Fall 2022, Mentorship of: Prof. Xinming Huang –

Presenting the Project Report–

Target Geo-localization Based on Feature Matching

WPI Directed Research

Presented by:

Sairaam Venkataramani

Teammate:

Shubham Malhotra

12/01/2022

Abstract:

Determination of the location of an automobile where GPS is denied and where GPS does not guarantee high accuracy in determining the precise location (position and orientation) tracking of the vehicle in real-world motivates us to look for alternative means to establish the ground truth of the vehicle in consideration. One of the methods that has gained popularity is vision-based matching of images and hence geo-localization of the target. We find Image-based geo-localization to be a relatively new and challenging problem in computer vision. Geo-localization, simply, is defined as: given a photo of a location, determine where it is in the real-world Google Earth map. We direct the problem of geo-localization to estimate the location of the concerned vehicle in the local image. Different methods have been used to achieve this, popularly cross-view image matching between two sets of given image pairs, an extensive literature survey performed of the existing methods has described in the literature survey discussed in the next section. Our problem utilizes RANSAC based feature-matching between image pairs to perform image transformations and to estimate the homography and its parameters. We then, using the pixel location of the automobile determined using different strategies, determine the precise location of the vehicle in global satellite coordinates from the images. The same technique can be extended to a video input form to perform accurate geo-localization.

Table of Contents

1. Introduction
   1.1 Problem Statement
   1.2 Project Objectives and Summary of Contributions
2. Literature Review
   2.1 Geo-localization methods
   2.2 Determination of Object Orientation from Images
3. Methodology and Implementation
   3.1 Feature Matching
      3.1.1 RANSAC Algorithm
      3.1.2 Matching Feature Points Estimations
      3.1.3 Results of Feature Matching Algorithms
      3.1.4 SuperGlue Pre-trained Network
      3.1.5 Verification of the Homography Estimates
   3.2 Object Detection and Localization
      3.2.1 Object Detection using Transfer Learning approach
      3.2.2 Object Detection and Localization using YOLOV4 algorithm
      3.2.3 Object Tracking using DeepSORT Algorithm (An Overview)
   3.3 Object Detection, Localization and Orientation Determination
      3.3.1 Object Detection, Localization and Orientation Determination using Localizer Library
   3.4 Determination of Global Coordinates from the Pixel Location
      3.4.1 Formulation
      3.4.2 Results
   4. References

1. Introduction

In this section we review the motivation for our work and valuable industry background information. We continue to analyze the current gaps in the domain research specifically focused on target geo-localization. With this information, we formulate a robust problem statement and discuss the objectives and contributions of our project work.

1.1 Problem Statement

Develop a one-shot solution to determination of the ground-truth locations of autonomous vehicles to be utilized for verification in the Simultaneous Localization and Mapping (SLAM) process. This will be utilized to determine vehicle locations in GPS-denied areas as well.

1.2 Project Objectives and Summary of Contributions

The objectives of the project are accomplished by breaking down the problem statement into multiple phases.

1. Feature Matching:
    a. Understanding RANSAC Algorithm
    b. Matching feature points determination
    c. Homography Estimation - Superglue pre-trained network
    d. Verification of determined homography matrix
2. Object Detection and Localization
    a. Transfer learning based on Resnet-50 architecture
    b. Object detection using YOLO V5
    c. Modify YOLO V5 architecture to determine local features specific to an object
3. Object Detection, Localization and Orientation determination
    a. Object detection, localization and orientation determination using Localizer library
4. Computation of geographic location from pixel locations
    a. Arriving at a formulaic expression
    b. Estimation of the location values

Superglue is an algorithm that determines the matching feature points between images and employs attention Graph Neural Networks and Multiplex Graph architecture for this purpose. We shall further investigate about the algorithm in the next section. The estimated feature points are used to orient the drone image in the direction of the satellite image. The

object detection steps return us the position and the orientation of the ground vehicle to be utilized in SLAM.

As we move towards the end of object detection, we experiment with formulae to determine the location of the target vehicle in global coordinates.

2. Literature Review

In this section, we briefly review the different works that have been done in the field of geo-localization and target geo-localization and vision-based localization. We also discuss more literature around the ways to estimate the orientation of objects in an image using computer vision techniques and deep learning techniques.

2.1 Geo-localization Methods

In paper [1], a mismatch in the image pairs is created due to the difference in the images. Here, Sequential Frame Registration is followed by Calibration. Next, the author performs U-NET based segmentation followed by semantic shape matching. In the calibration phase, Scale Invariant Feature Transform has been used and ORB has been used for the image registration. The **U-NET** with skip connections ensures finer segmentation. Dice score is utilized here along with ADAM optimizer. Here the geo-localization errors are reduced die to Semantic Shape Matching: where morphological operations determine boundaries and exact shape information.

An automatic UAV image registration has been performed [2]. SIFT is disadvantageous because of it cannot account for large rotational invariance and repeated patterns. The scheme proposed in this paper is a Dense Feature Matching process, an one-to-many matching strategy. Since ratio test does not determine if a feature point is a correct match, geometric match verification has been utilized and histogram voting ensures correct matching. This achieves robust and accurate geo-registration and benefits from large searching area with geometric constraints. The main limitation of this method is that it can work well only with slightly tilted images and lesser depths.

In paper [3], automatic georeferencing of images taken by UAVs are performed. **Here the requirement of ground control points is done away with**. Here, the camera position and orientation are not restricted. The performance evaluation with flight data concludes that we need to choose a correct navigation system which consider the gimbal errors that result from the attitude errors of the UAV.

The next paper on vision-based absolute localization [4] begins to align with our target objective as **it assays to estimate the location of a UAV using aerial images from a camera.** Here, mutual information is used over the sum of squared distances. The drifting of the UAVs due to environmental disturbances and control parameter errors is a disadvantage. This helps us to estimate the absolute position of the vehicle from 2D motions.

Extraction of road segments by tracking cars and searching for matches in the corresponding map by geometric hashing is discussed in paper [5]. This requires the input images to be ortho-rectified aerial images. It detects parts of road network only. The main disadvantages are the presence of road traffic and flat ground assumption.

Similar to the strategy used in the paper [4], this paper performs visual localization for aerial-satellite image matching framework. Here [6], however, a contrastive learning strategy has been used. It is a **CNN based Siamese Neural Network architecture** where the inputs ate the satellite images and the aerial images. The output is the global coordinate of the center of the system. The distance between the positive sample and the anchor sample are minimized and between the negative sample and anchor are maximized. The network used for image matching and retrieval are a fully connected layer with ReLU. It uses Stochastic Gradient Descent based optimizer on PyTorch. The main advantage of representation learning is that it enables learning low-level features for image similarity measurement and image retrieval and predicting the center coordinates.

A practical cross-view image matching method between UAV and satellite for UAV-based geo-localization [7] has been studied. It requires labelled data and to reduce the difference between features, view difference between aerial images and UAV images is ignored. **ResNet-50** has been used and the activation layer performs non-linear transformation of output to ensure the model can learn and characterize complex features. The residual layer has been used to solve the network degradation problem. Using cross entropy, the matching accuracies of a) different feature sizes, b) multiple queries and c) real UAV images are calculated.

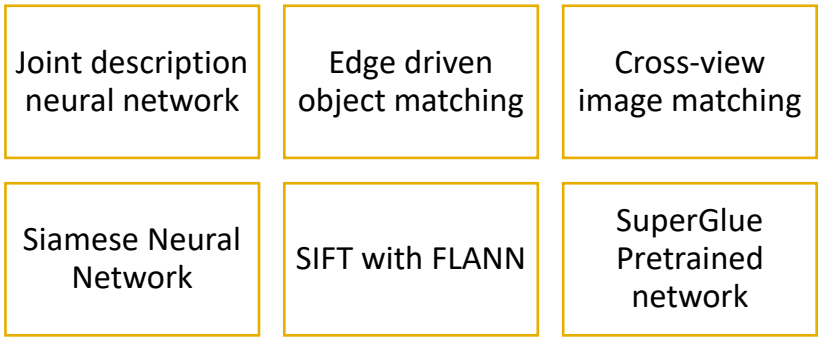
Hand-crafted features perform poor since UAV and SAR have large appearance differences. This is the motivation of the paper [8]. The deep learning-based extraction uses semantic information in the image to extract the more in line with human perspective. **This performs creation of edge-maps by VGG-19** pretrained on ImageNet to get the high-level representation of the image. The matching quality of the source and the target image are determined.

The first work [9] that performs target geo-localization builds on the disadvantages of SSD and NNC that they require large computations and computing time. This is **a quality aware method based on scale adaptive deep convolutional features**. This is a template matching method and VGG-16 has been used to extract high-dimensional features for image matching.

One of the candidate methods that was will be well-suited to our project was the **multi-view image matching of optical satellite and UAV based on a Joint Description Neural Network** [10]. This performs a dense multi-view feature extraction where CNN extracts high-dimensional feature maps. It comprises of hard descriptors and soft descriptors. The hard

descriptors estimate the salient features and feature maps. The soft descriptors determine the gradient information and feature points. FLANN is used for matching feature points and RANSAC is used to screen out false matches. It utilizes triplet margin ranking loss function. The images should have pixel level correspondence and radiometric and geometric differences to learn pixel level similarity. It has a limitation of a very long matching time.

The methods than can be adopted are shortlisted below:



2.2 Determination of object orientation from images

A novel technique called photometric stereo has been discussed in paper [11]. The concept of photometric stereo is to vary the direction of incident illumination between successive images while holding the viewing direction constant. This shows that it provides sufficient information to determine surface orientation at each image point. The technique is photometric because it uses the radiance values recorded at a single point image location in successive views rather than the relative positions of displaced features.

The next paper [12] works on **determining object orientation from a single image using multiple information sources.** This paper gives a detailed analysis of the three techniques which use assumptions about the real world to determine the orientation of objects from a single visual image. Three techniques are proposed for estimating object orientation. Histogram Template Matching employs a nearest-neighbor classifier using the normalized correlation function as a distance measure between the histogram of the input image and a set of training histograms. Binary Connectivity Analysis analyzes the connectivity of an object's silhouette and uses the resulting image features to determine orientation. Ellipse Fitting uses the parameters of an ellipse in the image to specify the orientation of the corresponding circular object surface. Location of the image ellipse is accomplished by exploiting knowledge about object boundaries and image intensity gradients. The combined techniques for estimating the orientation were tested on 138 images of transistors. The limitations of this research are: a) The orientation determining methods used, however, were able to employ some simplifying assumptions that

exploited the natural constraints inherent in the problem domain, b) Histogram template matching is limited almost exclusively to scenes which are highly structured.

Using gaussian image to find the orientation of objects has been discussed in paper [13]. Current three-dimensional vision algorithms can generate depth maps or vector maps from images, but few algorithms extract high-level information from these depth maps. This paper identifies one algorithm that determines an object's orientation by matching object models to depth map data. The object models are constructed by mapping surface orientation data onto spheres. This process is based on a mathematical theorem that can be applied only to convex objects, but some extensions for nonconvex objects are presented. The paper shows that a global approach can be used successfully in cases where objects do not touch one another. Another important result illustrates the size of the space of rotations. It shows that even when 6,000 rotations are almost uniformly distributed for matching, errors of 17 degrees are still possible.

