J Opt (December 2017) 46(4):425–435
DOI 10.1007/s12596-017-0395-0

RESEARCH ARTICLE

Target geo-localization based on camera vision simulation of UAV

Prashanth Pai[1] · **V. P. S. Naidu**[2]

Received: 20 July 2016 / Accepted: 28 March 2017 / Published online: 4 April 2017

Abstract This paper presents a simulation study on estimating the Geo-Location of a target based on multiple image of the target taken from a gimbaled camera mounted on a unmanned aerial vehicle (UAV), which orbits around the target with a radius such that the target is always in the field of camera vision. The Camera Vision Simulation of the UAV is implemented by using an ortho Geo-TIFF (Geo-Spatial Tagged Information File Format) as imagery reference, positional and attitude attributes of UAV, Gimbal and Camera and internal characteristic of the simulated Camera. Target is localized using the simulation images taken from multiple bearing waypoints by applying the Geo-Location algorithm using the simulation parameters as reference. For improving the accuracy of the estimation, error reduction techniques like true average, moving average and recursive least square are also suggested and implemented.

Keywords Target Geo-Location · Multiple bearing · Simulation · Geo-TIFF · Unmanned aerial vehicle (UAV) · Error reduction technique

✉ V. P. S. Naidu
vpsnaidu@gmail.com

Prashanth Pai
pp251@student.le.ac.uk

[1] Department of Engineering, University of Leicester, Leicester, UK

[2] Multi Sensor Data Fusion Lab, CSIR-National Aerospace Laboratories, Bangalore, India

Introduction

Unmanned aerial vehicles (UAV) are aircrafts without a human pilot aboard. UAV are preferably used in carrying out mission considered to be "Dull, Dirty and Dangerous" as suggested by Tice [1]. UAV application mainly involves the sensors placed in the aerial vehicle for collecting the surrounding environment data which are of particular interest and sending it to Ground Control Station for performing the necessary operations. Target Geo-Location is one such application, where the visual data captured from the mounted gimbal camera placed on UAV is sent to the Ground Control Station for determining the Geo-Location Coordinate of a Target Point of Interest which is present in the camera vision.

This paper presents a simulation approach for validating and verifying the Target Geo-Location Algorithm suggested by Pratyusha [2]. The work done in the paper Target Geo-Location [2] by Pratyusha mainly involved determination of Target based on single bearing image data as reference. This paper suggests determining the geo-location of a target using multiple bearing image data, and also implementing error reduction techniques for improving the accuracy in the result.

The focus of this paper is to create a vision simulation of a gimbaled camera mounted on a UAV, based on the application of Geo-Location algorithm. The simulation UAV model imitates to take a circular path around the target with a particular orbital radius such that the target is always under field of vision. The paper also focuses on determining the Geo-Location of the target point of interest using the image generated from the simulation, and applying error reduction techniques for improving the accuracy of the estimate.

Estimation of circular waypoint

UAV is considered in this simulation to take a circular path around the target due to its ease of implementation and efficient capability of tracking target [3]. The vision simulation of a camera mounted on a UAV taking a circular path around a given target with a known radius is realized by generating the images to be taken only from the equally spaced circular waypoints in its path (Total of 36 equally spaced circular waypoints is considered in this study). Hence, estimation of circular waypoint is essential to determine the location of UAV from which the multiple bearing images will be taken for the simulation. The circular waypoint locations are determined by the application of Haversine's formula [4], as mentioned in Eq. (1) with the knowledge of geo-location coordinate of target, distance of separation (i.e. orbital radius) and bearing angle of separation.

Iterative 'i'

$$\begin{aligned} Uav\_ll(i,1) &= \sin^{-1}(\sin(t\arg et\_lat) * \cos(del) \\ &\quad + \cos(t\arg et\_lon) * \sin(del) * \cos(i * \tau)) \\ Uav\_ll(i,2) &= t\arg et\_lon + \arctan 2(\sin(i * \tau) * \sin(del) \\ &\quad * \cos(t\arg et\_lat), \cos(del) - (\sin(tgt\_lat) \\ &\quad * \sin(mav\_ll(i,1))) \end{aligned} \tag{1}$$

where, i is the sample number (iteration count or Bearing Angle index). $Uav\_ll(i,1)$ is the UAV latitude location in radians for 'i^{th}'sample, $Uav\_ll(i,2)$ is the UAV longitude location in radians for 'i^{th}'sample, $t\arg et\_lat$ is the latitude coordinate of the target considered in radians, $t\arg et\_lon$ is the longitude coordinate of the target considered in radians, $del = d/6371000$, 'd' is the radius of UAV orbit, 6,371,000 m is the Radius of the Earth, $\tau = 360/n$ is the fundamental bearing angle considered, n is the total number of circular waypoints.

Geo-location algorithm

The Simulation database creation (Synthetic Image Generation) and the Target Localization using the simulation image generated, are both implemented in this study, based on the Geo-Location Algorithm [2].

Geo-Location Algorithm basically relates the 2D pixel coordinate of an object present in image plane with its 3D world coordinate also called as inertial coordinate. In this work we apply the Geo-Location Algorithm for determining the real world coordinate with the knowledge of pixel coordinate of the point of interest. For applying the Geo-Location Algorithm with the pixel coordinates of the point of interest, it is also essential to know the different Coordinate frames, Transformation Matrices between

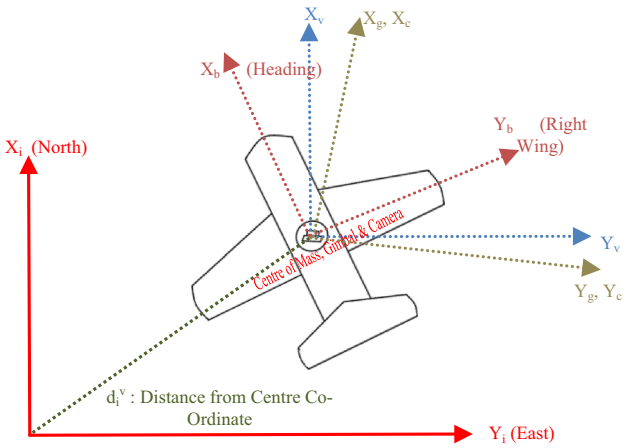

Fig. 1 Coordinate frames

Coordinate frames, camera intrinsic parameters such as Camera Calibration Matrix and Image Depth.

