Autonomous and Mobile Robotics

Visual Servoing for Unmanned Aerial Vehicles

(prepared by Lorenzo Rosa)
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 $\rightarrow$ 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 surveillance, 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

\[ L = C_L \frac{\rho}{2} A \cdot V^2 \]

Moments
- \( L \) : Roll moment
- \( M \) : Pitch moment
- \( N \) : Yaw moment

Forces
- \( L \) : Lift
- \( D \) : Drag
- \( Y \) : Sideslip force
- \( T \) : Thrust
- \( G \) : Weight

Angles
- \( \alpha \) : Angle of attack
- \( \beta \) : Sideslip angle
- \( \epsilon \) : Thrust-vector angle
**system modeling for control design**

generalized coordinates (UAV)  

- cartesian coordinates \((x, y, z)\) of the origin of \(F_b\) in w.r.t. inertial frame \(F\)
- orientation \((\psi, \theta, \phi)\) of \(F_b\) w.r.t. \(F\)

generalized coordinates (camera)  

camera pan \(\theta_p\) and tilt \(\theta_t\) 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 constant (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

control inputs:

- roll rate $u_\phi$
- pan rate $u_p$
- tilt rate $u_t$

\[
\begin{align*}
\dot{x} &= v \cos \psi \\
\dot{y} &= v \sin \psi \\
\dot{\psi} &= -\frac{g}{v} \tan \phi \\
\dot{\phi} &= u_\phi \\
\dot{\theta}_p &= u_p \\
\dot{\theta}_t &= u_t
\end{align*}
\]
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 \(J_i(s, Z)\)
\[
\dot{s} = J_i(s, Z) \begin{pmatrix} v_c \\ \omega_c \end{pmatrix}
\]
by using the camera “jacobian” \(J_c(\psi, \phi, \theta_p, \theta_t)\), we can finally relate
features motion to UAV + pan-tilt system

\[
\dot{s} = J_i(s, Z)J_c(\phi, \theta, \psi)
\]

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 (it can be proven), 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 \( \phi = \text{const} \) 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

$$G(\psi) = \begin{pmatrix} \cos \psi & O_{2 \times 4} \\ \sin \psi & \mathbf{O}_{4 \times 1} \\ \mathbf{O}_{4 \times 1} & \mathbf{I}_{4 \times 4} \end{pmatrix}$$

the features dynamics become

$$\dot{s} = J_i J_c G \begin{pmatrix} v \\ \dot{\psi} \\ \dot{\phi} \\ \dot{\theta}_p \\ \dot{\theta}_t \end{pmatrix} = J \begin{pmatrix} v \\ \dot{\psi} \\ \dot{\phi} \\ \dot{\theta}_p \\ \dot{\theta}_t \end{pmatrix} = J_{1-2} \begin{pmatrix} v \\ \psi \end{pmatrix} + J_{3-5} \begin{pmatrix} u_\phi \\ u_p \\ u_t \end{pmatrix}$$

while the dynamics of the output variables are:

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

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

\[
\begin{align*}
\dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\
\dot{x}_2 &= f_2(x_1, x_2) + g_2(x_1, x_2)u
\end{align*}
\]

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: \( x_2 \rightarrow \alpha(x_1) \Rightarrow e_1 \rightarrow 0 \)
modified system - model

assuming that a direct control of the yaw rate is available (by means of the virtual input $u_\psi$) and consider $u_\phi$ as an exogenous signal

\[
\begin{align*}
\dot{x} &= v \cos \psi \\
\dot{y} &= v \sin \psi \\
\dot{\psi} &= u_\psi \\
\dot{\phi} &= u_\phi \\
\dot{\theta}_p &= u_p \\
\dot{\theta}_t &= u_t.
\end{align*}
\]

we get the following modified dynamics:

\[
\begin{pmatrix}
\dot{s} \\
\dot{\theta}_p
\end{pmatrix} =
\begin{pmatrix}
J_1 & J_3 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\dot{\phi}
\end{pmatrix} +
\begin{pmatrix}
J_2 & J_4 & J_5 \\
0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
u_\psi \\
u_p \\
u_t
\end{pmatrix} = J_A \begin{pmatrix}
v \\
\dot{\phi}
\end{pmatrix} + J_B \begin{pmatrix}
u_\psi \\
u_p \\
u_t
\end{pmatrix}
\]
modified system - control

to control the modified dynamics:

\[
\begin{pmatrix}
\dot{s} \\
\dot{\theta}_p
\end{pmatrix} =
\begin{pmatrix}
J_1 & J_3 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
v \\
\dot{\phi}
\end{pmatrix} +
\begin{pmatrix}
J_2 & J_4 & J_5 \\
0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
u_p \\
\dot{u}_p \\
\dot{u}_t
\end{pmatrix} = J_A \begin{pmatrix} v \end{pmatrix} + J_B \begin{pmatrix} u \end{pmatrix}
\]

