Autonomous and Mobile Robotics

Visual Servoing for Unmanned Aerial Vehicles

(prepared by Lorenzo Rosa)

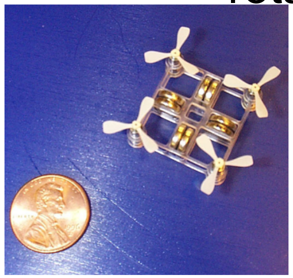
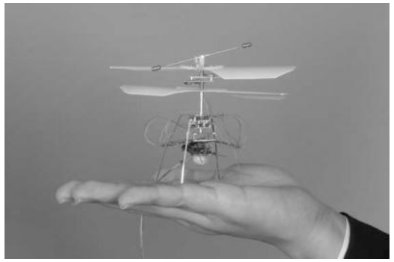
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

unmanned aerial vehicles (UAVs)

autonomous/semi-autonomous vehicles of variable size

- rotary wing (e.g. quadrotors, coaxials)


- fixed wing (aeroplanes)



mainly used in repetitive or risky operations:

- surveillance/data acquisition (area monitoring, patrolling, meteorology, geology, traffic/pollution monitoring)
- risky/disaster scenarios (search and rescue, fire-fighting, volcanology)
- service/entertainment (transportation and delivery, cinematography)

fixed vs rotary wings UAVs

fixed wings:

- high endurance (time of flight can be long), high payload capabilities (e.g. more sensors, more computational power)
- a runway is needed to take off and land
  (small models can be launched/caught)
- non-zero forward velocity is needed to fly
  (due to aerodynamic constraints)

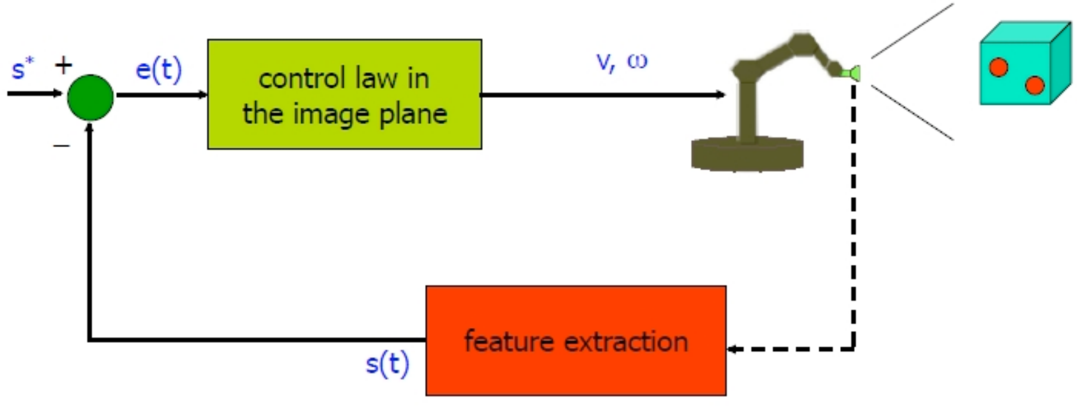
rotary wings

- high manoeuvrability
- vertical Take Off and Landing
  (can land on very small areas)
- able to perform stationary/slow flight
  (useful to perform long time tasks in the same position)
- can easily fly in small and cluttered environment
  (e.g. by performing hovering and slow motion)

visual servoing (recall)

Image Based Visual Servoing (IBVS)

- control the robot to ensure convergence of features error in the image plane
- control law is designed considering feature dynamics
- configuration is eye-in-hand (robot motion → camera motion)



for UAVs the resulting motion depends on the vehicle.
note that, if the target is still:

- loitering for Fixed wings (can not stop on the target)
- hovering for quadrotors

used for sourveillance, monitoring, patrolling, ...

task definition

task:

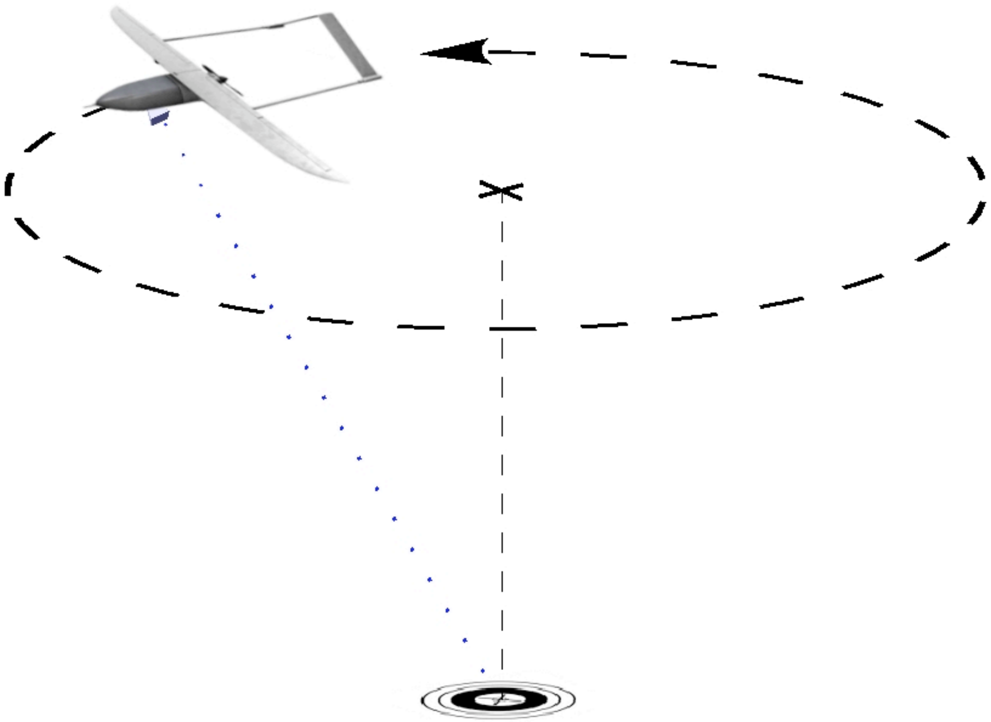
chosen a target, we want to track a visual point feature (centroid or a characteristic point on the target) in order to perform continuous monitoring by keeping the UAV in flight above it

system:

UAV (either rotary or fixed wing), equipped with:

- proprioceptive sensors
  - inertial Measurements Unit (attitude)
  - encoders (camera pan and tilt angles)
- exteroceptive sensors
  - altimeter (altitude)
  - camera (environment, target)

visual servoing - fixed wing uav



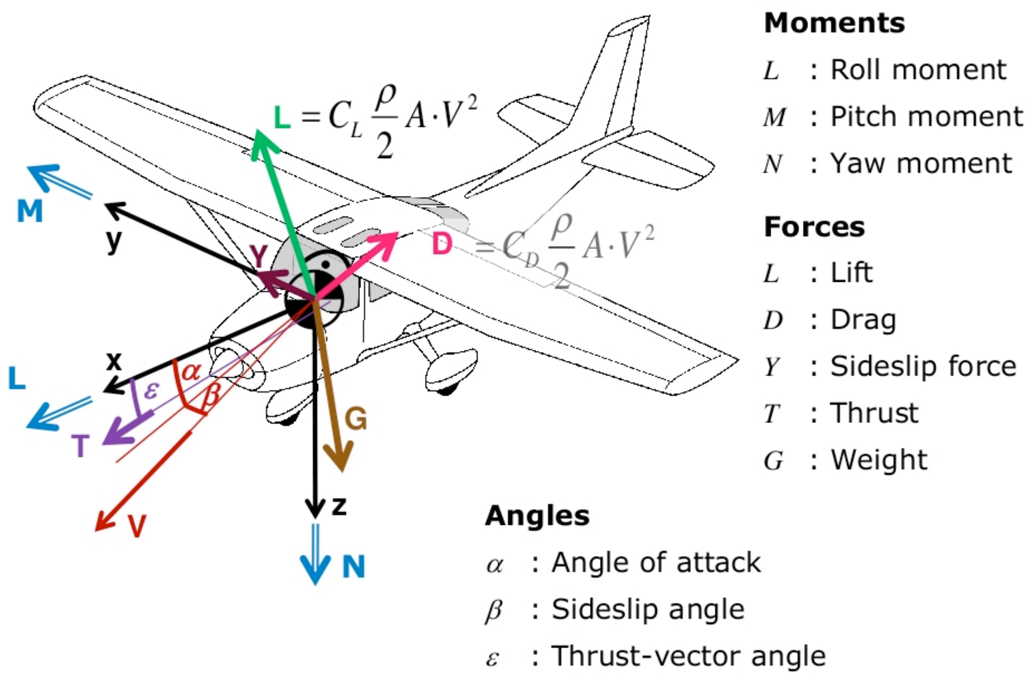
forces/moments diagram

motion is given by

- mechanical components (gravity, inertia, …)
- aerodynamic effects (lift, drag, …)

the complete model is rather complex

- some components can be "statically" (by aerodynamics) or dynamically stabilized
- it is common to have low level control loops to stabilize altitude, attitude, cruise speed
- a simplified model can be used to design control for high level tasks

