Ahmad Maulvi Alfansuri

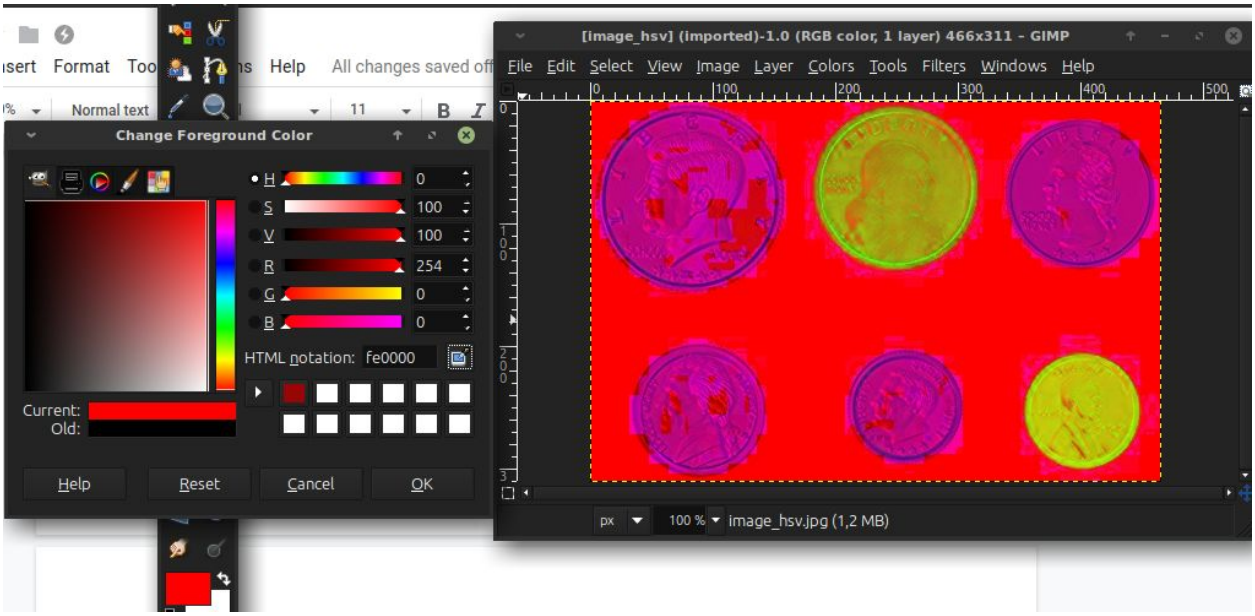
G64160081

Soal :

Diberikan gambar coin.jpg

Seleksi gambar tersebut hanya pada bagian koinnya saja.

Algoritma yang saya gunakan adalah melakukan conversi gambar hsv. Lalu melakukan seleksi dengan bound berdasarkan warna background yang saya dapatkan dari color picker di GIMP.



Hasil pick color bound pada GIMP

Setelah didapat kan masking filter. Masking filter masih terdapat banyak noise. Untuk menghilangkan noise digunakan medianfilter. Setelah itu dilakukan erosi dan dilasi dengan menyesuaikan iterasi agar didapat ukuran seleksi yang pas pada koin.

Jawaban

1. Import setiap library yang dibutuhkan

```
import numpy as np
import cv2

# bit.ly/LKP8-PCD
# 272822. Palette monokai hilite.me
```

