Visual Control for Unmanned Aerial Vehicles (UAVs)

by

Pascual Campoy

Computer Vision Group

www.vision4uav.eu

Centro de Automática y Robótica

Universidad Politécnica de Madrid

The U.A.V.s

Gas powered helicopter

Electric powered helicopter

Multirotors

Some applications

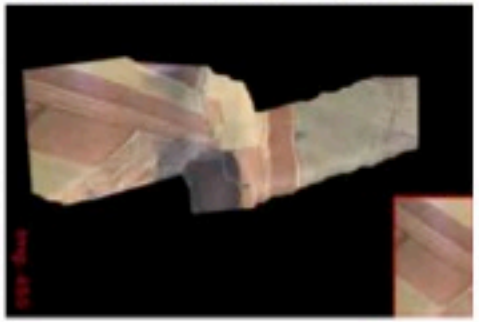
Inspección de volcanes

Inspección de instalaciones

Estabilización imágenes

Mosaicos virtuales

Gestión información visual



Visual Control

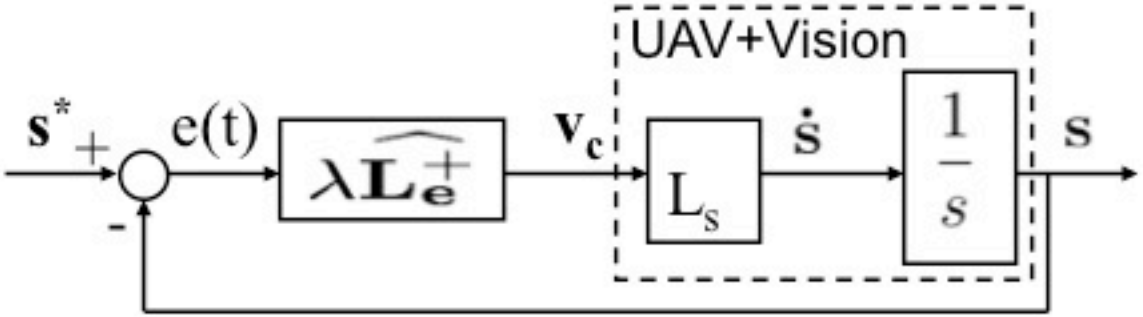
u vector of actuators

s vector of the visual features (2D or 3D)

table of contents

- Visual Control: structure and control law
- Image Based Visual Control
- Pose Based Visual Control
  - Ground camera
  - On-board camera
- Hybrid and Partitioned Visual Control

Model and control structure



$\boldsymbol{v}_c$ is the 6D vector of velocities

$\boldsymbol{s}$ is the KD vector of visual features

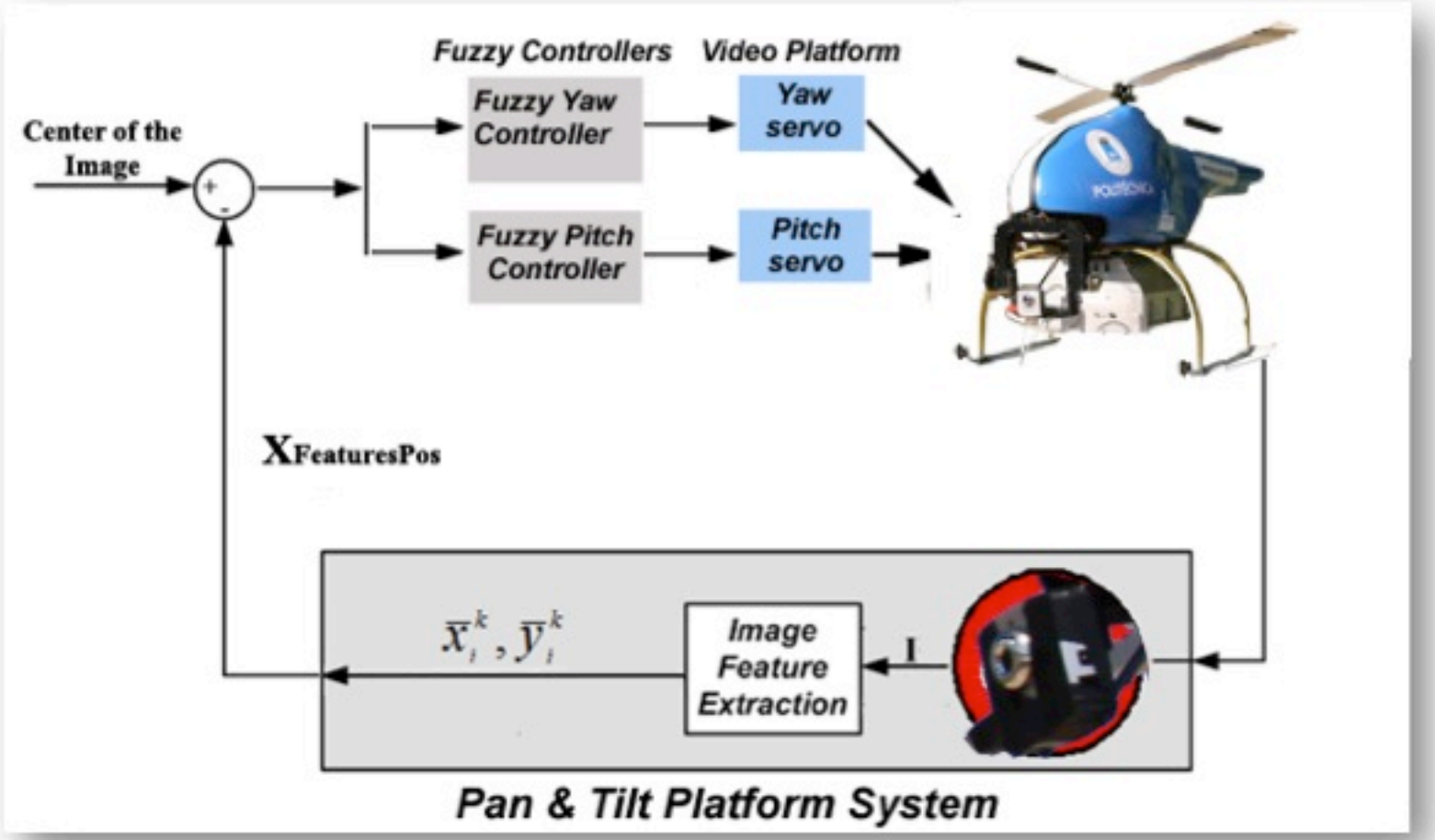
$\mathbf{L_s} \in \Re^{kx6}$ interaction matrix or feature Jacobian

$$\text{if } \dot{s^\star} = 0 \Rightarrow \boxed{\dot{e} = -\lambda \mathbf{L_s} \widehat{\mathbf{L_e^+}} e}$$

$$\boxed{\text{if } \widehat{\mathbf{L_e^+}} \approx \mathbf{L_s^+} \Rightarrow e \text{ is locally stable (i.e. } e \to 0 \text{ when } e \approx 0)}$$

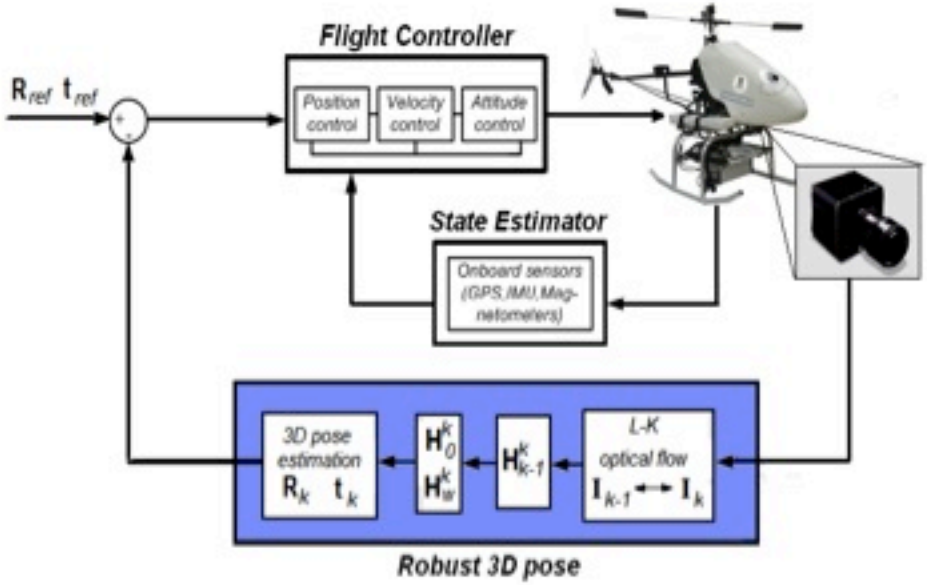
Image based Visual Servoing: IBVS

- Vector ***s*** are visual features in the image plane



Pose based Visual Servoing: PBVS

- Vector ***s*** are pose coordinates in the 3D space



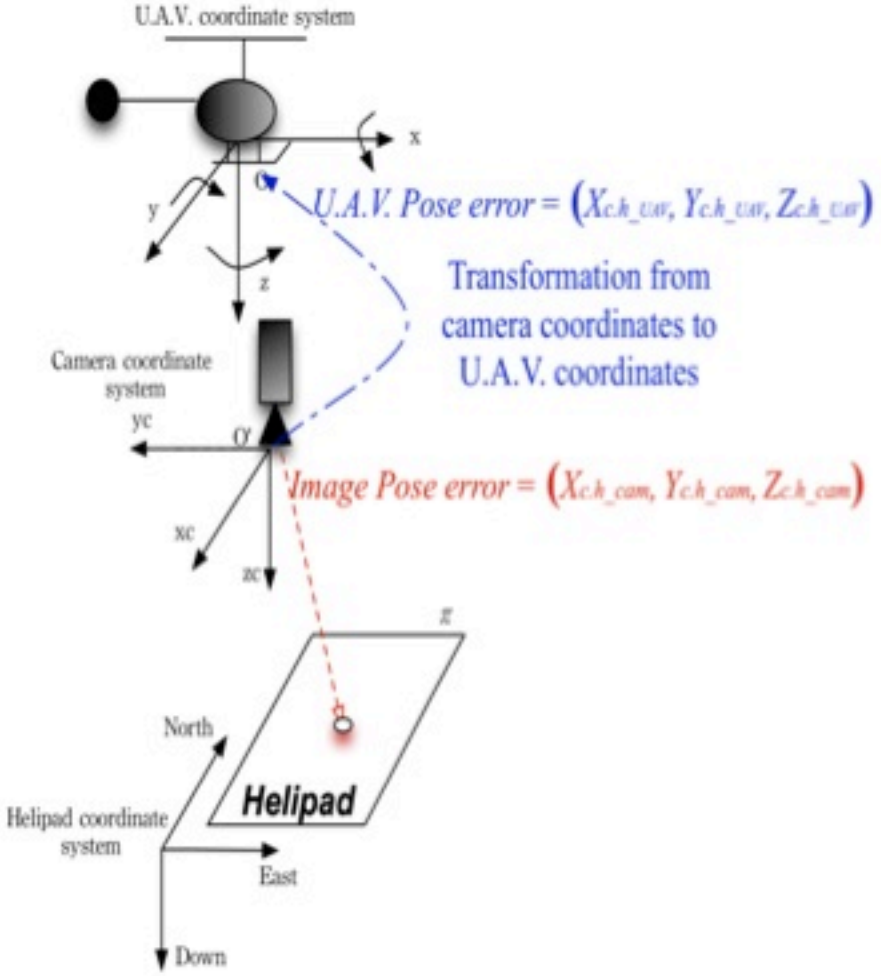
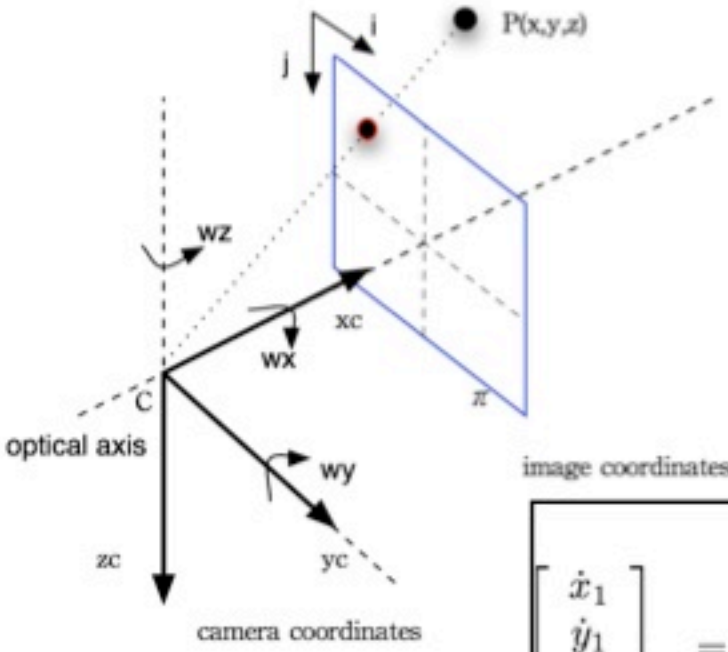


table of contents

- Visual Control: structure and control law
- Image Based Visual Control
  - 2D points' position
  - Other visual features
- Pose Based Visual Control
  - Ground camera
  - On-board camera
- Hybrid and Partitioned Visual Control

I.B.V.S.: 2D points' position ...

$\mathbf{s}=[x_1\ y_1\ \ldots\ x_k\ y_k\ ]^T$ **s** is the position k points in the image



$$\begin{cases} x &= X/Z \\ y &= Y/Z \end{cases} \quad \begin{cases} \dot{x} &= \dot{X}/Z - X\dot{Z}/Z^2 = (\dot{X} - x\dot{Z})/Z \\ \dot{y} &= \dot{Y}/Z - Y\dot{Z}/Z^2 = (\dot{Y} - y\dot{Z})/Z \end{cases}$$

$$\dot{\boldsymbol{X}} = -\boldsymbol{v}_c - \boldsymbol{\omega}_c \times \boldsymbol{X} \Leftrightarrow \begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X \end{cases}$$

image coordinates

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \vdots \end{bmatrix} = \underbrace{\begin{bmatrix} -1/Z_1 & 0 & x_1/Z_1 & x_1 y_1 & (1+x_1^2) & y_1 \\ 0 & -1/Z_1 & y_1/Z_1 & 1+y_1^2 & -x_1 y_1 & -x_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{L_s} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

- more than 3 points are required for good conditioned $\mathbf{L}_\mathbf{s}^+$,

  since 3 points lead to global minima and in some cases to a singular L_s

- Z_i is required for every point in the image

I.B.V.S.: … 2D points' position…

L_s approximation:

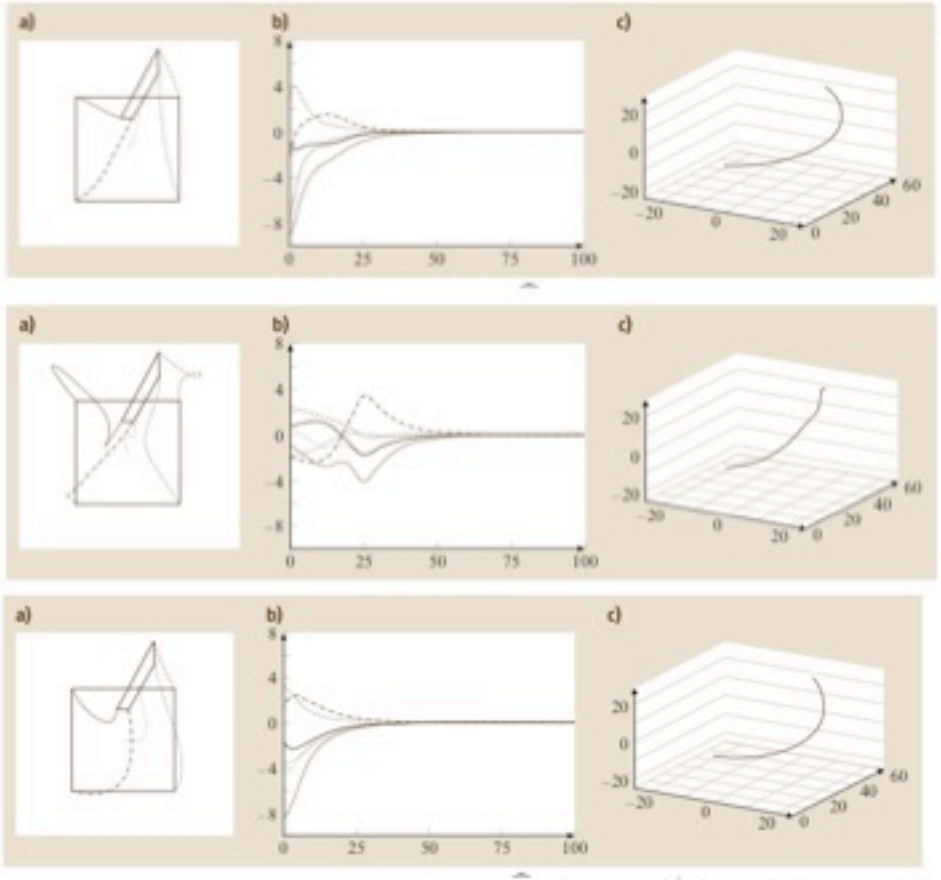
- Z_i is estimated at each iteration

$$\widehat{\mathbf{L}_\mathbf{e}^+} = \mathbf{L}_\mathbf{s}^+$$

- Z_i is used at desired position

$$\widehat{\mathbf{L}_\mathbf{e}^+} = \mathbf{L}_{\mathbf{s}^*}^+ \text{ constant}$$

