Department of Electrical
and
Computer Systems Engineering

Technical Report
MECSE-1-2004

Aircraft Pose Estimation from Homography

E. Beets, S. Boukir, and D. Suter

MONASH
UNIVERSITY

Aircraft Pose Estimation from Homography

Eric Beets, Samia Boukir, and David Suter

Abstract

*The aim of this work was to determine, experimentally, whether the relative pose (position and orientation of an aircraft), with respect to the ground plane, could be **usefully** determined by vision alone. It is relatively well known that such is, in principle, possible by exploiting the image homography between two views of the same planar surface.*

1. Introduction

This work is related to the ARC SPIRT grant funded project "Vision for Aircraft Landing". In that project, the main aim is to develop visual means to enable a robotic aircraft to land (without the assistance of a trained remote control pilot). As such, the project requires the determination of what useful information can be extracted visually. Specifically, in this case, whether one can determine the relative change in pose, given two views taken from the aircraft at two different time instants (see figure 1).

In some sense, the present work is complementary to the work reported in [Suter, Hamel & Mahony 02]. In that work, the principle of using an homography to visually servo an aircraft to a given position, w.r.t. a ground plane (or, indeed, any planar surface that is visible) was investigated. However, that paper focused on the control aspects of the task – assuming that the visual processing could provide the required quantities (rotation and translation, up to a scale, between the two views). The work reported here aims to determine the accuracy to which one can realistically extract these parameters by that method. Moreover, [Suter, Hamel & Mahony 02], since it dealt with control aspects, was more concerned with a particular airframe (the X4-flyer), whereas this work is independent of the viewing platform – there are no dynamics involved. A major motivation of this work is to aid in the visual positioning and landing of any aircraft – but particularly the fixed wing type craft typically flown by Aerosonde. This work is an alternative approach to that of [Mahony & Suter 01]. It was also intended to be integrated with the work of [Tung, Suter & Bab-hadiashar 03].

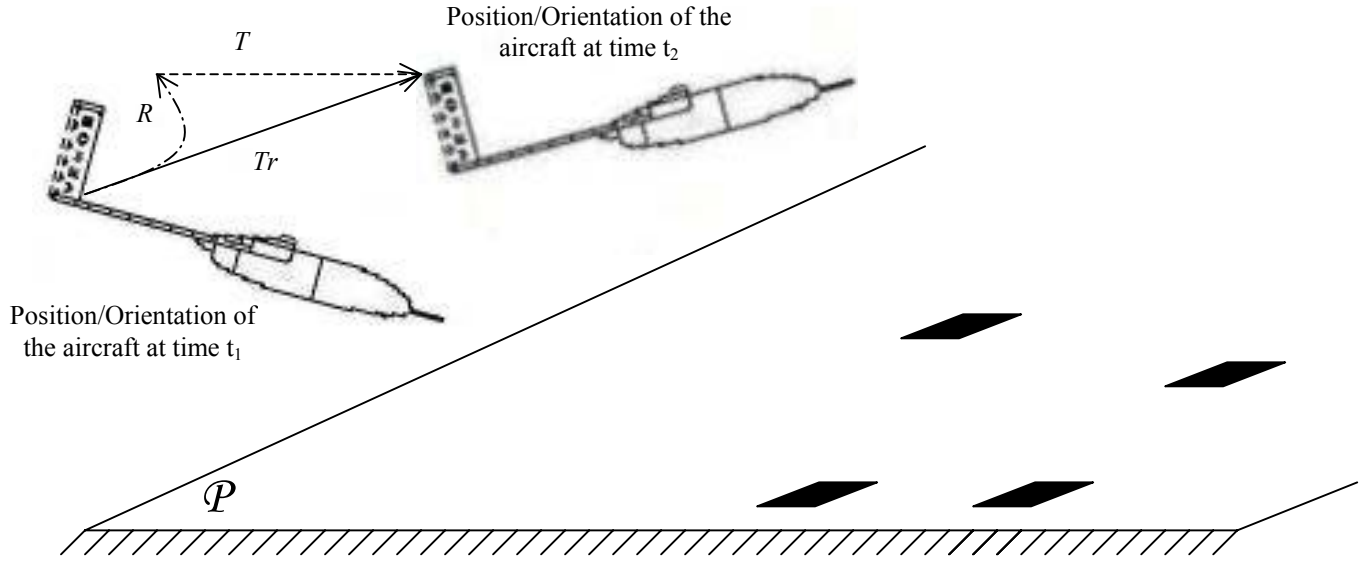


Fig.1 – *Illustration of the aircraft pose-estimation task*

Transformation between two instants (*Tr*)

Rotation of the transformation (*R*)

Translation of the transformation (*T*)

Visual features on the land

The approach taken here is to employ pose estimation method based on the tracking of a rigid planar patch [Tsai & Huang 82]. The steps involved are :

1. A corner tracker algorithm is used to track at least 4 points belonging to the plane of interest. This work used the token tracker available in OpenCV. This token tracker consists of two steps: first, it determines strong corners in an image and then, it calculates the optical flow between two images using the iterative hierarchical Lucas-Kanade method [Shi & Tomasi 94] to follow the tracked points.
2. A pose estimation algorithm based upon homography extraction [Tsai & Huang 82].

The code developed uses several libraries:

- Direct Show : to read video sequences (AVI files).
- OpenCV : to read images, track points and compute matrices.
- OpenGL : to visualize the results (2D and 3D).

2. Homography estimation

The three-dimensional motion parameters of a rigid planar patch can be determined by computing the singular value decomposition (SVD) of a 3x3 matrix containing the eight so called "pure parameters." This planar transformation is called a homography. Furthermore, aside from a scale factor for the translation parameters, the number of solutions is either one or two, depending on the multiplicity of the singular values of the matrix [Tsai & Huang 82].

This method provides the rotation, the translation of the moving camera (mounted on an aircraft in our case) up to a scale factor and the orientation of the 3D plane of interest. To lift the ambiguity and ensure a unique solution, Faugeras suggests the use of a third image in the estimation scheme [Faugeras & Lustman 88]. Thus, the first estimation (using 2D points of image 1 and image 2) gives two different planes orientation (defined by their normals). The second estimation (using image 2 and image 3) leads to two solutions too. The common solution to both estimations is the right one.

2.1 – 2D Mapping

We consider a 3D point M belonging to a plane (P), moving to M' belonging to a plane (P'), respectively such that:

$M\begin{bmatrix} x \\ y \\ z \end{bmatrix} \in (P)$ with $\begin{bmatrix} X \\ Y \end{bmatrix}$ the coordinates of the perspective projection of M in image I

and

$M'\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \in (P')$ and $\begin{bmatrix} X' \\ Y' \end{bmatrix}$ the coordinates of the perspective projection of M' in image I'.

The projective transformation A (or homography) between the two planes is as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = A\begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = z \bullet A\begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \Rightarrow \frac{z'}{z}\begin{bmatrix} x'/z' \\ y'/z' \\ 1 \end{bmatrix} = A\begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} S.X' \\ S.Y' \\ S \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

with $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ and S being an indeterminate scale factor.

The former development involves the well-known perspective projection equations for a pin-hole camera:

$$\begin{cases} X = f \dfrac{x}{z} \\ Y = f \dfrac{y}{z} \end{cases}$$

Without a loss of generality, the focal length f has been set to 1.

2.2 – Least squares estimation of the homography

The following system links the 2D coordinates of two corresponding image points via the homography A:

$$\begin{bmatrix} S.X' \\ S.Y' \\ S \end{bmatrix} = \underset{(3x3)}{A} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

This system is composed of 9+1 unknowns (parameters of the homography A and the scale factor S).

$$\Leftrightarrow \begin{bmatrix} S.X' \\ S.Y' \\ S \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} S.X' = a_{11}.X + a_{12}.Y + a_{13} \\ S = a_{31}.X + a_{32}.Y + a_{33} \end{cases}$$

$$\Rightarrow a_{31}.X.X' + a_{32}.Y.X' + a_{33}.X' = a_{11}.X + a_{12}.Y + a_{13}$$

The estimation of the parameters a_i is possible if we set, for example, $a_{33} = 1$, a commonly used assumption. Indeed, A is defined up to a scale factor.

$$\Leftrightarrow a_{11}.X + a_{12}.Y + a_{13} - a_{31}.X.X' - a_{32}.Y.X' = X' \qquad (1)$$

We have as well:

$$\Rightarrow \begin{cases} S.Y' = a_{21}.X + a_{22}.Y + a_{23} \\ S = a_{31}.X + a_{32}.Y + a_{33} \end{cases}$$

$$\Rightarrow a_{31}.X.Y' + a_{32}.Y.Y' + a_{33}.Y' = a_{21}.X + a_{22}.Y + a_{23}$$

Thus:

$$\Leftrightarrow a_{21}.X + a_{22}.Y + a_{23} - a_{31}.X.Y' - a_{32}.Y.Y' = Y' \qquad (2)$$

This leads to the following linear system of 8 unknowns (a_i):

$$\underbrace{\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -X_1.X'_1 & -Y_1.X'_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -X_1.Y'_1 & -Y_1.Y'_1 \\ \dots & & & & & & & \\ X_n & Y_n & 1 & 0 & 0 & 0 & -X_n.X'_n & -Y_n.X'_n \\ 0 & 0 & 0 & X_n & Y_n & 1 & -X_n.Y'_n & -Y_n.Y'_n \end{bmatrix}}_{M_{(2n\times 8)}} \cdot \underbrace{\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix}}_{\underline{x}} = \underbrace{\begin{bmatrix} X'_1 \\ Y'_1 \\ \dots \\ X'_n \\ Y'_n \end{bmatrix}}_{\underline{b}}$$

We need at least 4 corresponding projectively independent (i.e. such that any 3 of them are not aligned) points (M_i, M'_i) between 2 images to solve this system (so $n \geq 4$).

Least squares resolution:

$$\begin{aligned} & M.\underline{x} = \underline{b} \\ \Leftrightarrow\ & M^t.M.\underline{x} = M^t.\underline{b} \\ \Leftrightarrow\ & \underline{x} = \left(M^t.M.\right)^{-1}.M^t.\underline{b} \end{aligned}$$

3. Singular value decomposition (SVD) of the homography

To compute the SVD decomposition of the homography, appropriate routines from OpenCV library were employed.

The function SVD decomposes matrix A (consisting of the 8 pure parameters a_i) into a product of a diagonal matrix and two orthogonal matrices:

$$A = U^t.W.V$$

where:

A is an arbitrary $M \times N$ matrix,
U is an orthogonal $M \times M$ matrix,
V is an orthogonal $N \times N$ matrix,
W is a diagonal $M \times N$ matrix with non-negative diagonal elements.
W is such that:

$$W = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

4 – Computing actual motion parameters from pure parameters

If the singular values of the homography A are all distinct, e.g. $\lambda_1 > \lambda_2 > \lambda_3$, then there are exactly two solutions for the motion and geometrical parameters of a rigid planar patch aside from a scale factor for the translation and geometrical parameters [Tsai & Huang 82].

We denote the motion parameters by a rotation matrix R and a translation vector T. The plane is defined by its normal n.

Closed-form solution to the pose-estimation of a planar patch is as follows:

$$R = U.\begin{bmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -s\beta & 0 & s\alpha \end{bmatrix}.V^t \;,\; T = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = w^{-1}\left[-\beta.U_1 + \left(\frac{\lambda_3}{\lambda_2} - s\alpha\right).U_3 \right] \;,\; n = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = w\left[\delta.V_1 + V_3\right]$$

where

$$\delta = \pm\left(\frac{\lambda_1^2 - \lambda_2^2}{\lambda_2^2 - \lambda_3^2}\right)^{1/2} \;,\; \alpha = \frac{\lambda_1 + s.\lambda_3.\delta^2}{\lambda_2\left(1+\delta^2\right)} \;,\; \beta = \pm\sqrt{1-\alpha^2} \;,\; s = \det(U).\det(V)$$

In each of the two solutions, sgn(β) = - sgn(δ), and w is an arbitrary coefficient (we will set it at 1). Let us notice that only the orientation of the planar patch is estimated. Its position is not known.

To solve the ambiguity problem, i.e. to choose the right solution from above, Faugeras and Lustman suggest three ways to proceed [Faugeras & Lustman 88]:

1) Look at a second plane.
2) Use a third image.
3) Use relationships between features in the plane.

We have chosen the second way which provides then two pairs of solutions (S_1,S_2) and (S'_1,S'_2) for the motion and geometrical parameters of the rigid planar patch. In general, there is only one compatible pair and the problem has therefore a unique solution [Faugeras & Lustman 88] as confirmed by our experiments.

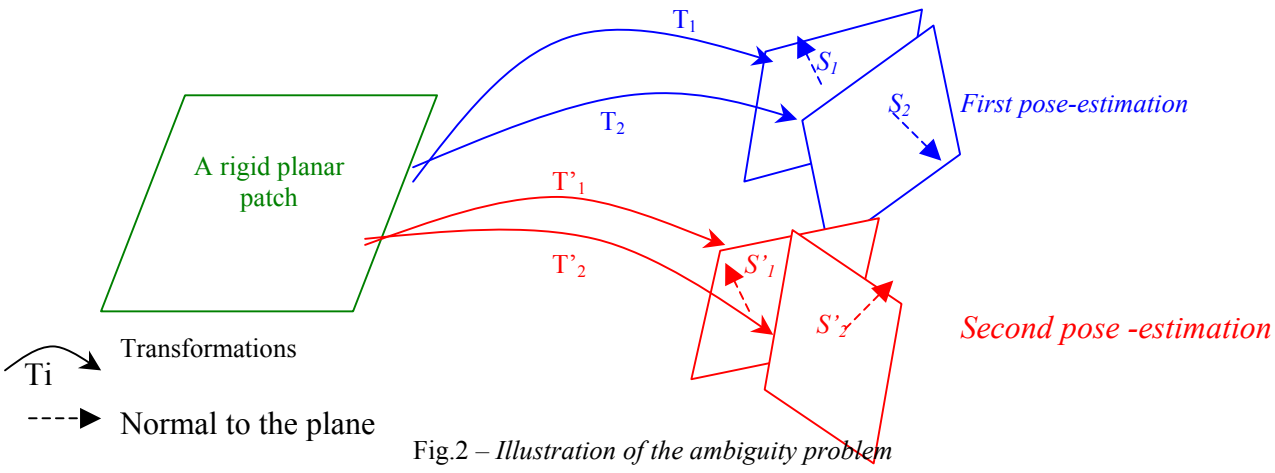


Fig.2 – *Illustration of the ambiguity problem*

In practice, to determinate the right solution, we compute four different screw products between the four obtained normal vectors. The compatible pair of normal vectors is then the one which minimizes the absolute value of this product.
However, an ambiguity in the direction of the normals remains. For example, the planes 1 and 3 (see fig. 3) are parallel but their normals have opposite direction.

