Hough Transform

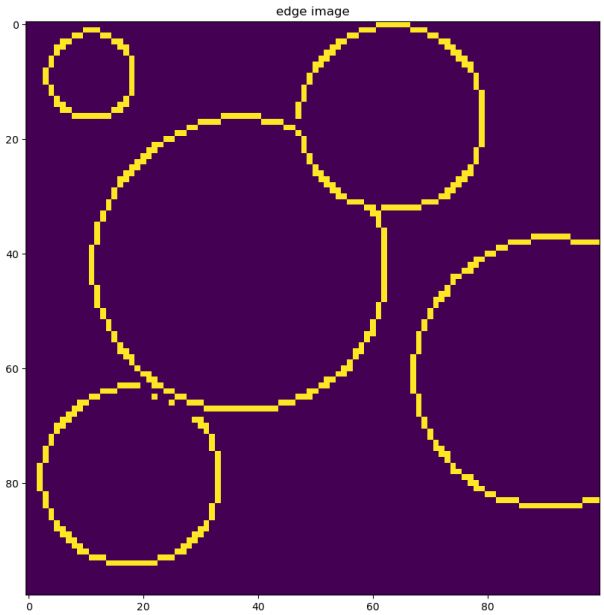
Exercise 1

בתרגיל זה נכתוב בעצמנו Hough Transform שמזהה מעגלים בתמונה.

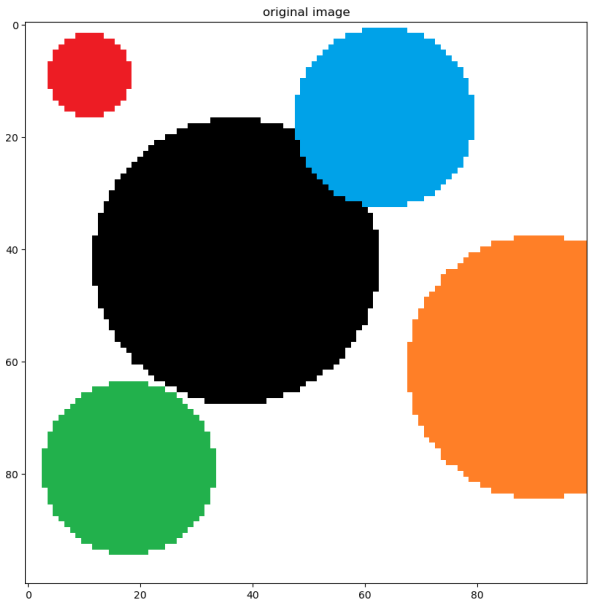
שלב ראשון בתהליך זה לזהות את הקצוות בתמונה ולכן נפעיל Canny על התמונה, זה יהיה הקלט לאלגוריתם שנכתוב.

```
# TODO: Canny edge detection of image
mag_im = cv2.Canny(im, 50, 400)
plt.figure(figsize=figsize)
plt.imshow(mag_im)
plt.title("edge image")
plt.show()
```

תמונה אחרי Canny



תמונת מקור



יצירת Accumulation Matrix

מטריצת האקומולציה היא מטריצה תלת-ממדית עם שני צירים (a,b) לשיעור הנקודה של מרכז המעגל, וציר נוסף לרדיוס המעגל, טווח הרדיוסים בעצם משקף איזה גדלי מעגלים נסמן בתמונה אני בחרתי 0-30px.

בכל הצירים יהיה לנו bucket לכל מספר שלם בתחום כי ה step הגדרתי להיות 1 עבור כל הצירים.

אם נגדיל את rmax במחברת המקורית ל 30 נקבל התאמה טובה יותר לעיגול הכי גדול בתמונה.

```
rmin = 0
rmax = 30
r_step = 1
r = np.arange(rmin,rmax+1,r_step)
#~~~FILL REST HERE~~~
a_step = 1
a = np.arange(0,mag_im.shape[1],a_step)
b_step = 1
b = np.arange(0,mag_im.shape[0],b_step)
# TODO: init accumulation matrix (one line)
# watch out of the order- which comes first? rows or cols?
acc_mat = np.zeros((b.shape[0],a.shape[0],r.shape[0]))
```

