Laporan Tugas Besar
Content-Based Image Retrieval
“ Rider Finder “

Tugas Kelompok

Pemrosesan & Manajemen Data Multimedia

IF5153

Oleh :

Mohamad Hanifan (23518026)
Fais Zharfan Azif (23518037)
Izuardo Zulkarnain (23518039)

Dosen :
Dr.techn. Saiful Akbar, ST., M.T.
Fitra Arifiansyah S.Kom., MT.

MAGISTER INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2019

I. Latar Belakang

Kamen rider adalah tokoh yang diidolakan oleh anak-anak hingga orang dewasa. Sejak awal kamen rider dibuat, telah ada berbagai macam kamen rider yang diciptakan dengan jenis-jenis yang berbeda. Karena jenis yang bermacam-macam, orang menjadi susah untuk mencari kamen rider yang sejenis ketika namanya tidak diketahui dan orang tersebut hanya memiliki satu gambar contoh kamen rider idolanya.

Untuk itu, dibuatlah sebuah aplikasi yang dapat menerima *input* sebuah gambar kamen rider dan mengembalikan beberapa gambar yang merupakan kamen rider dengan jenis yang sama dengan input gambar yang diberikan.

II. Masalah

Mencari kamen rider dengan jenis yang sama berdasarkan gambar *query* yang diberikan, dengan asumsi *query* yang diberikan adalah gambar sebuah kamen rider.

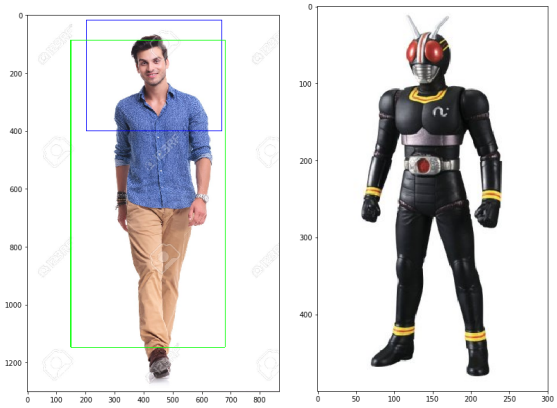
III. Batasan Masalah

Batasan dari masalah yang akan diselesaikan dalam pengerjaan tugas ini adalah sebagai berikut:

A. Gambar *query* harus berupa gambar *close-up* muka dari kamen rider
B. Dataset dibatasi dengan gambar muka/topeng dari kamen rider dan bukan gambar keseluruhan badan
C. Dataset adalah *close-up* dari topeng karakter
D. Pencarian hanya berdasarkan karakteristik topeng dalam gambar. Tidak menggunakan teks, proses *learning*, dsb.

IV. Studi Kelayakan Fitur/ Teori/ Hipotesa

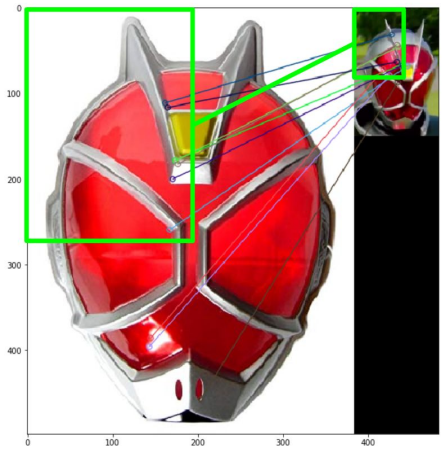
1. Head Detection



Gambar 1. Head detection

Head detection dilakukan agar objek yang dibandingkan hanya wajah ketika gambar yang didapat adalah gambar satu badan penuh. Pada tahap ini, deteksi kepala untuk manusia sudah dapat dilakukan, tetapi untuk deteksi kepala dari kamen rider seringkali tidak terdeteksi sehingga ini dapat menjadi masalah tersendiri dan pada akhirnya metode ini tidak digunakan.

2. Region Feature Matching



Gambar 2. Region feature matching

Untuk meningkatkan akurasi matching, sebuah gambar dapat dibagi menjadi 4 region, yaitu region atas kanan, atas kiri, bawah kanan, dan bawah kiri. Matching kemudian akan dilakukan pada region yang sama agar perbandingan dapat lebih simpel dan tepat.

Untuk metode ini, walaupun region sudah disamakan, hasil dari matching menggunakan metode SIFT, SURF, dan ORB masih sangat acak dan random sehingga metode ini tidak digunakan.