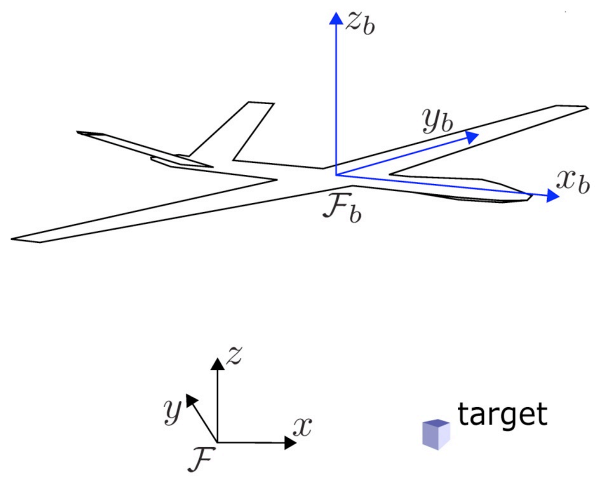


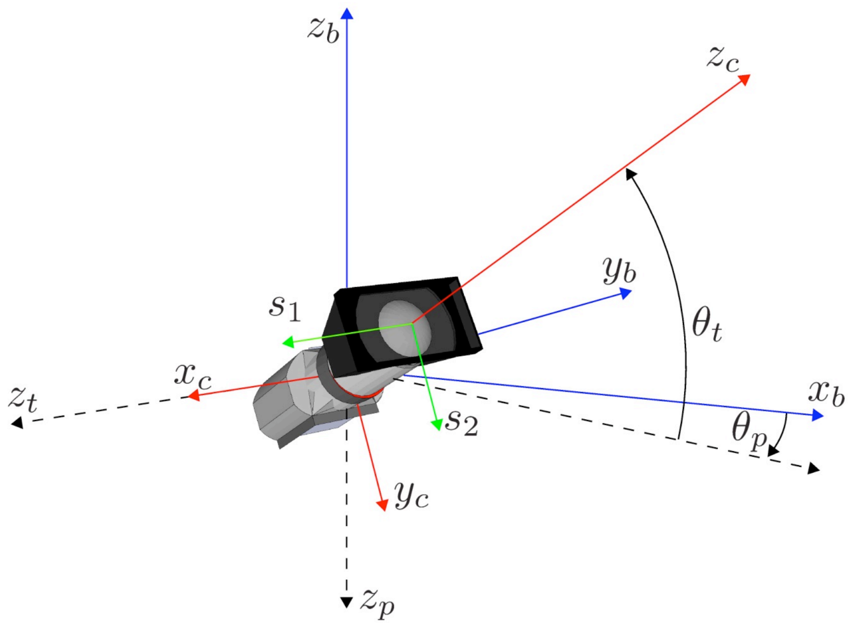
system modeling for control design

generalized coordinates (UAV)

cartesian coordinates $(x,\, y,\, z)$ of the origin of $\mathcal{F}_b$ in w.r.t inertial frame $\mathcal{F}$

orientation $(\psi,\, \theta,\, \phi)$ of $\mathcal{F}_b$ w.r.t. $\mathcal{F}$





generalized coordinates (camera)

camera pan θ_p and tilt θ_t angle

system modelling for control design

to ease the study (w.l.o.g.) we consider the following simplifying assumptions:

- no wind
- UAV cruising at costant (known) speed v and altitude
- pitch, sideslip and attack angles are zero
- camera is centered in the UAV c.o.g.
- camera pan and tilt joints are centered in camera focus

corresponding simplified model of UAV + pan-tilt system $\Longrightarrow$

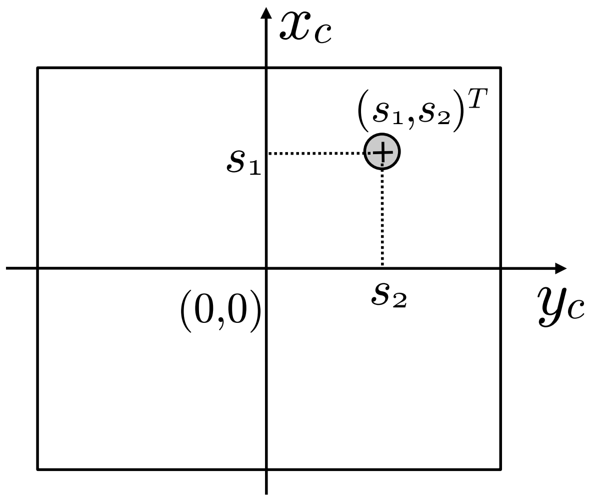
$$\begin{cases} \dot{x} &= v\cos\psi \\ \dot{y} &= v\sin\psi \\ \dot{\psi} &= -\dfrac{g}{v}\tan\phi \\ \dot{\phi} &= u_\phi \\ \dot{\theta}_p &= u_p \\ \dot{\theta}_t &= u_t \end{cases}$$

control inputs:

- roll rate u_ϕ
- pan rate u_p
- tilt rate u_t

system modelling – image features

target position in the image plane $(x_c,\ y_c)$ is expressed by point features coordinates $s=(s_1\ ,\ s_2)^T$



denoting by Z the depth of the target, features motion in the image plane is related to camera motion by the so-called interaction matrix $J_\mathrm{i}(s,Z)$

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_i(\boldsymbol{s}, Z) \begin{pmatrix} \boldsymbol{v}_c \\ \boldsymbol{\omega}_c \end{pmatrix}$$

by using the camera "jacobian" $J_\mathrm{c}(\psi,\phi,\theta_p,\theta_t)$, we can finally relate features motion to UAV + pan-tilt system

$$\dot{s} = \mathbf{J}_i(\mathbf{s}, Z) J_c(\phi, \theta, \psi) \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}$$

NOTE: in J_i we need the target depth (possible choices are desired, initial or mean value)

task definition

task (recall):

chosen a target, we want to track a visual point feature (centroid or a characteristic point on the target) in order to perform continuous monitoring by keeping the UAV in flight above it

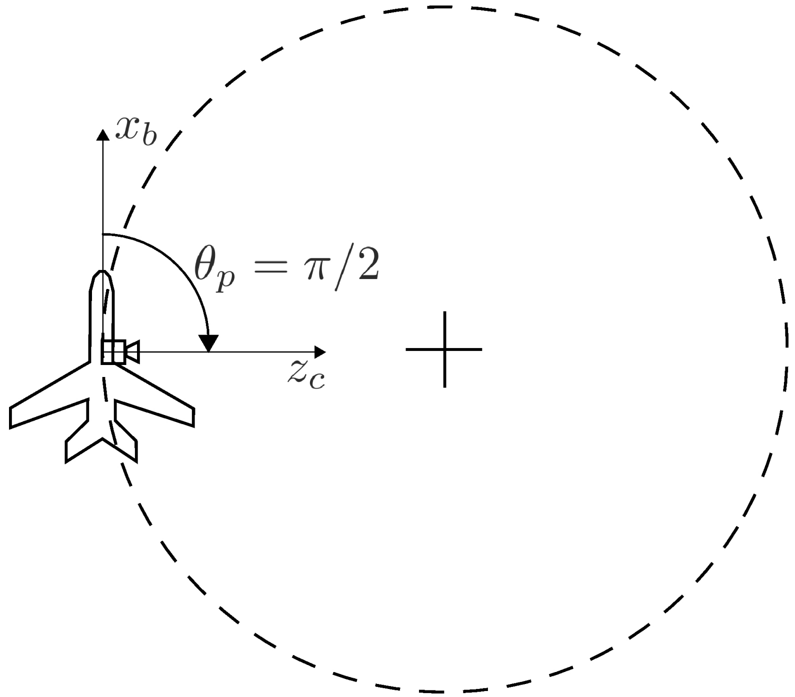
task (for fixed wing UAV)

move the UAV along a circular trajectory centered above the target, while keeping the target in the center of image plane

- a purely visual definition of the task is not sufficient
  - pan-tilt only is sufficient for zeroing the error in the image plane
  - we are using a single point feature: no scale (depth) information
- we can extend the task by adding a constraint on pan angle:

$$\{s = 0,\ \theta_p = \pi/2\}$$

circular trajectories



why circular?

