

Pengolahan Citra Dan Visi Komputer

PCVK

RTI235007

DASAR-DASAR DETEKSI OBYEK

TEACHING TEAM

MATA KULIAH PENGOLAHAN CITRA DAN VISI KOMPUTER

Tujuan Pembelajaran

- Memahami konsep dan prinsip dasar deteksi objek.
- Menjelaskan aspek pemindaian dalam deteksi objek.
- Mengenali metode dasar deteksi objek menggunakan OpenCV.

Outline Materi

1. Dasar-dasar Deteksi Objek
2. Konsep dan Prinsip Deteksi Objek
3. Aspek Pemindaian pada Deteksi Objek
4. Face Detection

DETEKSI OBYEK

Deteksi Obyek (*Object Detection*)

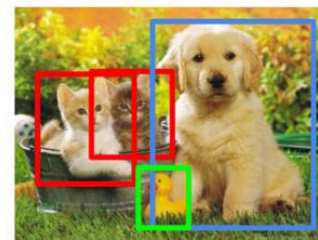
- Salah satu teknik visi computer (computer vision) yang berfungsi untuk mengidentifikasi dan menemukan objek dalam sebuah gambar atau video
- Proses deteksi objek akan menggambar kotak pembatas di sekitar objek yang terdeteksi, sehingga dapat diketahui dimanakah lokasi objek tersebut pada gambar / video atau berupa bounding box yang menandai lokasi objek di dalam citra.
- Tujuan: mendeteksi semua instance objek dari citra tersebut

Classification



CAT

Object Detection



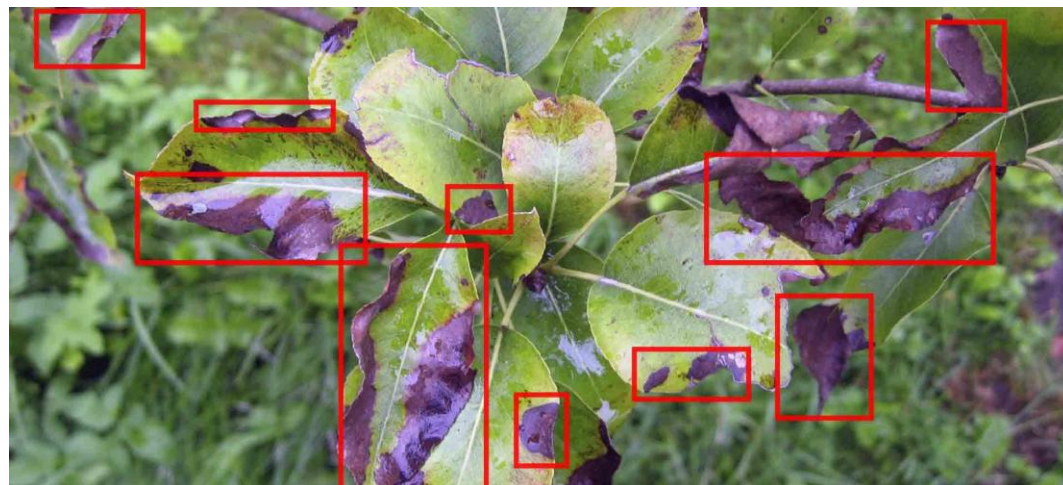
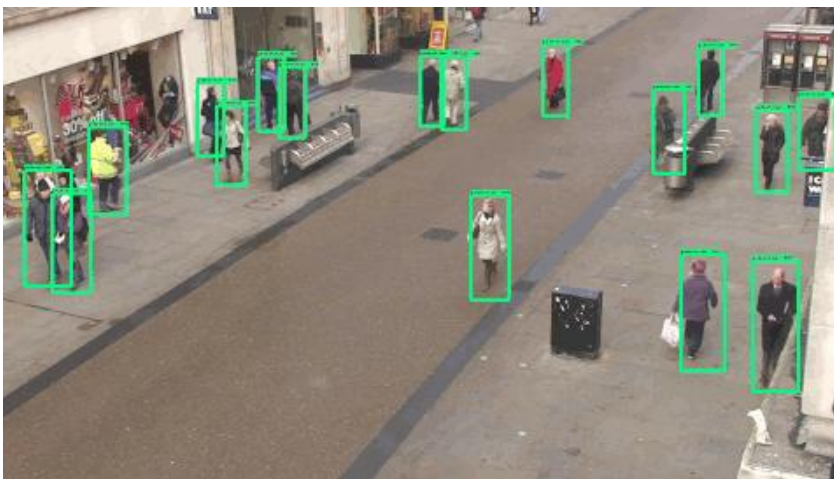
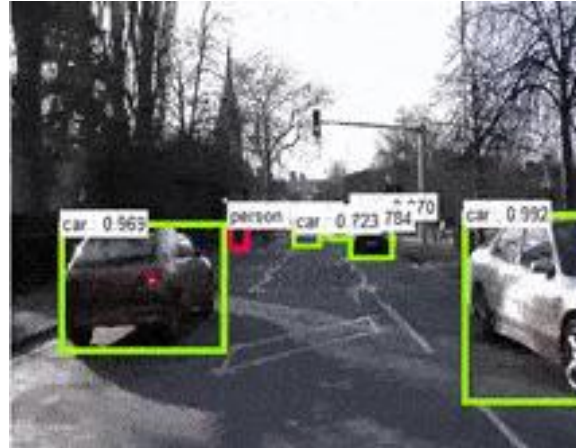
CAT, DOG, DUCK

