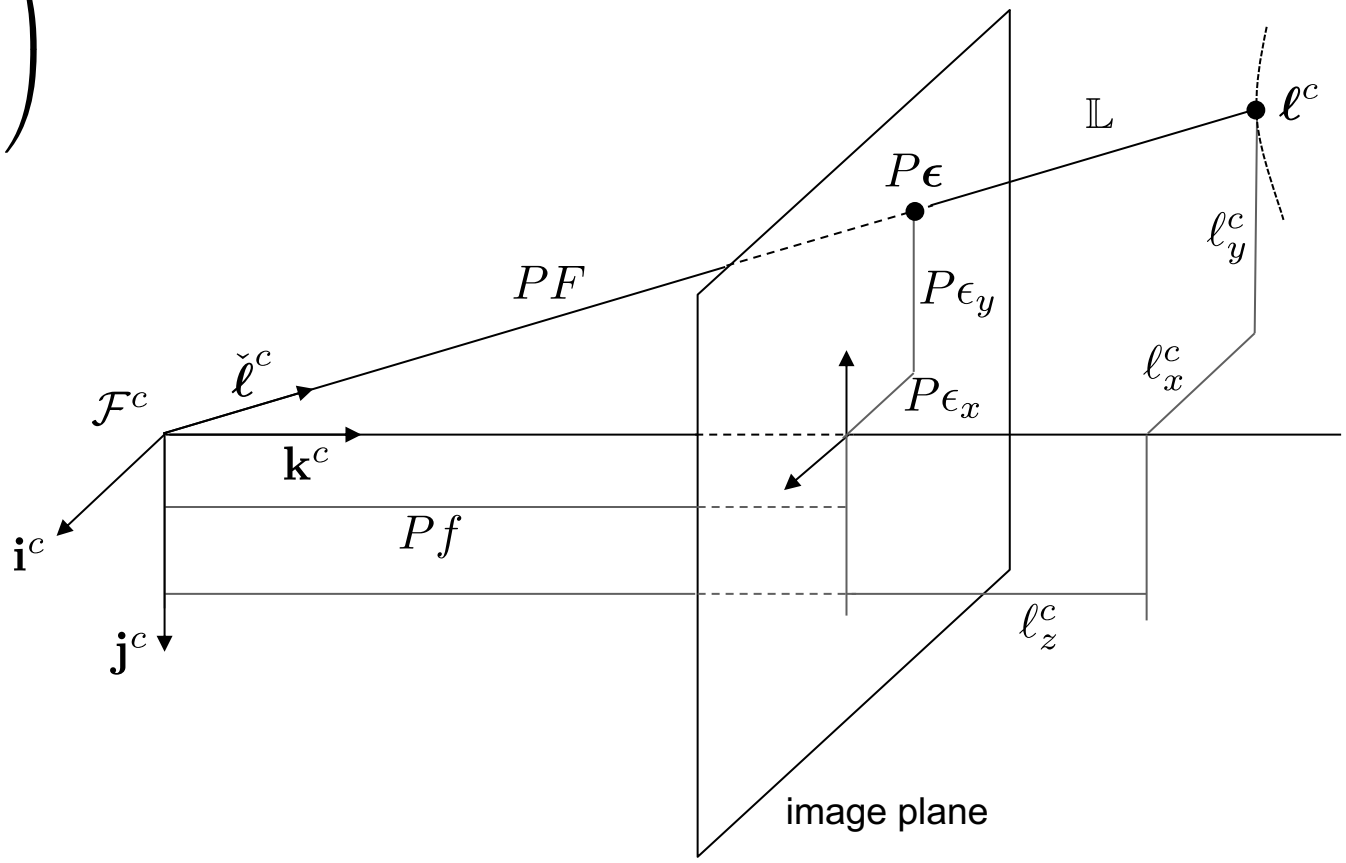
Camera Model

Unit direction vector to target:

$$\frac{\boldsymbol{\ell}^c}{\mathbb{L}} = \frac{1}{F}\begin{pmatrix}\epsilon_x \\ \epsilon_y \\ f\end{pmatrix} = \frac{1}{\sqrt{\epsilon_x^2+\epsilon_y^2+f^2}}\begin{pmatrix}\epsilon_x \\ \epsilon_y \\ f\end{pmatrix}$$

Define notation:

$$\check{\boldsymbol{\ell}} \triangleq \begin{pmatrix}\check{\ell}_x \\ \check{\ell}_y \\ \check{\ell}_z\end{pmatrix} \triangleq \frac{\boldsymbol{\ell}}{\mathbb{L}}$$



Gimbal Pointing

- Two scenarios
  - Point gimbal at given world coordinate
    - "Point to this GPS location"
  - Point gimbal so that optical axis aligns with certain point in image plane
    - "Point at this object"
- Gimbal dynamics
  - Assume rate control inputs

$$\dot{\alpha}_{\text{az}} = u_{\text{az}}$$

$$\dot{\alpha}_{\text{el}} = u_{\text{el}}$$

Scenario 1: Point Gimbal at World Coordinate

$\mathbf{p}^i_{\text{obj}}$: known location of object in inertial frame (world coordinate)
$\mathbf{p}^i_{\text{MAV}} = (p_n, p_e, p_d)^\top$: location of MAV in inertial frame

Objective:
Align optical axis of camera with desired relative position vector

$$\boldsymbol{\ell}^i_d \triangleq \mathbf{p}^i_{\text{obj}} - \mathbf{p}^i_{\text{MAV}}$$

Body-frame unit vector that points in desired direction of specified world coordinates

$$\check{\boldsymbol{\ell}}^b_d = \frac{1}{\|\boldsymbol{\ell}^i_d\|} \mathcal{R}^b_i \, \boldsymbol{\ell}^i_d$$

Scenario 2: Point Gimbal at Object in Image

Objective:
Maneuver gimbal so that object at pixel location ϵ is pushed to center of image

Desired direction of optical axis:

$$\check{\boldsymbol{\ell}}_d^c = \frac{1}{\sqrt{f^2 + \epsilon_x^2 + \epsilon_y^2}} \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}$$

Desired direction of optical axis expressed in body frame:

$$\check{\boldsymbol{\ell}}_d^b = \mathcal{R}_g^b \mathcal{R}_c^g \, \check{\boldsymbol{\ell}}_d^c$$

Gimbal Pointing Angles

We know direction to point gimbal

What are corresponding azimuth and elevation angles?

