

### Introduction:

The market for using unmanned aerial vehicles (UAV) to inspect the utility transmission and distribution infrastructure is expected to reach \$4.1B annually by 2024. Current civilian / commercial UAV's are vulnerable to many attacks, out of which GPS spoofing, GPS Jamming, Video Replay Attack, DeAuth Attack being a few. There's been a lot of research around detection / prevention of most of them. But, attacks related to video system are less researched. If UAV's are cyber attacked, there is a chance that Video Replay Attack is performed. There are no ways to detect it. In this project, I used Solar shadows of poles to estimate location and time.

### Related Work:

Junejo, Imran N., and Hassan Foroosh. "GPS coordinates estimation and camera calibration from solar shadows." *Computer Vision and Image Understanding* 114.9 (2010): 991-1003.

The above paper implemented the use of shadows, to geo-locate static cameras. This project aim is to implement it and extend it to UAV footage and also to check reliability and optimality of the implementation.

### Theory:

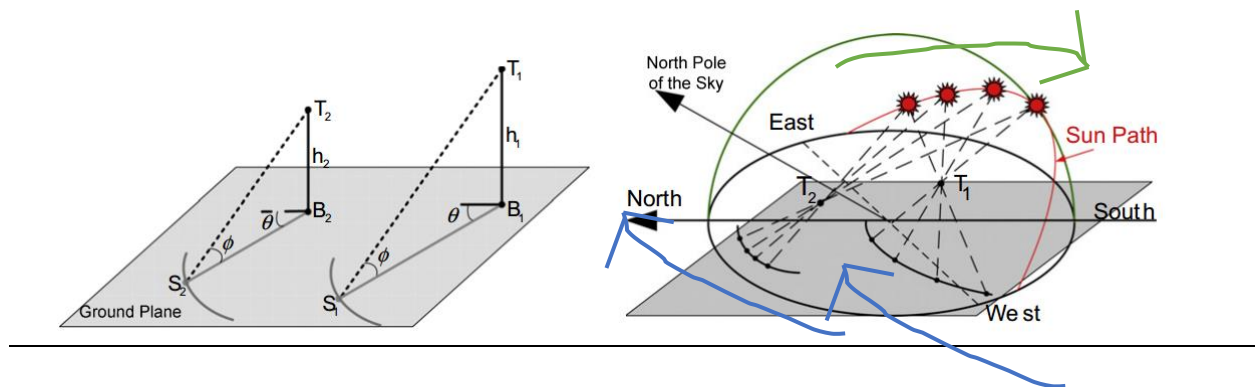


Fig 1: Shadows formed by two poles and pattern formed as sun moves

Relatively sun moves  $1^\circ$  for approximately 4 minutes w.r.t earth.

For a single frame altitude angle of sun can be calculated, but its not enough to determine time (it might be on either side) and loc. Same picture can be taken from different views.

Two frames, that can be used to differentiate the change by a significant amount will be able to determine time, but not location.

We need at least 3 frames to determine location. More frames would reduce the noise effect and make it robust.