- $$\widehat{\mathbf{L}_\mathbf{e}^+} = (\mathbf{L}_\mathbf{s}/\mathbf{2} + \mathbf{L}_{\mathbf{s}^*}/\mathbf{2})^+$$

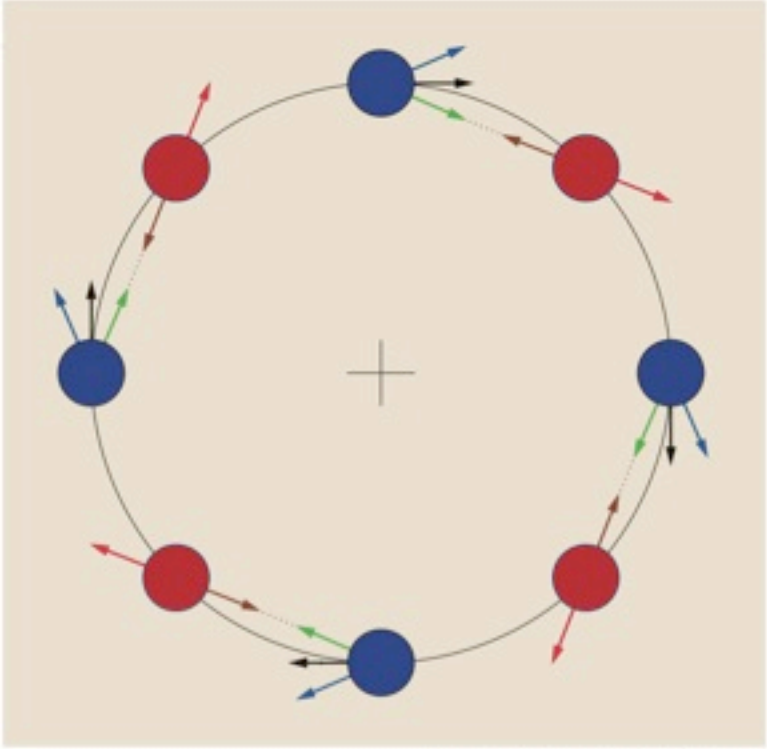


Springer Handbook of Robotics
Siciliano, Khatib (Eds.) ©Springer 2008

I.B.V.S.: ... 2D points' position

Geometrical interpretation:

$$\widehat{\mathbf{L}_\mathbf{e}^+} = \mathbf{L}_\mathbf{s}^+ \qquad \widehat{\mathbf{L}_\mathbf{e}^+} = \mathbf{L}_{\mathbf{s}^*}^+ \text{ constant} \qquad \widehat{\mathbf{L}_\mathbf{e}^+} = (\mathbf{L_s}/\mathbf{2} + \mathbf{L_{s^*}}/\mathbf{2})^+$$



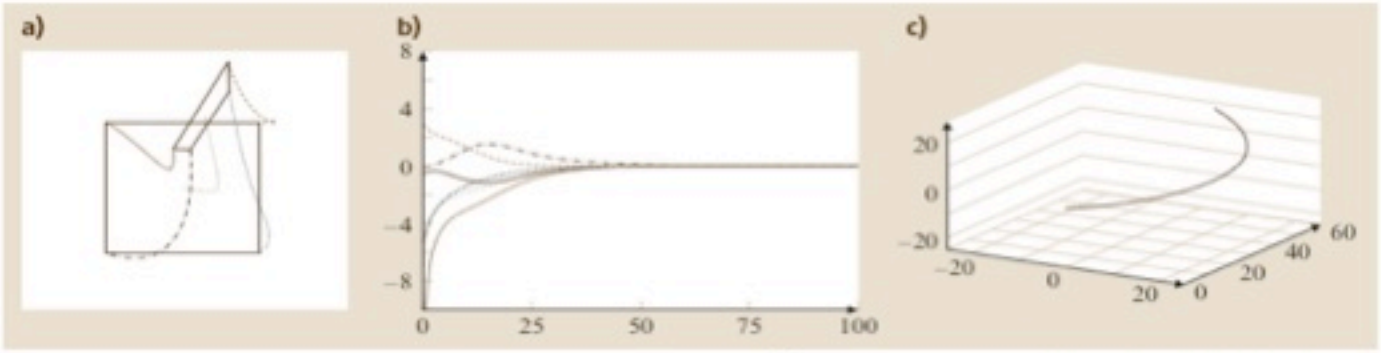
Springer Handbook of Robotics
Siciliano, Khatib (Eds.) ©Springer 2008

I.B.V.S.: 2D polar point's position

$$\mathbf{s} = \begin{bmatrix} \rho_1 & \theta_1 & \dots & \rho_k & \theta_k \end{bmatrix}^T$$

L_s calculation:

$$\begin{bmatrix} \rho_1 \\ \theta_1 \\ \vdots \end{bmatrix} = \underbrace{\begin{bmatrix} -cos\theta_1/Z_1 & -sin\theta_1/Z_1 & \rho_1/Z_1 & (1+\rho_1)^2 sin\theta_1 & -(1+\rho_1)^2 cos\theta_1 & 0 \\ sin\theta_1/(\rho_1 Z_1) & -cos\theta_1/(\rho_1 Z_1) & 0 & cos\theta_1/\rho_1 & sin\theta_1/\rho_1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{L_s} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$



Springer Handbook of Robotics
Siciliano, Khatib (Eds.) - ©Springer 2008

I.B.V.S.: direct matrix estimation

Direct estimation off-line:

- $\mathbf{L_s}\Delta\mathbf{v_c} = \Delta\mathbf{s}$ k equations and kx6 unknown values

  $\mathbf{L_s}\mathbf{A} = \mathbf{B}$ taking N measurements

  $\widehat{\mathbf{L_s}} = \mathbf{B}\mathbf{A}^{+}$ kxN equations

- $\mathbf{L_s^{+}}\Delta\mathbf{s} = \Delta\mathbf{v}_c$ 6 equations and kx6 unknown values

  $\widehat{\mathbf{L_s^{+}}} = \mathbf{A}\mathbf{B}^{+}$ 6xN equations by taking N measurements

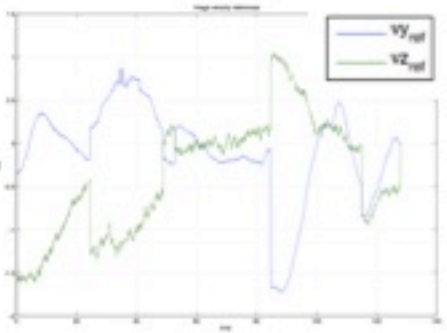
Direct estimation on-line:

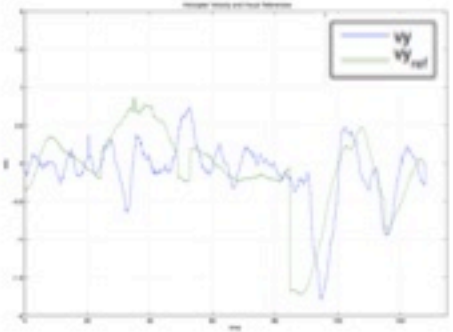
$$\widehat{\mathbf{L_s}}(t+1) = \widehat{\mathbf{L_s}}(t) + \alpha[\Delta\mathbf{s} - \widehat{\mathbf{L_s}}(t)\Delta\mathbf{v}_c]\frac{\Delta\mathbf{v}_c^T}{||\mathbf{v}_c||}$$

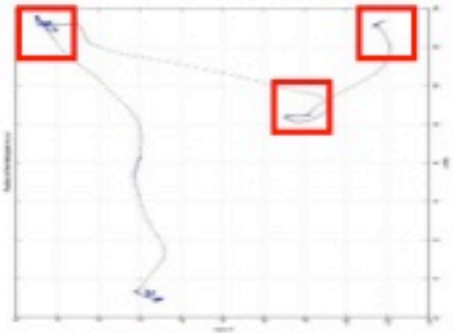
I.B.V.S.: example 1

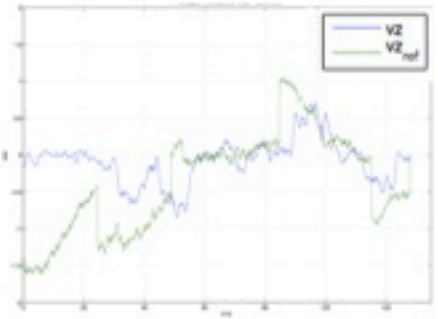
L_s calculation: case $v_z = \omega_x = \omega_y = \omega_z = 0$

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \lambda \underbrace{\begin{bmatrix} -Z_1 & 0 \\ 0 & -Z_1 \end{bmatrix}} \begin{bmatrix} x_1^* - x_1 \\ y_1^* - y_1 \end{bmatrix}$$









Proceedings of the 2006 IEEE International Conference on Robotics and Automation

Orlando, Florida - May 2006

I.B.V.S.: example 2

X-Y UAV control --> ON-BOARD SYSTEM

Object tracking: example 1

planar objective following using pan and tilt plataform

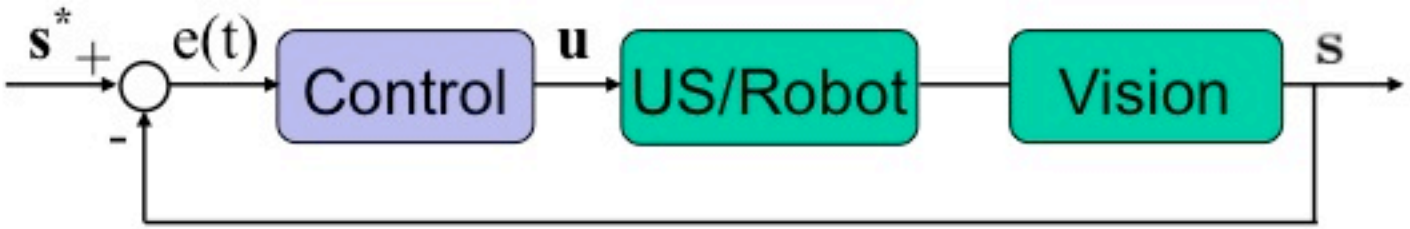
Object tracking: example 2

Object tracking: example 3

table of contents

- Visual Control: structure and control law
- Image Based Visual Control
  - 2D points' position
  - Other visual features
- Pose Based Visual Control
  - Ground camera
  - On-board camera
- Hybrid and Partitioned Visual Control

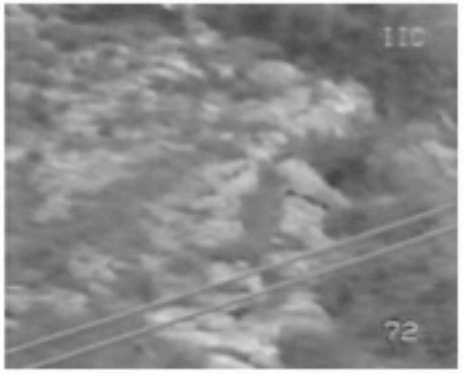
I.B.V.S.: Other visual features



Motivation for other visual features:

1. Other specifications
   (e.g. distance point-line)

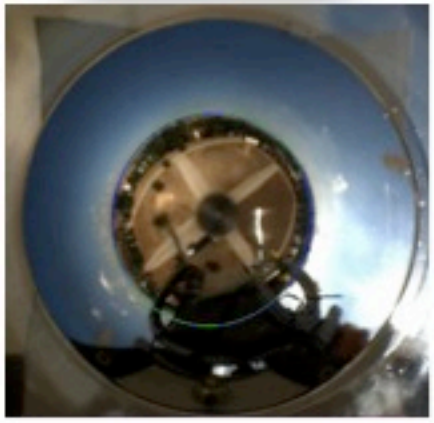
2. Facilitate image processing
   (e.g. lines, contours)



3) Ameliorate the control
   (e.g. linearization, decoupling)

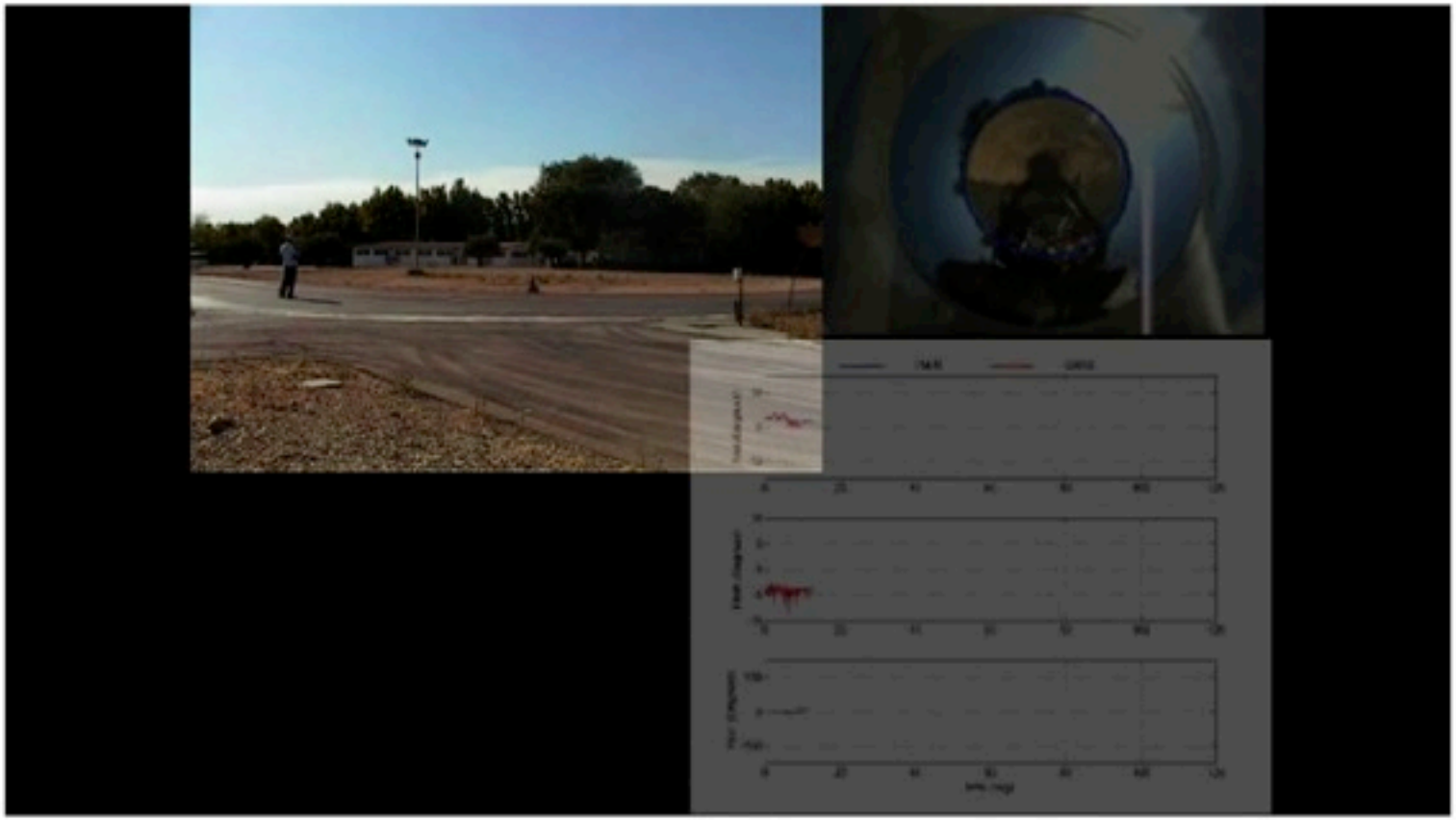
Other visual features: example 1

Catadrioptic camera and features



Other visual features: example 1

Omnivision for hovering



Other visual features: example 2...

The **vision system**:

-Provides information on the distance to the line at each sampling.

-Provides information on the type of section in which you enter.

-Operates at a maximum frequency of 29 Hz, although in a state of emergency may be lower.

