2-bloques-mm

July 19, 2022

[1]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
```

[2]:
```
import glob

import PIL.ExifTags
import PIL.Image
```

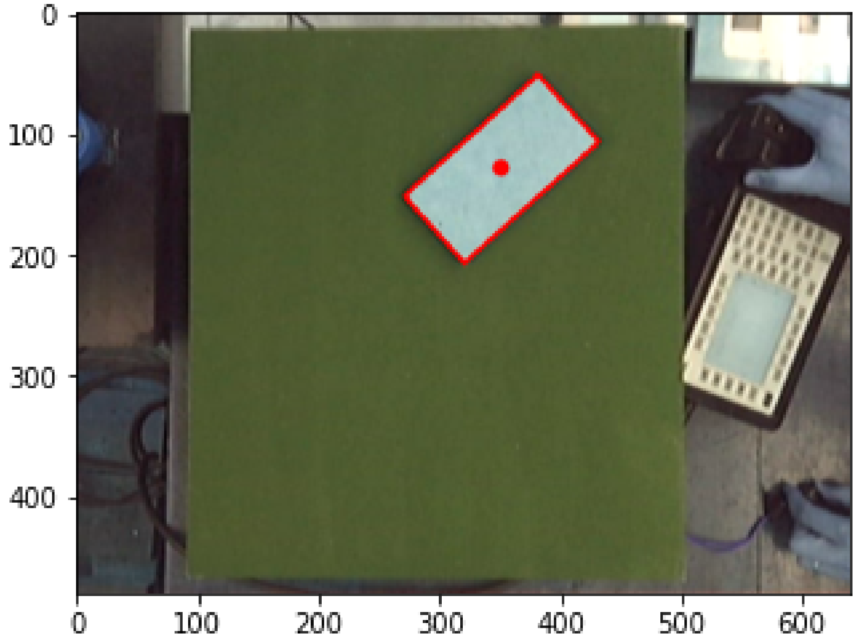
1 Busqueda de bloques

[3]:
```
bloques = [f'./imagenes_tp/img_bloques/imgBloque{i}.png' for i in range(1,16)]
```

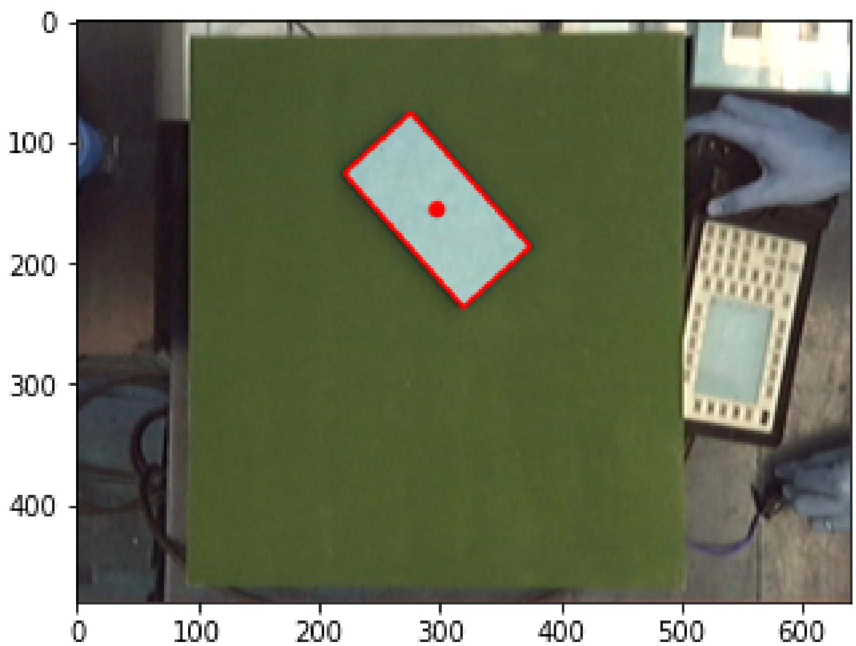
[4]:
```
centers = []
vertices = []
dimensions = []
orientation = []
bloques_bin = []
for b in bloques:
    print(b)
    img = cv.imread(b)
    imgray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    # Binarizamos con Otzu
    ret, img_bin = cv.threshold(imgray,30,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
    #bloques_bin.append(img_bin)

    img_cut = img_bin[20:500, 0:500]
    contours, hierarchy = cv.findContours(img_cut, cv.RETR_TREE, cv.
↪CHAIN_APPROX_SIMPLE, offset=(0,20))

    img_out = img.copy()
    cv.drawContours(img_out, contours, -1, (0,255,0), 3)

    bloque_contours = []
    for c in contours:
```

```
        if cv.contourArea(c) < 10000:
            continue
        bloque_contours.append(c)

    img_out = img.copy()
    cnt = bloque_contours[0]
    area = cv.contourArea(cnt)
    long = cv.arcLength(cnt, False)
    center, dim, rot = cv.minAreaRect(cnt)
    vert = cv.boxPoints([center, dim, rot])
    print('Longitud: {} - Área: {}'.format(long, area))
    print('Centro: {} - Dimension: {} - Rotacion: {}'.format(center, dim, rot))