2. Kodingan dari LKP6.

```
# Pinjeman dari LKP 6
# Buat dapetin fungsi hsv

# fungsi conversi pixel rgb menjadi hsv
def conv_pixel_bgr_to_hsv(pixel):
    # buat pixel
```

```
    new_pixel = np.zeros((3), np.uint32)
    # ambil nilai v
    v = max(pixel)
    # jadikan nilai v integer
    v = int(v)

    # inisiasi s dan h
    s = 0
    h = 0
    # jika v == 0. s = 0. agar tidak terjadi error pembagian terhadap 0
    if(v != 0):
        s = (v - min(pixel)) * 1.0 / v

    # ekstrak b g r
    b, g, r = pixel

    # jadikan b g r integer
    b = int(b)
    g = int(g)
    r = int(r)
    # jika v == min(pixel), agar v - min(pixel) tidak 0 dan menghindari pembagian
terhadap 0
    if(v == min(pixel)):
        h == 0
    else:
        # rumus mendapatkan nilai h
        if(v == r):
            h =  (60 * (g - b))/(v - min(pixel))
        elif(v == g):
            h = 120 + (60 * (b - r))/(v - min(pixel))
        elif(v == b):
            h = 240 + ((60 * (r - g))/(v - min(pixel)))
        if(h < 0):
            h += 360

    # set normalisasi nilai h, s, v
    v = int(v)
    # 0 < s < 1. Kalikan 255 agar menjadi nilai 0 < s < 255
    s = int(255 * s)
    # karena maks(h) = 360. Maka bagi h dengan 2 agar menjadi < 255
    h = int(h / 2)
    # set pixel baru
    new_pixel = np.array([h,s,v])
    # return nilai pixel baru
    return new_pixel

# fungsi conv_image_bgr_to_hsv akan melakukan convert image dengan format bgr
```

```
menjadi hsv
def conv_image_bgr_to_hsv(image):
    # ambil dimensi
    (row, col, chan) = image.shape
    # buat canvas
    new_image = np.zeros((row,col,3), np.uint8)
    # loop
    for y in range(row):
        for x in range(col):
            # set pixel pada canvas baru
            new_image[y, x] = conv_pixel_bgr_to_hsv(image[y, x])
    # kembalikan image baru
    return new_image

# fungsi show akan menampilkan gambar dan menulis pada disk gambar tersebut seusai
judul yang diberikan
def show(image, title):
    # tulis ke disk
    cv2.imwrite(title, image)
    # tampilkan gambar
    cv2.imshow(title, image)

# Fungsi masking akan menselect gambar yang dipilih berdasarkan bound warna yang
diberikan.
# Fungsi akan mengembalikan image mask, dengan warna putih sebagai pixel yang
diseleksi
def masking(image, lower_color, upper_color):
    # ambil dimensi
    (row, col, chan) = image.shape
    # buat mask dengan satu channel
    masker = np.zeros((row,col,1), np.uint8)
    # loop
    for y in range(row):
        for x in range(col):
            # select pixel dengan bound yang diberikan
            if( np.all(lower_color <= image[y, x]) and \
                np.all(upper_color >= image[y, x]) ) :
                # warnai pixel dengan warna putih
                masker[y,x,0] = 0
            else:
                masker[y,x,0] = 255

    # kemblaikan image mask
    return masker

# Fungsi select_image akan mengambil pixel image dimana pixel tersebut diselect
oleh image mask.
```

```
def select_image(image, masker):
    # ambil dimensi
    (row, col, chan) = image.shape
    # buat canvas untuk gambar baru
    new_image = np.zeros((row,col,3), np.uint8)
    for y in range(row):
        for x in range(col):
            # jika mask berwarna putih
            if(np.all(masker[y,x] == 255)):
                # ambil pixel image. Warnai image baru dengan pixel tersebut
                new_image[y,x] = image[y,x]
    # kembalikan image
    return new_image
```

3. Fungsi erosi

```
# Fungsi erosi
def ero(img, kernel):
    # ambil dimensi
    row, col, _ = img.shape
    # buat canvas
    canvas = np.zeros((row, col, 1), np.uint8)
    # looping matrix pixel
    for i in range(0, row):
        for j in range(0, col):
            hasil = 0

            # Mengecek edge. Apakah cukup untuk memasukkan kernel n x n
            if (i - kernel // 2 < 0) or (i + kernel // 2 > row - 1) or (j - kernel
// 2) < 0 or (
                    j + kernel // 2 > col - 1):
                continue

            # Loop submatrix n x n pada matrix
            for ii in range(i - kernel // 2, i + kernel // 2 + 1):
                for jj in range(j - kernel // 2, j + kernel // 2 + 1):
                    # cek pixel apakah pixel apakah pixel putih
                    if img[ii][jj] > 0:
                        # jika iya. jumlahkan hasil. hasil akan digunakan untuk
melakukan pengecekan
                        # apakah pixel tersebut fit, hit, atau tidak keduanya
                        hasil += 1

            # cek fit. Fit didapatkan apabila hasil akan sama dengan ukuran kernel
(structure).
```

```
            # Jika pixel fit warnai pixel dengan putih (select)
            if (hasil == kernel * kernel):
                canvas.itemset((i, j, 0), 255)
    return canvas
```