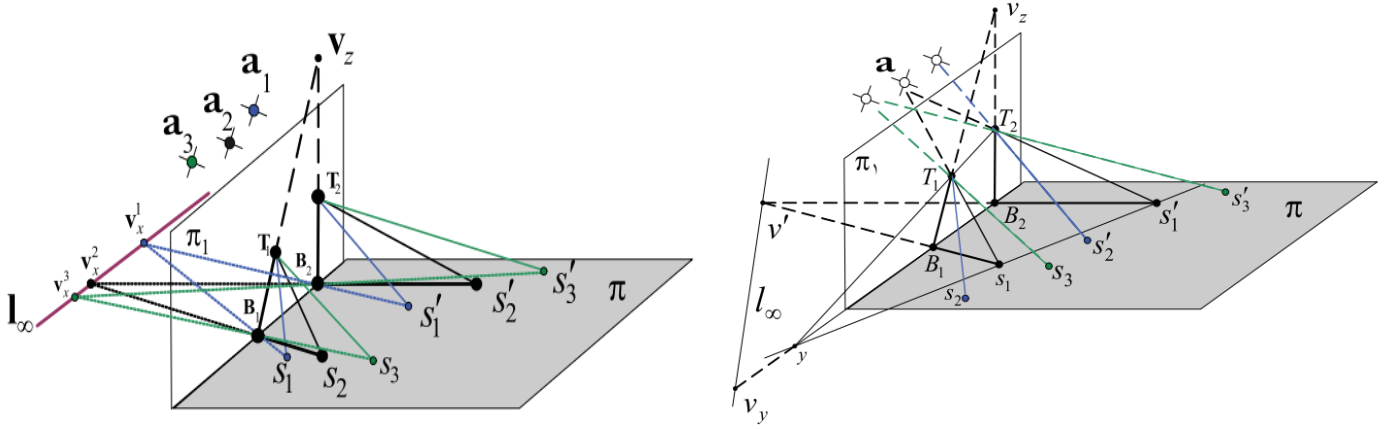


Fig 2: Sun rays, shadow points and  $P_{inf}$  points of three frames

From Fig 2, TB are poles, and BS are shadows cast by it because of sun at 'a'. It is assumed that poles are parallel in real world, which makes the shadows cast by the it will also be parallel in the real world. And also line joining pole-ends and line joining shadow end points will also be parallel in real world.

In the image because of affinity, these become unparallel and intersect at some far-away point known as vanishing point OR point at infinity. Let  $V_z$  be the vanishing point formed by two poles,  $V'$  be the vanishing point formed by shades and  $v$  be the vanishing point formed by line joining pole-ends and line joining shadow end-points. Since we have 3 frames to authenticate, we have 3  $v'$  and 3  $v$  points, with one per each frame.

Since we can control what the camera can see, we can get the internal matrix of the camera being used using checker board. Let the matrix be  $K$ . Then Image of Absolute Conic (IAC) can be estimated using  $K$  as  $\omega = K^{-T} K^{-1}$ .

**Altitude angle ( $\phi$ )**, can be calculated from the  $P_{inf}$  points calculated above using equations,

$$\cos \phi_i = \frac{v_i'^T \omega v_i}{\sqrt{v_i'^T \omega v_i'} \sqrt{v_i^T \omega v_i}} \quad \sin \phi_i = \frac{v_z^T \omega v_i}{\sqrt{v_z^T \omega v_z} \sqrt{v_i^T \omega v_i}}$$

From this we get altitude angle of sun for each frame. Assuming that at sunrise the angle is  $0^\circ$  and at sunset the angle is  $180^\circ$ , we can divide the time between sunset and sunrise into fragments. From the calculated altitude angle and the direction from the next frame, we can calculate the **time** at which the altitude angle is possible. If the shadow length decreases, then its AM, if its increasing its PM.

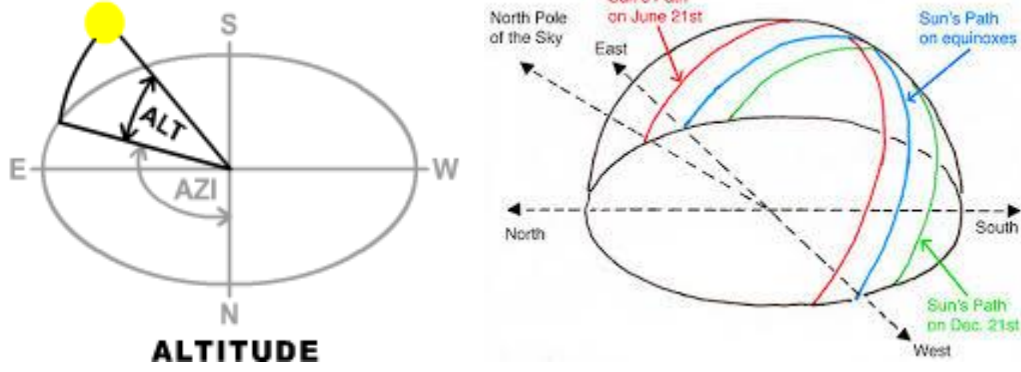


Fig 3: Altitude angle and azimuth angle, along with path of sun in various months.