    centers.append(center)
    vertices.append(vert)
    dimensions.append(dim)
    orientation.append(rot)

    for c in bloque_contours:
        # compute the center of the contour
        M = cv.moments(c)
        cX = int(M["m10"] / M["m00"])
        cY = int(M["m01"] / M["m00"])
        cv.drawContours(img_out, bloque_contours, -1, (255,0,0), 3)
        cv.circle(img_out, (cX, cY), 7, (255, 0, 0), -1)
    plt.imshow(img_out)
    plt.show()
```

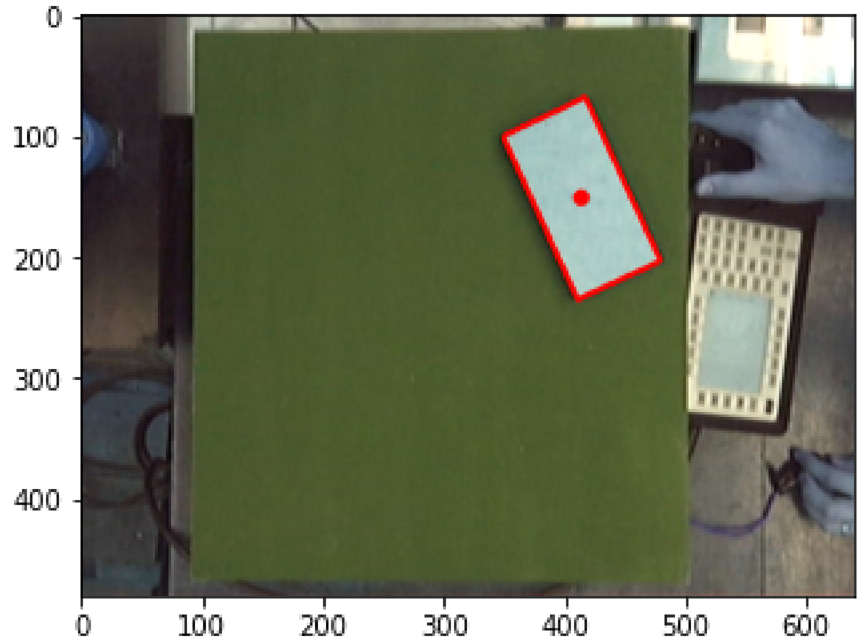
```
./imagenes_tp/img_bloques/imgBloque1.png
Longitud: 455.77878522872925 - Área: 11132.5
Centro: (351.3536376953125, 128.78176879882812) - Dimension: (75.96466064453125,
150.368408203125) - Rotacion: 48.01278305053711
```



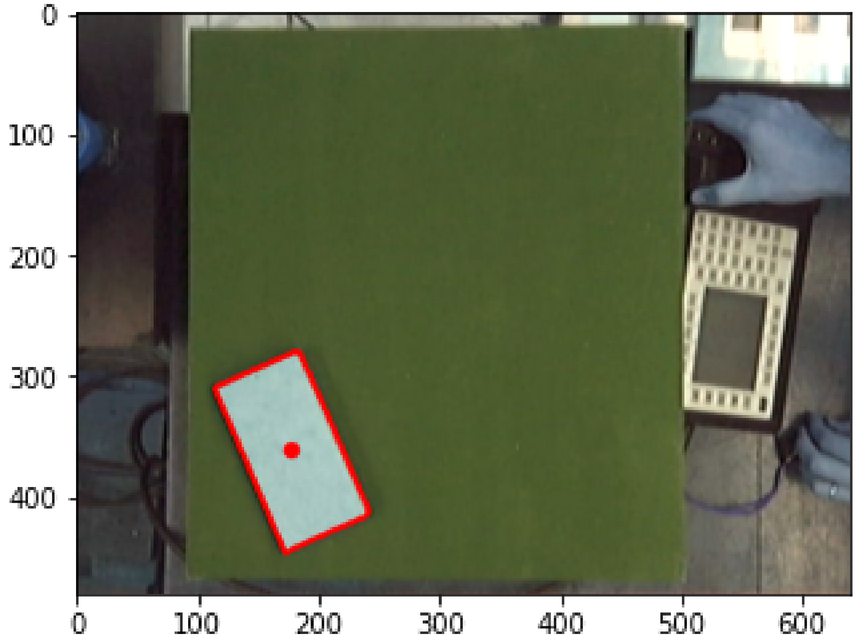
./imagenes_tp/img_bloques/imgBloque2.png