Optical axis in camera frame $= (0,0,1)^\top$

Objective: Select commanded gimbal angles $\alpha^c_{\rm az}$ and $\alpha^c_{\rm el}$ so that

$$\check{\boldsymbol{\ell}}^b_d \triangleq \begin{pmatrix} \check{\ell}^b_{xd} \\ \check{\ell}^b_{yd} \\ \check{\ell}^b_{zd} \end{pmatrix} = \mathcal{R}^b_g(\alpha^c_{\rm az}, \alpha^c_{\rm el})\mathcal{R}^g_c \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\alpha^c_{\rm el}\cos\alpha^c_{\rm az} & -\sin\alpha^c_{\rm az} & \sin\alpha^c_{\rm el}\cos\alpha^c_{\rm az} \\ \cos\alpha^c_{\rm el}\sin\alpha^c_{\rm az} & \cos\alpha^c_{\rm az} & \sin\alpha^c_{\rm el}\sin\alpha^c_{\rm az} \\ -\sin\alpha^c_{\rm el} & 0 & \cos\alpha^c_{\rm el} \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha^c_{\rm el}\cos\alpha^c_{\rm az} \\ \cos\alpha^c_{\rm el}\sin\alpha^c_{\rm az} \\ -\sin\alpha^c_{\rm el} \end{pmatrix}$$

Gimbal Pointing Angles

Objective: Select commanded gimbal angles gimbal angles $\alpha_{\rm az}^c$ and $\alpha_{\rm el}^c$ so that

$$\begin{pmatrix} \check{\ell}_{xd}^b \\ \check{\ell}_{yd}^b \\ \check{\ell}_{zd}^b \end{pmatrix} = \begin{pmatrix} \cos\alpha_{\rm el}^c \cos\alpha_{\rm az}^c \\ \cos\alpha_{\rm el}^c \sin\alpha_{\rm az}^c \\ -\sin\alpha_{\rm el}^c \end{pmatrix}$$

Solving for $\alpha_{\rm el}^c$ and $\alpha_{\rm az}^c$ gives desired azimuth and elevation angles:

$$\alpha_{\rm az}^c = \tan^{-1}\left(\frac{\check{\ell}_{yd}^b}{\check{\ell}_{xd}^b}\right)$$
$$\alpha_{\rm el}^c = -\sin^{-1}\left(\check{\ell}_{zd}^b\right)$$

Gimbal servo commands can be selected as:

$$u_{\rm az} = k_{\rm az}(\alpha_{\rm az}^c - \alpha_{\rm az})$$
$$u_{\rm el} = k_{\rm el}(\alpha_{\rm el}^c - \alpha_{\rm el})$$

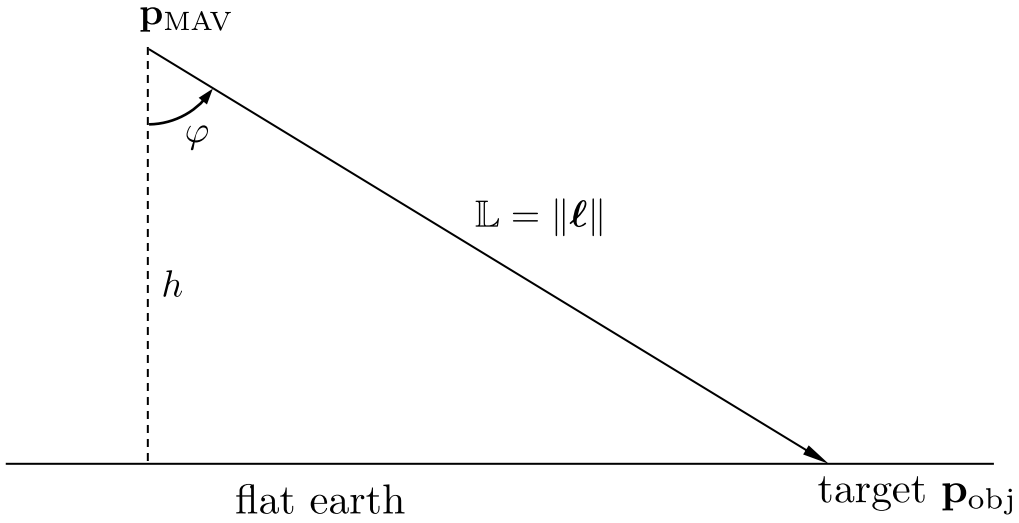
$k_{\rm az}$ and $k_{\rm el}$ are positive control gains

Geolocation

Relative position vector between target and MAV: $\boldsymbol{\ell} = \mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}}$

Define:

$$\mathbb{L} = \|\boldsymbol{\ell}\| \text{ (range to target)}$$
$$\check{\boldsymbol{\ell}} = \boldsymbol{\ell}/\mathbb{L} \text{ (unit vector pointing from aircraft to target)}$$



From geometry:

$$\begin{aligned}\mathbf{p}^i_{\text{obj}} &= \mathbf{p}^i_{\text{MAV}} + \mathcal{R}^i_b \mathcal{R}^b_g \mathcal{R}^g_c \boldsymbol{\ell}^c \\ &= \mathbf{p}^i_{\text{MAV}} + \mathbb{L}\left(\mathcal{R}^i_b \mathcal{R}^b_g \mathcal{R}^g_c \check{\boldsymbol{\ell}}^c\right)\end{aligned}$$

where

$$\begin{aligned}\mathbf{p}^i_{\text{MAV}} &= (p_n, p_e, p_d)^\top \\ \mathcal{R}^i_b &= \mathcal{R}^i_b(\phi, \theta, \psi) \\ \mathcal{R}^b_g &= \mathcal{R}^b_g(\alpha_{\text{az}}, \alpha_{\text{el}})\end{aligned}$$

$\mathbb{L}$ is unknown $\rightarrow$ solving geolocation problem reduces to estimating range to target $\mathbb{L}$