**Azimuth angle ( $\psi$ )**, can be calculated using the equations

$$\cos \psi_i = \frac{v_i'^T \omega v_x}{\sqrt{v_i'^T \omega v_i'} \sqrt{v_x^T \omega v_x}} \quad \sin \psi_i = \frac{v_i'^T \omega v_y}{\sqrt{v_i'^T \omega v_i'} \sqrt{v_y^T \omega v_y}}$$

**Latitude ( $\lambda$ )**, can be calculated using the equation  $\lambda = \tan^{-1}(\rho_1 \cos \alpha + \rho_2 \sin \alpha)$

Where

$$\alpha = \tan^{-1} \left( \frac{\rho_1 - \rho_3}{\rho_4 - \rho_2} \right)$$

$$\rho_1 = \frac{\cos \phi_2 \cos \psi_2 - \cos \phi_1 \cos \psi_1}{\sin \phi_2 - \sin \phi_1}$$

$$\rho_2 = \frac{\cos \phi_2 \sin \psi_2 - \cos \phi_1 \sin \psi_1}{\sin \phi_2 - \sin \phi_1}$$

$$\rho_3 = \frac{\cos \phi_2 \cos \psi_2 - \cos \phi_3 \cos \psi_3}{\sin \phi_2 - \sin \phi_3}$$

$$\rho_4 = \frac{\cos \phi_2 \cos \psi_2 - \cos \phi_3 \cos \psi_3}{\sin \phi_2 - \sin \phi_3}$$

Assuming  $\delta$  to denote declination angle and  $h$  to denote local hour angle (at the place where the picture was taken), then they along with the values found above, satisfy the conditions

$$\sin h \cos \delta - \cos \phi \sin \theta = 0 \quad \text{and}$$

$$\cos \delta \cos \lambda \cos h + \sin \delta \sin \lambda - \sin \phi = 0$$

Solving those equations using trigonometric identities,

$$\sin h \cos \delta = \cos \phi \sin \theta \text{ -----(1)}$$

$$\cos h \cos \delta = (\sin \phi - \sin \delta \sin \lambda) / \cos \lambda$$

Squaring the above equations and adding, we get

$$\cos^2 \delta = \cos^2 \phi \sin^2 \theta + \frac{(\sin \phi - \sin \delta \sin \lambda)^2}{\cos^2 \lambda}$$

$$1 - \sin^2 \delta = \cos^2 \phi \sin^2 \theta + \frac{(\sin \phi - \sin \delta \sin \lambda)^2}{\cos^2 \lambda}$$

$$\sin^2 \delta - 2 \sin \lambda \sin \delta + \cos^2 \lambda (\cos^2 \phi \sin^2 \theta - 1) + \sin^2 \phi$$

Above equation is a quadratic equation in  $\sin \delta$ , other values are known from above calculations

By calculating the roots of above equation, we can calculate declination angle. From  $\delta$ , we can calculate local hour angle ( $h$ ) from Eqn. (1).

From  $\delta$ , we can calculate day number (N) from the equation, 
$$N = \frac{365}{2\pi} \sin^{-1} \left( \frac{\delta}{\delta_m} \right) - N_0$$

Where  $\delta_m$  is the maximum absolute declination angle of earth =  $23.45^\circ$

And  $N_0 = 284$  : Number of days from first equinox to Jan 1<sup>st</sup>.