Tujuan Deteksi Obyek

- Pendeteksian objek merupakan tahap pertama yang dilakukan karena hasil dari deteksi objek dapat memungkinkan untuk memperoleh informasi lebih lanjut mengenai objek yang terdeteksi dan lokasi objek tersebut, antara lain:
 - mengenali contoh spesifik (misalnya, untuk mengidentifikasi subjek dari wajah yang terdeteksi)
 - melacak objek pada urutan gambar (misalnya, untuk melacak wajah dalam video)
 - mengekstrak informasi lebih lanjut tentang objek (misalnya, untuk menentukan jenis kelamin subjek)

Contoh Implementasi Object Detection

- Video surveillance (CCTV)
- Crowd counting (Penghitungan kerumunan)
- Anomaly detection
- Self-driving cars



Konsep Dasar Citra dan Objek

- Objek dikenali dari perbedaan intensitas, warna, dan tekstur terhadap latar belakang.
- Elemen penting: edges, corners, contours, dan texture patterns.

Prinsip Dasar Deteksi Objek

Langkah utama:

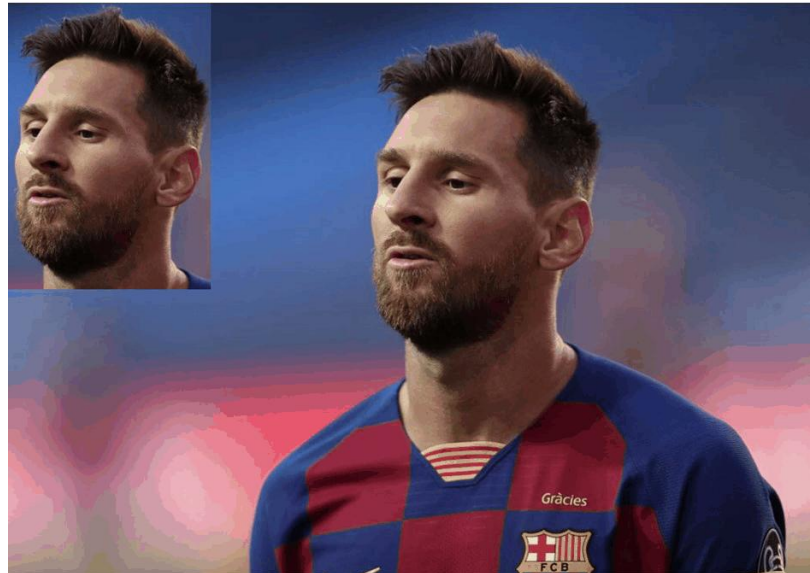
1. Ekstraksi fitur awal
2. Deskripsi fitur
3. Matching/Klasifikasi
4. Lokalisasi (bounding box)

Metode Deteksi Dasar

1. Template Matching
2. Edge Detection
3. Corner Detection
4. Contour Detection
5. Grid Detection

1. Template Matching

- Template matching adalah metode untuk mencari dan menemukan lokasi gambar template pada gambar yang lebih besar. Diperlukan dua gambar untuk menerapkan metode template matching, yaitu:
 - **Source image (I):** gambar masukan yang diharapkan cocok dengan template.
 - **Template image (T):** potongan objek yang akan dicari pada source image.