Longitud: 449.8792916536331 - Área: 11008.5
Centro: (298.1031799316406, 156.44198608398438) - Dimension: (149.1449737548828, 75.25286865234375) - Rotacion: 48.12213134765625



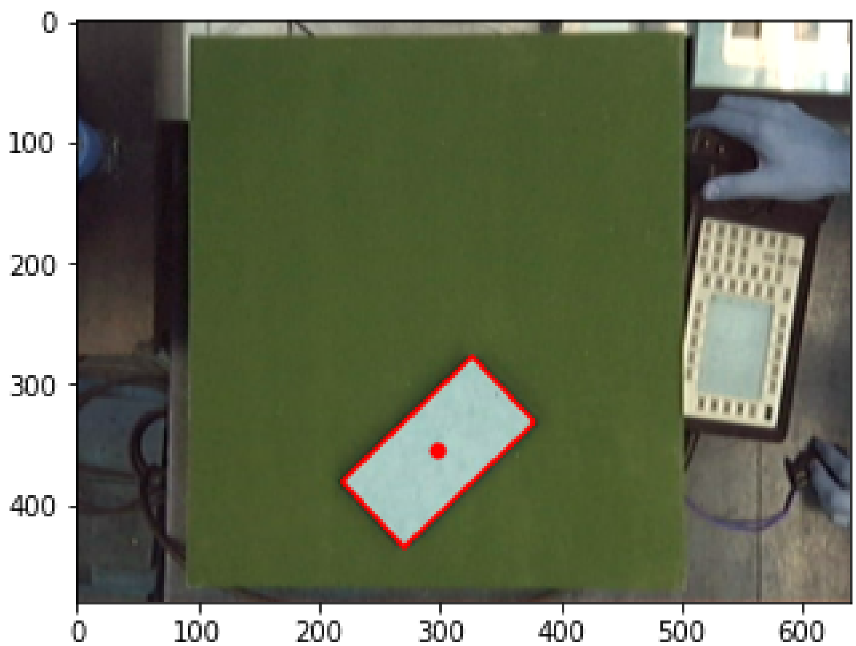
./imagenes_tp/img_bloques/imgBloque3.png
Longitud: 481.457931637764 - Área: 11217.5
Centro: (414.2087707519531, 151.40403747558594) - Dimension: (150.0602264404297, 76.54962158203125) - Rotacion: 65.30844116210938



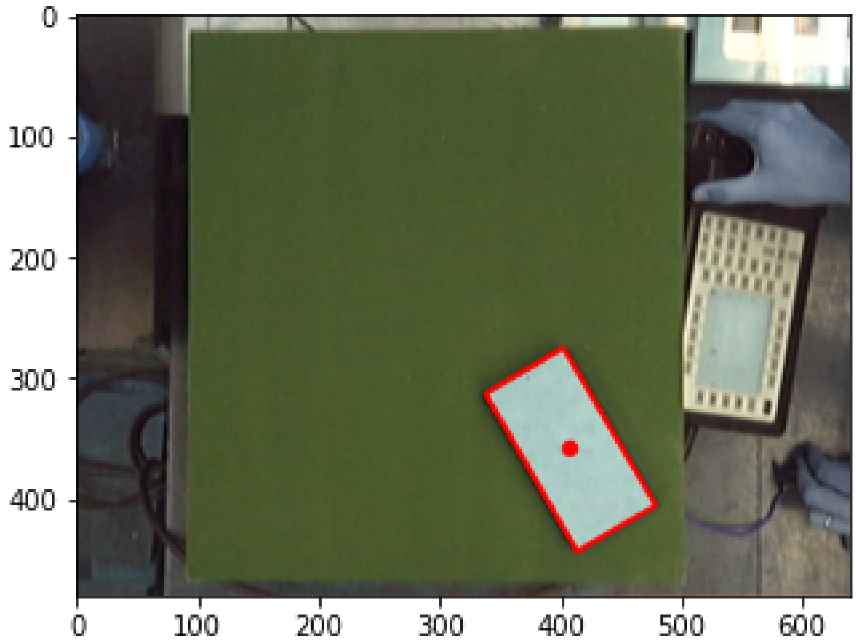
./imagenes_tp/img_bloques/imgBloque4.png
Longitud: 476.6589421033859 - Área: 11384.5
Centro: (178.5289306640625, 361.8793640136719) - Dimension: (150.0324249267578, 78.36514282226562) - Rotacion: 66.61477661132812



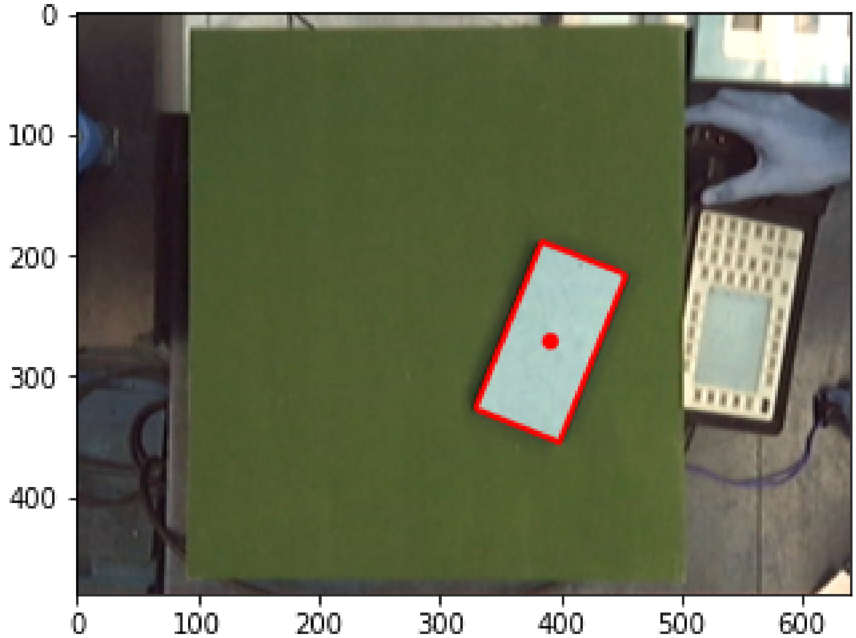
```
./imagenes_tp/img_bloques/imgBloque5.png
Longitud: 451.77878749370575 - Área: 11130.0
