In [1]:

```
import cv2
import numpy as np
import math
import os
import time
from matplotlib import pyplot as plt
from keras.models import load_model
```

```
c:\users\shivangi pandey\appdata\local\programs\python\python36\lib\site-p
ackages\h5py\__init__.py:36: FutureWarning: Conversion of the second argum
ent of issubdtype from `float` to `np.floating` is deprecated. In future,
it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

Crop the round bounded rectangle by rotating its axis to align it's axis with the global one

In [2]:

```
def crop_minAreaRect(image, rect):

    centre, dimensions, theta = rect

    width = int(dimensions[0])+4
    height = int(dimensions[1])+4

    box = cv2.boxPoints(rect)
    box = np.int0(box)

    M = cv2.moments(box)

    if(M['m00'] == 0):
        return (False,None)

    cx = int(M['m10']/M['m00'])
    cy = int(M['m01']/M['m00'])

    center = (cx,cy)
    if theta < -45:
        theta = theta + 90
        width, height = height, width

    theta *= math.pi / 180 # convert to rad
    v_x = (math.cos(theta), math.sin(theta))
    v_y = (-math.sin(theta), math.cos(theta))
    s_x = center[0] - v_x[0] * (width / 2) - v_y[0] * (height / 2)
    s_y = center[1] - v_x[1] * (width / 2) - v_y[1] * (height / 2)
    mapping = np.array([[v_x[0],v_y[0], s_x], [v_x[1],v_y[1], s_y]])

    return (True,cv2.warpAffine(image, mapping, (width, height), flags=cv2.WARP_INVERSE
_MAP, borderMode=cv2.BORDER_REPLICATE))
```

Use saved model (previously trained model with accuracy 89% in DetectionModel) to predict the GCPs

In [3]:

```
def predictValue(img,model):

    if(model == None):
        return -1

    # expand (10,10) to (1,10,10)
    img = (np.expand_dims(img,0))
    predictions_single = model.predict(img)

    return np.argmax(predictions_single[0])
```

Detect which of the Countours are valid contour, that satisfy the property of our GCP

To cover wide range, we are considering polygon vertices between 2-6 (should be 4 in practical)

In [4]:

```
def detectValid(cont, ero, oriIm):

    #get the rectangle bounded by the contour (rotated rectatngle)
    rect = cv2.minAreaRect(cont)
    #Crop the rounded rectangle as an image
    resu = crop_minAreaRect(ero, rect)

    if(not resu[0]):
        return (False,None,None)

    im = resu[1]
    im = cv2.bitwise_not(im)

    _, Cs, _ = cv2.findContours(im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

    img2 = oriIm.copy()

    m = 0
    maxC = None

    for c in Cs:
        if(c is None or len(c) == 0):
            continue
        peri = cv2.arcLength(c, True)
        hull = cv2.convexHull(c)
        app = cv2.approxPolyDP(hull, 0.04 * peri, True)

        if(len(app) >= 2 and len(app) <= 6):
            Ar = cv2.contourArea(app)
            if(Ar >= m):
                maxC = app
                m = Ar

    if(maxC is None):
        return (False,maxC,resu[1])
    return (True,maxC,resu[1])
```

Here Histogram is used to get optimum tresholding value.

For this, I'm considering 10th brightest pixel present in the grayscale image

In [9]:

```
def findGCPPixelShade(histogram):
    i = 255
    count = 0;
    while(i>=150):
        if histogram[i,0] != 0:
            count += 1
            if count == 5:
                return i
        i -= 1
    return i
```

Following processes are taking place, here

- Resize image, convert into grayscale,

- apply bilateral filter for blurring, adaptive thresholding, 2 iterations of dilation and 1 time erosion

- Extract contours with area less than 50 and a valid contour by using above method

- Create a binary image by thresholding value obtained from histogram of the image

- Count the no. of white pixels in to contour region in the binary image, if it's not 0, than pass the above cropped image to the trained model, to classify weather it's GCP or not

- model will return 1 if GCP, 0 otherwise

In [13]:

```
def extractContours(imPath,resize,name,model):
    image = cv2.imread(imPath)
    small = cv2.resize(image,resize)

    #Convert into gray scale
    gray_image = cv2.cvtColor(small, cv2.COLOR_BGR2GRAY)

    # create the histogram
    histogram = cv2.calcHist([gray_image], [0], None, [256], [0, 256])
    index = findGCPPixelShade(histogram)

    ret,binImg = cv2.threshold(gray_image,index,255,cv2.THRESH_BINARY)

    #smoothning filter
    blur2 = cv2.bilateralFilter(gray_image,15,10,10)

    th2 = cv2.adaptiveThreshold(blur2,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BIN
ARY,73,-40)

    kernel = np.ones((1,1),np.uint8)
    dilation = cv2.dilate(th2,kernel,iterations = 2)
    erosion = cv2.erode(dilation,kernel,iterations = 1)

    im2, contours, hierarchy = cv2.findContours(erosion,cv2.RETR_EXTERNAL,cv2.CHAIN_APP
ROX_SIMPLE)

    img = small.copy()

    # for each contour
    max = 0
    con = []
    k = 0
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if(area <=50):
            epsilon = 0.005*cv2.arcLength(cnt,True)
            approx = cv2.approxPolyDP(cnt,epsilon,True)
            x,y,w,h = cv2.boundingRect(approx)
            if(approx is None):
                continue
            v = math.fabs(w-h)

            if(v <= 10):
                area = cv2.contourArea(approx)
                app = detectValid(approx, erosion, img)

                if(not app[0]):
                    continue

                if(app[2].shape[0] > 20 or app[2].shape[1] > 20):
                    continue
                area2 = cv2.contourArea(app[1])
                areaDif = math.fabs(area2 - area)

                a1 = cv2.countNonZero(binImg[y:y+h+4,x:x+w+4])
                if(a1 != 0):
                    modelImg = cv2.resize(app[2],(10,10))
                    print('*',end='-')
                    if(predictValue(modelImg,model) == 1):
```

```
                    con.append(approx)
                else:
                    img = cv2.rectangle(img,(x-2,y-2),(x+w+4,y+h+4),(0,0,255),1)

    #Return (x,y) points and the image with valid rectangles
    points = []
    for c in con:
        x,y,w,h = cv2.boundingRect(c)
        points.append((x,y))
        img = cv2.rectangle(img,(x-2,y-2),(x+w+4,y+h+4),(255,0,0),1)

    return (len(con),img,points)
```

An image is returned after each iternation with red rectangles for values = 0 (not GCP according to the model) and Blue for GCP classified value by the model

All the rectangles are identified using above image processing methods, which further are sent to machine learning model for further classification

Current accuracy is 89%, which can be imporved significantly with more data

In [14]:

```
def finalResult(path):
    model = load_model('GCP_Detection_Model.h5')
    for i,ls in enumerate(os.listdir(path)):

        print("\nfor image : "+ls+"\n")

        result =  extractContours(path+"/"+ls,(1366,768),ls,model)

        if(result[0] == 0):
            print("No marker detected!")
        else:
            print("\n(x,y) : ",result[2])

        cv2.imshow(ls, result[1])
        cv2.waitKey()
        cv2.destroyAllWindows()
```

In [15]:

```
finalResult('./AssignmentDataset')
```

```
c:\users\shivangi pandey\appdata\local\programs\python\python36\lib\site-p
ackages\keras\models.py:282: UserWarning: No training configuration found
in save file: the model was *not* compiled. Compile it manually.
  warnings.warn('No training configuration found in save file: '
for image : DJI_0000_set2 (105).JPG

*-*-*-*-
(x,y) :  [(54, 522)]

for image : DJI_0000_set2 (106).JPG

No marker detected!

for image : DJI_0000_set2 (107).JPG

*-No marker detected!