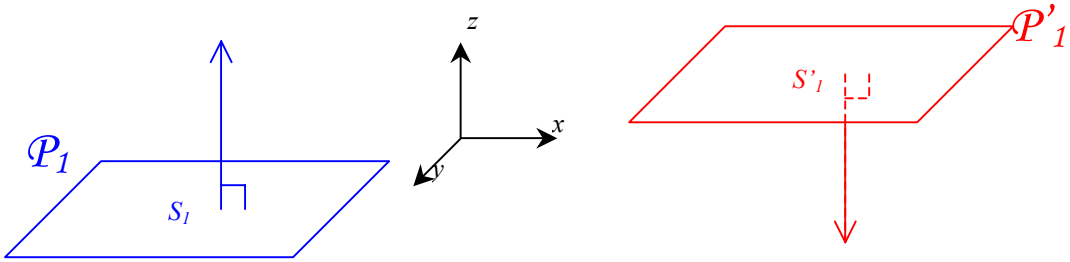


Fig.3 – *Illustration of the sign ambiguity*

With P_1 and P'_1, defined as:

$$P_1 = a_1.x + b_1.y + c_1 z = 0 \quad \text{with } n_1 = (a_1 \quad b_1 \quad c_1)^t \quad \text{and here } n_1 = (0 \quad 0 \quad 1)^t$$

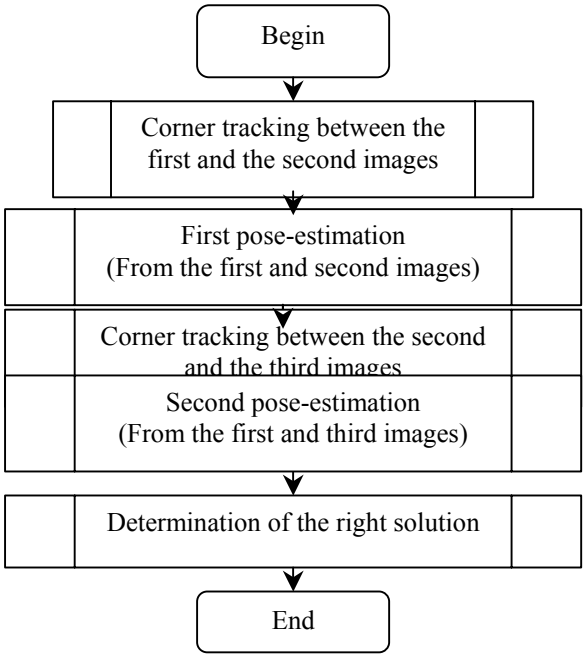
$$P'_1 = a'_1.x + b'_1.y + c'_1 z = 0 \quad \text{with } n'_1 = (a'_1 \quad b'_1 \quad c'_1)^t \quad \text{and here } n'_1 = (0 \quad 0 \quad -1)^t$$

To solve this sign ambiguity, we just apply to the estimated normals the same sign we obtained for the first estimated normal.

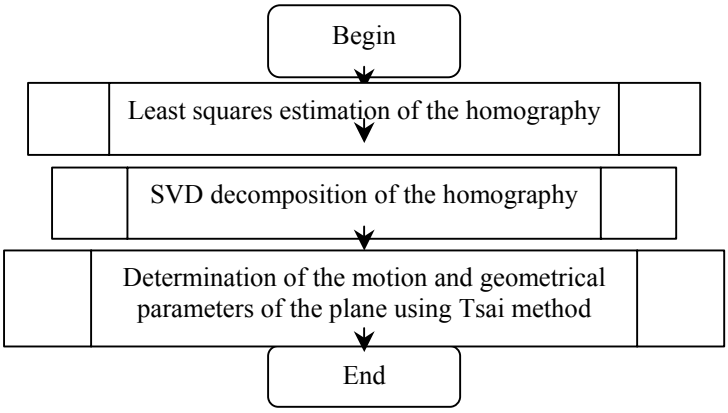
5 – Summary of the Pose-estimation algorithm

The 3D motion estimation of a rigid planar patch algorithm is summarized here with flowcharts.

5.1 - General algorithm



5.2 – Pose-estimation algorithm



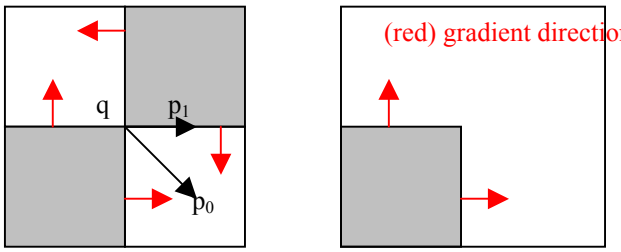
6. Experiments

6.1 – Corner Tracking

We used the corner finding and corner tracker provided by OpenCV. During the initialisation step, if the video sequence contains significant corners, these routines will find and track them. However, we can also choose them manually with the mouse (we can add or remove points). In the following illustrations, the green points are the tracked points.

6.1.1 – Corner detection

The OpenCV function `GoodFeaturesToTrack` finds corners with big eigenvalues of the corner strength matrix. The function first calculates the minimal eigenvalue for every pixel of the source image and then performs non-maxima suppression (only local maxima in 3x3 neighborhood remain). The next step is rejecting the corners with the minimal eigenvalue less than a threshold. Finally, the function ensures that all the corners found are distanced enough from one another by getting two strongest features and checking that the distance between the points is satisfactory. If not, the point is rejected.
For accuracy, one should try to refine the location of the corners to sub-pixel accuracy.



Sub pixel accurate corner

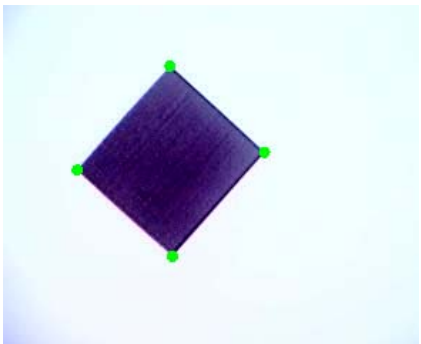
The OpenCV function `FindCornerSubPix` iterates to find the accurate sub-pixel location of a corner.The core idea of this algorithm is based on the observation that every vector from the center q to a point p located within a neighborhood of q is orthogonal to the image gradient at p subject to image and measurement noise. Thus:

$$\varepsilon_i = \nabla I_{p_i}^T . (q - p_i)$$

where ∇I_{p_i} is the image gradient at the one of the points p in a neighborhood of q. The value of q is to be found such that ε_i is minimized. A system of equations may be set up with ε_i 's set to zero:

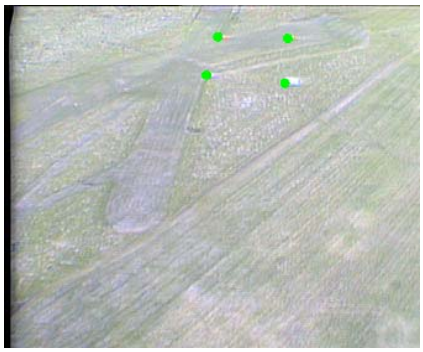
$$\left(\sum_i \nabla I_{p_i} . \nabla I_{p_i}^T \right) \bullet q - \left(\sum_i \nabla I_{p_i} . \nabla I_{p_i}^T . p_i \right) = 0$$

where the gradients are summed within a neighborhood ("search window") of q. Calling the first gradient term G and the second gradient term b gives: $q = G^{-1} . b$. The algorithm sets the center of the neighborhood window at this new center q and then iterates until the center keeps within a set threshold.



Significant corners – easily found.

Low contrast image



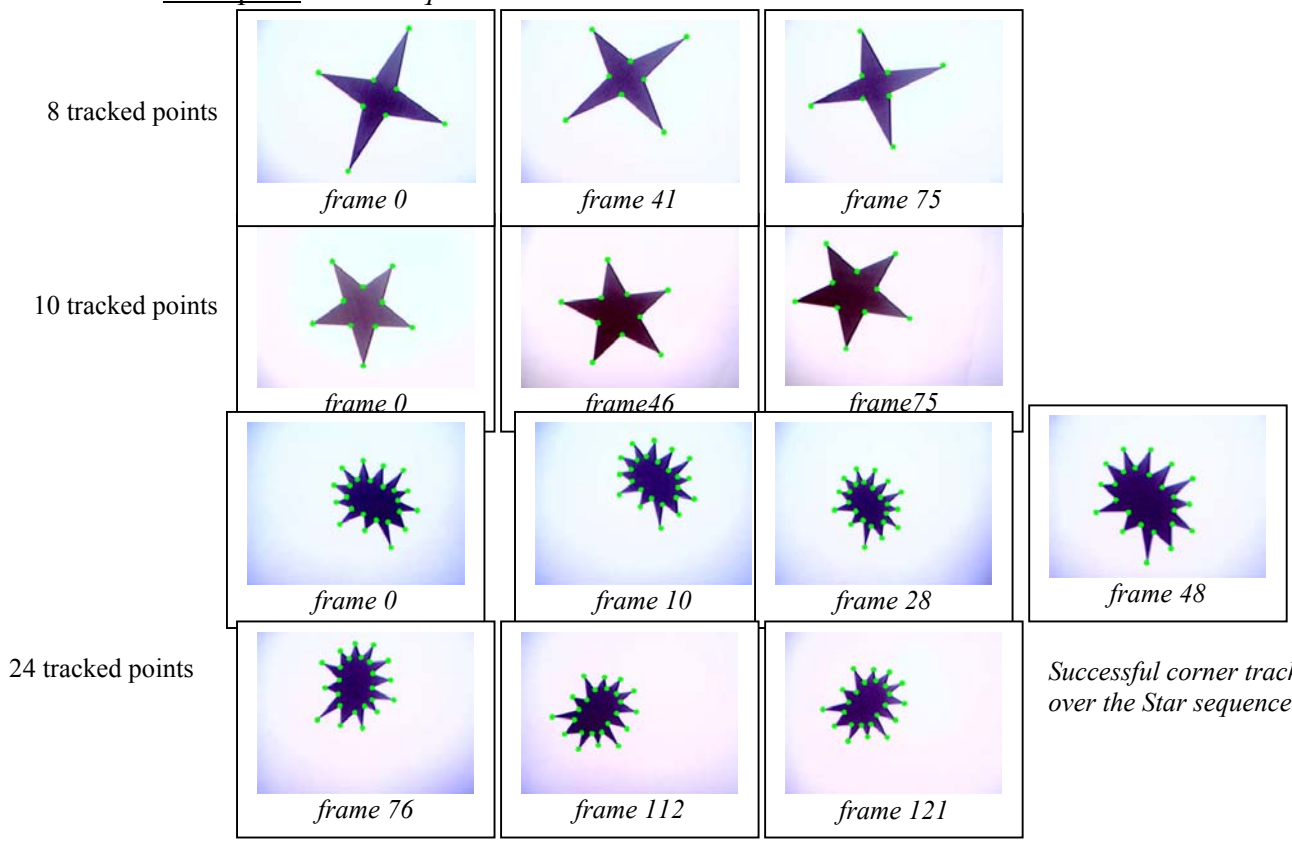
Manual selection of additional features

Automatic and manual selection of corners

6.1.2 – Tracking

The OpenCV function `CalcOpticalFlowPyrLK` calculates the optical flow between two images for the given set of points using a hierarchical pyramid approach to handle large motions. The function finds the flow with sub-pixel accuracy. Both parameters *pyrA* and *pyrB* comply with the following rules: if the image pointer is 0, the function allocates the buffer internally, calculates the pyramid, and releases the buffer after processing. Otherwise, the function calculates the pyramid and stores it in the buffer unless the flag `CV_LKFLOW_PYR_A[B]_READY` is set.

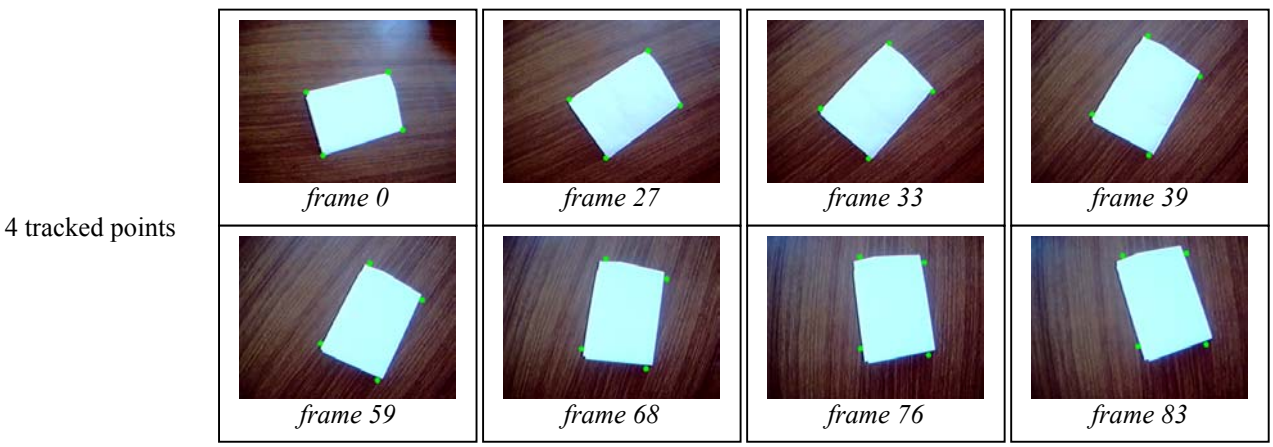
Example 1: "*Star sequence*"



Successful corner tracking over the Star sequence

However, the tracking is not as good when the rotational component of the motion is significant shown below.

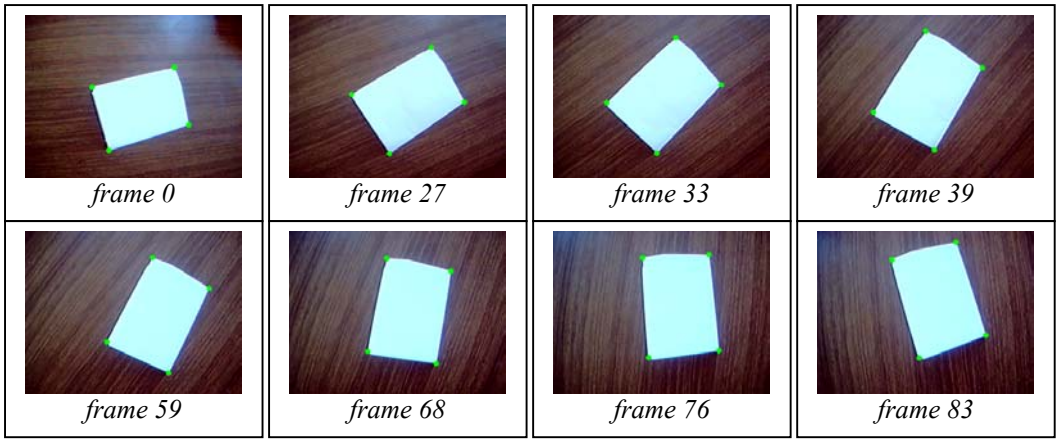
Example 2: "*Rotation Square sequence"*



Problematic corner tracking over the Rotating Square sequence

As we can see, little by little the four tracked points are lost. They are still representing a square, but they are not any more corners of the real square, therefore, the corresponding estimated rotation will be of poor quality. This shows the need for sub-pixel refinement of corner locations. The resulting tracking is of much better quality as shown below.