Range to Target – Flat Earth Model

From geometry: $\cos\varphi = \dfrac{h}{\mathbb{L}}$

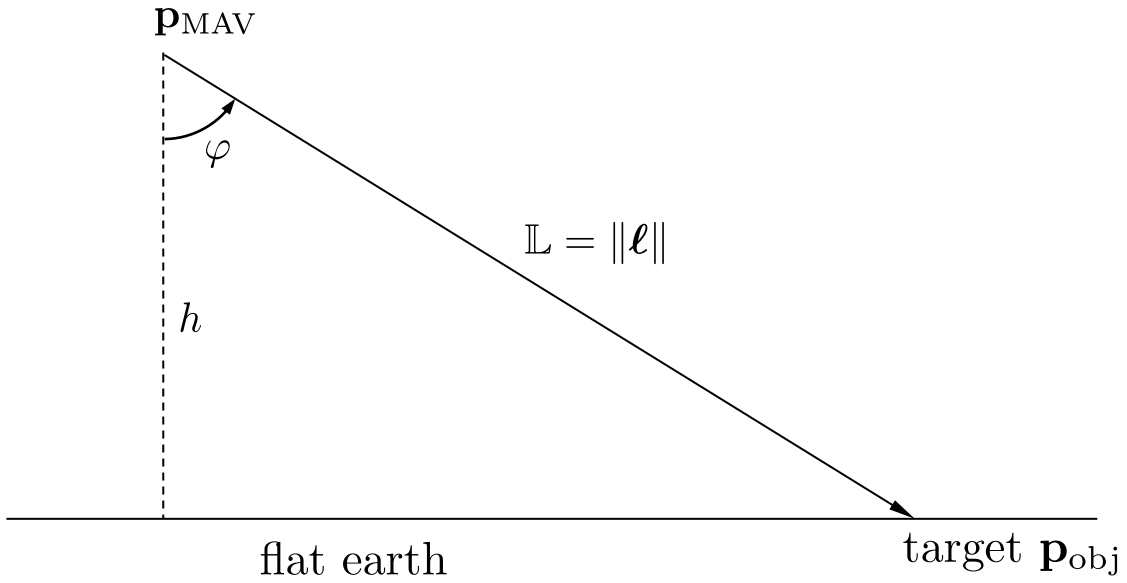
From Cauchy-Schwartz equality: $\cos\varphi = \mathbf{k}^i \cdot \check{\boldsymbol{\ell}}^i = \mathbf{k}^i \cdot \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\boldsymbol{\ell}}^c$

Equating: $\mathbb{L} = \dfrac{h}{\mathbf{k}^i \cdot \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\boldsymbol{\ell}}^c}$

From before: $\mathbf{p}_{\text{obj}}^i = \mathbf{p}_{\text{MAV}}^i + \mathbb{L}\left(\mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\boldsymbol{\ell}}^c\right)$

Therefore:

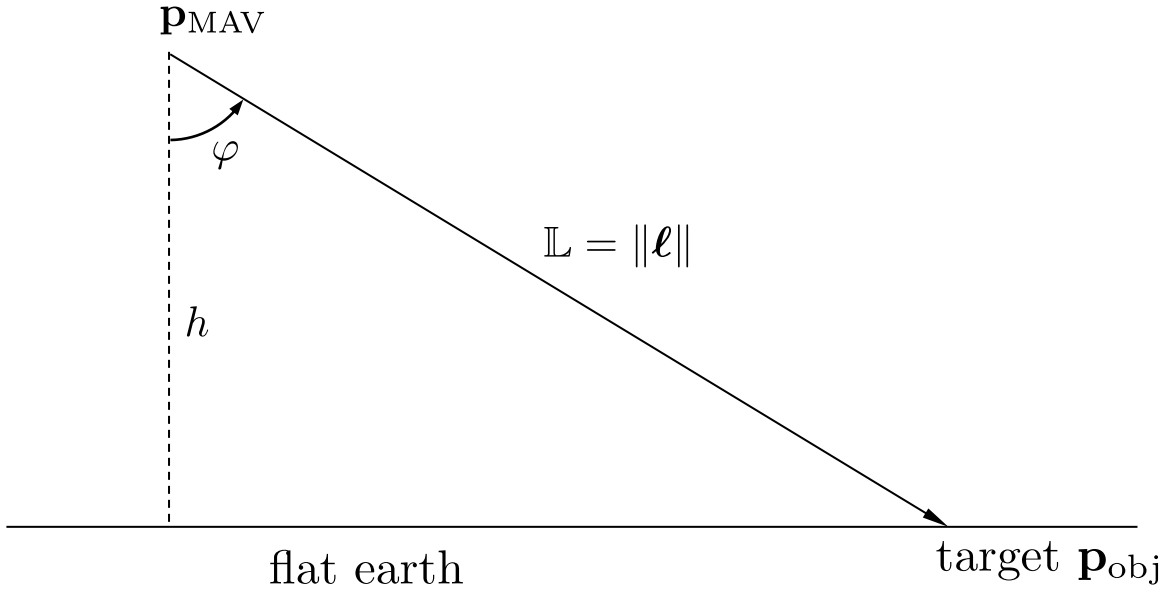
$$\boxed{\mathbf{p}_{\text{obj}}^i = \begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} + h \frac{\mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\boldsymbol{\ell}}^c}{\mathbf{k}^i \cdot \mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\boldsymbol{\ell}}^c}}$$



Geolocation Errors

The position of the object is

$$\mathbf{p}^i_{\text{obj}} = \begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} + h \frac{\mathcal{R}^i_b \mathcal{R}^b_g \mathcal{R}^g_c \check{\boldsymbol{\ell}}^c}{\mathbf{k}^i \cdot \mathcal{R}^i_b \mathcal{R}^b_g \mathcal{R}^g_c \check{\boldsymbol{\ell}}^c}$$



Equation is highly sensitive to measurement errors

- attitude estimation errors
- gimbal pointing angle errors
- gimbal/autopilot alignment errors

Two types of errors

- bias errors (address with calibration)
- random errors (address with EKF)

Geolocation Using EKF

The states are $x = (\mathbf{p}_{\text{obj}}^{i^\top}, \mathbb{L})^\top$.

Need propagation equations $\dot{x} = f(x, u)$.