Other visual features: example 2

table of contents

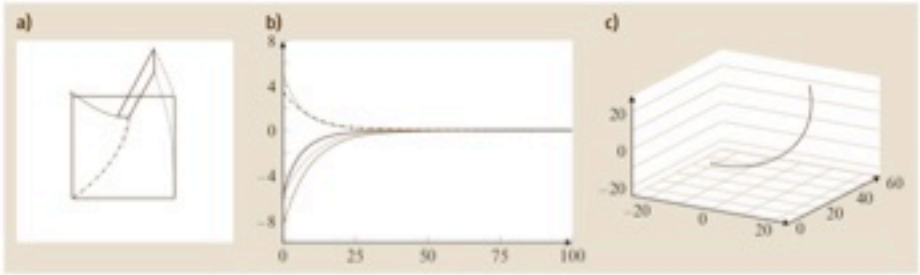
- Visual Control: structure and control law
- Image Based Visual Control
  - 2D points' position
  - Other visual features
- Pose Based Visual Control
  - Ground camera pose estimation
  - On-board camera pose estimation
- Hybrid and Partitioned Visual Control

P.B.V.S.: 6D pose

$$\mathbf{s} = \begin{bmatrix} \mathbf{t} & \theta\mathbf{u} \end{bmatrix}^T$$ where $\mathbf{t}$ is the translation vector and $\theta\mathbf{u}$ is the angle/axis rotation

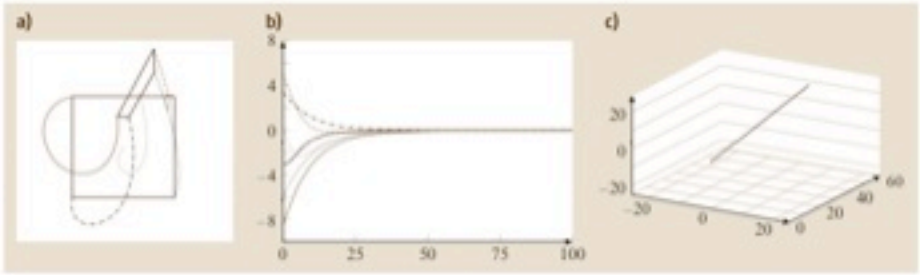
- $\mathbf{s} = \begin{bmatrix} {}^c\mathbf{t}_0 & \theta\mathbf{u} \end{bmatrix}^T \quad \mathbf{s}^* = \begin{bmatrix} {}^{c^*}\mathbf{t}_0 & 0 \end{bmatrix}^T$

$$\begin{cases} \mathbf{v}_c &= -\lambda[{}^{c^*}\mathbf{t}_0 - {}^c\mathbf{t}_0] + [{}^c\mathbf{t}_0]_\times \theta\mathbf{u} \\ \omega_c &= -\lambda\theta\mathbf{u} \end{cases}$$



- $\mathbf{s} = \begin{bmatrix} {}^{c^*}\mathbf{t}_0 & \theta\mathbf{u} \end{bmatrix}^T \quad \mathbf{s}^* = 0$

$$\begin{cases} \mathbf{v}_c &= -\lambda \mathbf{R}^\top {}^{c^*}\mathbf{t}_c \\ \omega_c &= -\lambda\theta\mathbf{u} \end{cases}$$

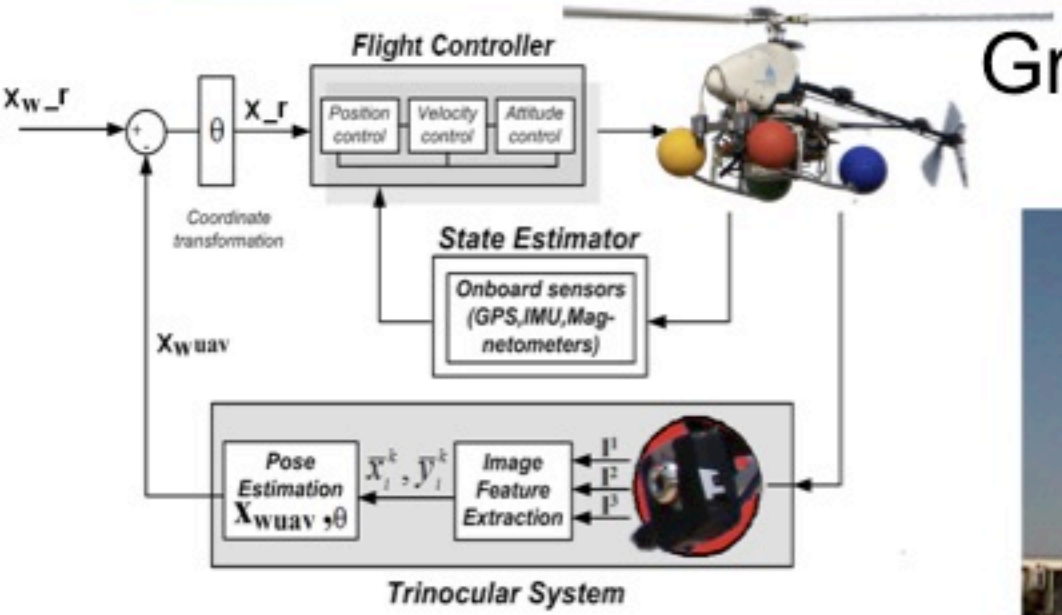


stability condition: $\mathbf{L}_{\widehat{\mathbf{e}}}\widehat{\mathbf{L}^+_{\widehat{\mathbf{e}}}} > 0$

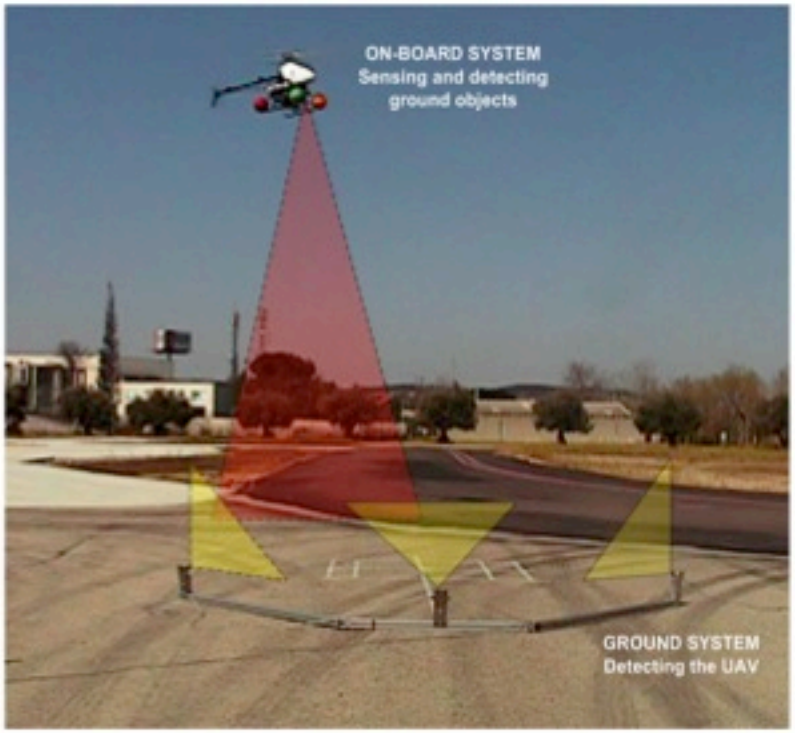
"even small errors in computing the point positions in the image can lead to pose errors that can impact significantly the accuracy and the stabilitty of the system"

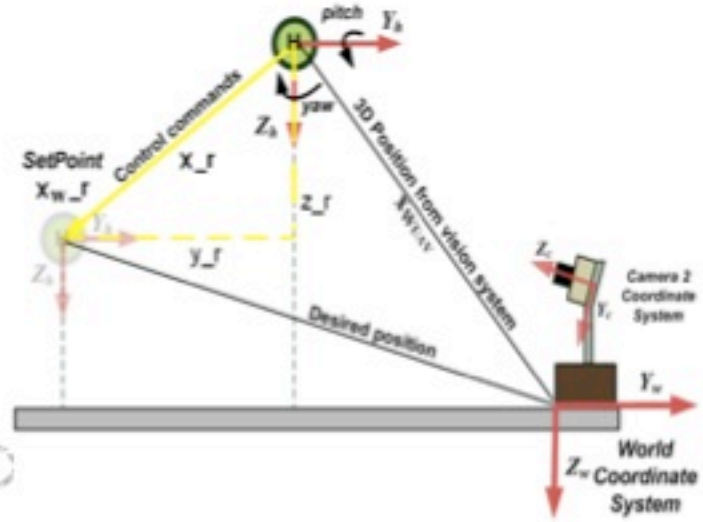
Springer Handbook of Robotics
Siciliano, Khatib (Eds.) · ©Springer 2008

Pose based Visual Servoing



Ground pose estimation





External Pose estimation

Feature Extraction and Tracking

-Detection and tracking of on-board landmarks
- Based on the CamShift algorithm

Camera1

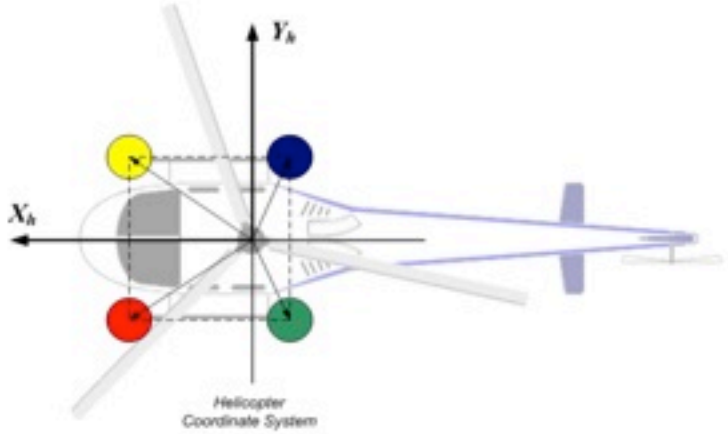
Camera2

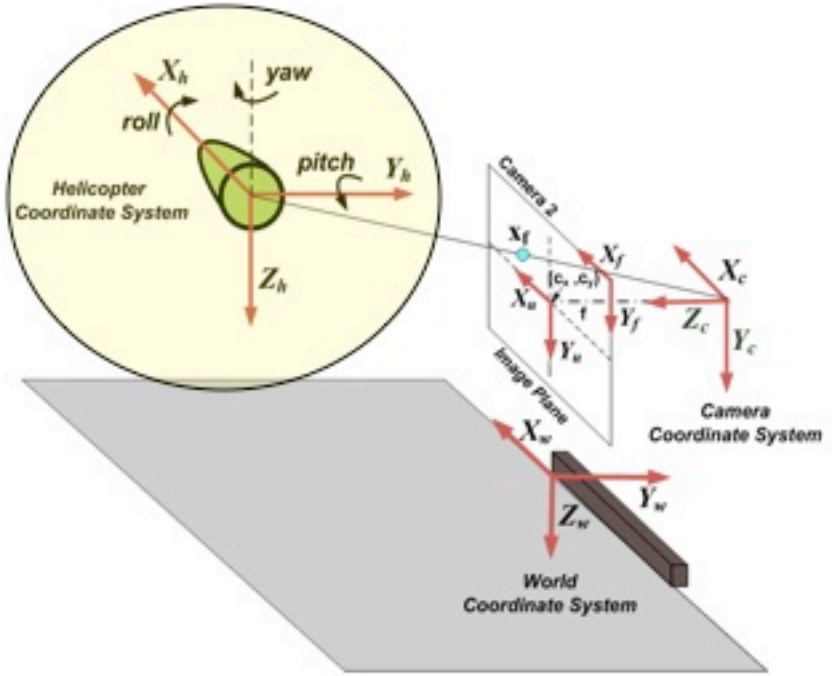
Camera3

External Pose estimation

Pose estimation

-**Intrinsic** parameters of the trinocular system → **Known**
-**Extrinsic** parameters of the trinocular system → **Known**
-**Landmarks'** distribution → **Known**





Estimation

1. Landmarks' **3D** Reconstruction
2. Estimation of **Yaw** and **3D position**
3. approx: **16 fps**

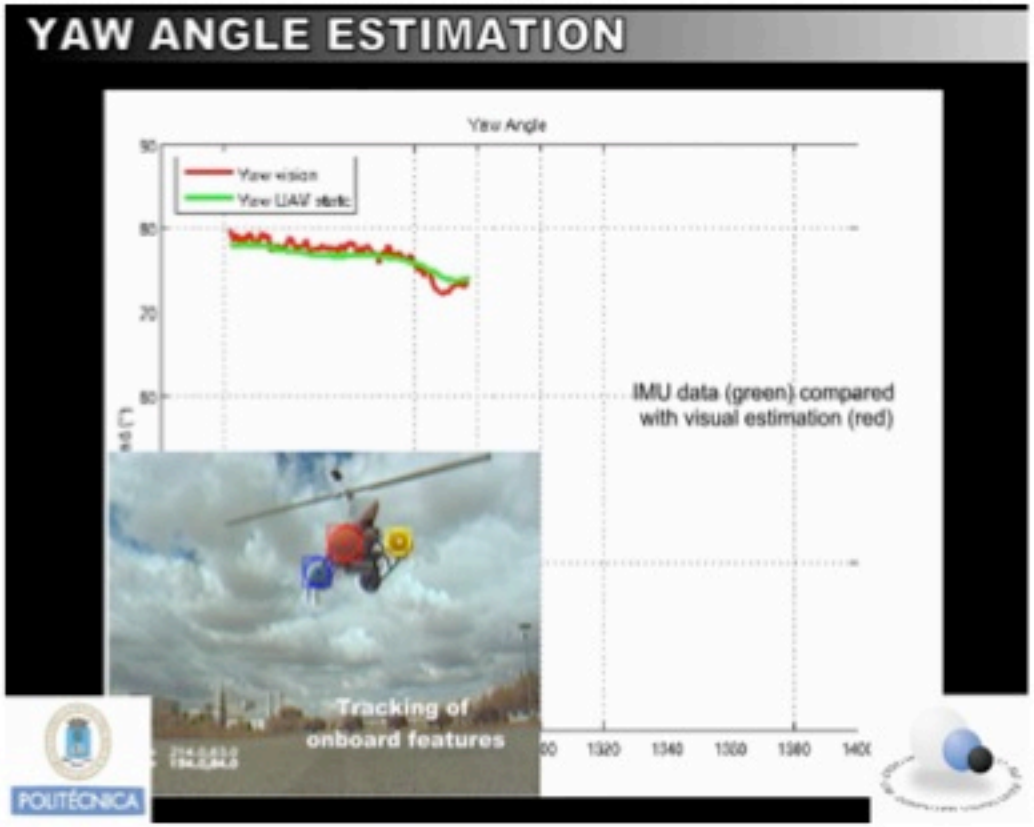
Experiments and Results

Pose Estimation --> TRINOCULAR SYSTEM

1. Different flights in **manual Mode** → for **testing** the pose estimation **algorithm**

Yaw Estimation

* RMSE 8.3º

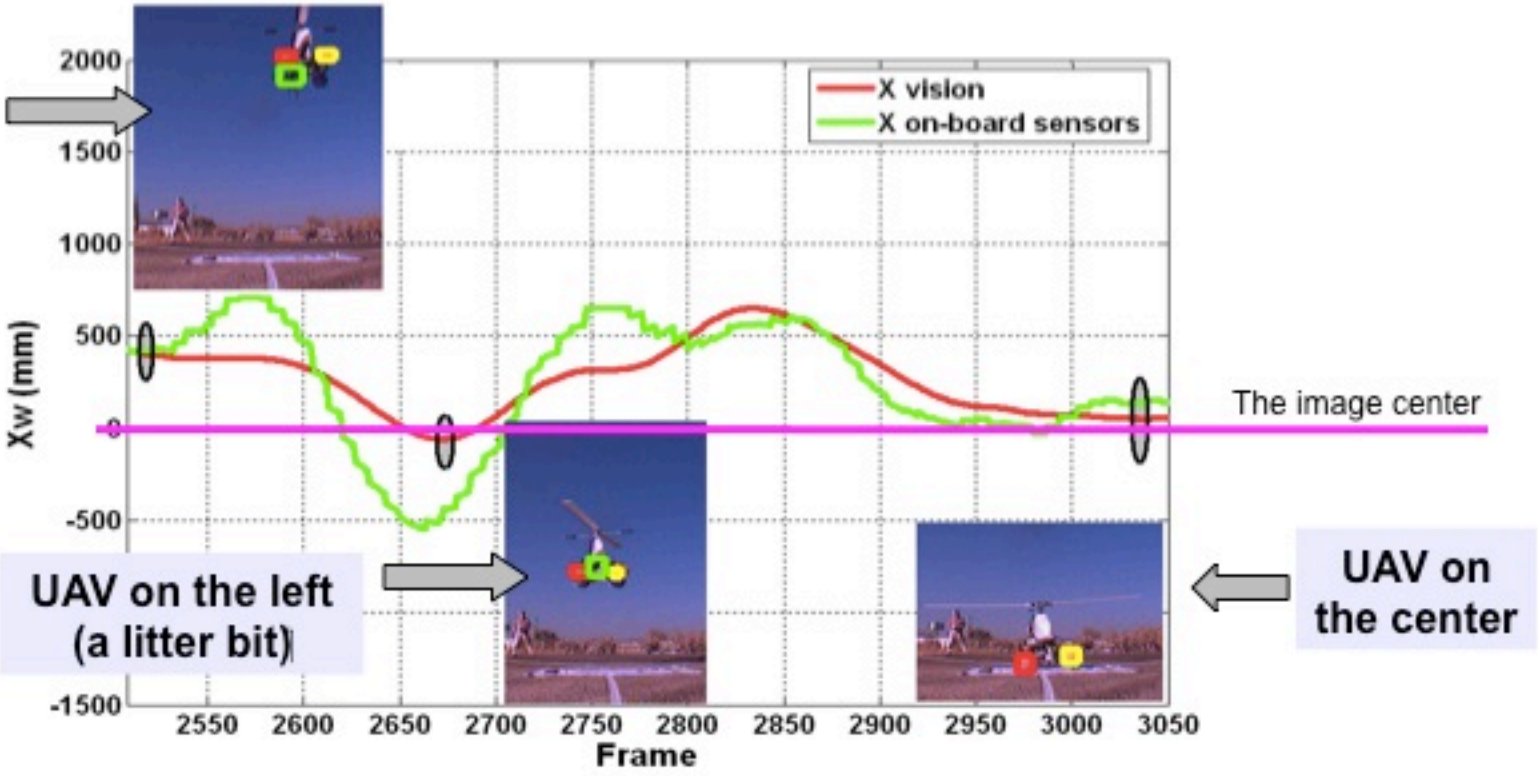


* **Similar behavior** of the **IMU** data and **Vision** data

Experiments and Results

Pose Estimation --> TRINOCULAR SYSTEM

Position estimation during a **landing** task in **manual mode** (RC)