3. Histogram Matching
   a. BGR Histogram Matching

      BGR Histogram Matching dilakukan dengan melakukan kalkulasi perbandingan histogram. Setelah didapat *value* histogram dari gambar query, lalu dilakukan matching value histogram gambar query dengan histogram gambar pada dataset. Metode ini mampu menemukan gambar yang memiliki karakteristik warna yang sama.

      Walaupun demikian, pada tahap ini, ditemukan masalah bahwa jika gambar memiliki kecerahan berbeda, maka histogramnya berbeda dan nilai kemiripan menjadi kecil.

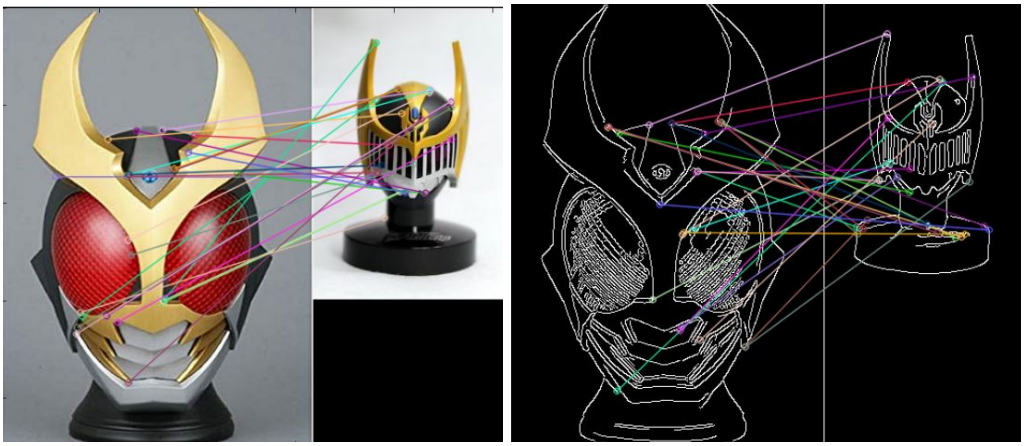
Gambar 3. Gambar dengan kecerahan berbeda

b. HSV Hue-Value Matching

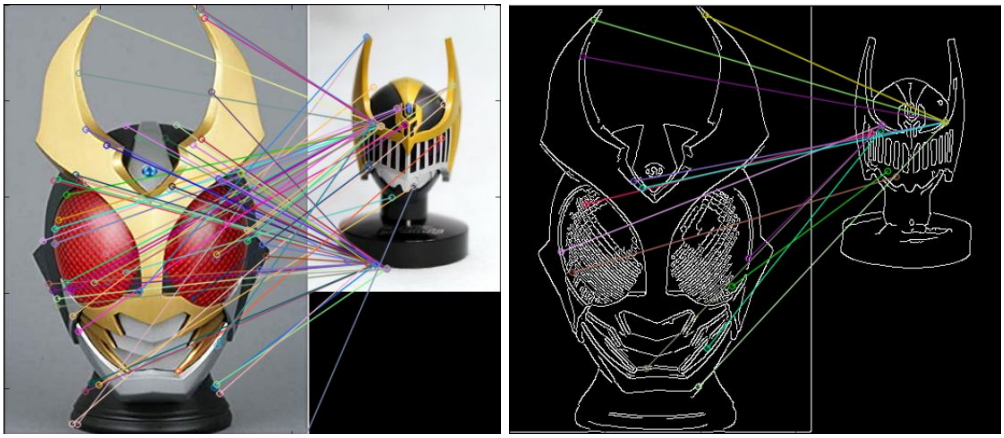
Gambar 4. Gambar dengan bentuk berbeda dengan histogram mirip

Berangkat dari permasalahan BGR histogram matching, digunakanlah histogram matching berbasis HSV. HSV digunakan karena pembagian nilai H, S, dan V yang cocok untuk menangani kecerahan gambar dengan cara mengabaikan nilai *saturation* agar gambar yang memiliki kecerahan berbeda tetap dapat dianggap mirip. HSV histogram ini mampu mengatakan gambar x memiliki kemiripan yang tinggi.