Assuming the object is stationary: $\dot{\mathbf{p}}_{\text{obj}}^{i} = 0$

Also:

$$\dot{\mathbb{L}} = \frac{d}{dt}\sqrt{(\mathbf{p}_{\text{obj}}^{i} - \mathbf{p}_{\text{MAV}}^{i})^\top(\mathbf{p}_{\text{obj}}^{i} - \mathbf{p}_{\text{MAV}}^{i})} = \frac{(\mathbf{p}_{\text{obj}}^{i} - \mathbf{p}_{\text{MAV}}^{i})^\top(\dot{\mathbf{p}}_{\text{obj}}^{i} - \dot{\mathbf{p}}_{\text{MAV}}^{i})}{\mathbb{L}} = -\frac{(\mathbf{p}_{\text{obj}}^{i} - \mathbf{p}_{\text{MAV}}^{i})^\top \dot{\mathbf{p}}_{\text{MAV}}^{i}}{\mathbb{L}}$$

where: $\mathbf{p}_{\text{MAV}}^{i} = \mathbf{p}_{\text{obj}}^{i} - \mathbb{L}\left(\mathcal{R}_b^i \mathcal{R}_g^b \mathcal{R}_c^g \check{\boldsymbol{\ell}}^c\right)$

and for constant-altitude flight: $\dot{\mathbf{p}}_{\text{MAV}}^{i} = \begin{pmatrix} \hat{V}_g \cos\hat{\chi} \\ \hat{V}_g \sin\hat{\chi} \\ 0 \end{pmatrix}$

Geolocation Using EKF

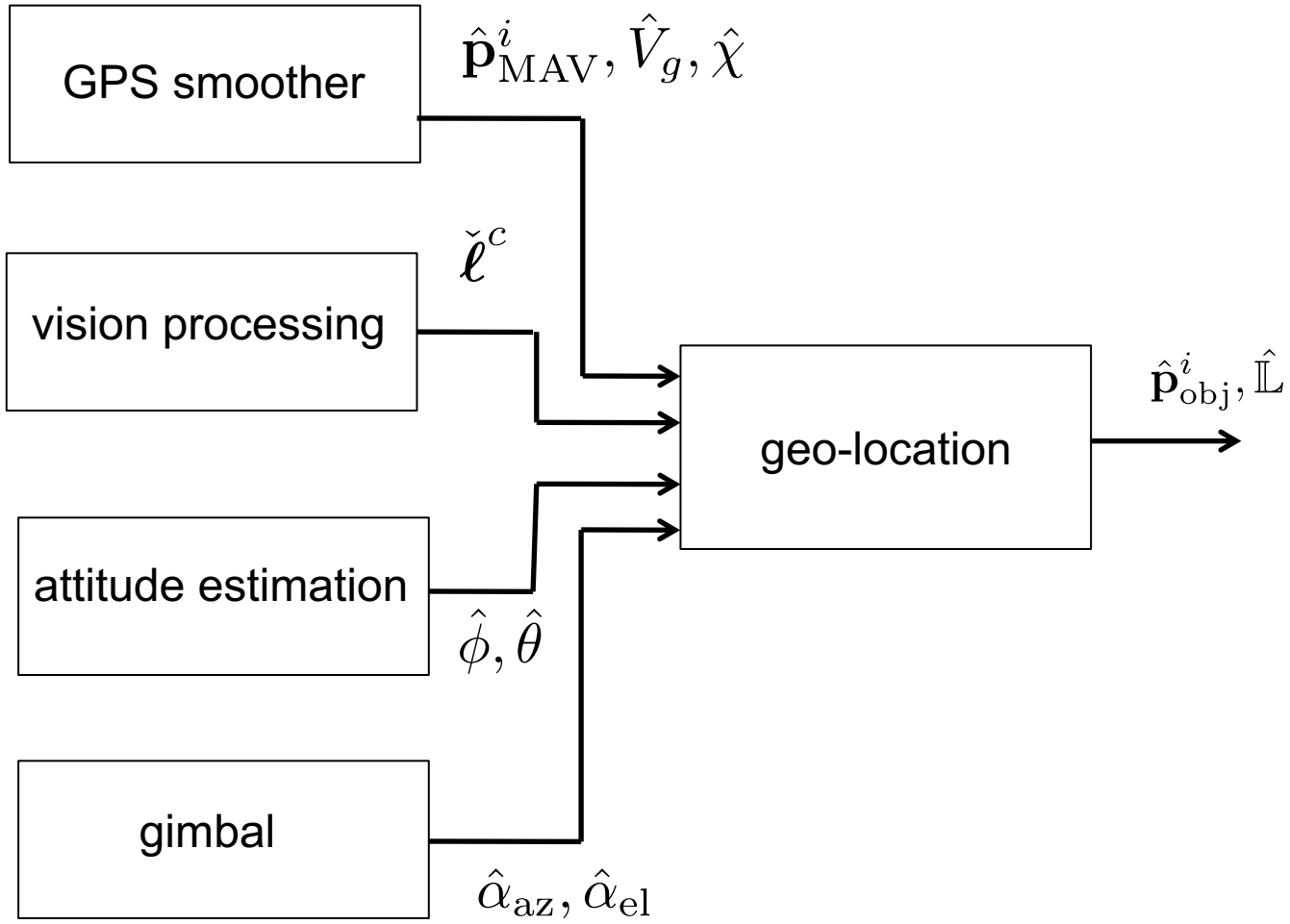
Prediction equations:
$$\begin{pmatrix} \dot{\hat{\mathbf{p}}}^i_{\text{obj}} \\ \dot{\hat{\mathbb{L}}} \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{(\hat{\mathbf{p}}^i_{\text{obj}} - \hat{\mathbf{p}}^i_{\text{MAV}})^\top \dot{\hat{\mathbf{p}}}^i_{\text{MAV}}}{\hat{\mathbb{L}}} \end{pmatrix}$$

Jacobian of prediction equation:
$$\frac{\partial f}{\partial x} = \begin{pmatrix} 0 & 0 \\ -\frac{\dot{\hat{\mathbf{p}}}^{iT}_{\text{MAV}}}{\hat{\mathbb{L}}} & \frac{(\hat{\mathbf{p}}^i_{\text{obj}} - \hat{\mathbf{p}}^i_{\text{MAV}})^\top \dot{\hat{\mathbf{p}}}_{\text{MAV}}}{\hat{\mathbb{L}}^2} \end{pmatrix}$$

Measurement equation:
$$\mathbf{p}^i_{\text{MAV}} = \mathbf{p}^i_{\text{obj}} - \mathbb{L}\left(\mathcal{R}^i_b \mathcal{R}^b_g \mathcal{R}^g_c \check{\boldsymbol{\ell}}^c\right)$$

Jacobian of measurement equation:
$$\frac{\partial h}{\partial x} = \begin{pmatrix} I & \mathcal{R}^i_b \mathcal{R}^b_g \mathcal{R}^g_c \check{\boldsymbol{\ell}}^c \end{pmatrix}$$

Geolocation Architecture



Target Motion in Image Plane

- The objective is to track targets seen in the image plane.
- Feature tracking, or other computer vision algorithms, are used to track pixels in image plane.
- Feature tracking is noisy, and so pixel location must be low pass filtered.
- Motion of the target in the image plane is due to two sources:
  - Self motion, or ego motion. Primarily due to rotation of the MAV.
  - Relative target motion.
- Since we are primarily interested in the relative motion, we must subtract the pixel movement due to ego motion.

