Autonomous Video Scoring and Dynamic Attitude Measurement

Oleg A. Yakimenko,* Vladimir N. Dobrokhodov,† Isaac I. Kaminer‡
Naval Postgraduate School, Monterey, CA 93943-5146

Robert M. Berlind§
Yuma Proving Ground, Yuma, AZ, 85365

The paper focuses on the development and evaluation of an autonomous payload tracking capability for determining time, state and attitude information (TSPI) of all types of airdrop loads. This automated capability of accurately acquiring TSPI data, will reduce the labor time and eliminate man-in-the-loop errors. The paper analyses the problem and then proceeds with the description of the PerceptiVU Target Tracking System (TTS) software adopted for obtaining the TSPI. The key features of this software include a choice of three basic tracking algorithms (dynamic centroid, hottest spot thresholding, dynamic correlation), capability of capturing from both standard analog video sources (such as NTSC and/or RS170) and digital video sources, control of the entire system with an off-the-shelf joystick controller. The paper further describes algorithms to be used in conjunction with the data provided by the TTS to determine system's state variables. A position estimation solution is based on tracking a payload's center (or any other predetermined point) by several cameras with known positions. A pose (position and orientation) estimation solution is based on tracking of four distinctive non-coplanar points. Pre-selected and artificially marked points on the moving target cooperatively serve as beacons, therefore providing precise measurements of the line of sign toward these points. This allows unique position and attitude estimation and no need for additional pattern recognition. In conclusion, the paper provides examples of video data processing and parameters estimation.

I. Introduction

THIS paper addresses the development of the software capable of determining three-dimensional payload position and possibly payload's attitude based on observations obtained by several fixed cameras on the ground. One of the objectives pursued by the authors is to allow for estimation of system's states needed for the development of advanced flight dynamic modeling and simulation of aerodynamic decelerating systems, without having any sensors onboard. The current method of measuring position and velocity of air delivery payloads is time consuming. After a drop, during which three to four fixed-zoom ground cameras record the flight, each video is manually "read" for payload position in the field of view. This is accomplished frame by frame, with the video reader clicking on a pixel that represents to them the visual "centroid" of the payload. Each video frame has a bar code with the azimuth, elevation and time stamp, so the data for each frame is stored automatically as the pixel is clicked. After the videos are read, the data is processed to determine position at each frame during the drop. The payload position is then numerically differentiated to calculate velocity. The automated capability of accurately acquiring time, state and attitude information (TSPI) will hasten the processing of each video by autonomously tracking the payload once initialized. The track will reference a pixel for each frame, and the pixel will be reference to a lookup table for azimuth and elevation. Since playback will be at a constant rate, only the first frame time is needed for synchronization.

This development of such autonomous capability (TSPI retrieving system) should address the following three independent problems:

- processing of video data itself with the goal of obtaining the frame coordinates of a certain point of the payload, say payload's geometric center, or even several tracking points,

* Research Associate Professor, Dept. of Mechanical and Astronautical Engineering, Code MAE/Yk (oayakime@nps.edu), Associate Fellow AIAA.
† Research Assistant Professor, Dept. of Mechanical and Astronautical Engineering, Code MAE/Dv, Senior Member AIAA.
‡ Associate Professor; Dept. of Mechanical and Astronautical Engineering, Code MAE/Ka, Senior Member AIAA.
§ Senior Engineer, Air Delivery and Soldier Systems Division, Member AIAA.

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

- resolving the position estimation problem assuming that information from two or more cameras is available; and
- resolving position and pose estimation problem assuming that information about at least four noncoplanar tracking points is available.

In what follows Sections 2-4 address each of these problems. Although appropriate software to solve each problem was developed and successfully tested in simulations and with the use of real drops data which was available, additional efforts and more tests are needed to provide a complete product that could be successfully used at the proving ground for autonomous pose estimation. So this paper does not pretend to introduce a complete system, rather it presents results of the preliminary analysis of hardware and software capabilities to address the problem of autonomous video scoring and dynamic attitude measurement.

II. Video data processing

Consider diagram in Fig.1. It shows the task flow of the video data processing system. The key element in this system is an offline version of vision-based Target Tracking Software (TTS), which provides tracking of a selected object in a local camera coordinate frame based on the analysis of a sequence of video frames.

Video signal processing consists of the *video recording*, *fragmenting*, and *processing* steps.

Video recording addresses two principal issues. The *first* is the acquiring of live video data, and the *second* is removal of out-of-frame events. This provides initial video stabilization required at the fragmentation stage. For the stabilization purposes one can utilize two different techniques depicted in Fig.2. The key idea consists of stabilizing a raw video stream either by its recording in the digital format or by using an external video stabilizing unit. At the moment, the Pinetron PDR-H401 (http://www.pinetron.com) is used to stabilize analog video signals.

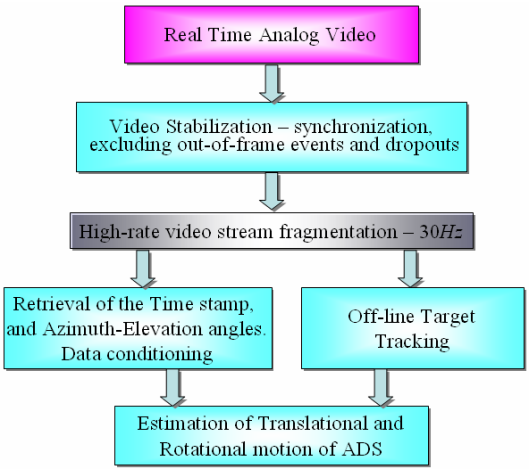


Figure 1. Block diagram of video data processing system.

Video fragmenting capability is supported by a simple "Video Snapper" utility provided by PerceptiVU Inc (www.PerceptiVU.com). It takes stabilized video and fragments it at the rate of up to 30 frames per second. To achieve the highest possible rate it uses low-level driver support provided by the Matrox card. High fragmentation rate allows for easy reconstruction of dynamics of the descending system. Separate frames are saved as the gray scaled bitmap files ready for the next, processing step. The files are named in ascending numerical order thus providing an additional synchronization capability.

Hardware implementation at this step utilizes Matrox Meteor II framegrabber (http://www.matrox.com/imaging/products/meteor2/home.cfm) installed in a high performance computer: Pentium 4/3.2*GHz* processor with 800*MHz* front side bus and 1*Gb* RAM, and 250*Gb*/7200*rpm* hard drive. This setup together with a framegrabber allows stable fragmentation delivering 30 frames/second of high resolution imagery. Video Snapper capability allows the following key features: *i*) capture from standard analog video sources (NTSC, RS170) as well as 1394 digital video sources; *ii*) capture and digitally save pictures or a movie sequence of pictures; *iii*) control the GUI with a mouse pointer. Figure 3 shows a screenshot of the Snapper GUI.

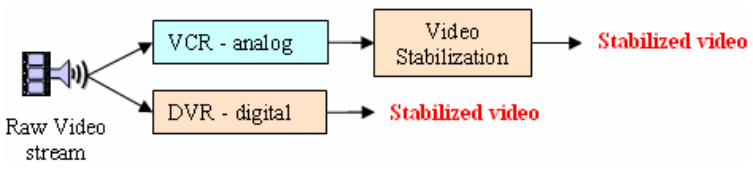


Figure 2. Stabilization of the raw video stream.

Video processing step includes two algorithms. The first one reads bar-coded information (BCI) containing the GPS time stamp and azimuth/elevation orientation of the camera's line-of-sight (LOS) (the barcode is imprinted into the bitmap image as seen on the left in Fig.3). This BCI algorithm developed at the YPG by Wade Porter happens to be quite sensitive to the contrast and sharpness of the video frames at the location of the bar-code. The algorithm

uses thresholding technique to read binary information and therefore, its decoding capability depends on the difference between the white and black strips in the bar code.

The second algorithm uses off-line version of the vision-based TTS developed by the PerceptiVU. Several MatLab scripts allow reading the TTS and BCI results, data conditioning and synchronizing. The data obtained by these scripts is further passed to the pose estimation algorithm.

Due to the various polarity and target tracking options provided by the TTS software its performance is less sensitive to the quality of the picture. Vision-based TTS software also provides following features:

- Three tracking algorithms available:
  - Dynamic centroid;
  - Hottest spot thresholding;
  - Dynamic correlation;
- Three polarity options available:
  - White hot;
  - Black hot;
  - Auto polarity;
- Five fixed target size and a custom size options are available:
  - Custom size;
  - Very small target;
  - Small target;
  - Medium target;
  - Large target;
  - Very large target;
- Interactive tracking gain;
- Allows for data logging in ASCII file.

An example of the TTS GUI is presented on the Fig.4.

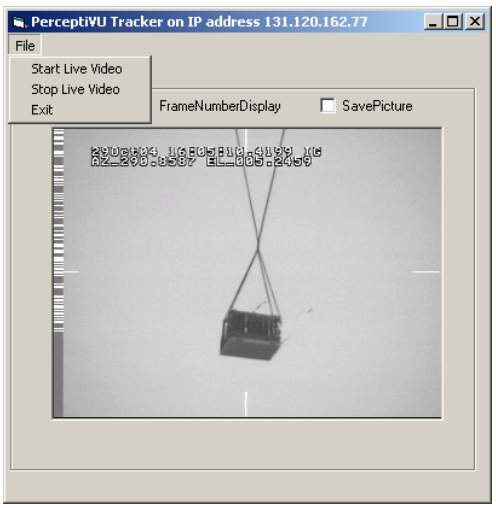


Figure 3. Video Snapper GUI.