4. Edge Matching



Gambar 5. Brute force edge matching dengan dan tanpa edge detection



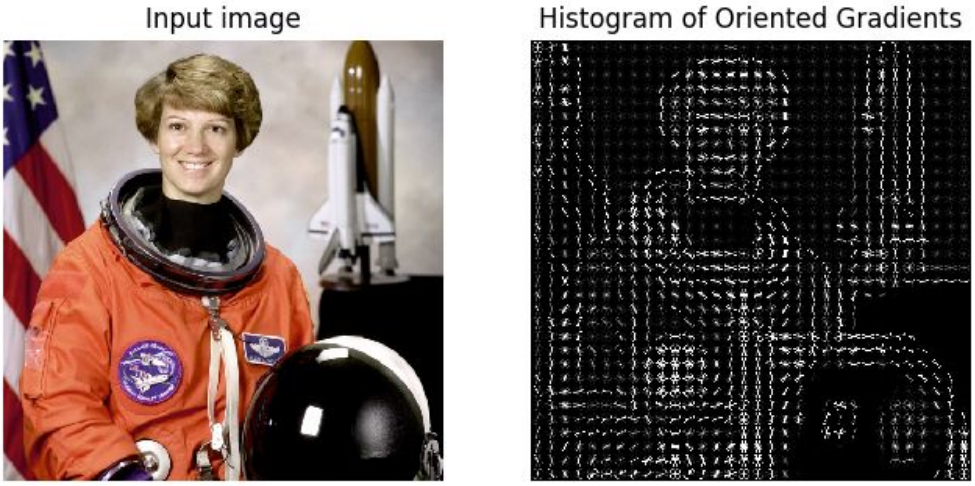
Gambar 6. SIFT edge matching dengan dan tanpa edge detection

Dengan melakukan *edge detection*, bisa didapatkan bentuk dari gambar saja yang harusnya bisa digunakan untuk mendeteksi bentuk gambar dengan lebih akurat tanpa noise. Jika dilihat di Gambar 5 dan Gambar 6, Gambar 6 yang menggunakan metode SIFT setelah dilakukan edge detection memiliki hasil *matching* yang tidak tumpang tindih seperti pada brute force.

Namun, metode ini merupakan metode yang melakukan *edge detection, corner detection,* hingga *shape detection*, secara otomatis sehingga tidak banyak yang dapat dimodifikasi selain *thresholdnya*. Untuk data bentuk yang jelas seperti mobil, kursi, atau objek-objek berbentuk unik, metode ini akan menghasilkan hasil yang bagus. Tetapi, dalam kasus kamen rider, metode ini tidak dapat membedakan jenis-jenis kamen rider yang berbeda dan proses pencarian edge, shape, dsb. Tidak dapat dimodifikasi untuk disesuaikan dengan kebutuhan sehingga metode ini tidak digunakan.

5. Shape Matching
   a. HOG Descriptor

      Histogram Of Oriented Gradients (HOG) adalah descriptor fitur yang digunakan untuk deteksi objek. HOG mendeskripsikan fitur berdasarkan histogram lokal dari orientasi gradien yang diberi bobot dengan magnitude gradien. Contoh hasil HOG ditunjukkan pada Gambar 7.



Gambar 7. Hasil transformasi menggunakan HOG

   b. Shape Matching

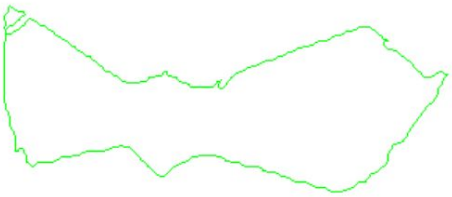
Gambar 8. Bentuk yang mirip

      Shape matching berfokus pada kontur luar dari gambar, sehingga dengan gambar *close-up,* bentuk topeng akan terlihat lebih menonjol sehingga metode shape matching dapat menemukan mana topeng yang mirip dan mana yang tidak. Gambar 8 menunjukkan 2 buah gambar yang dianggap memiliki bentuk yang sama.

6. Inner Contour Shape

   Inner contour shape adalah metode lanjutan dari shape match, dimana ketika pada shape match hanya kontur luar yang dipertimbangkan, inner contour shape dapat mencari bentuk kontur dari topeng bagian dalam seperti visor, bagian mulut, atau antena saja.