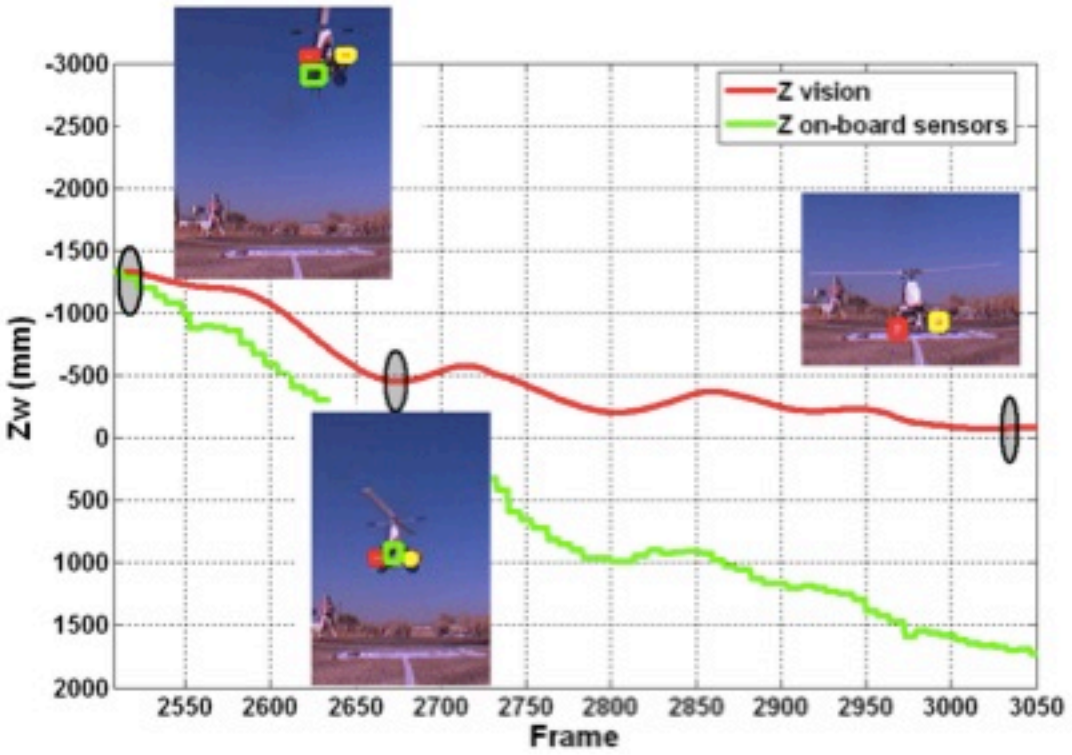


* Visual estimation **corresponds** with **real UAV position**

Experiments and Results

Pose Estimation --> TRINOCULAR SYSTEM

Position estimation during a **landing** task in **manual mode** (RC)



* Visual estimation **corresponds** with **real UAV position**
* **Improvement** of the UAV's position estimation

Partitioned control: … example 1

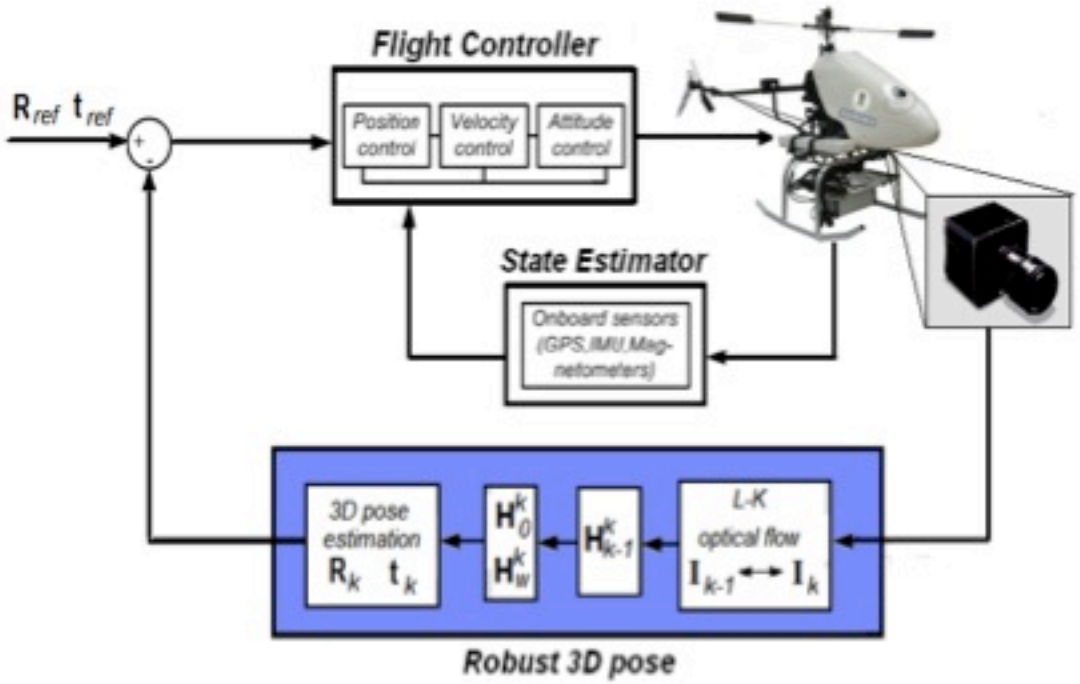
Z control

table of contents

- Visual Control: structure and control law
- Image Based Visual Control
  - 2D points' position
  - Other visual features
- Pose Based Visual Control
  - Ground camera pose estimation
  - On-board camera pose estimation
- Hybrid and Partitioned Visual Control

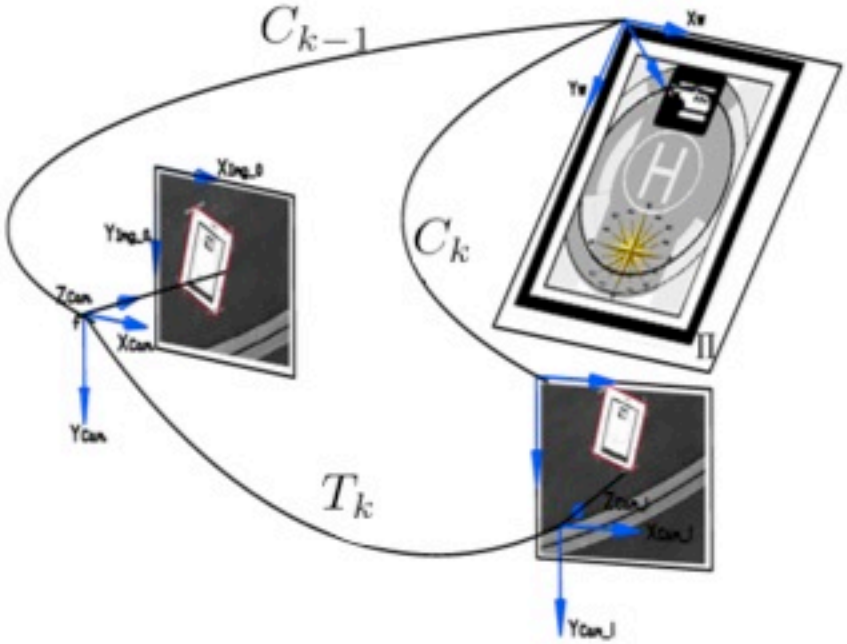
Visual Odometry: **Objective**

Estimate the egomotion using on-board cameras



Visual Odometry: working principle

Estimates incrementally the pose of the vehicle by examination of the on-board image changes



Visual Odometry: **Steps**

1. Image acquisition and correction
2. Feature detection and description
3. Feature matching
4. Robust matching for pose estimation
5. Pose optimization

Visual Odometry: **Steps**

1. Image acquisition and correction
    1. Acquisition using either single cameras, stereo cameras, or omnidirectional cameras.
    2. Correction: preprocessing techniques for lens distortion removal, noise removal, etc.
2. Feature detection and description
3. Feature matching
4. Robust matching for pose estimation
5. Pose optimization

Visual Odometry: **Steps**

1. Image acquisition and correction
2. Feature detection and description
   1. Feature detection: corner detectors (Moravec, Forstner, Harris, Shi-Tomasi, FAST) or blob detectors (SIFT, SURF, CENSUR)
   2. Feature description: local appearance or invariant descriptors (SIFT, SURF, BRIEF, ORB, BRISK, FAST)
3. Feature matching
4. Robust matching for pose estimation
5. Pose optimization

Visual Odometry: **Steps**

1. Image acquisition and correction
2. Feature detection and description
3. Feature matching
   - Local tracking (LK, KLT)
     - vs.
   - Global matching
4. Robust matching for pose estimation
5. Pose optimization

Visual Odometry: **Steps**

1. Image acquisition and correction
2. Feature detection and description
3. Feature matching
   - Local tracking (LK, KLT)
     - vs.
   - Global matching
     - Similarity measurement
     - Mutual consistency
     - Motion consistency
4. Robust matching for pose estimation
5. Pose optimization

Table of contents

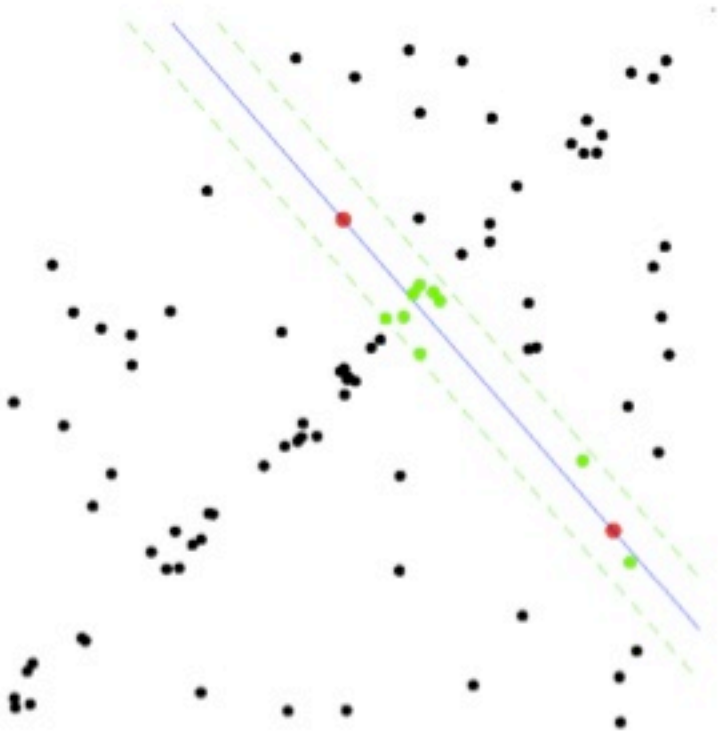
1. Image acquisition and correction
2. Feature detection and description
3. Global feature matching
4. Robust matching and pose estimation
5. Pose optimization

Robust matching

- Problem: false matched points (i.e. outliers) result in errors in pose estimation
  (caused in image acquisition (noise, blur, ..), feature detetor/ descriptor or matching)
- Solution: remove outliers don't fitting predominant model.
- RANSAC is the standard
  - it stands for random sample consensus
  - first by Fishler & Bolles, 1981

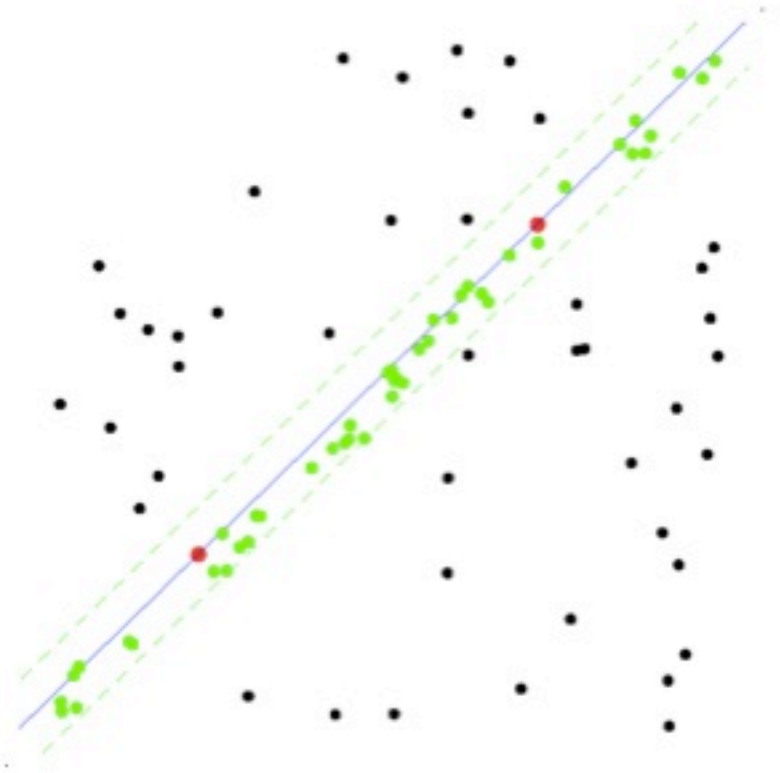
Source: Scaramuzza

RANSAC: working principle



1. Randomly choose *s* samples
   Typically *s* = minimum sample size that lets fit a model
2. Fit a model (e.g., line) to those samples
3. Count the number of inliers that approx. fit the model
   (distance to model <d)

RANSAC: working principle



1. Randomly choose *s* samples
   Typically *s* = minimum sample size that lets fit a model
2. Fit a model (e.g., line) to those samples
3. Count the number of inliers that approx. fit the model (distance to model <d)
4. Repeat *N* times
5. Choose the model that has the largest set of inliers

RANSAC: number of iterations

The number of iterations necessary to guarantee a correct solution is:

$$N = \frac{log(1-p)}{log(1-(1-\varepsilon)^s)}$$

s is the number of points to obtain a model
ε is the rate of outliers in the data
p is the probability of success

Example: p=99.9%, s=2, ε =25% ➔ N= 8.35

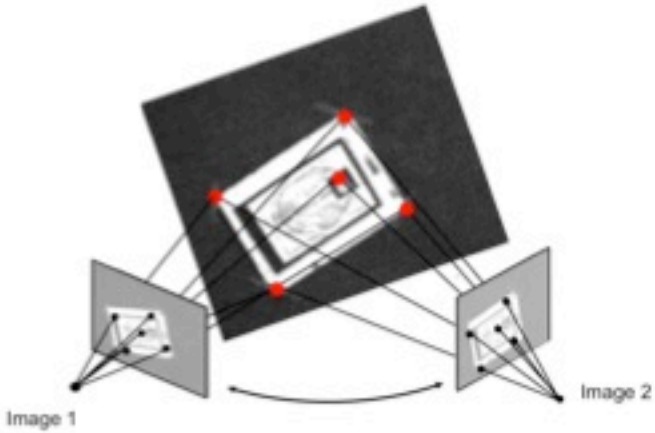
Features:

