Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

10th International Conference
on Electrical Engineering
ICEENG 2016

Target Localization from a UAV Using 3D Terrain Engine

By

Ali Abbas * Assef Jafar ** Zouhair Dahrouj ***

Abstract:

We present a new approach for localizing a ground target from a UAV using 3D terrain engine. We use inaccurate auxiliary sensors on the UAV to obtain an approximate measurement of the camera pose, we use this pose to move the virtual camera inside the engine, and then automatically we find multiple matches between the two images to find the 3D coordinates of the matches using 3D terrain engine. Finally, we test the co-planarity of the 3D points under the camera, depending on this test, we use coplanar or non-coplanar algorithm to estimate accurate global camera pose. The accurate pose is used for localizing targets seen in the image by transmitting a virtual ray from a pixel according to the camera pose until intersect the 3D terrain model in the requested point. We tested the proposed approach on a synthetic and real data. Experimental results proved the feasibility and robustness of the proposed approach and the precision is the same order as the 3D terrain engine.

Keywords:

Target localization, 3D terrain engine and Feature matching

---

* *Ali Abbas*: *ali.abbass@hiast.edu.sy* Corresponding Author is a PhD candidate at the Department of Electromechanic Systems, HIAST

** *Assef Jafar*: *assef.jafar@hiast.edu.sy* with the Department of Electromechanic Systems, HIAST

*** *Zouhair Dahrouj*: *zouhair.dahrouj@hiast.edu.sy* with the Department of Informatics, HIAST Higher Institute for Applied Science and Technology, Damascus, P.O.box: 31983, Syria, http://www.hiast.edu.sy/en

1. Problem formulation:

Given a real image from a UAV (Unmanned Aerial Vehicle) with a ground target seen inside it, our goal is to localize this target, which mean determine its geographic location in complex environment (not necessary a flat terrain). To solve this problem, we align the real image with a virtual image taken from a 3D terrain engine as shown in Figure (1), then we transmit a virtual ray from the target pixel according to the camera pose until intersect the 3D terrain model in the requested point. However, the big problem now is how to align these two images automatically in real time (this problem known as "image registration"). To solve the alignment problem, we assume that the camera's FOV (Field of View) is known, as well as an approximate pose taken from a set of inaccurate sensors mounted on the UAV, GPS for location (Global Positioning System) and IMU for angles (Inertial Measurement Unit). Given these hypotheses, we are looking for the accurate location (longitude, latitude, altitude) and the accurate rotations (azimuth, elevation, roll) that maps the camera frame to the frame of the 3D terrain engine, then we use this pose to move the virtual camera inside the 3D terrain engine to obtain a precise alignment between the real and virtual images.

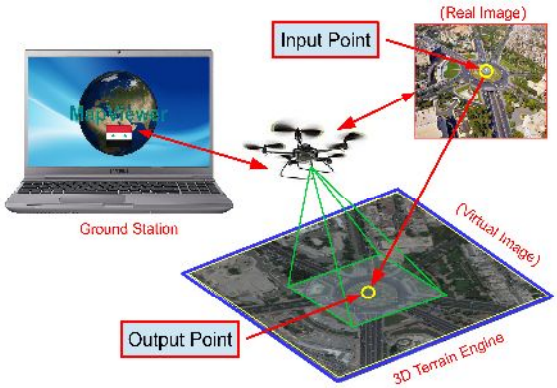


***Figure (1):** Problem definition (Umayyad Square in Damascus)*

2. Related Works:

2.1. Global pose estimation:

In general, to solve the problem of global pose estimation for a camera with 6-DOF (Degrees of Freedom), we need a 3D model (consists minimum from 3 points) which we can automatically find their 2D projections on the image plane. In the case where the camera is mounted on a UAV, two dominant approaches trying to solve this problem. The first one is based on the assumption that the earth model is plane textured with a satellite Geo-referenced images [1] (3D model here is a 3D plane), this approach turns

to image-image registration problem, which is basically use feature-based methods such as SIFT (Scale Invariant Feature Transform [2]) or SURF (Speeded Up Robust Features [3]) techniques. The second one which is proposed by Hyon Lim et al for real time video-based localization in scenes reconstructed offline using SFM (Structure from Motion) (3D model here is a point cloud). This algorithm efficiently combines key-point tracking in video with direct 2D feature to 3D point matching, without requiring scale-invariant image features. This approach can fail when the camera faces a part of the scene poorly represented in the map [4]. To avoid the limitations of the previous approaches we proposed a new approach that depends on a 3D terrain engine. 3D terrain engine allows us to deal with a complex model of the earth and process multiple views of the same target. Although there is very little published literature on (image/3D model) registration, a number of researchers have worked on related problems such as 3D modeling and multi-sensor registration [5] [6]. After finding the correspondences between the 3D-2D points, the problem turns to estimate pose from those correspondences. We divide the traditional solutions for this problem into two groups: closed-form solutions and iterative solutions. In [7] there is a thorough comparison between different closed-form solutions. In this work, we used iterative algorithm that use multiple points to handle errors of the camera's measurements [8].