Gambar 9. Kontur yang dideteksi secara keseluruhan



Gambar 10. Contoh kontur visor yang terdeteksi

Inner contour didapat dari proses edge detection dengan canny edge yang menghilangkan keseluruhan warna dan hanya menyisakan edge. Kemudian, dilakukan dilatasi pada hasil tersebut agar edge semakin terlihat. Setelah itu, menggunakan fungsi findContours, bisa didapatkan berbagai macam bentuk-bentuk kontur keseluruhan pada gambar yang kemudian dapat dibagi berdasarkan ukuran relatifnya terhadap ukuran gambar. Misalnya, ketika kontur berukuran 90% dari keseluruhan gambar, bisa dipastikan bahwa itu adalah kontur luar. Kontur dengan ukuran 40-60% dan berlokasi di tengah gambar bisa berarti itu merupakan kontur visor atau mulut, dan sebagainya.

V. Implementasi

A. Dataset

Gambar 11. Dataset

Dataset berisi kumpulan wajah/ topeng karakter secara *close-up* seperti contoh pada Gambar 11. Dataset memiliki 50 data yang terdiri dari 10 jenis karakter berbeda, dengan tiap karakter memiliki 5 gambar yang relevan.

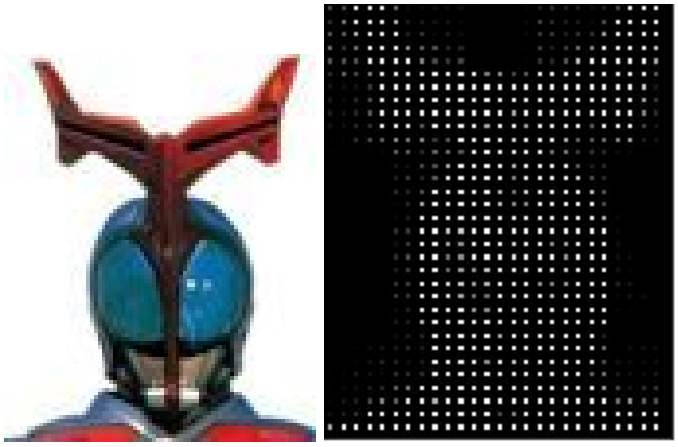
B. Fitur

1. HOG Descriptor

   HOG Descriptor dilakukan dengan cara mengambil HOG dari gambar query, merubahnya dengan threshold tertentu. Tidak hanya gambar query, seluruh dataset juga diubah menjadi format HOG, hal ini dilakukan agar tingkat akurasi dan presisi pada saat proses Contour matching meningkat. Kode dan hasil proses ini dapat dilihat pada Gambar 12 dan Gambar 13.

```
import matplotlib.pyplot as plt

from skimage.feature import hog
from skimage import data, exposure


image = cv2.imread("images/KABUTO1.jpg")

fd, hog_image = hog(image, orientations=1, pixels_per_cell=(3, 3),
                    cells_per_block=(1, 1), visualize=True, multichannel=True)


# Rescale histogram for better display
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))
```

Gambar 12. Kode program HOG



Gambar 13. Hasil transformasi HOG pada gambar Kamen Rider

2. Contour / Shape Matching

Contour / Shape Matching dilakukan setelah gambar query dan gambar yang ada di dataset ditransformasi menjadi gambar HOG, Setelah itu, hasil dari perbandingan ini dibatasi dengan nilai threshold tertentu dan diurutkan dari yang paling mirip hingga yang tidak mirip. Hasil yang sudah diurutkan inilah yang akan ditampilkan ke user sebagai hasil dari retrieval. Kode proses ini dapat dilihat pada Gambar 14.

```
m1 = cv2.matchShapes(query,hog_image,cv2.CONTOURS_MATCH_I2,0)
print('Processing : {} with value {}'.format(filename, m1))
if m1 < 10:
    all_results.append(cv2.imread(filename))
    all_distance = np.append(all_distance, m1)
    all_names.append(filename)
```