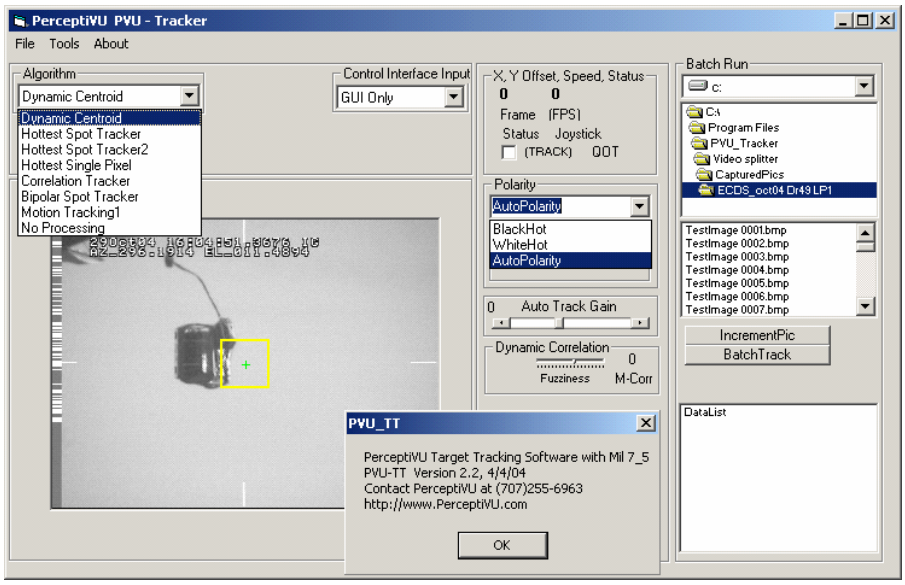


Figure 4. PerceptiVU TSS GUI.

Several utilities and M-scripts were developed to support collection of the data for the pose estimation algorithm. The block diagram in Fig.5 shows how the developed utilities support the video data processing described above (and shown in Fig.1).

As discussed earlier, initially the synchronized video stream is fragmented by PerceptiVU video snapper. A set of grayscale bitmap image files is produced at this step. In order to be automatically processed by the TTS software these files should be renamed. Each file name must be of the same length and should consist of a permanent part that can be determined by a user and a number in an ascending order. The renaming procedure is written as an M-script and its result is presented at Fig.4 (see the enumerated set of files in the GUI).

At the next step this renamed set of files is used twice. First it is used to read the bar-coded data from each frame and secondly to get position of target in a local camera frame. These two data sets are synchronized automatically based on the number of the file as an index and on the GPS time stamp embedded into each frame.

The barcode reading algorithm (left branch on Fig.5) reads a sequence of files, renames them, and then calls the bar-code script for each frame. Results of running this code are saved as an ASCII Tab-delimited file.

Vision based TTS software (right branch on Fig.5) provided by PerceptiVU performs tracking of the payload. It takes enumerated sequence of images as an input. A user can select the payload in the first frame of the sequence, determine its size, choose suitable polarity, select tracking algorithm and tracking gain. Position of the payload in the camera coordinate frame (*x*- and *y*-offsets) for each frame is saved in an ASCII comma-separated file (to do this a user should enable "data logging checkbox" in Tools menu, see Fig.4.). Two additional parameters are currently also included in this file. First one is the Quality of Tracking (QOT). It is an indicator of the level of contrast in the tracking gate. Large QOT values indicate good contrast; low QOT values indicate little contrast. The second one is a Tracking State (TRCK). When TRCK is 1 the tracker is engaged and 0 – otherwise. The resulting ASCII file serves as an input to the data conditioning algorithm.

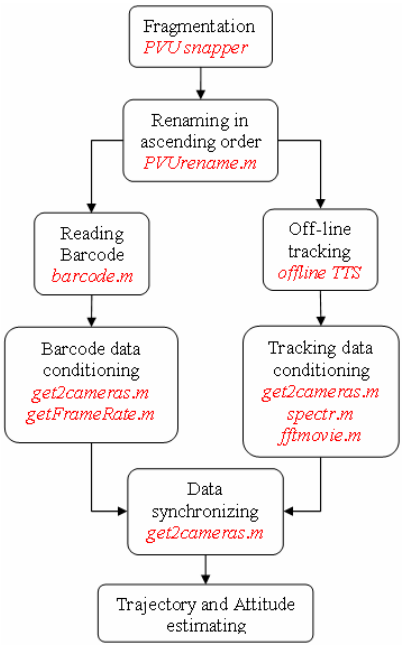


Figure 5. Data collection and Signal conditioning.

Video encoding and fragmentation may introduce a distortion of the image that results in corrupting the data for the bar-code reading procedure. This in turn may cause dropouts or incorrect reading. The vision-based TTS software may also produce invalid tracking results. In order to correct for these errors the resultant files from both reading procedures are subject to further processing (conditioning).

Conditioning of the bar-code data is based on the simple fact that time is monotonous and increasing, and LOS orientation angles are smooth for a descending system (see Fig.6). Thus the algorithm first tests the time stream for any dropouts or inconsistency and determines the average fragmentation rate. If any dropouts are found, the algorithm restores corrupted time stamp by linearly interpolating the time stamp between valid points based on the last average rate estimated. Though the time stream is corrected, these points (azimuth, elevation angles) are marked by introducing an additional Boolean value that takes 1 if the data is valid and 0 otherwise. Results of the GPS time reconditioning are presented on Fig.6.

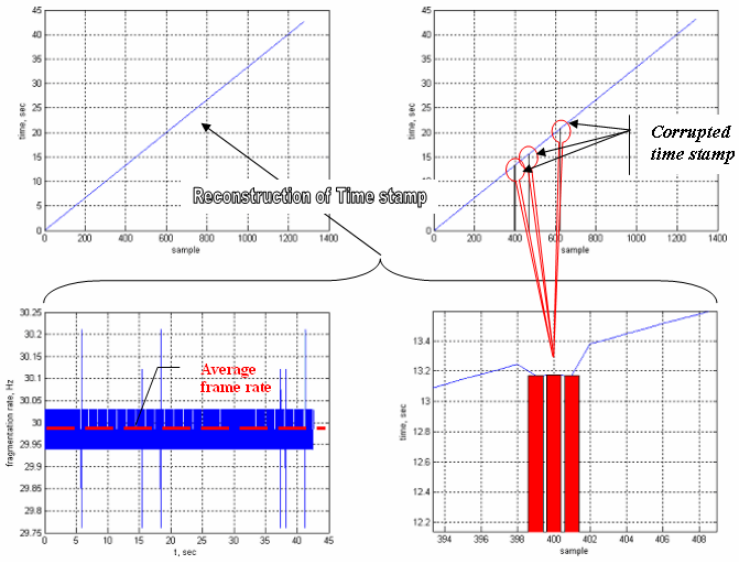


Figure 6. Reconstruction of time stamp.

Conditioning of the TTS results is based on the Fast Fourier Transform. It was observed that TTS software might lose the target while still producing very stable continuous tracking signals (*x*- and *y*- offsets). This can be caused by frequent changes of illumination, fast dynamics of the tracking object, poor visibility conditions, and poor quality of the video stream. Due to the high rate of fragmented video it is often not possible to visually recognize the frames where TTS fails. This is complicated by the fact that TTS might quickly reacquire the target again without notifying the user. Therefore additional care should be taken while preparing data for the estimating algorithm.

The TTS data conditioning consists of analyzing of short moving window of the tracking signal of interest (1-2 seconds that corresponds to 30-60 points) with small overlap of 10-15%. Tracking failure results in a significant falling in the power spectrum as shown in Fig.7. Previous technique of introducing a Boolean indicator on data validity is used in this case as well.

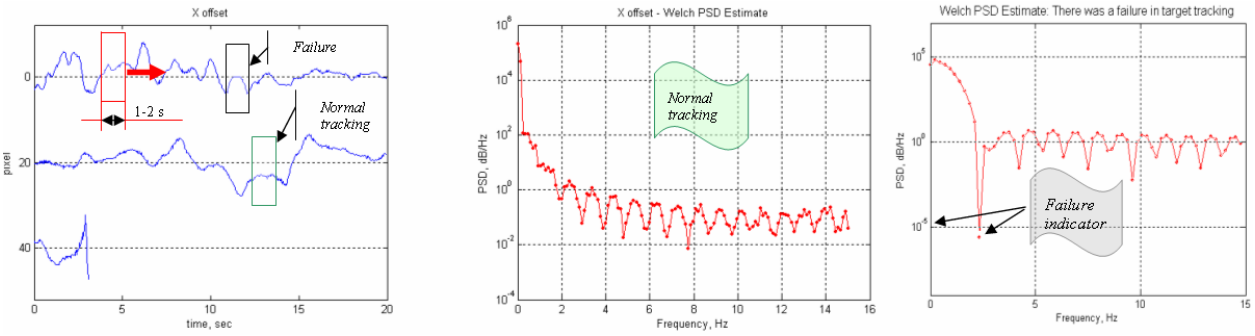


Figure 7. Validation of tracking results.

Finally, the data from both streams (both branches of Fig.5) are synchronized versus GPS time and passed on to the position estimation algorithm discussed next.

III. Payload position estimation

This section deals with developing and testing of algorithms that resolve the three-dimensional position of payload's centroid $\vec{P}(t)=\left[x_{pl}(t),y_{pl}(t),z_{pl}(t)\right]^T$, when its projection $\{u^i_{pl}(t),v^i_{pl}(t)\}$ onto the image plane of *i* cameras is available. It is assumed that LTP (coordinate frame $\{u\}$) positions $\vec{C}^i=\left[x^i_c,y^i_c,z^i_c\right]^T$) of each camera as well as their focal length, f^i, azimuth, $Az^i(t)$, and elevation, $El^i(t)$, are known.

In the absence of measurement errors the following equalities hold for each instance of time:

$$\frac{R\left[\vec{C}^i-\vec{P}\right]\begin{bmatrix}1\\0\\0\end{bmatrix}}{\left\|R\left[\vec{C}^i-\vec{P}\right]\right\|}=\cos\left(Az^i+\tan^{-1}\frac{u^i_{pl}}{f^i}\right),\quad \frac{\left[\vec{C}^i-\vec{P}\right]^T\begin{bmatrix}0\\0\\1\end{bmatrix}}{\left\|\vec{C}^i-\vec{P}\right\|}=\cos\left(El^i+\tan^{-1}\frac{v^i_{pl}}{f^i}\right), \tag{1}$$

where $R=diag([1,1,0])$.