Co-ordinate frames

Coordinate frames involved in transformation of 2D pixel (Camera) coordinate to 3D real world (Inertial) coordinates [5] are (Fig. 1):

- *Inertial Frame* (X_i, Y_i, Z_i) Inertial frame or 3D real world frame coordinate describes an object location in real world coordinates with X_i describing its North position, Y_i describing East position and Z_i describing the distance from centre of the Earth.
- *Vehicle Frame* (X_v, Y_v, Z_v) Vehicle frame is co-linear with Inertial Frame, with the origin located at the centre of mass of the plane.
- *Body Frame* (X_b, Y_b, Z_b) The origin of the Body Frame is located at the centre of mass of the plane, with X_b, Y_b, Z_b, describing the vehicle nose, right wing and belly of the plane.
- *Gimbal Frame* (X_g, Y_g, Z_g) Frame originates at the gimbal centre of rotation and is oriented so that X_g describes the direction of optical axis, Z_g describes the width direction in the image plane, Y_g describes the height direction in the image plane.
- *Camera Frame* (X_c, Y_c, Z_c) Camera Frames has the origin located at optical centre of the camera. X_c, Y_c and Z_c describing the width direction in image plane, height direction in image plane and direction of optical axis of the camera lens.

Transformation matrices

The transformations matrix is important to relate the position of the target point of interest in the camera frame coordinate (image plane) to its position in the inertial frame coordinate

(real world 3D location). Transformation matrix is defined between adjacent relative Coordinate frames, and consists of translatory vector and rotational matrix. The transformation matrices used for determining the Target Geo-Location from its image coordinate are mentioned below [5].

Transformation from inertial to vehicle frame

As mentioned in "Coordinate frames" section Coordinate frames, the inertial frame and vehicle frame are co-linear with each other. Hence, transformation matrix T_i^v can be defined by translation between the origins of the frame as mentioned in Eq. (2).

$$T_i^V = \begin{bmatrix} I & -d_i^V \\ 0 & 1 \end{bmatrix} \tag{2}$$

where,

$$d_i^V = \begin{bmatrix} x_{UAV} \\ y_{UAV} \\ -h_{UAV} \end{bmatrix}$$

x_{UAV} and y_{UAV} represents the East (Latitude) and North (Longitude) Location of UAV calculated from GPS, h_{UAV} is the altitude of UAV.

Transformation from vehicle frame to body frame

Body frame provides the attitude of the UAV with respect to the Vehicle Frame. Hence transformation matrix can be defined using rotational angles of UAV, roll (θ), pitch (φ) and yaw (ψ) as mentioned in Eq. (3).

$$T_b^v = \begin{bmatrix} R_v^b & 0 \\ 0 & 1 \end{bmatrix} \tag{3}$$

where,

$$R_v^b = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\varphi s_\theta c_\psi - c_\varphi s_\psi & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & s_\varphi c_\theta \\ c_\varphi s_\theta c_\psi + s_\varphi s_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi & c_\varphi c_\theta \end{bmatrix}$$

$c_\varphi = \cos\varphi$ and $s_\varphi = \sin\varphi$.

Transformation from body frame to gimbal frame

Transformation from body to gimbal frame is given by the translation vector d_b^g, which defines the distance of separation between the two frames and rotational matrix R_b^g, which defines the rotational orientation of gimbal with respect to the UAV body. The rotational orientation can be determined by using angles which defines the 2D orientation of a gimbal namely, pan (α_{az}) and tilt (α_{el}).

$$T_b^g = \begin{bmatrix} R_b^g & -d_b^g \\ 0 & 1 \end{bmatrix} \tag{4}$$

where,

$$R_b^g = R_{y,\alpha_{el}} R_{z,\alpha_{az}} = \begin{bmatrix} c_{el}c_{az} & c_{el}s_{az} & s_{el} \\ -s_{el} & c_{az} & 0 \\ -s_{el}c_{az} & -s_{el}s_{az} & c_{el} \end{bmatrix}$$

$d_b^g = [X_b^g, Y_b^g, Z_b^g]^T = [0,0,0]^T$ is the translatory vector. (In this simulation, the center of mass of body and gimbal is considered to be the same),α_{az} denotes the azimuth angle of rotation about Z_g and α_{el} denotes the elevation angle of rotation about Y_g after α_{az},$c_{el} = \cos\alpha_{el}$ and $s_{az} = \sin\alpha_{az}$.

Transformation from gimbal frame to camera frame

Transformation from gimbal frame to camera frame is used for aligning the image frame with respect to the gimbal frame. In this experiment, camera orientation is kept such that camera frame is collinear with gimbal frame, and the gimbal center of mass and optical center is assumed to be the same location. Hence, the transformation matrix T_g^c is given by as mentioned in Eq. (5).

$$T_g^c = \begin{bmatrix} R_g^c & -d_g^c \\ 0 & 1 \end{bmatrix} \tag{5}$$

where,

$$R_g^c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$d_g^c = [X_g^c, Y_g^c, Z_g^c]^T = [0,0,0]^T$ is the translatory vector between camera and gimbal frame.