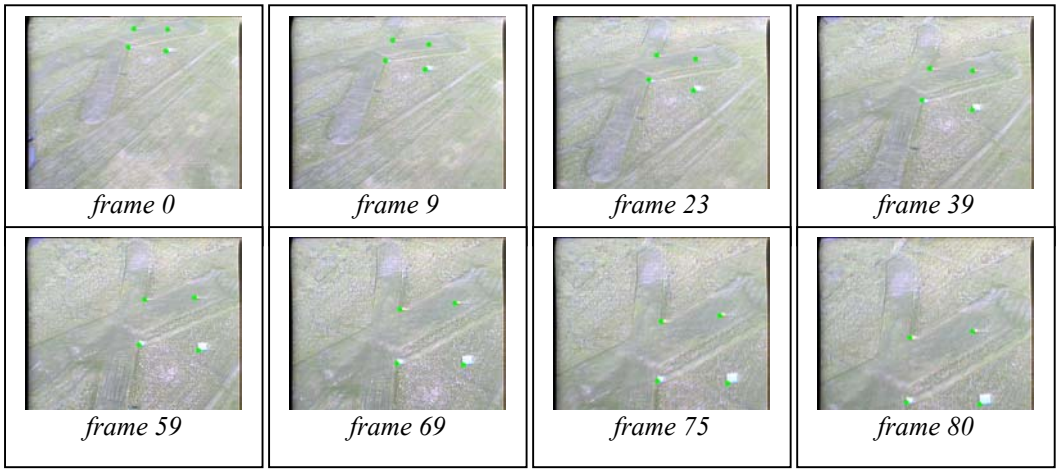
4 tracked points



Successful corner tracking over the Rotating Square sequence

Example 3: "*Aerosonde sequence*".

4 tracked points

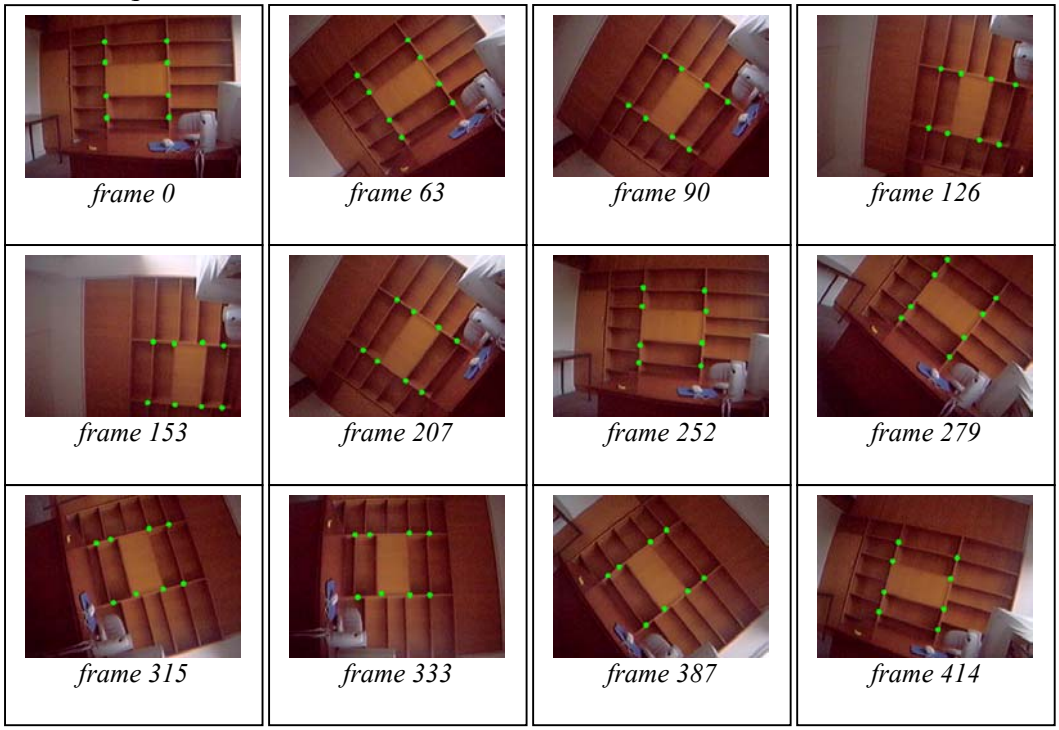


Successful corner tracking over the Aerosonde sequence (with a manual selection of features)

The motion of this sequence is mainly a translation with a slight rotation.

Example 4: "*Indoor sequence*."

8 tracked points



Successful corner tracking over the Indoor sequence (with a manual selection of features).

These various examples show the Lucas-Kanade feature tracker [Shi & Tomasi 94] can perform adequately. However, a refinement of corner locations is indispensable after each tracking step to keep an accurate feature tracking over the sequence.

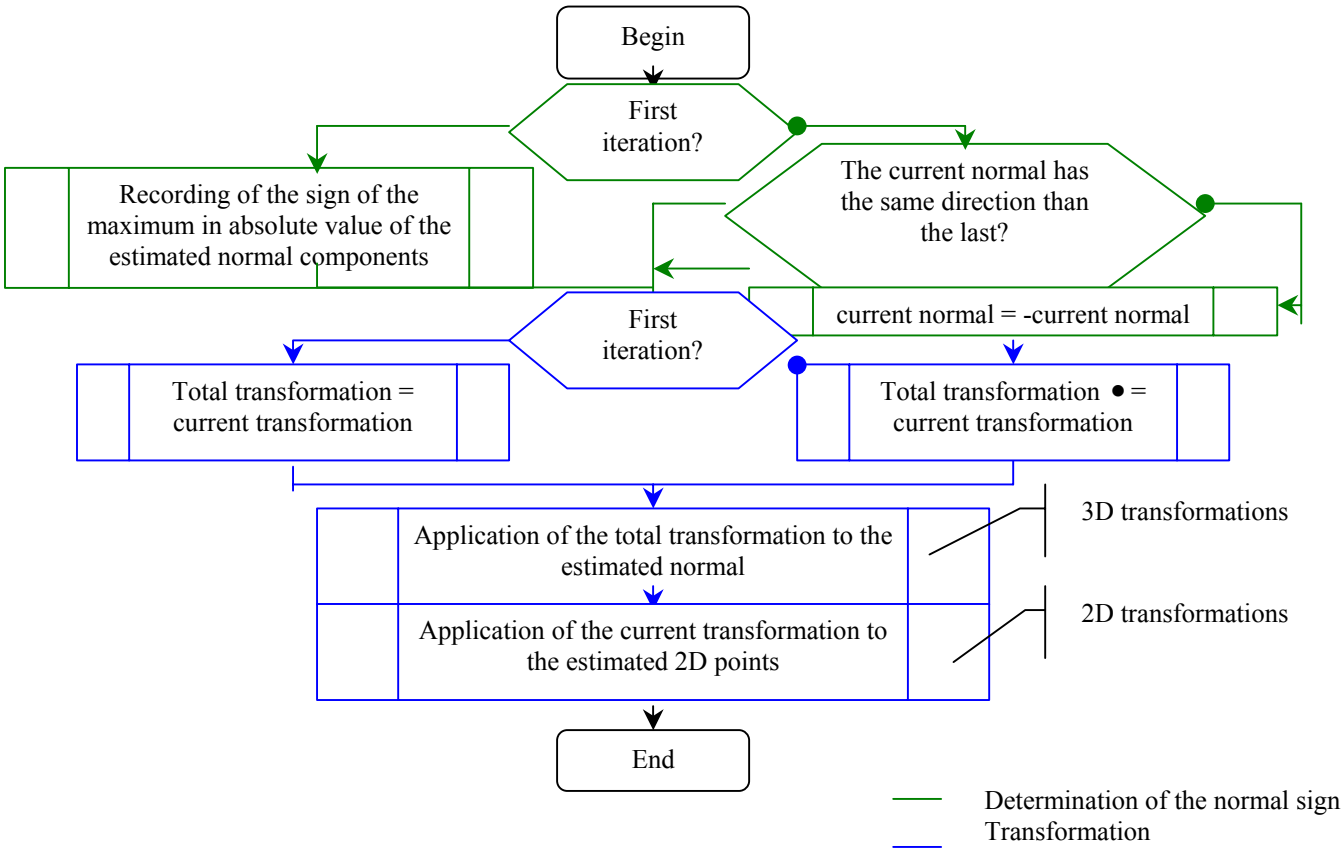
We used the following parameters for the relevant functions:

```
cvGoodFeaturesToTrack(     img0, m_temp[0] ,
                       m_temp[1]      , &m_features[0][0],
                       &m_count       , 0.01,
                       10 );

cvFindCornerSubPix(   img0, &m_features[0][0],  m_count,
                       cvSize(5,5)            , cvSize(-1,-1),
                            cvTermCriteria( CV_TERMCRIT_ITER, 10,
0.1f ));

cvCalcOpticalFlowPyrLK( img0, img1,
                       pyr0, pyr1,
                       &m_features[I][0], &m_features[J][0],
                       m_count , cvSize(10,10), 3,
                       &m_status[0], NULL,
                       cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS
                       ,10,0.1), pyr0_ready ? CV_LKFLOW_PYR_A_READY :
                       0 );
```

6.2 – Pose-Estimation

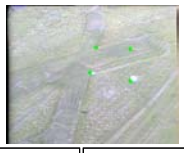
6.2.1 –Transformations

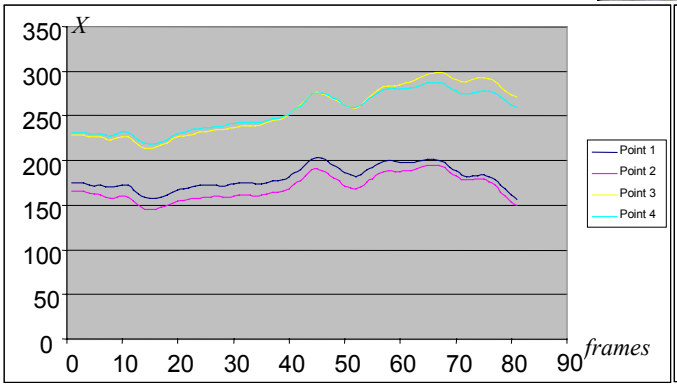
After having estimated the 3D transformation from three images of a planar patch, we obtain the translation, the rotation and the normal parameters to the plane.
For a better illustration and also to show the stability of the pose-estimation process through the sequence, we represent the normal parameters with respect to the initial coordinates system (related to the initial position of the camera). This involves the following.

