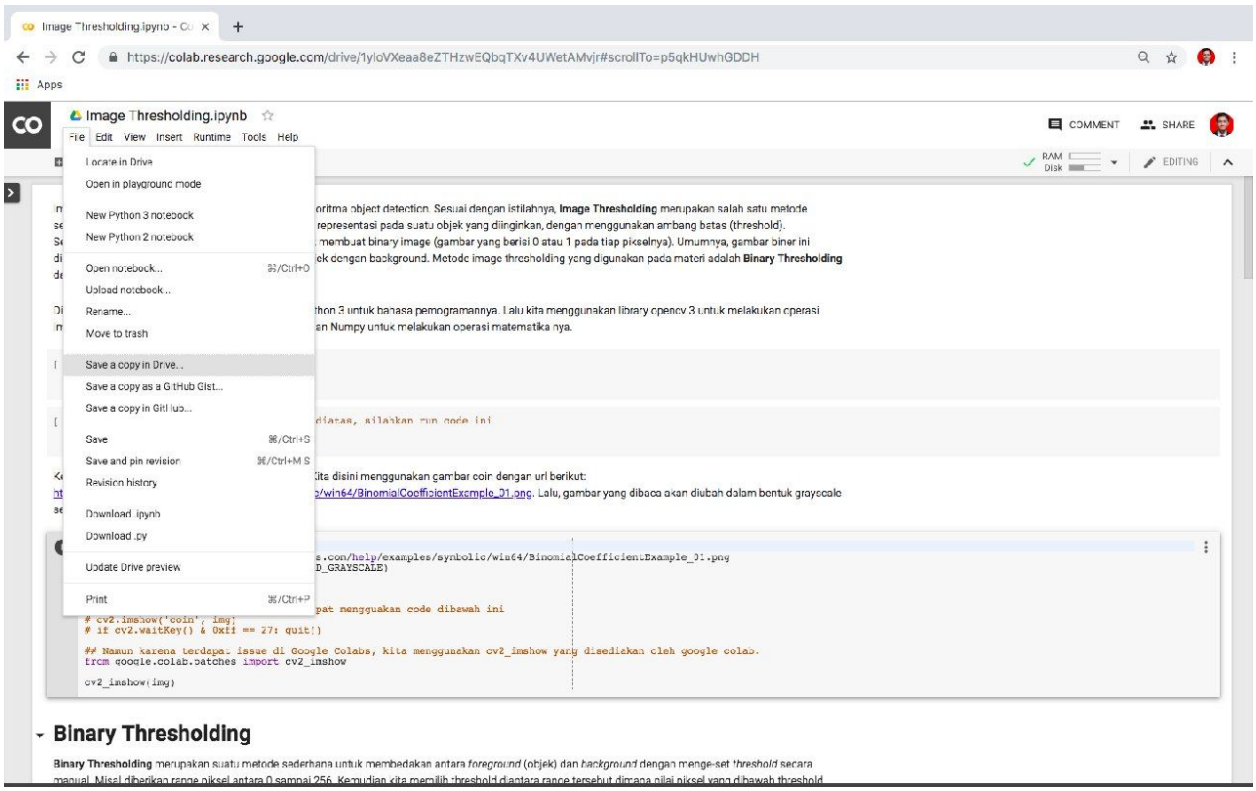
Sebelumnya say ucapkan terimakasih kepada kak Egi dan teman-teman DSI yang mempersilahkan saya untuk memaparkan materi Image Thresholding. Jadi materi Image Thresholding ini akan membahas dua metode, yakni:

- Binary Thresholding
- Otsu Thresholding

https://colab.research.google.com/drive/1yloVXeaa8eZTHzwEQbqTXv4UWetAMvjr

Mungkin temen-temen bisa membuka link tersebut dan langsung mencoba-coba disana. Tapi sebelumnya, dimohon untuk meng-kopi google colab tersebut di drive atau github masing-masing. Karena nnati bakalan diminta akses bila langsung mencoba di link yang saya kasih tersebut.