The fourth strategy presented [14] more closely aligns with the implementation possible in real-time **using computer-vision techniques**. In model-based vision, this accurate matching of 3D anatomical surfaces with sensory data such as 2D X-ray projections can be formulated as the estimation of the spatial pose (position and orientation) of a 3D smooth object from 2D video images. The authors present a new method for determining the rigid body transformation that describes this match. To correctly deal with projection lines that penetrate the surface, the authors consider the minimum signed distance to the surface along each line. The combination of the authors' problem formulation in 3D, their computation of line to surface distances with the octree-spline distance map, and their simple minimization technique based on the Levenberg-Marquardt algorithm results in a method that solves the 3D/2D matching problem for arbitrary smooth shapes accurately and quickly.

An advanced topic has been dealt in this paper [15], here coarse geometric properties of a scene are estimated by learning appearance-based models of geometric classes, even in cluttered scenes. They provide a **multiple-hypothesis framework** for robustly estimating the scene structure from a single image and obtaining confidences for each geometric label. They provide a thorough quantitative evaluation of our algorithm on a set of outdoor images and demonstrate its usefulness in two applications: object detection and automatic single-view reconstruction.

The methods that can be adopted are shortlisted as:

Contour extraction	Localizer library-based strategy

3. Methodology and Implementation

3.1 Feature Matching

3.1.1 RANSAC Algorithm:

RANSAC stands for Random Sample Consensus. In our algorithm, we detect the feature points that correspond between a pair of images. Here, RANSAC is well suited for fitting models when a dataset contains a high number of outlying points. The RANSAC method is popularly used in various use cases: curve fitting, classification, state estimation, number of computer vision tasks.

The following is the pseudo code of RANSAC algorithm:

1. Initialize solution
2. Loop until termination criteria are filled:

- Sample a small number of points from the original dataset
- Fit model with sample dataset
- Based on the fit, divide points in the original dataset as inliers and outliers
- Check the validity of solution; skip to next iteration if not valid (optional)
- Calculate score for solution
- If the score is best so far, update the final inliers
  Calculate final model with all the final inliers included (optional)
  return final inliers.

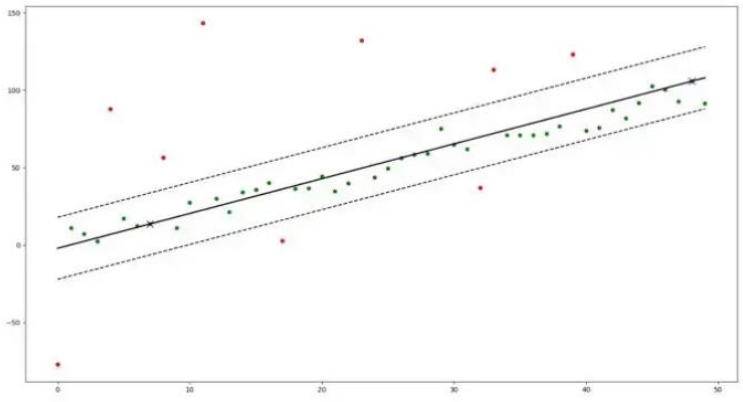
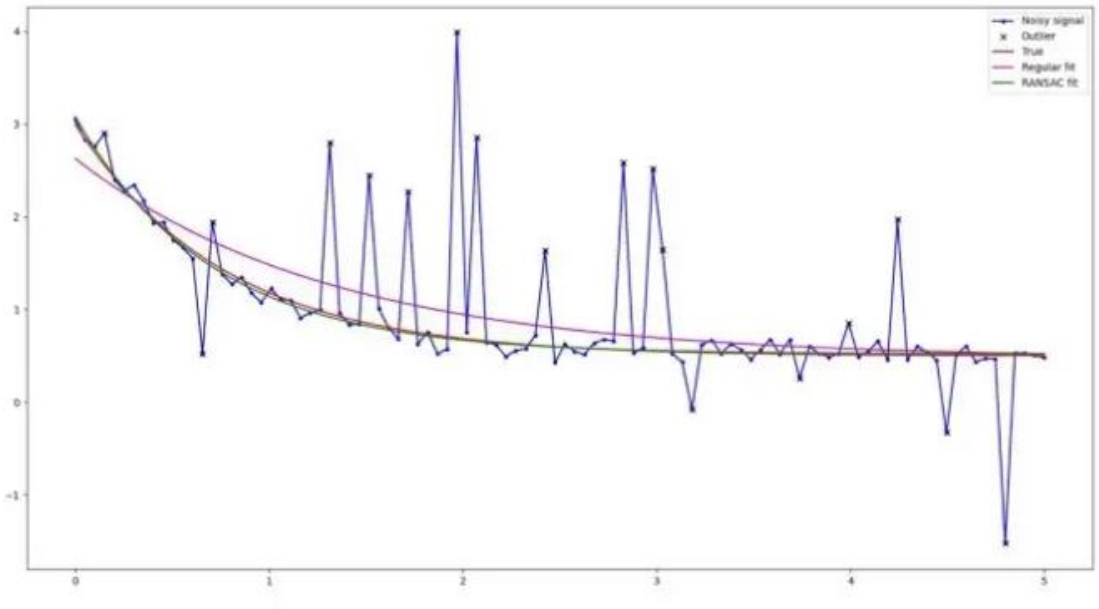


Figure 1. RANSAC algorithm illustrated with a simple linear model. We take sample (blue), fit model with this sample (black), divide all the points into inliers (green) and outliers (red), and calculate a score for this fit. We repeat this until some criteria are filled, and then pick the best solution, based on the recorded scores.

A curve fitting example where we fit a scipy curve for the data points has been demonstrated. The code has been attached in the file.

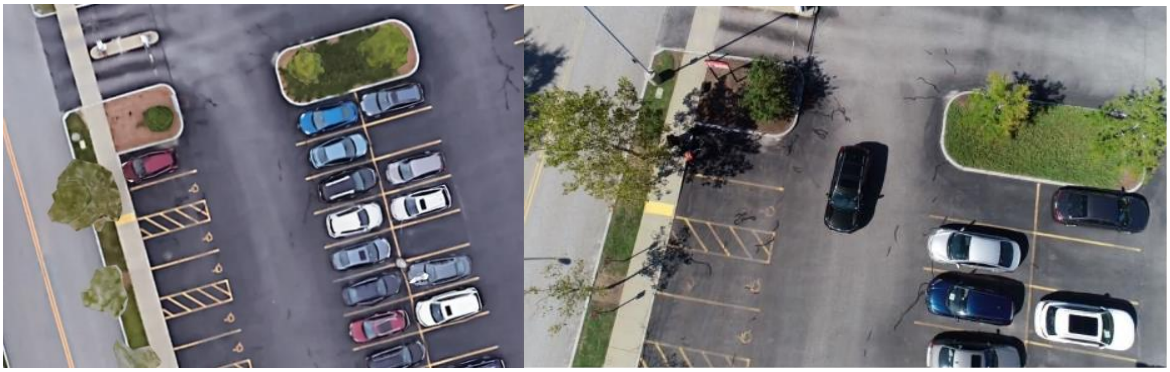


Regular curve fit and curve fit with RANSAC to same noisy signal with outliers. The points that are considered as outliers by the solver are marked with black markers (x).

Thus, we can understand that the advantage of RANSAC is its ability to perform robust estimation of eligible data points that fit the curve. We will be utilizing this in our algorithms with SIFT, SURF, AKAZE, and ORB.

3.1.2 Matching feature points estimation:

The input images that are considered for determining the feature point matches are below:

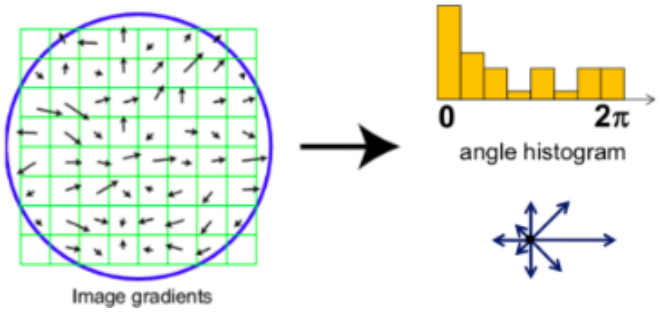


The input images fed into the Feature matching networks are the satellite images taken from Google Earth Pro (left) application and the image captured with the drone (right).

The matches between the feature points are estimated using various algorithms as follows:

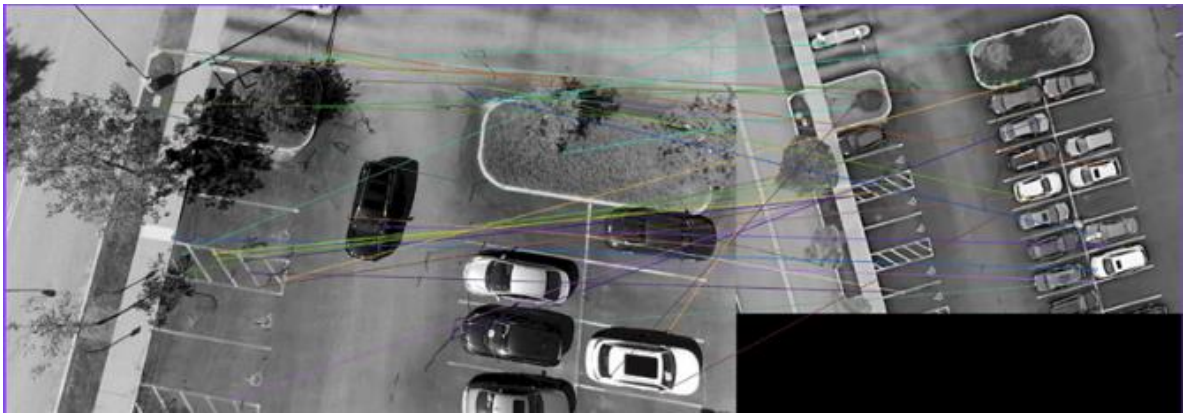
a) <u>Scale Invariant Feature Transform (SIFT):</u>

SIFT is a 4-step process: It consists of feature point detection, feature point localization, orientation assignment and feature descriptor generation. Here, the scale or the diameter is a very important parameter. It relies on Laplacian of Gaussian and Gaussian Kernel to produce images in different scales with different levels of gaussian blurring. Then, the edges are detected as zero-crossing points and as low-peaks after convolving with the Gaussian kernel. Then the orientation assignment is performed and a descriptor from the angle histogram is generated.



The image above shows the orientation histogram and the resulting orientation assignment.