4. Fungsi dilasi

```
def dil(img, kernel):
    # ambil dimensi
    row, col, _ = img.shape
    # buat canvas
    canvas = np.zeros((row, col, 1), np.uint8)
    # loop matrix pixel
    for i in range(0, row):
        for j in range(0, col):
            hasil = 0
            # mengecek ujung edge. Apakah
            if (i - kernel // 2 < 0) or (i + kernel // 2 > row - 1) or (j - kernel
// 2) < 0 or (
                    j + kernel // 2 > col - 1):
                continue

            # Loop submatrix n x n dari gambar.
            for ii in range(i - kernel // 2, i + kernel // 2 + 1):
                for jj in range(j - kernel // 2, j + kernel // 2 + 1):
                    # cek apakah pixel berwarna putih
                    if img[ii][jj] > 0:
                        # jika iya. jumlahkan hasil. hasil akan digunakan untuk
melakukan pengecekan
                        # apakah pixel tersebut fit, hit, atau tidak keduanya
                        hasil += 1

            # Jika hasil > 0. Maka pixel fit. Sehingga gambar akan mengalami
perluasan (dilasi)
            if (hasil > 0):
                canvas.itemset((i, j, 0), 255)
    return canvas
```

5. Fungsi untuk menampilkan gambar dan mensave gambar

```
def show(image, title):
    # tulis ke disk
    cv2.imwrite(title, image)
```

```
    # tampilkan gambar
    cv2.imshow(title, image)
```

6. Fungsi untuk meload gambar

```
def load_image(filename="soal.png"):
    image = cv2.imread(filename)
    return image
```

7. Fungsi utama. Dimasukkan fungsi main agar fungsi menjadi modular

```
def main():
    # load image pada soal
    image = load_image('coin.jpg')
    # convert gambar ke hsv
    image_hsv = conv_image_bgr_to_hsv(image)
    show(image_hsv, "image_hsv.jpg")

    # mask bagian background.
    # nilai bound didapatkan dengan color picker di background
    mask = masking(image_hsv, np.array([0x00, 0x00, 0xfe]), np.array([0xff, 0xff,
0xff]))
    # tampilkan dan save hasil masking
    show(mask, "mask.jpg")

    # Lakukan medianBlur. Untuk menghilangkan bercak putih.
    # Digunakan library agar kodingan tidak terlalu panjang
    mask_smooting = cv2.medianBlur(mask, 9)
    # tampilkan dan save hasil smoothing median bloor
    show(mask_smooting, "smooth.jpg")

    # Hasil seleksi. Dengan mask hasil smoothing sementara
    selected = select_image(image, mask_smooting)
    show(selected, "hasil gambar pertama.jpg")

    # kernel untuk melakukan morphing
    kernel5 = np.ones((5,5), np.uint8)

    # auto version
    # karena hasil seleksi masih terlalu luas lakukan erosi dan dilasi
    # dengan jumlah erosi lebih banyak untuk mengecilkan area seleksi
    erode = cv2.erode(mask_smooting, kernel5, iterations = 3)
    improved_mask = cv2.dilate(erode, kernel5, iterations = 2)
```

```
    # tampilkan hasil improved masking
    show(improved_mask, "improved_mask.jpg")
    # tampilkan hasil seleksi gambar dari improved masking
    improved_selected = select_image(image, improved_mask)
    show(improved_selected, "improved selected.jpg")

    # for i in range(0,16):
    #     b = dil(b,3)

    # for i in range(0,16):
    #     b = ero(b,3)


    cv2.waitKey(0)

# Fungsi main yang menjalankan program
main()
```