- RANSAC is non deterministic, whose solution tends to be stable when N grows
- N is usually multiply by a factor of 10
- Advanced implementations estimate ε after every iteration

RANSAC for Visual Odometry

1. Randomly choose *s* samples
2. Fit the motion model

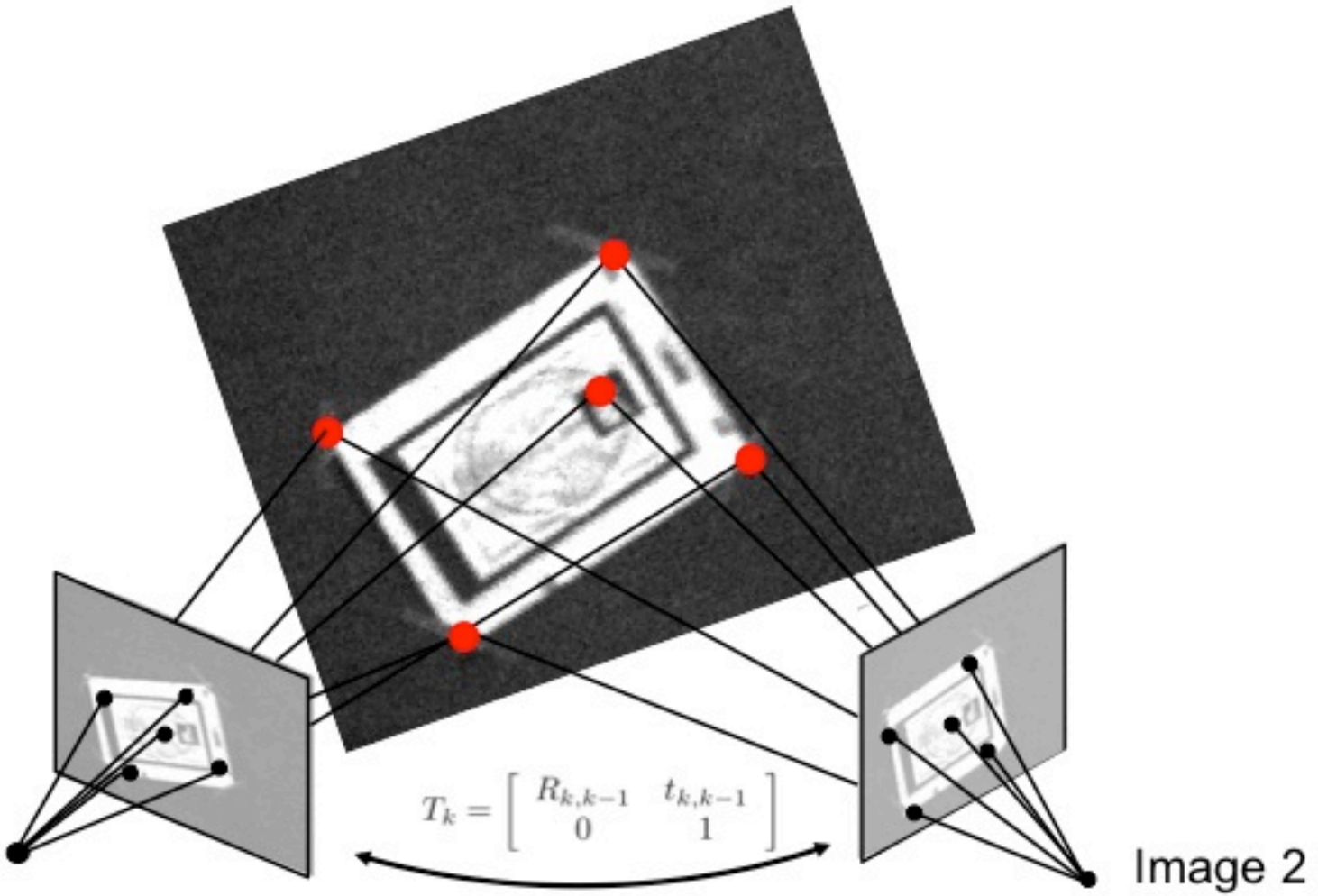
   Obtain $$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$



   it can be calculated by minimizing the following points correspondences: 2D-2D, 3D-3D or 3D-2D

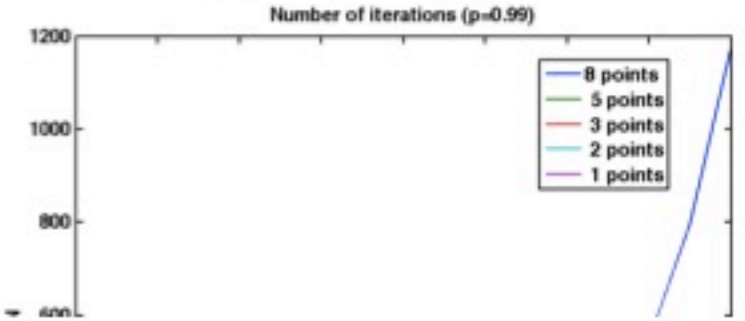
4. Count the number of inliers that approx. fit the model (distance to model <d)
5. Repeat *N* times
6. Choose the model that has the largest set of inliers

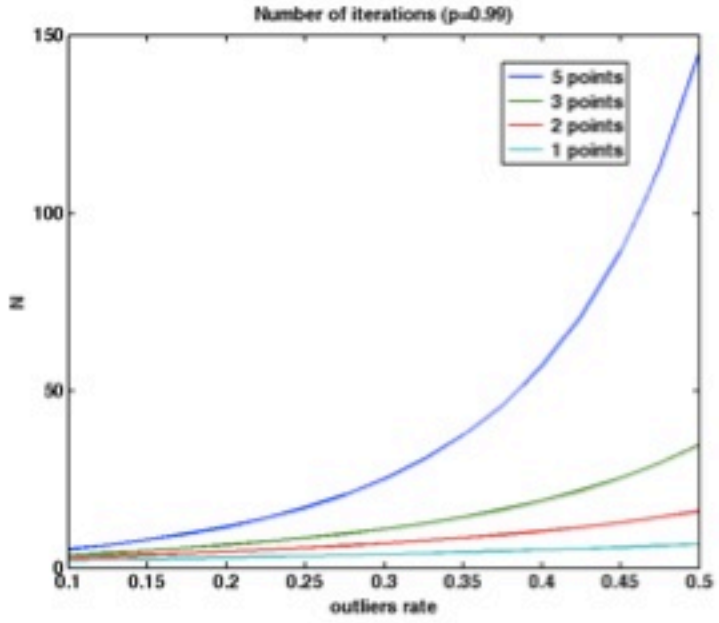
RANSAC for V.O.: motion model



RANSAC for V.O.: nr. of points

$$N = \frac{log(1-p)}{log(1-(1-\varepsilon)^s)}$$



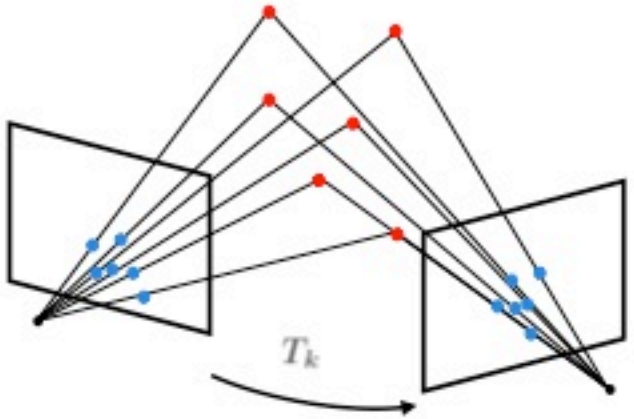


- For a 6 DOF uncalibrated/calibrated camera:
  8 points non coplanar points algorithm by Longguet-Higgins' (1981)
- For a 6 DOF calibrated camera:
  5 points are enough Krupta (1913), efficient implementation by Nister (2003)
- If 2angles are known:
  3 points are enough by Fraundorfer et alt. (2010), 2 angles estimation by far point by Narodisky et alt.(2011)
- If 3 angles are known:
  2 points are enough by Kneip at alt. (2011)
- For planar motion
  2 points are enough by Ortin et alt. (2001)
- For wheeled vehicles of 2DOF
  1 point is enough by Scaramuzza et alt. (2011)

Motion from Image Feature Correspondences: 2D-2D

- The minimal-case solution involves 5-point correspondences
- The solution is found by determining the transformation that minimizes the reprojection error of the triangulated points in each image

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$



$$p_2^T E\, p_1 = 0 \qquad \text{Epipolar constraint}$$

$$\mathrm{E} = [t]_x R \qquad \text{Essential matrix}$$

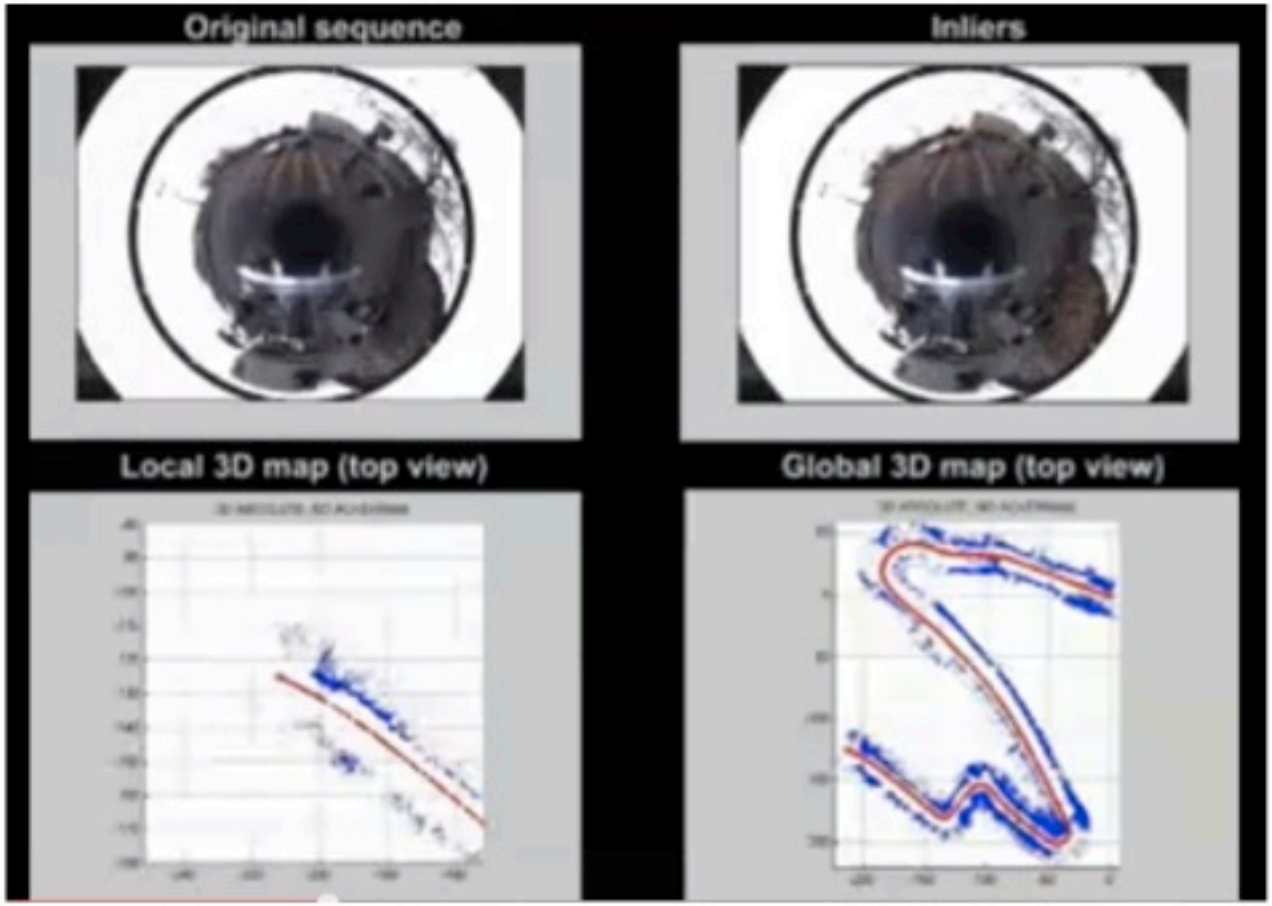
$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \qquad p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

RANSAC for V.O.: nr of points

Is it really better to use minimal sets in RANSAC?

- If one is concerned with certain speed requirements, YES
- However, might not be a good choice if the image correspondences are very noisy: in this case, the motion estimated from a minimal set wil be inaccurate and will exhibit fewer inliers when tested on all other points
- Therefore, when the computational time is not a real concern and one deals with very noisy features, **using a non-minimal set may be better than using a minimal set**

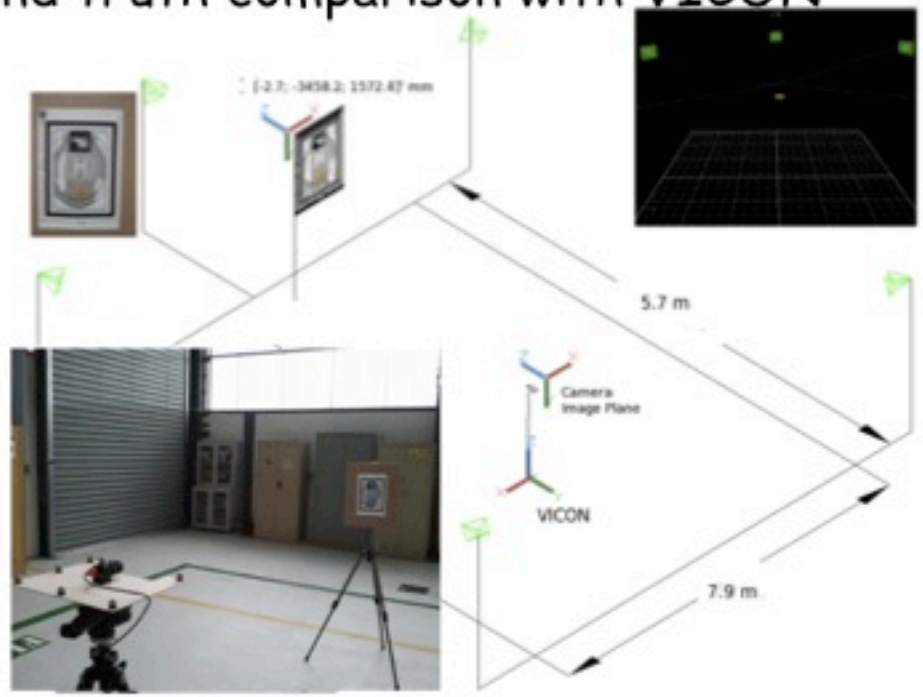
RANSAC for V.O.: results for 1 point

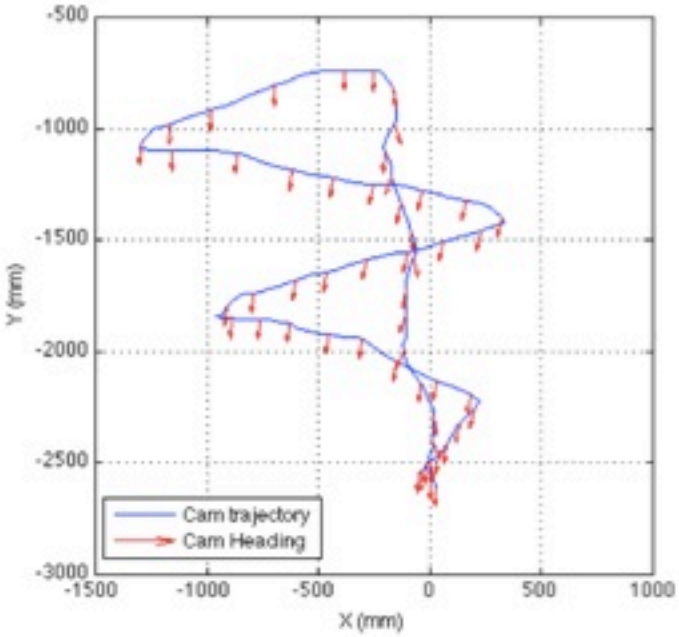


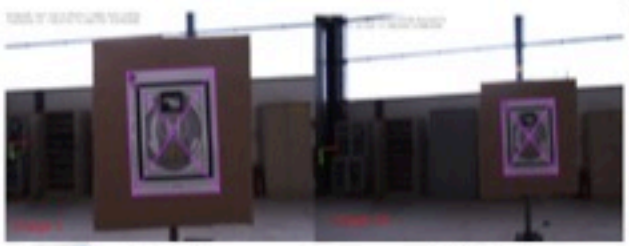
This video can be seen at
http://youtu.be/t7uKWZtUjCE