- intuitively (<u>it can be proven</u>), only moving along these trajectories the UAV will maintain the set-point $\{s = 0,\ \theta_p = \pi/2\}$
- moreover, along these trajectories we will have $\phi = \mathrm{const}$ thus no control input on roll angle is needed at steady state
- a circular trajectory allows the UAV to monitor the target from every side (useful, e.g. to estimate target position)
- trajectory direction (CW, CCW) will depend on the sign of the pan angle

control approach

by letting

$$\boldsymbol{G}(\psi) = \left(\begin{array}{c|c} \begin{matrix} \cos\psi \\ \sin\psi \end{matrix} & \boldsymbol{O}_{2\times 4} \\ \hline \boldsymbol{O}_{4\times 1} & \boldsymbol{I}_{4\times 4} \end{array} \right)$$

the features dynamics become

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_i \boldsymbol{J}_c \boldsymbol{G} \begin{pmatrix} v \\ \dot{\psi} \\ \dot{\phi} \\ \dot{\theta}_p \\ \dot{\theta}_t \end{pmatrix} = \boldsymbol{J} \begin{pmatrix} v \\ \dot{\psi} \\ \dot{\phi} \\ \dot{\theta}_p \\ \dot{\theta}_t \end{pmatrix} = \boldsymbol{J}_{1-2} \underbrace{\begin{pmatrix} v \\ \dot{\psi} \end{pmatrix}}_{\text{drift terms}} + \boldsymbol{J}_{3-5} \underbrace{\begin{pmatrix} u_\phi \\ u_p \\ u_t \end{pmatrix}}_{\text{control inputs}}$$

while the dynamics of the output variables are:

$$\begin{pmatrix} \dot{\boldsymbol{s}} \\ \dot{\theta}_p \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_{1-2} \\ 0 \quad 0 \end{pmatrix} \begin{pmatrix} v \\ \dot{\psi} \end{pmatrix} + \begin{pmatrix} \boldsymbol{J}_{3-5} \\ 0 \quad 1 \quad 0 \end{pmatrix} \begin{pmatrix} u_\phi \\ u_p \\ u_t \end{pmatrix}$$

feedback linearization is not possible due to a singularity exactly at the set-point

backstepping control (sketch)

a possible approach to stabilize a cascade system in the form

$$\dot{x}_1 = f_1(x_1) + g_1(x_1)x_2$$
$$\dot{x}_2 = f_2(x_1, x_2) + g_2(x_1, x_2)u$$

actual control input

is given by the backstepping technique

backstepping technique (informal description)

given a system in lower triangular form:

- by using a virtual input $\alpha(x_1)$ stabilize (Lyapunov criteria) the first set of variables x_1
- using the real input, force the second set of variables x_2 to match the virtual input: $x_2 \to \alpha(x_1) \Rightarrow \mathrm{e}_1 \to 0$

modified system - model

assuming that a direct control of the yaw rate is available (by means of the virtual input u_ψ) and consider u_ϕ as an exogenous signal

$$
\left.\begin{aligned}
\dot{x} &= v\cos\psi \\
\dot{y} &= v\sin\psi \\
\dot{\psi} &= \boxed{u_\psi}
\end{aligned}\right\}\ \text{Unicycle like model}
$$

$$
\begin{aligned}
\dot{\phi} &= u_\phi \\
\dot{\theta}_p &= u_p \\
\dot{\theta}_t &= u_t .
\end{aligned}
$$

we get the following modified dynamics:

$$
\begin{pmatrix} \dot{\boldsymbol{s}} \\ \dot{\theta}_p \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_3 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v \\ \dot{\phi} \end{pmatrix} + \begin{pmatrix} \boldsymbol{J}_2 & \boldsymbol{J}_4 & \boldsymbol{J}_5 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_\psi \\ u_p \\ u_t \end{pmatrix} = \boldsymbol{J}_A \underbrace{\begin{pmatrix} v \\ \dot{\phi} \end{pmatrix}}_{\text{modified drift terms}} + \boldsymbol{J}_B \underbrace{\begin{pmatrix} u_\psi \\ u_p \\ u_t \end{pmatrix}}_{\text{modified control inputs}}
$$

modified system - control

to control the modified dvnamics

$$\begin{pmatrix} \dot{\boldsymbol{s}} \\ \dot{\theta}_p \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_3 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v \\ \dot{\phi} \end{pmatrix} + \begin{pmatrix} \boldsymbol{J}_2 & \boldsymbol{J}_4 & \boldsymbol{J}_5 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_\psi \\ u_p \\ u_t \end{pmatrix} = \boldsymbol{J}_A \begin{pmatrix} v \\ \dot{\phi} \end{pmatrix} + \boldsymbol{J}_B \begin{pmatrix} u_\psi \\ u_p \\ u_t \end{pmatrix}$$

we can set the following vector input ($K > 0$)

$$\begin{pmatrix} u_\psi \\ u_p \\ u_t \end{pmatrix} = -\boldsymbol{J}_B^{-1} \left(\boldsymbol{K}\boldsymbol{e} + \boldsymbol{J}_A \begin{pmatrix} v \\ \dot{\phi} \end{pmatrix} \right)$$

we get decoupled exponential convergence
for the error vector $e=(s\ ,\ \theta_p$ - $\pi/2)^T$ $\Longrightarrow$ $\boxed{\dot{\boldsymbol{e}} = -\boldsymbol{K}\boldsymbol{e}}$

comparison of original and modified dynamics

original system

$$\begin{aligned}\dot{x} &= v\cos\psi\\ \dot{y} &= v\sin\psi\\ \boxed{\dot{\psi} = -\frac{g}{v}\tan\phi}\\ \dot{\phi} &= u_\phi\\ \dot{\theta}_p &= u_p\\ \dot{\theta}_t &= u_t\end{aligned}$$

$$\dot{e} = \begin{pmatrix}\dot{s}\\ \dot{\theta}_p\end{pmatrix} = \begin{pmatrix}\boldsymbol{J}_{1-2}\\ 0\ \ 0\end{pmatrix}\begin{pmatrix}v\\ \dot{\psi}\end{pmatrix} + \begin{pmatrix}\boldsymbol{J}_{3-5}\\ 0\ \ 1\ \ 0\end{pmatrix}\begin{pmatrix}u_\phi\\ u_p\\ u_t\end{pmatrix}$$

introduction of virtual input u_ψ

modified drift term

modified input vector

modified system (with stabilizing control)

$$\begin{aligned}\dot{x} &= v\cos\psi\\ \dot{y} &= v\sin\psi\\ \boxed{\dot{\psi} = u_\psi}\\ \dot{\phi} &= u_\phi\\ \dot{\theta}_p &= u_p\\ \dot{\theta}_t &= u_t.\end{aligned}$$

$$\dot{e} = \begin{pmatrix}\dot{s}\\ \dot{\theta}_p\end{pmatrix} = \begin{pmatrix}\boldsymbol{J}_1 & \boldsymbol{J}_3\\ 0 & 0\end{pmatrix}\begin{pmatrix}v\\ \dot{\phi}\end{pmatrix} + \begin{pmatrix}\boldsymbol{J}_2 & \boldsymbol{J}_4 & \boldsymbol{J}_5\\ 0 & 1 & 0\end{pmatrix}\begin{pmatrix}u_\psi\\ u_p\\ u_t\end{pmatrix}$$

$$\begin{pmatrix}u_\psi\\ u_p\\ u_t\end{pmatrix} = -\boldsymbol{J}_B^{-1}\left(\boldsymbol{K}\boldsymbol{e} + \boldsymbol{J}_A\begin{pmatrix}v\\ \dot{\phi}\end{pmatrix}\right) \Rightarrow \dot{\boldsymbol{e}} = -\boldsymbol{K}\boldsymbol{e}$$

backstepping to the original system

considering the original system

$$\dot{\boldsymbol{e}} = \begin{pmatrix} \dot{\boldsymbol{s}} \\ \dot{\theta}_p \end{pmatrix} = \boldsymbol{J}_A \begin{pmatrix} v \\ \dot{\phi} \end{pmatrix} + \boldsymbol{J}_B \begin{pmatrix} -\frac{g}{v}\tan\phi \\ u_p \\ u_t \end{pmatrix}$$