Gambar asli koin

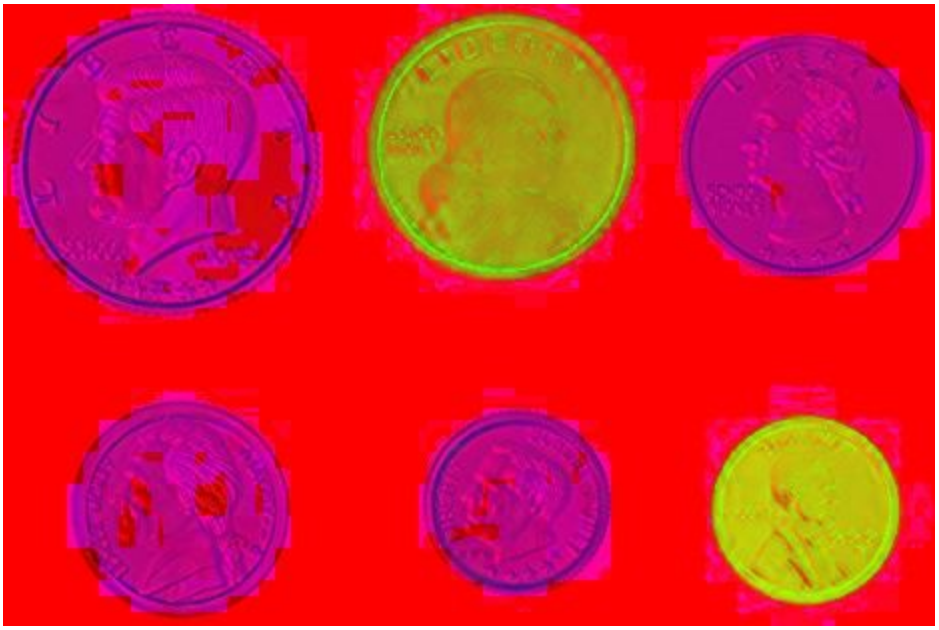
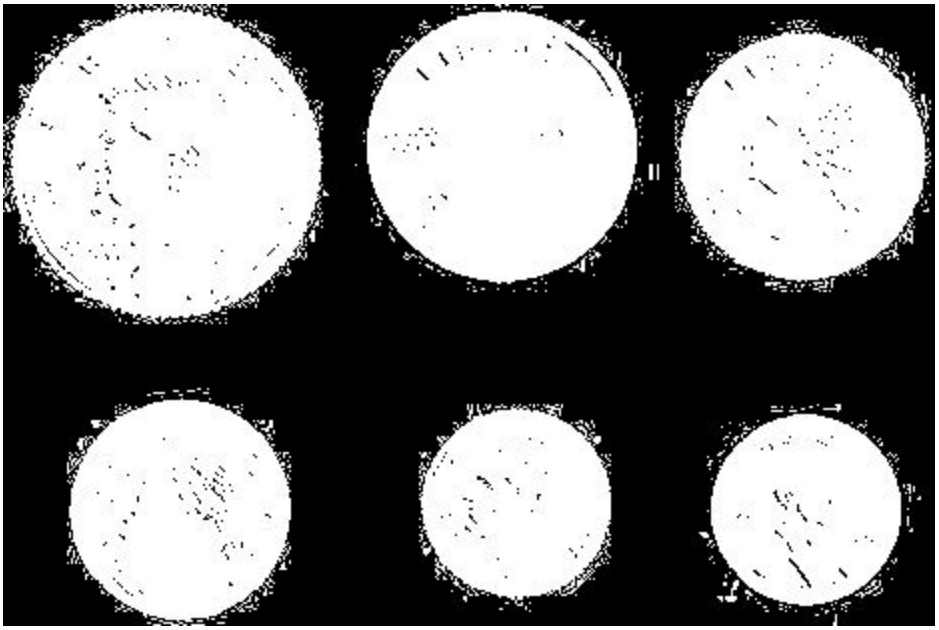
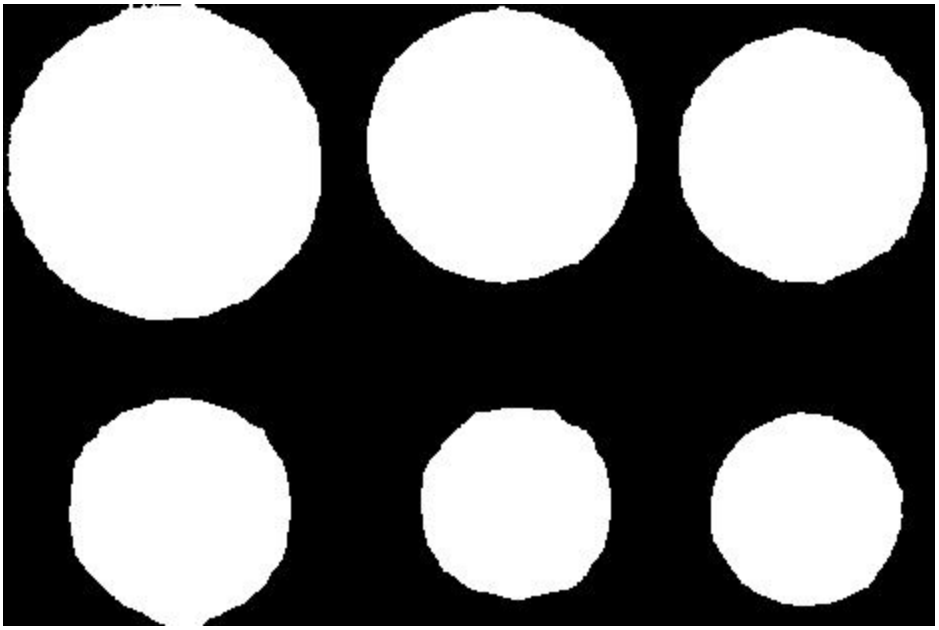


