LAPORAN TUGAS BESAR
PENGOLAHAN CITRA DIGITAL - B

Dosen : Muhammad Fachrie, S.T., M.Cs.

Disusun Oleh :

Satriya Adhitama	5200411545
M Fahlevi A	5200411504
Nurcholis Majid	5200411090

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA

2021/2022

PENDAHULUAN

1.1 Latar Belakang

Pengolahan citra saat ini menjadi hal yang cukup penting untuk dapat diterapkan pada berbagai teknologi yang ada. Keterdukungan perangkat teknologi terhadap kemampuan untuk dapat melakukan akuisisi hingga pengolahan citra merupakan salah satu aspek untuk mendukung kemutakhiran dan ketergunaan perangkat. Kita bisa dengan mudah mendapati banyak dari perangkat elektronik yang dimiliki menggunakan pengolahan citra sebagai salah satu fitur maupun kinerja utama dari perangkat, seperti kamera, *smartphone*, laptop, *scanner*, proyektor, dan lain sebagainya.

Citra atau gambar digital ini merupakan hasil akuisisi cahaya dari lingkungan berisi kumpulan warna sebagai representasi dari dunia nyata. Sumber dari citra tidak hanya dari dunia nyata, namun bisa didapatkan sebagai hasil dari pemrosesan citra oleh perangkat lunak. Citra yang terdiri dari kumpulan warna dalam satuan pixel ini dapat digunakan untuk berbagai kebutuhan. Citra dapat diolah menggunakan metode atau cara tertentu agar dapat diubah sedemikian rupa sehingga dapat menghasilkan keluaran citra sesuai dengan yang diinginkan.

Pada penerapannya, citra dapat dilakukan klasifikasi berdasarkan variabel atau pendekatan tertentu sehingga kumpulan dari beberapa citra dapat dikelaskan sesuai dengan yang diinginkan. Klasifikasi ini dapat dilakukan menggunakan pengolahan citra untuk memperoleh pola pada pixel citra. Pola ini dapat berupa warna dominan yang muncul, tekstur gambar, text terdapat pada gambar, terang dan gelapnya warna, dan lain sebagainya.

Pada penelitian ini, akan mengimplementasikan tentang pengolahan citra digital tentang segmentasi dan pencerminan secara horizontal pada objek. Objek yang digunakan adalah gambar orang dengan background berwarna hijau. Sehingga, hasil yang diharapkan adalah mesin dapat mengganti background berwarna hijau dengan background yang diinginkan.

1.2 Identifikasi Masalah

- ➔ Bagaimana cara pemrosesan citra digital untuk dapat melakukan penghapusan background green screen;
- ➔ Bagaimana cara pemrosesan citra digital untuk dapat melakukan pencerminan secara horizontal;
- ➔ Pola citra apa yang digunakan untuk dapat menjadi acuan penghapusan background;
- ➔ Bagaimanakah implementasi pemrograman untuk dapat menghapus background green screen dan pencerminan horizontal pada citra secara bersamaan.

1.3 Tujuan

- ➔ Melakukan segmentasi citra digital pada gambar dengan background berwarna hijau;
- ➔ Melakukan penskalaan dan pencerminan citra digital pada gambar setelah segmentasi;
- ➔ Mengetahui proses dan algoritma untuk dapat melakukan segmentasi citra digital;

KAJIAN PUSTAKA

2.1 Citra Digital

Citra digital dapat diartikan sebagai representasi diskrit dari informasi pemrosesan data secara spasial (*layout*) dan intensitas (*colour*) . Citra juga merupakan sinyal multidimensi [1]. *Layout* merupakan tata letak/susunan dari elemen visual dalam sebuah citra. Warna berisi informasi komposisi pembentukan satuan terkecil pada citra digital, yaitu pixel dengan susunan biner. RGB merupakan salah satu komposisi warna yang sering digunakan, dengan merepresentasikan tiap pixel dengan angka biner untuk membentuk chanel warna merah, hijau, dan biru. Channel warna ini kemudian akan ditumpuk, sehingga mendapatkan hasil citra dengan susunan warna pixel yang sesuai.