2.2. Target localization:

Dobrokhodov et al. developed a system for target tracking based on vision algorithms. This system simultaneously estimate GPS coordinates, speed and heading of the target [9], in this method we need to build a robust tracker and we can localize one target only.
Barber et al. proposed a high accuracy geolocation algorithm to direct the payload towards the target accurately. These methods include flight path optimization, wind estimation, bias estimation, and recursive least squares filtering and he achieved localization accuracies of 2 to 5 meters regardless of wind conditions [10], but this method needs many measurements for the same target to use recursive least squares filtering and obtain the required results.
Conte et al. presents a method to localize a ground target accurately using a Micro Aerial Vehicle (MAV) equipped with a video camera sensor. This method depend on a satellite or aerial image registration technique, which calculate the target geolocation by registering the ground target image taken from an on-board video camera with a Geo-referenced satellite image. This method does not require accurate knowledge of the aircraft position and attitude. Therefore, it is especially suitable for MAV platforms that do not have the capability to carry accurate sensors due to their limited payload weight and power resources [11]. but this method depends on the assumption of flat terrain and need a robust tracker for one target only.
Finally, our needs are developing a robust method that did not need a tracker algorithm

and we can apply it to localize multiple targets simultaneously without the assumption of flat terrain.

3. Introduction:

During the past decades, many countries used the UAV in monitoring and reconnaissance operations such as fire detection, monitoring of oil pipelines and border areas. However, these tasks only done by rich countries because of the high costs for those UAVs. Recent developments in material science, control engineering and communications led to the development of a low cost UAV capable of carrying a digital camera and fitted with a communication system and a set of sensors that help determine the location and orientation of the UAV (such as GPS and IMU). However, these sensors suffer when they are cheap from the problem of low accuracy and high sensitivity to noise. The UAV itself is unstable and under the influence of wind. All these reasons make the task of estimating the location and angles of the UAV accurately using only these sensors of impossible issues. For this reason, we used digital camera mounted on the UAV as an optical sensor to improve the estimation accuracy.
We organize the rest of this paper as follows. First, we discuss the geometry of the problem and then outline the various steps in the proposed approach. Finally, we present experimental results on synthetic and real data followed by conclusions at the end.

4. Problem geometry:

We discuss here the real camera model, virtual camera model and camera calibration process.

4.1. Camera model (real image):

We work in projective 2- and 3- space, representing points in homogeneous coordinates. A 3D point v is represented as $(X, Y, Z, 1)^T$ and a 2D point v' is represented as $(x, y, 1)^T$. With the geometry shown in Figure (2), a 3D point v in world coordinate system can be represented in the camera coordinate system as v', given by

$$v' = Rv + T \tag{1}$$

Where $T = (T_x, T_y, T_z)^T$ is the translation vector and R denotes the 3 × 3 rotation matrix. The camera mapping from 3D to 2D is given by perspective projection equation

$$v' = P_m v \tag{2}$$

Where P_m is the 3×4 projection matrix. Given P_m and the depth Z at each pixel x in the image, the corresponding 3D point v can be obtained using equation (2). The projection matrix P_m can be decomposed as

$$P_m = K[R \mid T] \tag{3}$$

Where K is a 3×3 upper triangular matrix specifying the internal camera calibration parameters.

$$K = \begin{bmatrix} f_x & \alpha & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Where f_x and f_y are the focal lengths in the x and y directions, α is the skew parameter, and (p_x, p_y) is the principal point location. Since the camera is known a priori, it may be calibrated offline to find f and the other components of K.

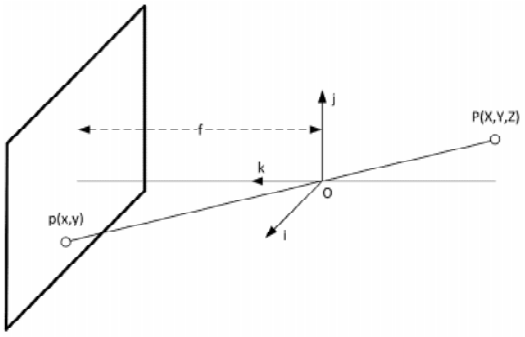


Figure (2): *Camera Model (Real Image)*

4.2. OpenGL camera model (virtual image):

Virtual camera in OpenGL is defined via view parameters eye $e(e_x, e_y, e_z)$, view $v(v_x, v_y, v_z)$, right $r(r_x, r_y, r_z)$ and up $u(u_x, u_y, u_z)$ measured in world space as shown in Figure (3). The view transform consists of two operations: a translation T, followed by a rotation R. We can combine the two operations into one single View Matrix M:

$$M = RT = \begin{bmatrix} r_x & r_y & r_z & A \\ u_x & u_y & u_z & B \\ -v_x & -v_y & -v_z & C \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Where: A=-(e.r), B=-(e.u) and C=-(e.v). The OpenGL camera has a limited FOV, which exhibits a view frustum, and specified by four parameters: f, a, zN and zF. Where,
zN, zF: distances to near, far planes.
f: focal length.
a: viewport height / width. The projection matrix is given by:

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & \frac{f}{a} & 0 & 0 \\ 0 & 0 & \frac{zF + zN}{zN - zF} & -1 \\ 0 & 0 & \frac{2\,zF\,zN}{zN - zF} & 0 \end{bmatrix} \quad (6)$$

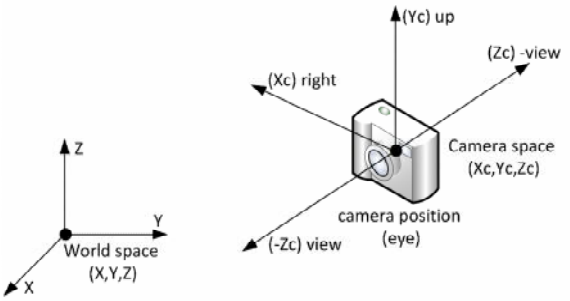


***Figure (3):** OpenGL camera model (Virtual Image)*

4.3. Camera calibration:

A prerequisite of our system is that the intrinsic camera parameters are known and constant. We use the calibration method described in [12] to simultaneously estimate the focal length and principal point parameters, standard values are assumed for the remaining intrinsic (i.e. zero skew, zero radial distortion and unit aspect ratio), or we can use MATLAB implementation [13].

5. Proposed approach:

We will explain some processes before describing the approach steps, such as intensity normalization, feature extraction and pairwise feature matching.

5.1. Intensity normalization:

We define the pixel intensity as an integer number between 0 and 255. Suppose that the

2D image is V and its mean is μ, then we obtain a normalized version U such that:

$$U = \max(th_1, \min(th_2, V - \mu + 128)) \quad (7)$$

Where th_1 and th_2 are the minimum and maximum thresholds respectively, such that $U \in [th_1, th_2]$.
U is invariant to uniform intensity changes that affect all the pixels of V at the same ratio (uniform lightning) as shown in Figure (4).

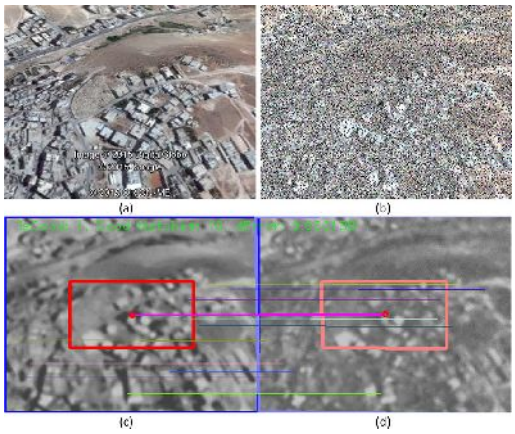


Figure (4): *Intensity Normalization, (a) Input image. (b) Noisy image: Input image + salt/pepper noise + additive Gaussian noise with mean=128, stddev=30 and darkened with gamma correction by γ=4. (c,d) Matching between original gray image and filtered image.*

5.2. Feature Extraction:

Our system utilizes the very effective SIFT key-point detector and descriptor [2] implemented on a GPU (Graphics Processor Unit) to represent point features in real time [14]. SIFT features are invariant to scale and rotation and partially invariant to viewpoint and illumination changes [15]. Hence, these kind of features are very suitable for wide baseline matching and found to be highly distinctive and repeatable in performance evaluation [16].

5.3. Pairwise Feature Matching:

A variety of approaches have been proposed to speedup nearest neighbor matching in high-dimensional spaces (like the 128-dimensional SIFT descriptor space) [17] [18]. These algorithms in general designed to run on a single CPU and known to provide speedups of about one or two orders of magnitude over linear search, but the speedup

comes with the cost of a potential loss in accuracy [19]. On the other hand, given that the number of features is limited to some hundreds (in our work we just use 200 key-points), nearest neighbor search implemented on a GPU can achieve an equivalent speedup, but delivers the exact solution. Hence, we employ a GPU accelerated feature matching approach.