Camera Calibration Matrix

The perspective transformation provides the relationship between the position of target point of interest in camera frame (Pixel coordinates) with its location in inertial frame (3D world coordinates). But this accounts only the external parameters (positional orientation) not the internal characteristics of the camera, which also defines where a 3D real world points would be mapped in the camera image frame. The internal characteristics of a camera can be defined Camera Calibration Matrix (C_c) [6] as mentioned in Eq. (6).

$$C^c = \begin{bmatrix} f_x & f_\theta & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where, C_c is the Camera Calibration Matrix, f_x and f_y is the focal length measured along width pixel and height pixel of image plane in meters, f_θ is the skew of the pixel in meters, c_x and c_y is the principal point of the camera in pixels.

Image Depth

In order to compute the 3D world coordinate from the pixel coordinate, estimating the transformation matrix which accounts for external parametric variation and camera matrix which defines the intrinsic characteristic of a camera is essential. Along with these two parameters, it is also required to measure an unknown parameter called Image Depth λ, which gives the information on the real world distance between camera centre and target point of interest along the optical axis (Fig. 2). The Image Depth λ can be measured [6] by using Eq. (7).

$$\lambda = \frac{Z^i_{cc}}{Z^i_{cc} - Z^{-i}_{obj}} \tag{7}$$

where, Z^i_{cc} is the z component of P^i_{cc},$P^i_{cc} = [T^c_g T^g_b T^b_v T^v_i]^{-1} P^c_{cc}$ is the location of center of camera in inertial frame, $P^c_{cc} = [0 \quad 0 \quad 0 \quad 1]^T$ is the location of center of camera in camera frame, Z^{-i}_{obj} is the z component of P^{-i}_{obj}, $P^{-i}_{obj} = [C_c T^c_g T^g_b T^b_v T^v_i]^{-1} P^c_{cc}$ is the un-scaled location of the object in inertial frame.

Determining geo-location from pixel coordinate

Geo-Location coordinate of the target point of interest located at pixel coordinate $q = (x_{ip} \quad y_{ip} \quad 1 \quad 1)^T$ in the image plane can be determined by using the transformation matrices, camera matrix and Image Depth calculated [7] in "Transformation matrices" "Camera calibration matrix" "Image depth" sections in Eq. (8).

$$P^i_{obj} = \left[C_c \cdot T^c_g \cdot T^g_b \cdot T^b_v \cdot T^v_I\right]^{-1} \Lambda q \tag{8}$$

where, P^i_{obj} Geo-Location coordinate of the target, $\Lambda = \begin{pmatrix} \lambda I & 0 \\ 0 & 1 \end{pmatrix}$ and λ is the Image Depth.

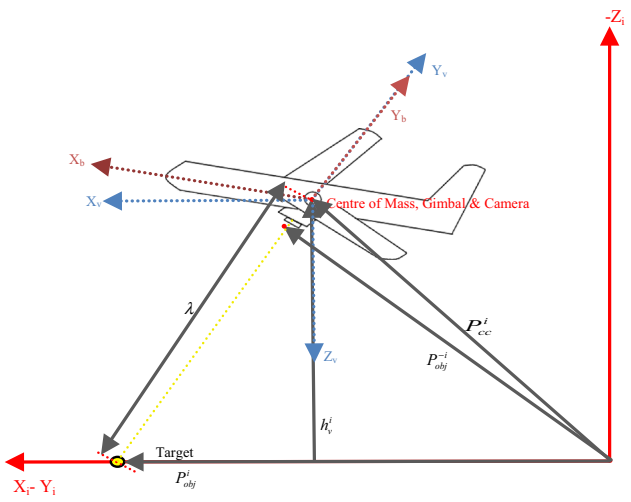

Fig. 2 Visualization of the Image Depth

The Target Geo-Location can also be determined by the knowledge of parameters used to calculate Image Depth λ by applying triangular law [7] in Fig. 3, as mentioned in Eq. (9).

$$P^i_{obj} = [Utmx, Utmy, Utmz, 1]^T = P^i_{cc} + \lambda(P^{-i}_{obj} - P^i_{cc}) \tag{9}$$

where, P^i_{cc} is the location of centre of camera measured in inertial frame and P^{-i}_{obj} is the un-scaled geo-location coordinate of the target.

Synthetic scene generation

Synthetic Scene Generation or Simulation Image Generation is process of creating images from a simulated camera model placed on a UAV, by using aerial Geo-TIFF (Geospatial Tagged Image File Format) for imagery reference. The Synthetic Scene Generation is achieved in this study by the application of Geo-Location Algorithm. The algorithm is used for determining the Geo-Location Coordinates of each and every pixels of the simulated camera model by using the simulation model parameters defined for the UAV, Gimbal and the Camera which includes various positional and attitude attributes, along with intrinsic parameter (For Camera only). The UAV Heading Angle parameter is chosen to be the negative value of Bearing Angle for providing a more realistic simulation (Fig. 3). The obtained pixel raster matrix results of Geo-Location coordinate values are then filled with RGB imagery data value depending on the following conditions:

- If the Image Depth λ estimated for a given pixel in the raster matrix is found to be zero, then the pixel is shaded with sky blue, indicating the sky region.
- If the Geo-Location Coordinate value finds a pixel in the Aerial Geo-TIFF image having a matching Geo-Location Coordinate value tagged with it, then the imagery data for the given pixel is filled using the matching pixel imagery data found in Geo-TIFF.
- If the Geo-Location Coordinate value in the raster matrix doesn't contain any matching pixel in Geo-TIFF file having the same location coordinate value tagged, is shaded with black, indicating that the imagery data for the given geo-location is unknown.