Centro: (299.0718994140625, 355.9086608886719) - Dimension: (75.65641021728516,
150.0962677001953) - Rotacion: 46.332218170166016
```



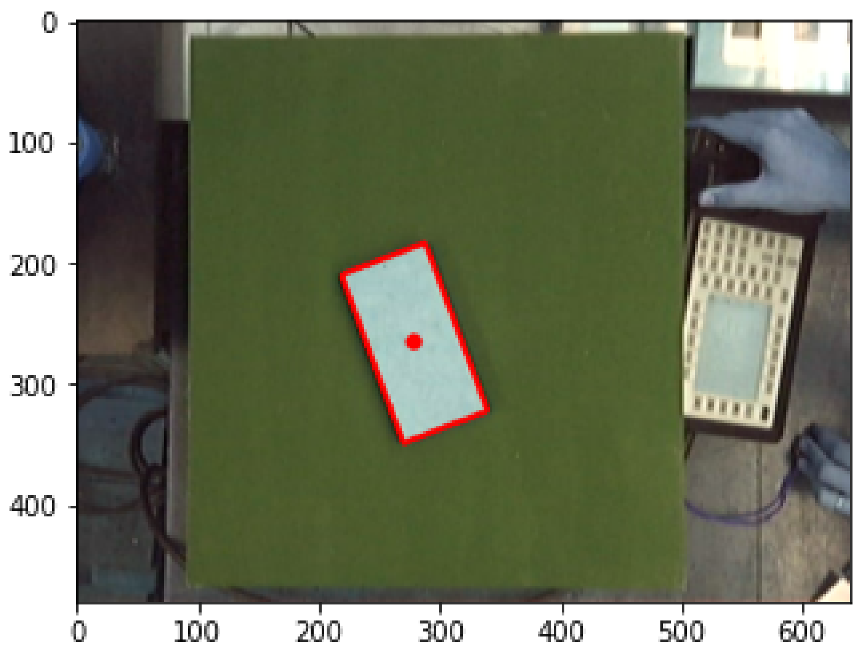
./imagenes_tp/img_bloques/imgBloque6.png
Longitud: 480.61225938796997 - Área: 11307.0
Centro: (408.4663391113281, 358.72796630859375) - Dimension: (152.16903686523438, 76.08451843261719) - Rotacion: 59.74356460571289



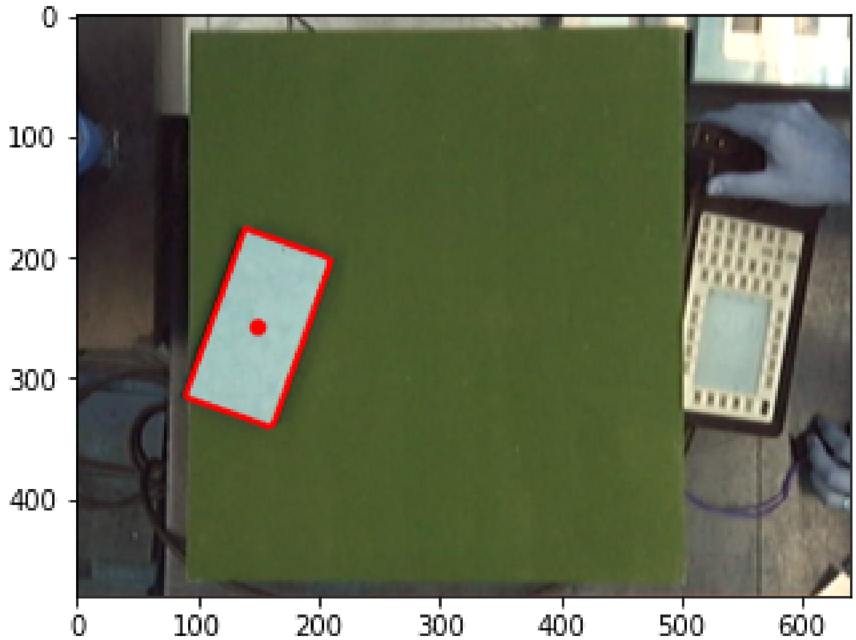
./imagenes_tp/img_bloques/imgBloque7.png
Longitud: 480.34523379802704 - Área: 11133.5
Centro: (392.0344543457031, 271.41375732421875) - Dimension: (75.7636947631836, 150.0418243408203) - Rotacion: 21.801406860351562



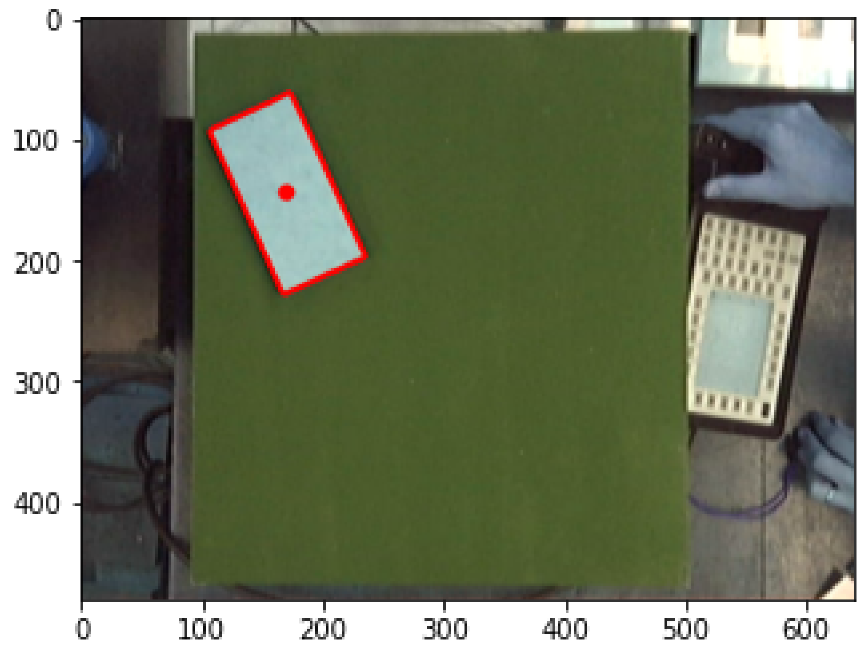
./imagenes_tp/img_bloques/imgBloque8.png
Longitud: 475.20309841632843 - Área: 10996.5