5.4. Approach steps:

We resume the proposed approach by the following steps as shown in Figure (5), taking in consideration the co-planarity of the 3D points (Because when dealing with planes there rises doubt about the pose because several poses that are very different have the same perspective projection [20]):

1. Calibrate the camera to find its internal parameters, then calculate its horizontal field of view FOV_h from its focal length f and the image width w (in pixels) using the following equation:

$$FOV_h = 2\arctan\left(\frac{w}{2f}\right) \tag{8}$$

Then change FOV for the 3D terrain engine using FOV_h value.

2. Estimate initial pose (longitude, latitude, altitude) and (azimuth, elevation, roll) from GPS/IMU.
3. Move the virtual camera according to initial pose.
4. Capture the real image from the mounted camera and the virtual image from the 3D terrain engine. Then apply intensity normalization on them, next find best correspondences between the two images automatically using SIFT and pairwise matching algorithms after filtering them. This filtering process done by sorting matches distances in ascending order, then we take the first N=30 points that spread in the whole image (we reject points that are near taken points).
5. Estimate the robust Homography matrix H from the correspondences by rejecting the outliers using RANSAC (RANdom SAmple Consensus) [21]. Apply the Homography matrix to reject the corresponding points that have a distance more than a predefined threshold.
6. Get the 3D coordinates of the accepted correspondences from the 3D terrain engine.
7. Test if these points are coplanar. If coplanar then rotate virtual camera multiple times to obtain a new multi-views, next apply coplanar calibration algorithm with multiple views [22], else if non-coplanar then apply the non-coplanar calibration algorithm with one view [8].
8. Use the estimated rotation matrix and the translation vector to extract global

camera pose.

9. Move the virtual camera according to the estimated pose, then transmit a virtual ray from the target pixel according to this pose until intersect the 3D terrain model in the requested point.

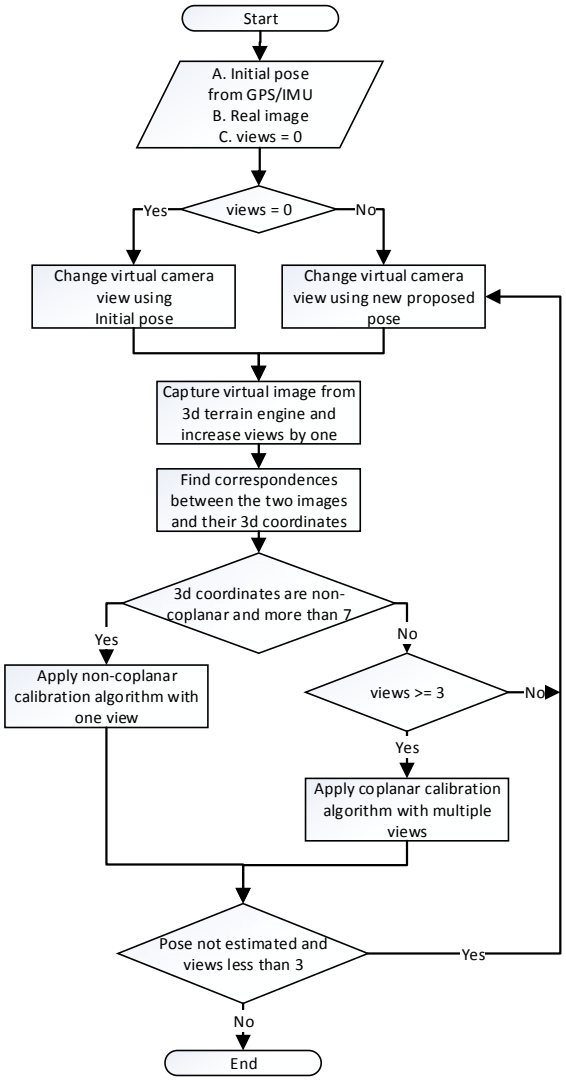


***Figure (5):** Pose estimation approach*

6. Testing environment:

We created a new testing environment **MapViewer.exe** using C++Builder XE5 as shown in Figure (6). This testing environment contains the following tools:

6.1. 3D terrain engine:

We define a 3D terrain engine in this work by a 3D software model or representation of the Earth that provides the user with the ability to move around in the virtual environment freely (by changing the viewing angle, position, and built using OpenGL library). In these days, exists many 3D terrain engines that we can use, such as Google Earth, NASA World Wind and Bing Maps. Through the research we used Google Earth because it has a COM interface (Component Object Model) which let us control the camera view programmatically from our application as shown in Figure (7).