The Matlab code snippet for generating Synthetic Image for a single bearing image is given below. *uav, gimbal,- camera* are structures defining parameters involved with UAV, Gimbal and Camera, *A, info, lattiff, lontiff* provides information related with Geo-TIFF file with *latiff, lontiff* arrays providing latitude and longitude information for defining the co-ordinate information of each pixel in Geo-TIFF.

```
function out =
scenegenerate(uav,gimbal,cam,A,info,lattiff,lontiff)
% lat,lon to utm conversion
[utmx, utmy, zone] = ll2utm(uav.lat, uav.lon);

% Inertial frame to Vehicle frame translation
vXi = utmx; vYi = utmy; vZi = uav.alt;
vTi = htransl(-vXi, -vYi, -vZi);

% Vehicle to Body frame transformation
bTv = hrotx(deg2rad(-uav.phi)) * hroty(deg2rad(-
uav.theta)) * hrotz(deg2rad(-uav.psi));

% Body to Gimbal frame transformation
alpha = 180-gimbal.alpha; beta = gimbal.beta;
gTb = htransl(-gimbal.bXg, -gimbal.bYg, -gimbal.bZg)
* hrotx(deg2rad(-beta)) * hroty(deg2rad(-alpha));

% Gimbal frame to Camera frame translation and
rotation
cTg = htransl(-cam.gXc, -cam.gYc, -cam.gZc) *
eye(4);

% Camera Calibration Matrix
C = hcam(cam.f, cam.sx, cam.sy, cam.cx, cam.cy);
x_ip = repmat(1:cam.w,1,cam.h);
y_ip = repmat(1:cam.h,cam.w,1); y_ip = y_ip(:)';
q = [x_ip; y_ip; ones(size(x_ip));
ones(size(x_ip))];
i_pbar_obj = (C * cTg * gTb * bTv * vTi) \ q;
c_p_cc = [0; 0; 0; 1];
i_p_cc = (cTg * gTb * bTv * vTi) \ c_p_cc;

% Image Depth
lambda = i_p_cc(3)./(i_p_cc(3) - i_pbar_obj(3,:));
Q = [lambda.*x_ip; lambda.*y_ip; lambda;
ones(size(lambda))];

%Geo-Location Estimation for each pixel in the Image
 i_p_obj = inv(C * cTg * gTb * bTv * vTi) * Q;
 [cam.lats, cam.lons] =
 utm2ll(i_p_obj(1,:),i_p_obj(2,:),zone*ones(size(x_ip
 )));
 xtif = round((cam.lons -
 info.BoundingBox(1,1))/info.PixelScale(1,1));
 ytif = size(A,1) - round((cam.lats -
 info.BoundingBox(1,2))/info.PixelScale(2,1));
 ytif1 = round((cam.lats -
 info.BoundingBox(1,2))/info.PixelScale(2,1));
% Generate Image by Mapping the Obtained Pixel Geo-
Location
%Co-Ordinate with Geo-Tiff Information
for n = 1:length(x_ip)

 % If Image Depth is negative, shade pixel with blue
    if lambda(n) <= 0
        cam.img(cam.h+1-y_ip(n),cam.w+1-x_ip(n),:) =
[79, 110, 176];

% Pixel with unavailable imagery data shaded with
black
    elseif (xtif(n)<=0 || ytif(n)<=0 ||
xtif(n)>info.Width || ytif(n)>info.Height)
        cam.img(cam.h+1-y_ip(n),cam.w+1-x_ip(n),:) =
[0, 0, 0];

%Imagery data taken from matching pixel of Geo-TIFF
    else
        cam.img(cam.h+1-y_ip(n),cam.w+1-x_ip(n),:) =
A(ytif(n),xtif(n),:);
        cam.imglats(cam.h+1-y_ip(n)) =
lattiff(ytif(n));
        cam.imglons(cam.w+1-x_ip(n)) =
lontiff(xtif(n));
    end
end
out=cam;
```

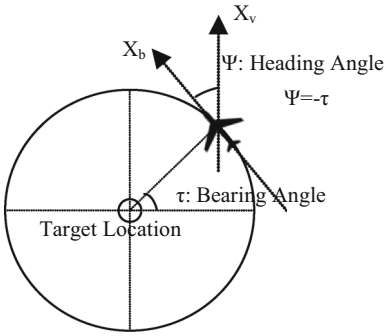

Fig. 3 UAV Heading Angle variation for change in bearing angle

Estimation of target geo-location

Target Geo-Localization is the process of estimating the real world location of a ground-based target present in the image taken from a camera placed in UAV, by applying Geo-Location Algorithm. In this study the geo-location of a target is estimated for multiple bearing image generated in simulation by applying the Geo-Location algorithm as mentioned in Eq. (8). The same UAV, Gimbal and Camera Parameters which were used for simulation of a given bearing image. The Matlab code snippet for the determining Geo-Location of target present in bearing image is given below. *x_ip, y_ip* describes the pixel coordinate of the target in the image, *cam, gimbal, uav* describes the simulation parameter involved with the UAV, Gimbal and Camera. Code for computing the Transformation and Camera Matrix is not mentioned in snippet as the steps for computing are similar to the ones mentioned in simulation image generation. *i_p_cc, lambda, i_pbar_obj* are also saved as output for further computations.

```
function out=geolocation(x_ip,y_ip,cam,gimbal,uav)
%
Code Space for Transformation Matrix Computation and
Camera Matrix Computation remains the same as
mentioned in Scene Generation
%
%Unscaled Object Target Location in Inertial Frame
out.i_pbar_obj = (C * cTg * gTb * bTv * vTi) \ q;
c_p_cc = [0; 0; 0; 1];

%Camera Center Location w.r.t Inertial Co-Ordinates
out.i_p_cc = (cTg * gTb * bTv * vTi) \ c_p_cc;

% Image Depth
out.lambda = out.i_p_cc(3)./(out.i_p_cc(3) -
out.i_pbar_obj(3,:));

%  Target Geo-Location Estimation
if out.lambda > 0
    Q = [out.lambda.*x_ip; out.lambda.*y_ip;
out.lambda; ones(size(out.lambda))];
    i_p_obj = inv(C * cTg * gTb * bTv * vTi) * Q;
    [out.lat, out.lon] =
utm2ll(i_p_obj(1,:),i_p_obj(2,:),zone);
end
```