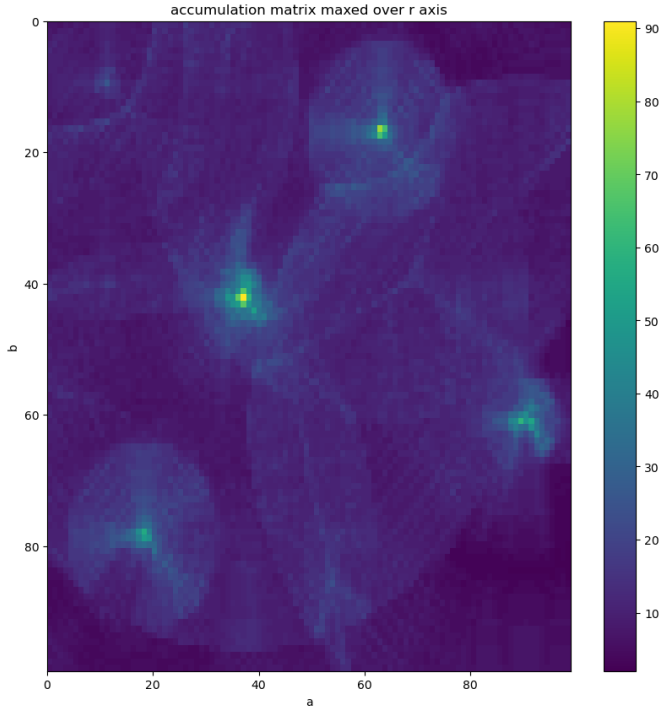
מילוי מטריצת האקומולציה

נעבור על כל הנקודות שזיהינו ב canny בתור edge

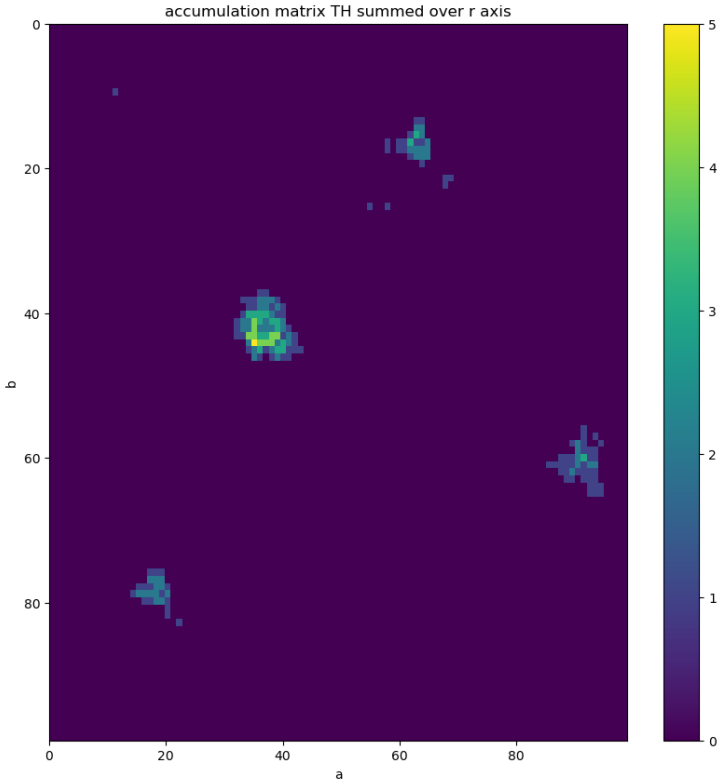
עבור כל נקודה נקדם במטריצת האקומולציה את כל המעגלים שהיא היתה יכולה להיות חלק מהם,תלוי בטווח הרדיוס שבחרנו, כלומר אם נקודה במטריצה רחוקה מהרדיוס המקסימלי לא נרצה לצייר מעגל שמרכזו באותה נקודה. r_ind יחזיר לנו את האינדקס לרדיוס הקרוב ביותר במטריצת האקומולציה שמתאים למרחק בין ה edge לאותה נקודה(r0).

```
edge_inds = np.argwhere(mag_im > 0)

# run on all a,b and edge indices and find corresponding R
for yx in edge_inds:
    x = yx[1]
    y = yx[0]

    for a_ind, a0 in enumerate(a):
        for b_ind, b0 in enumerate(b):
            # TODO: find best corresponding r0 (1 line)
            r0 = np.sqrt((a0-x)**2 + (b0-y)**2)
            # something to make it faster
            if r0 > rmax:
                continue

            #TODO: find best index in r dimension (1 line)
            r_ind = np.argmin(np.abs(r - r0))

            # TODO: update accumulation matrix (1 line)
            acc_mat[b_ind, a_ind,r_ind] +=  1
```

נצייר את מטריצת האקומולציה, בגלל שמטריצת האקומלציה תלת-מימדית נחשב עבור נקודה בתמונה את הסכום של כל המונים בציר של הרדיוס ונקבל תמונה דו-ממדית שהpeakים הם הנקודות הצהובות.