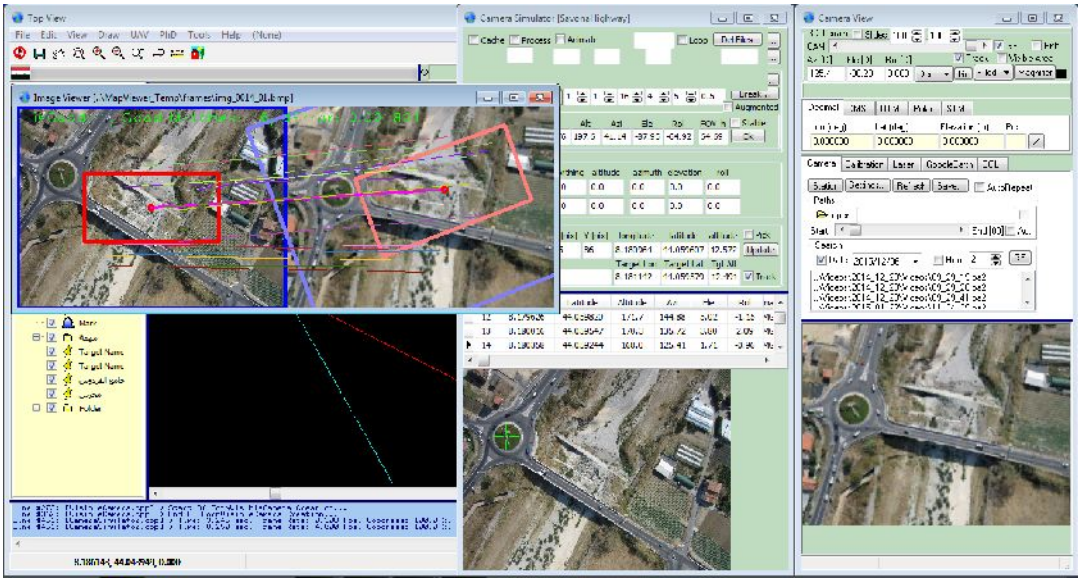


Figure (6): *Our Testing Environment (MapViewer)*

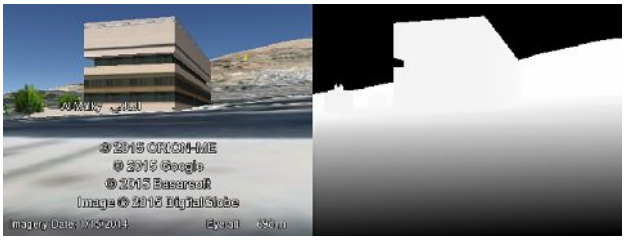


Figure (7): *Google Earth 3D Engine (Virtual image and depth map)*

6.2. Camera simulator:

Camera simulator is an image processing module which let us test computer vision algorithms as shown in Figure (8), this module consists of the following pieces:

- Pose estimation parameters: programmed using OpenCV library [23] to estimate the precise pose of the camera using 3D terrain engine (as described above).

• Errors: give us the possibility to add additive Gaussian noise for the telemetry data to test our algorithms in synthetic mode.

• Reference: let us define a target in image coordinates and the corresponding true geographic coordinates to use them as reference when we validate our algorithm. This piece also finds automatically geographic coordinates from 3D Terrain Engine that correspond to the target pixel and show the errors in meters between the two results.

• Image data: we store for each image a record in database consists of the following fields: ID, Longitude, Latitude, Altitude, Azimuth, Elevation, Roll and Filename. These fields define one image with its telemetry data taken from auxiliary sensors IMU/GPS.