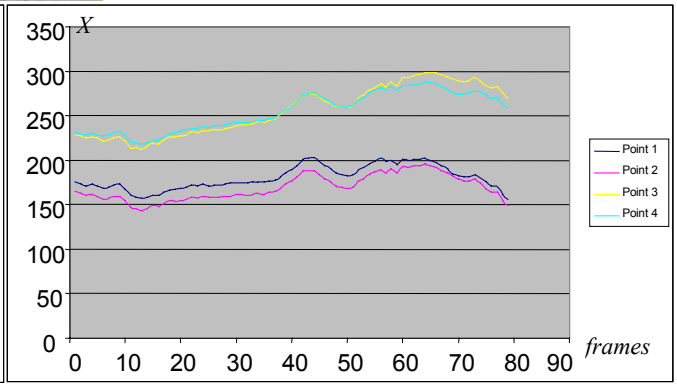


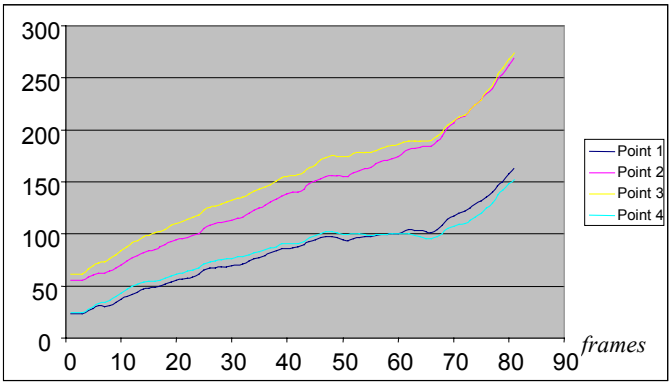
6.2.2 – Estimation results

Example 1: *Aerosonde sequence* with 4 points

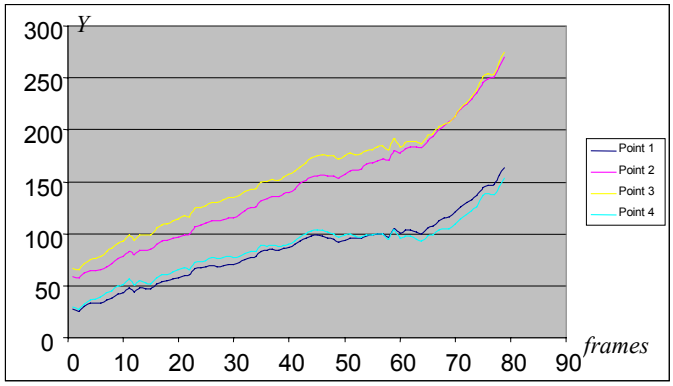






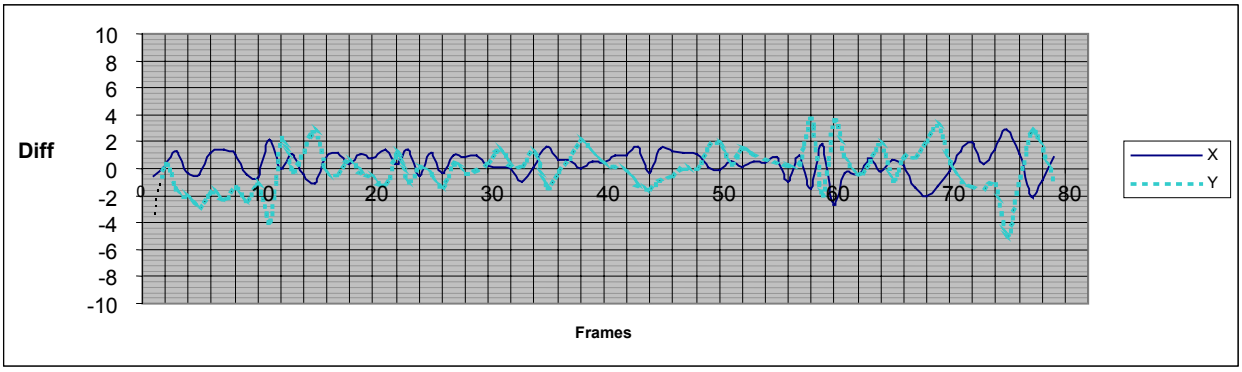


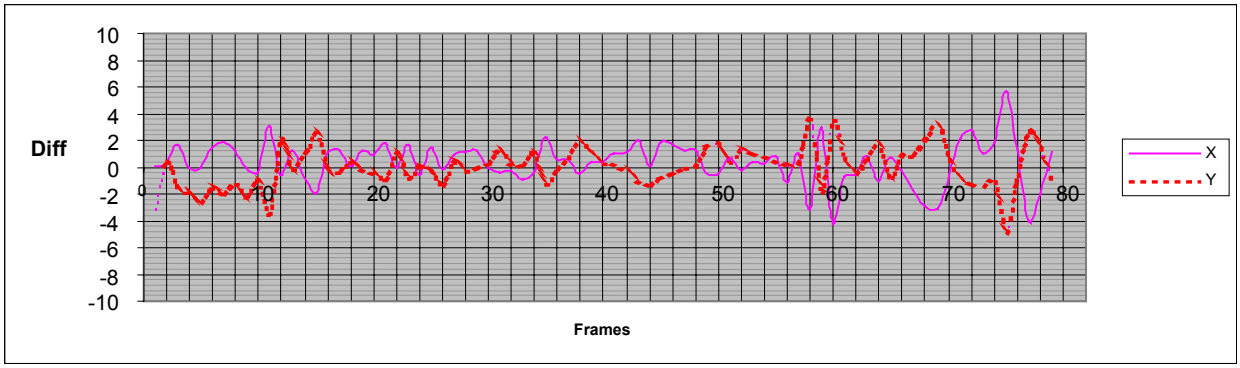
Tracked points

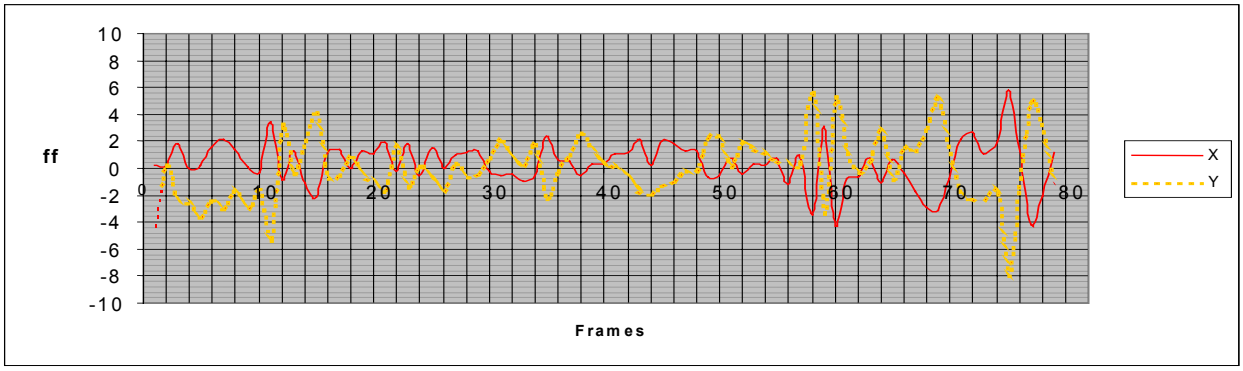


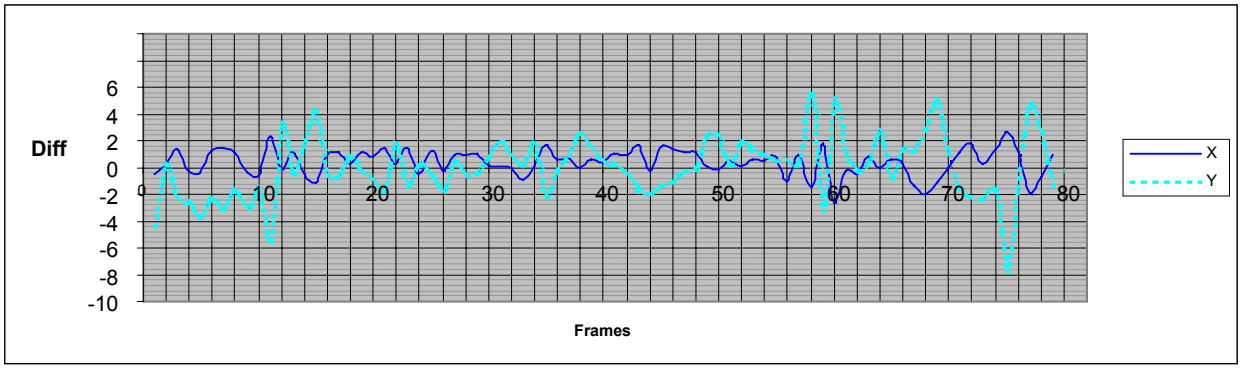
Estimated points

Estimation of 2D points positions from the estimated 3D transformation on the *Aerosonde sequence* (4 tracked points). This is one simple "sanity check" is to re-project and compare with the original tracked points.



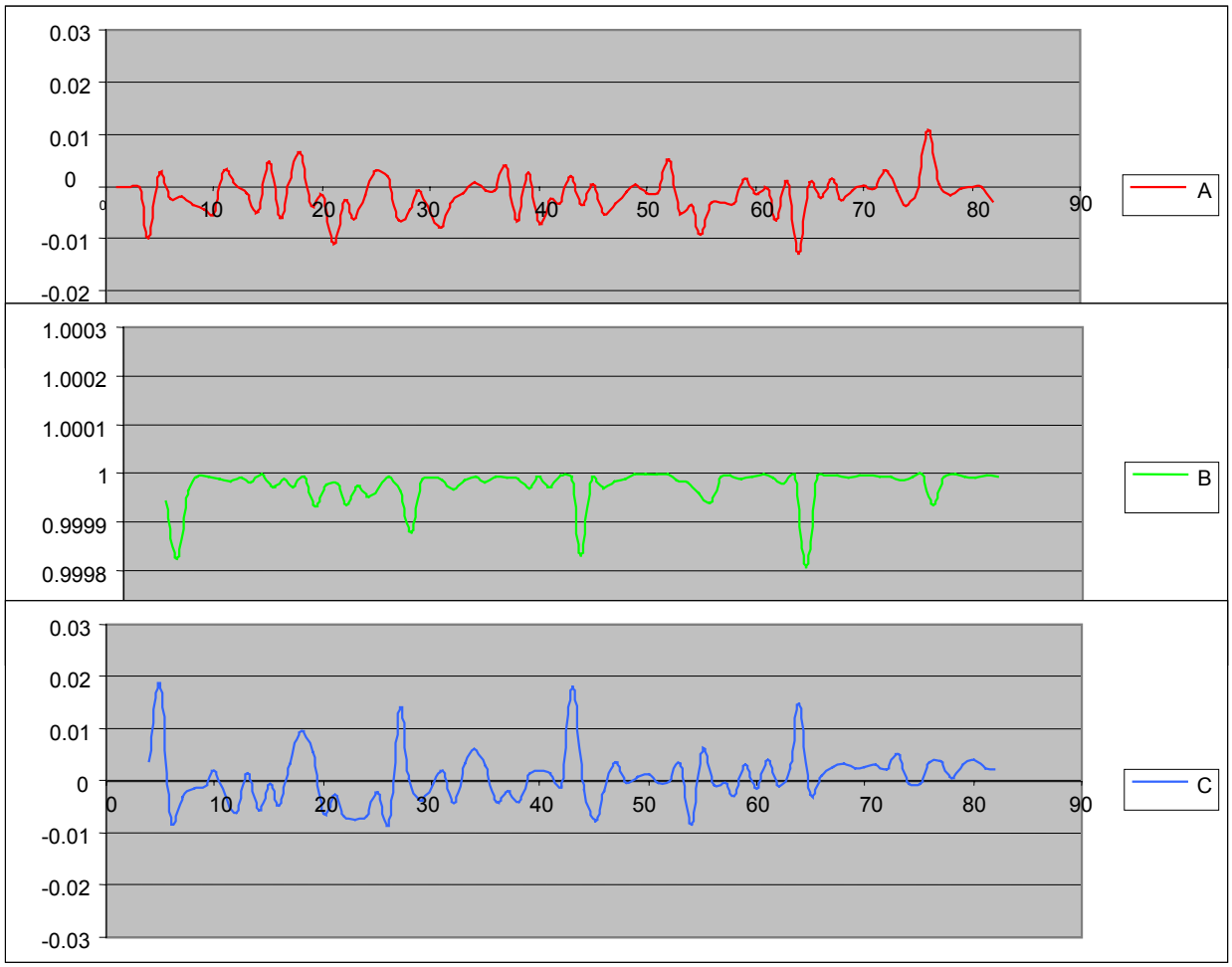






Errors between the tracked points and the estimated points The increase in discrepancy towards the end of the sequence appears to be due to tracking error : it is something that needs investigating further

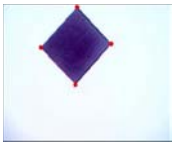
The three components of the estimated normal:

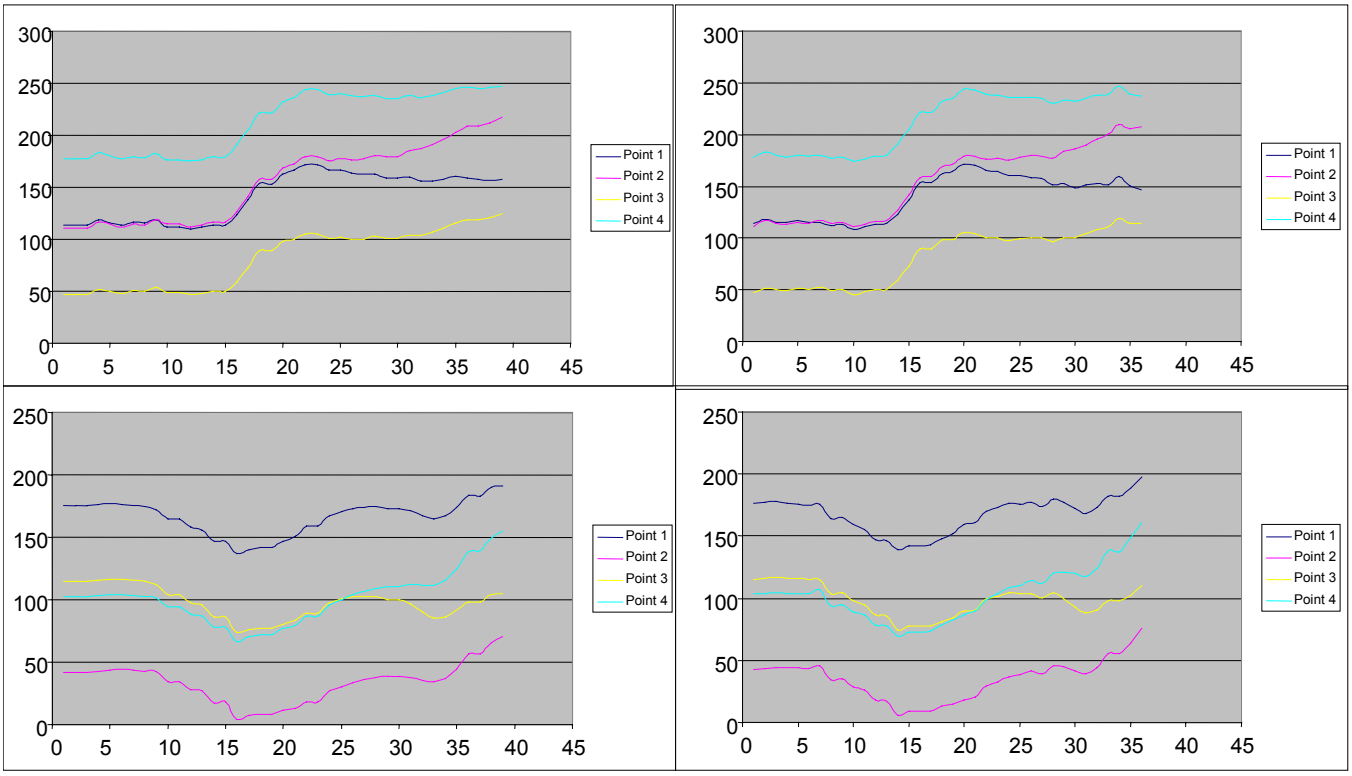


sequence (4 tracked points)

We have normalized the estimated normal. We have obtained a normal close to: $n = (0 \quad 1 \quad 0)^t$. Since the plane did not changed orientation with respect to the ground plane, significantly, during the sequence, these results are encouraging. The standard deviations of the three normal components are σ_A=0.003872, σ_B=0.000037, σ_C=0.005257 which show a satisfactory estimation stability.

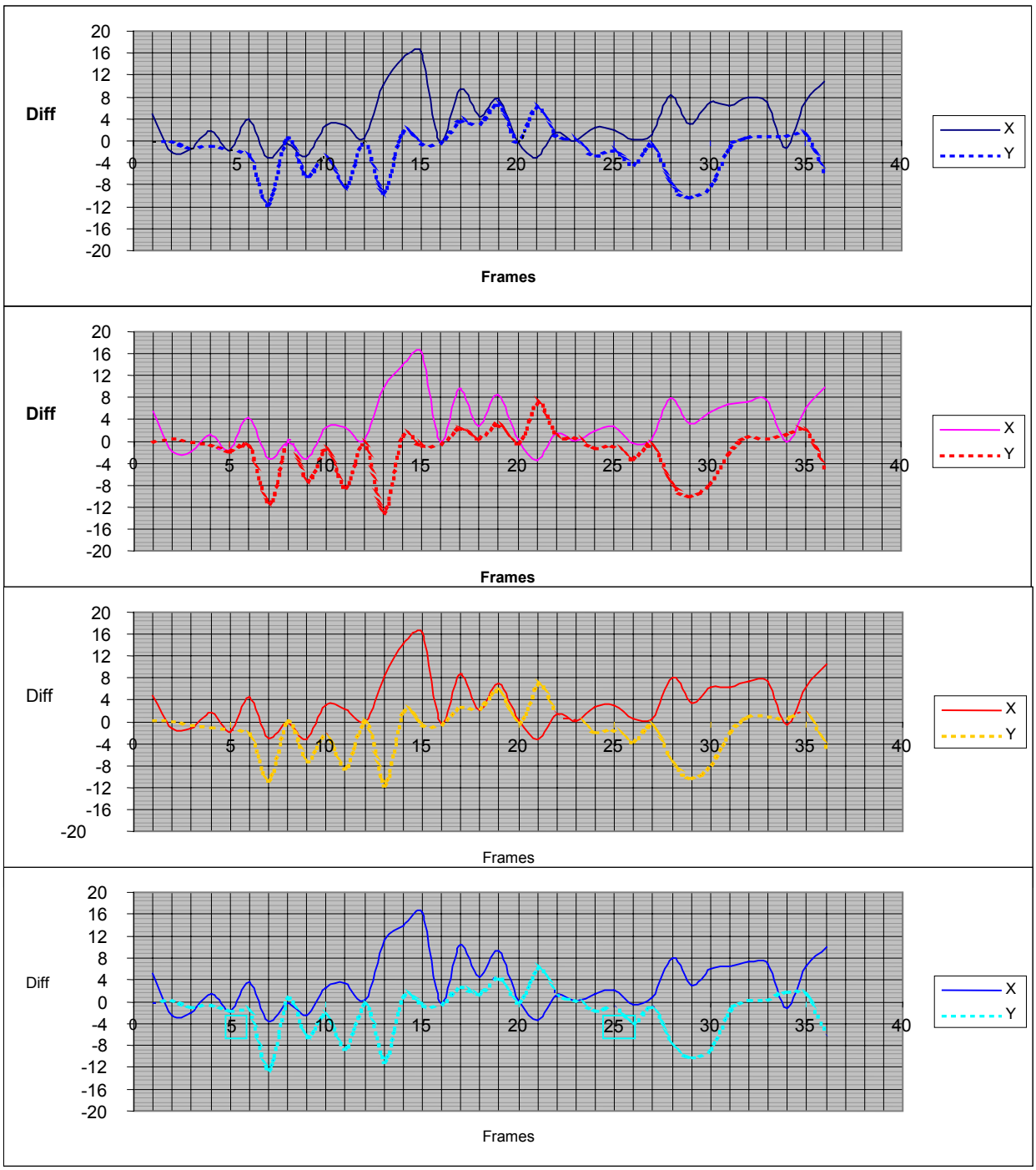
Example 2: *Square sequence*:
Experiments such as the above are not that informative if we do not have precise ground truth. Here we provide results of experiments with more artificial data.