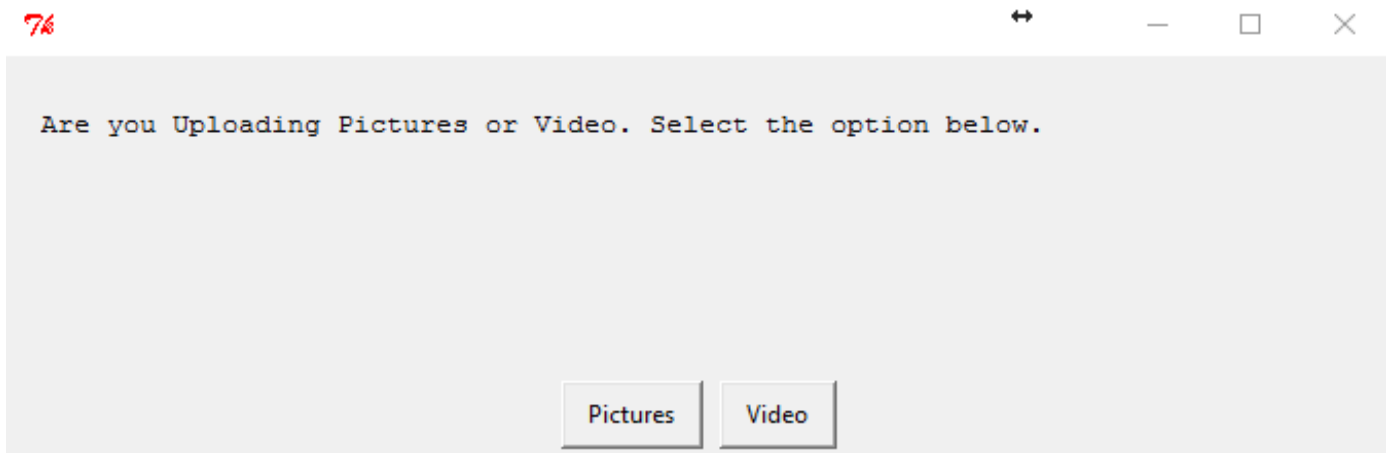
From local hour angle, we can estimate the **longitude ( $\lambda$ )** by considering a pair of known longitude and it's hour angle by the eqn 
$$\gamma = \gamma_0 + (h - h_0)$$

I used Greenwich longitude, which was easy to find the hour angle for online.

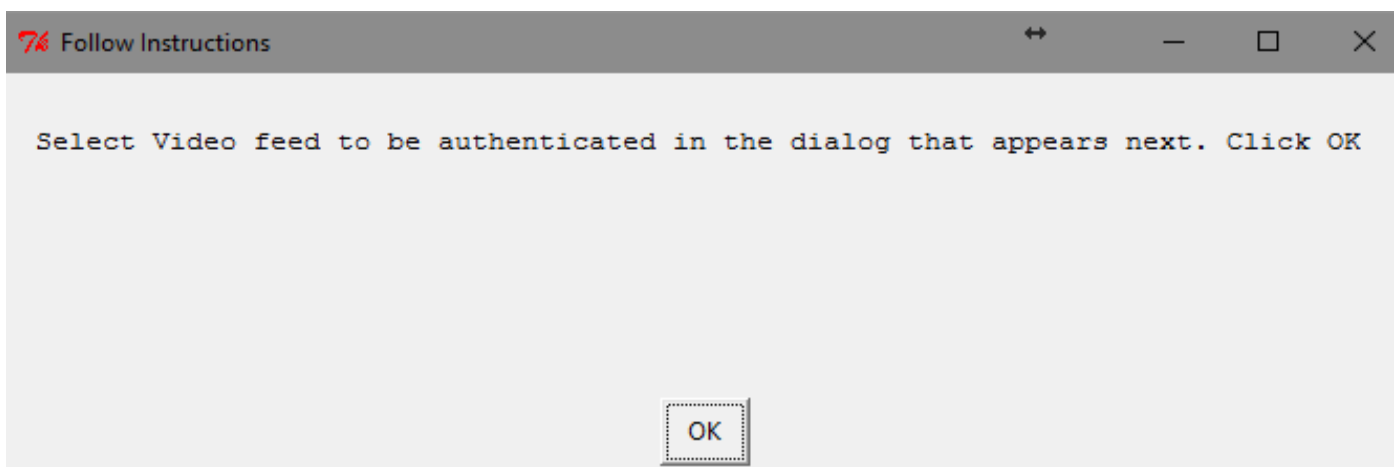
Using the method above, the location, day, time can be estimated approximately.