We use the open source SIFT algorithm as it is patented. The images are fed to the SIFT function in OpenCV and the outputs are obtained. The resulting feature matching is displayed below:



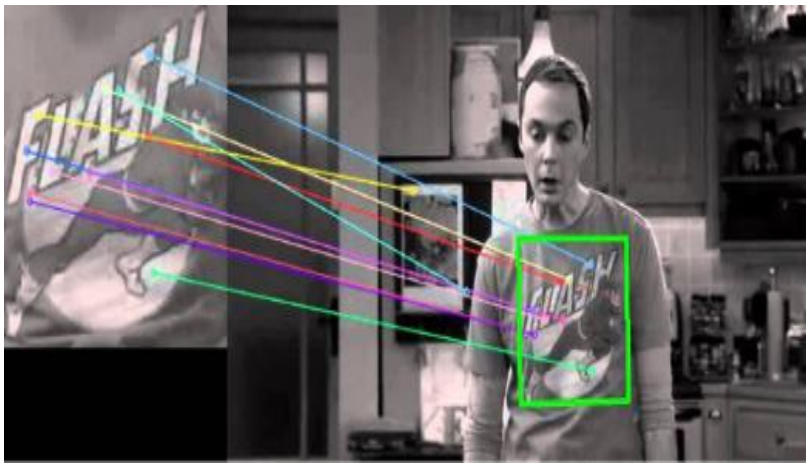
Feature matching between the drone image(left) and the satellite image(right) using SIFT. We can see the number of matches the algorithm has determined.

b) <u>Speeded-Up Robust Features (SURF):</u>

SURF is a 2step process. It consists of feature extraction and description. However, SURF proposes a fast computation of operators using box filters. The sum of intensities over a rectangular region are calculated here. Here Hessians are use but have a disadvantage that Gaussians must be discretized and cropped. This leads to loss of repeatability for Hessians based detectors.

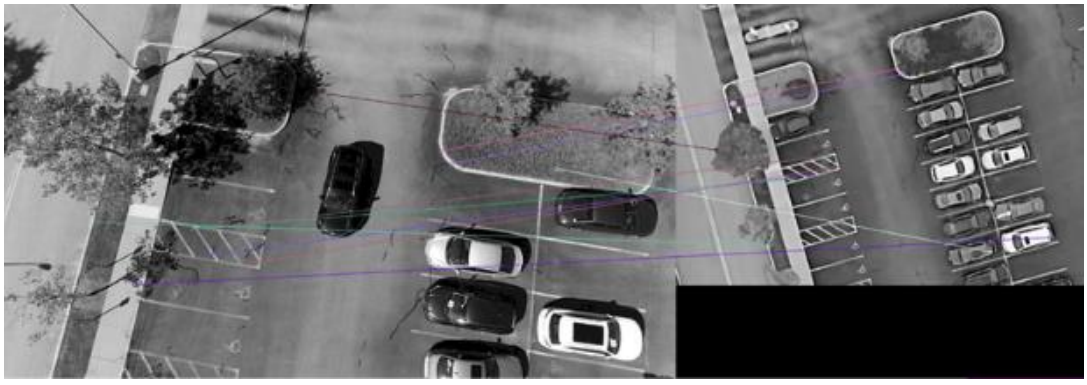
The three steps are:

1. Convolution with Gaussian Kernel
2. Computing the second order derivatives
3. Approximation with box filters



A sample feature matching pattern using SURF

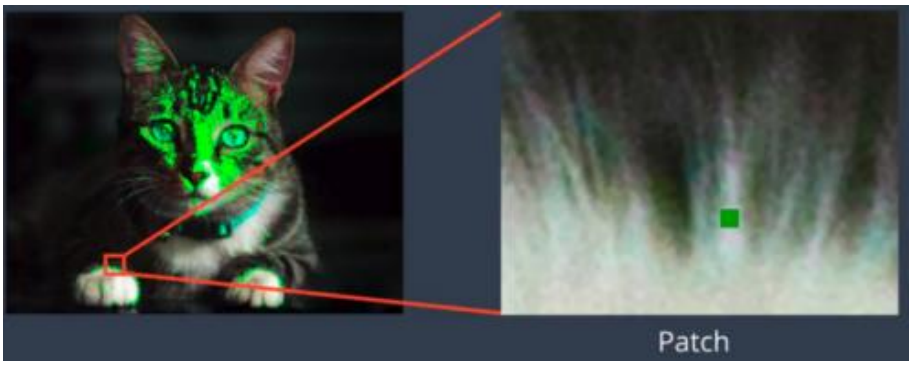
We performed the feature matching on our dataset as well and the results are as follows:



Feature matching between the drone image(left) and the satellite image(right) using SURF. We can see the number of matches the algorithm has determined as lines.

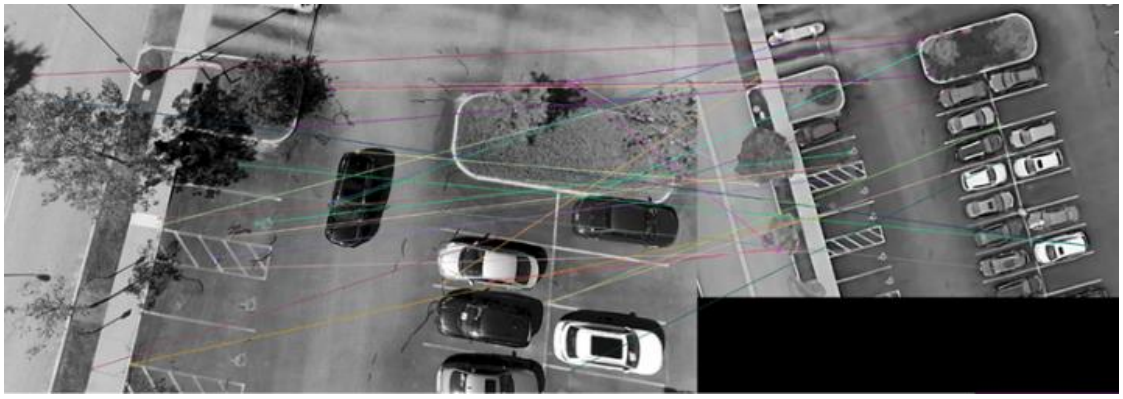
c) Binary Robust and Independent Elementary Features (BRIEF):

BRIEF overcomes the disadvantage in SIFT and SURF of rotational variance. Here, a binary string representation of the feature descriptors are created. This reduces the storage of feature descriptors as 512 bytes for SIFT and 256 bytes for SURF. This eliminates the intermediate feature description process. The binary string is found out using the method mentioned in the research paper for BRIEF [16].



The green point in the patch acts as a centroid of intensities and the is an edge

The performance of BRIEF is evaluated and posted below:



Feature matching between the drone image(left) and the satellite image(right) using BRIEF. We can see the number of matches the algorithm has determined as colored lines between the images.

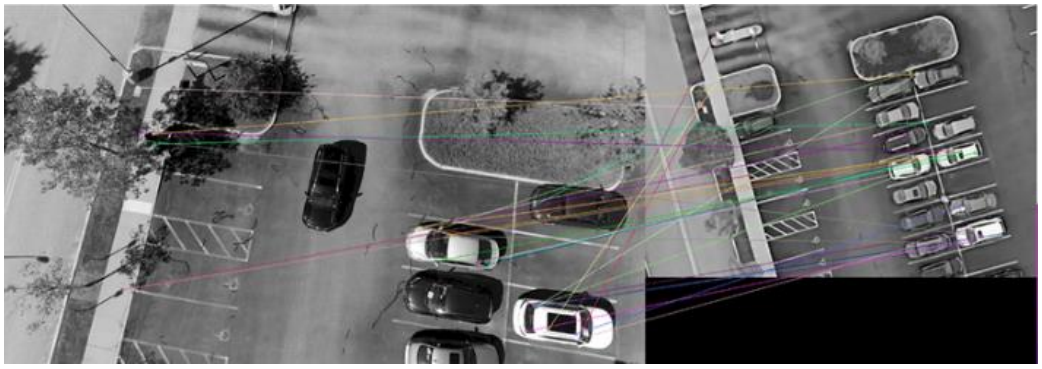
d) Oriented FAST and Rotated BRIEF (ORB):

Here, FAST algorithms is used for feature detection and BRIEF is used for feature descriptor generation. It establishes rotational invariance by calculating moments around a circular region. It improves the BRIEF algorithm that performs poorly with rotation and ORB steers according to the orientation of points.

We discuss ORB in terms of its working and performance here as shown in figures below.

We can observe the key point being determined from two pixels around the key point.

The performance of ORB is as follows:



Feature matching between the drone image(left) and the satellite image(right) using BRIEF. We can see the number of matches the algorithm has determined as colored lines between the images. We observe many more feature points, but the feature points do not have many matches.

3.1.3 Results of the feature matching algorithms:

We observe from the above table that SURF model performs the best, but the accuracy is not sufficient. Let us look at the next table to understand the main reason why all the popular feature detection and matching algorithms under perform with our dataset.

ALGORITHMS	FEATURE POINTS DETECTED	ACCURACY
SIFT	12	25
SURF	10	40
BRIEF	14	21
ORB	11	15

Performance comparisons among different feature matching algorithms. SuperGlue is a new model that will be discussed in the next sub section.

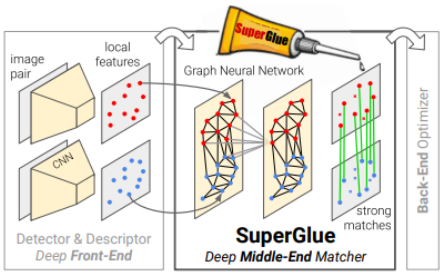
ALGORITHMS	SIFT	SURF	BRIEF	ORB	SuperGlue
Scale Invariance	✓	✓	✓	✓	✓
Rotational invariance	✓	✓	✓	✓	✓
Color differences	✗	✗	✗	✗	✓
Resolution changes	✗	✗	✗	✗	✓
Occlusion	✓	✓	✗	✗	✓

We observe that SIFT performs partially poor with respect to scale invariance as the Laplacian operator decays as the scale of the image increases. BRIEF does not perform well with rotational invariance. All the four algorithms fail due to two main reasons: due to a) color difference existing between the images of different modalities and b) resolution changes that exist between the images. However, an algorithm has been proposed using neural network that matches two sets of local features by jointly finding correspondences and rejecting non-matchable points [17]. This is called the SuperGlue algorithm.

3.1.4 SuperGlue Pre-trained network:

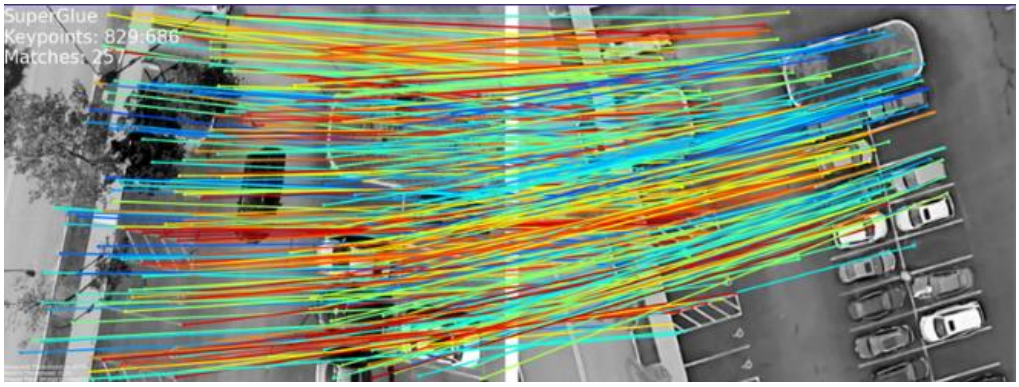
SuperGlue pre-trained network [17] finds matches between two sets of local features by jointly finding correspondences and rejecting non-matchable points. Assignments are estimated by solving a differentiable optimal transport problem whose costs are predicted by a Graph Neural Network.

SuperGlue acts as a strong middle end by being between feature extraction and bundle adjustment. It uses MLP to analyse intra-edges and inter-edges It has an optimal matching layer that assigns partial assignment matrices and AUC and accuracy metrics.