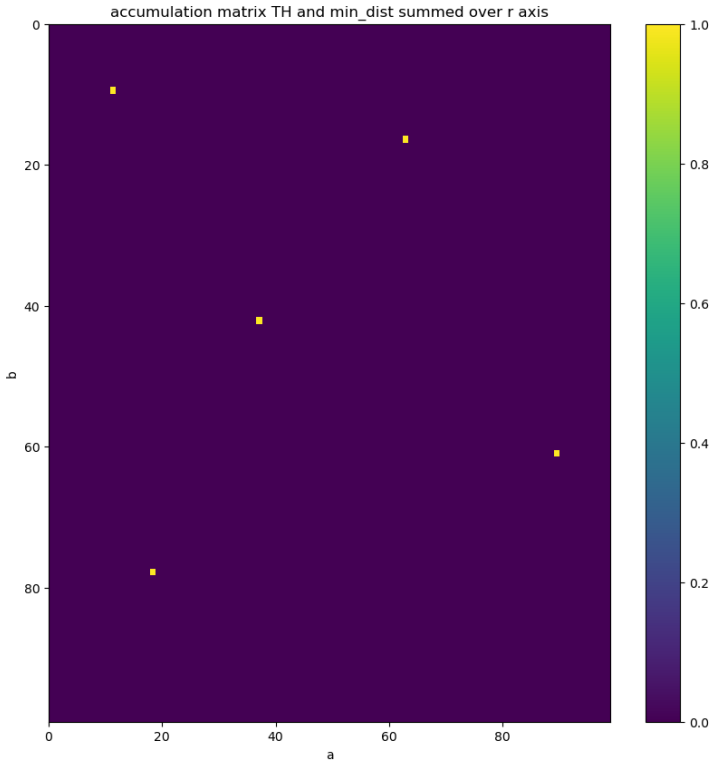
אחרי Threshold על התמונה TH=25 (Unchanged)



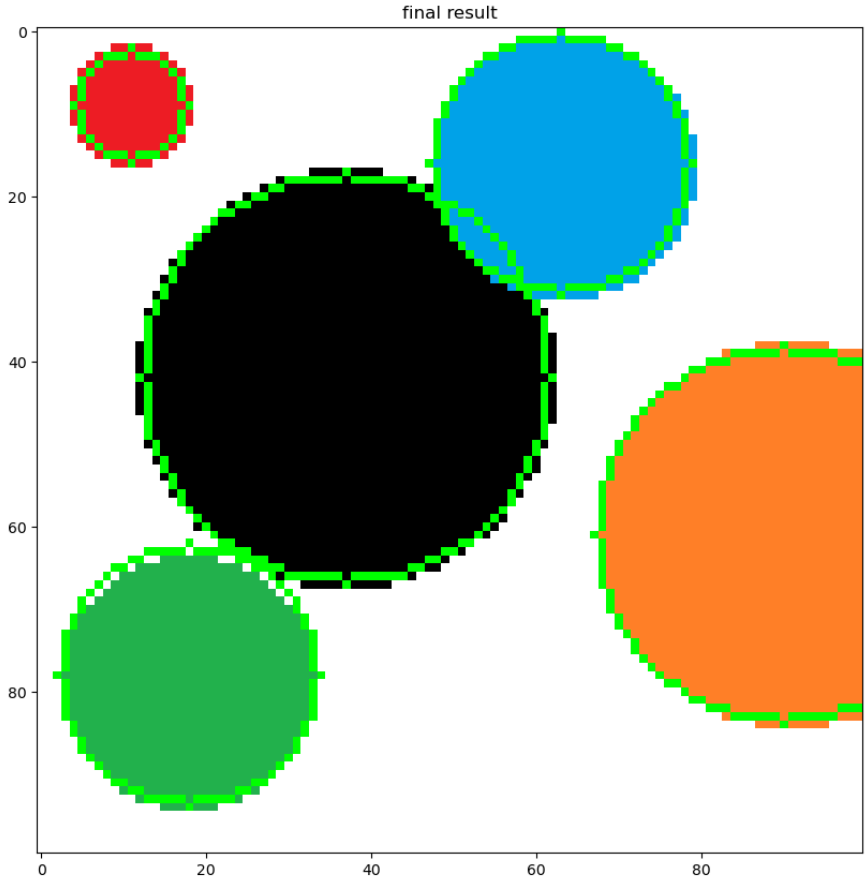
מחיקת שכנים והשארת רק הנקודות במטריצת האקומולציה עם ערך מקסימלי

```
min_dist = 15
acc_mat_th_dist = acc_mat_th.copy()
# run on all above TH bins
for i in range(edge_inds.shape[0]):
    b0, a0, r0 = edge_inds[i]
    # search in all other above TH bins
    for j in range(i+1, edge_inds.shape[0]):
        b1, a1, r1 = edge_inds[j]

        # if the two above are neighbors (below min_dist) - delete the less important
        if ((r0-r1)*r_step)**2+((a0-a1)*a_step)**2+((b0-b1)*b_step)**2 < min_dist**2:
            if acc_mat[b0, a0, r0] >= acc_mat[b1, a1, r1]:
                acc_mat_th_dist[b1, a1, r1] = 0

            else:
                acc_mat_th_dist[b0, a0, r0] = 0
```