Gambar 14. Kode program shape matching

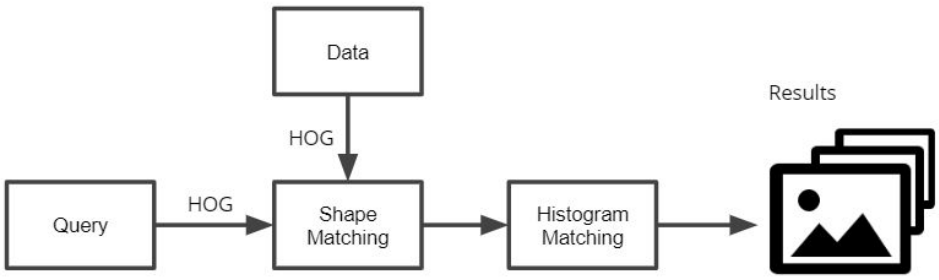
3. HSV Histogram Matching

   HSV Histogram Matching dilakukan dengan cara mengambil histogram dari gambar query, membuang nilai *saturation* nya, lalu membandingkannya dengan histogram dari gambar hasil shape matching yang nilai *saturation* nya juga diabaikan. Perbandingan dilakukan untuk semua data yang lolos dari proses shape matching. Setelah itu, hasil dari perbandingan ini dibatasi dengan nilai threshold tertentu dan diurutkan dari yang paling mirip hingga yang tidak mirip. Hasil yang sudah diurutkan inilah yang akan ditampilkan ke user sebagai hasil dari retrieval. Kode utama proses pengambilan nilai hsv dan pembuangan saturation beserta perbandingannya dapat dilihat di Gambar 15.

```
image = cv.imread(one_image)
hsv_test1 = cv.cvtColor(image, cv.COLOR_BGR2HSV)

h_bins = 50
v_bins = 600
histSize = [h_bins, v_bins]

# hue varies from 0 to 179, value from 0 to 255
h_ranges = [0, 180]
v_ranges = [0, 256]
ranges = h_ranges + v_ranges # concat lists

# Use the 0-th and 2-st channels, ignore saturation
channels = [0, 2]

hist_base = cv.calcHist([hsv_base], channels, None, histSize, ranges, accumulate=False)
cv.normalize(hist_base, hist_base, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)
hist_test1 = cv.calcHist([hsv_test1], channels, None, histSize, ranges, accumulate=False)
cv.normalize(hist_test1, hist_test1, alpha=0, beta=1, norm_type=cv.NORM_MINMAX)

base_test1 = cv.compareHist(hist_base, hist_test1, method)
```

Gambar 15. Kode utama program histogram matching

C. Proses Query

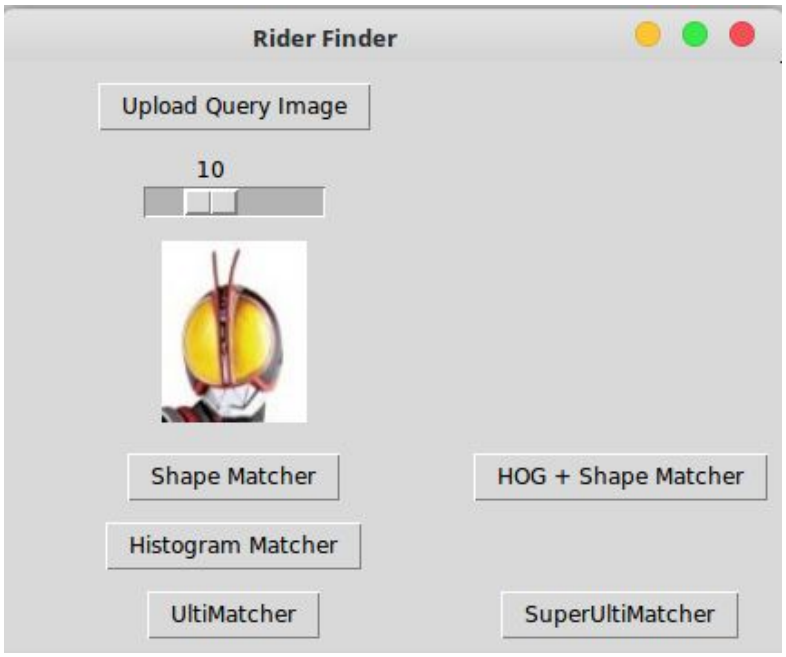


Gambar 16. Proses Query

Proses query dilakukan dalam tiga tahap ekstraksi fitur, yaitu tahap transformasi ke HOG, tahap Shape Matching, dan tahap HSV Histogram Matching. Pada tahap HOG, baik gambar query maupun gambar yang ada di dataset akan dilakukan transformasi menjadi HOG image, ini berfungsi untuk menghilangkan warna background yang dominan dan merubah gambar menjadi *contour* atau *shape image*. Kemudian, hasil dari HOG lah yang akan dibandingkan bentuknya melalui Shape Matching. Data hasil yang didapat setelah Shape Matching kemudian dibandingkan lagi dengan gambar *query* dengan HSV Histogram Matching. Hasil dari histogram matching akan di ranking berdasarkan kedekatannya, dan sejumlah gambar hasil akan diberikan sesuai dengan keinginan pengguna.