adding and subtracting $\boldsymbol{J}_B \left(u_\psi \quad u_p \quad u_t \right)^T$ we get

$$\dot{\boldsymbol{e}} = \boldsymbol{J}_A \begin{pmatrix} v \\ \dot{\phi} \end{pmatrix} + \boldsymbol{J}_B \begin{pmatrix} u_\psi \\ u_p \\ u_t \end{pmatrix} + \boldsymbol{J}_B \begin{pmatrix} \xi \\ 0 \\ 0 \end{pmatrix} \longrightarrow \xi = -\frac{g}{v}\tan\phi - u_\psi$$

residual term
due to mismatch between
virtual and actual control

and the error dynamics will be modified by the residual dynamics

$$\dot{\xi} = -\frac{g}{v}\frac{1}{\cos^2\phi}\dot{\phi} - \dot{u}_\psi = -\frac{g}{v}\frac{1}{\cos^2\phi}u_\phi - \dot{u}_\psi = \boxed{w} \quad \Longrightarrow \quad \dot{\boldsymbol{e}} = -\boldsymbol{K}\boldsymbol{e} + \xi\,\boldsymbol{J}_{B,1}$$

auxiliary input
depending on u_ϕ

backstepping to the original system

by setting the auxiliary input as

$$w = -\boldsymbol{e}^T \boldsymbol{J}_{B,1} - k_\xi \, \xi, \qquad k_\xi > 0$$

yields the convergence of the residual ξ to zero and (thus) the convergence of the error e to zero

(it <u>can be proven</u> by using Lyapunov function)

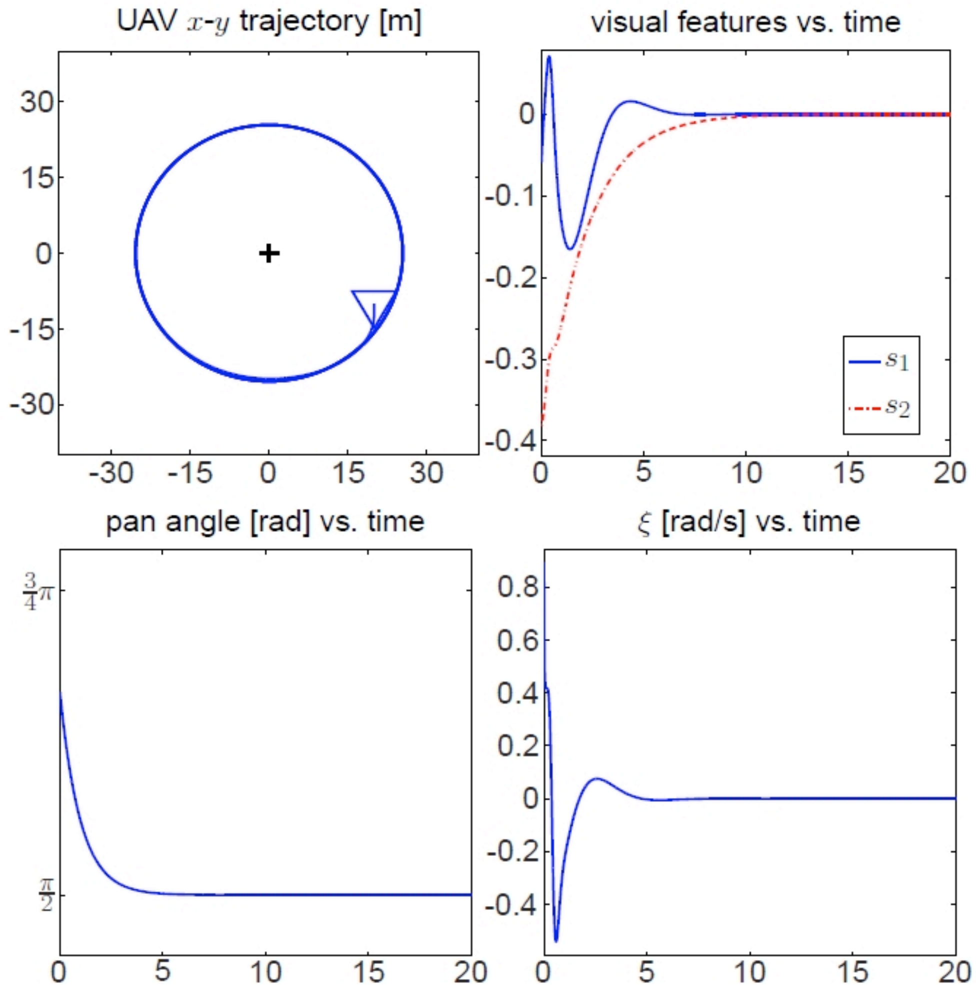
finally the roll rate (actual input to the real system) will be

$$\begin{aligned} \dot{x} &= v \cos\psi \\ \dot{y} &= v \sin\psi \\ \dot{\psi} &= -\frac{g}{v} \tan\phi \\ \boxed{\dot{\phi} = u_\phi} \\ \dot{\theta}_p &= u_p \\ \dot{\theta}_t &= u_t \end{aligned}$$

$$u_\phi = \frac{v}{g} \cos^2\phi \, (\boldsymbol{e}^T \boldsymbol{J}_{B,1} + k_\xi \, \xi - \dot{u}_\psi)$$

calculated before to stabilize the modified system

basic result



initial conditions: $(x_0, y_0, \psi_0, \phi_0, \theta_{p0}, \theta_{t0}) = (20, -10, \frac{3}{2}\pi, 0, \frac{2}{3}\pi, -\frac{\pi}{4})$

improvements

one may want to enforce a desired radius: task priority

- recall (from mechanics) that $\omega = v/R$
- introduce a feedforward term on the yaw rate

$$\dot{\psi}_d = \frac{v}{\rho_d}$$

- modify the control in a task-priority sense: primary task is tracking the target, secondary task is enforce the desired radius

$$\begin{pmatrix} u_\psi \\ u_p \\ u_t \end{pmatrix} = \underbrace{-(\boldsymbol{J}_B^{1-2})^\dagger \left(\boldsymbol{K}_s \boldsymbol{s} + \boldsymbol{J}_A^{1-2} \begin{pmatrix} v \\ \dot{\phi} \end{pmatrix}\right)}_{\text{control for feature tracking}} + \boldsymbol{P} \begin{pmatrix} \dot{\psi}_d \\ k_p \left(\frac{\pi}{2} - \theta_p\right) \\ 0 \end{pmatrix}$$

enforce desired angular velocity

projector in the null space of the main task

$$\boldsymbol{P} = (\boldsymbol{I} - (\boldsymbol{J}_B^{1-2})^\dagger \boldsymbol{J}_B^{1-2})$$

improvements

avoid backstepping: linear roll control

- at each time interval, convert the desired virtual control in a desired roll value