2.2 Image Manipulation

Manipulasi citra dilakukan dengan memanipulasi sinyal multidimensi dengan sistem yang dapat dilakukan pada sirkuit digital sederhana hingga lanjutan pada komputer paralel. Tujuan dari manipulasi ini dibagi menjadi 3, yaitu [2]:

- ***Image Processing***

  Manipulasi citra berupa melakukan perubahan susunan warna maupun komponen lainnya pada citra orisinil sehingga menjadi keluaran citra baru yang berbeda.

- ***Image Analysis***

  Manipulasi citra dengan tujuan mengukur variabel tertentu yang terdapat pada citra.

- ***Image Understanding***

  Manipulasi citra untuk memahami keseluruhan komponen hingga pola yang muncul pada citra. Hasil dari pemahaman ini adalah mampu mendeskripsikan citra sesuai dengan variabel yang ada.

2.3 Image Segmentation

Segmentasi citra merupakan salah satu topik penting ilmu komputer terutama dalam bidang pengolahan citra digital dan visi komputer. Tujuan segmentasi citra adalah untuk mempartisi gambar menjadi beberapa wilayah yang tidak tumpang tindih dengan karakteristik yang homogen, seperti intensitas, warna, dan tekstur. Seiring dengan berkembangnya teknologi pada aplikasi yang memproses sebuah objek seperti rekonstruksi tiga dimensi, pengenalan benda, pengenalan tulisan, deteksi wajah, pengkodean obyek dan lain-lain maka proses segmentasi menjadi semakin diperlukan. Hasil segmentasi juga harus semakin akurat karena ketidak akuratan hasil segmentasi akan mempengaruhi pula hasil proses selanjutnya.

2.4 Operasi Geometri

Operasi geometri adalah proses perubahan hubungan spasial antara setiap pixel pada sebuah citra. Operasi geometri memetakan kembali pixel citra input dari posisi awal (x1,y1) ke posisi baru (x2,y2) pada citra output. Proses operasi geometri yang terdapat pada studi kasus ini.

- ***Image Flipping***

  Refleksi atau pencerminan adalah proses pengolahan citra secara geometri dengan memindahkan nilai-nilai pixel pada posisi awal (x1,y1) menuju ke posisi baru di (x2,y2) pada citra output sesuai dengan posisi pencerminan. Posisi pencerminan ada tiga jenis yaitu pencerminan terhadap sumbu x, pencerminan terhadap sumbu y, dan pencerminan terhadap sumbu x dan y.

- ***Image Scaling***

  Penskalaan adalah sebuah operasi geometri yang memberikan efek memperbesar atau memperkecil ukuran citra input sesuai dengan variabel penskalaan citranya. Ukuran baru hasil penskalaan didapat melalui perkalian antara ukuran citra input dengan variabel penskalaan.

METODOLOGI

3.1 Sample Data

Untuk segmentasi, penskalaan, dan pencerminan citra yang dilakukan akan menggunakan sampel data berupa gambar sebagai berikut :


green-screen


green-screen2


latar1


latar2


latar3

3.2 Teknik Analisis

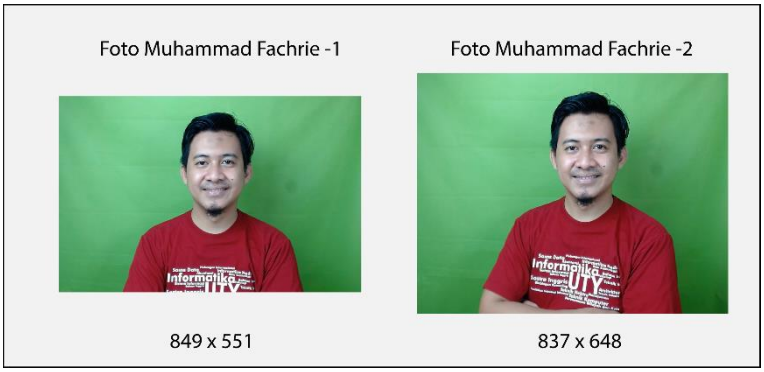
Analisis yang dilakukan adalah mempartisi gambar menjadi beberapa wilayah yang tidak tumpang tindih dengan karakteristik yang homogen, seperti intensitas, warna, dan tekstur. Segmentasi citra didasarkan pada satu dari dua property nilai intensitas yaitu mendeteksi diskontinuitas atau mendeteksi similaritas, kemudian selanjutnya memilih salah satu nilai dari suatu daerah citra untuk mewakili daerah tersebut. Pemilihan nilai dilakukan secara acak, dan memodifikasi susunan piksel berdasarkan pada beberapa transformasi geometri.