Pixel LPF and Differentiation

Define the following

$$\bar{\boldsymbol{\epsilon}} = (\bar{\epsilon}_x, \bar{\epsilon}_y)^\top, \quad \text{Raw pixel measurements,}$$
$$\boldsymbol{\epsilon} = (\epsilon_x, \epsilon_y)^\top, \quad \text{Filtered pixel location,}$$
$$\dot{\boldsymbol{\epsilon}} = (\dot{\epsilon}_x, \dot{\epsilon}_y)^\top, \quad \text{Filtered pixel velocity}$$

Basic idea: LPF $\bar{\boldsymbol{\epsilon}}$ to obtain $\boldsymbol{\epsilon}$, and use dirty derivative to obtain $\dot{\boldsymbol{\epsilon}}$:

$$\boldsymbol{\epsilon}(s) = \frac{1}{\tau s + 1}\bar{\boldsymbol{\epsilon}}, \qquad \dot{\boldsymbol{\epsilon}}(s) = \frac{s}{\tau s + 1}\bar{\boldsymbol{\epsilon}}.$$

Digital Approximation

Tustin approximation $\qquad s \mapsto \frac{2}{T_s}\frac{z-1}{z+1},$

Converting to the z-domain gives

$$\boldsymbol{\epsilon}[z] = \frac{1}{\frac{2\tau}{T_s}\frac{z-1}{z+1}+1}\bar{\boldsymbol{\epsilon}} = \frac{\frac{T_s}{2\tau+T_s}(z+1)}{z-\frac{2\tau-T_s}{2\tau+T_s}}\bar{\boldsymbol{\epsilon}}$$
$$\dot{\boldsymbol{\epsilon}}[z] = \frac{\frac{2}{T_s}\frac{z-1}{z+1}}{\frac{2\tau}{T_s}\frac{z-1}{z+1}+1}\bar{\boldsymbol{\epsilon}} = \frac{\frac{2}{2\tau+T_s}(z-1)}{z-\frac{2\tau-T_s}{2\tau+T_s}}\bar{\boldsymbol{\epsilon}}.$$

Taking the inverse z-transform gives the difference equations

$$\boldsymbol{\epsilon}[n] = \left(\frac{2\tau-T_s}{2\tau+T_s}\right)\boldsymbol{\epsilon}[n-1] + \left(\frac{T_s}{2\tau+T_s}\right)(\bar{\boldsymbol{\epsilon}}[n]+\bar{\boldsymbol{\epsilon}}[n-1])$$
$$\dot{\boldsymbol{\epsilon}}[n] = \left(\frac{2\tau-T_s}{2\tau+T_s}\right)\dot{\boldsymbol{\epsilon}}[n-1] + \left(\frac{2}{2\tau+T_s}\right)(\bar{\boldsymbol{\epsilon}}[n]-\bar{\boldsymbol{\epsilon}}[n-1]),$$

where $\boldsymbol{\epsilon}[0] = \bar{\boldsymbol{\epsilon}}[0]$ and $\dot{\boldsymbol{\epsilon}}[0] = 0$.

Digital Approximation

Define $\beta = (2\tau - T_s)/(2\tau + T_s)$, and note that $1 - \beta = 2T_s/(2\tau + T_s)$. Then

$$\boldsymbol{\epsilon}[n] = \beta\, \boldsymbol{\epsilon}[n-1] + (1-\beta) \underbrace{\left(\frac{\bar{\boldsymbol{\epsilon}}[n] + \bar{\boldsymbol{\epsilon}}[n-1]}{2} \right)}_{\text{Average of last two samples}}$$

$$\dot{\boldsymbol{\epsilon}}[n] = \beta\, \dot{\boldsymbol{\epsilon}}[n-1] + (1-\beta) \underbrace{\left(\frac{\bar{\boldsymbol{\epsilon}}[n] - \bar{\boldsymbol{\epsilon}}[n-1]}{T_s} \right)}_{\text{Euler approximation of derivative}},$$

where $\boldsymbol{\epsilon}[0] = \bar{\boldsymbol{\epsilon}}[0]$ and $\dot{\boldsymbol{\epsilon}}[0] = 0$.

Apparent Motion

Let $\check{\boldsymbol{\ell}} \triangleq \boldsymbol{\ell}/\mathbb{L} = (\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}})/\left\|\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{MAV}}\right\|$ be the normalized relative position vector.

The Coriolis formula (expressed in camera frame) gives

$$\frac{d\check{\boldsymbol{\ell}}^c}{dt_i} = \frac{d\check{\boldsymbol{\ell}}^c}{dt_c} + \boldsymbol{\omega}^c_{c/i} \times \check{\boldsymbol{\ell}}^c.$$

True relative translational motion between target and MAV

Motion of target in image plane

Apparent motion due to rotation of MAV and gimbal

Apparent Motion, cont.