$$\bar{\phi} = \arctan\left(-u_\psi \frac{v}{g}\right)$$

- obtain the desired roll value by a linear control

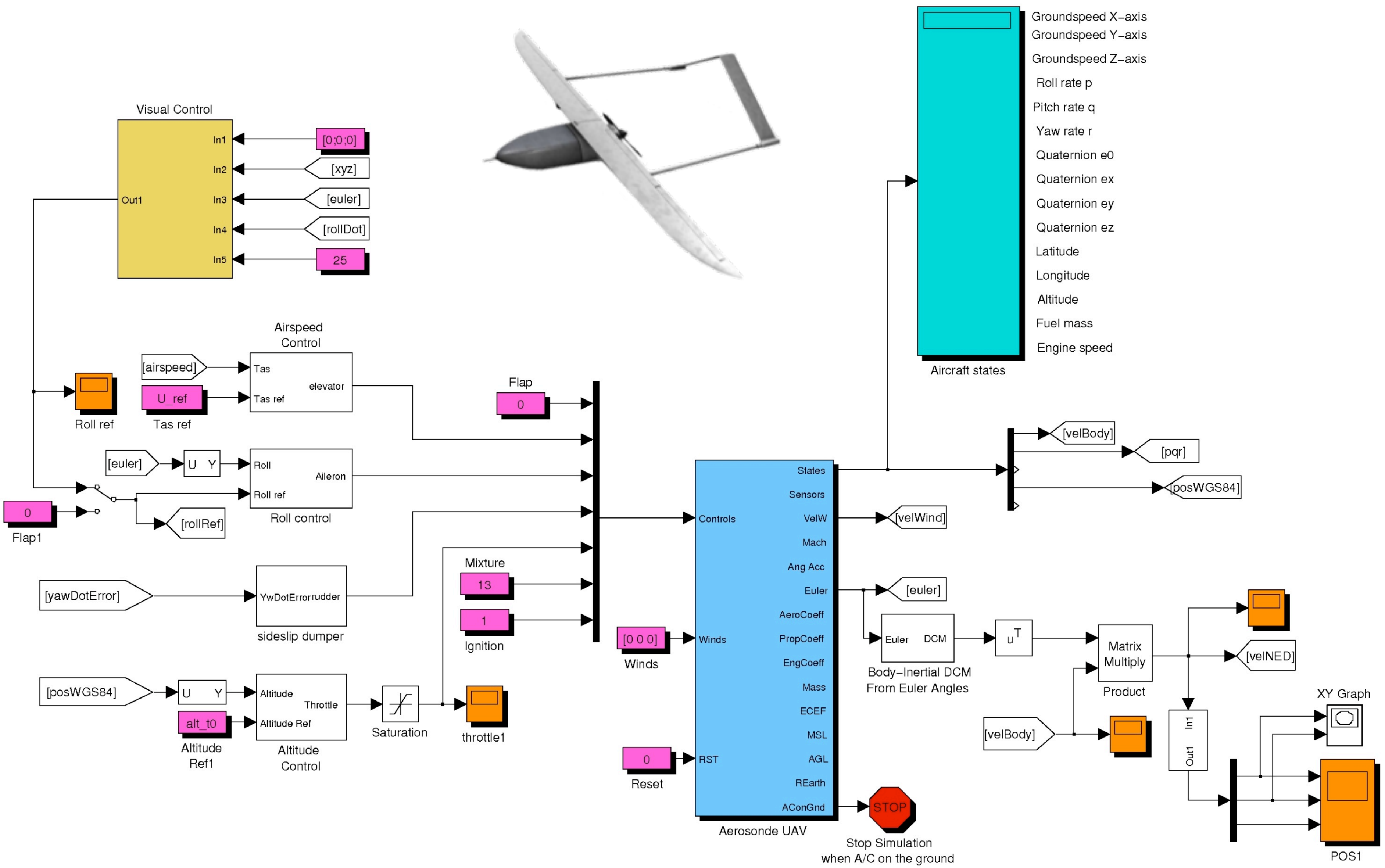
$$u_\phi = k_\phi(\bar{\phi} - \phi), \qquad k_\phi > 0$$

- computationally less expensive

avoid approximation: estimate target depth

- it can be made by using a (nonlinear) depth estimator
- removes the approximation in the interaction matrix J_i

aerosonde simulator – simulink

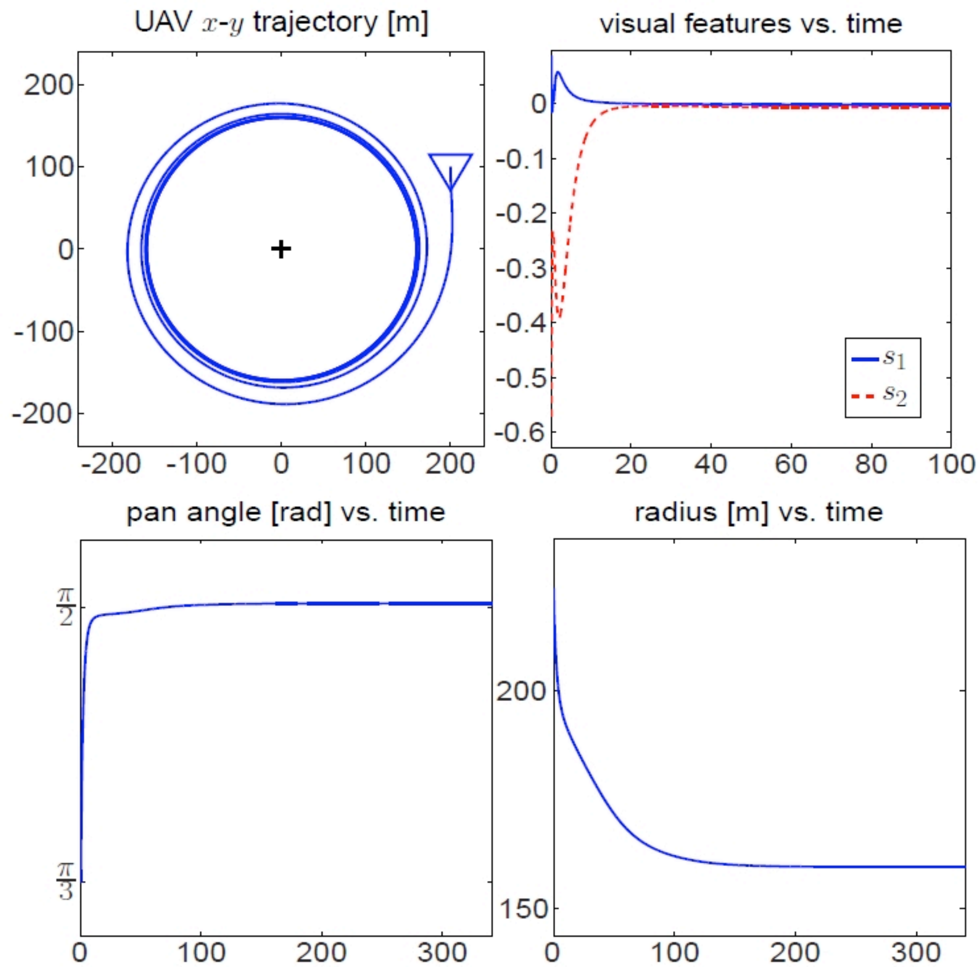


aerosonde simulator – simulink

simulation of a complete model of a real fixed wing UAV

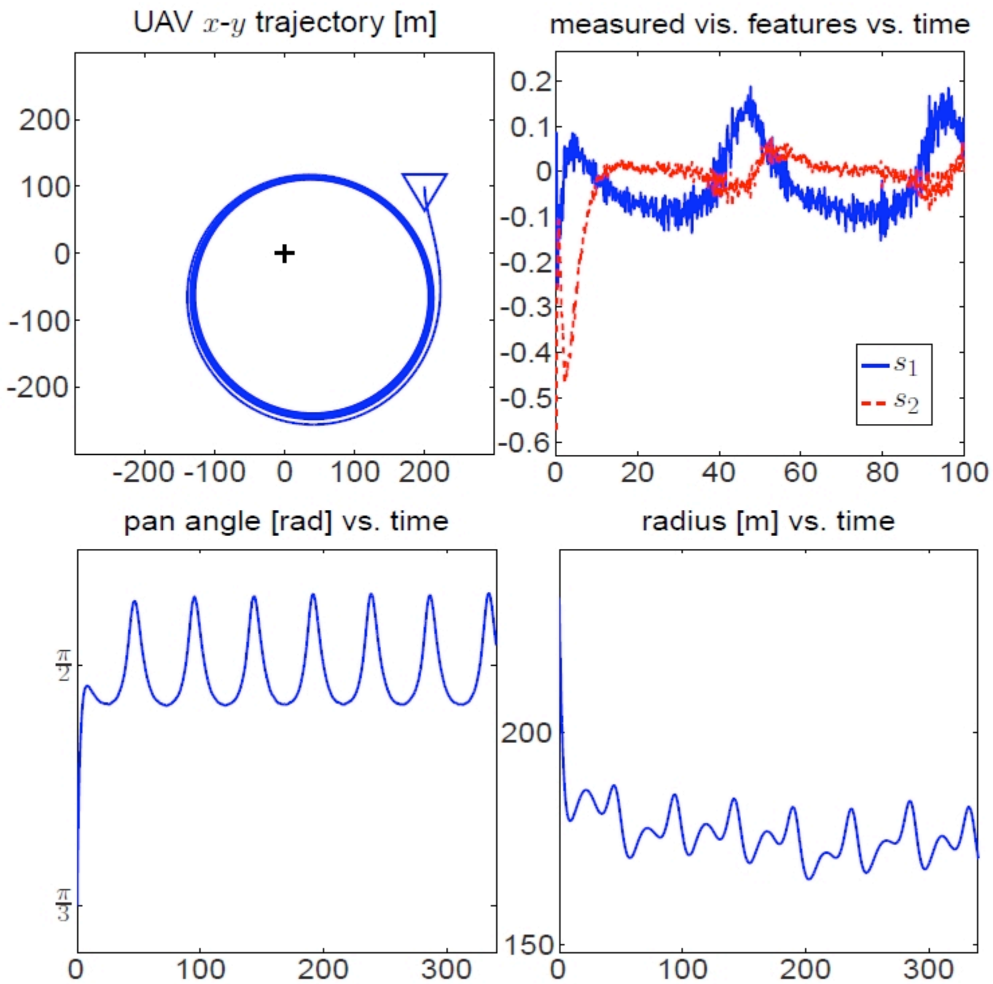
- earth model
- atmosphere model
- aerodynamics effects
- complete aircraft model
- wind disturbances
- camera noise
- pan-tilt noise