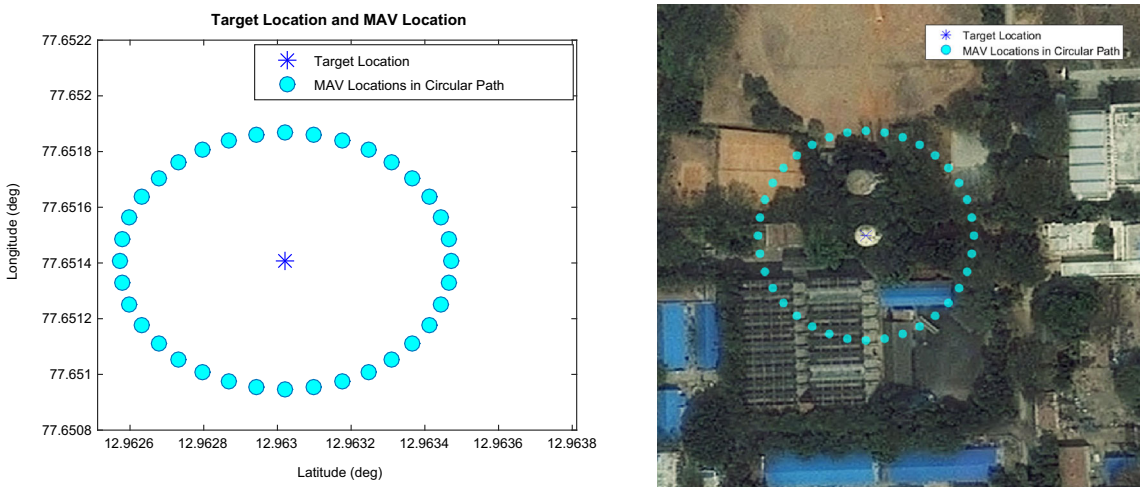

Fig. 4 Circular waypoint considered in the Simulation

Error reduction techniques

The estimation of geo-location coordinate calculated for each images taken from different bearing angles are bound to be error prone even if ideal conditions are considered in the simulation. The error in simulations like in this study can be mainly due to the inaccurate localization of pixel coordinate, which can be due to either, incorrect selection of pixel coordinate or target being located at integer pixel element, or both the criteria. Also, when a practical real time experiment is considered, the error in estimation can increase due to the uncertain parametric environment created by different error bias [8]. Hence for reducing the error observed in the estimation of geo-location coordinates, localization methods based on multiple bearing images can be considered. Averaging is one such method, where results of multiple bearing images are considered to reduce localization error. The Averaging methods used in the paper are.

Average estimation

Average Estimation is determined by computing the arithmetic mean of estimation as shown in Eq. (10).

$$\begin{aligned} Utmx\_tavg &= \frac{1}{n}\sum_{i=1}^{n} Utmx(i) \\ Utmy\_tavg &= \frac{1}{n}\sum_{i=1}^{n} Utmy(i) \end{aligned} \tag{10}$$

where, *Utmx_tavg, Utmy_tavg* is the True Average Value of target x and y coordinate in UTM coordinates, n is the total sample number (n = 36 in this study), *UTMx(i), UTMy(i)* is the x and y UTM Coordinate true value computed for *ith* sample number.

Moving average estimation

Moving Average Estimation is determining the mean of subset of value, where only a few recent determined from bearing sample number location of x and y UTM coordinate values are considered (in this study, 5 recent values chosen) for computing the mean of the subset. Moving Average computes only subset of values as mentioned in Eq. (11), in contrast with True Average value which considers the entire dataset.

$$\begin{aligned} Utmx\_mavg(k) &= \frac{1}{w}\sum_{i=k-w+1}^{k} Utmx(i) \\ Utmy\_mavg(k) &= \frac{1}{w}\sum_{i=k-w+1}^{k} Utmy(i) \end{aligned} \tag{11}$$

where, *Utmx_mavg(k), Utmy_mavg(k)* is the Moving Average Value determined for the k^{th} Sample Number, w is the window size for the Moving Average.

Recursive least square estimation

Recursive least square (RLS) is a simple method of recursively fitting a set of points to some function of choice by minimizing the sum of the squares of the offset of the points. The results obtained using RLS is identical to the true average, but the process of obtain it is more efficient. Another benefit of using RLS is that it provides intuition behind such results as the Kalman filter. The Matlab code snippet used for determining estimation from RLS

Table 1 UAV Circular waypoint Geo-Location coordinates calculated using Haversine's Formula

Target Geo-Location coordinate: Latitude: 12.963021580681730° Longitude: 77.651407302614828°			
UAV orbit radius: 50 m			
Sample number ('*i*')	Bearing angle ('τ') (in degrees)	UAV location coordinate	
		Latitude (degrees)	Longitude (degrees)
1	10	12.96355298	77.65150345
2	20	12.96352863	77.65159668
3	30	12.96348888	77.65168416
4	40	12.96343493	77.65176322
5	50	12.96336842	77.65183147
6	60	12.96329138	77.65188682
7	70	12.96320613	77.65192761
8	80	12.96311528	77.65195259
9	90	12.96302158	77.65196101
10	100	12.96292788	77.65195259
11	110	12.96283703	77.65192761
12	120	12.96275178	77.65188682
13	130	12.96267474	77.65183146
14	140	12.96260823	77.65176322
15	150	12.96255428	77.65168415
16	160	12.96251453	77.65159668
17	170	12.96249019	77.65150345
18	180	12.96248199	77.6514073
19	190	12.96249019	77.65131115
20	200	12.96251453	77.65121793
21	210	12.96255428	77.65113045
22	220	12.96260823	77.65105139
23	230	12.96267474	77.65098314
24	240	12.96275178	77.65092778
25	250	12.96283703	77.65088699
26	260	12.96292788	77.65086201
27	270	12.96302158	77.6508536
28	280	12.96311528	77.65086201
29	290	12.96320613	77.65088699
30	300	12.96329138	77.65092778
31	310	12.96336842	77.65098314
32	320	12.96343493	77.65105139
33	330	12.96348888	77.65113045
34	340	12.96352863	77.65121792
35	350	12.96355298	77.65131115
36	360	12.96356117	77.6514073