START

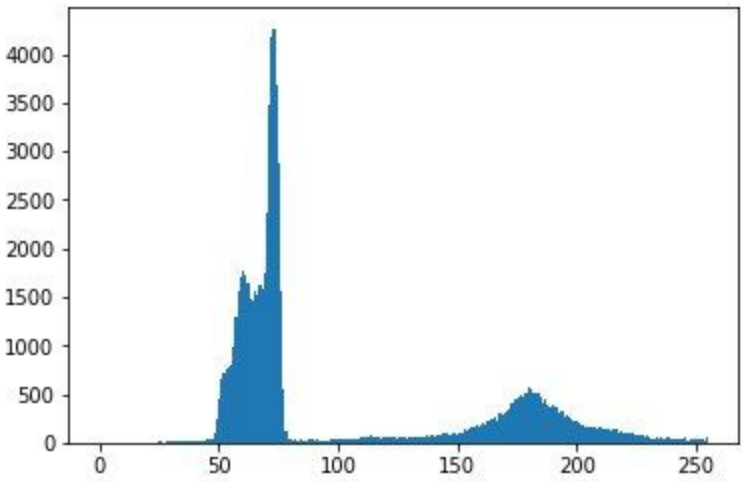
Image Thresholding merupakan salah satu metode sederhana untuk segmentasi gambar, membuat suatu representasi pada suatu objek yang diinginkan, dengan menggunakan ambang batas (threshold). Sederhananya, Image Thresholding ini bertujuan untuk membuat image segmentation berupa binary image (gambar yang berisi 0 atau 1 pada tiap pikselnya). Umumnya, gambar ini digunakan untuk membedakan antara background dan foreground (objek). Image segmentation yang telah dibuat ini, nantinya akan digunakan ke proses selanjutnya. Salah satu proses tersebut adalah digunakan untuk Deteksi Objek. Di jupyter notebook yang saya berikan, saya menggunakan bahasa pemrograman python versi 3 dan

library computer vision opencv versi 3. Mungkin yang udah bisa run, bisa dicoba-coba saja sambil melihat pemaparan yang saya sampaikan ini.

Seperti yang saya sebutkan sebelumnya, disini kita menggunakan dua metode. yaitu binary thresholding dan otsu thresholding. Binary Thresholding merupakan suatu metode sederhana untuk membedakan antara foreground (objek) dan background dengan menge-set threshold secara manual. Misal diberikan range piksel antara 0 sampai 256 pada suatu image. Kemudian kita memilih threshold diantara range tersebut dimana nilai piksel yang dibawah threshold dianggap background dan nilai piksel diatas threshold dianggap sebagai objek atau foreground.

Ide penggunaan thresholding ini dapat dilihat pada distribusi nilai piksel pada suatu image.

Disini saya menggunakan contoh gambar berikut. saya kasih nama coin.png. Image ini saya convert ke bentuk gray dimana gmabar ini nantinya memiliki satu layer saja. berbeda dengan gambar berwarna yang terdiri dari 3 layer RGB. Saya ambil distribusi piksel yang ada di gambar ini menggunakan histogram.



Sumbu x melambangkan piksel (o sampai 255). Sumbu y memberikan info banyaknya suatu intensitas piksel pada suatu image. Dapat dilihat di histogram diatas, terdapat dua puncak yang berbeda. Distribusi ini disebut distribusi bi-modal. Task metode ini adalah memisahkan antara wilayah background dan foreground menggunakan threshold yang kita tentukan sendiri. Dapat dilihat di histogram diatas, kita dapat menentukan threshold yang dapat memsahkan dua modal tersebut. saya ambil contoh menggunakan threshold sebesar 130.

```
# Thresholding function
def get_binary_image(img, threshold):
  img_binary = copy.copy(img)
  img_binary[img_binary > threshold] = np.amax(img_binary)
  img_binary[img_binary <= threshold] = 0
  return img_binary
```

Berikut algoritma sederhana yang bisa dipakai, bila dibawah threshold kita set 0, sisanya kita set nilai maksimum piksel, yaitu 255

Image asli

Image segmentasi

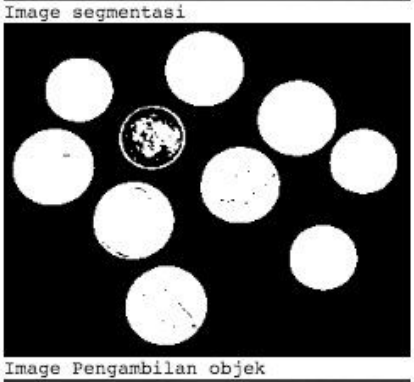
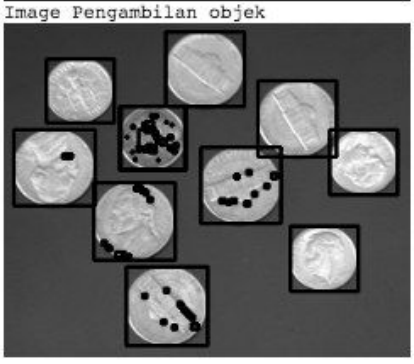