RANSAC for V.O.: results for 5 points

Ground truth comparison with VICON






img1

img50

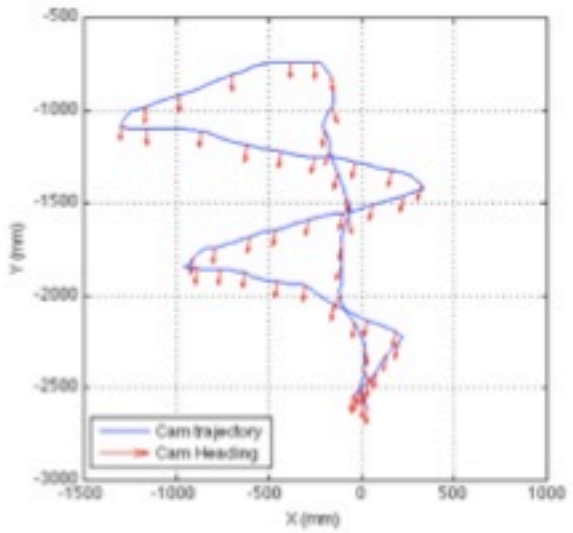
img100

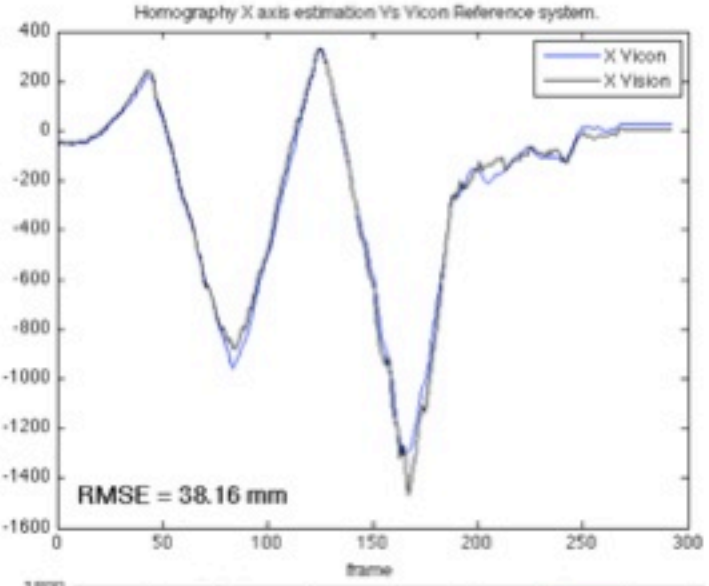
img150

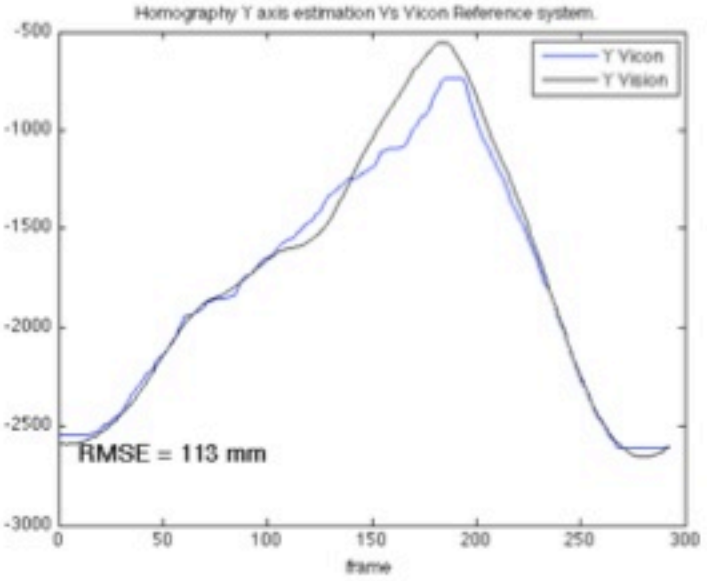
img200

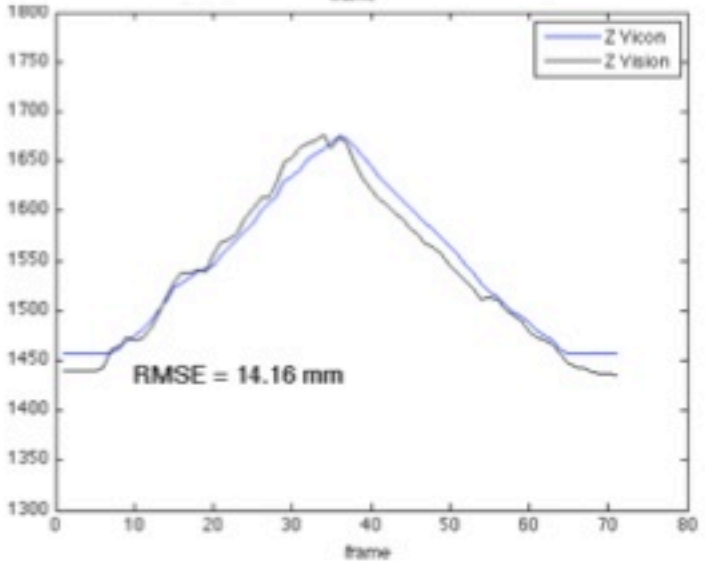
img250

RANSAC for V.O.: results for 5 points



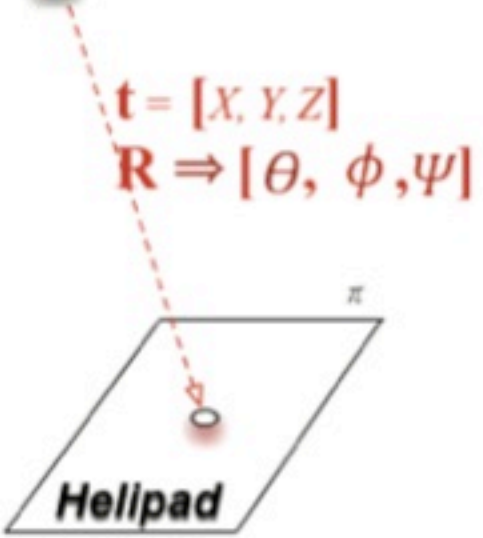


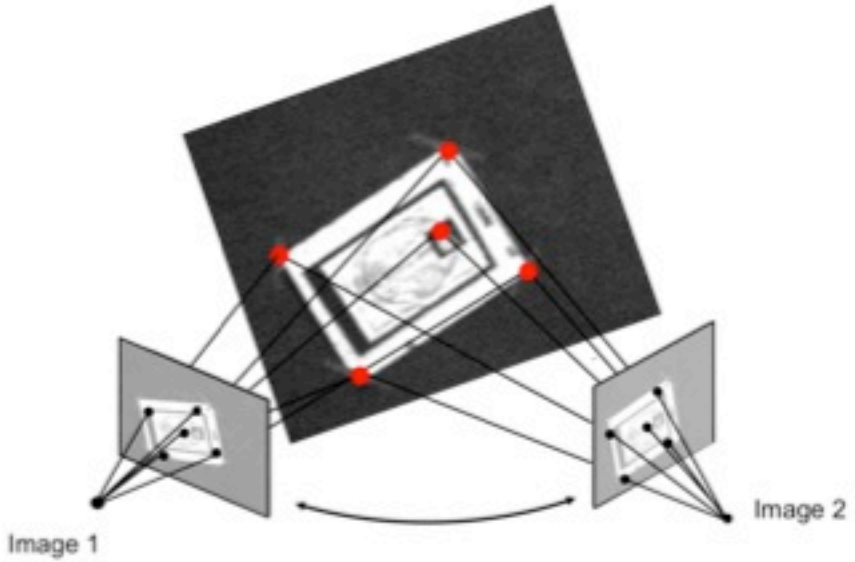




RANSAC for V.O.: results for 5 points

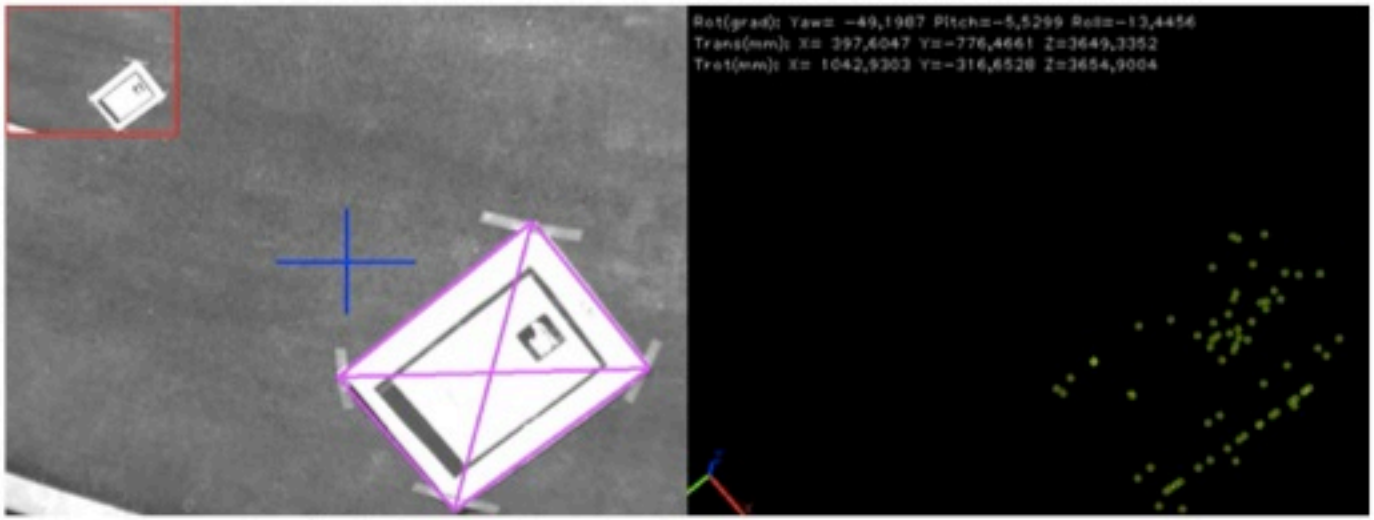
Using only **visual information** provided by an on-board camera and a know landmark, **estimate** camera-aircraft **relative position w.r.t a helipad**