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

\[
\begin{pmatrix}
u_p \\
\dot{u}_p \\
\dot{u}_t
\end{pmatrix} = -J_B^{-1} \left(Ke + J_A \begin{pmatrix} \dot{v} \end{pmatrix} \right)
\]

we get decoupled exponential convergence for the error vector \(e = (s, \theta_p - \pi/2)^T\):

\[
\dot{e} = -Ke
\]
comparison of original and modified dynamics

original system

\[
\begin{align*}
\dot{x} & = v \cos \psi \\
\dot{y} & = v \sin \psi \\
\dot{\psi} & = -\frac{g}{v} \tan \phi \\
\dot{\phi} & = u_{\phi} \\
\dot{\theta}_p & = u_p \\
\dot{\theta}_t & = u_t
\end{align*}
\]

modified system (with stabilizing control)

\[
\begin{align*}
\dot{x} & = v \cos \psi \\
\dot{y} & = v \sin \psi \\
\dot{\psi} & = u_{\psi} \\
\dot{\phi} & = u_{\phi} \\
\dot{\theta}_p & = u_p \\
\dot{\theta}_t & = u_t
\end{align*}
\]

\[
\dot{e} = \begin{pmatrix}
\dot{\psi} \\
\dot{\theta}_p
\end{pmatrix} = \begin{pmatrix}
J_{1-2} & J_{3-5}
0 & 0
\end{pmatrix} \begin{pmatrix}
v \\
\psi
\end{pmatrix} + \begin{pmatrix}
J_2 & J_4 & J_5
0 & 1 & 0
\end{pmatrix} \begin{pmatrix}
u_{\phi} \\
u_p \\
u_t
\end{pmatrix}
\]

\[
\begin{pmatrix}
u_{\psi} \\
u_p \\
u_t
\end{pmatrix} = -J_B^{-1} \left( Ke + J_A \begin{pmatrix}
v \\
\dot{\phi}
\end{pmatrix} \right) \quad \Rightarrow \quad \dot{e} = -Ke
\]
backstepping to the original system

considering the original system

\[
\dot{e} = \begin{pmatrix}
\dot{s} \\
\dot{\theta}_p \\
\dot{\phi}
\end{pmatrix} = J_A \begin{pmatrix}
v \\
\dot{\phi}
\end{pmatrix} + J_B \begin{pmatrix}
-\frac{g}{v} \tan \phi \\
u_p \\
u_t
\end{pmatrix}
\]

adding and subtracting \( J_B \begin{pmatrix}
\upsilon_{\psi} \\
u_p \\
u_t
\end{pmatrix}^T \) we get

\[
\dot{e} = J_A \begin{pmatrix}
v \\
\dot{\phi}
\end{pmatrix} + J_B \begin{pmatrix}
\upsilon_{\psi} \\
u_p \\
u_t
\end{pmatrix} + J_B \begin{pmatrix}
\xi \\
0 \\
0
\end{pmatrix} \quad \Rightarrow \quad \xi = -\frac{g}{v} \tan \phi - \upsilon_{\psi}
\]

residual term due to mismatch between virtual and actual control

and the error dynamics will be modified by the residual dynamics

\[
\dot{\xi} = -\frac{g}{v \cos^2 \phi} \dot{\phi} - \upsilon_{\psi} = -\frac{g}{v \cos^2 \phi} u_{\phi} - \upsilon_{\psi} = \boxed{u} 
\]

auxiliary input depending on \( u_{\phi} \)

\[
\Rightarrow \quad \dot{e} = -Ke + \xi J_{B,1}
\]
backstepping to the original system

by setting the auxiliary input as

\[ w = -e^T J_{B,1} - k_\xi \xi, \quad k_\xi > 0 \]

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

(it can be proven by using Lyapunov function)

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

\[
\begin{align*}
\dot{x} &= v \cos \psi \\
\dot{y} &= v \sin \psi \\
\dot{\psi} &= -\frac{g}{v} \tan \phi \\
\dot{\phi} &= u_\phi \\
\dot{\theta}_p &= u_p \\
\dot{\theta}_t &= u_t
\end{align*}
\]

\[ u_\phi = \frac{v}{g} \cos^2 \phi (e^T J_{B,1} + k_\xi \xi - \hat{u}_\psi) \]

calculated before to stabilize the modified system
**basic result**

![UAV x-y trajectory [m]](image1)

![visual features vs. time](image2)