Image Pengambilan objek



dan berikut adalah hasil thresholding tersbut. gambar kedua merupakan gambar hasil segmentasi, lalu gambar ketiga saya lakukan pengambilan contour untuk mendeteksi lokasi objek yang saya inginkan. Untuk penjelasan pengambilan contour mungkin akan dibahas di lain waktu. Algoritma yang saya berikan diatas itu merupakan Scratch code. ibrary opencv sendiri memberikan fungsi threshold ini sebagai berikut.

```
threshold = 130
max_value = np.amax(img)
threshold_type = cv2.THRESH_BINARY
(th_binary_cv2, img_binary_cv2) = cv2.threshold(img, threshold, max_value, threshold_type)
```

dan didapatkan hasil yang sama dengan code saya yang from scratch.

Image asli

Image segmentasi

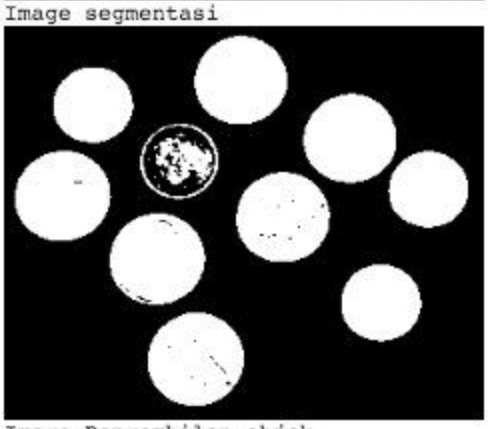
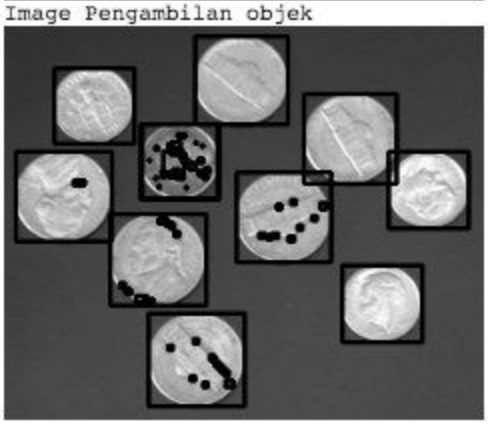


Image Pengambilan objek



Metode Binary Thresholding mengharuskan kita untuk menge-set threshold secara manual. Bisa saja, threshold yang kita pilih masih kurang representatif. Munculah suatu metode Otsu Thresholding, yang diambil dari nama penciptanya, Nobuyuki Otsu (大津展之 Ōtsu Nobuyuki), dimana pemilihan threshold ini dapat dilakukan secara *otomatis*. Untuk mendapatkan threshold yang optimal, metode Otsu ini mencari threshold yang meminimalkan varians intra-kelas (varians dalam kelas), yang didefinisikan sebagai jumlah variasi varian dari dua kelas (background dan objek):
Didenotasikan sebagai berikut

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Weight ω_0 dan ω_1 merupakan probabilitas dari dua kelas 0 dan 1 yang terpisahkan oleh threshold t. Lalu, σ_0^2 dan σ_1^2 merupakan varians dari kedua kelas.

untuk penjelasan variabelnya saya kasih dapam bentuk image dikarenakan ada problem ketika mengopi variabel tersebut.

Probabilitas kedua kelas $\omega_{0,1}(t)$ didapatkan menggunakan histogram berukuran L *bins* dengan rumus berikut:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

Metode ini menunjukkan bahwa meminimalkan varians intra-kelas sama dengan meningkatkan varians inter-kelas (varians luar kelas) dengan rumus berikut:

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned}$$

ω menunjukkan probabilitas kelas dan μ menunjukkan rata-rata (means) kelas, dimana $\mu_0(t)$, $\mu_1(t)$, dan μ_T dirumusakan sebagai berikut:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

Dari rumus-rumus diatas, didapatkan suatu relasi sebagai berikut:

$$\begin{aligned}\omega_0\mu_0 + \omega_1\mu_1 &= \mu_T \\ \omega_0 + \omega_1 &= 1\end{aligned}$$

Berikut penjelasan semua rumus yang digunakan pada metode ini. Mohon maaf menggunakan image karena ada problem ketika mengopi variabelnya. Dari rumus tersebut, didapatkan Algoritma untuk menjalankan metode ini.