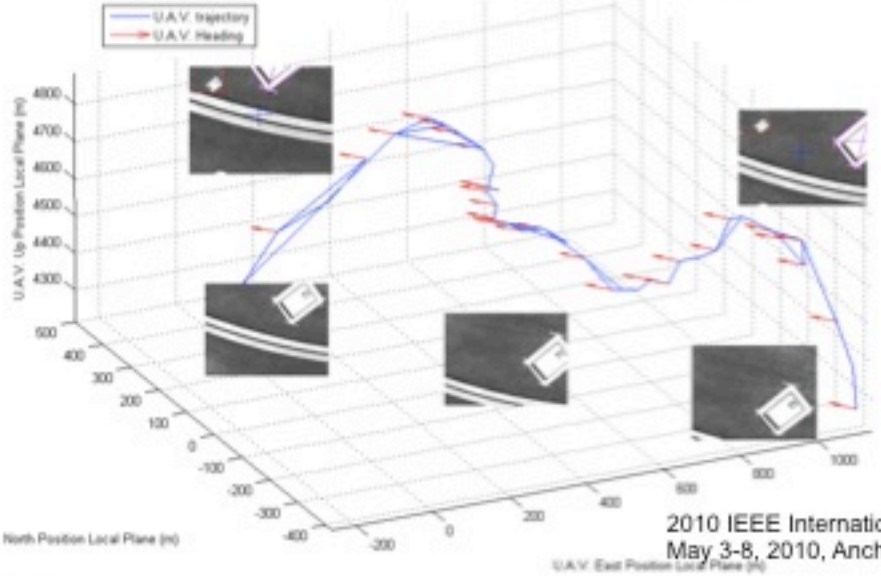




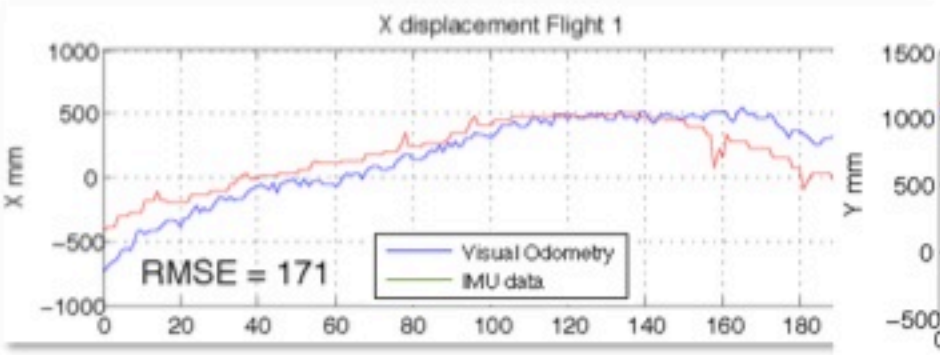
RANSAC for V.O.: results for 5 points

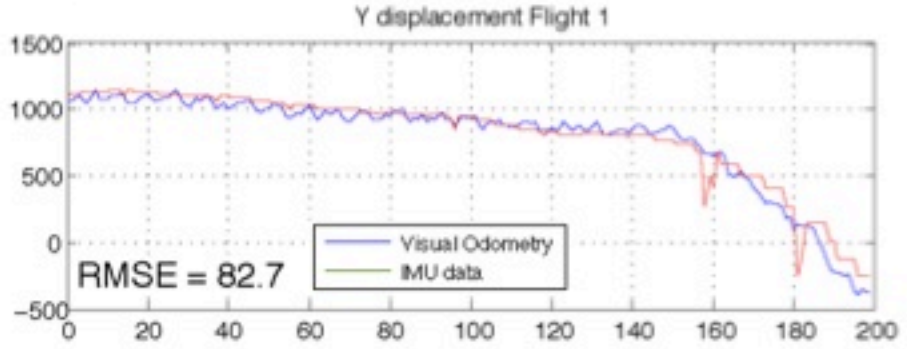
Hover at 4.5m

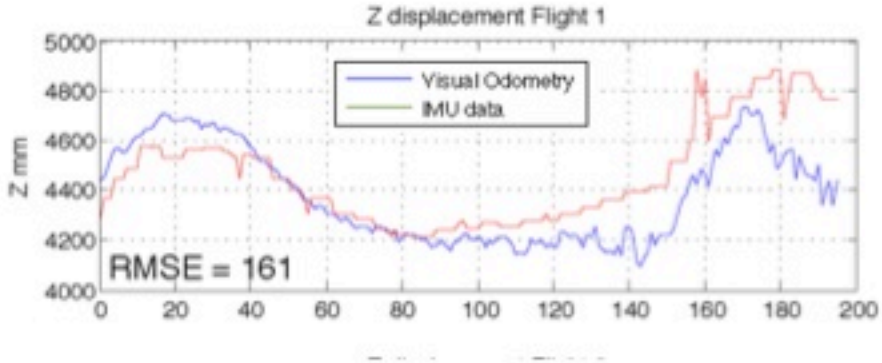


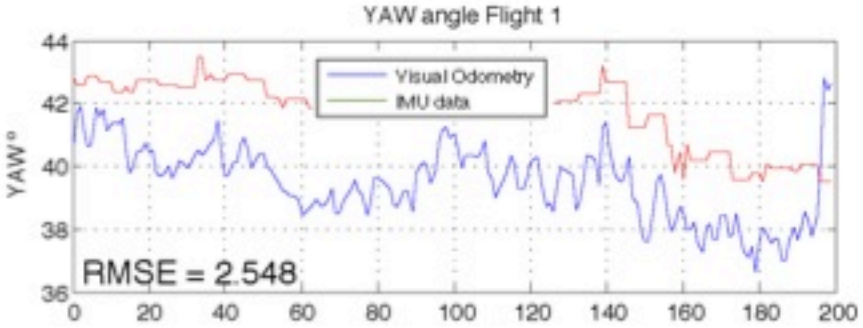


RANSAC for V.O.: results for 5 points



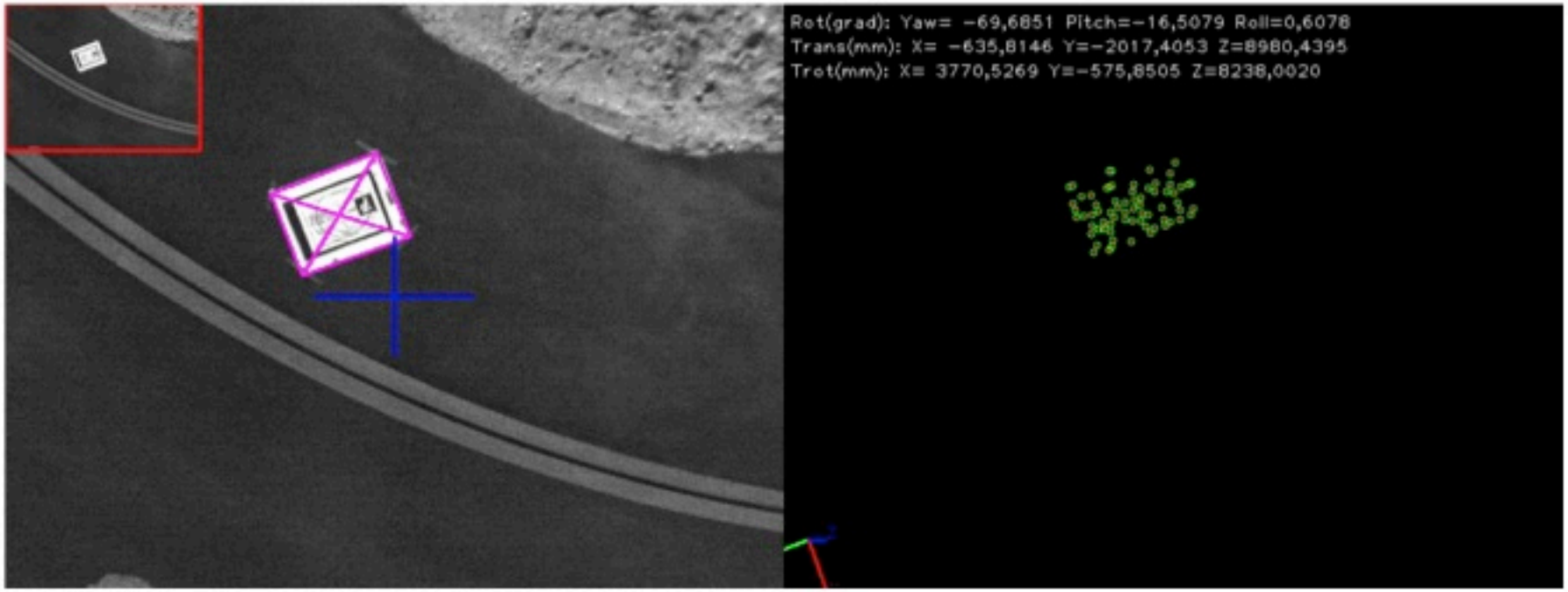




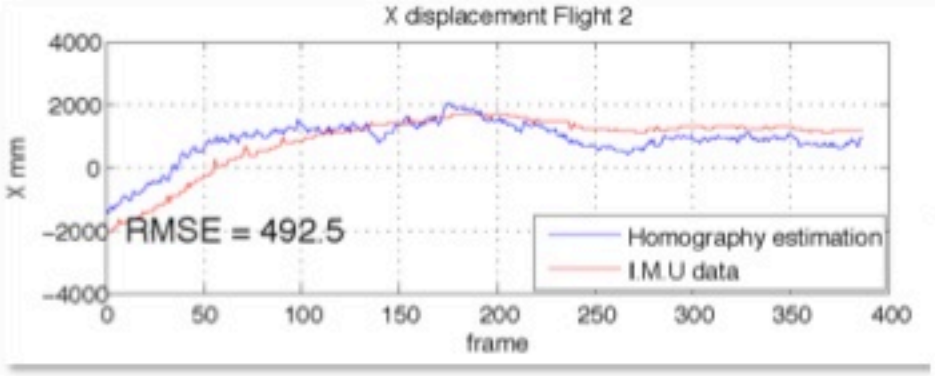


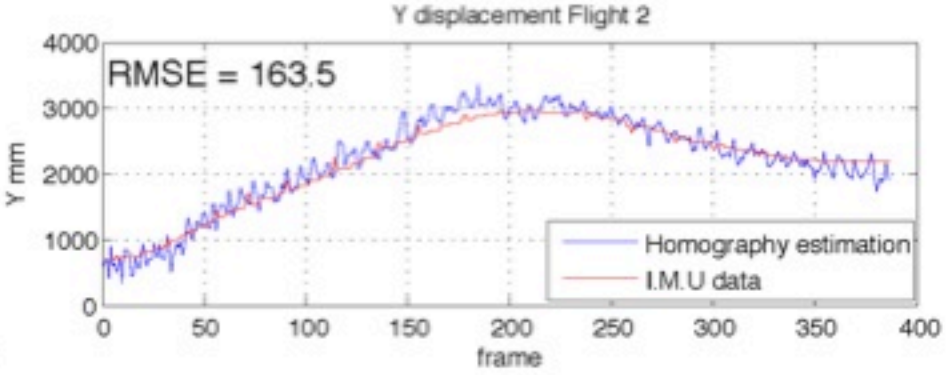
RANSAC for V.O.: results for 5 points

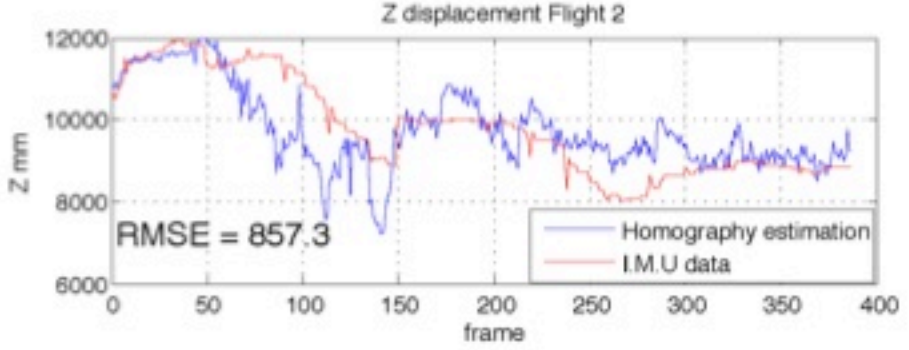
Hover at 10m

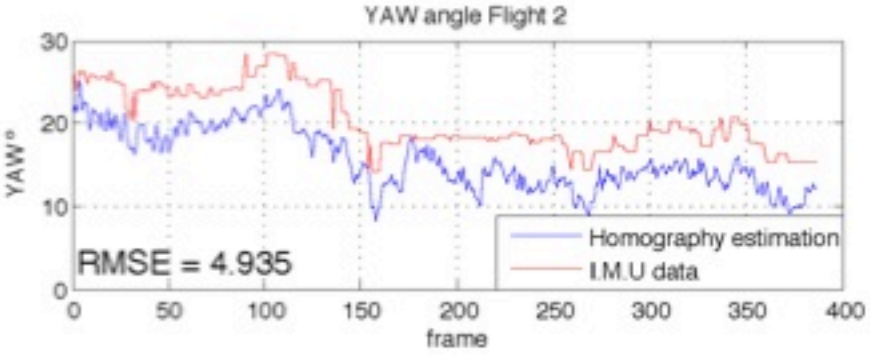


RANSAC for V.O.: results for 5 points



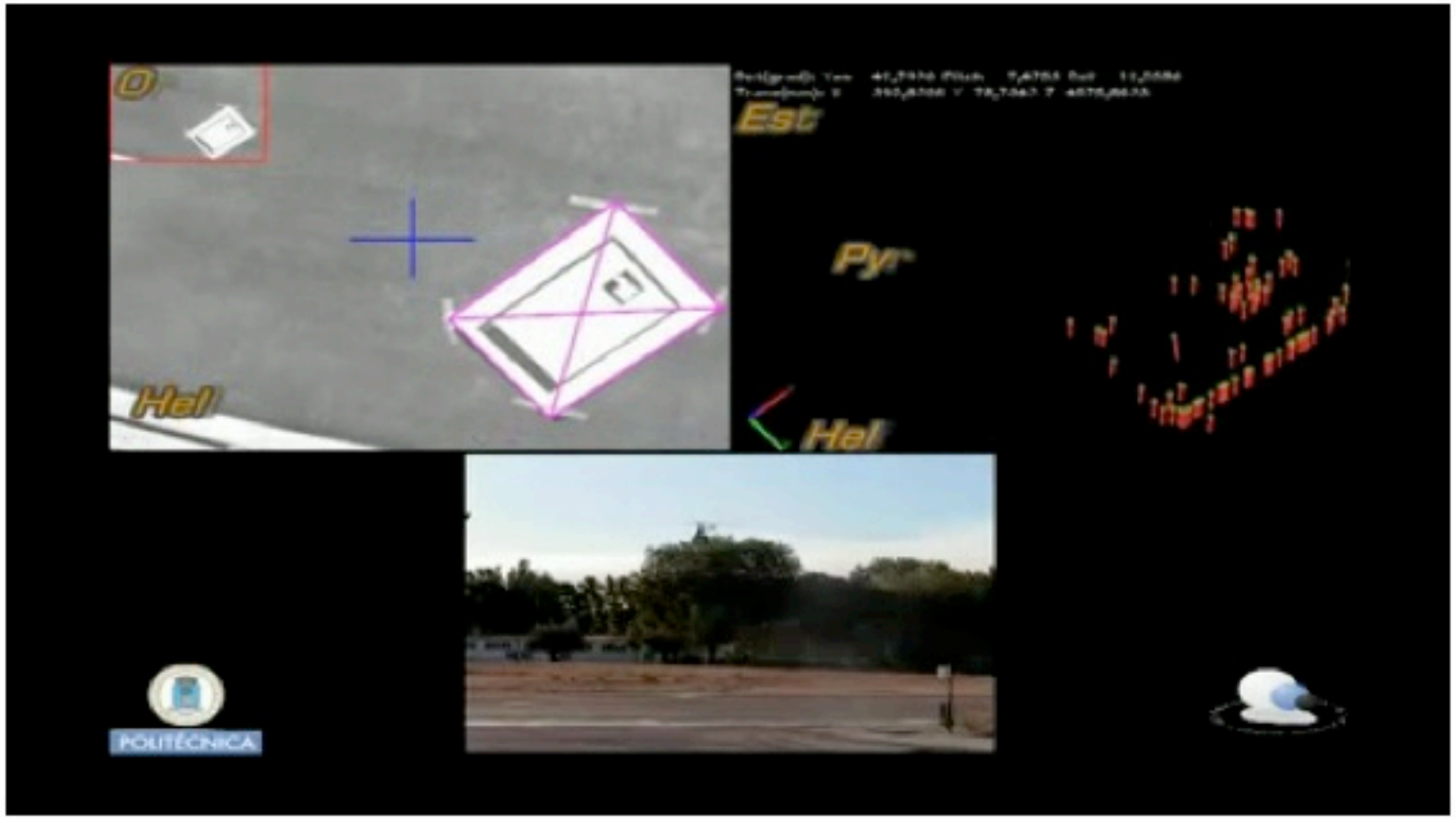






Experiments and Results

Pose Estimation --> ON-BOARD SYSTEM method 1

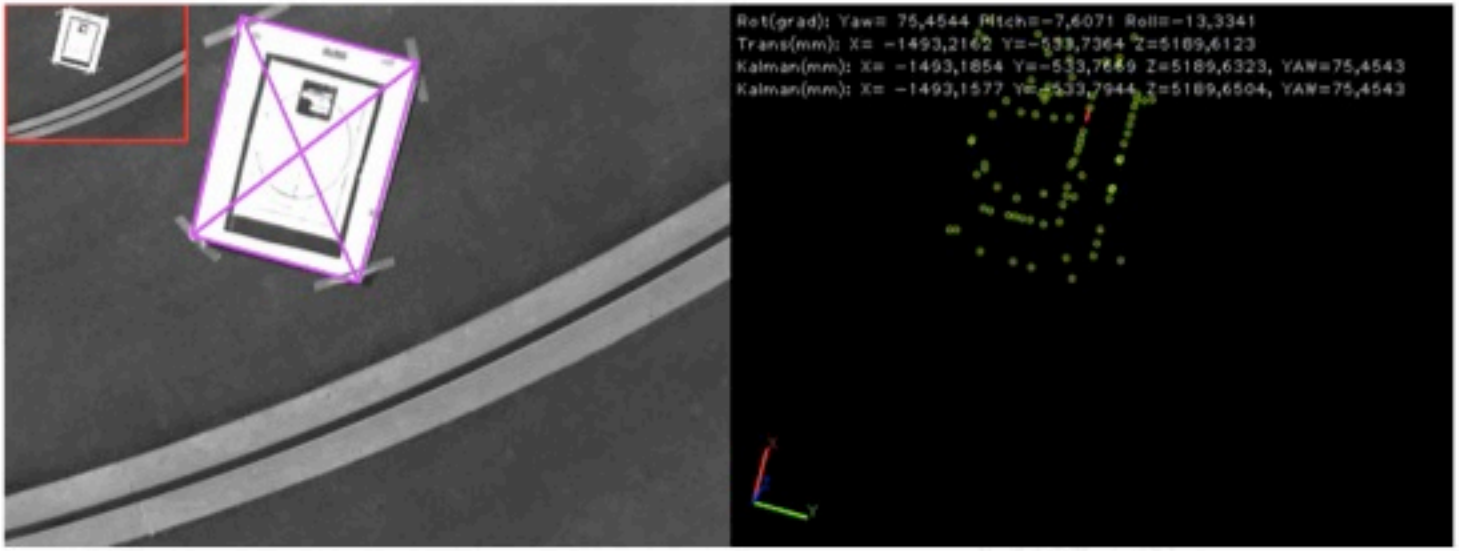


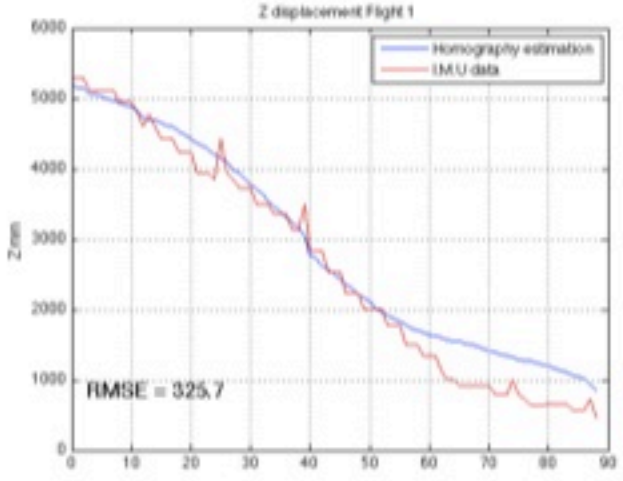
Experiments and Results

Autonomous landing

RANSAC for V.O.: results for 5 points

Manual Landing





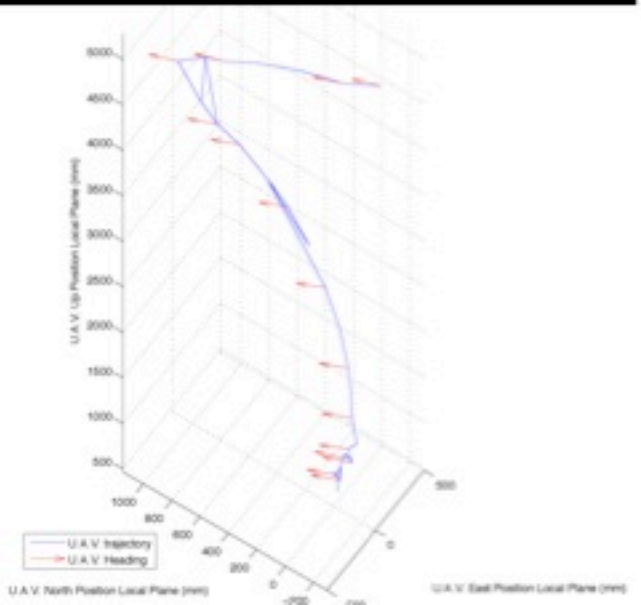
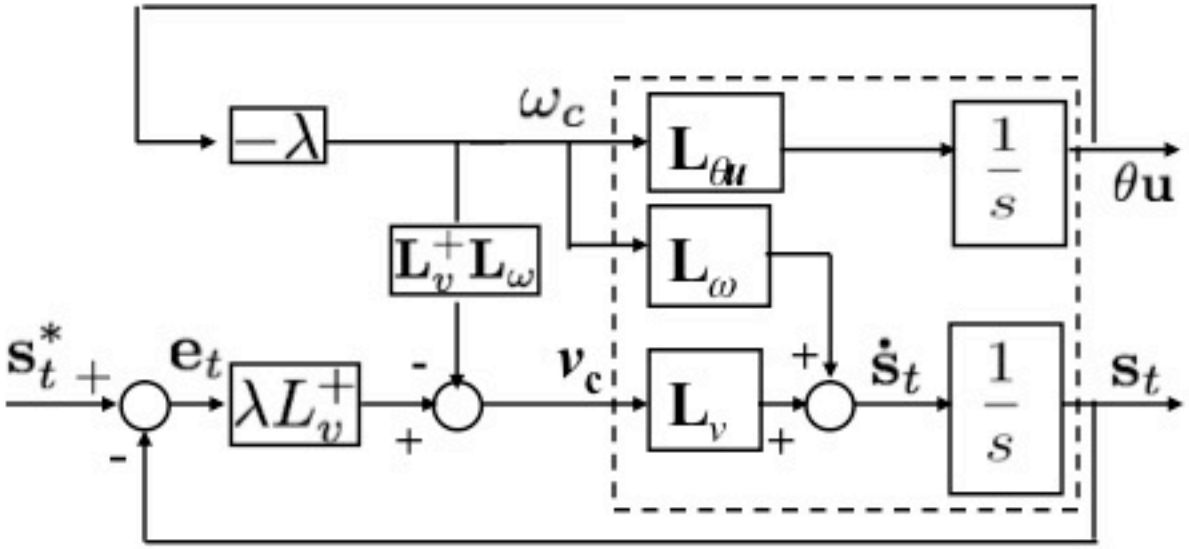


table of contents

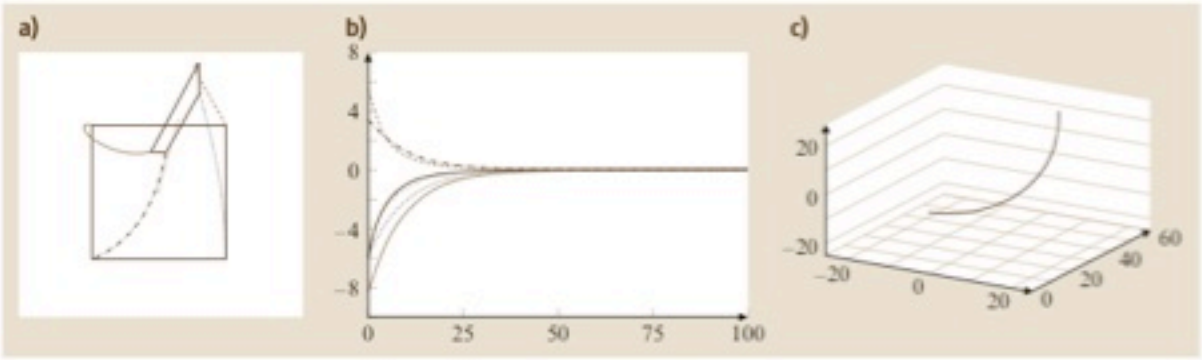
- Visual Control: structure and control law
- Image Based Visual Control
  - 2D points' position
  - Other visual features
- Pose Based Visual Control
  - Ground camera pose estimation
  - On-board camera pose estimation
- Hybrid and Partitioned Visual Control