Tracked points *Estimated points*

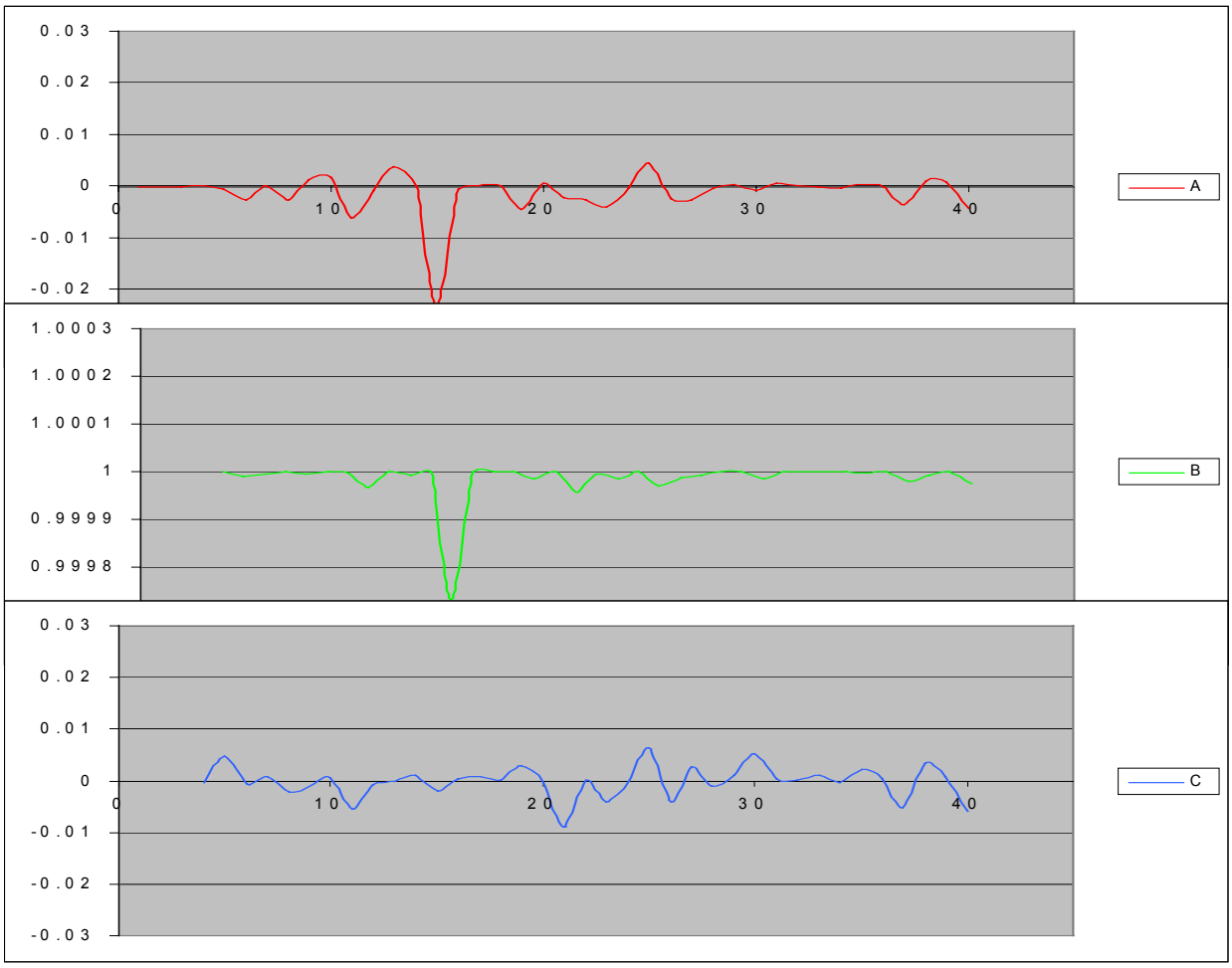
Estimation of 2D points positions from the estimated 3D transformation on the *Square sequence* (4 tracked points).



Errors between the tracked points and the estimated points

Around the frame 15, we can notice that the error between the tracked points and the estimated points is about 17 pixels. This important error is due to the poor video sequence quality.

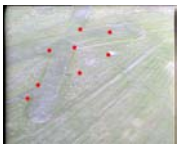
The three components of the estimated normal are as follows:

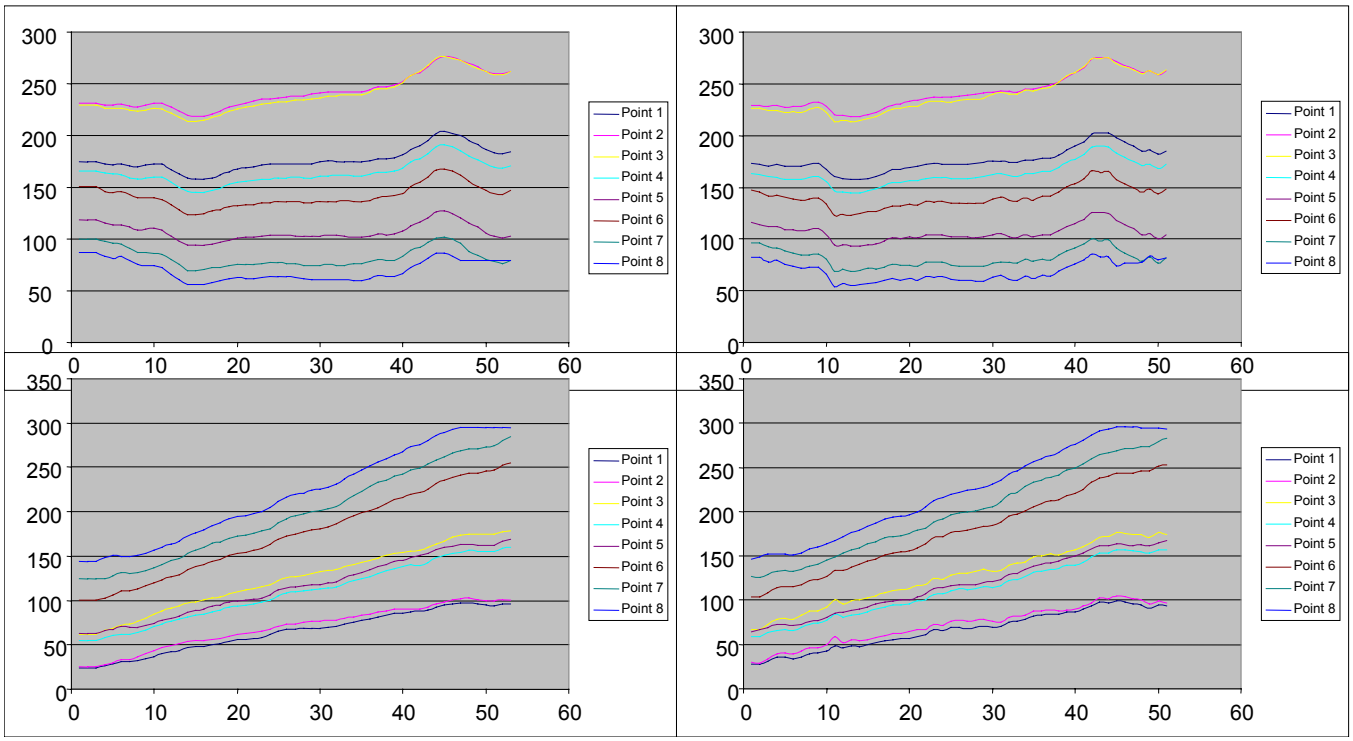


Estimation of the normal $n = (A, B, C)^t$ *to the plane on the Square sequence (4 tracked points)*

The standard deviations of the three normal components are σ_A=0.004188, σ_B=0.000043, σ_C=0.003047, thus the same accuracy is achieved here.

Example 3: *Aerosonde sequence* with several points:

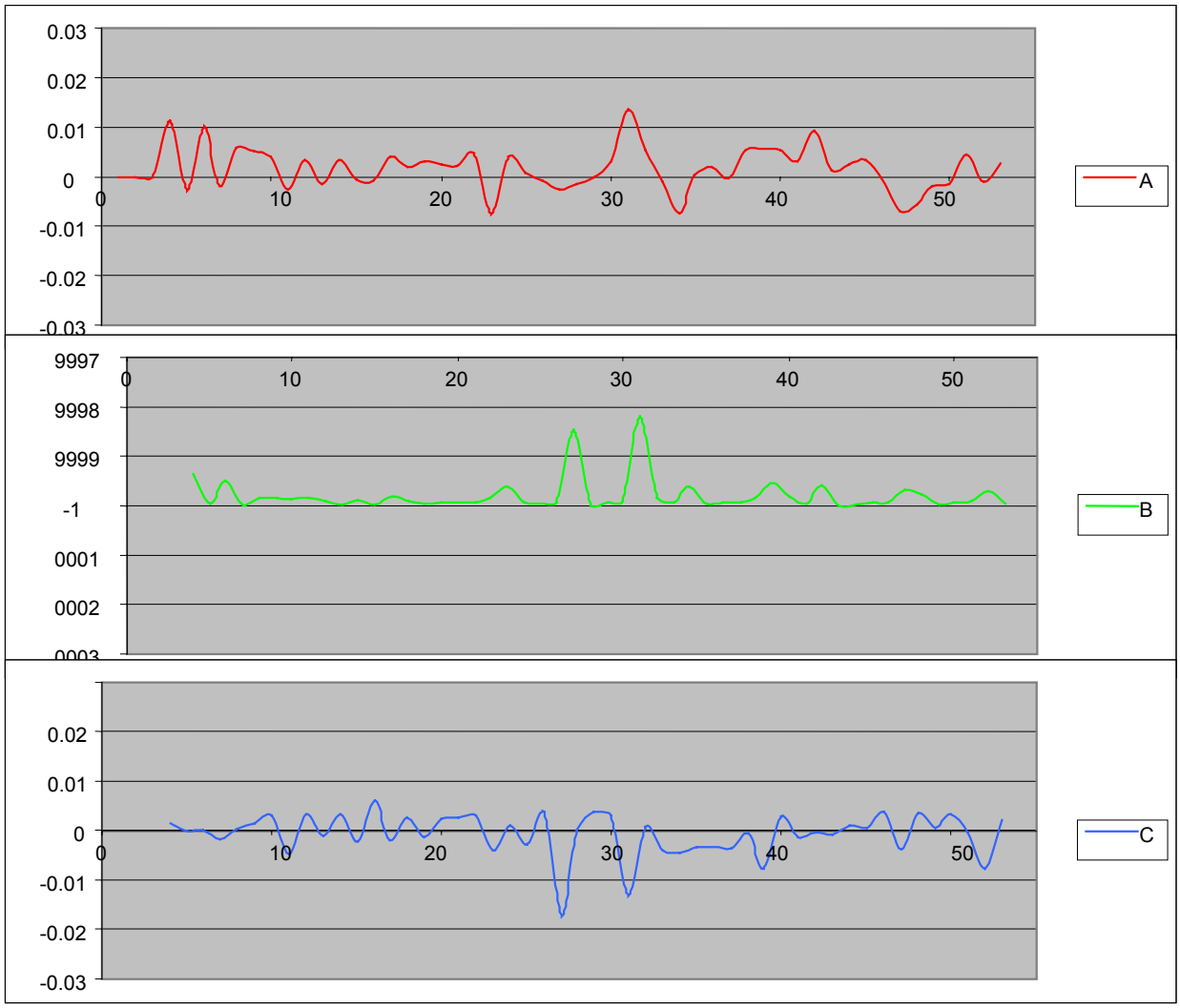




Tracked points *Estimated points*

Estimation of 2D points positions from the estimated 3D transformation on the *Aerosonde sequence* (8 tracked points).

The three components of the estimated normal:

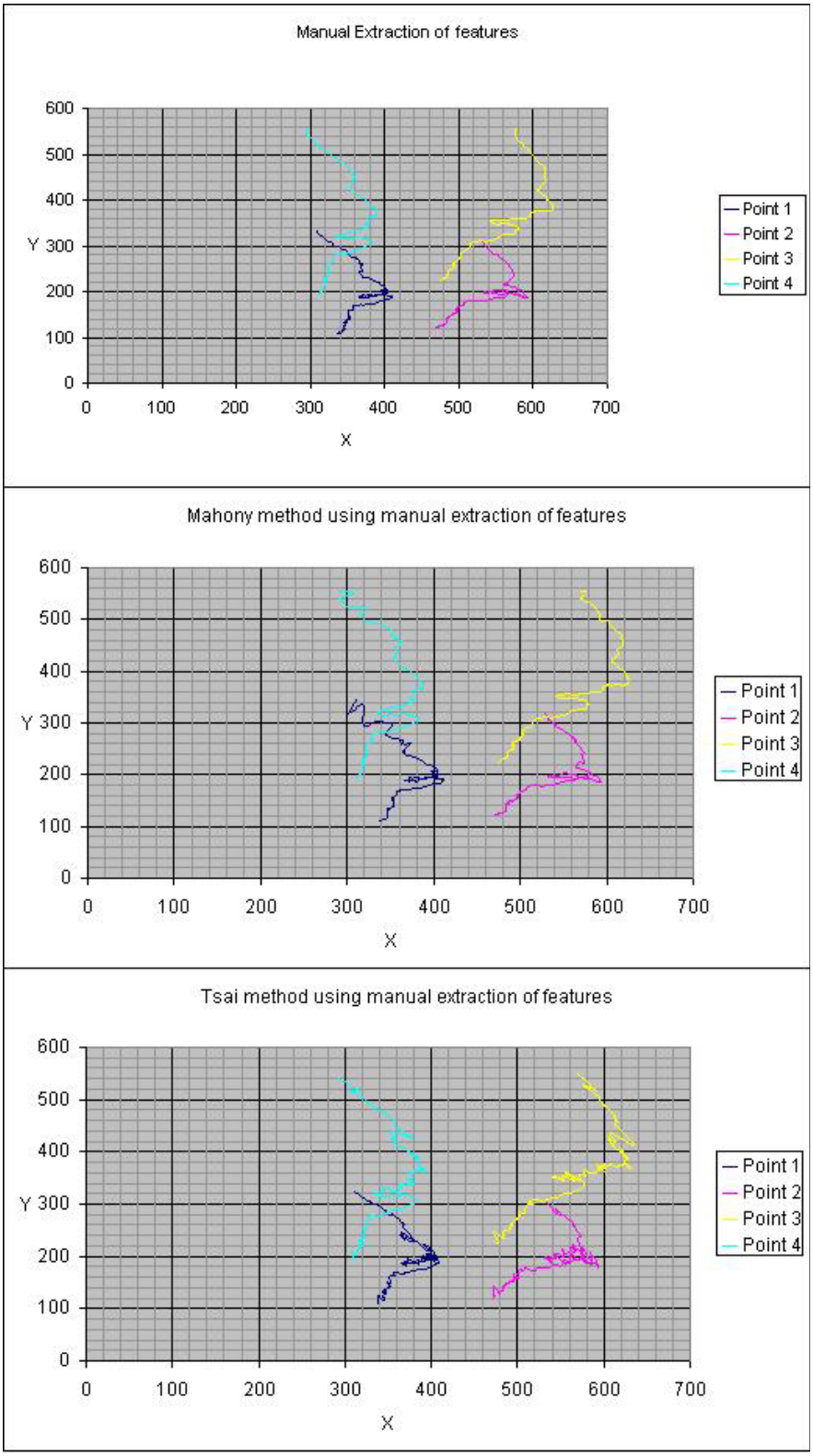


Estimation of the normal $n = (A, B, C)^t$ *to the plane on the Aerosonde sequence (8 tracked points)*

The standard deviations of the three normal components are σ_A=0.008937, σ_B=0.000217, σ_C=0.003686. The results obtained with four points are of equivalent quality as shown in example 1.