Centro: (279.3534851074219, 266.1330261230469) - Dimension: (149.47378540039062, 75.49943542480469) - Rotacion: 69.56716918945312



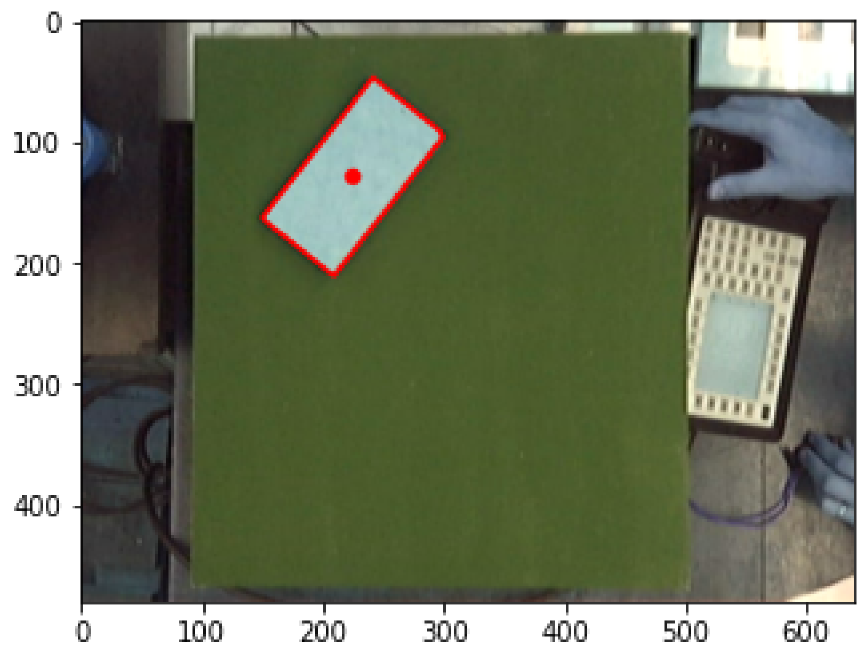
./imagenes_tp/img_bloques/imgBloque9.png
Longitud: 476.4751765727997 - Área: 11248.0
Centro: (150.64501953125, 258.4842529296875) - Dimension: (77.56439971923828, 148.94139099121094) - Rotacion: 19.583993911743164



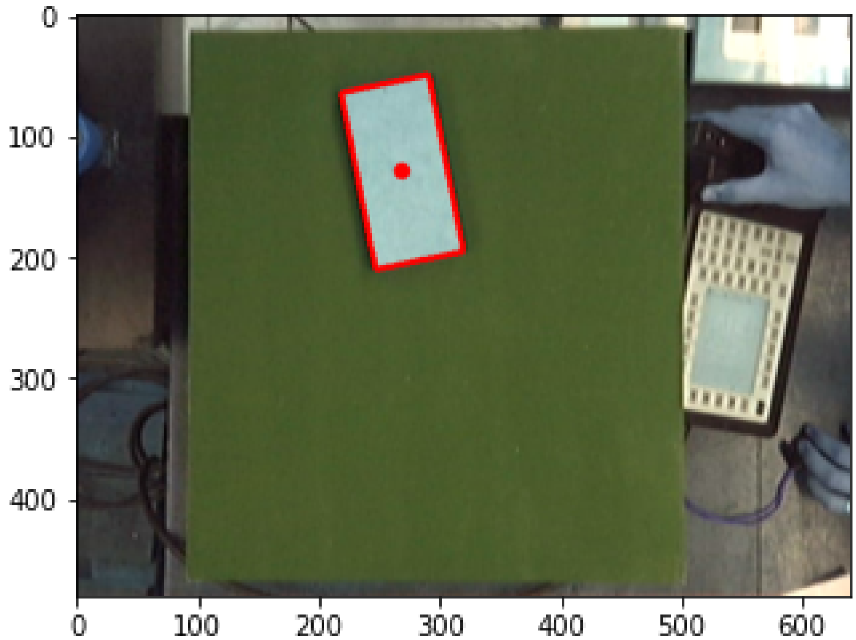
./imagenes_tp/img_bloques/imgBloque10.png
Longitud: 476.80107748508453 - Área: 11166.5
Centro: (170.59930419921875, 144.8184814453125) - Dimension: (150.45875549316406, 76.05697631835938) - Rotacion: 65.55604553222656



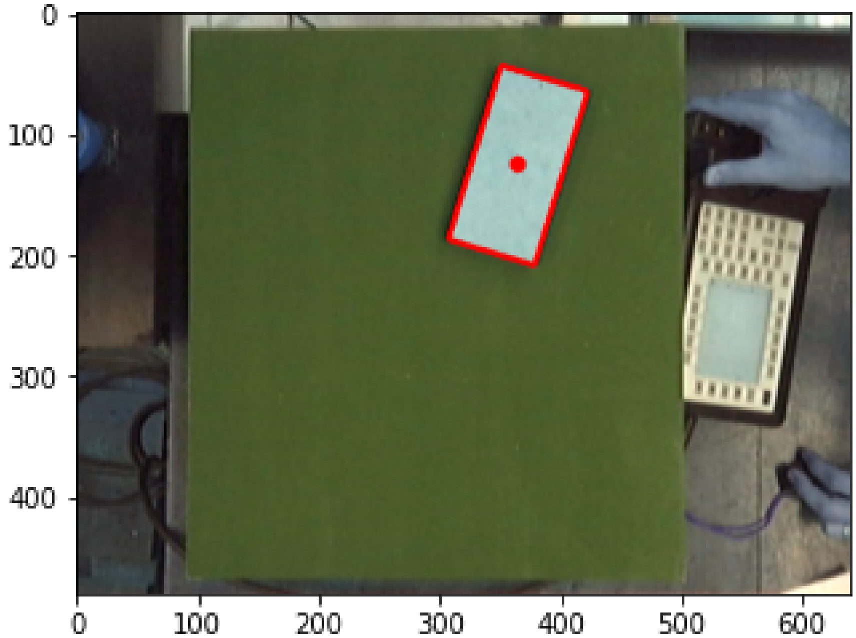
./imagenes_tp/img_bloques/imgBloque11.png
Longitud: 461.5950139760971 - Área: 11280.5