However, since the measurement errors are always present these two equalities become inequalities and can be rewritten as

$$\frac{\left[\vec{C}^i-\vec{P}\right]^T\begin{bmatrix}0\\0\\1\end{bmatrix}}{\left\|\vec{C}^i-\vec{P}\right\|}-\cos\left(El^i+\tan^{-1}\frac{v^i_{pl}}{f^i}\right)=\Delta^i_{El},\quad \frac{\begin{bmatrix}1\\0\\0\end{bmatrix}R\left[\vec{C}^i-\vec{P}\right]}{\left\|R\left[\vec{C}^i-\vec{P}\right]\right\|}-\cos\left(Az^i+\tan^{-1}\frac{u^i_{pl}}{f^i}\right)=\Delta^i_{Az} \tag{2}$$

Therefore, each camera contributes two nonlinear equations of the form (2). It follows that to resolve the original problem for three components of vector $\vec{P}$ we need to have at least two cameras.

And the optimization problem for determining components of vector $\vec{P}$ maybe now formulated as follows. Having full information from $N \geq 2$ cameras find vector $\vec{P}(t) = \left[x_{pl}(t), y_{pl}(t), z_{pl}(t) \right]^T$ that minimizes the following compound functional:

$$J = \sum_{i=1}^{N} \left({\Delta^i_{Az}}^2 + {\Delta^i_{El}}^2 \right). \tag{3}$$

It's worth noting that in general the original formulas (3) can be written in many ways employing different geometric identities. It is quite possible that using alternative formulation may add robustness to the solution. But at this point improving algorithm robustness was not an issue since simulations with both emulated and real drop data worked fairly well as is shown next.

These simulations include two basic sets of experiments:

- Simulation 1.1 when the data from three cameras was emulated (created artificially); and
- Simulation 1.2 when real data from two cameras was used.

Figure 8 shows the first of two Simulink models developed to address this problem. This interactive model assumes three cameras and allows for varying their positions with respect to payload release point, as well as enables selection of one of the two ADS descent trajectories (first – a simple spiral and second one taken from the model of Ref.1 incorporating the real YPG wind profile.) Advantage of the Sim.1.1 using emulated cameras data (Fig.9) is that the developed perspective position estimation (PPE) algorithm can be thoroughly tested since the payload position is known.

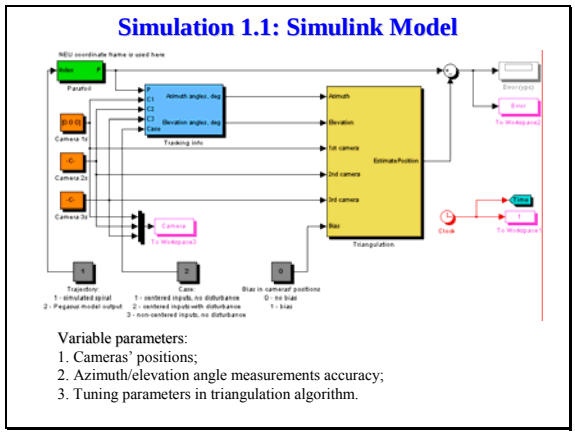


Figure 8. Interactive Simulink model for Simulation 1.1.

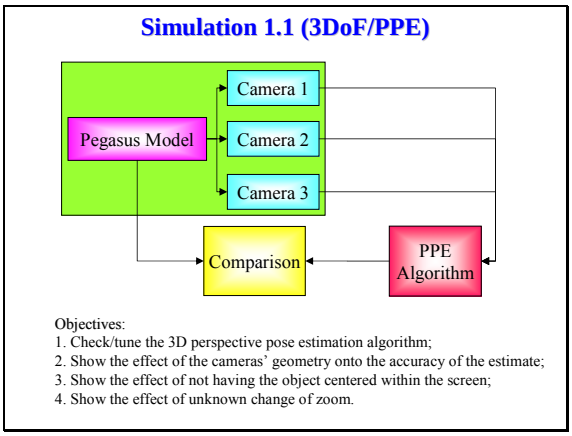


Figure 9. Essence of Simulation 1.1 for payload position estimation.

Figures 10-13 present some results of the first simulation. Fig.10 includes 3D projection of a simple no-wind spiral (top left picture), the horizontal projection of simulation setup with positions of three cameras around the spiral (top right), azimuth (bottom left) and elevation angles (bottom right) as seen from these three cameras. Based on the azimuth-elevation data provided by three cameras with no disturbance present and knowing the precise location of these cameras makes it possible to estimate position of the payload very accurately with an estimate error on the order of MATLAB rounding error (Fig.11).

For the sake of sensitivity analysis, Fig.12 shows how inaccuracy in knowing the cameras location maps into the payload estimation error. As seen from this figure, for the given geometry, which is fairly close to the right pyramid, an estimation error is approximately a half of the camera position error.

In case of disturbances being present (introduced artificially) the accuracy of PPE algorithm degrades. For example, as seen from Fig.13 adding one-degree Gaussian noise (uncertainty) to the azimuth-elevation data causes significant errors in estimates of the payload position. For the simple spiral geometry they are as much as around $\pm 20m$ (top plot). Having recursive PPE algorithm tuned for higher accuracy (smaller tolerance) brings those errors down to $\pm 7m$ (bottom plot).

So far it was assumed that the image of payload was always in the center of the frame for all three cameras. Another issue that was also addressed during this study is uncertainty in the cameras focus lengths f^i. In principle as was shown above even with two cameras we have one spare degree of freedom so that we could include the focus

of one camera into the list of unknown parameters and still solve the problem. With three cameras the focal length of all three cameras may be included into the list of variation parameters.

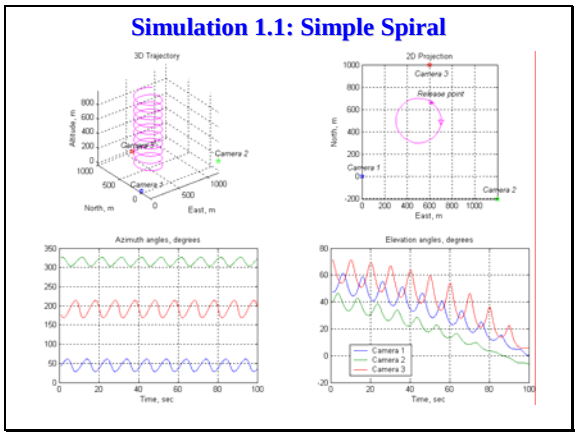


Figure 10. Simple spiral and cameras tracking angles.

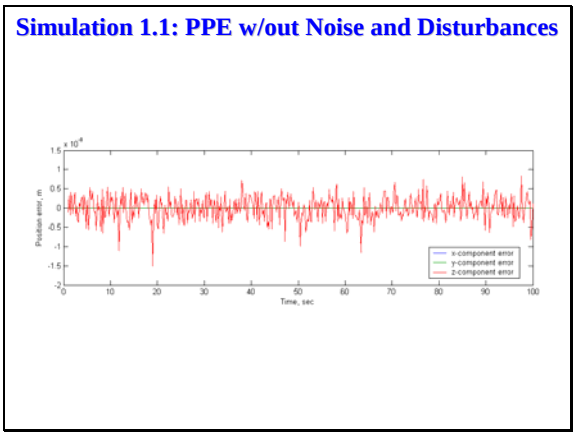


Figure 11. Accuracy of PPE algorithm with no noises and disturbances.

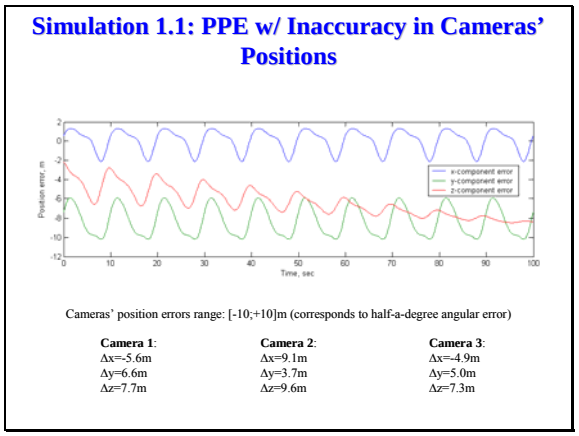


Figure 12. Payload position estimation errors caused by inaccuracy in cameras positions.

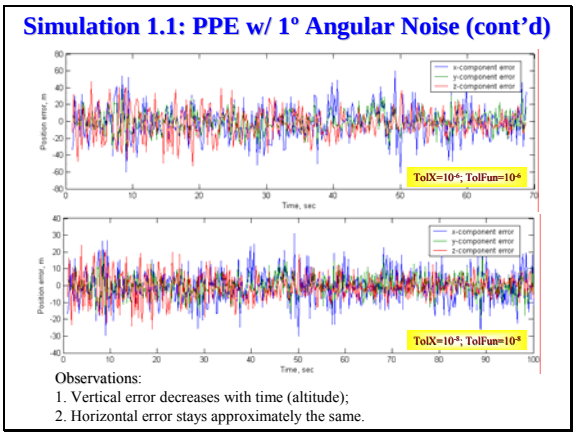


Figure 13. Tuning PPE algorithm in presence of one-degree noise.

Differentiating obvious equations for actual payload's azimuth and elevation angles corrected for its non-centered location within the frame as shown on Fig.14

$$Az^{*i} = Az^i + \tan^{-1}\frac{u^i_{pl}}{f^i},\ El^{*i} = El^i + \tan^{-1}\frac{v^i_{pl}}{f^i}, \tag{4}$$

results in basic relationship between focal length uncertainty δf^i and actual payload's azimuth and elevation for each camera

$$\delta Az^{*i} = \left(\frac{1}{f^i}\delta u^i_{pl} - \frac{u^i_{pl}}{f^{i2}}\delta f^i\right)\frac{1}{\cos^2\left(\frac{u^i_{pl}}{f^i}\right)} \approx \frac{1}{f^i}\left(\delta u^i_{pl} - \frac{u^i_{pl}}{f^i}\delta f^i\right),\ \delta El^{*i} \approx \frac{1}{f^i}\left(\delta v^i_{pl} - \frac{v^i_{pl}}{f^i}\delta f^i\right). \tag{5}$$

Therefore having uncertainty focal length translates into uncertainty in angular measurements made by cameras and in turn (as seen from Fig.14) into the payload's position estimate error. This is demonstrated in Fig.15 where focal length of one of the cameras was suddenly changed without introducing its new value into the PPE algorithm (after 40 sec it was reverted to the original value). Obviously this causes position estimate errors to rise.