Algorithm is mentioned below. Here RLS Functions *A_n, A_n1,b_xn, b_xn1, b_yn, b_yn1, P_n and P_n1* are determined to compute *X_n1(j)* and *Y_n1(j)*, the x and y target coordinate value in inertial coordinate for the *jth* bearing sample number considered.

```
A_N=[];
for j=1:n
    nres=sprintf('FrameRes%d.mat',j*angle);
    nrest=strcat(pathname_tgl,'\',nres);
    estimate=load(nrest);
%Camera Center Location w.r.t Inertial Co-Ordinates
i_p_cc=estimate.Current_UavLocation.output_target.i_
p_cc;
%Image Depth
lambda=estimate.Current_UavLocation.output_target.la
mbda;
%Unscaled Object Target Location in Inertial Frame
i_pbar_obj=estimate.Current_UavLocation.output_targe
t.i_pbar_obj;
%Compute a_n1,b_xn1,b_yn1 values
a_n1=eye(1);
b_xn1=i_p_cc(1)+(lambda.*(i_pbar_obj(1)-i_p_cc(1)));
b_yn1=i_p_cc(2)+(lambda.*(i_pbar_obj(2)-i_p_cc(2)));
% Determine A_N,b_xn,b_yn,P_n values
    if(isempty(A_N))
        A_N= [a_n1];
        b_xn= [b_xn1];
        b_yn= [b_yn1];
        P_n=inv((A_N)'*A_N);
% Determine the X and Y UTM Co-Ordinate of Target
        X_n1(j)=P_n*(A_N)'*b_xn;
        Y_n1(j)=P_n*(A_N)'*b_yn;
    else
% Determine A_N,b_xn,b_yn,P_n values
        P_n1= P_n - ((P_n*(a_n1)'*(a_n1)*P_n)/(1 +
(a_n1*P_n*(a_n1)')));
        A_n1=[A_N' a_n1]';
        b_xn1=[b_xn' b_xn1]';
        b_yn1=[b_yn' b_yn1]';
% Determine the X and Y UTM Co-Ordinate of Target
        X_n1(j)=P_n1*(A_n1'*b_xn1);
        Y_n1(j)=P_n1*(A_n1'*b_yn1);
        P_n=P_n1;
        A_N=A_n1;
        b_xn=b_xn1;
        b_yn=b_yn1;
    end
end
```

Table 2 Simulation model parameters considered for image generation

Sample number location i = 1,2,…,36. Bearing angle $\tau = 360/n = 360/36 = 10°$		
UAV parameters	Gimbal parameters	Camera parameters
Latitude: *Uav_ll(i,1)* (in degrees)	Pan (α): 0°	*w x h*: 640 × 480 pixels
Longitude: *Uav_ll(i,2)* (in degrees)	Tilt (β): 0°	f_x: 142.8571428571 pixel units
Altitude: 60 m	X_b^g: 0 m	f_y: 142.8571428571 pixel units
Heading (ψ): $-(i*\tau)$ (degrees)	Y_b^g: 0 m	c_x: 320.5 pixel
Roll (θ): 0°	Z_b^g: 0 m	c_y: 240.5 pixel
Pitch (φ): 0°		X_g^c,Y_g^c,Z_g^c: 0 m

Results and discussions

The Simulation setup for the Unmanned Aerial Vehicle (UAV) equipped with a camera, orbiting around target is realized using Matlab. For this study, the target is chosen to

Fig. 5 UAV Simulation Images generated from different bearing angle Location (bearing angle (τ): *Top Left* 10°, *Top Right* 90°, *Bottom Left* 180°, *Bottom Right* 270°)