1. Template Matching

- OpenCV mengimplementasikan template matching menggunakan fungsi **matchTemplate()**.

1. method=TM_SQDIFF

$$R(x, y) = \sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2$$

2. method=TM_SQDIFF_NORMED

$$R(x, y) = \frac{\sum_{x', y'} (T(x', y') - I(x + x', y + y'))^2}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$$

3. method=TM_CCORR

$$R(x, y) = \sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))$$

4. method=TM_CCORR_NORMED

$$R(x, y) = \frac{\sum_{x', y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x', y'} T(x', y')^2 \cdot \sum_{x', y'} I(x + x', y + y')^2}}$$

1. Template Matching

5. method=TM_CCOEFF

$$R(x, y) = \sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))$$

6. method=TM_CCOEFF_NORMED

$$R(x, y) = \frac{\sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x', y'} T'(x', y')^2 \cdot \sum_{x', y'} I'(x + x', y + y')^2}}$$

Dokumentasi template matching pada OpenCV dapat dilihat pada link berikut:
https://docs.opencv.org/3.4/de/da9/tutorial_template_matching.html

2. Edge Detection

- Operasi yang dijalankan untuk mendeteksi garis tepi (edges) yang membatasi dua wilayah citra yang memiliki tingkat kecerahan yang berbeda
- Suatu titik (x,y) dikatakan sebagai tepi dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya
 - Sobel Edge Detection
 - Laplacian Edge Detection
 - Canny Edge Detection

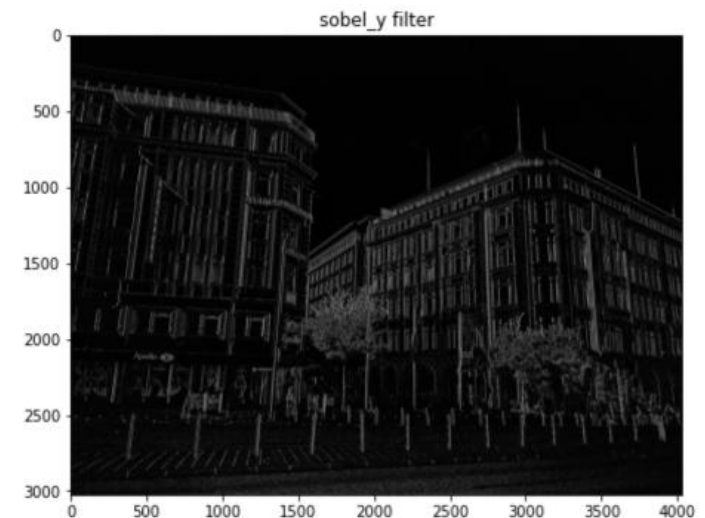
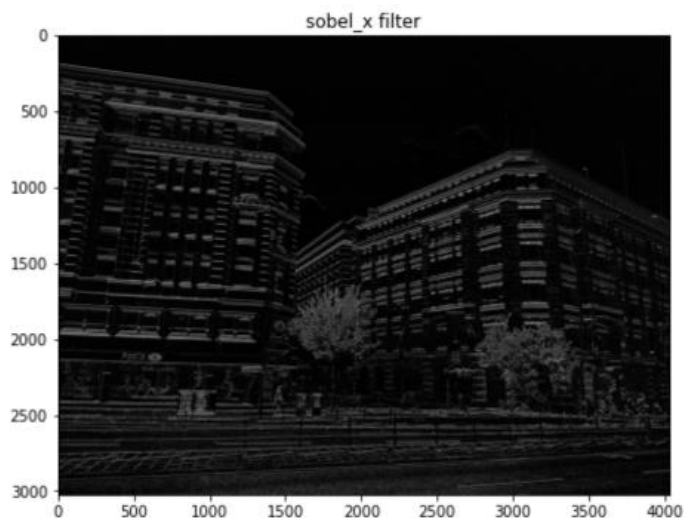
2. Edge Detection

Sobel Edge Detection

- Filter Sobel bekerja dengan menghitung gradien intensitas citra pada setiap piksel dalam sebuah citra