for image : DJI_0000_set2 (108).JPG

*-No marker detected!

for image : DJI_0000_set2 (109).JPG

*-No marker detected!

for image : DJI_0000_set2 (110).JPG

*-*-
(x,y) :  [(79, 377)]

for image : DJI_0000_set2.JPG

*-No marker detected!

for image : DJI_0036.JPG

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
(x,y) :  [(586, 661), (329, 515), (438, 310)]

for image : DJI_0083.JPG

*-*-*-*-
(x,y) :  [(833, 526), (791, 144)]

for image : DJI_0086.JPG

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
(x,y) :  [(857, 624), (508, 418), (487, 413), (505, 412), (507, 403), (80
0, 197), (774, 29)]

for image : DJI_0421.JPG

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
(x,y) :  [(568, 750), (666, 741), (766, 682), (240, 598)]

for image : DJI_0422.JPG

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
(x,y) :  [(62, 750), (190, 693), (137, 690), (135, 657), (670, 655), (91,
572), (104, 568), (46, 553), (22, 541), (182, 526), (174, 510), (227, 11
1)]

for image : DJI_0577.JPG
```

```
*-*-No marker detected!

for image : DJI_0616.JPG

*-*-
(x,y) :  [(85, 614)]

for image : DJI_0617.JPG

*-*-No marker detected!

for image : DJI_0630.JPG

*-*-*-No marker detected!

for image : DJI_0631.JPG

*-*-No marker detected!

for image : DJI_0632.JPG

*-*-*-
(x,y) :  [(323, 383)]

for image : DJI_0633.JPG

*-*-No marker detected!

for image : DJI_0634.JPG

*-*-No marker detected!

for image : DJI_0791.JPG

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
(x,y) :  [(497, 762), (487, 762), (475, 760), (475, 750), (459, 748), (46
0, 745), (527, 742), (475, 739), (479, 726), (501, 721), (521, 704), (597,
657), (560, 650), (566, 646)]

for image : DJI_0792.JPG

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
*-*-*-*-*-
(x,y) :  [(26, 749), (46, 739), (11, 730), (8, 726), (68, 724), (115, 69
8), (63, 698), (0, 692), (68, 690), (15, 690), (116, 689), (136, 686), (7,
686), (4, 682), (72, 676), (28, 672), (24, 672), (6, 671), (167, 668), (5
7, 665), (80, 663), (60, 662), (67, 657), (180, 653), (200, 639), (320, 62
2)]

for image : DSC00001.JPG

*-No marker detected!

for image : DSC00004.JPG

*-No marker detected!

for image : DSC01400.JPG
```

```
No marker detected!

for image : DSC01405.JPG

*-*-No marker detected!

for image : DSC01410.JPG

*-*-No marker detected!

for image : DSC01453.JPG

*-
(x,y) :  [(708, 222)]

for image : DSC01588.JPG

*-*-*-
(x,y) :  [(920, 49)]

for image : DSC01590.JPG

*-No marker detected!

for image : DSC01798.JPG

*-*-*-*-*-
(x,y) :  [(872, 315)]

for image : DSC02118.JPG

*-*-*-No marker detected!

for image : DSC02119.JPG

*-*-*-
(x,y) :  [(1082, 388)]

for image : DSC02120.JPG

*-No marker detected!

for image : DSC08876.JPG

*-
(x,y) :  [(947, 32)]

for image : DSC09300.JPG

*-No marker detected!

for image : M1_F1.3_0337.JPG

*-*-No marker detected!

for image : M1_F1.3_0338.JPG

*-*-No marker detected!

for image : M1_F1.3_0339.JPG
```

```
*-*-*-*-*-*-*-*-
(x,y) :  [(905, 526), (907, 516), (911, 493), (911, 490)]

for image : M1_F1.3_0405.JPG

*-*-*-*-*-
(x,y) :  [(643, 625), (220, 378)]
```