SuperGlue Network Architecture

The performance of SuperGlue pre-trained network is posted below:



Feature matching between the drone image(left) and the satellite image(right) using SuperGlue. We can see the number of matches the algorithm has determined as colored lines between the images. 829 feature points are detected, and 257 matches are obtained.

We observe that the SuperGlue algorithm performs well despite resolution changes and color differences between the images of different modalities. This is due to the use of self- and cross-attention that leverages both spatial relationships of key points and their visual appearance.

First, we add the satellite and the drone images to the scannet_sample_pairs.txt located in the SuperGluePretrained_Master folder. Next, we run the match_pairs.py file which calculates the matches between images and stores the matched pair of images and, the resulting feature points in a .npz file.

scannet_sample_pairs_with_gt.txt - Notepad

File Edit View

```
scene0711_00_frame-001680.jpg scene0711_00_frame-001995.jpg 0 0 1163.45 0. 653.626 0. 1164.79 481.6 0.
scene0713_00_frame-001320.jpg scene0713_00_frame-002025.jpg 0 0 1161.75 0. 658.042 0. 1159.11 486.467 0
scene0721_00_frame-000375.jpg scene0721_00_frame-002745.jpg 0 0 1165.37 0. 650.862 0. 1166.11 488.595 0
scene0722_00_frame-000045.jpg scene0722_00_frame-000735.jpg 0 0 1161.04 0. 648.21 0. 1161.72 485.785 0.
scene0726_00_frame-000135.jpg scene0726_00_frame-000210.jpg 0 0 1165.37 0. 650.862 0. 1166.11 488.595 0
scene0737_00_frame-000930.jpg scene0737_00_frame-001095.jpg 0 0 1163.45 0. 653.626 0. 1164.79 481.6 0.
scene0738_00_frame-000885.jpg scene0738_00_frame-001065.jpg 0 0 1165.72 0. 649.095 0. 1165.74 484.765 0
scene0743_00_frame-000000.jpg scene0743_00_frame-001275.jpg 0 0 1165.72 0. 649.095 0. 1165.74 484.765 0
scene0744_00_frame-000585.jpg scene0744_00_frame-002310.jpg 0 0 1165.72 0. 649.095 0. 1165.74 484.765 0
scene0747_00_frame-000000.jpg scene0747_00_frame-001530.jpg 0 0 1161.75 0. 658.042 0. 1159.11 486.467 0
scene0752_00_frame-000075.jpg scene0752_00_frame-001440.jpg 0 0 1161.75 0. 658.042 0. 1159.11 486.467 0
scene0755_00_frame-000120.jpg scene0755_00_frame-002055.jpg 0 0 1165.72 0. 649.095 0. 1165.74 484.765 0
scene0758_00_frame-000165.jpg scene0758_00_frame-000510.jpg 0 0 1165.72 0. 649.095 0. 1165.74 484.765 0
scene0768_00_frame-001095.jpg scene0768_00_frame-003435.jpg 0 0 1165.48 0. 654.942 0. 1164.54 477.277 0
scene0806_00_frame-000225.jpg scene0806_00_frame-001095.jpg 0 0 1165.37 0. 650.862 0. 1166.11 488.595 0
image1.png image2.png 0 0 1165.37 0. 650.862 0. 1166.11 488.595 0. 0. 1. 1165.37 0. 650.862 0. 1166.11
```

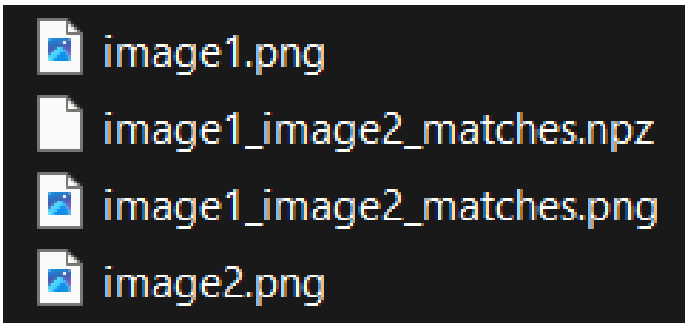
Upon matching, we run Magic Leap's match_pairs.py to determine the key points within the image.

The match_pairs.py file utilizes various feature matching algorithms and connects it to the SuperGlue graph neural network to determine the feature matches.

```
match_pairs.py > ...
338
339         if do_viz:
340             # Visualize the matches.
341             color = cm.jet(mconf)
342             text = [
343                 'SuperGlue',
344                 'Keypoints: {}:{}'.format(len(kpts0), len(kpts1)),
345                 'Matches: {}'.format(len(mkpts0)),
346             ]
347             if rot0 != 0 or rot1 != 0:
348                 text.append('Rotation: {}:{}'.format(rot0, rot1))
349
350             # Display extra parameter info.
351             k_thresh = matching.superpoint.config['keypoint_threshold']
352             m_thresh = matching.superglue.config['match_threshold']
353             small_text = [
354                 'Keypoint Threshold: {:.4f}'.format(k_thresh),
355                 'Match Threshold: {:.2f}'.format(m_thresh),
356                 'Image Pair: {}:{}'.format(stem0, stem1),
357             ]
358
```

Magic Leap's match_pairs.py to determine the key points and the display of the matches between the images.

The determined keypoints are stored as an .npz file. Also, an image showing matches between the image pairs are generated as shown below:



The above folder shows the generated image1_image2_matches.npz file and image1_image2_matches.png files.

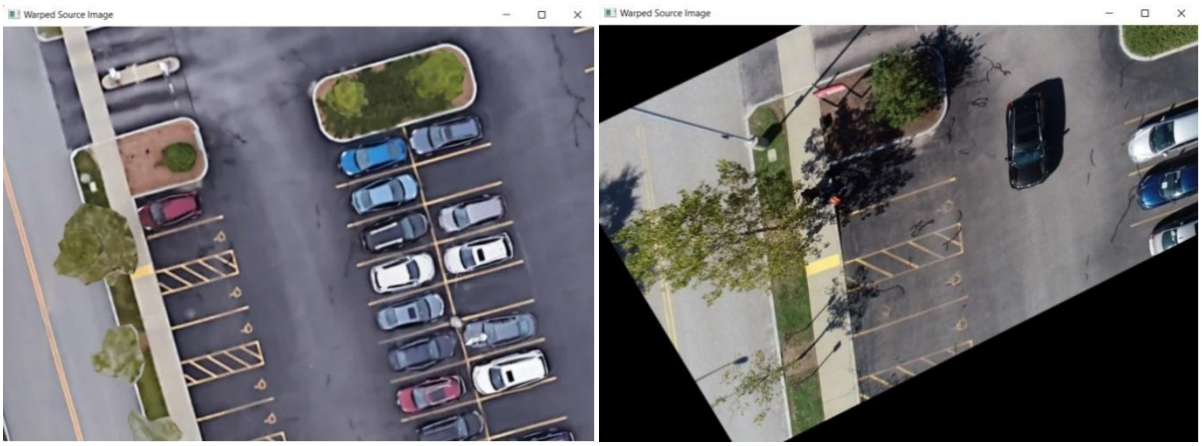
Thus, after determining the key points, we process the key points and pass it through the findHomography function of OpenCV to determine the estimated homography.

We then warp the image according to the estimated homography matrix to obtain the warped image.

```python
import numpy as np
import cv2 as cv2
path = 'scene0711_00_frame-001680_scene0711_00_frame-001995_matches.npz'
npz = np.load(path)
print(npz.files)
print(npz['keypoints1'][92])
# print(len(npz['matches']))
src_pts = []
dst_pts = []
for i in range(len(npz['matches'])):
    if npz['matches'][i] != -1:
        src_pts.append(npz['keypoints0'][i])
        dst_pts.append(npz['keypoints1'][npz['matches'][i]])

src_pts = np.array(src_pts)
dst_pts = np.array(dst_pts)
im_src = cv2.imread('scene0711_00_frame-001680.jpg')
im_dst = cv2.imread('scene0711_00_frame-001995.jpg')

h, status = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC)

print(h)

im_out = cv2.warpPerspective(im_src,h,(im_dst.shape[1],im_dst.shape[0]))

cv2.imshow("Warped Source Image", im_dst)
cv2.waitKey(0)
cv2.imshow("Warped Source Image", im_out)
cv2.waitKey(0)
```

The python file we coded to estimate the homography matrix and to warp the homography matrix.

The resultant drone image that is warped in accordance with the homography matrix is displayed:



Satellite image(left) vs. image warped according to **homography matrix 1.**

The first homography matrix is calculated and the results are as below:

```
Homography matrix 1:
[[ 8.80837772e-01  4.34515667e-01 -6.84038684e+01]
 [-4.23059729e-01  8.00866900e-01  1.59052510e+02]
 [ 4.50779707e-05  4.25884595e-04  1.00000000e+00]]
```

Homography matrix for satellite image vs. drone image

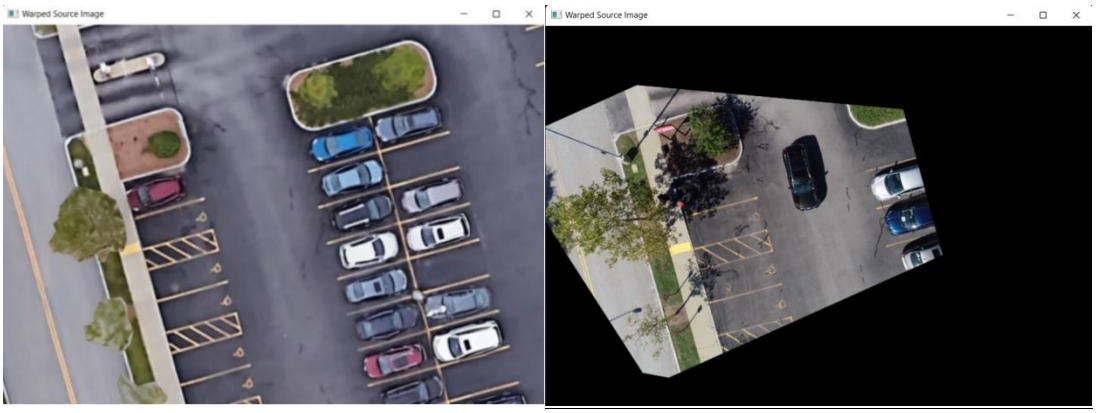
Thus, we have warped the images according to the homography matrix. We see that the two images ***do not match well***.

3.1.5 Verification of the homography estimates

We see that the initial satellite and drone images ***do not match well***.

So, what we do next is that we take the warped image and again insert it into the scannet_sample_pairs.txt and get the matching feature points as "*warped_image_image2.npz*". Now, we calculate the homography matrix for this transformation and warp the image.