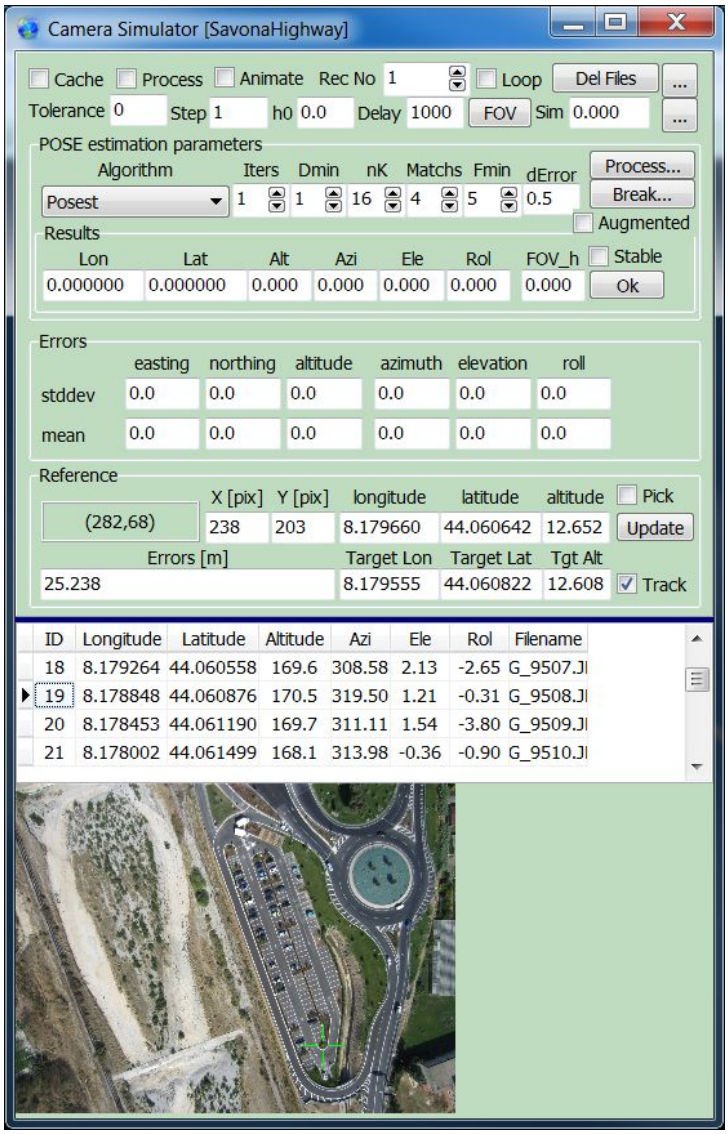


***Figure (8):** Camera Simulator (MapViewer)*

7. Experimental results:

We tested our approach on synthetic and real images with dimensions of 384 × 288 pixels on a PC with Intel CPU i7 4-cores and NVIDIA GeForce GTX 570 graphics card and 4 GB RAM, the frame rate was 12 FPS. We used UTM (Universal Transverse Mercator) coordinate system in tests instead of geographic coordinates because they are in meters. We use a dataset of images taken from a UAV with the following information:

- Place name: Savona Highway Exit
- UAV Type: Fixed wing
- Camera: Canon Ixus 220HS
- Number of images: 98
- Number of strips: 8
- Flight quote: 160 m
- Image format: 4000 x 3000 pixel

We use our camera simulator module to select randomly a terrain target for each image in the dataset. For each target, we determine its image coordinates (in pixels) and the corresponding true geographic coordinates.

7.1. Before registration:

We found automatically the geographic coordinates which correspond to the target pixel (stored previously) using 3D Terrain Engine by ray tracing algorithm and we obtained the results shown in Figure (9) and Figure (10). The mean error was 30 meters.

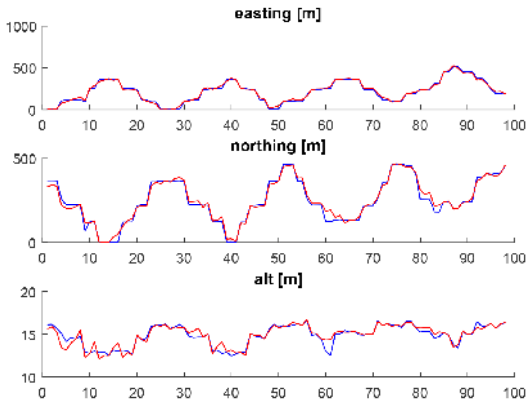
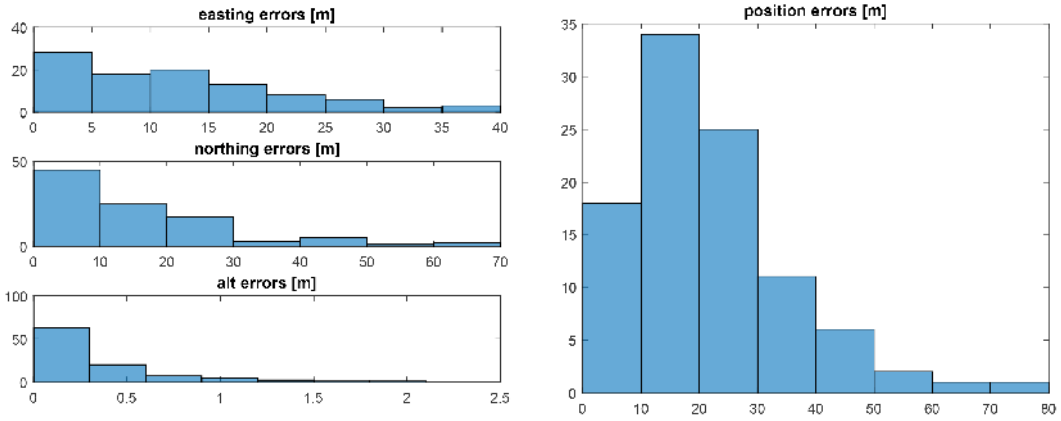