![pan angle [rad] vs. time](image3)

![ξ [rad/s] vs. time](image4)

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 $\omega = \frac{v}{R}$
- 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

\[
\begin{pmatrix}
    u_p \\
    u_p \\
    u_t
\end{pmatrix}
= -\left( J^{-2}_B \right)^\dagger \left( K_s s + J^{-2}_A \left( \frac{v}{\phi} \right) \right) + P \begin{pmatrix}
    \dot{\psi}_d \\
    k_p \left( \frac{\pi}{2} - \theta_p \right) \\
    0
\end{pmatrix}
\]

- **enforce desired angular velocity**
- **control for feature tracking**
- **projector in the null space of the main task**

\[
P = (I - \left( J^{-2}_B \right)^\dagger J^{-2}_B)
\]
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
  \[ u_\phi = k_\phi (\bar{\phi} - \phi), \quad k_\phi > 0 \]
- 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: \((x_0, y_0, z_0, \psi_0, \phi_0, \theta_{p0}, \theta_{i0}) = (200, 100, 100, \frac{3}{2}\pi, 0, \frac{\pi}{3}, -\frac{\pi}{4})\)
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  \( U_2 = b(\omega_1^2 - \omega_2^2) \)
- pitch torque \( U_3 = b(\omega_1^2 - \omega_3^2) \)
- yaw torque \( U_4 = d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \)

\[
\begin{align*}
mx' &= (c_\psi s_\theta c_\phi + s_\psi s_\phi)U_1 \\
my' &= (s_\psi s_\theta c_\phi - s_\phi c_\psi)U_1 \\
mz' &= mg - (c_\theta c_\phi)U_1 \\
I_x \ddot{\phi} &= pr(I_y - I_z) + Jrq\Omega_r + lU_2 \\
I_y \ddot{\theta} &= qr(I_z - I_x) - Jrp\Omega_r + lU_3 \\
I_z \ddot{\psi} &= pq(I_x - I_y) + Jr\dot{\Omega}_r + U_4 \\
\end{align*}
\]
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

\[
\begin{align*}
    m\ddot{x} &= (c_\psi s_\theta c_\phi + s_\psi s_\phi)U_1 \\
    m\ddot{y} &= (s_\psi s_\theta c_\phi - s_\phi c_\psi)U_1 \\
    m\ddot{z} &= mg - (c_\theta c_\phi)U_1 \\
    I_x\ddot{\phi} &= pr(I_y - I_z) + J_r q_\Omega_x + lU_2 \\
    I_y\ddot{\theta} &= qr(I_z - I_x) - J_r p_\Omega_x + lU_3 \\
    I_z\ddot{\psi} &= pq(I_x - I_y) + J_r \dot{\Omega}_r + U_4
\end{align*}
\]

$U_1$ is substituted by the measured thrust $T$

\[
\begin{align*}
    \dot{x}_\psi &= \frac{T}{m} \sin \theta \cos \phi \\
    \dot{y}_\psi &= -\frac{T}{m} \sin \phi
\end{align*}
\]

angles $\phi$ and $\theta$ can be considered as new control inputs

\[
\begin{align*}
    \phi &= u_y \\
    \theta &= u_x
\end{align*}
\]

Rotation about $z$ axis to compensate $\psi$

\[
\begin{bmatrix}
    \dot{x}_\psi \\
    \dot{y}_\psi \\
    \dot{z}_\psi
\end{bmatrix} = \begin{bmatrix}
    \cos \psi & \sin \psi & 0 \\
    -\sin \psi & \cos \psi & 0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    \dot{x} \\
    \dot{y} \\
    \dot{z}
\end{bmatrix}
\]

$(x_\psi, y_\psi, z_\psi)$ is the new frame
**control design**

features dynamics is given by

$$\hat{s} = J_i(s, Z)J_c(\phi, \theta, \psi) \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = J_V V + J_\Omega \Omega$$

defining the error on the features is

$$e = \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} u \\ v \end{bmatrix}$$

and setting the desired velocities as

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

we get decoupled exponential convergence
Control 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

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

resulting control scheme
**control design**

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

\[
\theta_d = \arcsin \left(\frac{m}{I} K_a (V_{dx} - V_x) \right)
\]
\[
-\phi_d = \arcsin \left(\frac{m}{I} K_a (V_{dy} - V_y) \right)
\]

one possible way is given by the inversion of feature dynamics

\[
\hat{V} = J_V^\# \begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix} - J_\omega \Omega
\]

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
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)

![Graph showing smooth convergence of target centroid](image)

*smooth convergence of target centroid (on image plane)*
Experiments – pure hovering
experiments – robot following