This gives us a perfect homography matrix:



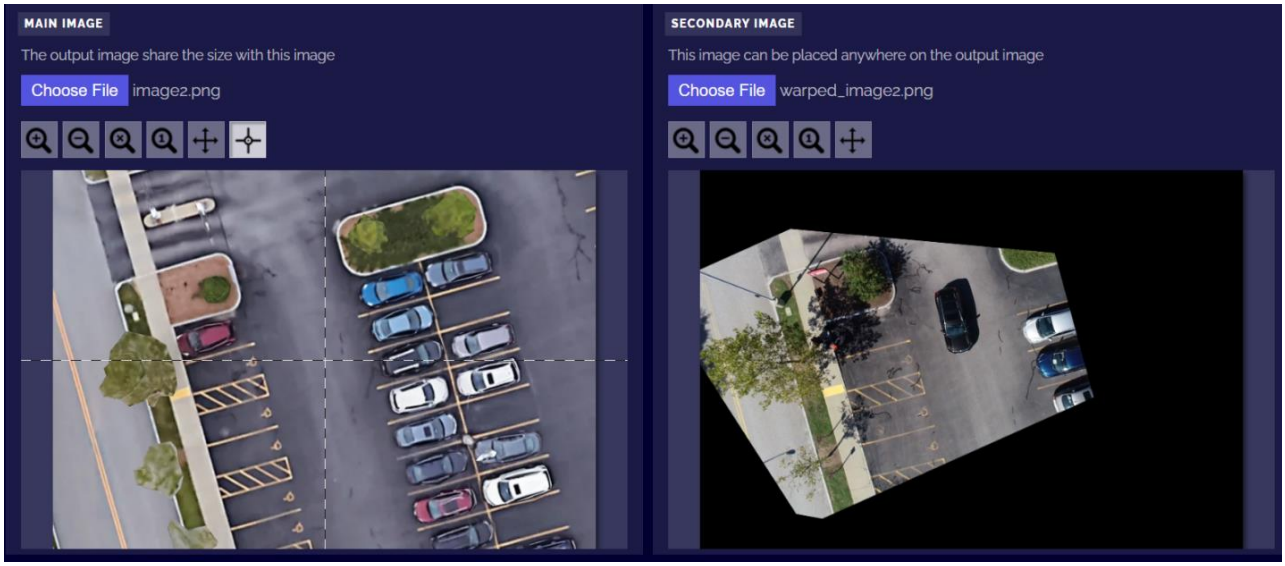
Satellite image(left) vs. image double warped according to ***homography matrix 2***.

```
Homography matrix 2:
[[ 6.28869364e-01  8.92198948e-02 -3.75580794e+01]
 [ 4.69537980e-02  6.52093069e-01  6.89589594e+01]
 [-1.42094297e-04 -1.84454913e-04  1.00000000e+00]]
```

Homography matrix for satellite image vs. warped drone image

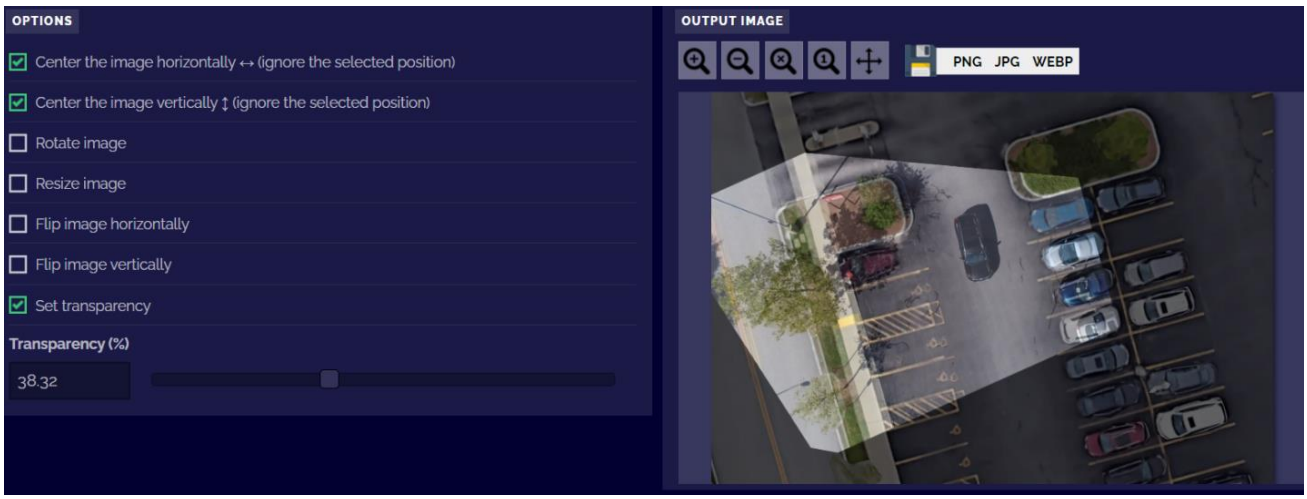
To verify the accuracy of final homography determination, we overlay the two final images using an online image overlaying software:

The online overlaying software at pinetools.com performs the necessary overlaying of the input satellite image and the doubly transformed drone image with desired transparency levels.



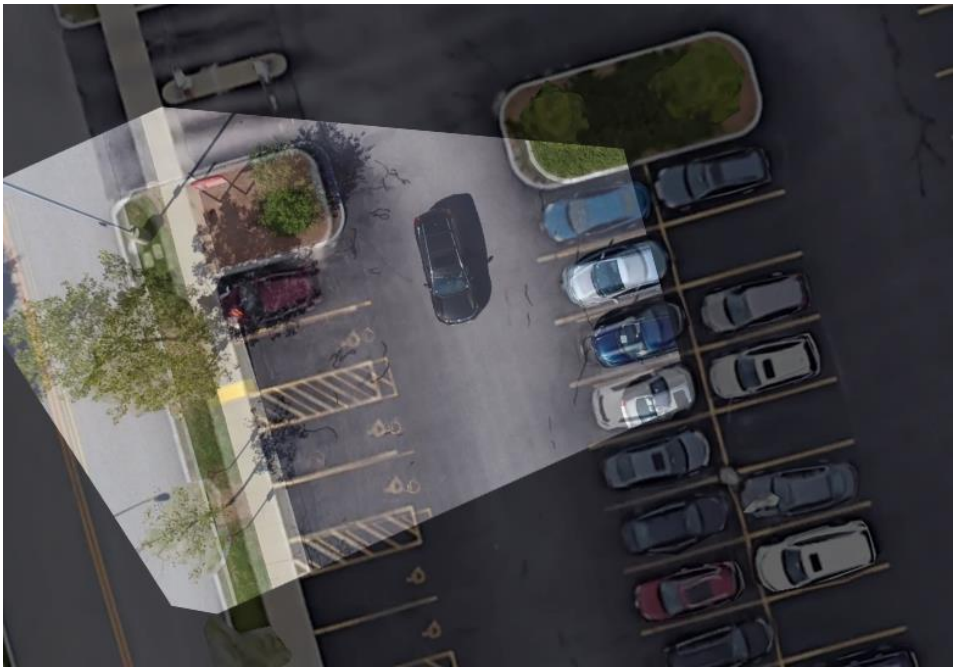
Inputs to the overlaying software: [Overlay images online (pinetools.com)]

The online overlaying software at pinetools.com performs the necessary overlaying of the input satellite image and the doubly transformed drone image with desired transparency levels.



Overlaying software results: [Overlay images online (pinetools.com)]

The results of final overlaying between the two images from the software is shown below.



We observe from the overlayed images that the accuracy is 94% along x-direction and the accuracy along the y-direction is 99% measured using a scale software.

Thus, the final homography matrix is the product of homography matrix 1 and homography matrix 2 in the reverse direction.

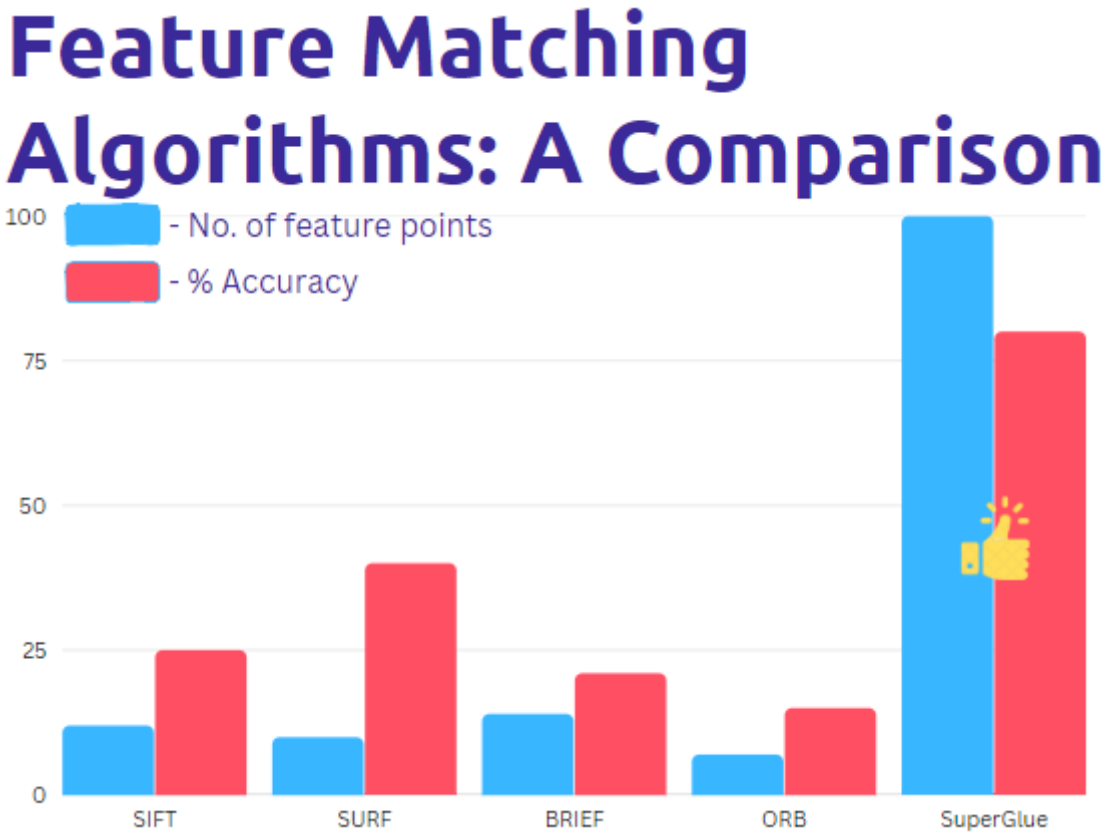
Thus h_final transformation matrix is calculated as below:

$$\begin{bmatrix} 0.51322711 & 0.32793725 & -66.321121 \\ -0.2314235 & 0.5712371 & 169.39264 \\ -0.0000019588 & 0.000215852 & 0.98038398 \end{bmatrix}$$

Final homography matrix found out as the product of h2 and h1.
(i.e.) **h_final = h2.h1**

Thus, we have achieved a high accuracy in matching the images and have successfully verified the match by overlaying as well.

A comparison between the feature matching algorithms has been presented in the form of a bar-chart below:



Thus, we conclude that the SuperGlue algorithm detects enough feature points and the accuracy is very high.

3.2 Object Detection and Localization

We now progress to the next step in determining the location of the vehicle in global coordinates.

3.2.1 Object detection using Transfer Learning approach:

First, we try to perform object detection using COCO dataset sample images. We use an SSD Resnet 101 architecture with enough dense layers to train on images.