**Implementation:**

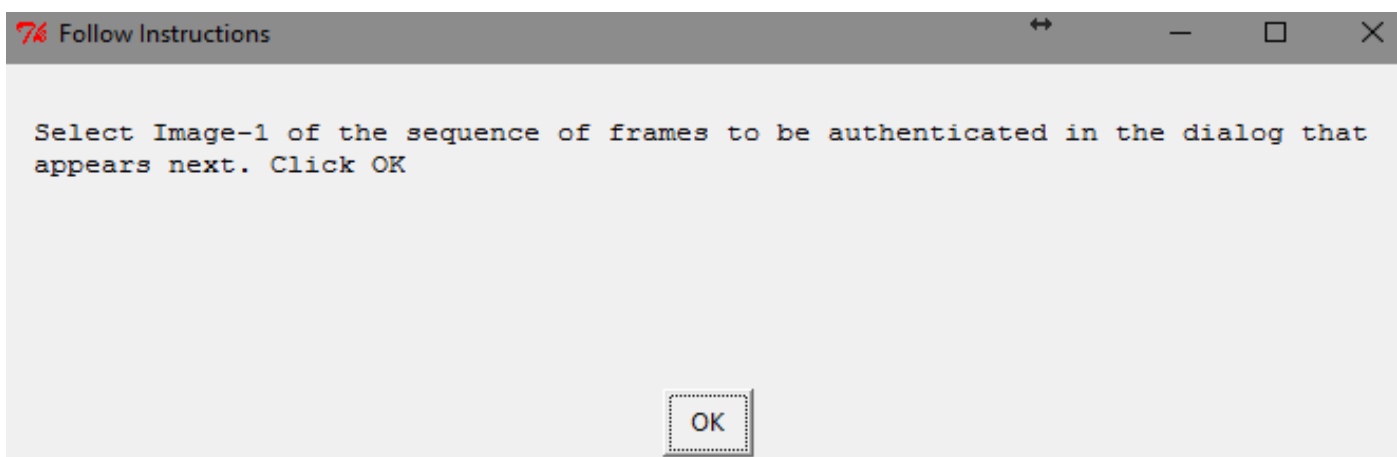
I implemented as a system, which provides step-by-step instruction for the user to follow. On running the main file, a GUI prompt shows up for the user to select video or pictures.



If video option selected, it will be processed and every frame will be saved as an image. From the images user will be asked to select three images.



If pictures option was selected, user will be shown instructions to select three images in a sequence to be authenticated.



After selecting the three images, user will be shown GUI instructions to select points for processing.