Algoritma Otsu

Algoritma dari metode ini dapat dipaparkan sebagai berikut:

1. Hitung histogram dan probabilitas setiap tingkat intensitas piksel
2. Set inisialisasi $\omega_i(0)$ dan $\mu_i(0)$
3. Lakukan iterasi pada setiap kemungkinan *threshold* $t = 1, \cdots$ maksimum intensitas piksel
    1. Update nilai ω_i dan μ_i
    2. Hitung nilai $\sigma_b^2(t)$
4. *Threshold* yang diinginkan didapatkan dari nilai maksimum dari $\sigma_b^2(t)$

Dari algoritma ini, saya buat code pythonnya sebagai berikut.

```
[ ]  # Otsu Thresholding from Scratch

     def get_otsu_threshold(img):
       hist = cv2.calcHist([img],[0],None,[256],[0,255])
       total = sum(hist)
       hist_prob = hist/total

       wB = 0
       sumB = 0
       thres_otsu = 0
       max_val = 0
       sum_tot = np.dot(np.arange(0,256),hist_prob)
       for i in range(len(hist_prob)):
         wF = 1 - wB
         if wF > 0 and wB > 0:
           mF = (sum_tot - sumB) / wF
           val = wB * wF * (sumB / wB - mF) * (sumB / wB - mF)
           if val >= max_val:
             thres_otsu = i
             max_val = val
         wB = wB + hist_prob[i]
         sumB = sumB + i*hist_prob[i]
       return thres_otsu
```

Disini saya lebih memilih untuk meng-update Summation dari tiap kelas (Instead of Mean), karena ada problem ketika weight bernilai nol. Saya langsung run program tersebut, dan didapatkan nilai threshold 126 pada image coin.png.
hasil nya sebagai berikut

Threshold hasil metode Otsu: 126
Image asli

Image segmentasi

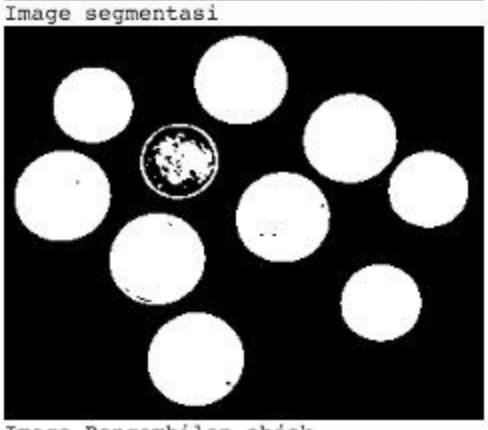
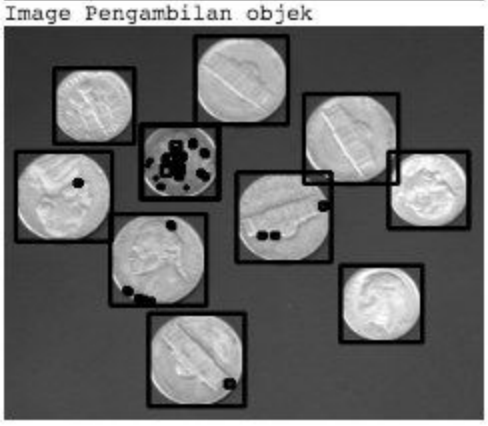