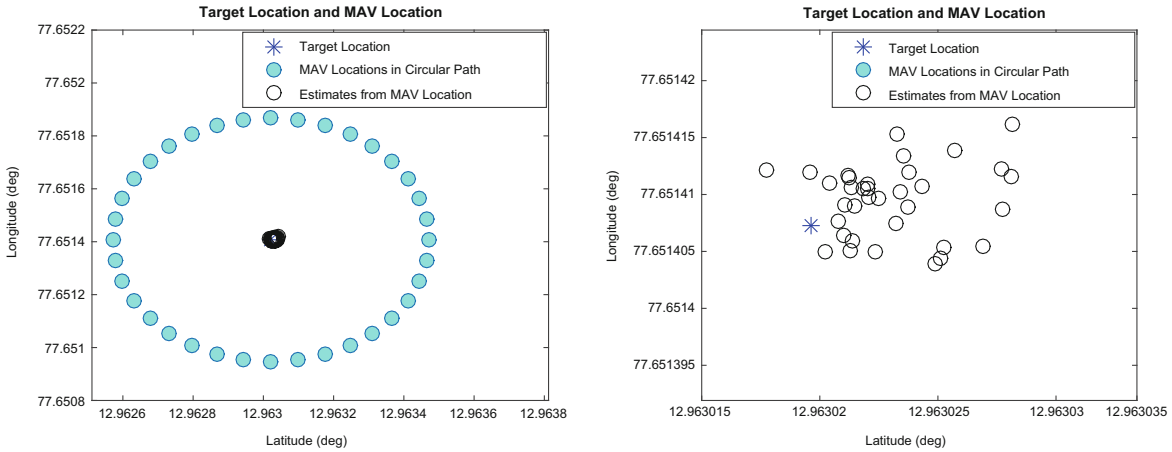

Fig. 6 Estimation of Target Geo-Location coordinate from different bearing angle

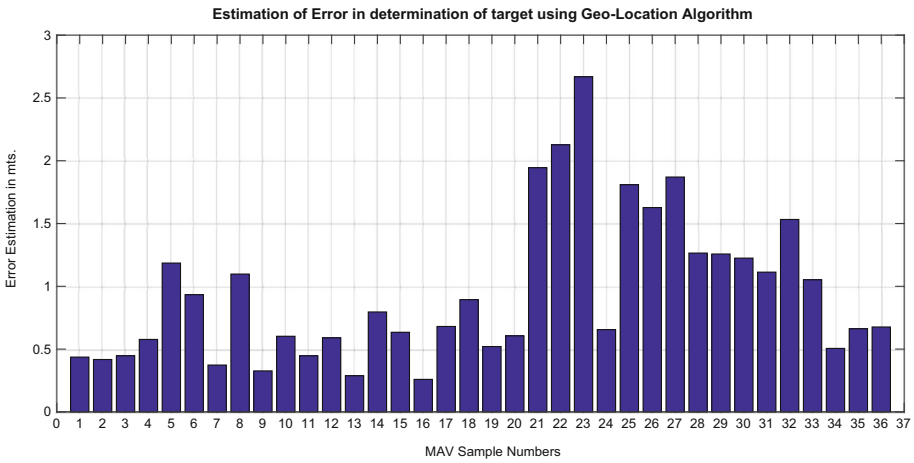

Fig. 7 Error in estimation of Geo-Location for each bearing sample number considered

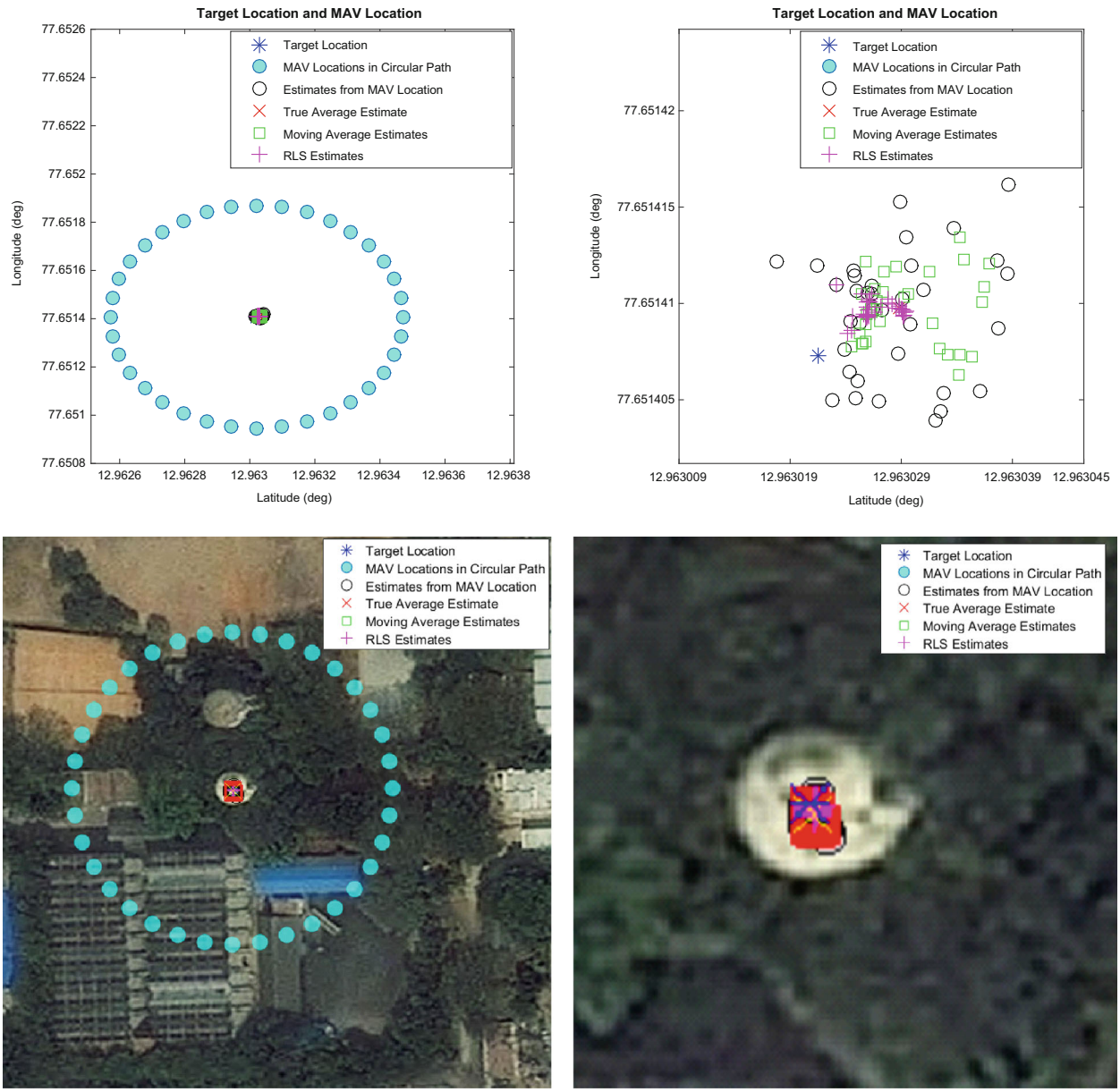

Fig. 8 Estimation of Geo-Location from different error reduction techniques

be NAL overhead Water Tank with Geo-Location Coordinate of Latitude 12.963021580681730° and Longitude 77.651407302614828° as per the Google Earth Geo-Coordinate database. The simulation UAV model is imitated to orbit the target with an altitude of 60 m and orbit radius of 50 m, so that the target chosen always remains under the camera field of view. Simulation Images are generated for 36 equally spaced circular waypoints (Fig. 4). The UAV Circular waypoint location calculated using Haversine Formula for each bearing angle with respect to target (Table 1).