Another set of simulations used real drop data (taken from the previous research on Pegasus) including real wind data (Fig.16). Again three cameras were placed around the drop zone and azimuth-elevation data from them was emulated. The developed PPE algorithm performed reliably and with the same performance as for the previous spiral-type trajectory.

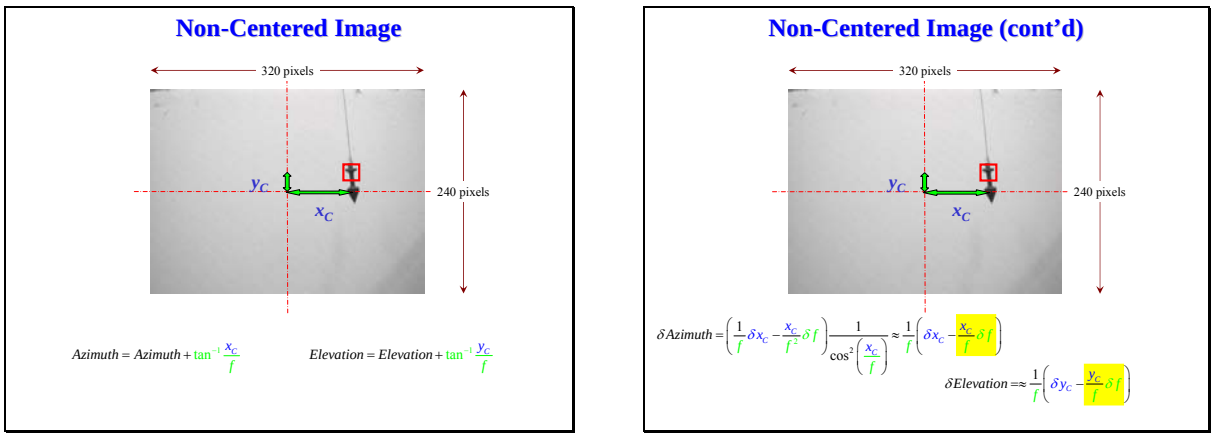


Figure 14. Explaining the influence of uncertainty in camera's focus length.

Obviously algorithm convergence as well as any open-end optimization algorithm depends on the proximity of the initial guess to the real solution. To demonstrate robustness of the developed PPE algorithm Fig.17 represents the situation where initial guess was miles away from the real payload position. As seen the algorithm performed well and converged to the actual payload's position in about 12 sec (top plot).

The bottom plot on Fig.17 represents an attempt to estimate the descent rate $\hat{V}_z$ in this last simulation. As can be seen the descent rate also converged to its actual value pretty fast. Why do we need an estimate of the descent rate? Well, knowing it could significantly improve robustness of the PPE algorithm in the presence of out-of-frame and occlusions events. This is the issue to be addressed in future research.

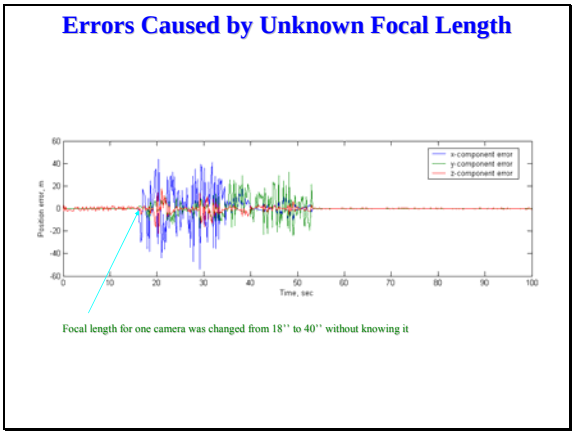


Figure 15. "Unrecorded" change of camera's focal length.

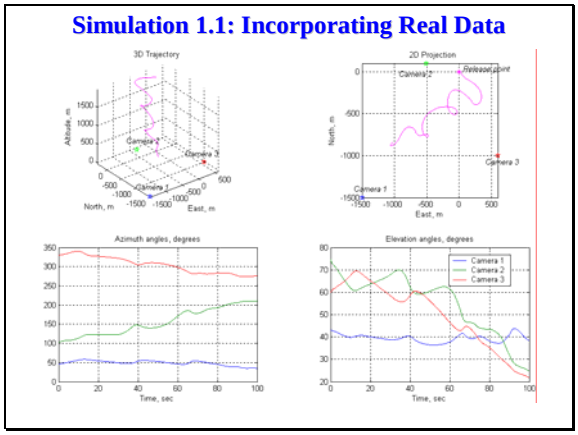


Figure 16. Real drop trajectory with three cameras emulated around the drop zone.

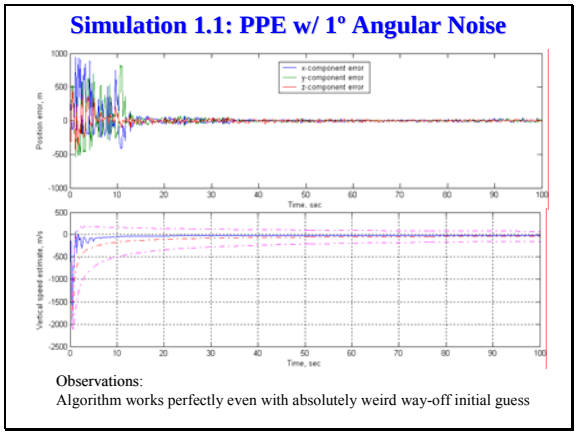


Figure 17. PPE algorithm robustness with way off initial guesses provided.

The second simulation (Sim.1.2) involved real data from two cameras. As shown on Fig.18 the video data from cameras 1 and 3 was provided. Unfortunately no other data on payload position was available making it impossible to estimate accuracy of the PPE algorithm (Fig.19).

This simulation included preprocessing the video data as explained in Section 1 in order to bring it down to the sequence of separate frames. From those the embedded azimuth-elevation data was extracted. Then it was augmented with the data from image processing providing frame coordinates $\{u_{pl}^{i}(t), v_{pl}^{i}(t)\}$.

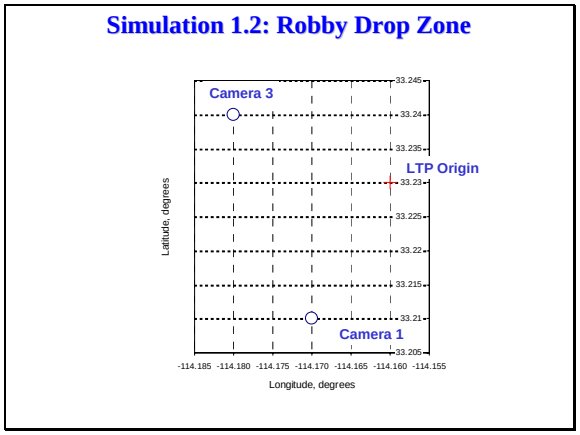


Figure 18. Real drop setup.

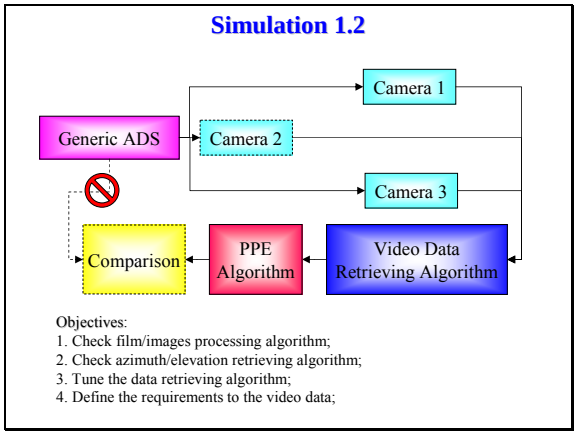


Figure 19. Essence of Simulation 1.2.

As shown in Fig.20 the Simulink model developed for the first simulation (Sim.1.1) was altered to accommodate real data stored in separate files rather than emulating data from three cameras. The combined video-data-retrieving algorithm and PPE algorithm performed fairly well and produced meaningful data as shown on Fig.21. By meaningful we mean that at least the payload release altitude, the only available piece of data beyond the data from two cameras, was perfectly matched by estimates of payload's position. Fig.21 presents azimuth-elevation data from two cameras (top left plots), payload's centroid location within the image frame for these two cameras (top right plots), and estimates of vector $\vec{P} = \left[x_{pl}(t), y_{pl}(t), z_{pl}(t)\right]^{T}$ components on the three bottom plots (3D projection, horizontal projection and time histories).

At this point it can be stated that all components of the combined video-data-retrieving algorithm and PPE algorithm have been developed and thoroughly tested. The only remaining test that would be useful before delivering the complete algorithm to the YPG, is having both video data from two or more cameras and GPS data from payload itself (as shown on Fig.22) to finally tune algorithms and to address the issues of required I/O interface and achievable accuracy one more time.

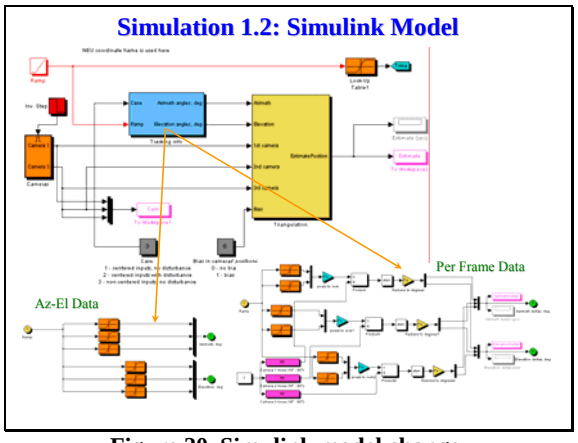


Figure 20. Simulink model change.

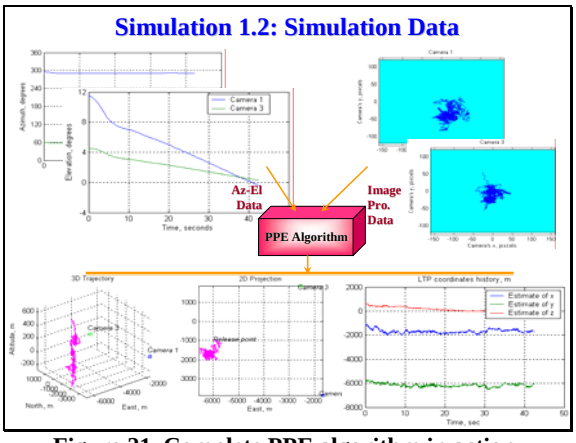


Figure 21. Complete PPE algorithm in action.