Figure (9): *Localization results (before registration), where horizontal axis shows frame index and the vertical axis shows values (blue curve show the reference locations and the red curve show estimated locations).*



(a) (b)

Figure (10): *Histogram of localization errors (before registration). (a) Horizontal axis shows errors (as a distance between the reference and estimated coordinates in meters as UTM projection) and the vertical axis shows occurrence count. (b) Horizontal axis shows errors (as a distance between the reference and estimated location in meters) and the vertical axis shows occurrence count.*

7.2. After registration:

We create a new dataset from the noisy one after correcting camera pose for each record in the dataset by applying our pose estimation algorithm (when the algorithm failed, we use the noisy pose). Now, we found automatically the geographic coordinates which correspond to the target pixel (stored previously) using 3D Terrain Engine by ray tracing algorithm and we obtained the results shown in Figure (11) and Figure (12). The mean error was 4 meters.

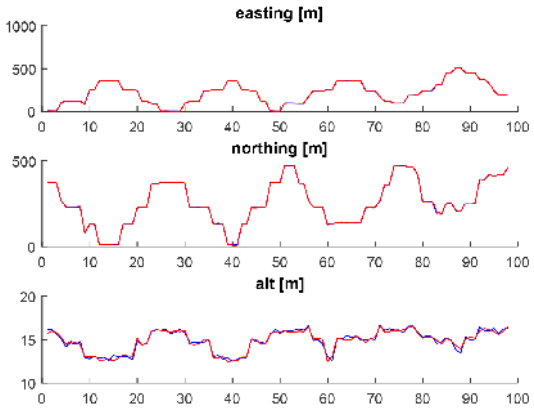
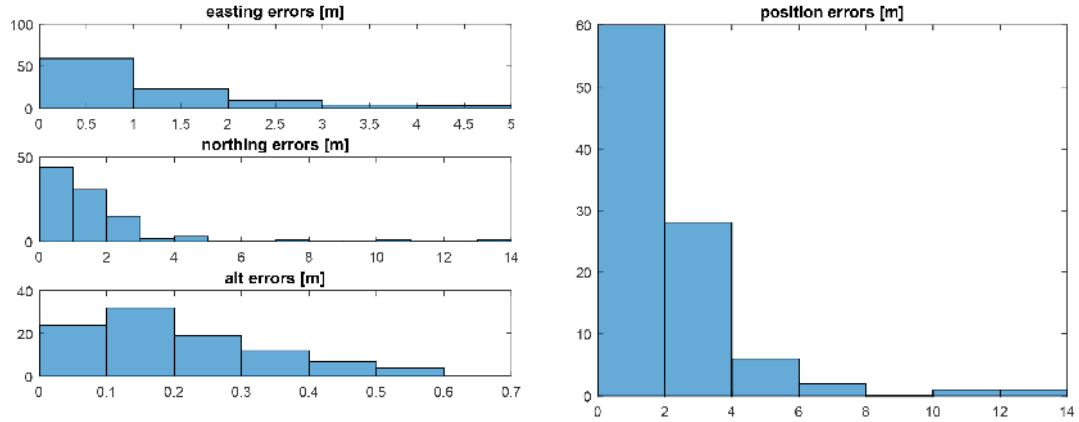


Figure (11): *Localization results (after registration), where horizontal axis shows frame index and the vertical axis shows values (blue curve show the reference locations and*

the red curve show estimated locations).



(a) (b)

***Figure (12):** Histogram of localization errors (after registration). (a) Horizontal axis shows errors (as a distance between the reference and estimated coordinates in meters as UTM projection) and the vertical axis shows occurrence count. (b) Horizontal axis shows errors (as a distance between the reference and estimated location in meters) and the vertical axis shows occurrence count.*

8. Conclusions and Future Work:

We presented a new approach for localizing a ground target from a UAV using 3D terrain engine. We used inaccurate auxiliary sensors on the UAV to obtain an approximate measurement of the camera pose. This pose used to move the virtual camera inside the engine, then automatically we found multiple matches between the two images to find the 3D coordinates of the matches using 3D terrain engine. Finally, we test the co-planarity of the 3D points under the camera, depending on this test, we use coplanar or non-coplanar algorithm to estimate accurate global camera pose. The accurate pose is used for localizing targets seen in the image by transmitting a virtual ray from a pixel according to the camera pose until intersect the 3D terrain model in the requested point. We tested the proposed approach on a synthetic and real data. Experimental results proved the feasibility and robustness of the proposed approach and the precision is the same order as the 3D terrain engine. Our approach works well with assumptions that the UAV flies at a suitable altitude. This depend on the 3D terrain engine used, for example when we use a 3D terrain engine which built on a DEM (Digital Elevation Model) we must fly with high altitudes to compensate the insufficient 3D model precision, but when we use a 3D City Model as engine this constraint is not needed. In addition, our system is near real time (12 FPS). However, our proposed approach shows a promising solution to target localization from a UAV with a camera