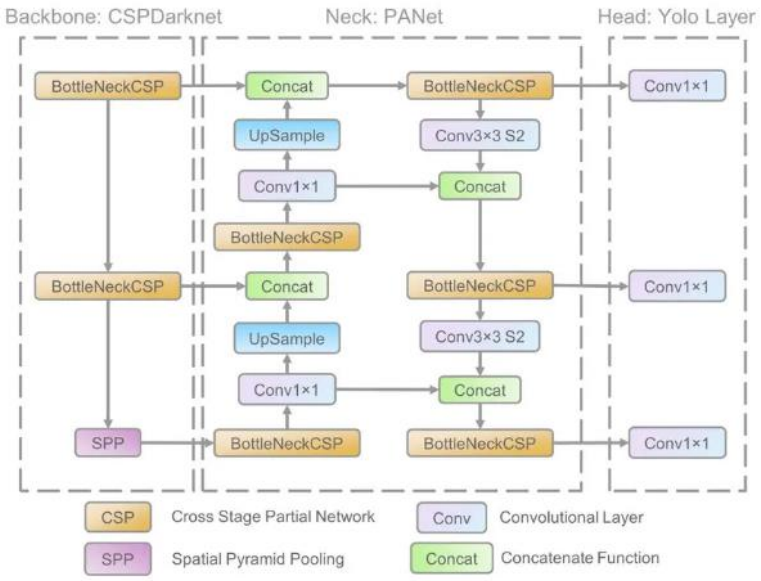
We use **labelimg** to manually label images and the labels are stored in a JSON file format. The final image obtained is attached below:

The green box represents the bounding box and it targets the black Honda car.

We observe that the accuracy is very low and the bounding box over bounds the target vehicle. Thus we need to look for alternate architectures.

3.2.2 Object detection and localization using YOLOV4 algorithm:

YOLO stands for You Look Only Once and it is one of the finest family of object detection models with state-of-the-art performances.



YOLOV4 architecture

The YOLO family of models consists of three main architectural blocks i) Backbone, ii) Neck and iii) Head.

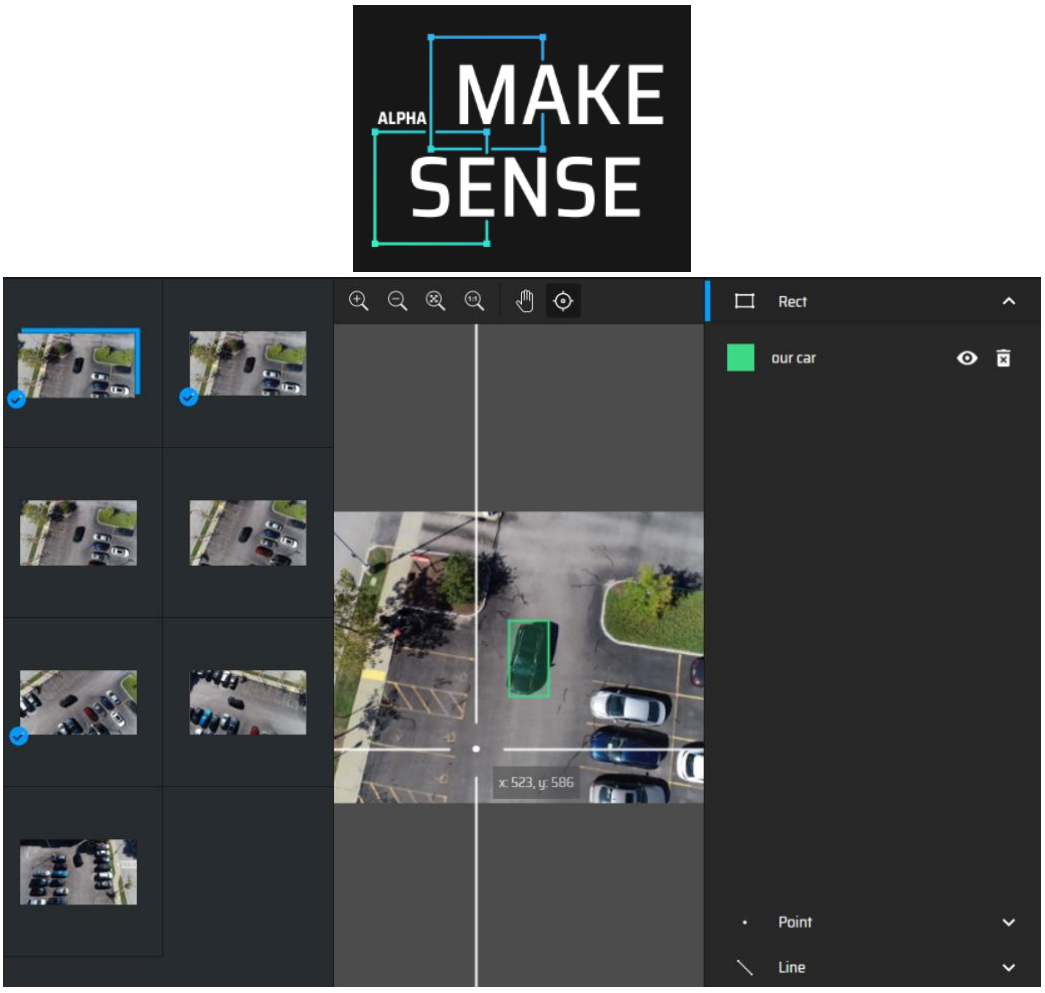
The online overlaying software at pinetools.com performs the necessary overlaying of the input satellite image and the doubly transformed drone image with desired transparency levels.

Apart from this YOLOv5 uses two choices for training –

1. **Activation and Optimization:** YOLOv5 uses leaky ReLU and sigmoid activation, and SGD and ADAM as optimizer options.
2. **Loss Function:** It uses Binary cross-entropy with logits loss.

We setup the YOLOV5 model on the Google Colab notebook. Here are the steps we follow to implement YOLOV5 on out dataset:

1. First, we split the drone images of the car into training and validation dataset.
2. Next, we label the train and validation images using makesense.ai and we save the results in YOLO format.



1. We run the Setup cell of our YOLOV5 model to install the model.

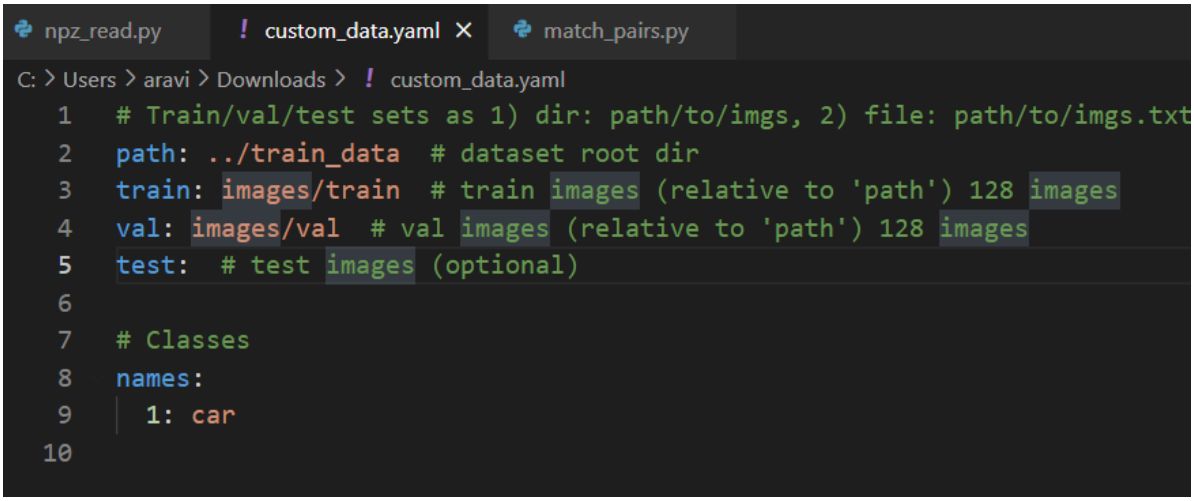
Setup

Clone GitHub repository, install dependencies and check PyTorch and GPU.

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt  # install

import torch
import utils
display = utils.notebook_init()  # checks
```

```
YOLOv5 🚀 v7.0-1-gb32f67f Python-3.7.15 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 22.6/78.2 GB disk)
```

2. Next, we upload our train_data folder into Google Colab.
3. Next, we need to modify the labels folder. We download the coco128.yaml from Yolov5 -> Data. We give the path of our labels folder and our labels. We then upload the folder into the notebook.




```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 3 --data custom_data.yaml --weights yolov5s.pt
```

This is an image of the custom_data.yaml file from where the labels will be loaded for the YOLOV5 model YOLOv5 training with custom data - YouTube

4. Next, we run the training on our custom dataset for 20 epochs. Since the mAP50 is low, we run it for ***600 epochs with a batch size of 8 to achieve an accuracy of 85.3.***

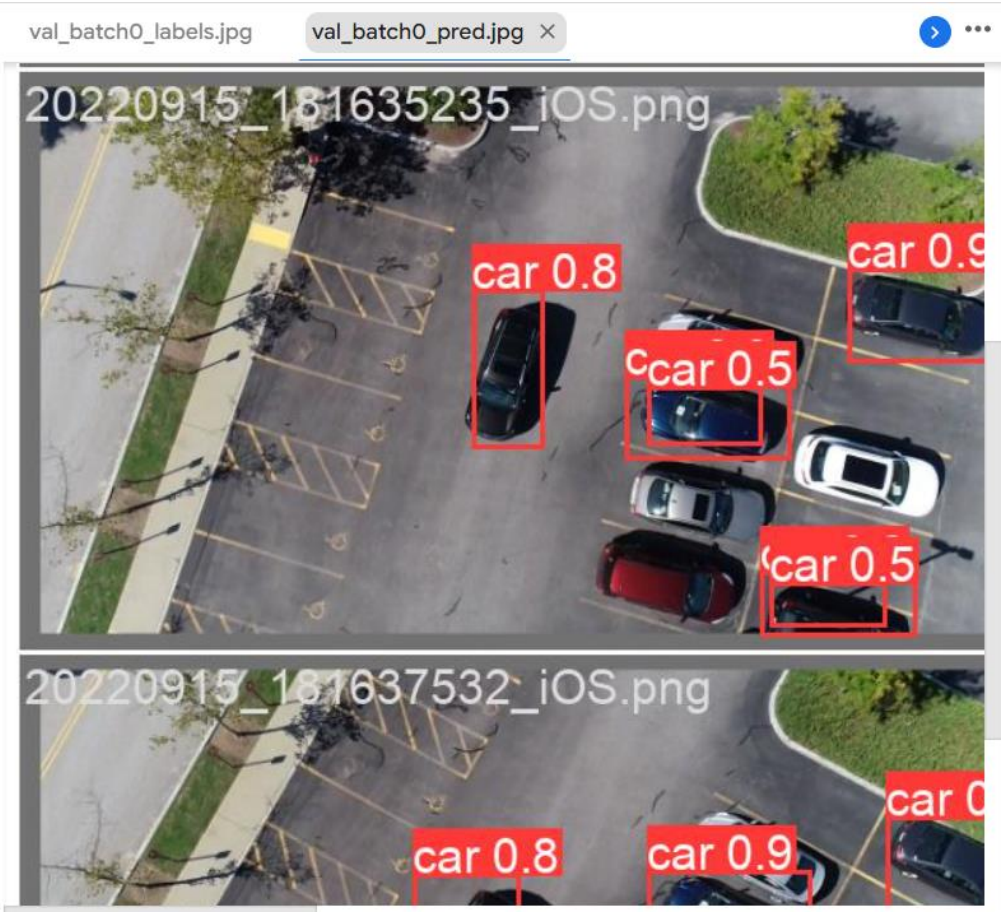
```
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95:
                   all          7         40     0.0777       0.05      0.037    0.00822
Results saved to runs/train/exp
```

20 Epochs, batch-size: 16

```
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all          7         40      0.904      0.708      0.853
Results saved to runs/train/exp3
```

200 Epochs, batch size: 8

5. We check the results of prediction from yolov5 -> runs -> train -> exp3 -> val_batch0_pred.jpg



6. We can see that only the dark-coloured cars are detected. This is a success for us. Next, we examine the results on detecting our car only.
7. After relabeling only our car, we can see the result on our final warped image as follows:

Bounding box determined on our warped image dataset

8. We next extract the bounding box information from YOLOV5 file structure. We use the pixel information to determine the location of the car in global coordinates.