Image hsv

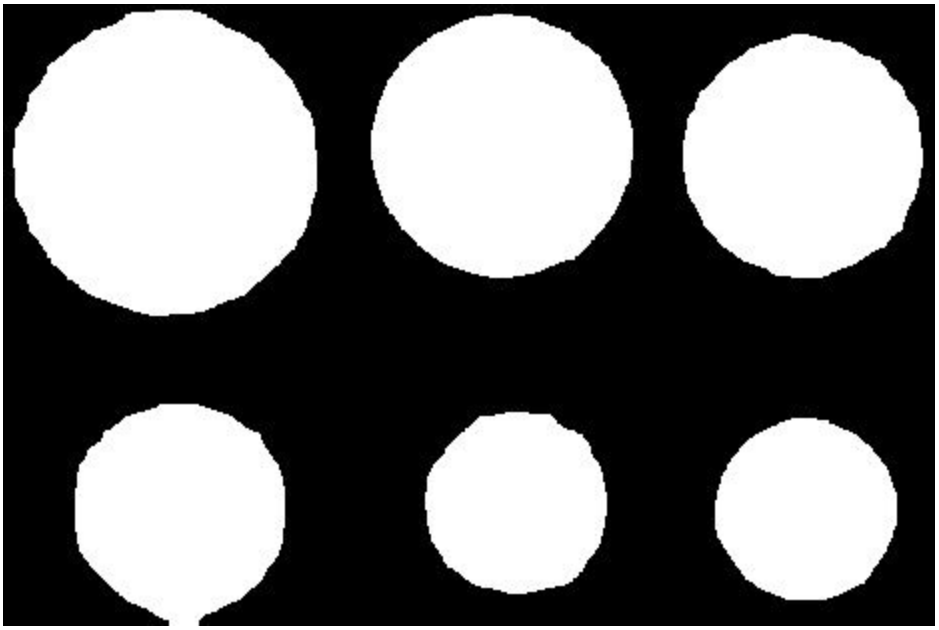


Hasil seleksi dari color bound pada hsv



Hasil smoothing dengan median blur

Hasil Seleksi dari masking yang sudah di smoothing



Masking yang sudah diimprove dengan erosi dan dilasi

Hasil masking dari improved masked