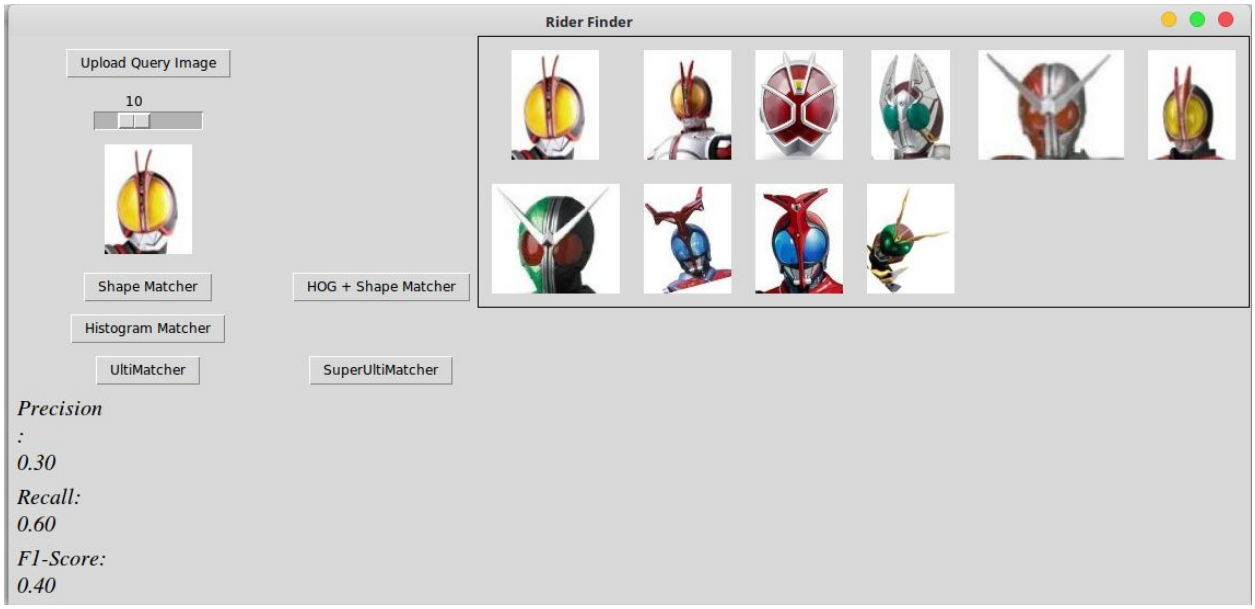
D. GUI Rider Finder

1. Tampilan Awal



Gambar 17. Tampilan GUI program Rider Finder

2. Tampilan hasil query



Gambar 18. Tampilan GUI hasil query

3. Penjelasan GUI
   - Upload Query Image : Tombol untuk mengunggah gambar query
   - Shape Matcher : Tombol untuk melakukan matching menggunakan shape / contour matching
   - Histogram Matcher : Tombol untuk melakukan histogram matching
   - UltiMatcher : Tombol untuk melakukan histogram dan shape / contour matching
   - HOG + Shape Matcher : Tombol untuk melakukan proses transformasi gambar dengan HOG, lalu matching menggunakan shape / contour matching
   - Super Ulti Matcher : Tombol untuk melakukan proses transformasi gambar dengan HOG, lalu matching menggunakan shape / contour matching, lalu urutkan berdasarkan nilai histogram

VI. Hasil

Setelah dilakukan percobaan pada query image dan dataset, hasil yang didapatkan ditampilkan pada Tabel 1.

Tabel 1. Tabulasi hasil percobaan query

Percobaan ke -	Recall	Precision	F1- Score
1	30%	60%	40%
2	30%	60%	40%
3	40%	67%	50%

VII. Kesimpulan

Matching menggunakan HOG terlebih dahulu secara umum memiliki dampak memperbaiki hasil pencarian, walaupun hasilnya tidak terlalu signifikan dalam studi kasus ini.

VIII. Saran

Untuk selanjutnya, fitur yang digunakan bisa lebih mengarah ke *matching* pada bagian-bagian regional dari muka, misal membandingkan *visor* antar kamen rider, antenanya, atau bagian mulutnya. Bagian-bagian regional bisa didapatkan dengan membagi region gambar kedalam beberapa sub-region, atau dengan deteksi kontur yang lebih dalam pada bagian muka.

Selain itu, dataset juga dapat ditambahkan agar lebih banyak, serta menambahkan dataset dengan *angle* yang berbeda-beda agar performa fitur dapat lebih terukur efektivitasnya.