HASIL DAN PEMBAHASAN

4.1 Inisialisasi Objek Gambar

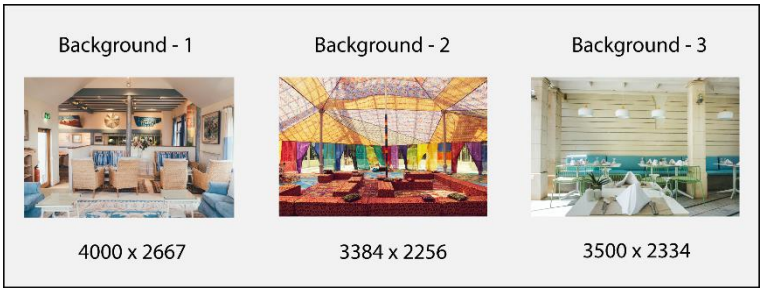
Implementasi yang dilakukan untuk pemrosesan citra digital akan menggunakan module yang dikustomisasi untuk dapat menjadikan gambar sebagai objek dengan atribut-atribut yang membantu untuk memudahkan dalam pemrosesan. Inisialisasi objek dilakukan dengan memasukkan nilai pada parameter pertama berupa array numpy 3 dimensi yang menyusun pixel pada citra. Kode untuk objek gambar dapat diakses di folder modules/Image/Image.py. Berikut ini adalah inisialisasi citra yang dilakukan:

a) Inisialisasi citra dengan latar belakang hijau




```
image_green_screen1 = Image(cv2.imread(ROOT + asset_dir + 'green-screen.jpg'), 'Foto Muhammad Fachrie - 1')
image_green_screen2 = Image(cv2.imread(ROOT + asset_dir + 'green-screen2.jpg'), 'Foto Muhammad Fachrie - 2')
```

b) Inisialisasi citra untuk latar belakang




```
background1 = Image(cv2.imread(ROOT + asset_dir + 'latar1.jpg'), 'Latar Belakang - 1')
background2 = Image(cv2.imread(ROOT + asset_dir + 'latar2.jpg'), 'Latar Belakang - 2')
background3 = Image(cv2.imread(ROOT + asset_dir + 'latar3.jpg'), 'Latar Belakang - 3')
```

4.2 Manipulasi Citra

4.2.1. *Flip Image*

Flipping citra dilakukan pada gambar dengan latar belakang hijau. Berikut adalah potongan kode untuk *flipping image*:

```
#baca ukuran citra
rows = len(citra[:])
cols = len(citra[0,:])
#buat matrix zero untuk menampung gambar hasil flipping
flipped_image = np.zeros((rows, cols, 3))        #angka 3 menyatakan 3 channel RGB
for row in range(rows):
    for col in range(cols):
        #hitung posisi kolom yang baru jika flip_horizontal = 1
        if (flip_horizontal == 1):
            new_col = (cols - 1) - col
        else:
            new_col = col
        #hitung posisi baris yang baru jika flip_vertikal = 1
        if (flip_vertikal == 1):
            new_row = (rows - 1) - row
```

```
        else:
            new_row = row
        #isi pixel pada flipped_image dengan posisi baris dan kolom baru
        flipped_image[new_row, new_col] = citra[row, col]
#konversi flipped_image menjadi uint8
flipped_image = flipped_image.astype(np.uint8)
return flipped_image
```