The simulation images are generated for each of the 36 equally spaced waypoints using the Matlab code as mentioned in "Synthetic scene generation" section. The UAV, Camera and Gimbal Parameters used for simulation for each bearing angle τ is (Table 2). Synthetic Image generated for some bearing angles are displayed in (Fig. 5).

The Target Geo-Location is estimated for each of the images generated from the 36 equally spaced circular waypoint, by applying Geo-Location Algorithm using the user defined target pixel coordinate location and simulation parameter involved (Fig. 6).

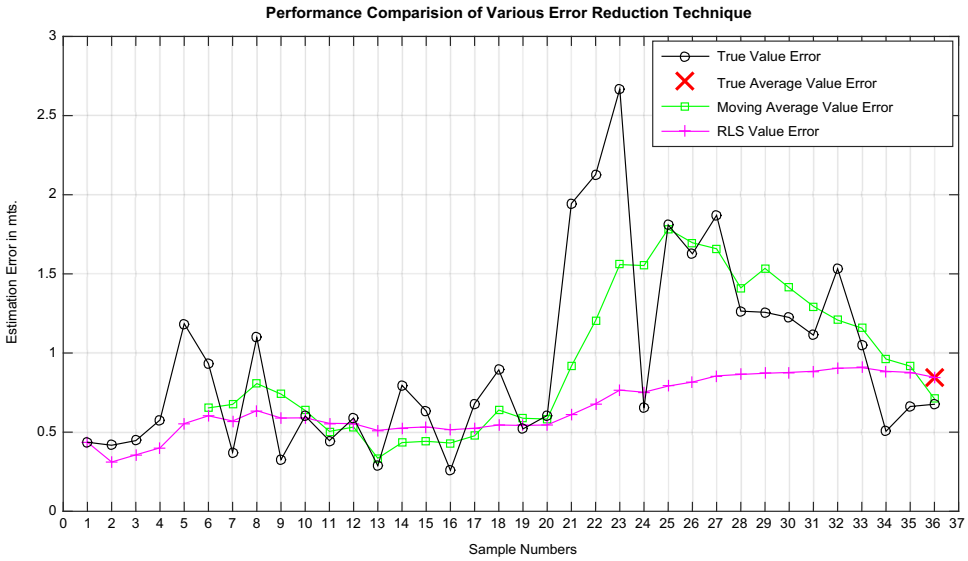

Fig. 9 Estimation error comparison of different error reduction techniques

Errors for each bearing angle true estimates are determined and a bar graph is plotted (Fig. 7). The average estimation, moving average estimation and RLS estimation techniques suggested in "Error reduction techniques" section are implemented for reducing the error. The estimations from various techniques used are plotted against target (Fig. 8) and errors from different estimation are compared with a line graph (Fig. 9).

Conclusion

The Geo-Location algorithm is used for generating Synthetic Scenery Image for each of the 36 equally spaced circular waypoint calculated using Haversine Formula. The target is first estimated by considering each and every bearing image independently, and later error reduction techniques are applied by considering multiple bearing images. From the results obtained in Geo-Location estimation using various error reduction techniques, it is clear that the RLS technique is more reliable with the estimation error constantly remaining under 0.8 m error, when compared with true estimation, where the peak error reading for a sample found to be nearly 2.6 m. For future work, the geo location algorithm and the error reduction techniques need to be practically implemented to test its efficiency in real time.

References

1. BP. Tice, Unmanned aerial vehicles:the force of the multiplier of the 1990s, Airpower J. V (1), (1991) https://web.archive.org/web/20090724015052/,Accessed on 19 Febraury 2016
2. PL Pratyusha, Estimation of ground Target Geo-Location using UAV onboard camera, M.Tech thesis, Department of Avionics, IST, JNTU, Kakinada, July 2015
3. TH. Summers, MR Akella, MJ Mears, Coordinated standoff tracking of moving target: control law and information architectures, J. Guid. Control. Dyn 32(1), 56–69 (2009). http://arc.aiaa.org/doi/abs/10.2514/1.37212, Accessed on 18 February, 2016
4. C. Veness, Calculate distance, bearing and more between Latitude/Longitude points, http://www.movable-type.co.uk/scripts/latlong.html, Accessed on 19 Febraury, 2016
5. R.W. Beard, T.W. McLain, *Coordinate frames, small unmanned aircraft theory and practice* (Princeton University Press, New Jersey, 2012), pp. 8–18
6. Y. Ma, S. Soatto, J. Kosecka, S.S. Sastry, *An invitation to 3-D vision: from images to geometric models* (Springer, Berlin, 2004)
7. J.D. Redding, T.W. McLain, R.W. Beard, C.N. Taylor, Vision-based target localization from a fixed wing miniature air vehicle, in *Proceedings of the 2006 American control conference, Minneapolis*, 2006. p.2862–2867
8. B. Barber, TW. McLain, B. Edwards, Vision-based landing of a fixed-wing miniature air vehicle, J. Aerosp. Comput. Inf. Commun. (2009), https://www.researchgate.net/publication/245439885, DOI: 10.2514/1.36201, Accessed on 19 February, 2016

Prashanth Pai

V. P. S. Naidu