Centro: (225.51092529296875, 129.0948944091797) - Dimension: (77.3231201171875, 149.13473510742188) - Rotacion: 38.41805648803711



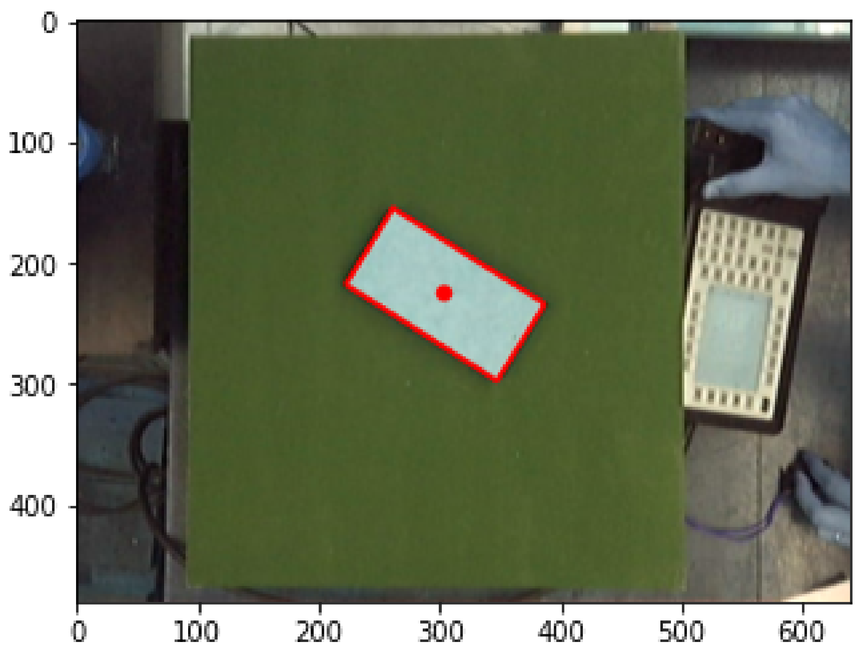
./imagenes_tp/img_bloques/imgBloque12.png
Longitud: 469.0365778207779 - Área: 11024.0
Centro: (269.27764892578125, 129.64132690429688) - Dimension: (150.707763671875, 75.18319702148438) - Rotacion: 78.92979431152344



./imagenes_tp/img_bloques/imgBloque13.png
Longitud: 478.0193328857422 - Área: 11195.0
Centro: (365.1622619628906, 125.82916259765625) - Dimension: (75.65897369384766, 151.77186584472656) - Rotacion: 16.966148376464844



./imagenes_tp/img_bloques/imgBloque14.png
Longitud: 471.511754155159 - Área: 11029.5
Centro: (304.71612548828125, 225.95089721679688) - Dimension:
(149.59518432617188, 75.6281509399414) - Rotacion: 32.195735931396484