Motion of target in image plane:

$$\frac{d\check{\boldsymbol{\ell}}^c}{dt_c} = \frac{d}{dt_c}\frac{\begin{pmatrix}\epsilon_x\\ \epsilon_y\\ f\end{pmatrix}}{F} = \frac{F\begin{pmatrix}\dot{\epsilon}_x\\ \dot{\epsilon}_y\\ 0\end{pmatrix} - \dot{F}\begin{pmatrix}\epsilon_x\\ \epsilon_y\\ f\end{pmatrix}}{F^2} = \frac{F\begin{pmatrix}\dot{\epsilon}_x\\ \dot{\epsilon}_y\\ 0\end{pmatrix} - \frac{\epsilon_x\dot{\epsilon}_x+\epsilon_y\dot{\epsilon}_y}{F}\begin{pmatrix}\epsilon_x\\ \epsilon_y\\ f\end{pmatrix}}{F^2}$$

$$= \frac{1}{F^3}\begin{pmatrix}F^2-\epsilon_x^2 & -\epsilon_x\epsilon_y\\ -\epsilon_x\epsilon_y & F^2-\epsilon_y^2\\ -\epsilon_x f & -\epsilon_y f\end{pmatrix}\begin{pmatrix}\dot{\epsilon}_x\\ \dot{\epsilon}_y\end{pmatrix} = \frac{1}{F^3}\begin{pmatrix}\epsilon_y^2+f^2 & -\epsilon_x\epsilon_y\\ -\epsilon_x\epsilon_y & \epsilon_x^2+f^2\\ -\epsilon_x f & -\epsilon_y f\end{pmatrix}\dot{\boldsymbol{\epsilon}}$$

$$= Z(\boldsymbol{\epsilon})\dot{\boldsymbol{\epsilon}},$$

where

$$Z(\boldsymbol{\epsilon}) \triangleq \frac{1}{F^3}\begin{pmatrix}\epsilon_y^2+f^2 & -\epsilon_x\epsilon_y\\ -\epsilon_x\epsilon_y & \epsilon_x^2+f^2\\ -\epsilon_x f & -\epsilon_y f\end{pmatrix}\dot{\boldsymbol{\epsilon}}.$$

Apparent Motion, cont.

Ego Motion, $\boldsymbol{\omega}^c_{c/i} \times \check{\boldsymbol{\ell}}^c$:

$$\boldsymbol{\omega}^c_{c/i} = \underbrace{\boldsymbol{\omega}^c_{c/g}}_{0} + \underbrace{\boldsymbol{\omega}^c_{g/b}}_{\mathcal{R}^c_g \mathcal{R}^g_b \begin{pmatrix} -\sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ \dot{\alpha}_{\text{az}} \end{pmatrix}} + \underbrace{\boldsymbol{\omega}^c_{b/i}}_{\begin{pmatrix} p \\ q \\ r \end{pmatrix}}.$$

Therefore

$$\dot{\check{\boldsymbol{\ell}}}^c_{\text{app}} \triangleq \boldsymbol{\omega}^c_{c/i} \times \check{\boldsymbol{\ell}}^c = \frac{1}{F} \left[\mathcal{R}^c_g \mathcal{R}^g_b \begin{pmatrix} p - \sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ q + \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ r + \dot{\alpha}_{\text{az}} \end{pmatrix} \right] \times \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}$$

Total Motion in Image Plane

$$\frac{d\check{\boldsymbol{\ell}}^c}{dt_i} = \underbrace{Z(\boldsymbol{\epsilon})\dot{\boldsymbol{\epsilon}}}_{\text{Target Motion}} + \underbrace{\frac{1}{F}\left[\mathcal{R}_g^c \mathcal{R}_b^g \begin{pmatrix} p - \sin(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ q + \cos(\alpha_{\text{az}})\dot{\alpha}_{\text{el}} \\ r + \dot{\alpha}_{\text{az}} \end{pmatrix}\right] \times \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ f \end{pmatrix}}_{\text{Ego Motion}}$$

Time to Collision

For all objects in the camera field of view, it is possible to compute the time to collision to that obstacle, defined as:

The time to collision if the vehicle were to proceed along the line-of-sight vector to the obstacle at the current closing velocity.

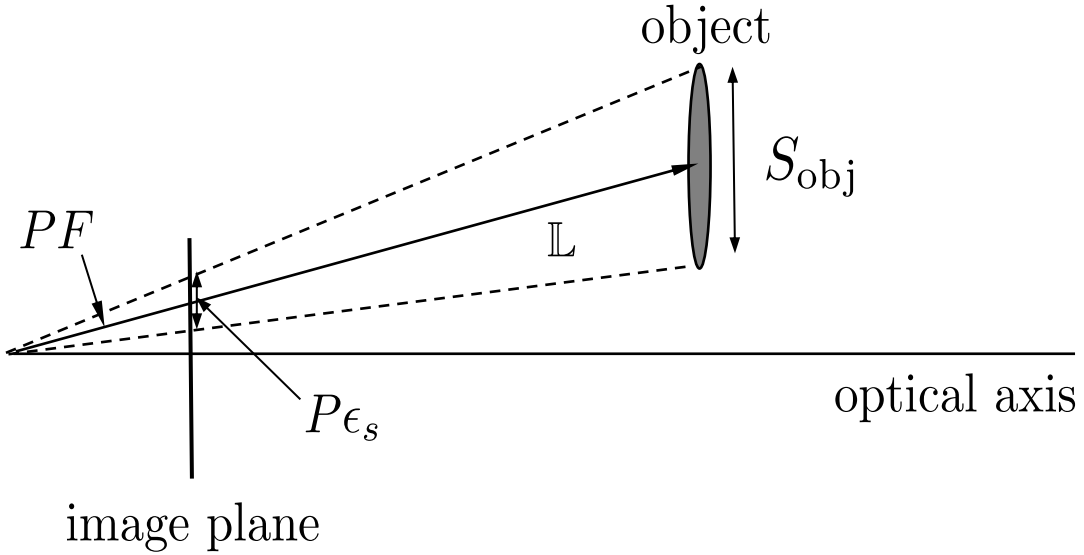
Therefore, for each obstacle, the time to collision is given by

$$t_c \triangleq -\frac{\mathbb{L}}{\dot{\mathbb{L}}}$$

Impossible to compute using only a monocular camera because of scale ambiguity.
Two techniques:

- Looming $\leftarrow$ Requires target size.
- Flat earth approximation $\leftarrow$ Requires height above ground.

Time to Collision - Looming



From similar triangles:

$$\frac{S_{\text{obj}}}{\mathbb{L}} = \frac{P\epsilon_s}{PF} = \frac{\epsilon_s}{F}$$

If S_{obj} is not changing, then

$$\begin{aligned}
\frac{\dot{\mathbb{L}}}{\mathbb{L}} &= \frac{\mathbb{L}}{S_{\text{obj}}}\left[\frac{\epsilon_s}{F}\frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{F}\right] \\
&= \frac{F}{\epsilon_s}\left[\frac{\epsilon_s}{F}\frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{F}\right] \\
&= \frac{\dot{F}}{F} - \frac{\dot{\epsilon}_s}{\epsilon_s} \\
&= \frac{\epsilon_x\dot{\epsilon}_x + \epsilon_y\dot{\epsilon}_y}{F} - \frac{\dot{\epsilon}_s}{\epsilon_s},
\end{aligned}$$

the inverse of which is the time to collision t_c.