simulation on aerosonde - basic



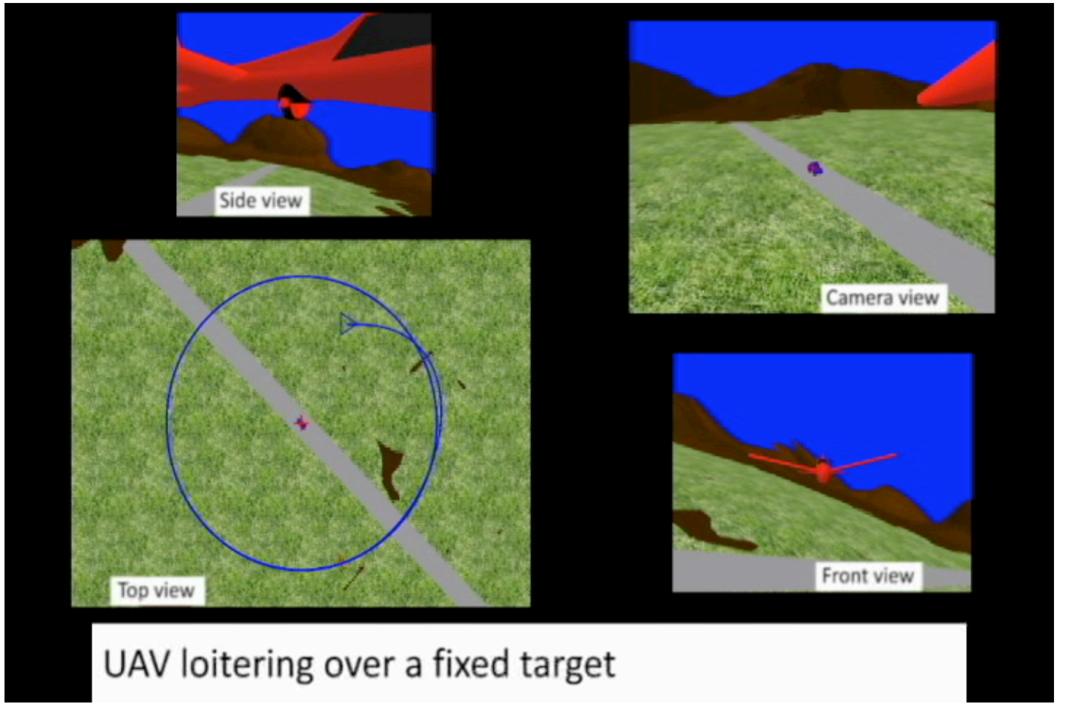
initial conditions: $(x_0, y_0, z_0, \psi_0, \phi_0, \theta_{p0}, \theta_{t0}) = (200, 100, 100, \frac{3}{2}\pi, 0, \frac{\pi}{3}, -\frac{\pi}{4})$

simulation on aerosonde – with noise

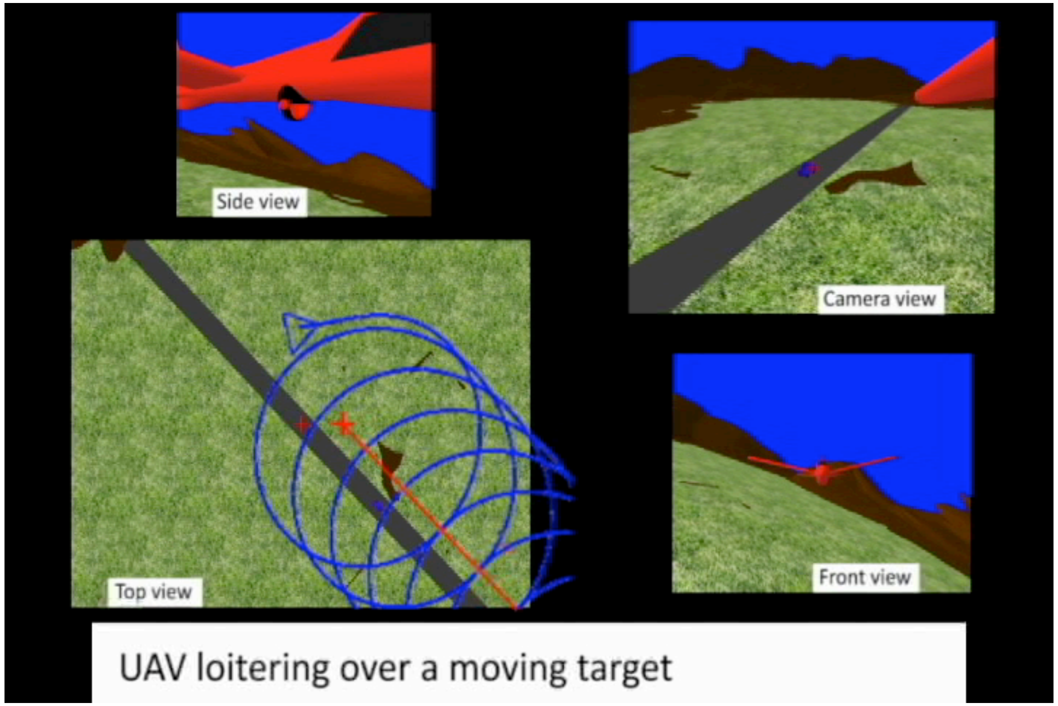


initial conditions: $(x_0, y_0, z_0, \psi_0, \phi_0, \theta_{p0}, \theta_{t0}) = (200, 100, 100, \frac{3}{2}\pi, 0, \frac{\pi}{3}, -\frac{\pi}{4})$

video – basic simulation



video – simulation with noise



visual servoing – quadrotor uav

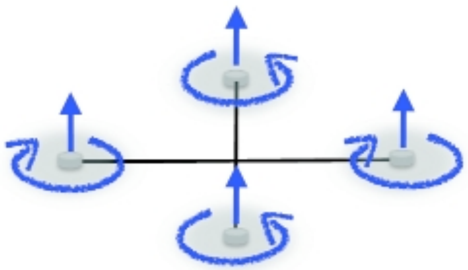
system description

four motor + propeller systems are arranged in a cross-like shape

- each motor spins at a proper angular speed (actual control input)
- each propeller produces a force that is proportional to the square of angular speed

$$F = \omega_i^2$$

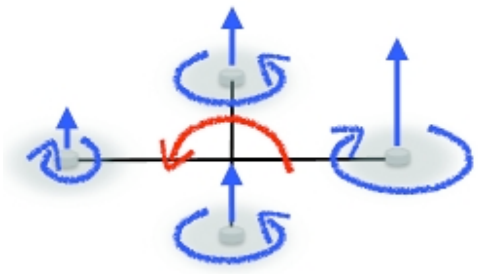
Two groups: two motors are rotating CW and two CCW



collective speed

↓

thrust force



relative speed in the group

↓

torques around x and y

relative speed between the two groups

↓

torque around z

task definition

task (recall):

chosen a target, we want to track a visual point feature (centroid or a characteristic point on the target) in order to perform continuous monitoring by keeping the UAV in flight above it

task (for quadrotor UAV)

regulate the position (x,y) of the UAV (hovering), while keeping the target in the center of image plane

- using a downlooking camera attached below the vehicle
- using a single point feature is not sufficient to control all the degrees of freedom
  - motion along the z_c (z camera coordinate) is unobservable