```
./imagenes_tp/img_bloques/imgBloque15.png
Longitud: 473.5117540359497 - Área: 11208.5
Centro: (374.7359619140625, 319.0224609375) - Dimension: (150.73167419433594,
76.63783264160156) - Rotacion: 32.0053825378418
```

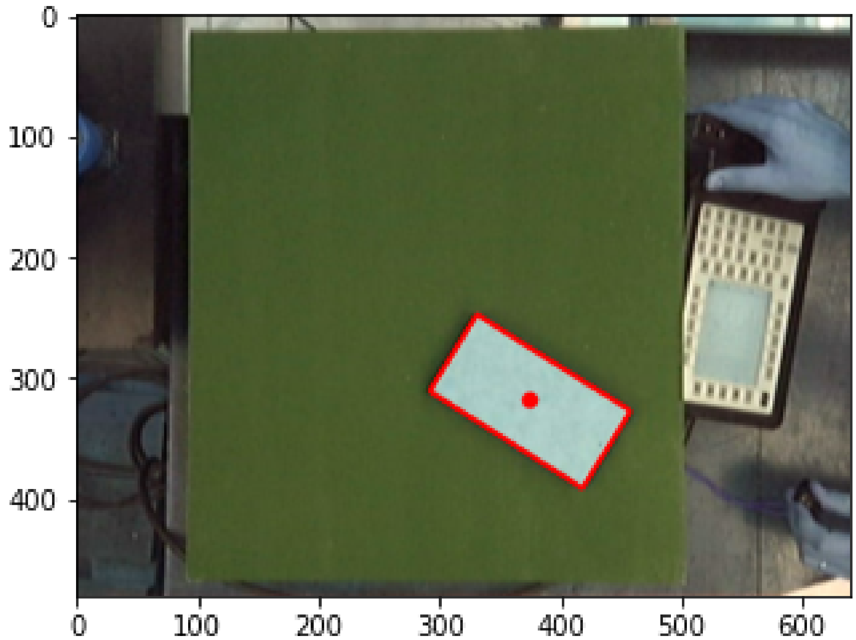


image = cv2.imread(args["image"]) gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) blurred = cv2.GaussianBlur(gray, (5, 5), 0) thresh = cv2.threshold(blurred, 60, 255, cv2.THRESH_BINARY)[1]

[5]:
```python
import pandas as pd
pd.DataFrame(centers).to_csv("./save_data/centers.csv", index=False)
#pd.DataFrame(vertices).to_csv("./save_data/vertices.csv", index=False)
pd.DataFrame(dimensions).to_csv("./save_data/dimensions.csv", index=False)
pd.DataFrame(orientation).to_csv("./save_data/orientation.csv", index=False)
```

[6]:
```python
vs = []
for v in vertices:
    #print(v)
    vs.append(v.tolist())
```

[7]:
```python
pd.DataFrame(vs).to_csv("./save_data/vertices.csv", index=False)
```

2 Transformacion de coordenadas a milimetros

2.1 Read Saved data

[8]:
```
import pandas as pd
```

[9]:
```
mtx = pd.read_csv("./save_data/camara_matrix.csv")
mtx = mtx.to_numpy()
```

[10]:
```
dist = pd.read_csv("./save_data/dist_coefficients.csv")
dist = dist.to_numpy()
dist
```

[10]:
```
array([[0.04997142, 0.        , 0.        , 0.        , 0.        ]])
```

[11]:
```
rotation = pd.read_csv("./save_data/rotation_df.csv").to_numpy()
rotation
```

[11]:
```
array([[ 0.99919631, -0.00779203,  0.03931953],
       [ 0.00771164,  0.99996785,  0.00219582],
       [-0.03933537, -0.00189083,  0.99922428]])
```

[12]:
```
translation = pd.read_csv("./save_data/translation_df.csv").to_numpy()
translation
```

[12]:
```
array([[-108.62975346],
       [-150.3410905 ],
       [ 703.32230507]])
```

2.2 Convertir a mm

[13]:
```
# R^-1
inv_R = np.linalg.inv(rotation)
#K^-1
inv_K = np.linalg.inv(mtx)
```

[14]:
```
def px_to_mm(u,v):
    #s * [uv1] = K * [R|t] * XYZ = K* (R*XYZ1 + t)
    # (s * [uv1] * K^-1 - t) * R^-1 = XYZ1
    # s * [uv1] * K^-1 * R^-1 - t * R^-1 = XYZ1

