III. 3. Realisasi Program

```
61  ret, thresh = cv2.threshold(clone, 150, 255, cv2.THRESH_BINARY_INV)
62  cnts = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE
63  cnts = imutils.grab_contours(cnts)
67
68  for contour in cnts:
69      area = cv2.contourArea(contour)
70      if 5000 > area > 500:
71          # cv2.drawContours(orig, contour, -1, (240, 0, 159), 2)
72          area = cv2.contourArea(contour)
73          M = cv2.moments(contour)
74          cX = int(M["m10"] / M["m00"])
75          cY = int(M["m01"] / M["m00"])
76          (x, y, w, h) = cv2.boundingRect(contour)
77          x = x + int(w/4)
78          y = y + int(h/4)
79          w = int(w/2)
80          h = int(h/2)
81          cv2.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)
82          slot.append((x, y, w, h))
83
84  np.save("lot7v2.npy", slot)
```

Pada gambar di atas di perlihatkan potongan program untuk mendeteksi slot parkir. Program di atas dieksekusi setelah tepian garis parkir di deteksi dan d gambar. Baris 62 dan 63 adalah kode untuk mencari kontur dengan *library* dari OpenCV. Untuk setiap kontur yang terdeteksi dilakukan pengulangan seperti pada baris 68. Jika area kontur memenuhi kriteria parkiran, maka di cari titik tengah kontur dengan *library* cv2.*moments*(). Setelah mendapat titik tengah, di cari koordinat dari setengah persegi yang melingkupi kontur. Pada akhir setiap pengulangan, koordinat persegi yang didapat di masukan ke dalam daftar *array.* Lalu pada akhir program, daftar *array* disimpan ke dalam berkas berformat *.*npy.*

```
cnts = np.load('lot4v2.npy')
print(cnts)
for c in cnts:
    (x, y, w, h) = c

    (score, diff) = compare_ssim(grayA[y:y+h, x:x+w], grayB[y:y+h, x:x+w], full=True)
    diff = (diff * 255).astype("uint8")

    print(score)
    if score < 0.7:
        colour = (0, 0, 255)
    else:
        colour = (0, 255, 0)
    cv2.rectangle(imgB, (x, y), (x + w, y + h), colour, 2)
```

Gambar di atas adalah program untuk mendeteksi kosong atau tidaknya slot parkir. Slot yang telah terdeteksi pada program sebelumnya akan di *load* dan digunakan pada program ini. Untuk setiap koordinat slot yang telah terdeteksi, akan dilakukan komparasi dengan *library compare_ssim.* Citra yang bandingkan hanya pada koordinat yang telah dideteksi saja. Sehingga tidak semua citra dideteksi, namun akibatnya proses komparasi dilakukan pada setiap slot dalam waktu yang berbeda. Semakin cepat processor, maka proses ini semakin lancar. Pendeteksian akan benar-benar terjadi secara *realtime*.