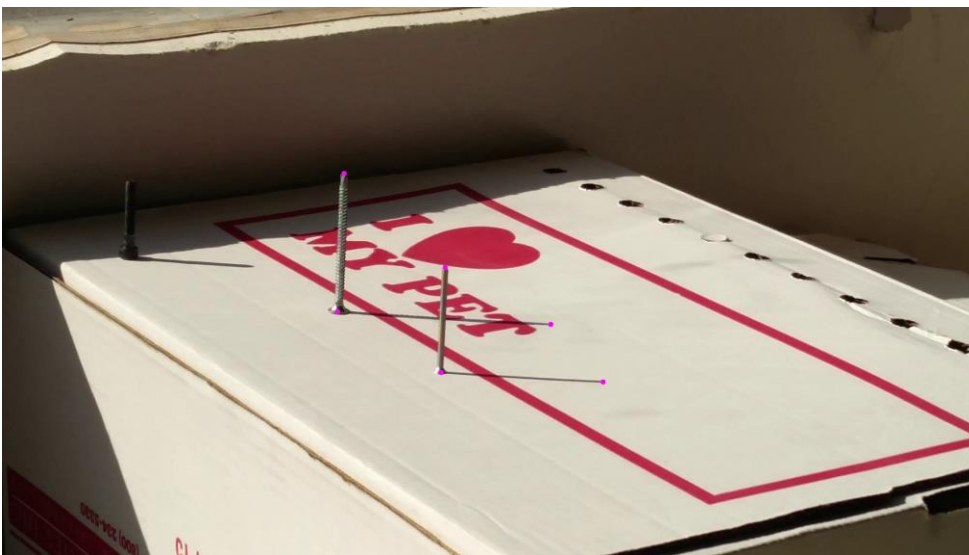
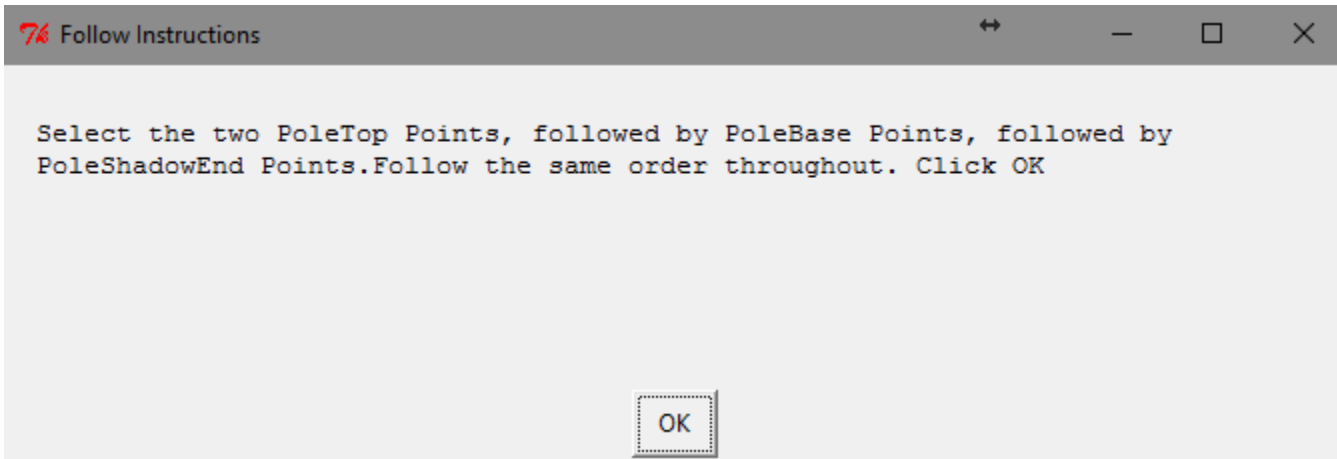