data and a 3D terrain engine. For more accurate and robust target localization, we will carry out additional studies on using 3D City models as a 3D engine, and we will work in the future on auto-calibration algorithms using 3D terrain engine to allow the user to change the field of view online, without the need to recalibrate offline. Finally, we can say that the 3D terrain engine succeeded when other methods failed.

References:

[1] K.-H. Son, Y. Hwang, and I. S. Kweon, *Uav global pose estimation by matching forward-looking aerial images with satellite images*, in Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. IEEE, 2009, pp. 3880–3887.

[2] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, International journal of computer vision, vol. 60, no. 2, pp. 91–110, 2004.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, *Speeded-up robust features (surf)*, Computer vision and image understanding, vol. 110, no. 3, pp. 346–359, 2008.

[4] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele, *Real-time image-based 6-dof localization in large-scale environments*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2012), June 2012.

[5] A. K. Agrawal, C. Shekhar, P. David, and J. DeHart, *Mapping ground video to aerial dem's*, 2009.

[6] F. Samadzadegan, M. Hahn, and S. Saeedi, *Position estimation of aerial vehicle based on a vision aided navigation system*, Proceedings of Visualization and Exploration of Geospatial Data, Stuttgart, 2007.

[7] B. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nolle,¨ *Review and analysis of solutions of the three point perspective pose estimation problem*, International journal of computer vision, vol. 13, no. 3, pp. 331–356, 1994.

[8] M. Lourakis and X. Zabulis, *Model-based pose estimation for rigid objects*, in Computer Vision Systems. Springer, 2013, pp. 83–92.

[9] V. N. Dobrokhodov, I. Kaminer, K. D. Jones, R. Ghabcheloo, et al., *Vision-based tracking and motion estimation for moving targets using small uavs*, in American Control Conference, 2006. IEEE, 2006, pp. 6–pp.

[10] D. B. Barber, *Accurate target geolocation and vision-based landing with application to search and engage missions for miniature air vehicles*, 2007.

[11] G. Conte, M. Hempel, P. Rudol, D. Lundstrom,¨ S. Duranti, M. Wzorek, and P. Doherty, *High accuracy ground target geo-location using autonomous micro aerial vehicle platforms*, in Proceedings of the AIAA-08 Guidance, Navigation, and Control Conference, 2008.

[12] A. Irschara, C. Zach, and H. Bischof, *Towards wiki-based dense city modeling*, in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on. IEEE, 2007, pp. 1–8.

[13] J. Heikkila and O. Silven, *A four-step camera calibration procedure with implicit image correction*, in Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on. IEEE, 1997, pp. 1106–1112.

[14] A. Irschara, V. Kaufmann, M. Klopschitz, H. Bischof, and F. Leberl, *Towards fully automatic photogrammetric reconstruction using digital images taken from UAVs*. na, 2010.

[15] A. Lingua, D. Marenchino, and F. Nex, *Performance analysis of the sift operator for automatic feature extraction and matching in photogrammetric applications*, Sensors, vol. 9, no. 5, pp. 3745–3766, 2009.

[16] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, *A comparison of affine region detectors*, International journal of computer vision, vol. 65, no. 1-2, pp. 43–72, 2005.

[17] C. Silpa-Anan and R. Hartley, *Optimised kd-trees for fast image descriptor matching*, in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008, pp. 1–8.

[18] K. Fukunaga and P. M. Narendra, *A branch and bound algorithm for computing k-nearest neighbors*, Computers, IEEE Transactions on, vol. 100, no. 7, pp. 750–753, 1975.

[19] M. Muja and D. G. Lowe, *Fast approximate nearest neighbors with automatic algorithm configuration*. VISAPP (1), vol. 2, 2009.

[20] T. Petersen, *A comparison of 2d-3d pose estimation methods*, Master's thesis, Aalborg University-Institute for Media Technology Computer Vision and Graphics, Lautrupvang, vol. 15, p. 2750.

[21] M. A. Fischler and R. C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, vol. 24, no. 6, pp. 381–395, 1981.

[22] Z. Zhang, *A flexible new technique for camera calibration*, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, no. 11, pp. 1330–1334, 2000.

[23] Itseez, *Opencv*, 2015, [Online; accessed 7-December-2015]. [Online]. Available: http://itseez.com