IV. Payload position and attitude estimation

This section addresses development and testing of perspective pose estimation algorithms that estimate payload's position (centroid) and orientation (three Euler angles). Solution to this problem requires data on multiple

known payload points (i.e. distances between points must know a priori). Since at this point no experimental data with several (rather than just one) clearly marked payload points is available only theoretical research and some simulations with emulated data were done.

The problem formulation is stated next. Suppose projections $\{u_j(t), v_j(t)\}$ onto the camera frame of several known payload points $\vec{P}_j(t) = [x_j(t), y_j(t), z_j(t)]^T$, j=1,…,M are given. Assume that these points can be seen by a single camera (lifting this assumption will be addressed later). Suppose that the LTP position $\vec{C} = [x_c, y_c, z_c]^T$ focal length f, azimuth $Az(t)$, and elevation $El(t)$ of the camera are also available. Given this data the problem is to determine perspective position and orientation estimate (PPOE) of the payload. Namely, to produce estimates of the position of the origin of the payload' body frame $\{b\}$ with respect to the local tangent plane $\{u\}$ and three Euler angles between these two coordinate frames (or just a rotation matrix from one coordinate frame to another, say ${}^b_u R$). The experimental setup for solving this problem is shown on Fig.23.

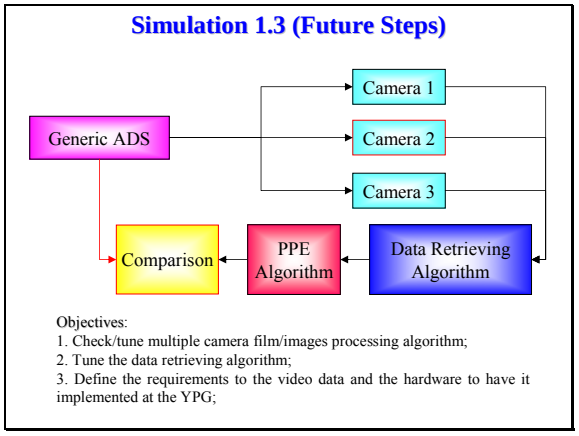


Figure 22. Proposed simulation to complete position estimation algorithm.

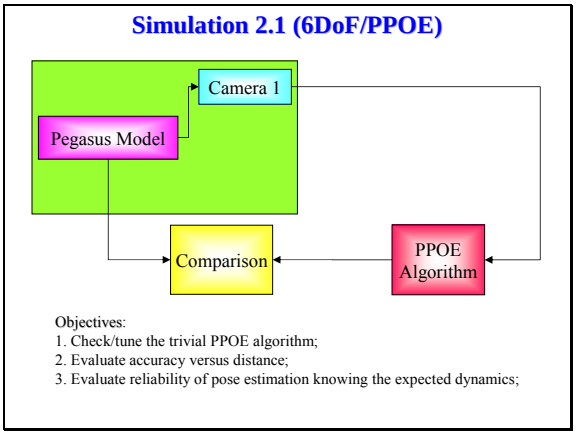


Figure 23. Experimental setup for the PPOE algorithm.

We first determine necessary conditions for the existence of a solution, namely the feasibility of the problem. Several questions need to be answered: How can one identify more than one point on the payload? What is the minimum number of points needed? What are the best points to use? Are these points visible at all times by a single camera or by several cameras?

Figure 24 shows samples of processed sequence of images from one of the available drops. It can be seen that yes, several points can indeed be distinguished. The best candidates are payload corners that should be marked. They are the easiest points to track by an image processing algorithm, they are characterized by the longest baseline which minimizes dilution of precision and they can be seen by several cameras simultaneously. The importance of the latter property is discussed later. Next question is which points to choose: the ones residing on one facet of payload (top left image on Fig.25), or on two adjacent facets (bottom left image). Or is it better to artificially introduce one point visible from every camera all the time (top right image)?

When placed on one facet every point will not be observed by a single camera at all times. Since one camera covers almost 180º of payload's yaw angle, having three cameras around drop zone will suffice to have every point on a single facet be seen by at least one of three cameras all the time. Now to provide uninterrupted data stream to the PPOE algorithm the data from three cameras should be blended together (that means that frame coordinates of all M points $\{u_j(t), v_j(t)\}$ will be available for every camera even if they are not seen physically).

The data on the points from two adjacent facets may also be used. However, in this case a single camera may not see all the points simultaneously (This is an important issue because each point must be uniquely identified by the image tracking algorithm. It might be the case that in order to restore the complete scene, all points need to be observed simultaneously). This decreases the "aperture" angle down to less than 90º and means that several cameras are needed to be used for the following data blending. Fig.26 shows a simple simulation with three points located on two adjacent facets of rotating payload observed by three cameras. The top left image presents a horizontal projection of the simulation setup with cameras' locations and trajectories of three points of interest, bar graphs to the right present number of points visible by each camera. Sometimes it is all three points on both facets, sometimes

it is only two points on a facet or even only point on another facet. And of course for the three-camera-setup there are times when none of the camera sees all three points. Surprisingly even if one more camera is added to this setup, the problem still persists (Fig.27).

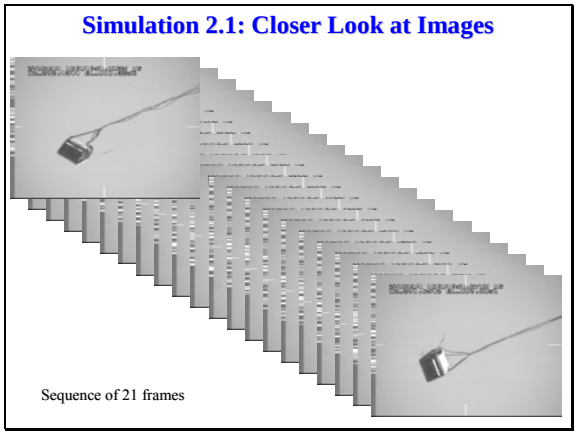


Figure 24. Samples of images available for PPOE problem solution.

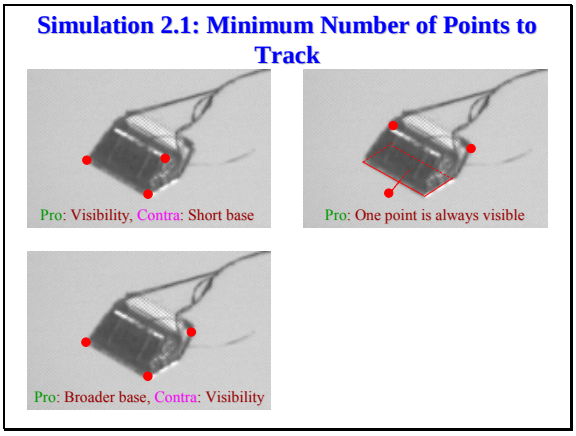


Figure 25. Possible triplets of traceable points.

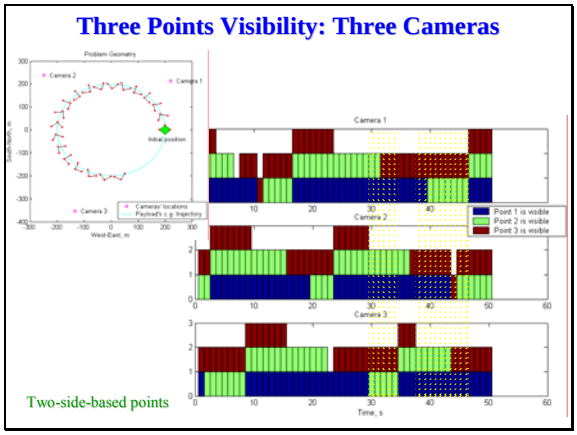


Figure 26. Can three cameras see all points on two adjacent facets simultaneously?

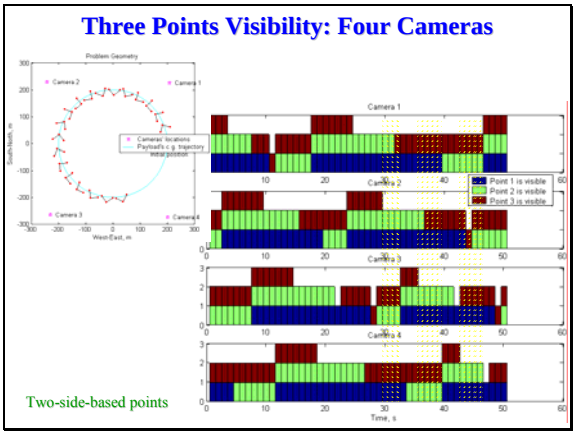


Figure 27. Does adding forth camera solve the problem?

So far we have answered all but one question. We know that we can distinguish more than one point. Most likely these points will be vertices of the payload container. Clearly, data from several cameras needs to be blended to provide smooth data stream for the PPOE algorithm. The last question: how many points are needed to reliably solve the PPOE problem has not been answered.

The transformation between two Cartesian coordinate systems ($\{b\}$ and $\{u\}$) can be decomposed into rotation and translation. In stereophotogrammetry, in addition, the scale may not be known. There are obviously three degrees of freedom to translation. Rotation has another three (direction of the axis about which the rotation takes place plus the angle of rotation about this axis). Scaling adds one more degree of freedom, totaling in seven degrees of freedom. Two points clearly do not provide enough constraints. However, three non-collinear points known in both coordinate systems provide nine constraints (three coordinates each), more than enough to permit determination of the seven unknowns. Therefore three points constitute the minimum number of points needed to be observed.

The PPOE problem has been addressed by the authors before (see Ref.2 and references therein). This problem, known in the literature as the perspective 3-point pose estimation, is presented in Fig.28. The figure illustrates geometry of the problem and shows three biquadratic equations that must be solved. It turns out that this problem has been studied for 150 years (Fig.29) and so far no analytical solution has been found. In fact, even the number of solutions to these three equations has not been determined. Some researchers suggest that as many as 15 possible solutions may exist (to be obtained numerically).