Image Pengambilan objek

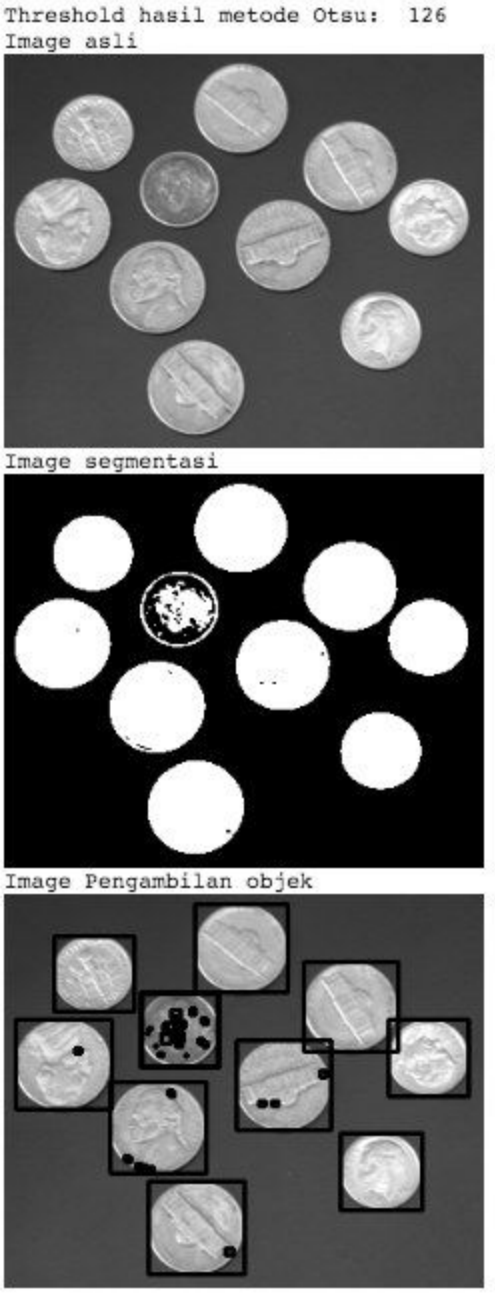


Opencv memiliki fitur untuk mendapatkan threshold sekaligus image segmentationnya menggunakan fitur cv2.threshold. sama seperti binary threshold, cuma yang membedakan adalah parameter type thresholdingnya.

```
max_value = np.amax(img)
threshold_type = cv2.THRESH_OTSU

(th_otsu_cv2, img_otsu_cv2) = cv2.threshold(img, 0, max_value, threshold_type)
```

typenya menggunakan CV2.THRESH_OTSU



Didapatkan hasil segmentasi yang sama seperti gambar diatas, berhubung sudah 16.45, saya akhiri dulu dengan kelemahan dari metode Image thresholding ini. Metode ini memiliki kelemahan, yaitu:

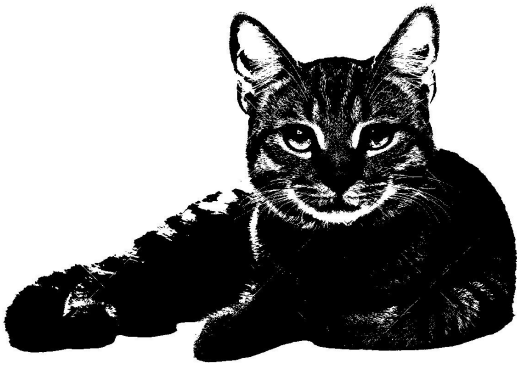
1) Image yang digunakan harus memiliki perbedaanyang jelas antara backgroud dan objek.
2) Objek yang kecil akan sulit tersegmentasi dengan metode ini

3) Noise pada image dapat mempengaruhi hasil segmentasi yang nanti nya akan berpengaruh pada hasil deteksi juga.

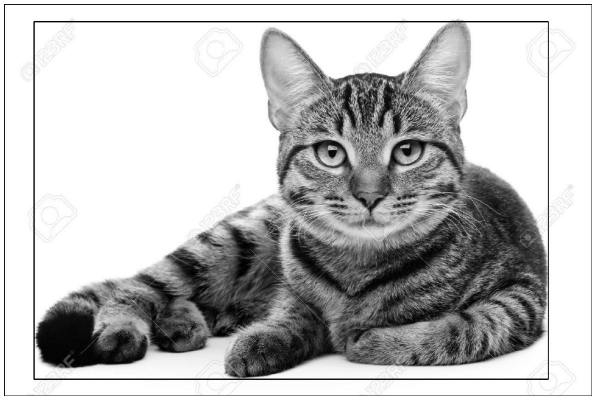
Sesi tanya jawab

Dari indra :
Thanks zok pembahasan materinya, mau nanya kalo penggunaan thresholding ini bisa ningkatin performance sbg pre processing misal di case object detection/classification?
Jawaban :
yup bisa banget. Selama ini mungkin temen temen langsung input image raw ke image classification atau object detection. Mungkin dengan input yang sudah diubah menggunakan image thresholding ini, bisa meningkatkan akurasi
Contoh gambar lain, saya menggunakan image ini,

Berikut hasil segmentasinya



Dan ini hasil object detectionnya dengan beberapa pembersihan area contour,



Soure : kulgram DSI