```
results = model(input_images)
labels, cord_thres = results.xyxyn[0][:, -1].numpy(), results.xyxyn[0][:, :-1].numpy(
```

9. Thus, we have trained the YOLOV5 model to detect our car successfully and determined the location of the bounding boxes.

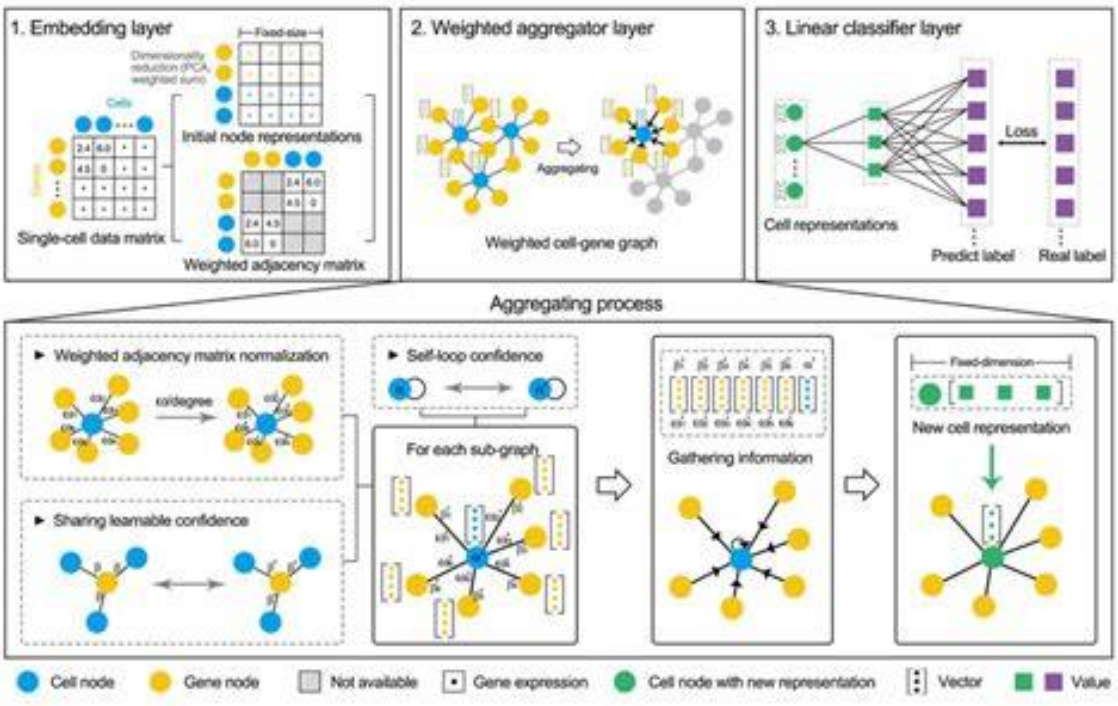
3.2.3 Object tracking using DeepSORT algorithm (an overview)

DeepSORT algorithm is used for multiple-object tracking. Whenever, we perform object tracking, we perform detection first and then combine it with the tracker. The two steps in the DeepSORT algorithm are:

1. Perform object detection using YOLOV4.
2. Multi-object tracking: Kalman filter to process the correlation of frame-by-frame data. Hungarian algorithm is used for correlation measurement. The CNN network is used for training and feature detection.

When we think of multiple object tracking, we think of correlation between data. We have to determine the correlation and Kalman filtering helps us process the correlations frame by

frame. Kalman filtering allows us to model tracking based on the position and velocity of an object and predict where it is likely to be. The Hungarian algorithm estimates the correlation extent. It represents the degree of correlation between the detected objects. The CNN is used to extract deep features of every object present in every frame and it helps overcome the problems of occlusion and ID switching problems.



The weighted GNN algorithm of DeepSORT.

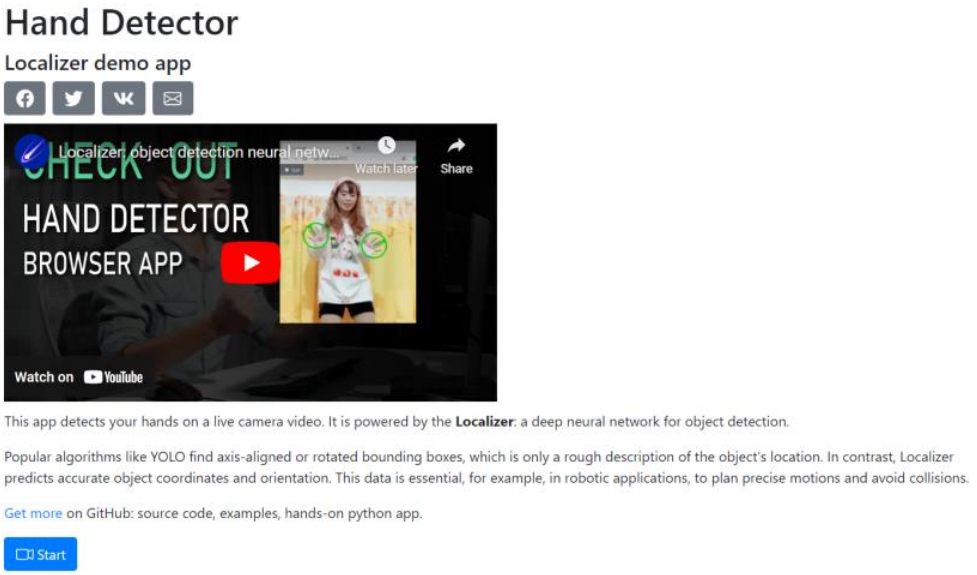
The algorithm can be implemented with YOLOV4 and an example of traffic sign detection can be applied to our problem to detect our car. The GitHub tutorial is linked as follows: AarohiSingla/DeepSORT-algorithm-with-YOLOv4 (github.com). This has been saved for future work.

3.3 Object Detection, Localization and Orientation determination

3.3.1 Object Detection, Localization and Orientation determination using Localizer Library

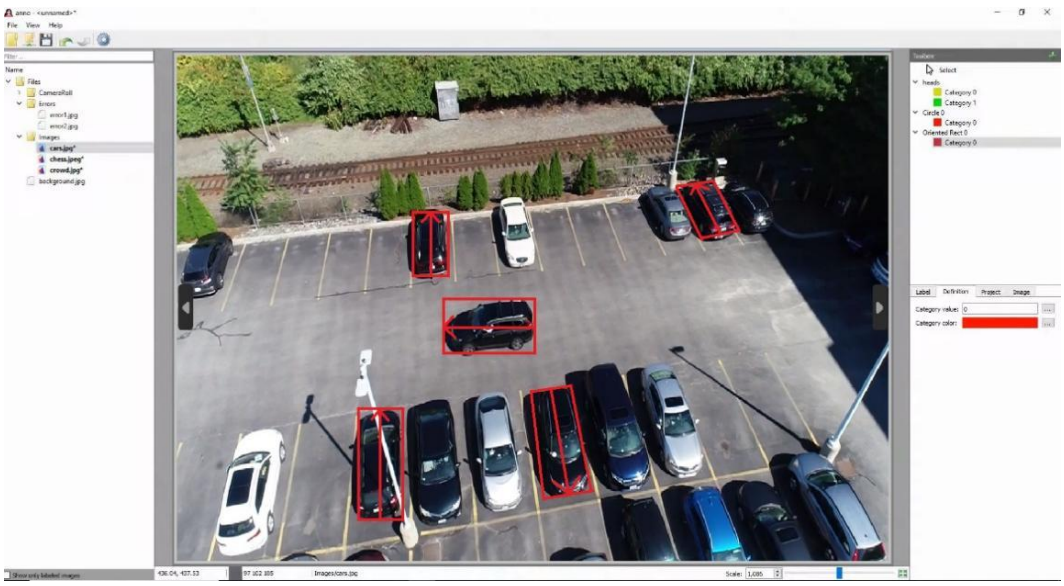
Although we have obtained the location of the bounding boxes from YOLO, it is not possible for us to determine the orientation of the car. From our research, we have decided

We use the Localizer library on Ubuntu.



This is the web app that determines the location and orientation of hand.

We first annotate the images using **Anno software**:



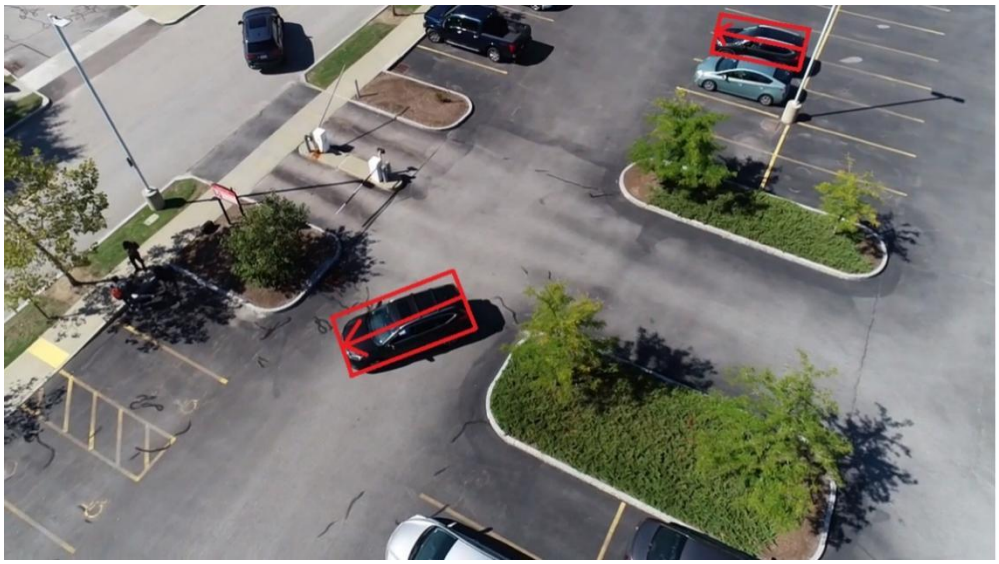
1. We add the image information into a JSON file.

```
[
 {
  "image": "myimage.png",
  "objects": [
   {
    "category": 0,
    "origin": {
     "x": 100,
     "y": 200,
     "angle": 3.14159
    }
   }
  ]
 }
]
```

2. We save the image information to mydataset.anno.