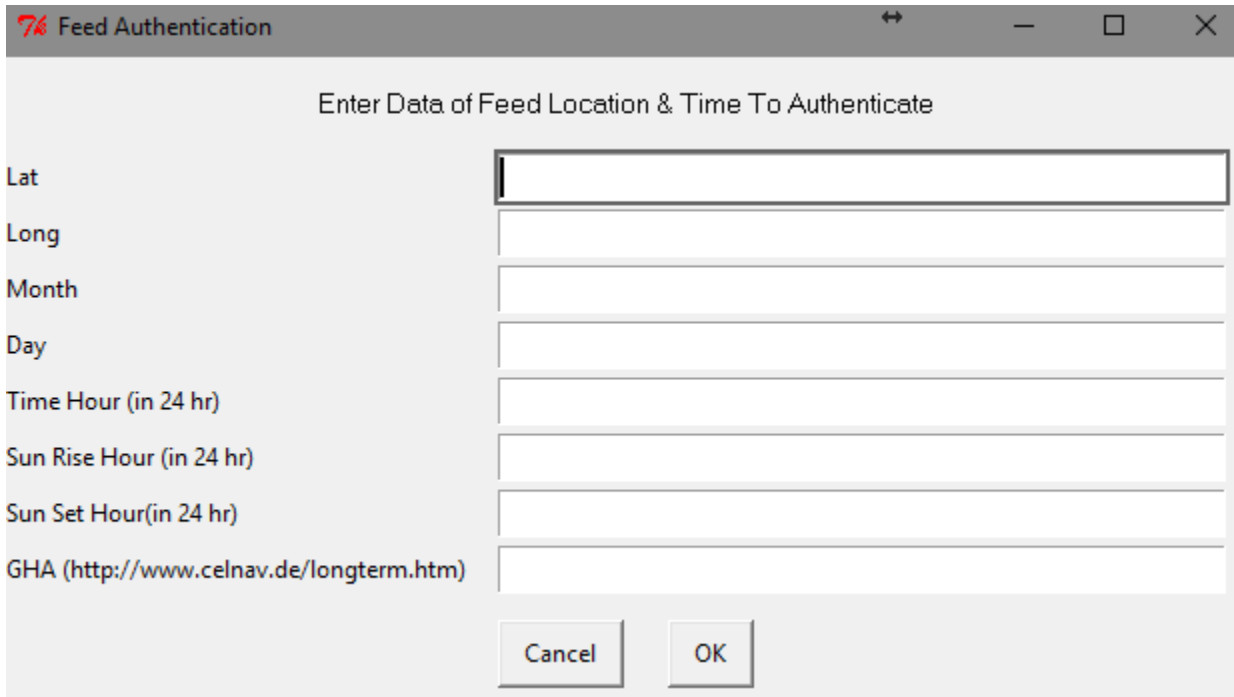