6.2.3 - Comparison between Mahony method and Tsai method

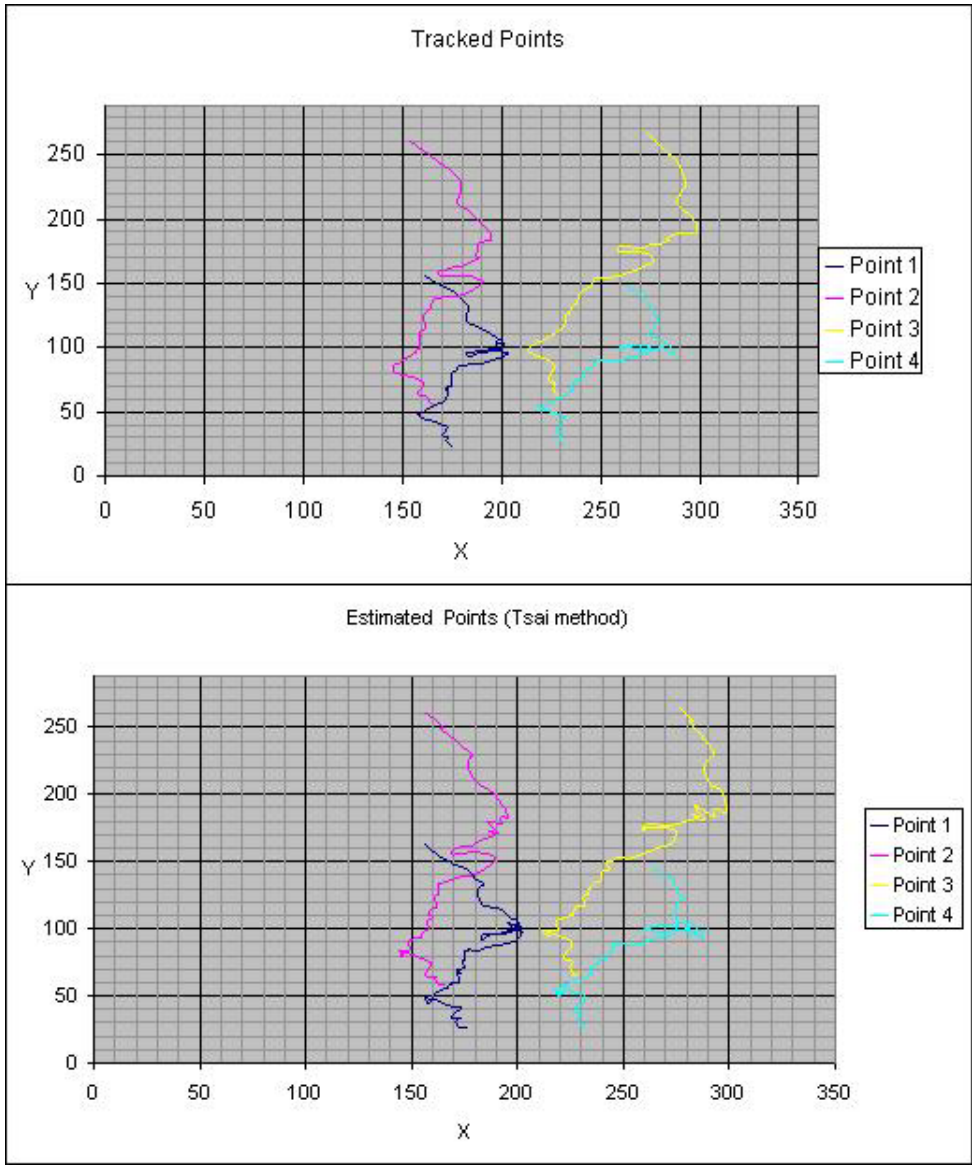
In this section, we compare the pose-estimation results we have obtained using a SVD method [Tsai & Huang 82] with a recursive approach developed by Mahony [Mahony & Suter 01]. The former method offers a closed-form solution to the pose-estimation problem without any filtering stage. It is therefore much less costly than the latter.



Estimated 2D point positions from the 3D transformation on the Aerosonde sequence

On the top, are the trajectories of four point features from the Aerosonde sequence. These features have been manually extracted in 2001. On the bottom, are shown both the pose-estimation results obtained by Mahony method and by Tsai method respectively. We can notice that, in the first iterations, the 2D trajectories estimated by Mahony method are noisier than the results we have obtained with Tsai method. However, because of its recursive nature, the estimated trajectories of Mahony method are significantly improved over time. One could adopt some recursive smoothing into the currently investigated method.

The figure below shows the 2D trajectories we have obtained using the Lucas-Kanade [Shi & Tomasi 94] feature tracker combined with a sub-pixel corner location refinement. These trajectories are smoother than the manually extracted ones and lead to a significant improvement in pose-estimation accuracy by Tsai method as shown on the bottom of this figure.



Estimation of 2D points positions from the 3D transformation on the Aerosonde sequence with Tsai method

Conclusion

We have shown through experiments on real image sequence that the SVD pose-estimation method of Tsai gives satisfactory results in carefully controlled experiments with ground truth available. The Lucas-Kanade corner tracking algorithm combined with a sub-pixel corner local refinement (available in OpenCV library) performed reasonably well even on tough sequences. In this regard, since we have used automatic point tracking, the results and method presented here are more reasonable than the *implementation of the* method to which he hoped to compare [Mahony and Suter 01]. The intention of this work was to ultimately compare the two methods more fairly and conclusively. Moreover, the aim included integrating these approaches with the work of another part of the project [Tung, Suter & Babhadiashar 03]. However, time and other factors, meant that this was not possible.
It is hoped that the software developed here, supplemented by the documentation in the appendix to this report, will be of use for further research in this area.

Acknowledgments

Eric Beets was a fourth year student in computer engineering at IUP (Génie Informatique, University of La Rochelle, France) undertaking this work as part of his "training work" requirements of five months. Samia was a visiting scholar, also from IUP, when this work was performed.

Eric would like to acknowledge the following. Firstly, he would like to thank the Australian team who hosted him during his five months stay in Australia. In particular:

- Gavin BRETT, from Aerosonde LTD
- David SUTER, from the Digital Perception Lab of Monash University

Secondly, he acknowledges:

- Samia BOUKIR, from La Rochelle University, for her assistance, advice and kindness.
- Frédéric CHENEVIERE, for his companionship and wise advice.
- The IUP Génie Informatique, University of La Rochelle, for the formation and the good atmosphere.
- The Région Poitou Charente, for its financial support.

Finally, he would like to acknowledge:

- His parents, for their daily support.
- His friends in France, for their availability (particularly Wilfrid, Baptiste and several others).

This work was made possible by the ARC Spirt Grant C0010681 and the sponsorship by Aerosonde.

References

Software Libraries:

Intel Image Processing Library – Reference Manual http://developer.intel.com
Microsoft MSDN http://msdn.microsoft.com
Open Source Computer Vision Library – Reference Manual http://developer.intel.com
OpenGL Programming Guide http://www.opengl.org

Articles:

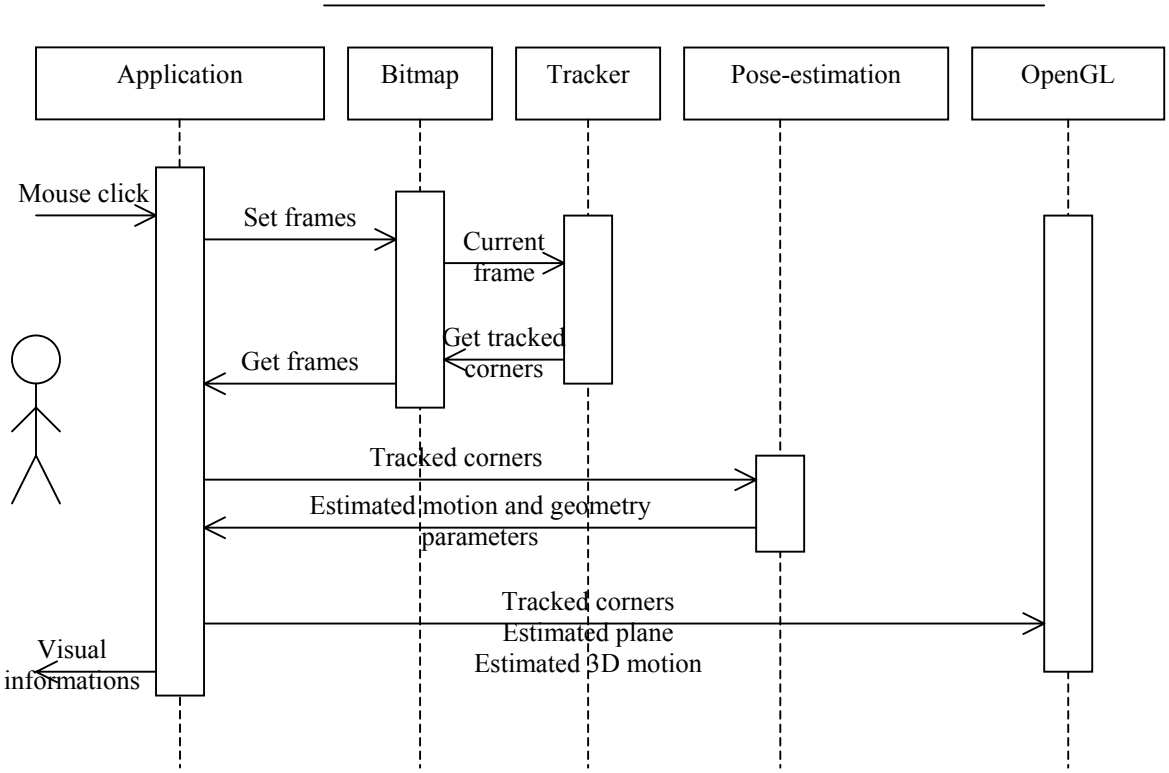
[Faugeras & Lustman 88] O.D. Faugeras, F. Lustman, "Motion and Structure From motion in a Piecewise Planar Environment." International Journal of Pattern Recognition and Artificial Intelligence, Vol. 2 No. 3, 1988 (pp 485-508).
[Mahony & Suter 01] R.Mahony and D. Suter "Tracking Camera Pose from Visual Data using Sequential Newton Iterates". Unpublished manuscript. Note: this algorithm was implemented by Mr.

F. Boissel and Mr. A. Renouf (who were also visitors from La Rochelle sponsored by Aerosonde for the industrial training experience during May-June 2001).
[Tsai & Huang 82] R. Y. Tsai, T. S. Huang, "Estimating Three-Dimensional Motion Parameters of a Rigid Planar Patch, II: Singular Value Decomposition." IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-30, No. 4, August 1982 (pp 525-533).
[Shi & Tomasi 94] J. Shi, C. Tomasi, "Good features to track." CVPR'94, IEEE Int. Conf. On Computer Vision and Pattern Recognition. 1994, (pp 393-600).
[Suter, Hamel & Mahony 02] D. Suter, T. Hamel, and R. Mahony. "Visual servoing based on homography estimation for the stabilization of an x4-flyer",In Proceedings 41st IEEE Conference on Decision and Control (CDC), volume 3, pages 2872-2877, 2002.
[Tung, Suter & Bab-Hadiashar 03] D. Tung, D. Suter and A. Bab-Hadiashar , "Aircraft Approach Angle Estimation: Vision Based Landing, " Monash University Technical Report MECSE-28-2003, Dept. Electrical and Computer Systems Engineering.

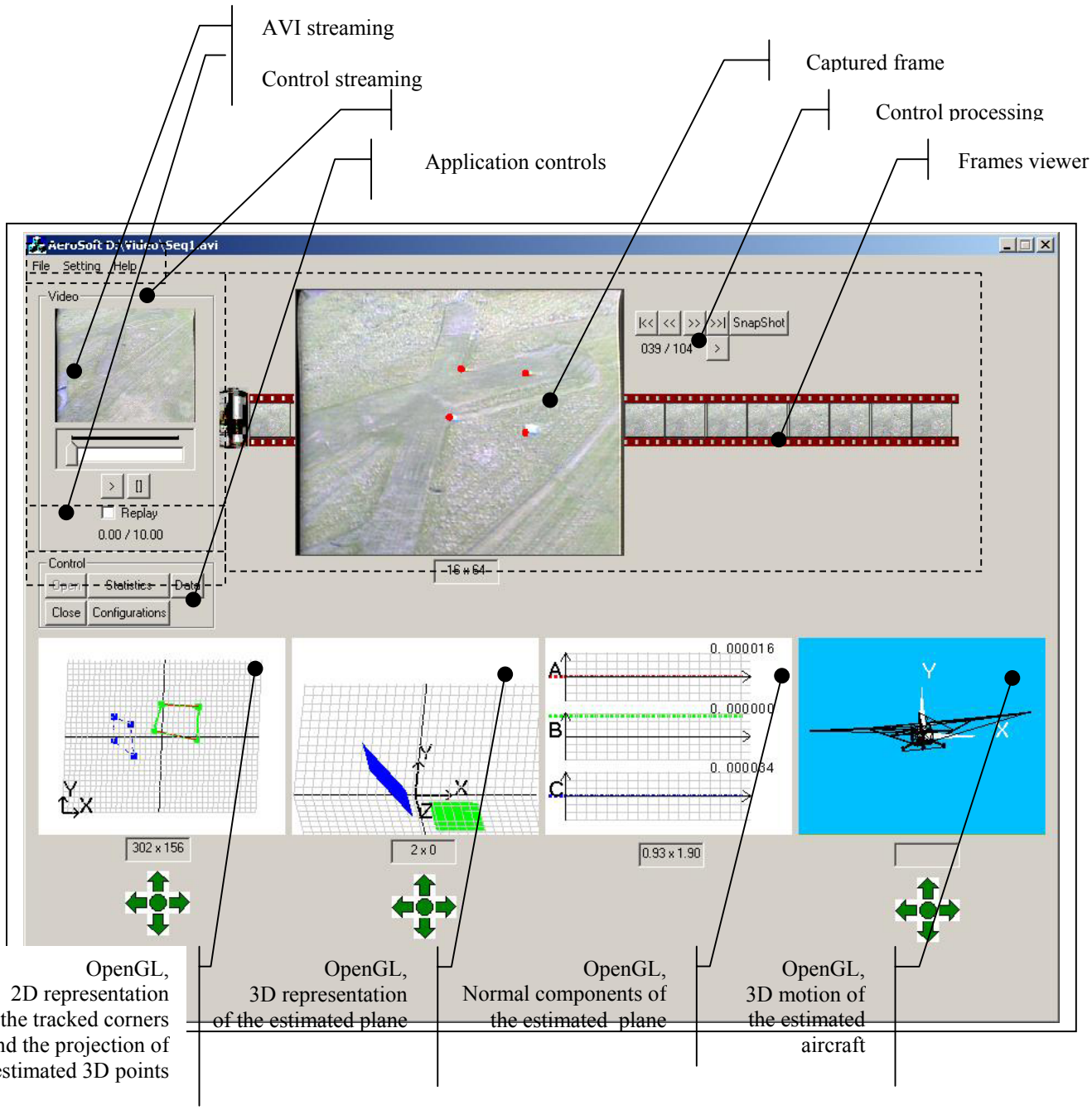
Appendix

User interface and Implementation Details

System sequence diagram:

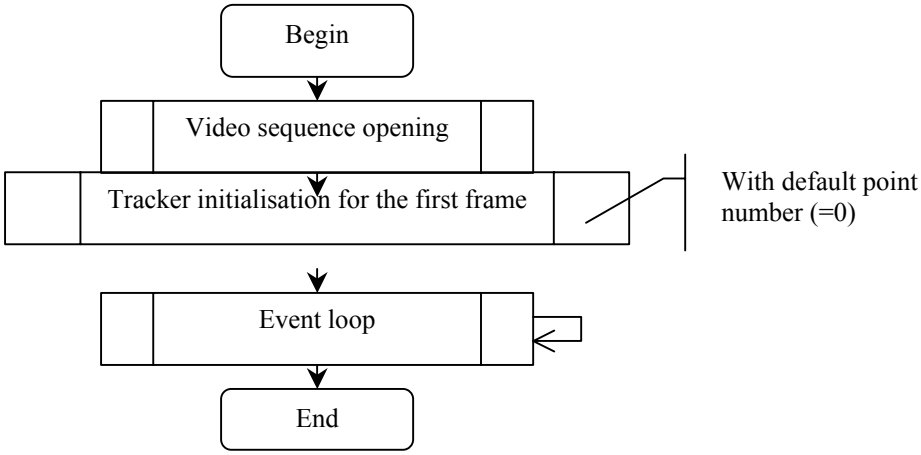


Interface appearance and general flowchart

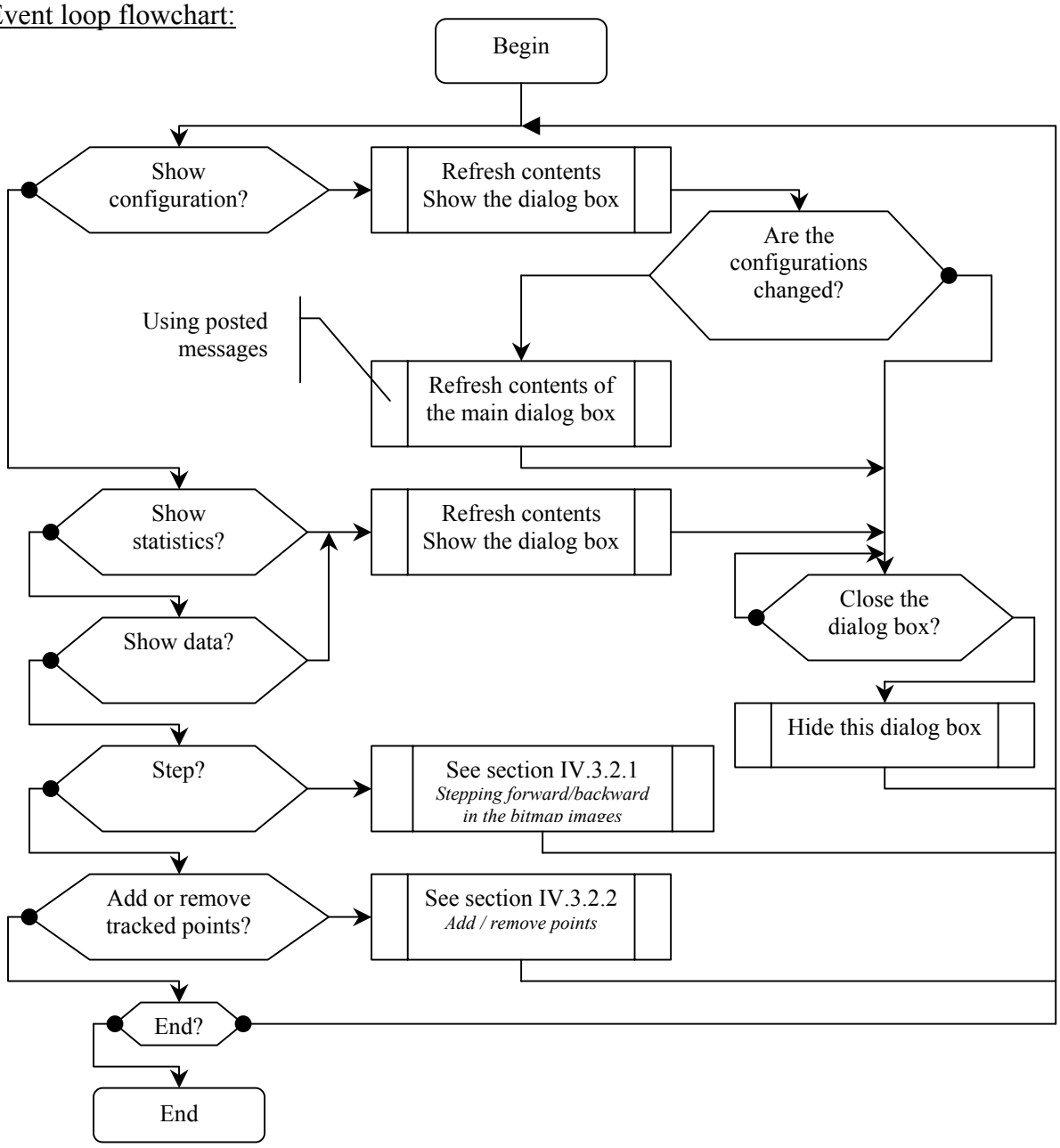


Aircraft pose-estimation interface

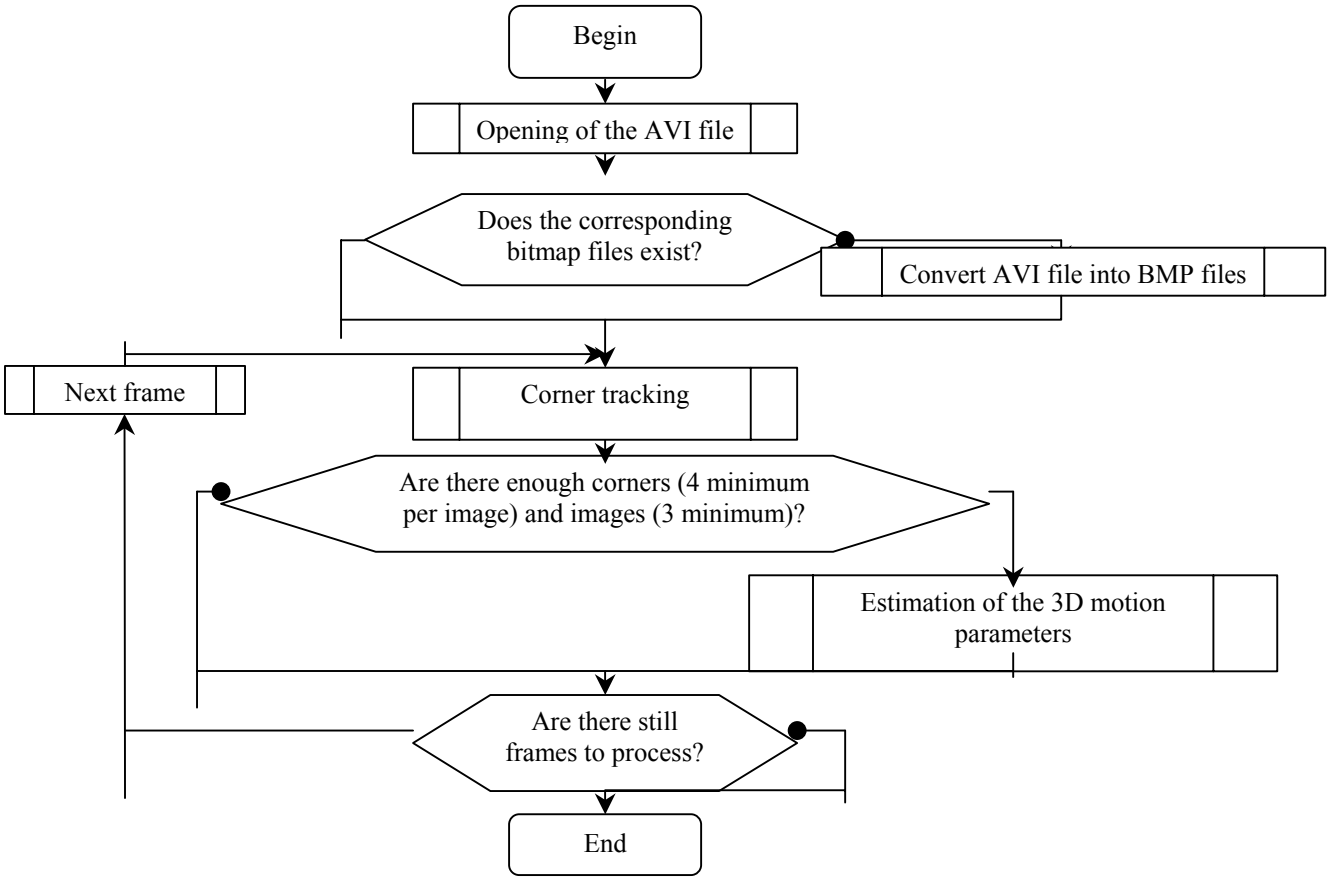
This interface is based on the following flowchart:



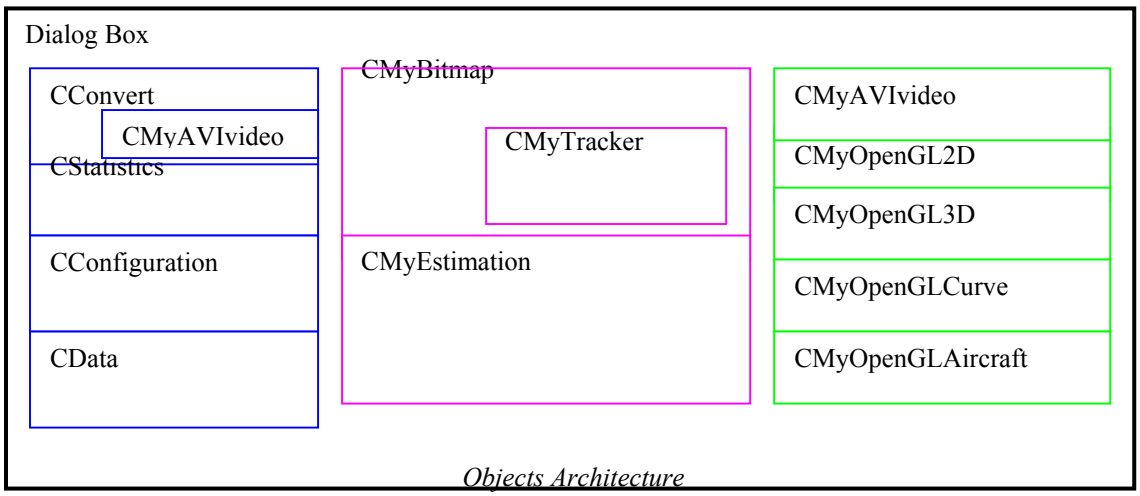
Event loop flowchart:



Video Sequence reading flowchart:



Objects architecture



Objects Architecture

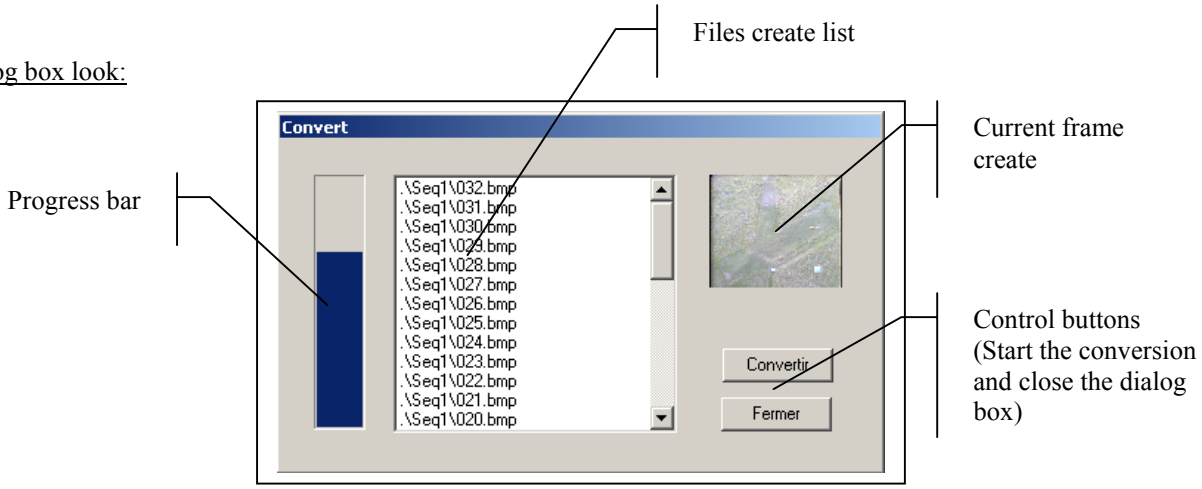
- Control Dialog boxes
- Invisible objects (used for corner tracking and motion estimation)
- Visible objects (used for results representation)

-

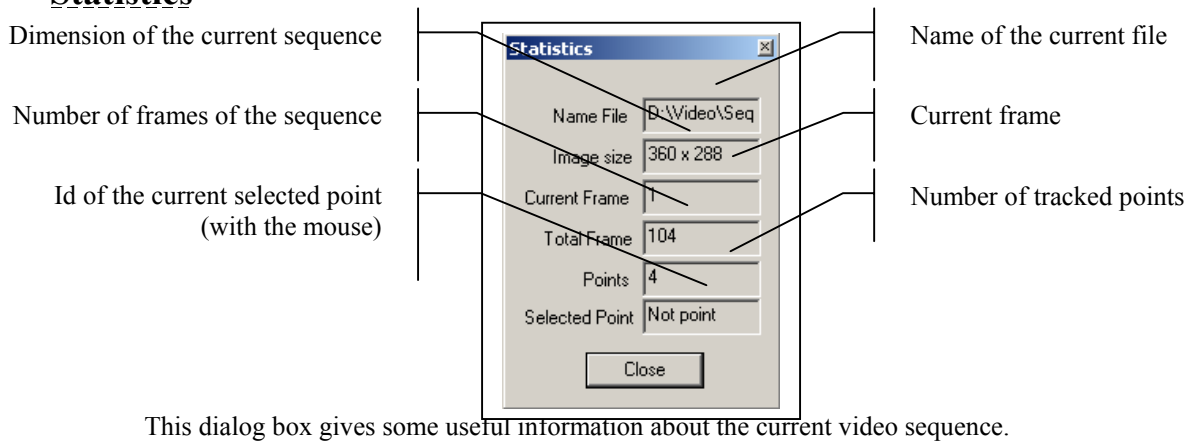
Control Dialog boxes

File conversion

Dialog box look:

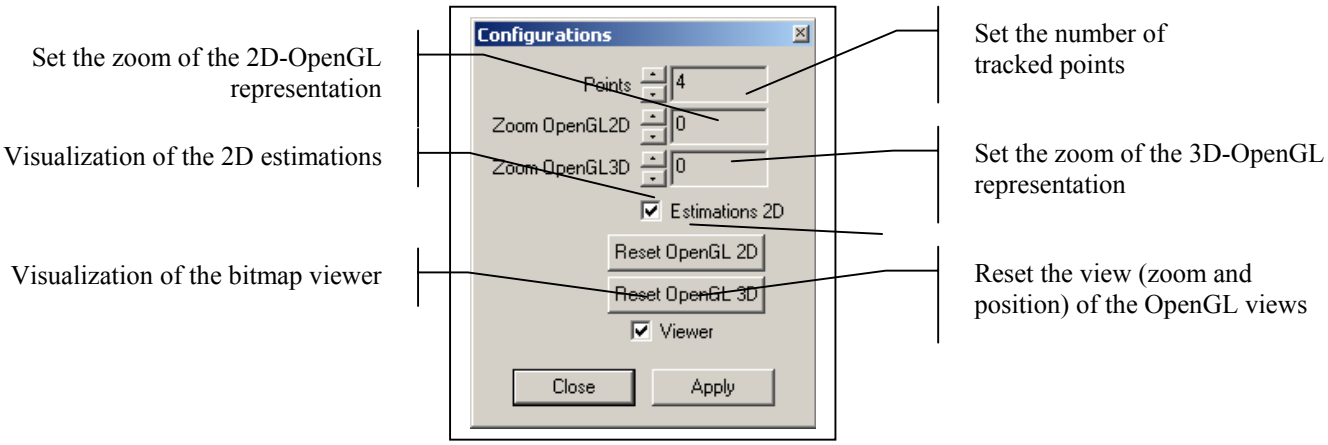


Statistics



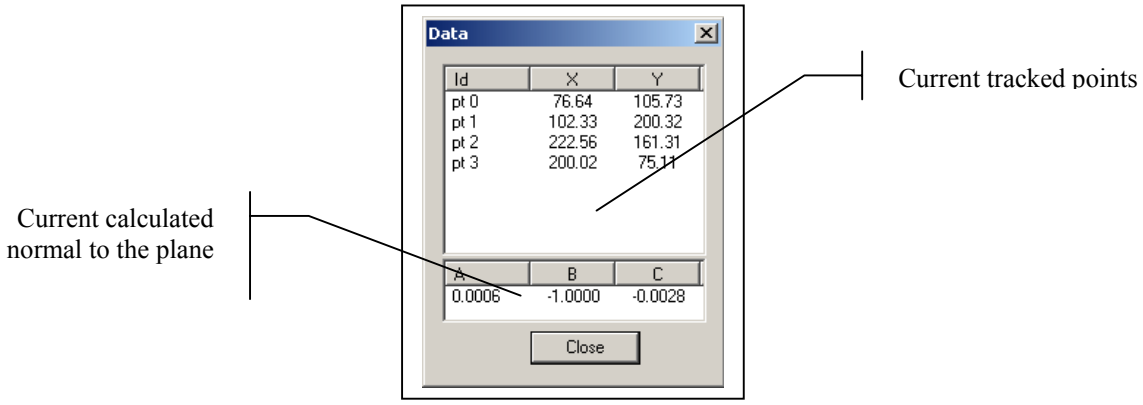
This dialog box gives some useful information about the current video sequence.

Configuration



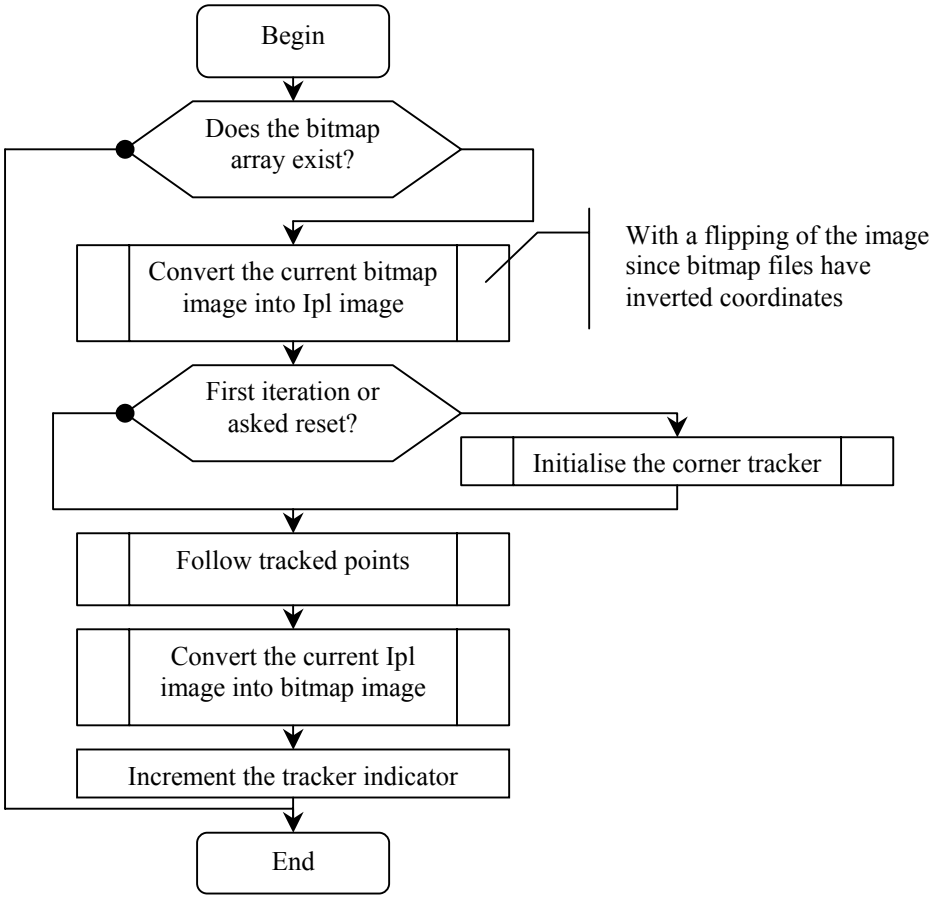
If a new point is added (by a click on the current image), its coordinates are unknown. Therefore, we have to initialise again the tracking with the proper number of tracked points.

Data



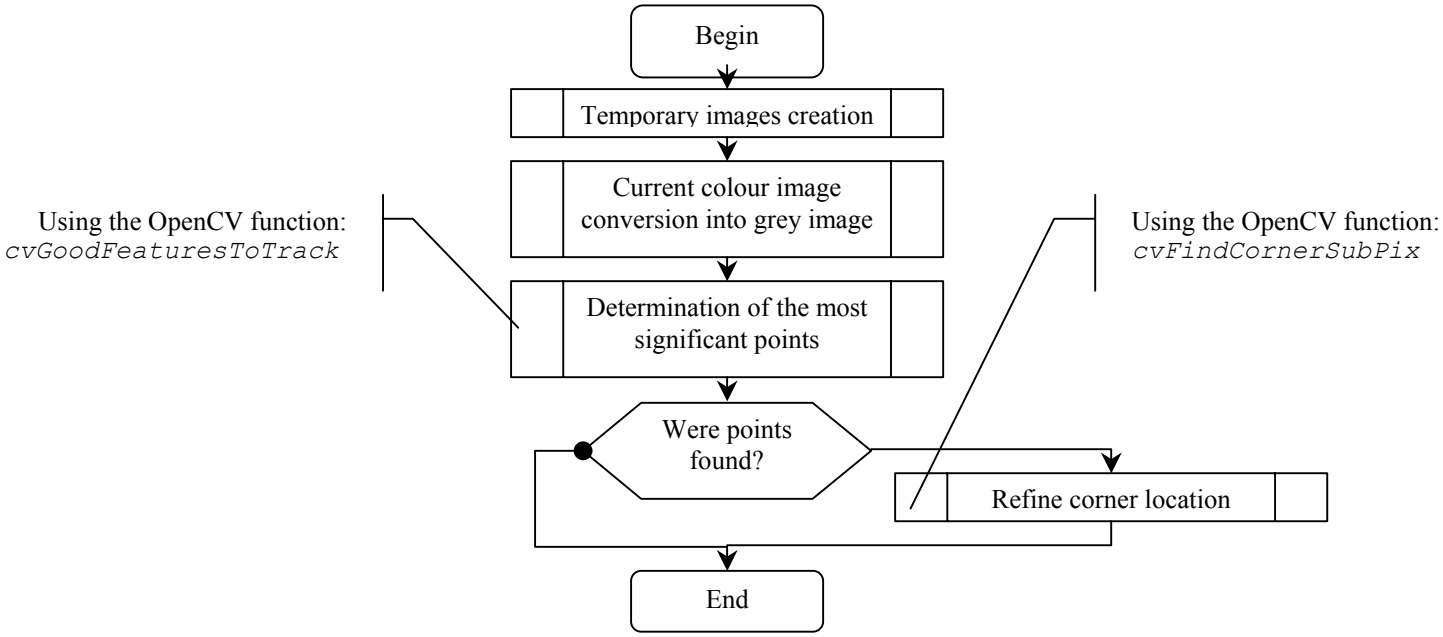
The three dialog boxes are managed with accessor functions to refresh their content.

Corner tracking process:

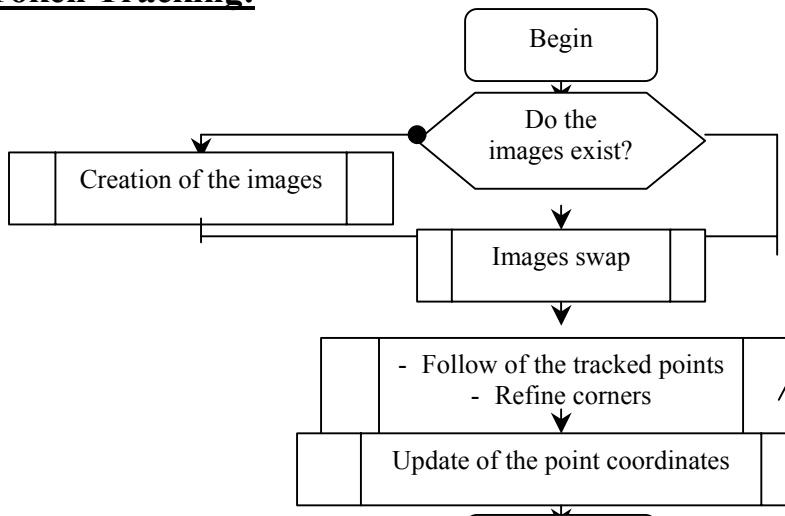


Tracker object

Token Tracker initialization:

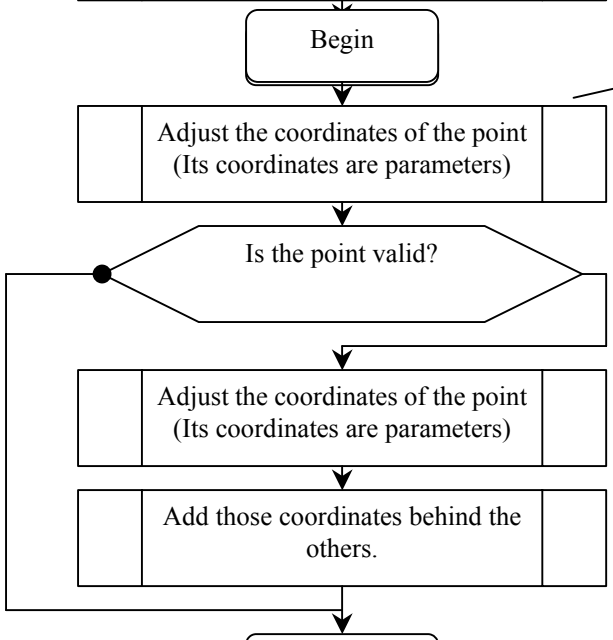


Token Tracking:



Using the OpenCV functions:
cvCalcOpticalFlowPyrLK
and
cvFindCornerSubPix

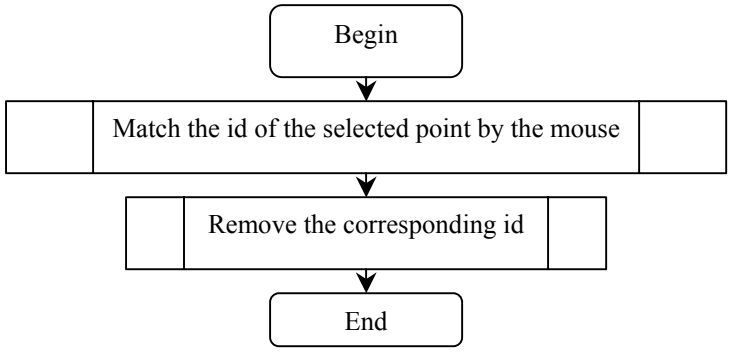
Add a point:



Using the OpenCV function:
cvFindCornerSubPix

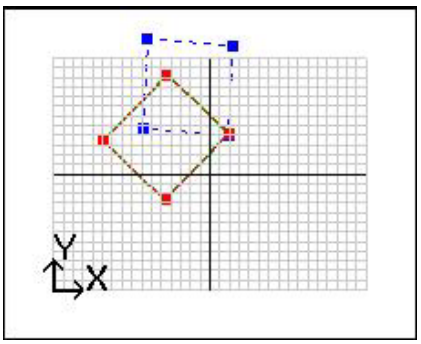
As we can see, we *adjust* the coordinates of the point: for instance, when we use the mouse to add a point (click in the image) we cannot select a corner accurately, so the OpenCV routine adjusts the point coordinates.

Remove a point:



Tracking and Pose Representations

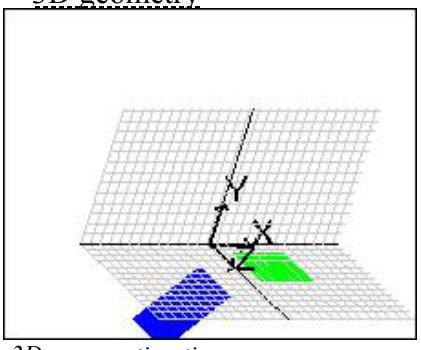
2D perspective projections



2D perspective projections

Tracked points (connected points)
Estimated points (using the estimated 3D transformations)
Wrong estimated points (rejected through the ambiguity resolution).

3D geometry

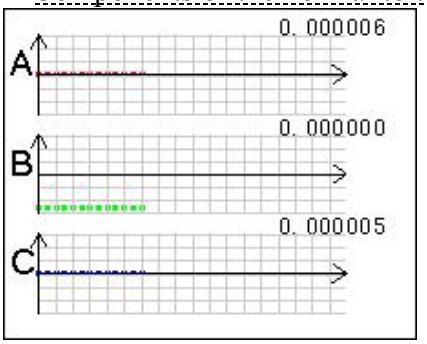


3D pose estimations

The right estimated plane
The wrong estimated plane (rejected through the ambiguity resolution)

In this representation, we can see the estimated normal with respect to the first coordinate system related to the initial position / orientation of the camera .

Components of the normal to the current plane

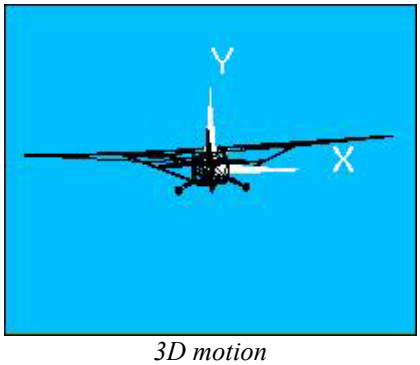


Normal components

First component of the estimated normal: A
Second component of the estimated normal: B
Third component of the estimated normal: C

We can see the evolution over time of the three components of the current estimated normal.

3D motion



3D motion

This last OpenGL representation shows the estimated pose of the aircraft over the sequence.