4.2.2. *Scale Image*

Scalling image dilakukan untuk mengatur ukuran pada citra backround, sehingga ukurannya akan proporsional untuk citra dengan latar belakang hijau. Berikut adalah potongan kode untuk *scalling image*:

```
  #baca ukuran image
rows = len(image[:])
cols = len(image[0,:])
#buat matrix zero untuk menampung gambar hasil translasi
new_rows = round(scale * rows)
new_cols = round(scale * cols)
scaled_image = np.zeros((new_rows, new_cols, 3))      #angka 3 menyatakan 3 channel RGB
for row in range(rows):
    for col in range(cols):
        #hitung baris baru dan kolom baru pada gamabr hasil penskalaan
        new_row = round(scale * row)
        new_col = round(scale * col)

        #isi pixel pada image baru (image skala)
        if (new_row < new_rows and new_col < new_cols):
            scaled_image[new_row, new_col] = image[row, col]
#konversi image_penskalaan menjadi uint8
return scaled_image.astype(np.uint8)
```

4.3 Proses Segmentasi

4.2.1. *Image Selection*

Proses seleksi dilakukan untuk menentukan pixel mana yang harus diubah ke pixel yang baru. Proses seleksi ini dilakukan dalam 3 tahapan sebagai berikut:

(a) *Masking*

Masking merupakan proses untuk menyeleksi pixel citra berdasarkan intensitas warna yang telah ditentukan. Untuk masking akan ditentukan batas atas dan bawah nilai RGB untuk menyeleksi pixel hijau. Berikut ini adalah nilai yang digunakan untuk *masking*:

```
lower_green = np.array([0, 60, 0]) # Batas bawah pixel green
upper_green = np.array([210, 255, 180]) # Batas atas pixel green
```

Batas *masking* ini akan diterapkan dengan memeriksa apakah channel RGB pada citra termasuk di dalam range dari nilai mask. Berikut adalah proses untuk masking:

```
def isInRange(rgb, lowerRGB, upperRGB):
 # Function untuk mengecek apakah pixel RGB berada di dalam range dari Lower dan upper RGB
 for i in range(3):
   if not (lowerRGB[i] < rgb[i] < upperRGB[i]):
     break
   elif (lowerRGB[i] < rgb[i] < upperRGB[i]):
     if (i == 2):
       return True
     else :
       continue
 return False
```

(b) Memeriksa dominasi pada channel warna hijau

Pada saat proses masking, ternyata terdapat ketidaksesuaian dalam penyeleksiannya. Beberapa pixel akan tetap masuk dalam jangkauan mask. Namun, bukanlah pixel yang berwarna hijau. Hal tersebut disebabkan karena warna lain lebih dominan (nilai channel lebih tinggi) daripada hijau. Berikut ini adalah potongan kode untuk memeriksa dominasi warna hijau:

```
def isColorDominant(rgb, channel):
  # Function untuk mengecek apakah pixel RGB memiliki warna channel yang dominan
  return (rgb[channel] == max(rgb))
```

(c) Menentukan toleransi selisih channel warna utama

Kedua proses sebelumnya masih belum menunjukkan hasil yang optimal untuk penyeleksian pixel. Ternyata, setelah diperiksa dominasi warna hijau memang sesuai, namun selisihnya dengan pixel lain tidak terlalu besar. Maka, untuk itu ditentukan toleransi dari selisih antara warna yang menjadi dominan (hijau) dengan channel warna lain. Berikut ini adalah potongan kode:

```
def isTolerant(rgb, channel=0, n=0):
  for i in range(3):
    if i != channel:
      if int(rgb[channel]) - int(rgb[i]) > n:
        if i == 2:
          return True
        else :
          continue
  return False
```