```
{
  "dataset": "path/mydataset.anno"
}
```

3. Next, we train the images and then predict all the images in the folder.
4. This is what the final image looks like:



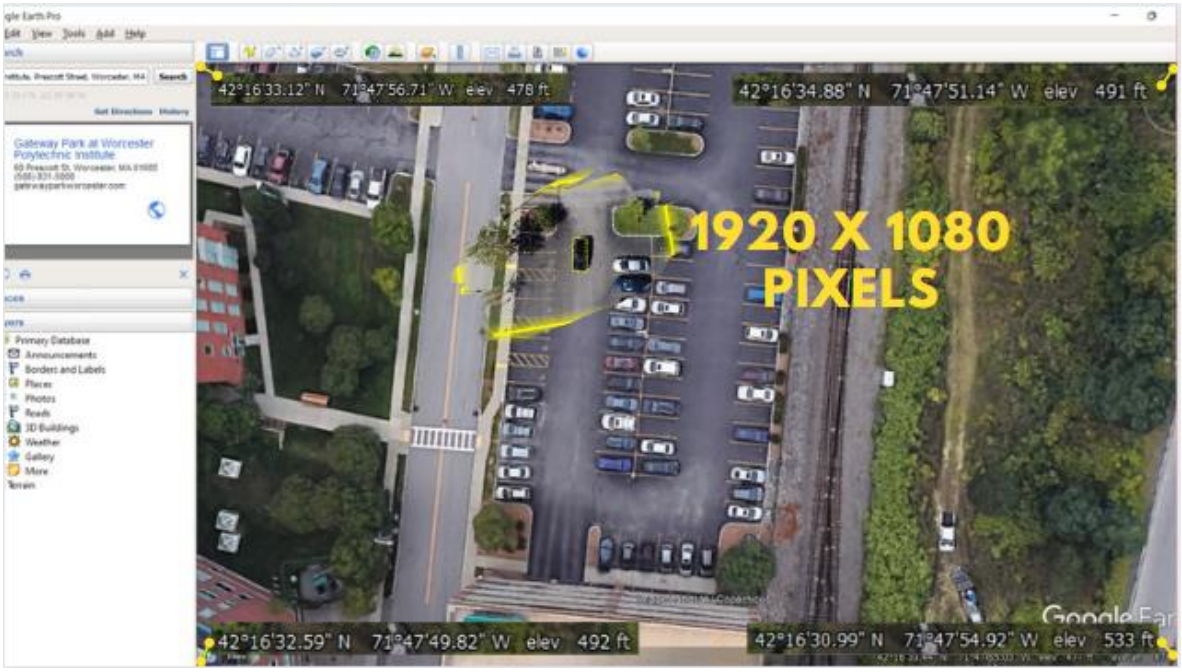
5. Thus, the app returns the rotation angle as 1.2217 radians and also returned the position of the vehicle and we read it from the structure of the output file.

3.4 Determination of global coordinates from the pixel locations

3.4.1 Formulation

X = co_ord1 + (co_ord2 – co_ord1) * x_pixel_pos / x_resolution

Y = co_ord1 + (co_ord3 – co_ord1) * y_pixel_pos / y_resolution

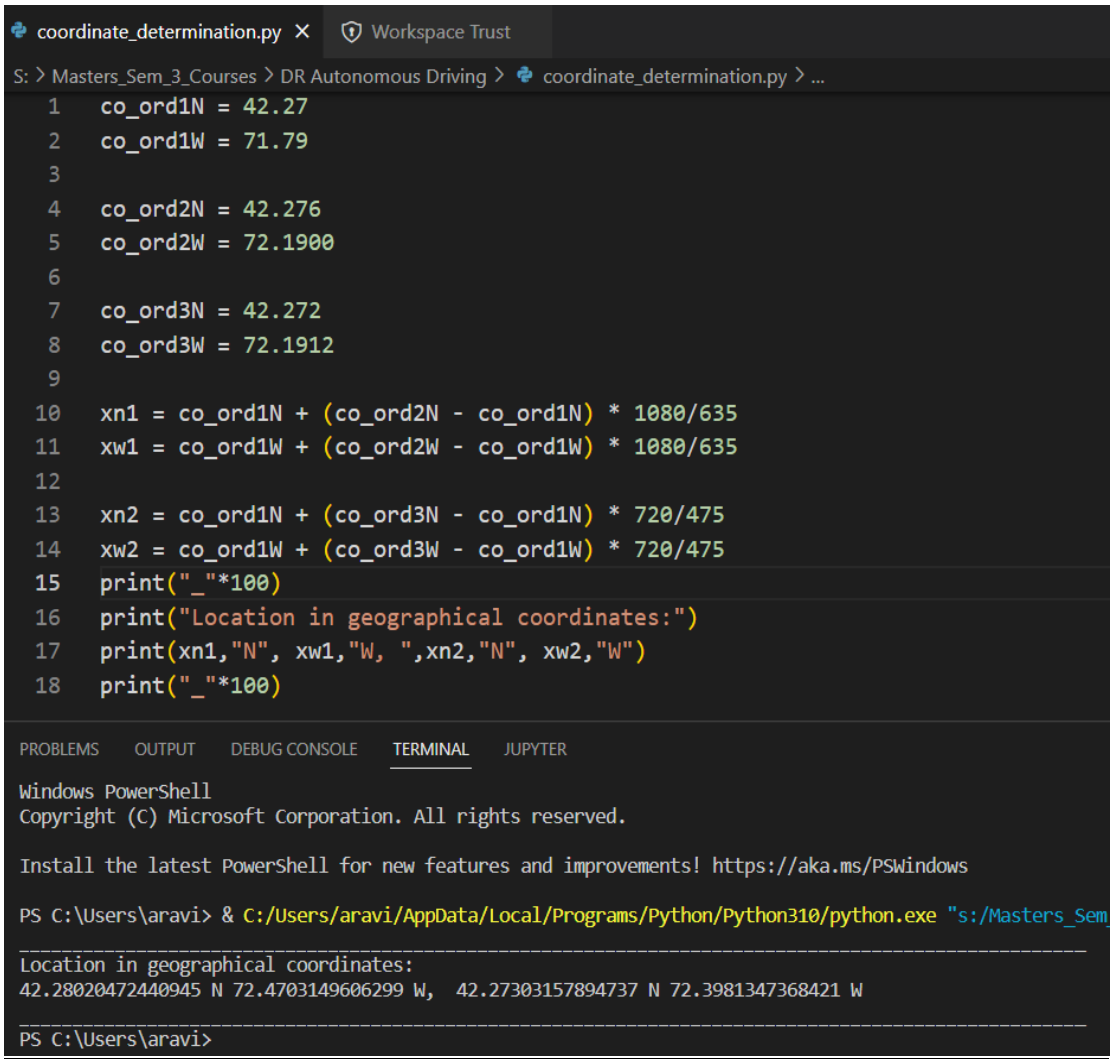


Google earth images with four corner coordinates

Thus, we have determined the X and Y locations in terms of direction coordinates.

3.4.2 Results

The results are displayed in the terminal. The location differs in the range of seconds as shown in the above image.



The python file and the results are attached in the above image.

3.5 Conclusion and future work

We began from determining the homography matrix between pairs of images to perform transformations. After determining the homography, we applied feature matching algorithms. The one that gave a very good quantitative result and accuracy was the SuperGlue pre-trained model. After determining the feature points, we determined the homography using the findHomography() function and applied image transformations using the warpPerspective() function. From there, we applied the object detection, orientation determination algorithms and determined the pixel location of the vehicle. Next, we converted the pixel positions to geographic coordinates using the devised formula and obtain the global coordinates to be utilized for verification of ground truth location in Simultaneous Localization and Mapping. Thus, we have attained the objectives of the project. In the future, we plan to collect data using Ouster OS1 lidar. For thar data, we perform SLAM and utilize our determined locations as the ground-truth positions of the vehicle in the map.

4. References

1. A. Nassar, K. Amer, R. ElHakim and M. ElHelw, "A Deep CNN-Based Framework For Enhanced Aerial Imagery Registration with Applications to UAV Geolocalization," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1594-159410, doi: 10.1109/CVPRW.2018.00201.

2. Zhuo, Xiangyu & Koch, Tobias & Kurz, Franz & Fraundorfer, Friedrich & Reinartz, Peter. (2017). Automatic UAV Image Geo-Registration by Matching UAV Images to Georeferenced Image Data. Remote Sensing. 9. 10.3390/rs9040376.

3. Hemerly, Elder. (2014). Automatic Georeferencing of Images Acquired by UAV's. International Journal of Automation and Computing. 11. 347-352. 10.1007/s11633-014-0799-0.

4. A. Yol, B. Delabarre, A. Dame, J. -É. Dartois and E. Marchand, "Vision-based absolute localization for unmanned aerial vehicles," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3429-3434, doi: 10.1109/IROS.2014.6943040.

5. Gellért Máttyus, Friedrich Fraundorfer, Aerial image sequence geolocalization with road traffic as invariant feature, Image and Vision Computing, Volume 52, 2016, Pages 218-229,ISSN 0262-8856.

6. S. Ahn, H. Kang and J. Lee, "Aerial-Satellite Image Matching Framework for UAV Absolute Visual Localization using Contrastive Learning," 2021 21st International Conference on Control, Automation and Systems (ICCAS), 2021, pp. 143-146, doi: 10.23919/ICCAS52745.2021.9649999.

7. Ding, Lirong & Zhou, Ji & Meng, Lingxuan & Long, Zhiyong. (2020). A Practical Cross-View Image Matching Method between UAV and Satellite for UAV-Based Geo-Localization. Remote Sensing. 13. 47. 10.3390/rs13010047.

8. R. Zhang, F. Xu, H. Yu, W. Yang and H. -C. Li, "Edge-Driven Object Matching for UAV Images and Satellite SAR Images," IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium, 2020, pp. 1663-1666, doi: 10.1109/IGARSS39084.2020.9324021.

9. Ren, Kan & Ding, Lei & Wan, Minjie & Gu, Guohua & Chen, Qian. (2022). Target localization based on cross-view matching between UAV and satellite. Chinese Journal of Aeronautics. 35. 10.1016/j.cja.2022.04.002.

10. Xu, Chuan & Liu, Chang & Li, Hongli & Ye, Zhiwei & Sui, Haigang & Yang, Wei. (2022). Multiview Image Matching of Optical Satellite and UAV Based on a Joint Description Neural Network. Remote Sensing. 14. 838. 10.3390/rs14040838.

11. Robert J. Woodham, "Photometric Method For Determining Surface Orientation From Multiple Images," Opt. Eng. 19(1) 191139 (1 February 1980)

12. Foster, J. Nigel and Sanderson, C. Arthur. Determining object orientation from a single image using multiple information sources.

13. Brou, P. (1984). Using the Gaussian image to find the orientation of objects. *The International Journal of Robotics Research*, *3*(4), 89-125.

14. S. Lavallee and R. Szeliski, "Recovering the position and orientation of free-form objects from image contours using 3D distance maps," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 378-390, April 1995, doi: 10.1109/34.385980.

15. D. Hoiem, A. A. Efros and M. Hebert, "Geometric context from a single image," *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 2005, pp. 654-661 Vol. 1, doi: 10.1109/ICCV.2005.107.

16. Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010, September). Brief: Binary robust independent elementary features. In *European conference on computer vision* (pp. 778-792). Springer, Berlin, Heidelberg.

17. Sarlin, P. E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4938-4947).