Hybrid Visual Control

Control scheme: D2.5



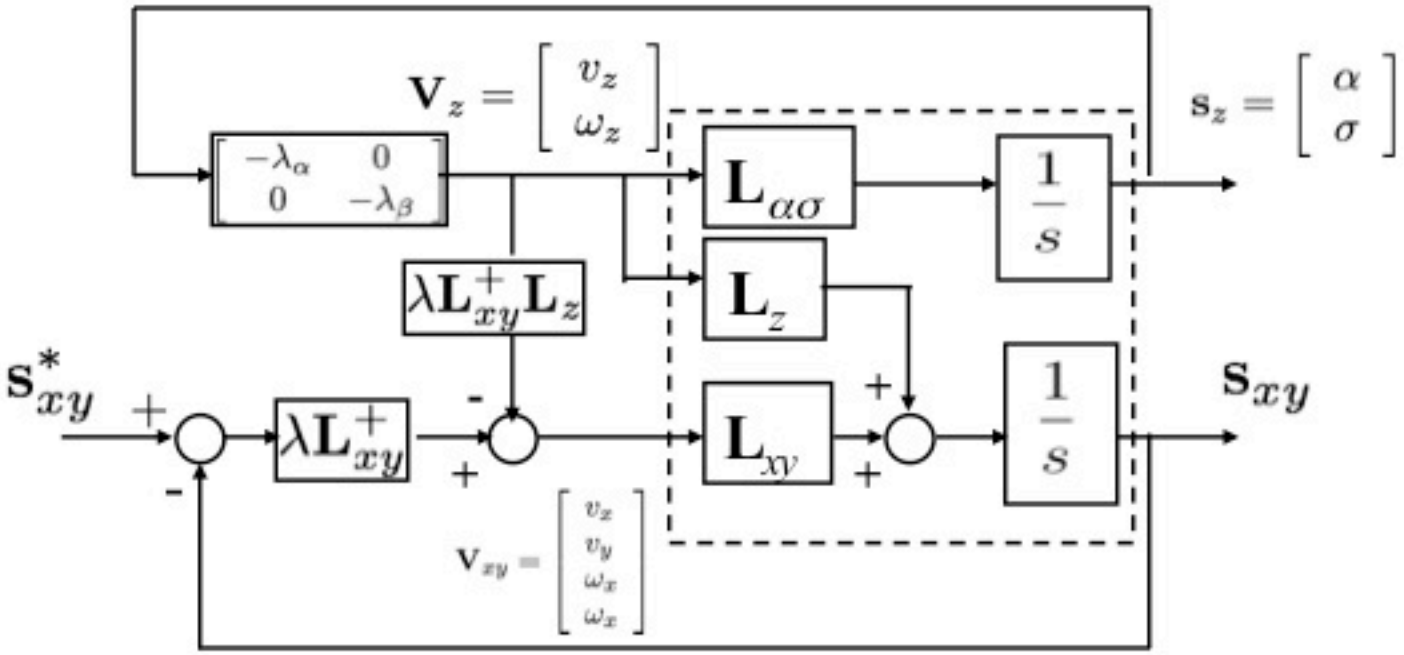
$$\mathbf{s}_t = \begin{bmatrix} x \\ y \\ logZ \end{bmatrix}$$

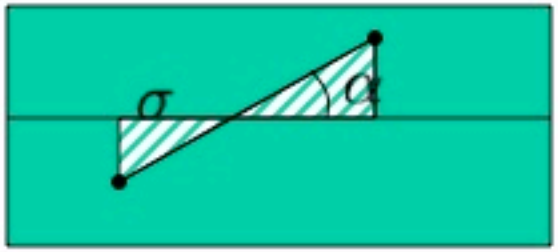


Springer Handbook of Robotics
Siciliano, Khatib (Eds.) - ©Springer 2008

Partitioned Visual Servoing

Control scheme:





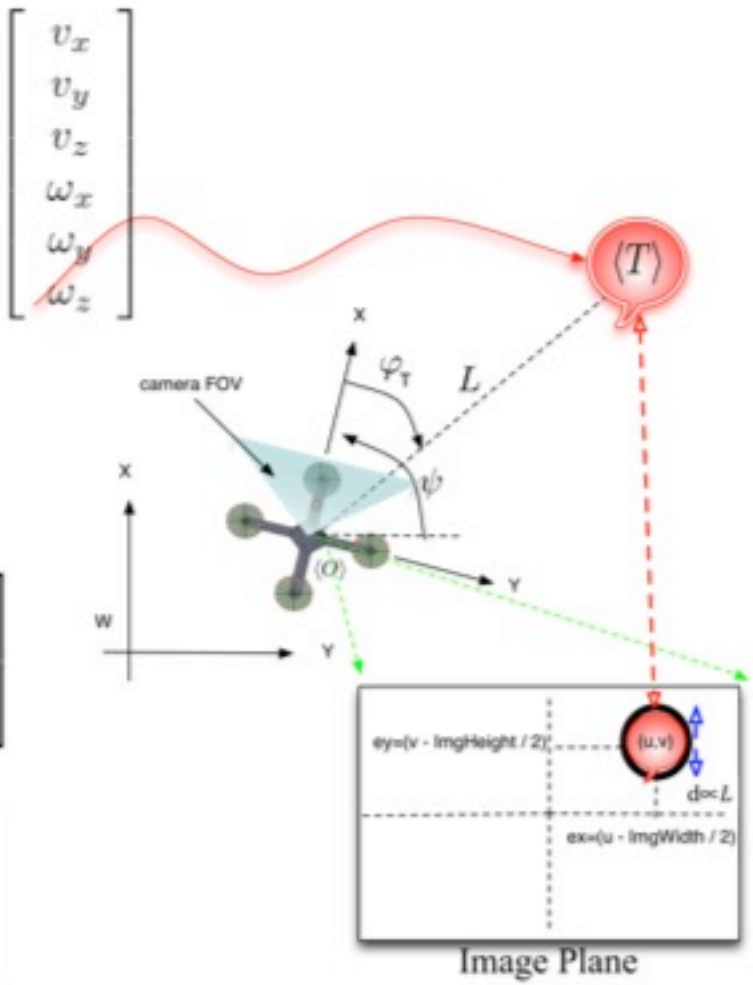
Partitioned control: example 1 ...

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x_t}{Z_T} & x_t y_t & -(x_t^2+1) & y_t \\ 0 & -\frac{1}{Z} & \frac{y_t}{Z_T} & (y_t^2+1) & -x_t y_t & -x_t \\ 0 & 0 & \frac{\phi_t}{Z_T} & \phi_t y_t & -\phi_t x_t & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & \frac{x_t}{Z_T} & -(x_t^2+1) \\ -\frac{1}{Z} & \frac{y_t}{Z_T} & -x_t y_t \\ 0 & \frac{\phi_t}{Z_T} & -\phi_t x_t \end{bmatrix} \begin{bmatrix} v_y \\ v_z \\ \omega_y \end{bmatrix}$$

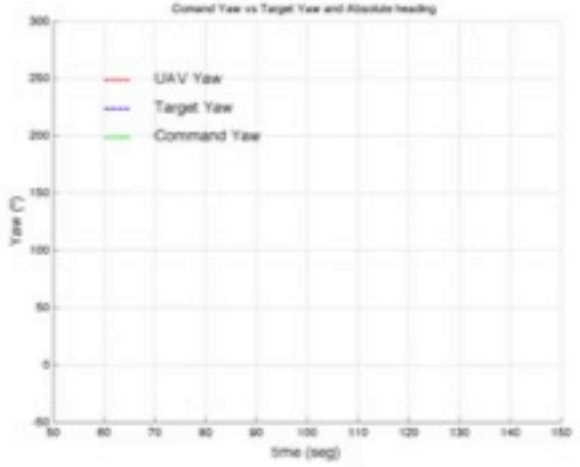
$$\begin{bmatrix} v_y \\ v_z \\ \omega_y \end{bmatrix} = -\lambda \begin{bmatrix} 0 & -Z & \frac{Z y_t}{\phi_t} \\ -Z x_t & 0 & \frac{Z(x_t^2+1)}{\phi_t} \\ -1 & 0 & \frac{x_t}{\phi_t} \end{bmatrix} \begin{bmatrix} x_t - x^* \\ y_t - y^* \\ \psi - \psi^* \end{bmatrix}$$

$$\lim_{x_t,\, y_t \to 0} \Lambda = \lim_{x_t,\, y_t \to 0} \mathbf{L}_s . * \mathbf{L}_s^{-1^T} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

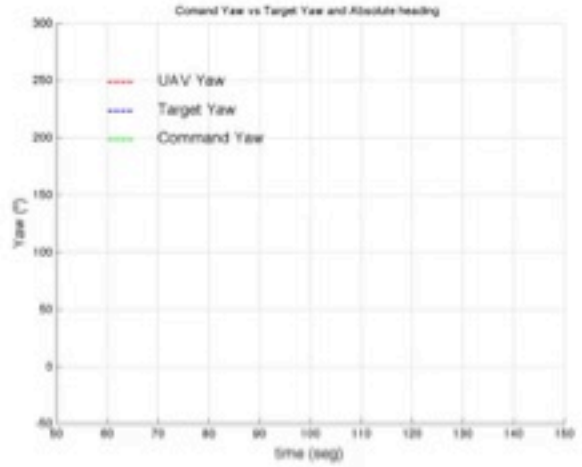


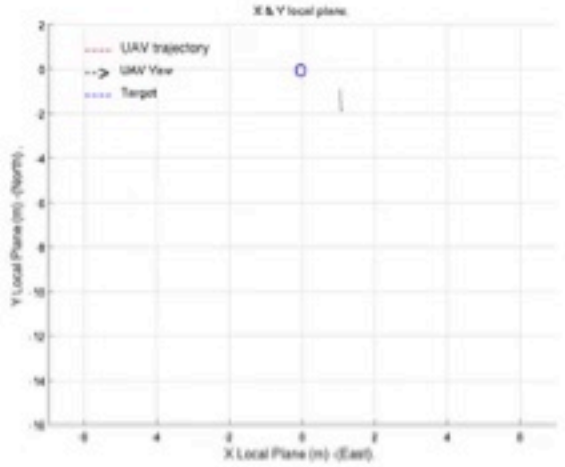
Partitioned control: ... example 1...

Partitioned control: … example 1



Partitioned control: ... example 2



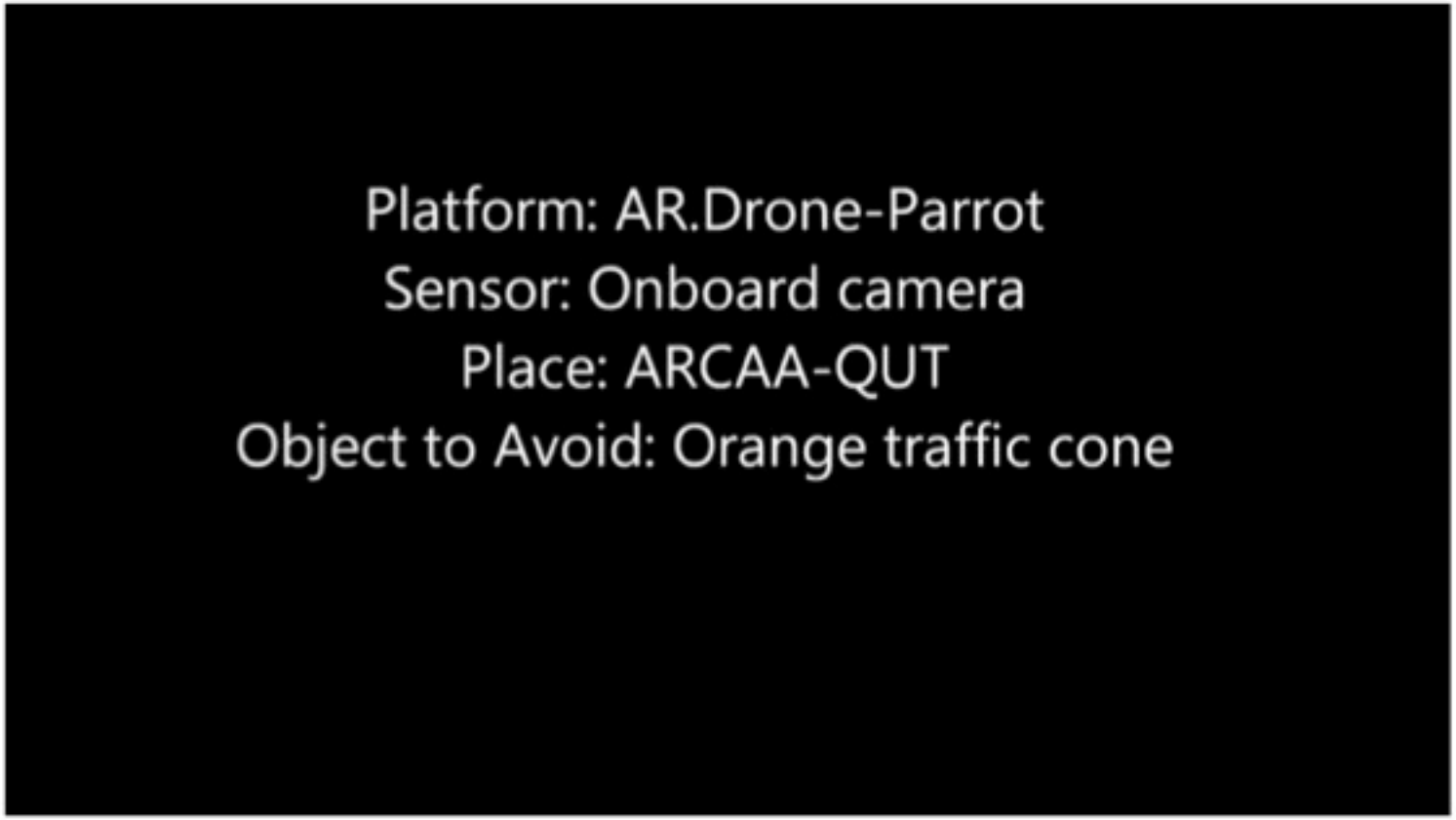


Partitioned control: example 2

See & avoid based on visual features
on the omnidirectional unitary sphere

Partitioned control: example 2

See & avoid based on visual features
on the omnidirectional unitary sphere



Thank you for your attention

Questions ?

more info: www.vision4uav.es