Eventually, the "mystery" of multiple solutions was resolved a set of only two, three or four admissible (feasible) solutions depending on the three-point geometry was established.[2] Figure 30 illustrates how the problem geometry impacts the number of solutions. For the case of a right tetrahedron the problem is reduced to finding intersections of three identical elliptical cylinders drawn around each of three axes (figure on the right). But in general (asymmetric) case the number of feasible solutions is two (two plots on the bottom). Figure 31 gives complete topography of admissible solutions. If three points (lying in the z=0 plane) are observed from the point belonging to a sort of inverted truncated tetrahedron referred further as an insoluble tetrahedron or IT the number of admissible solutions is four. Otherwise, the number of admissible solutions is two. (On the border of IT the number of admissible solutions is three.)

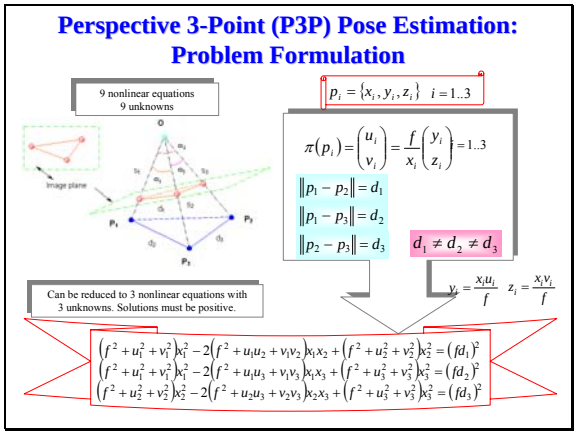


Figure 28. Formulation of P3P problem.

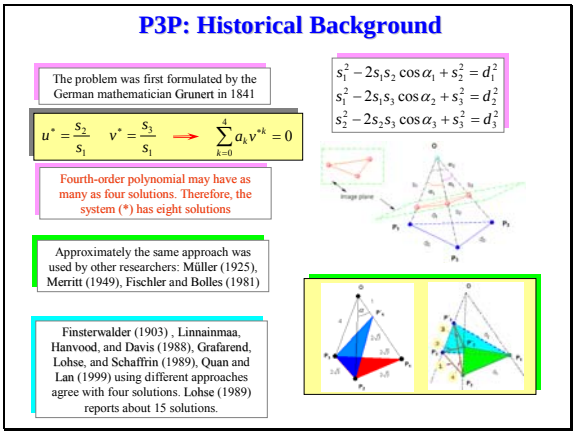


Figure 29. Historical background.

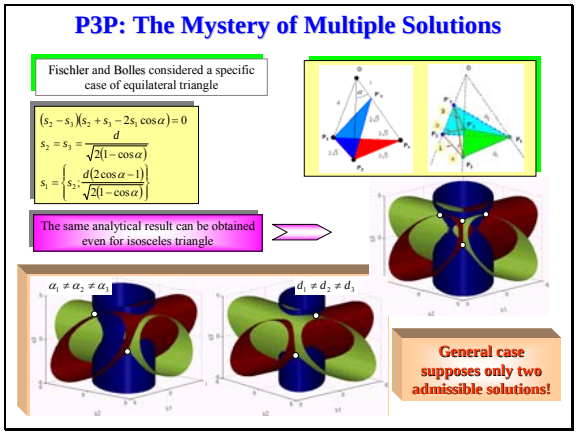


Figure 30. Geometrical representation of possible solutions.

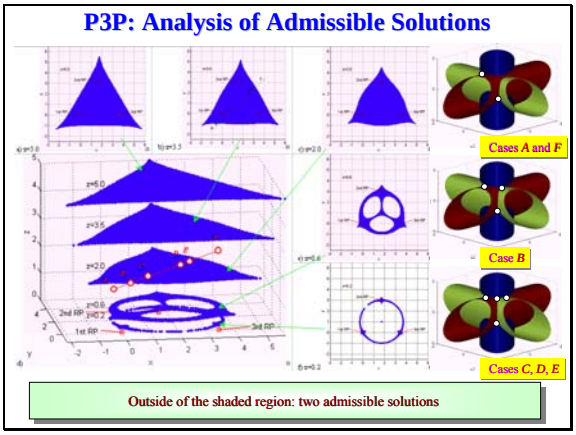


Figure 31. Solutions topography.

For the case of four solutions it is quite difficult to determine the correct one because they are very close to each other. However, in the case of two admissible solutions they can be easily distinguished from each other making it possible to select and track the correct one (the procedure suggested in Ref.2 suggests picking the solution with a correct (feasible) normal as shown on Fig.32).

The P3P algorithm and appropriate Simulink model (Fig.33) were developed and tested. Three points on a single facet of payload were emulated. As expected this simulation (Sim.2.1) performed fairly well producing quite accurate and fast solutions while payload was sufficiently high above the camera. However as payload descended the camera ended up inside the IT region characterized by four admissible solutions. This ambiguity can be resolved by tracking four or more known points.

In this case we can no longer expect to be able to find a transformation that maps the measured coordinates of points in one system exactly into the measured coordinates of these points in another. Rather, we will minimize the sum of squares of residual errors as was done for PPE algorithm.

However, for the case of four or more points finding the best set of transformation parameters is not easy. In practice, various empirical, graphical, and numerical procedures are used (Fig.34). These are iterative in nature. That is, given an approximate solution, such a method leads to a better, but still imperfect, answer. The iterative method is applied repeatedly until the remaining error is negligible.

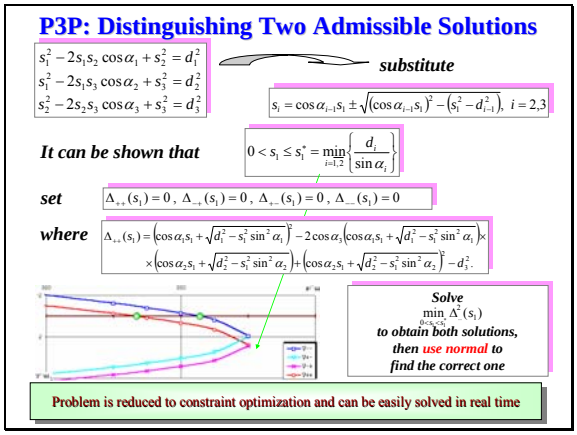


Figure 32. Distinguishing two admissible solutions.

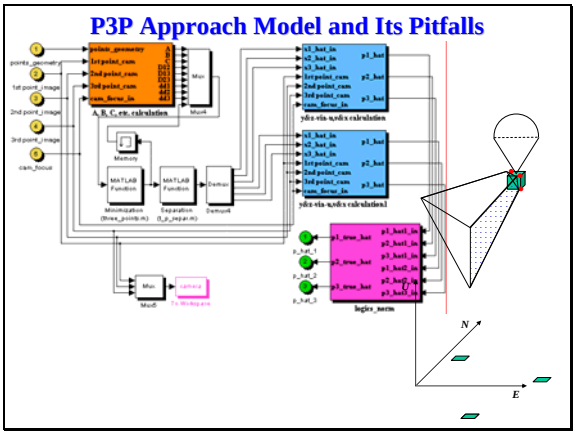
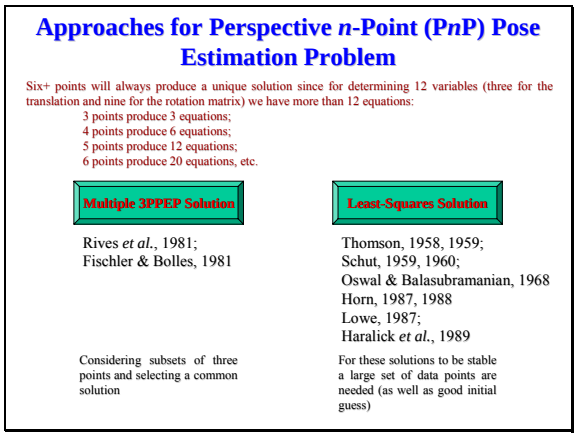


Figure 33. Pitfalls of P3P approach solution.



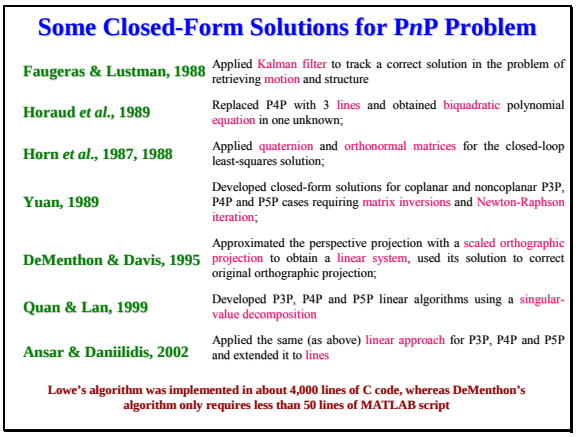


Figure 34. Approaches for P*n*P pose estimation.

Before the introduction of a more sophisticated method employing 4+ points, a review of some of the existing algorithms widely used in many contexts in computer vision is given.

The closed form solutions to the 3-point problem were discussed in several publications,[3,4] which offered solutions with well understood multiplicities.[5,6] Fischler and Bolles[7] extended their solution to 4 points by taking subsets of three points and using consistency checks to eliminate the multiplicity for most point configurations. Horaud *et al.*[8] developed a closed form solution on 4 points which avoids the reduction to a 3-point solution. These closed form methods can be applied to more points by taking subsets and finding common solutions to several polynomial systems, but the results are susceptible to noise and solutions ignore much of the redundancy in the data.

Among so-called linear algorithms three are worth mentioning here. Quan and Lan[9] derive a set of eighth degree polynomial constraints in even powers of the depth of each reference point by taking sets of three inherently quadratic constraints on three variables and eliminating two using Sylvester resultants. They apply this method to each point in turn. Similar algorithm suggested by Ansar and Daniilidis[10] is based on depth recovery. It avoids the degree increase, couples all *n* points in a single system of equations and solves for all *n* simultaneously. Recently, Fiore[11] has produced an algorithm for points which introduces two scale parameters in the world to camera transformation and solves for both to obtain the camera coordinates of points. Unlike Ansar and Daniilidis algorithm and that of Quan and Lan, Fiore's approach requires at least 6 points unless they are coplanar.