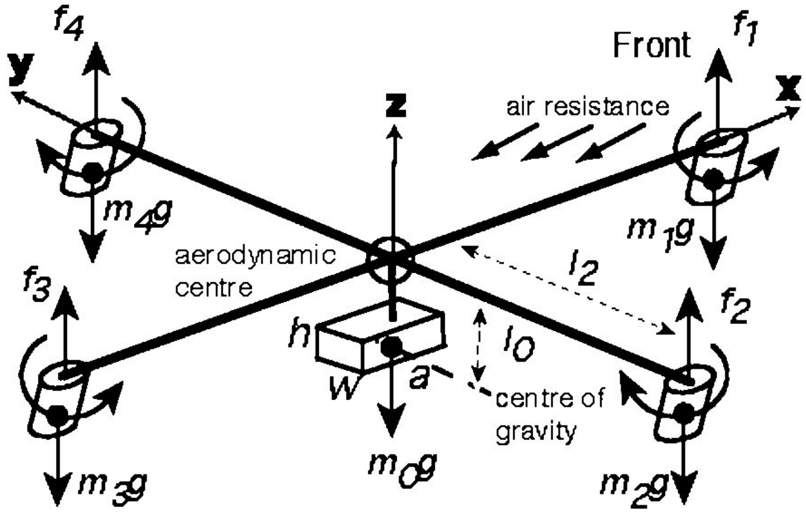
forces/moments diagram

motion is given by

- mechanical components (gravity, inertia, spinning propellers, …)
- aerodynamic effects (blade thrust / drag, blade flapping, …)

the complete model is rather complex

- some components can be estimated by identifying inertial/aerodynamic coefficients
- it is common to have low level control loops to stabilize propeller speed, altitude, attitude
- a simplified model can be used to design control for high level tasks



system modelling for control design

to ease the study (w.l.o.g.) we consider the following simplifying assumptions:

- no wind
- UAV is symmetrical (diagonal inertia matrices)
- secondary inertial/aerodynamic effects are neglected
- self-inducted aerodynamic disturbances are not modelled

Corresponding simplified model of UAV

having remapped the control inputs:

- collective thrust $U_1 = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$
- roll torque $U_2 = b(\omega_4^2 - \omega_2^2)$
- pitch torque $U_3 = b(\omega_1^2 - \omega_3^2)$
- yaw torque $U_4 = d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2)$

$$\Longrightarrow \begin{cases} m\ddot{x} = (c_\psi s_\theta c_\phi + s_\psi s_\phi)U_1 \\ m\ddot{y} = (s_\psi s_\theta c_\phi - s_\phi c_\psi)U_1 \\ m\ddot{z} = mg - (c_\theta c_\phi)U_1 \\ I_x\ddot{\phi} = pr(I_y - I_z) + J_r q\Omega_r + lU_2 \\ I_y\ddot{\theta} = qr(I_z - I_x) - J_r p\Omega_r + lU_3 \\ I_z\ddot{\psi} = pq(I_x - I_y) + J_r\dot{\Omega}_r + U_4 \end{cases}$$

system modelling for control design

also consider the following additional assumptions:

- attitude is stabilized by an high frequency low level controller (commonly available on most systems)
- altitude is separately controlled (the control input U_1 becomes an exogenous signal)
- camera is downlooking, fixed and centered in the UAV c.o.g.
- yaw angle is known and separately controlled

$$\boxed{\begin{aligned} m\ddot{x} &= (c_\psi s_\theta c_\phi + s_\psi s_\phi)U_1 \\ m\ddot{y} &= (s_\psi s_\theta c_\phi - s_\phi c_\psi)U_1 \end{aligned}}$$

$$m\ddot{z} = mg - (c_\theta c_\phi)U_1$$

$$I_x\ddot{\phi} = pr(I_y - I_z) + J_r q\Omega_r + lU_2$$

$$I_y\ddot{\theta} = qr(I_z - I_x) - J_r p\Omega_r + lU_3$$

$$I_z\ddot{\psi} = pq(I_x - I_y) + J_r\dot{\Omega}_r + U_4$$

U_1 is substituted by the <u>measured thrust</u> T

$\longrightarrow$

Rotation about z axis to compensate ψ

$$\begin{bmatrix} \ddot{x}_\psi \\ \ddot{y}_\psi \\ \ddot{z}_\psi \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}$$

(x_ψ, y_ψ, z_ψ) is the new frame

$$\begin{aligned} \ddot{x}_\psi &= \frac{T}{m}\sin\theta\cos\phi \\ \ddot{y}_\psi &= -\frac{T}{m}\sin\phi \end{aligned}$$

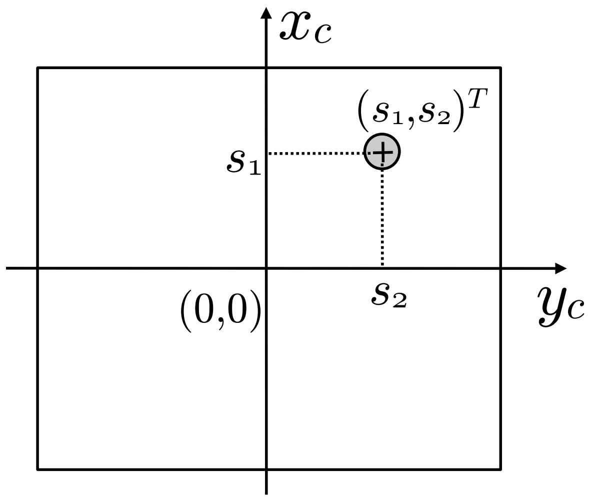
angles ϕ and θ can be considered as new control inputs

$$\phi = u_y$$
$$\theta = u_x$$

control design

features dynamics is given by

$$\dot{s} = \mathbf{J}_i(\mathbf{s}, Z) J_c(\phi, \theta, \psi) \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \mathbf{J}_V V + \mathbf{J}_\Omega \Omega$$



defining the error on the features is

$$e = \begin{bmatrix} e_u \\ e_v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} u \\ v \end{bmatrix}$$

and setting the desired velocities as

$$V_d = J_V^{\#} \left(-K \begin{bmatrix} u \\ v \end{bmatrix} - J_\omega \Omega \right)$$

$\Longrightarrow$

$$\dot{e} = -(J_V V_d + J_\omega \Omega) = -Ke$$

we get
decoupled exponential convergence

contol design

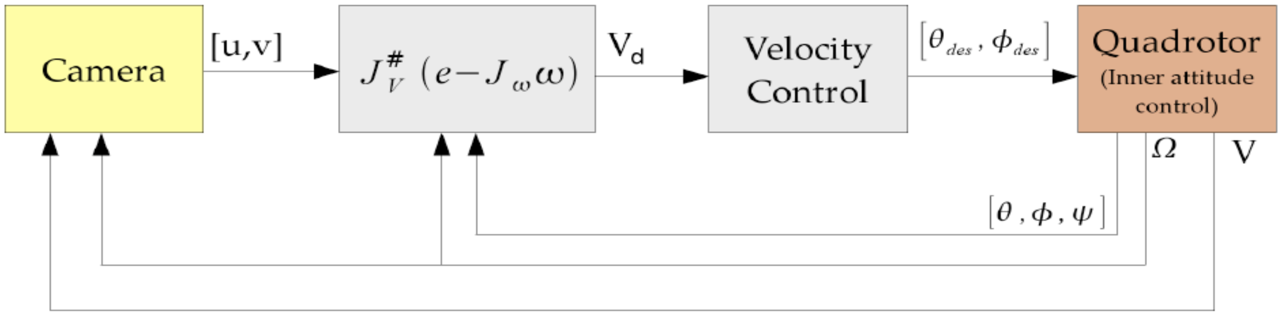
to realize desired velocities, set the inputs as

$$u_x = \frac{T}{m} K_a (V_{dx} - V_x)$$
$$u_y = \frac{T}{m} K_a (V_{dy} - V_y)$$

and, by inverting the equations, we obtain, for the angles

$$\theta_d = \arcsin \left(\frac{\frac{m}{T} K_a (V_{dx} - V_x)}{\cos \phi} \right)$$
$$-\phi_d = \arcsin \left(\frac{m}{T} K_a (V_{dy} - V_y) \right)$$

resulting control scheme



contol design

NOTE: to calculate desired angles we need the actual velocities V_x , V_y of the vehicle

$$\begin{aligned}\theta_d &= \arcsin\left(\frac{\frac{m}{T}K_a(V_{dx}-V_x)}{\cos\phi}\right)\\ -\phi_d &= \arcsin\left(\frac{m}{T}K_a(V_{dy}-V_y)\right)\end{aligned}$$

one possible way is given by the inversion of feature dynamics

$$\hat{V} = J_V^{\#}\left(\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} - J_\omega \Omega\right)$$

advantages

- only feature coordinates (besides attitude) are needed for computation
- easy implementation / low computational cost

disadvantages

- not so accurate (numerical derivation of feature coordinates)