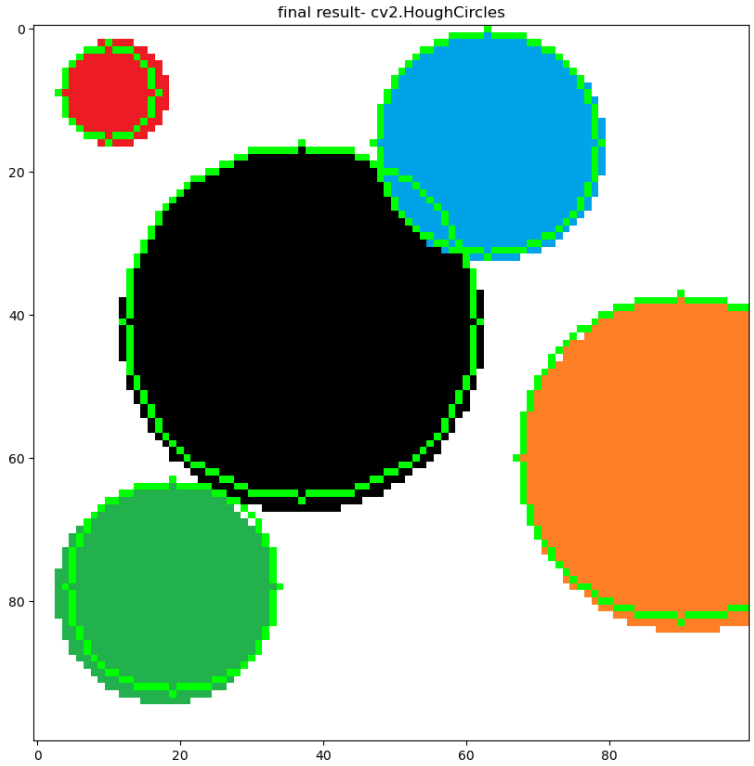
לאחר מחיקת שכנים



נצייר את המעגלים על גבי התמונה המקורית



CV2 Result Comparison



Coin Detection Ex2 – Using Cv2 Library

Parameters Chosen for CV2 Hough Circles:

```
# TODO: fill in the best values possible
# to detect the right circle dimeter and place
acc_ratio = 1
min_dist = 25
canny_upper_th = 150
acc_th = 35
circles = cv2.HoughCircles(im, cv2.HOUGH_GRADIENT, acc_ratio,
                           min_dist, param1=canny_upper_th,
                           param2=acc_th, minRadius=40, maxRadius=70)
```

In the image we have 3 types of American coins:

1. Quarter – 25 cents (the biggest)
2. Nickel – 5 cents (2nd biggest)
3. Dime – 10 cents (but smaller than the nickel)

The coin radius is measured in pixels and after some experimentations I found the following bounds that we'll detect with very good accuracy the coin values.

```
res = cv2.circle(res, (a, b), r, (0, 255, 0), 3)
    right_padding = 10
    bottomLeftCornerOfText = (int(a-right_padding), int(b))
    if 63 <= r:
      cv2.putText(res,"quarter",bottomLeftCornerOfText,
font,fontScale,(255,255,255),lineType)
    elif 53 <= r <= 62:
      cv2.putText(res,"nickel",bottomLeftCornerOfText, font,fontScale,fontColor,lineType)
    elif  40 <=  r <= 52:
      cv2.putText(res,"dime",bottomLeftCornerOfText, font,fontScale,(255,255,255),lineType)
```

Final Result