Details of the assessment :

Research and Brief:

As I'm a fresher and new to drones, I wasn't aware of GCPs, orthogonal mosaics etc. So I had to learn about those first. I read several blogs about it. In the starting, I thought to apply conv nets to detect GCPs (as I've read in one research paper on marker detection for automated landing). But since in our images, marker keeps its properties preserved (color, structure etc.) and only altitude and rotations of the marker is changing, I realized, Conv net would be a waste of resources and time.

So I've applied image processing algorithms to extract positions of GCPs in the image. But along with correct positions, it also gives some false positives. To remove false positives, we need to apply deep learning only on those patches extracted above (instead of the whole image), to rightly classify between markers and false positives.

For deep learning, I've used 3 layered architecture, which is classifying images with training accuracy of 90% and testing accuracy of 89.63%.

Procedure:

(Note - All the procedures are explained in more detain Ipython notebooks)

GCP Detection (Image Processing part)

1) Preprocessing on the image -
    a) Resize
    b) Convert into grayscale
    c) Smoothing (by bilateralFilter)
    d) adaptive thresholding
    e) Morphological Operation (2 times dilation and 1-time erosion with 3*3 mask)
2) Extract Contours:
    a) Approximate each contour whose area <= 50
    b) Find minAreaRect
    c) Crop the bounded Rectangle (Rotated Rectangle) by rotating the image
        i) Take its complement and again find contours.

ii) For any L shaped, the contour must be a 4 sided square/polygon (4 corners) (#corners are varying between 2 - 6 due to a difference in altitude)
iii) Check if the contour is of a certain shape
iv) Calculate the area

d) If abs(width-height) <= 10
    i) 4 sided contour is calculated by cropping the rounded rectangle from the complement of the original contour
        (a) Used Affine transformation
        (b) Rotate the whole image, then crop the rectangle using affine wrapping
    (2) If area difference less than 35
        (a) Create a binary thresholded image
            (i) Thresholding value is calculated using histogram analysis
        (b) Count the number of white pixels of the size of bounding rectangle from the above binary image
        (c) If number > 0, pass the cropped bounding rectangle to the machine learning model, if the output is 1, then store its (x,y) coordinates w.r.t the image and draw a blue bounding rectangle, otherwise if the output is 0 then draw a red rectangle.

Model Training (DetectionModel) :

A. Train Feed Forward Neural Network with 3 layers
    a. input layer - 1200 * 10 * 10 = 120000 neurons¶
    b. 1st hidden layer - 70 neurons - relu activation function
    c. Dropout with 0.5 probable dropouts to overcome overfitting
    d. 2nd hidden layer - 40 neurons - relu activation function
    e. 3rd hidden layer - 45 neurons - relu activation function
    f. Output Layer - softmax with 2 neurons (0 and 1)
    g. Train model using ADAM OPTIMIZER
    h. Loss Function - sparse_categorical_crossentropy
B. Accuracy
    a. Training - 90%
    b. Test - 89.63%

Preprocessing of the dataset (Data Augmentation):

a. Resize
   - all the cropped images to 10*10

b. Data Augmentation
   - Since data was less, 228 patches with GCP and 351 false patches, we need to augment our data
   - 228 -> 851 true images & 351 -> 601 false images
   - Used - rotations, flip left, flip bottom

c. Created pickle files to store the dataset

d. Dividing dataset into 20% test and 80% training data

Running Instructions :

1. Open notebook GCPDetection.pynb.
2. Put path of the folder (which contains your images) in the finalResult() method
   a. Eg. final result('./AssignmentDataset')
3. Run all the cells.
4. After each image following outputs will be shown:
   a. Image with marked rounding box (press any key on the keyboard to move to next image)
      i. Blue boxes - Model predicting true value
      ii. Red boxed - Model predicting the false value
   b. (x,y) locations of the bounding boxes

Note: The output still contains false positives which can be removed by training our model with more data. The above algorithm is not classifying GCPs that are merged with some another white patch.