Fig4: Image marked with points (pink)

After selecting points from all the three images, user will be prompted to enter few details required.



7% Feed Authentication

Enter Data of Feed Location & Time To Authenticate

Lat

Long

Month

Day

Time Hour (in 24 hr)

Sun Rise Hour (in 24 hr)

Sun Set Hour(in 24 hr)

GHA (<http://www.celnav.de/longterm.htm>)

Cancel OK

From this information provided, sun rise hour and sun set hour will be used to calculate the time the picture was taken.

### **Results:**

I couldn't get to go out and take the required video from the UAV, so I used my phone to take the video. I took video @ 1 frame per minute using an app.

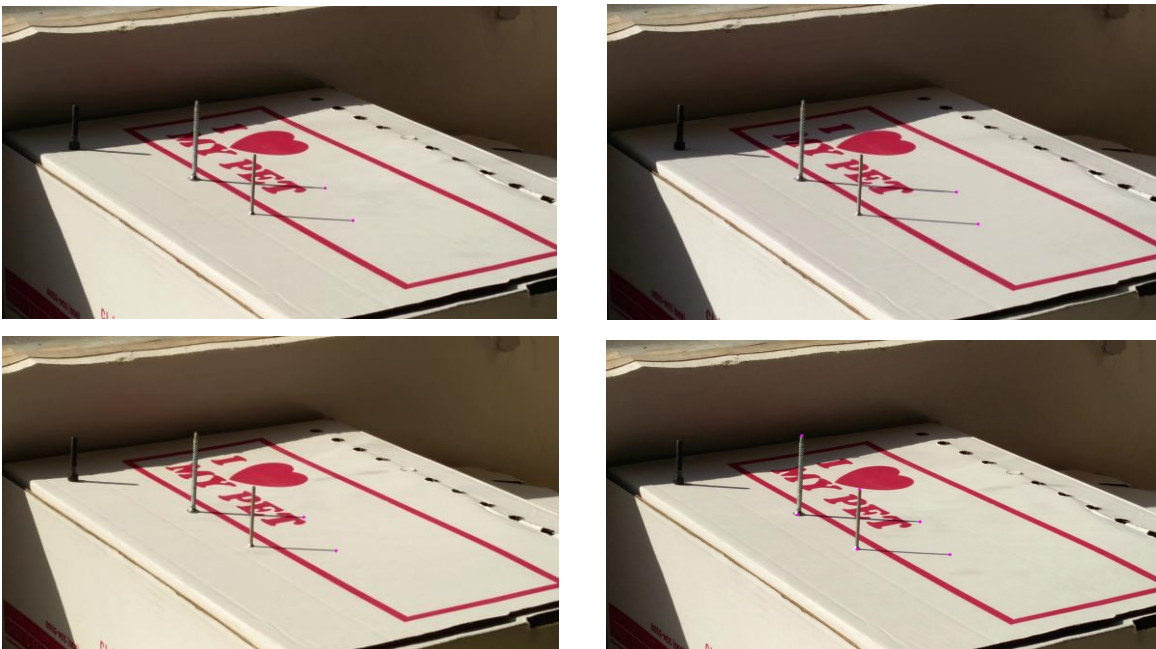


Fig 5: Images used to calculate the data. Marked with the points selected.



The video is processed to frame images and above four images are used in few permutations to calculate the information.

User Input: ['30.618980', '-96.338946', '5', '6', '16.8', '6.5', '20', '75']

34.0467033927	-107.679037278	2017-04-22	17.290372062
36.3358702944	-106.615763538	2017-04-25	17.1387713033
44.7230300356	-129.219302294	2017-04-10	17.332688675

The above data in Green are data provided by user and in Blue are data calculated by the implementation.

In my results the range of error was almost along the lines of the actual paper (except the third – guess I selected wrong points. I put it there to show the sensitivity of the implementation). In the actual paper, the error for margin was  $3^\circ$  for latitude and  $5^\circ$  for longitude and 20 days. They used all possible permutations for images to reduce error by noise.

I modified the program to create another main file that takes single image and outputs the possible times. For this, I was able to use an image from the data that we collected in previous occasions.



Fig 6: Images from the UAV used to calculate Hour

User Input: ['10.5', '7.5', '19.5']

9.4573336891

15.0067650854

Single Image is not sufficient to know time, it can be either



**Conclusion:**

Flight time of an UAV is around 30 min. (DJI Phantom 3 – 25 min). The images that can be analyzed should be at least 4-5 min apart -> 12-15 min of static video from the UAV.

Using most of the flight time to authenticate the feed is not an optimum way of using resources.

Video stabilization cannot be applied, since it might crop the original size of the image.

Homography might not help, since there is a chance that it might rectify shadow too for the assumption I considered.

Care should be taken not to do zoom of the camera, during the process, else Intrinsic matrix would change, unless zoom factor is known. Since the length of pole's in my images is less, it is possible that the effect of noise is more. If the length is long, then the effect might be less.

In conclusion, this implementation is not reliable and optimum way to authenticate the feed, with considered assumptions.

**Future Work:**

In this implementation, I assumed that the poles in the image should be stationary, but I think that it's not mandatory to estimate location and day number. For **time** I might have to find another method. I think it does not need for it to be same poles in the image. I will have to get footage from UAV and check if it works. If it works, then the method can be used, but it still need 15 min to authenticate the feed.

**RUN Instructions:**

Main Files to execute the program:

```
- Authenticate.py - Has option to take a video or images directly. If
video option selected, it will take a video file and divide the whole
video to frames and images are to be selected from it.
Step-by-step instructions are provided to follow. Three Images taken of
the same two parallel poles from same position at three different times
(at least 4 min apart to identify differences) are to be selected.
It asks to user to input lat, long, sunrise, sunset, Greenwich local hour
(link provided).

- GetHour.py - Takes a single image to calculate time at which the picture
was taken. Gives Step-by-step instructions to the user.
It asks user to input sunrise, sunset time on the day the picture was
taken.
To use the method the image needs to have two parallel poles and shadow
casted by them(bcoz of sunlight) in the image.
```

To execute the program, python is needed. To run the file goto the folder in command prompt and run

```
> python <filename>.py  
Step by step instructions in GUI will be shown.
```