There also exist many iterative solutions based on minimizing the error in some nonlinear geometric constraints. Nonlinear optimization problems of this sort are normally solved with some variation on gradient descent or Gauss-Newton methods. Typical of these approaches is the work of Lowe[12] and of Haralick.[13] There are also approaches

which more carefully incorporate geometry of the problem into the update step. For example, Kumar and Hanson[14] have developed an algorithm based on constraints on image lines using an update step adapted from Horn's[15] solution of the relative orientation problem. There are several such variations using image line data. Liu *et al.*[16] use a combination of line and point data. Lu, Hager and Mjolsness[17] combine a constraint on the world points, effectively incorporating depth, with an optimal update step in the iteration. DeMenthon and Davis[18] initialize their iterative scheme by relaxing the camera model to be scaled orthographic.

Camera calibration is closely related to pose estimation, but is more general as the calibration simultaneously estimates both pose and the intrinsic parameters of the camera. Abdel-Aziz, and Karara,[19] Sutherland[20] and Ganapathy[21] proposed a direct linear method DLT for solving for the 11 entries of the camera projection matrix from at least six corresponding points. The method is further improved by Faugeras and Toscani[22] using a different constraint on the projection matrix. Lenz and Tsai[23] proposed both linear and nonlinear calibration methods. Although these methods might be applied to pose determination, full calibration is a heavy over-parameterization for the pose problem, giving reduced stability and requiring more points.

To conclude this review it should be noted that all iterative approaches typically exhibit slow convergence for poor initial guess, convergence to local minima and require a large number of points for stability. Algebraic approaches applied to subsets are sensitive to noise and to selecting the common root from noisy data. Therefore, the approach by DeMenthon and Davis[18] (http://www.cfar.umd.edu/~daniel/) was chosen after careful consideration of several other methods.

The method combines two algorithms.[18,24,25] The first algorithm, POS (Pose from Orthography and Scaling) approximates the perspective projection with a scaled orthographic projection and finds the rotation matrix and the translation vector of the object by solving a linear system. The second algorithm, POSIT (POS with ITerations), uses in its iteration loop the approximate pose found by POS in order to compute better scaled orthographic projections of the feature points, then applies POS to these projections instead of the original image projections. POSIT converges to accurate pose measurements in a few iterations and can be used with many feature points at once for added insensitivity to measurement errors and image noise. Compared to classic approaches making use of Newton's method, POSIT does not require starting from an initial guess, and computes the pose using an order of magnitude fewer floating point operations; it may therefore be a useful candidate for real-time operation.

The original algorithm only computes a rotation matrix or to be more precise three vectors constituting this matrix, not Euler angles themselves. Because of the numerical nature of the solution this matrix is not orthonormal. So, necessary adjustments were made to normalize the rotation matrix to extract Euler angles from it.

One of the method's advantages in this application is that it easily handles any number of points (starting from four points). Of course these points must not reside in one plane, i.e. they should be non-coplanar points. Therefore, the points from two+ facets of the payload should be tracked (Fig.35). If we assume that we can detect and match in the image four or more non-coplanar feature points of the object (data from several cameras should be blended together), and that we know their relative geometry on the object, the algorithm produces the solution of the PPOE problem.

Two sets of simulations using modified DeMenthon algorithm included:

- Simulation 2.2: the data for four+ points was emulated (created artificially); and
- Simulation 2.3: the data from the real drop (taken from a sequence of separate frames manually) was used.

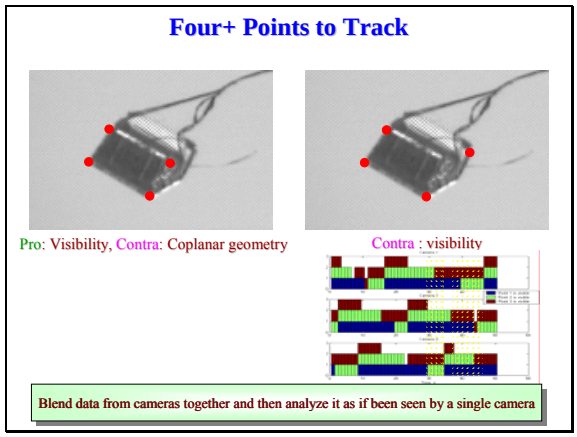


Figure 35. Coplanar versus non-coplanar points.

Both Matlab and Simulink models incorporating the developed PPOE algorithm were developed. The Simulink model is shown on Fig.36. Among many interactive features this model allows user to choose the number of points to track (although not all of them can be observed by the single camera as discussed above, the assumption is that the data from several cameras are combined together). So one can choose from four non-coplanar (on two adjacent facets) to eight points (in the latter case they represent all corners of payload parallelepiped container). Different trajectories can be considered and measurement noise can be added in the same manner as in the model of Section 3.

When this algorithm is deployed at YPG it will provide a very important feature of automatic tracking of the payload. Knowing (predicting) its position at the very next moment allows pointing all cameras directly to it. In such

a case the payload will always appear almost in the center of the frame. Moreover, since all algorithms are interactive, having traceable points in the center of the image drastically improves the accuracy of pose estimation. That's why this feedback feature has already been used in the algorithm as shown in Fig.26.

The setup for Sim.2.2 is presented in Fig.37. Initially the virtual camera is pointed at the release point virtual and automatically tracks the payload while it descends. Again, the advantage of this simulation with emulated data is that the developed PPOE algorithm can be thoroughly tested since the real position of all points is known a priori and can be compared with their estimates.

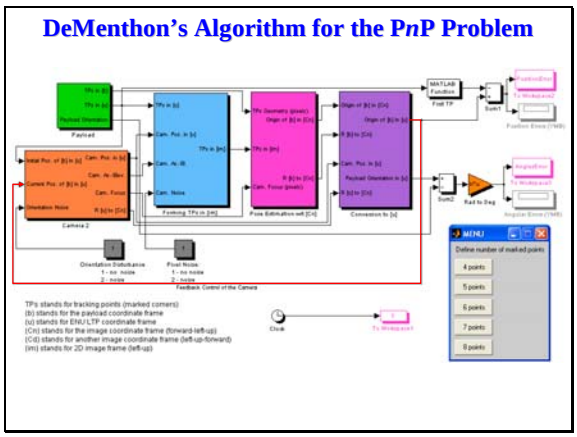


Figure 36. Simulink model employing PPOE algorithm.

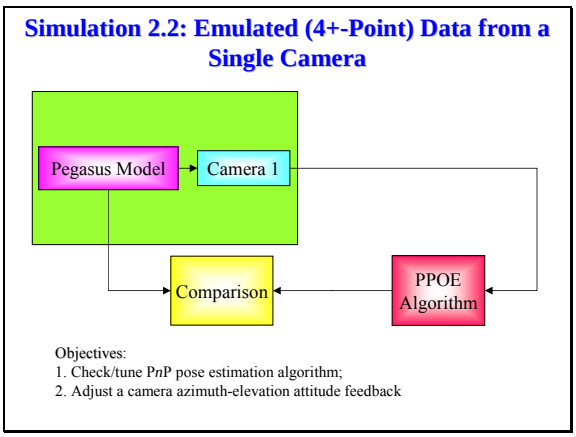


Figure 37. Simulation 2.2 setup.

The developed PPOE algorithm performed extremely well, above expectations. Results of the simulated drops are presented in Fig.38. The release point was at 1,500*m* above the camera level. Camera was about 600*m* away from the drop zone. Dimensions of the payload were chosen to be 2*m*x3*m*x2*m*. The payload rotated along all three axes with amplitude of 20º and 10º in pitch and bank, respectively. Four to eight points were used to test and tune the PPOE algorithm.

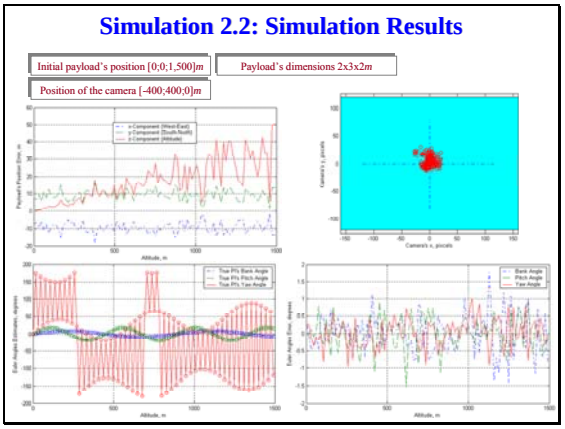


Figure 38. Simulation 2.2 results.

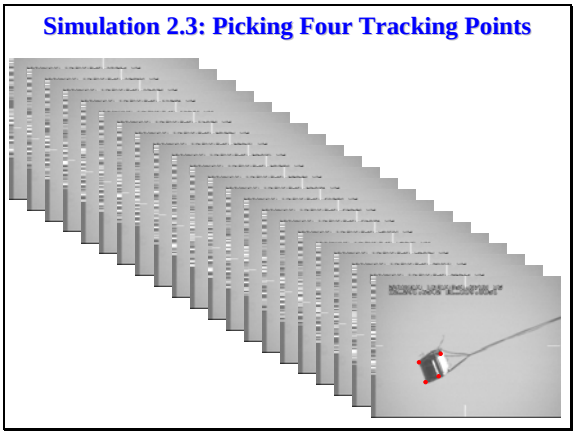


Figure 39. Digitized video sequence.

The bottom left plot represents the true Euler angles versus altitude. On the right the errors of Euler angle estimates are shown. Surprisingly they are very small – around ±0.5º. The errors in the payload position estimate versus altitude are shown on the top left plot. It can be seen that while horizontal error maintain the same value of about ±10*m* the vertical error decreases from about 30*m* at the release point to almost 0*m* at the touch down point. It is worth emphasizing that such accurate results were obtained specifically due to feedback of payload position estimate. This allowed for maintaining the image of the payload (traceable points) precisely in the center of each frame as shown in the top right plot of Fig.38.

From the standpoint of a number of points needed to be tracked we found that having just four non-coplanar points was sufficient to provide accurate estimates of translational and rotation parameters of the payload.

Inspired by the results of Sim.2.2 the next step incorporated real airdrop data. One of the available videos was digitized to produce a sequence of separate frames as shown on Fig.39. Four visible non-coplanar points residing on two adjacent facets were chosen. The frame coordinates of for each of these four points $\{u_j(t), v_j(t)\}$, j=1,…,4 were extracted manually and stored in separate ASCII files. Azimuth-elevation data for each frame as well as camera location were also available. The experimental setup for this simulation (Sim.2.3) is shown on Fig.40.