Time to Collision – Flat Earth

$$\mathbb{L} = \frac{h}{\cos\varphi}$$

Differentiation gives

$$\frac{\dot{\mathbb{L}}}{\mathbb{L}} = \frac{\dot{h}}{h} + \dot{\varphi}\tan\varphi.$$

From geometry we have

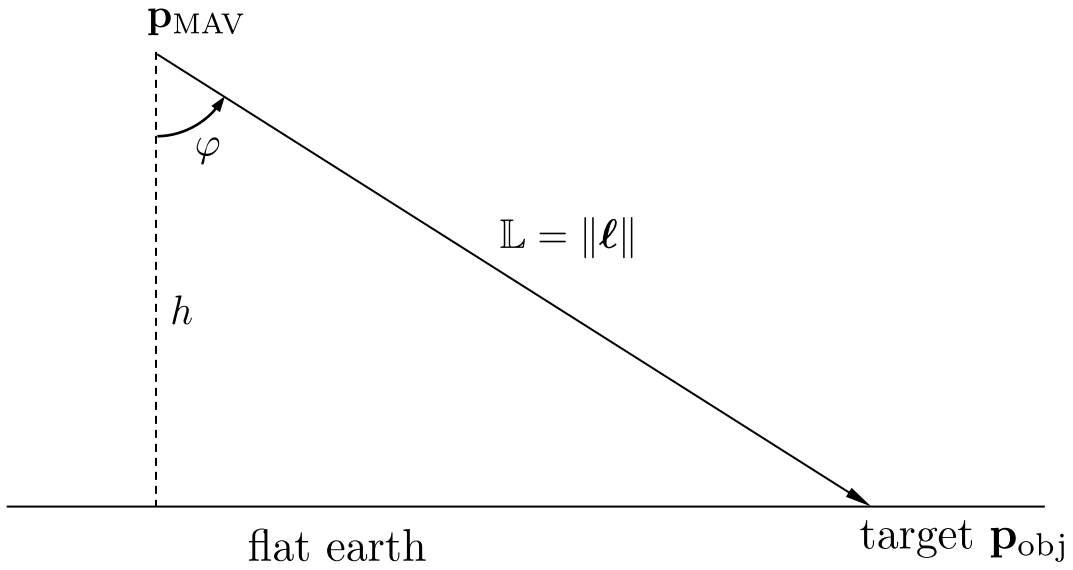
$$\cos\varphi = \check{\ell}^i_z.$$

Differentiate and solve for $\dot{\varphi}$:

$$\dot{\varphi} = -\frac{1}{\sin\varphi}\frac{d}{dt_i}\check{\ell}^i_z$$

Therefore,

$$\dot{\varphi}\tan\varphi = -\frac{1}{\cos\varphi}\frac{d}{dt_i}\check{\ell}^i_z = -\frac{1}{\check{\ell}^i_z}\frac{d}{dt_i}\check{\ell}^i_z$$



Precision Landing

For precision landing, we use the guidance model

$$\begin{aligned}
\dot{p}_n &= V_g \cos\chi \cos\gamma \\
\dot{p}_e &= V_g \sin\chi \cos\gamma \\
\dot{p}_d &= -V_g \sin\gamma \\
\dot{\chi} &= \frac{g}{V_g} \tan\phi \\
\dot{\phi} &= u_1 \\
\dot{\gamma} &= u_2.
\end{aligned}$$

Define:

$$\begin{aligned}
\mathbf{p}^i_{\text{MAV}} &= \begin{pmatrix} p_n, & p_e, & p_d \end{pmatrix}^\top \\
\mathbf{v}^i_{\text{MAV}} &= \begin{pmatrix} V_g \cos\chi\cos\gamma, & V_g \sin\chi\cos\gamma, & -V_g\sin\gamma \end{pmatrix}^\top \\
\mathbf{v}^{v_2}_{\text{MAV}} &= \begin{pmatrix} V_g, & 0, & 0 \end{pmatrix}^\top
\end{aligned}$$

Proportional Navigation (PN)

Proportional Navigation: align the line-of-sight rate $\dot{\boldsymbol{\ell}}$ with the negative line-of-sight vector $-\boldsymbol{\ell}$.

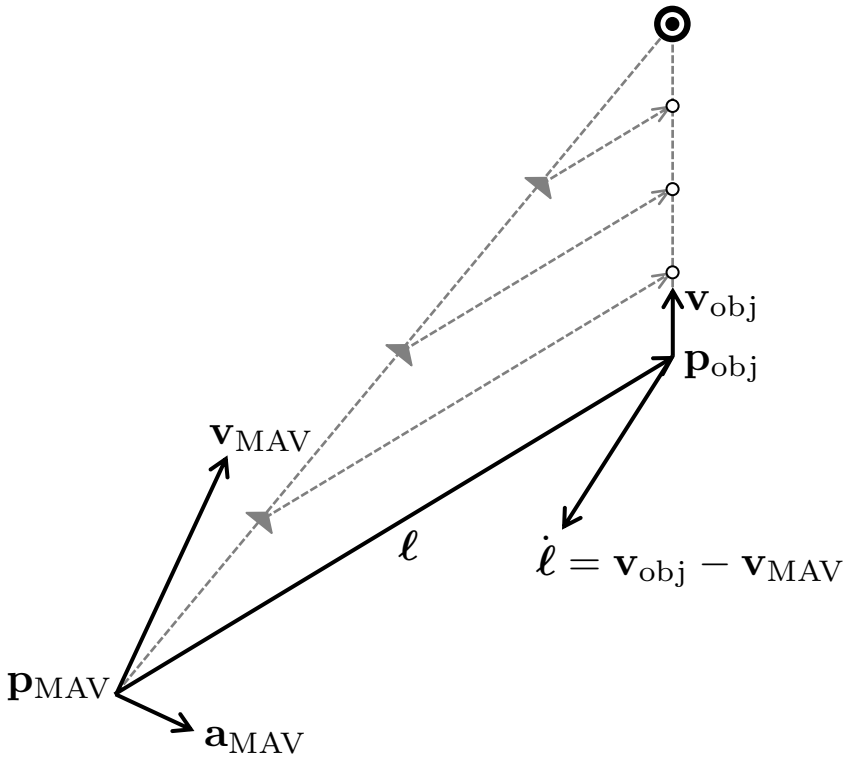
Define

$$\boldsymbol{\Omega}_\perp = \check{\boldsymbol{\ell}} \times \frac{\dot{\boldsymbol{\ell}}}{\mathbb{L}}.$$

Assuming constant velocity, we can only accelerate in the plane perpendicular to the velocity vector. Therefore, to zero $\boldsymbol{\Omega}_\perp$, the desired acceleration is

$$\mathbf{a}_{\text{MAV}} = N\boldsymbol{\Omega}_\perp \times \mathbf{v}_{\text{MAV}},$$

where $N > 0$ is the (tunable) navigation constant.