(d) Seleksi pixel

Dengan menggunakan logika yang telah dibuat sebelumnya, kita dapat menggabungkannya untuk melakukan proses seleksi seperti berikut:

```
def isSelected(rgb, lowerRGB, upperRGB):
  _isInRange = isInRange(rgb, lowerRGB, upperRGB)
  _isGreenDominant = isColorDominant(rgb, 1)
  _isTolerant = isTolerant(rgb, 1, 10)
  return (_isInRange & _isGreenDominant & _isTolerant)
```

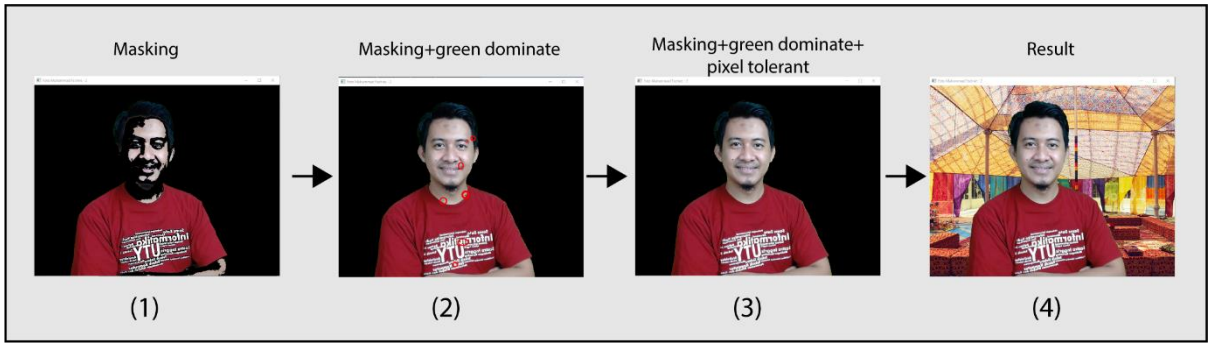
4.2.2. Manipulasi *Background*

Setelah logika untuk seleksi pixel pada bagian sebelumnya telah dibuat, maka kita dapat menerapkannya untuk memanipulasi latar belakang pada citra yang diinginkan. Beberapa proses logika tersebut dapat digabungkan menjadi seperti berikut:

```
for row in range(rows):
  for col in range(cols):
    if isSelected(image[row, col], lower_green, upper_green):
      for i in range(3):
        new_pixel = 0
        if not (len(background) == 0):
          if (rows > len(background[:]) | cols > len(background[0,:])):
            new_pixel = background[row, col, i]
        new_image[row, col, i] = new_pixel
    else :
      new_image[row, col] = image[row, col]
```

Jika proses seleksi berhasil, maka pemeriksaan dilakukan untuk setiap channel warna citra. Jika terdapat image untuk background, maka pixel yang baru akan menggunakan nilai yang sama dengan posisi pixel pada citra background. Jika tidak terdapat image background, maka pixel yang baru akan di-*assign* nilai 0. Namun, ketika pixel tidak termasuk dalam seleksi, pixel warna pada gambar baru akan di-*assign* dengan pixel gambar original yang merupakan gambar objek selain latar belakang.

Untuk proses yang terjadi, dapat diurutkan sebagai berikut:



4.4 Hasil

File main.py merupakan program utama yang digunakan untuk mengeksekusi proses secara keseluruhan. Berikut ini adalah tahapan yang dilakukan untuk mengubah citra awal menjadi citra yang diinginkan:

(a) *Flipping Image*

Flipping dilakukan secara horizontal pada gambar dengan latar belakang hijau. Berikut ini adalah potongan kode untuk melakukan *flip*:

```
image_green_screen.flip(1, 0) # Melakukan flip pada image dengan green screen
```

Dengan demikian, citra dengan latar belakang hijau akan menjadi seperti berikut:

(b) *Rescale Image*

Skala ulang dilakukan dengan tujuan agar ukuran citra *background* memiliki proporsi yang sama dengan ukurang citra dengan latar belakang hijau. Untuk memudahkan dalam mencari nilai skala, kita dapat menggunakan fungsi tambahan dengan mencari skala terbesar (max) diantara hasil perhitungan terhadap baris dan kolom.

```
def count_scale(original_len: list, target_len:list):
  scale_x = target_len[0] / original_len[0]
  scale_y = target_len[1] / original_len[1]
  return max(scale_x, scale_y)
```

```
# Menentukan nilai skala yang sesuai dengan proporsi gambar dengan greenscreen
image_green_screen_size = [len(image_green_screen.get_matrix()[:]), len(image_green_screen.get_matrix()[0,:])]
background_size = [len(background.get_matrix()[:]), len(background.get_matrix()[0,:])]
_scale = count_scale(background_size, image_green_screen_size)

background.scale(_scale) # Rescale background menjadi lebih kecil
```

Dengan perhitungan di atas, maka dapat dihasilkan ukuran gambar yang saling proporsional antara *image green screen* dengan *background*.

Background	Original		Target					
			Image_green_screen1			Image_green_screen2		
	X	Y	Scale	Result		Scale	Result	
				X	Y		X	Y
Background-1	4000	2667	0.21225	849	566	0.24296962879640044	972	648
Background-2	3384	2256	0.250886524822695	849	566	0.2872340425531915	972	648
Background-3	3500	2334	0.24257142857142858	849	566	0.2776349614395887	972	648

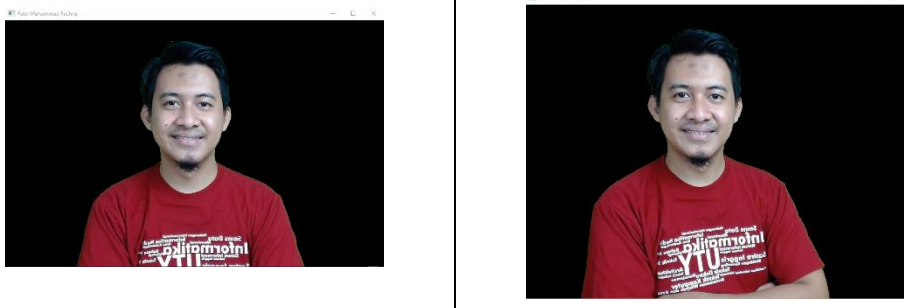
(c) *Image Segmentation and Changing Background*

Segmentasi dilakukan dengan melakukan proses seleksi pixel yang telah dibahas pada bagian sebelumnya. Kita dapat merubah *background* dari citra berlatar belakang hijau dengan mengisi parameter atribut change_background dengan atau tanpa objek *background*.

1) Tanpa argumen objek *image background*

```
image_green_screen.change_background() # Mengubah background greenscreen dengan pixel hitam
```

Output



2) Dengan objek *image background*

```
image_green_screen.change_background(background) # Mengubah background greenscreen dengan gambar baru
```

Output

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi dan analisis yang telah dilakukan telah menghasilkan keluaran berupa citra baru yang sebelumnya memiliki latar belakang hijau diubah menjadi latar belakang yang baru menggunakan citra lain. Ini menandakan bahwa segmentasi telah berhasil dilakukan untuk memisahkan antara latar belakang yang merupakan layar warna dominan hijau dengan objek citra lain yang terdapat di gambar. Kemudian, setelah berhasil melakukan seleksi, maka pixel pada citra yang dilakukan segmentasi akan diubah menjadi pixel baru menggunakan citra lain sebagai latar belakang.

5.2 Saran

- Untuk proses segmentasi lanjutan, diperlukan algoritma machine learning sehingga penyeleksian dapat dilakukan lebih mendetail dengan mempertimbangkan komponen atau pola lain yang muncul pada citra;
- Memerlukan algoritma yang lebih optimal untuk mempersingkat waktu pemrosesan;

DAFTAR PUSTAKA

[1] Liza Angriani , Indra bayu , dan Intan Sari Areni, "*Segmentasi Citra dengan Metode Threshold pada Citra Digital Tanaman Narkotika"*, 2015