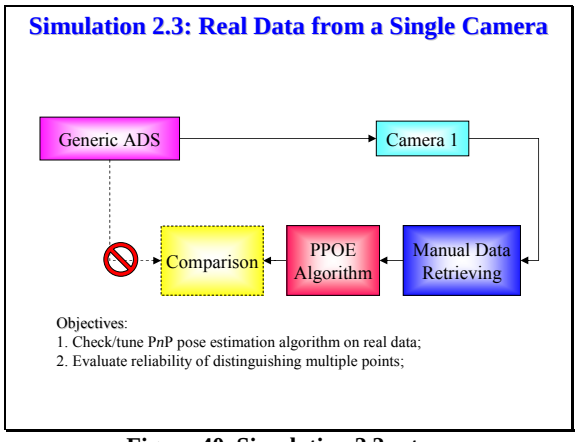


Figure 40. Simulation 2.2 setup.

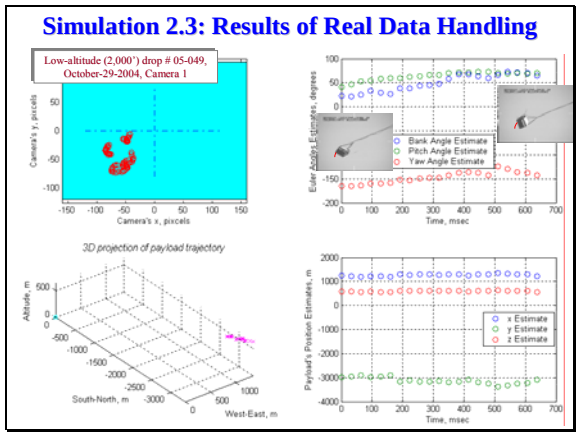


Figure 41. Results of Simulation 2.3.

The results of simulation are shown in Fig.41 All the data points used are presented in top left plot. Clearly, automatic tracking of the payload (Fig.38) outperforms the manual tracking. The payload for the real drop video data is not always in the center of the image. The restored 3D trajectory and coordinate time-histories of the payload are shown on two plots at the bottom of Fig.41. In the absence of any GPS/IMU data they cannot be validated but at least the simulation results match the release altitude that is known. As for the Euler angles, their time-histories are shown on top right plot. At least visually they match the data from the images fairly well.

Not much can be done at this point in the absence of experimental data. However, to complete this study two more should be developed. First, the blending algorithm that combines the data from several cameras has to be developed as shown in Fig.42. Then multi-point image processing algorithm should be developed. This will require distinct markings on the payload. Complete setup for the proposed final simulation is shown on Fig.43.

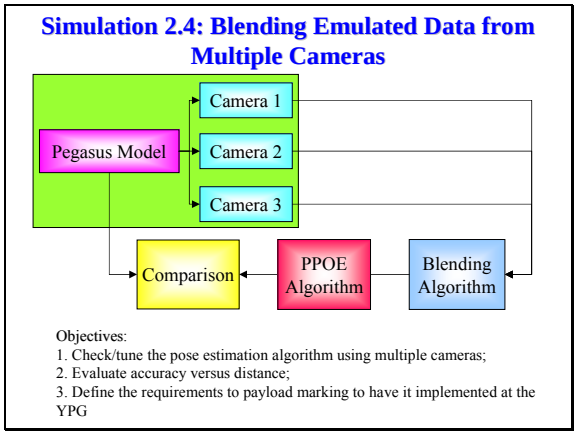


Figure 42. Accommodating data from several cameras.

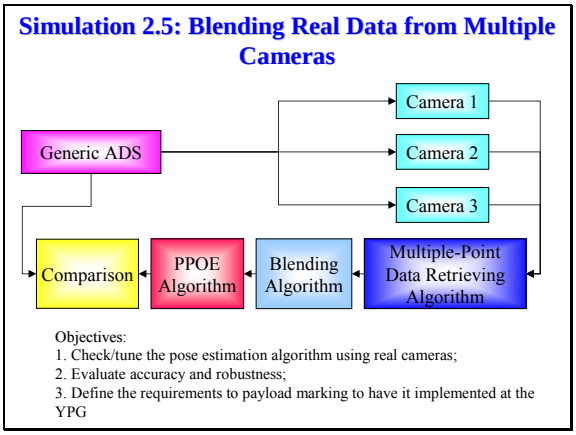


Figure 43. Complete setup for the autonomous PPOE simulation.

V. Conclusion

In the course of this study several Matlab/Simulink compatible tools were developed. They include software for video data processing, perspective position estimation and perspective position plus orientation estimation. These algorithms proved their accuracy and robustness in several numerical simulations based on (incomplete) sets of

experimental data. To proceed with the development of the complete self-contained system more experiments are needed to be carried out. The complete sets of experimental data including that of tracking of several properly marked points by multiple cameras should be incorporated into analysis. The problem of maintaining sufficient data about multiple predefined tracking points from all available cameras all the time with the goal of uninterrupted pose estimation also needs to be addressed. Finally, the IMU/GPS data for the payload is also needed to compare the results of estimation with the true data.

References

[1] Yakimenko, O.A., "On the Development of a Scalable 8-DoF Model for a Generic Parafoil-Payload Delivery System," *Proc. of the 18th AIAA Aerodynamic Decelerator Systems Technology Conference*, Munich, Germany, May 24-26, 2005.

[2] Yakimenko, O.A., Kaminer, I.I., Lentz, W.J., Ghyzel, P.A., "Unmanned Aircraft Navigation for Shipboard Landing using Infrared Vision," *IEEE Trans. on Aerospace and Electronic Systems*, 38(**4**), 2002, pp.1181-1200.

[3] Dhome, M., Richetin, M., Lapreste, J.T., Rives, G., "Determination of the Attitude of 3D Objects from a Single Perspective View," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(**12**), 1989, pp.1265–1278.

[4] Haralick, R.M., Lee, C., Ottenberg, K., Nolle, M., "Analysis and Solutions of the Three Point Perspective Pose Estimation Problem," *IEEE Conf. on Computer Vision and Pattern Recognition*, Maui, HI, 1991, pp.592-598.

[5] Huang, T.S., Netravali, A.N., Chen, H.H., "Motion and Pose Determination using Algebraic Methods," in V.Cappellini (Ed.) Time-Varying Image Processing and Moving Object Recognition, Vol.2, Elsevier Science Publication, 1990, pp243–249.

[6] Navab, N., Faugeras, O.D., "Monocular Pose Determination from Lines: Critical Sets and Maximum Number of Solutions," *Proc. of the IEEE Conf. Computer Vision and Pattern Recognition*, New York, NY, June 15-17, 1993, pp.254–260.

[7] Fischler, M.A., Bolles, R.C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. Association for Computing Machinery*, 24(**6**), 1981, pp.381-395.

[8] Horaud, R., Conio, B., Leboulleux, O., Lacolle, B., "An Analytic Solution for the Perspective 4-Point Problem," *Computer Vision, Graphics, and Image Processing*, 47(**1**), 1989, pp.33–44.

[9] Quan, L., Lan, Z.D., "Linear n-Point Camera Pose Determination," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(**8**), 1999, pp.774–780.

[10] Ansar, A., Daniilidis, K., "Linear Pose Estimation from Points and Lines," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(**5**), 2003, pp.578–589.

[11] Fiore, P.D., "Efficient Linear Solution of Exterior Orientation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(**2**), 2001, pp.140-148.

[12] Lowe, D.G., "Fitting Parameterized Three-Dimensional Models to Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(**5**), 1991, pp.441–450.

[13] Haralick, R.M., Joo, H., Lee, C.-N., Zhuang, X., Vaidya, V.G., Kim, M.B., "Pose Estimation from Corresponding Point Data," *IEEE Trans. on Systems, Man, and Cybernetics*, 19(**6**), 1989, pp.1426–1446.

[14] Kumar, R., Hanson, A.R., "Robust Methods for Estimating Pose and a Sensitivity Analysis," *Computer Vision and Image Understanding*, 60, 1994, pp.313–342.

[15] Horn, B.K.P., "Relative Orientation," *International Journal of Computer Vision*, 4(**1**), 1990, pp.59–78.

[16] Liu, Y., Huang, T.S., Faugeras, O.D., "Determination of Camera Location from 2D to 3D Line and Point Correspondences," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(**1**), 1990, pp.28–37.

[17] Lu, C.-P., Hager, G.D., Mjolsness, E., "Fast and Globally Convergent Pose Estimation from Video Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(**6**), 2000, pp.610-622.

[18] DeMenthon, D., Davis, L.S., "Model-Based Object Pose in 25 Lines of Code," *International Journal of Computer Vision*, 15(**1-2**), 1995, pp.123-141.

[19] Abdel-Aziz, Y.I., Karara, H.M., "Direct Linear Transformation into Object Space Coordinates in Close-Range Photogrammetry," *Proc. of the ASP/UI Symposium on Close-Range Photogrammetry*, Urbana, IL, 1971, pp.1–18.

[20] Sutherland, I.E., "Three-Dimensional Input by Tablet," *Proc. of the IEEE*, 62(**4**), 1974, pp.453-461.

[21] Ganapathy, S., "Decomposition of Transformation Matrices for Robot Vision," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Atlanta, GA, 1984, pp.130–139.

[22] Faugeras, O.D., Toscani, G., "Camera Calibration for 3D Computer Vision," *Proc. of the Int'l Workshop Machine Vision and Machine Intelligence*, Tokyo, Japan, 1987.

[23] Lenz, R., Tsai, R.Y., "Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3D Machine Vision Metrology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(**5**), 1988, pp.713–720.

[24] DeMenthon, D., Davis, L.S., "Recognition and Tracking of 3D Objects by 1D Search," *DARPA Image Understanding Workshop*, Washington, DC, 1993, pp.653-659.

[25] David, P., DeMenthon, D., Duraiswami, R., Samet, H., "SoftPOSIT: Simultaneous Pose and Correspondence Determination," *International Journal of Computer Vision*, 59(**3**) 2004, pp.259-284.