    #genero el vector [uv1] y lo traspongo
    uv = np.array([[u,v,1]], dtype=np.float64).T

    # t * R^-1
    d = inv_R.dot(translation)

    # [uv1]*K^-1*R^-1
```

```
    i = inv_R.dot(inv_K.dot(uv))

    # Resuelvo s (asumo Z=0). cociente de z
    s = d[2][0]/i[2][0]
    vector = inv_R.dot( s * inv_K.dot(uv) - translation)
    return vector
```

2.3 Ubicacion del centro en mm

[15]:
```
#print(f"CENTERS : {centers}")
cs = []
for center in centers:
    center_mm = px_to_mm(center[0], center[1])[:2]
    cs.append(center_mm)
    print(f"({center_mm[0][0]}, {center_mm[1][0]})")
#print("CENTERS MM: ", cs)
```

```
(139.76941914305286, 50.56909128616893)
(94.08573979683331, 74.50923474872253)
(193.75628376909057, 69.84356468798276)
(-8.323128550225952, 252.86829899641413)
(96.26989232029365, 246.32190707649045)
(190.19215561203148, 247.50135252134027)
(175.58224456531758, 172.83913444027306)
(78.6263922471499, 169.16438746579928)
(-33.44224960794508, 163.51443095901487)
(-16.791129005006706, 64.81497096555786)
(30.98532876228561, 51.07365585693049)
(68.97470899793278, 51.463168428620484)
(151.60632312171018, 48.00668443527267)
(100.26337404921577, 134.36723040605926)
(161.08403045402258, 213.82363630738715)
```

3 Dimension de bloques en mm

[16]:
```
saved_vertices = []
for v in vertices:
    v_data = []
    for fila in v:
        v_mm = px_to_mm(fila[0], fila[1])[:2]
        v_data.append(v_mm)
    saved_vertices.append(v_data)
print(saved_vertices[0])
```

```
[array([[69.7979542 ],
       [69.76352641]]), array([[165.40104537],
       [-16.91893145]]), array([[209.20033437],
```

```
[ 31.5229387 ]]), array([[114.07292849],
[118.22789523]])]
```

[17]:
```
saved_dims = []
for v in saved_vertices:
    w = np.sqrt((v[1][0]-v[0][0])**2+(v[1][1]-v[0][1])**2)
    h = np.sqrt((v[3][0]-v[0][0])**2+(v[3][1]-v[0][1])**2)

    if w > h:
        saved_dims.append([w,h])
    else:
        saved_dims.append([h,w])
```

[18]:
```
saved_dims
```

[18]:
```
[[array([129.04960109]), array([65.64349468])],
 [array([129.10657462]), array([64.9573402])],
 [array([128.95340362]), array([65.4152059])],
 [array([130.06763135]), array([68.58359549])],
 [array([129.90582958]), array([65.20116486])],
 [array([130.09429933]), array([65.31266715])],
 [array([128.90310949]), array([64.87775129])],
 [array([129.10203381]), array([65.38876927])],
 [array([129.60942645]), array([67.92954538])],
 [array([131.15871555]), array([66.39703208])],
 [array([129.11330471]), array([67.4479042])],
 [array([130.48589193]), array([65.09655317])],
 [array([130.42098375]), array([65.08619665])],
 [array([129.18590782]), array([65.35166339])],
 [array([129.09695697]), array([66.0486])]]
```

3.1 Calculo error en las mediciones

[19]:
```
real_w = 65
real_h = 130

error_x = []
error_y = []

for dim in saved_dims:
    error_x.append((abs(real_h - dim[0]) / dim[0])*100)
    error_y.append((abs(real_w - dim[1]) / dim[1])*100)
```

[20]:
```
import matplotlib.pyplot as plt
```

[21]:
```
plt.plot(range(len(saved_dims)), error_x, label = 'Error W (%)')
plt.ylabel('Perc relative error')
```

```
plt.plot(range(len(saved_dims)), error_y, label = 'Error H  (%)')
plt.legend()
plt.show()
```