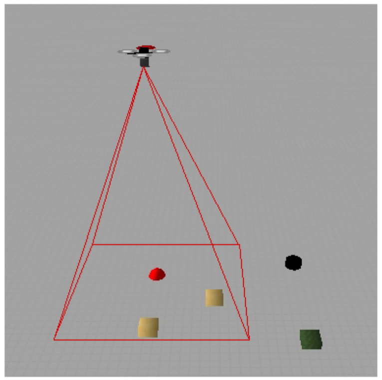
feature extraction – real implementation

feature extraction can be performed in many ways (requires color/shape segmentation, calculus of image moments, …)

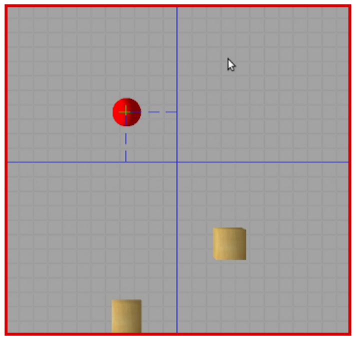
two implementations have been explored

- Camshift (provided by opencv libraries)
  - colour based (any shape)
  - robust to occlusions (even partial)
  - suffers light changes

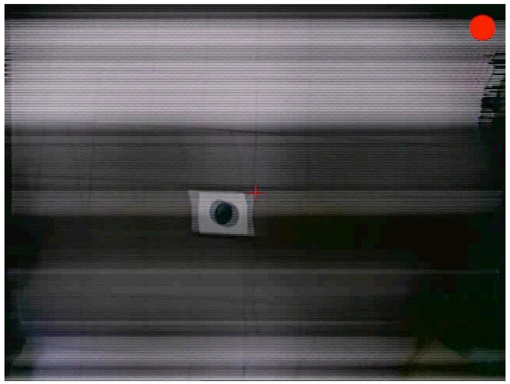
- circular target tracking (provided by VISP project)
  - shape and color based
  - can recover the target after occlusions



Simulated environment



Simulated image (without noise)



real camera - wireless (very noisy images)

real robot – the HummingBird

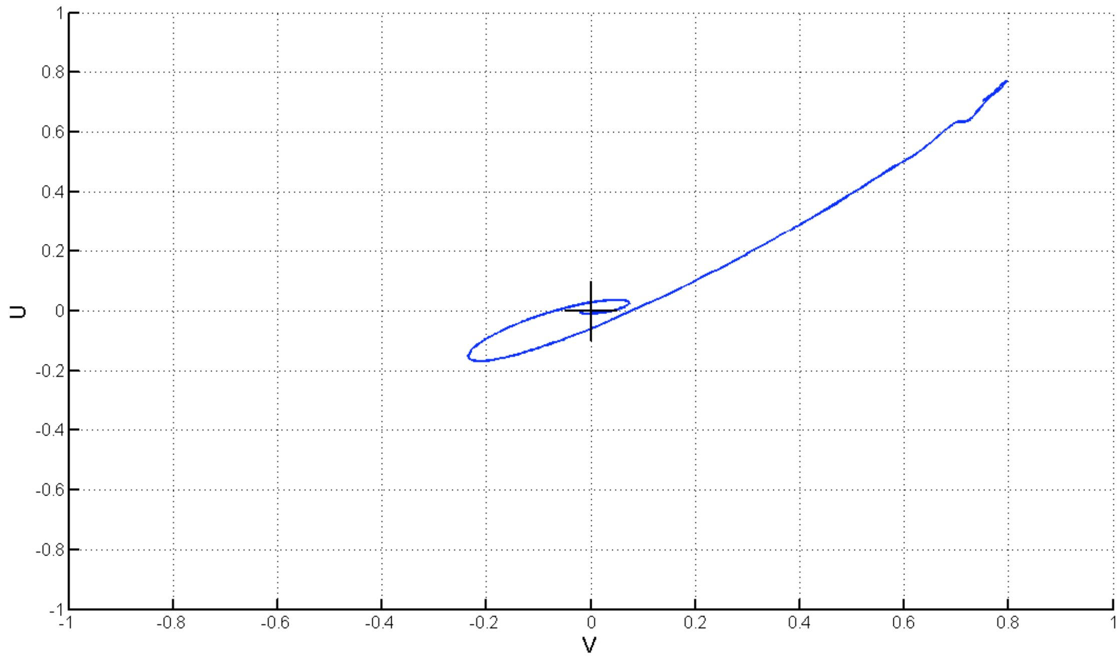
two ARM 7 processors

- low level control:
  - actual implementation of attitude control
  - gathering data from sensors (e.g. IMU for attitude)
- high level control
  - actual implementation of altitude control
  - manages communication with remote station (pc)
- wireless camera added (not really centered in the UAV c.o.g.)
- not enough computational power to perform image analysis onboard (a remote station receives images and provides angles references)

dynamic engine simulation

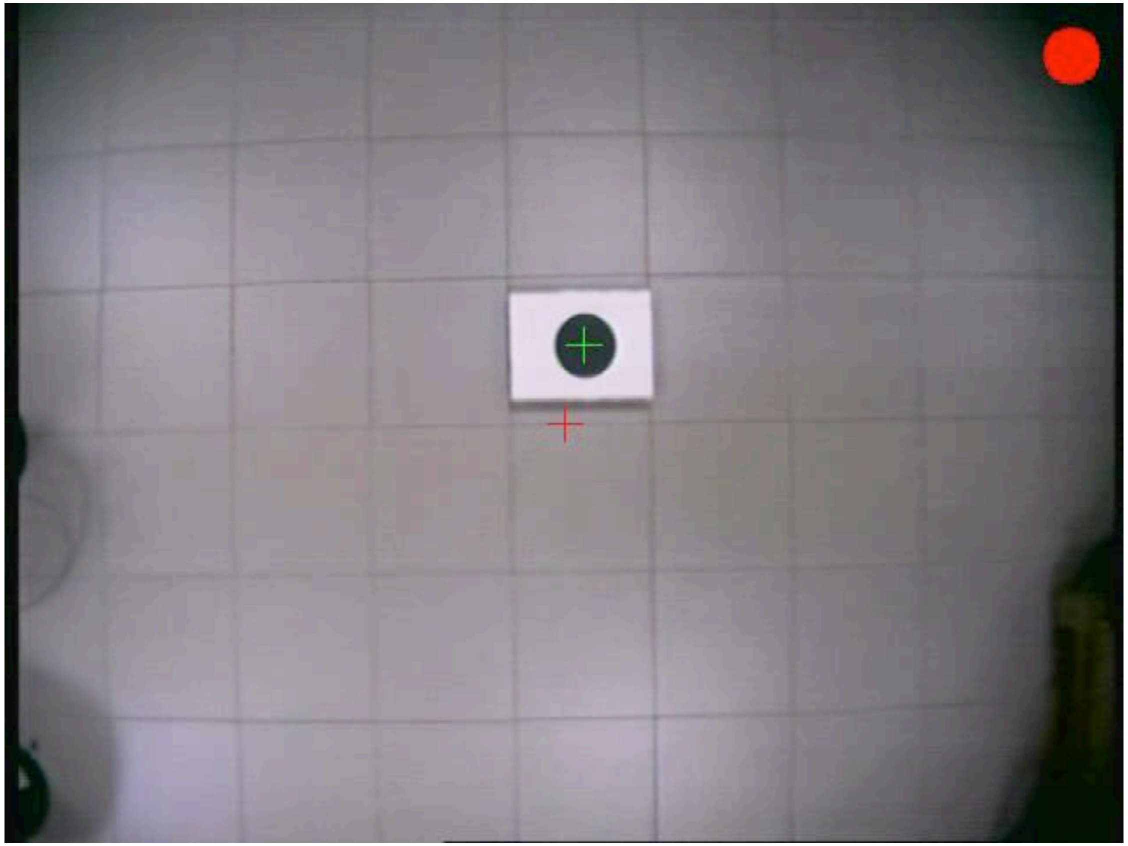
Gazebo simulator

- physics dynamic engine used for quadrotor model
- provides 3D visualization and camera simulation
- actual implementation of control and feature extraction algorithms (exactly the same will be used in experiments)



smooth convergence of target centroid
(on image plane)

Experiments – pure hovering



experiments – robot following