Acceleration Command

To implement the acceleration command, must convert to vehicle-2 frame as

$$\begin{aligned}\mathbf{a}_{\text{MAV}}^{v2} &= \mu N \boldsymbol{\Omega}_{\perp}^{v2} \times \mathbf{v}_{\text{MAV}}^{v2} \\ &= \mu N \begin{pmatrix} \Omega_{\perp,x}^{v2} \\ \Omega_{\perp,y}^{v2} \\ \Omega_{\perp,z}^{v2} \end{pmatrix} \times \begin{pmatrix} V_g \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \mu N V \Omega_{\perp,z}^{v2} \\ -\mu N V \Omega_{\perp,y}^{v2} \end{pmatrix},\end{aligned}$$

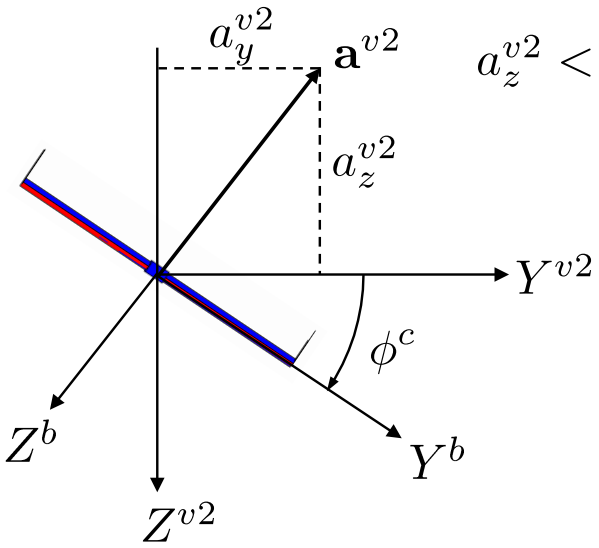
where

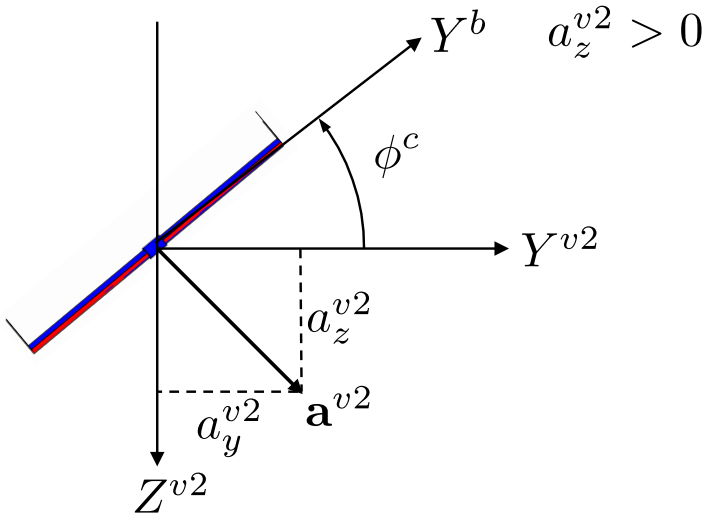
$$\begin{aligned}\boldsymbol{\Omega}_{\perp}^{v2} &= \mathcal{R}_b^{v2} \mathcal{R}_g^b \mathcal{R}_c^g \boldsymbol{\Omega}_{\perp}^c \\ \boldsymbol{\Omega}_{\perp}^c &= \check{\boldsymbol{\ell}}^c \times \frac{\dot{\boldsymbol{\ell}}^c}{\mathbb{L}} \\ \frac{\dot{\boldsymbol{\ell}}^c}{\mathbb{L}} &= \frac{\dot{\mathbb{L}}}{\mathbb{L}} \check{\boldsymbol{\ell}}^c + Z(\boldsymbol{\epsilon}) \dot{\boldsymbol{\epsilon}} + \dot{\check{\boldsymbol{\ell}}}_{\text{app}}^c\end{aligned}$$

where the inverse of time to collision $\dot{\mathbb{L}}/\mathbb{L}$ can be estimated using looming or flat earth model.

Polar Converting Logic

Use polar converting logic to convert acceleration command $\mathbf{a}^{v2}$ to roll angle and flight path angle commands:





$$\phi^c = \tan^{-1}\left(\frac{a_y^{v2}}{-a_z^{v2}}\right)$$

$$V_g\,\dot{\gamma}^c = \sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}.$$

$$\phi^c = \tan^{-1}\left(\frac{a_y^{v2}}{a_z^{v2}}\right)$$

$$V_g\,\dot{\gamma}^c = -\sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}.$$

Polar Converting Logic, cont.

General rule:

$$\phi^c = \tan^{-1}\left(\frac{a_y^{v2}}{|a_z^{v2}|}\right)$$

$$\dot{\gamma}^c = -\text{sign}(a_z^{v2})\frac{1}{V_g}\sqrt{(a_y^{v2})^2 + (a_z^{v2})^2}$$

Note discontinuity in ϕ^c at $(a_y^{v2}, a_z^{v2}) = (0, 0)$. When $a_z^{v2} = 0$, then

- When $a_y^{v2} > 0 \rightarrow \phi^c = \pi/2$
- When $a_y^{v2} < 0 \rightarrow \phi^c = -\pi/2$

Remove discontinuity as

$$\phi^c = \sigma(a_y^{v2})\tan^{-1}\left(\frac{a_y^{v2}}{|a_z^{v2}|}\right)$$

where

$$\sigma(a_y^{v2}) = \text{sign}(a_y^{v2})\frac{1 - e^{-ka_y^{v2}}}{1 + e^{-ka_y^{v2}}}$$