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1



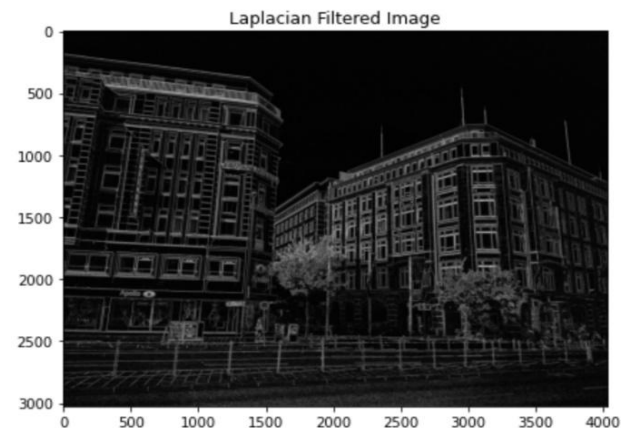
2. Edge Detection

Laplacian Edge Detection

- Metode ini sangat sensitive terhadap noise, sebuah gambar umumnya akan dihaluskan terlebih dahulu menggunakan Gaussian filter sebelum menerapkan Laplacian

0	-1	0
-1	4	-1
0	-1	0

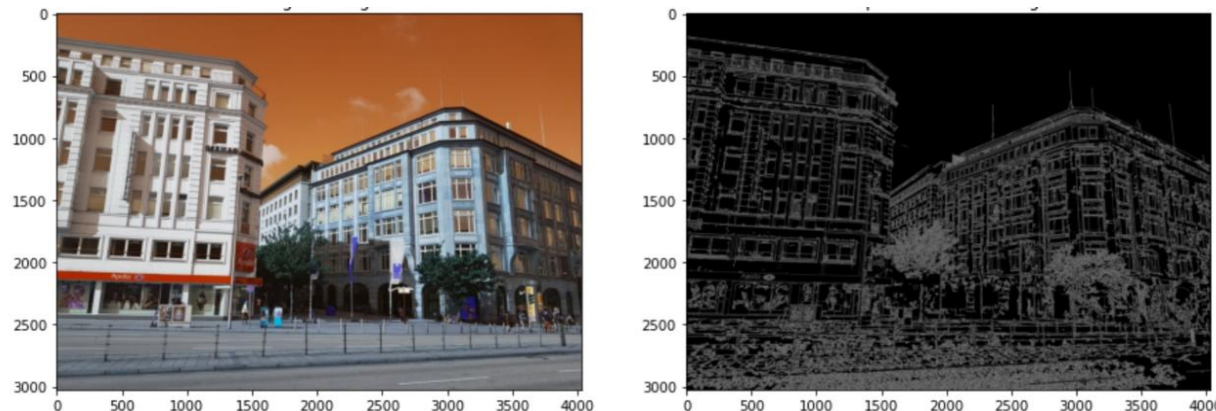
-1	-1	-1
-1	8	-1
-1	-1	-1



2. Edge Detection

Canny Edge Detection

- Metode ini bekerja dengan 4 tahap, yaitu:
 - Menghaluskan gambar dengan filter Gaussian untuk mengurangi noise
 - Menghitung gradien menggunakan salah satu operator gradien Sobel
 - Mengekstraksi titik tepi: Non-maximum suppression
 - Menghubungkan dan thresholding: Hysteresis



3. Corner Detection

- Sudut dapat diartikan sebagai persimpangan dua sisi.
- Titik sudut tidak bisa didefinisikan dari piksel tunggal, karena di sana hanya terdapat pada satu gradien di setiap titik.
- Gradien adalah arah perubahan intensitas kecerahan dalam suatu citra
- Metode:
 - Harris Corner Detection
 - Shi-Tomasi Corner Detection

3. Corner Detection

Harris Corner Detection

- Didasarkan pada variasi intensitas sinyal.
- Variasi intensitas yang besar menunjukkan adanya sudut pada citra
- Cara kerja metode Harris adalah dengan menemukan perbedaan intensitas untuk perpindahan (u, v) ke segala arah, dengan persamaan berikut:

$$E(u, v) = \sum_{x,y} \underbrace{w(x, y)}_{\text{window function}} \left[\underbrace{I(x + u, y + v)}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}} \right]^2$$

3. Corner Detection

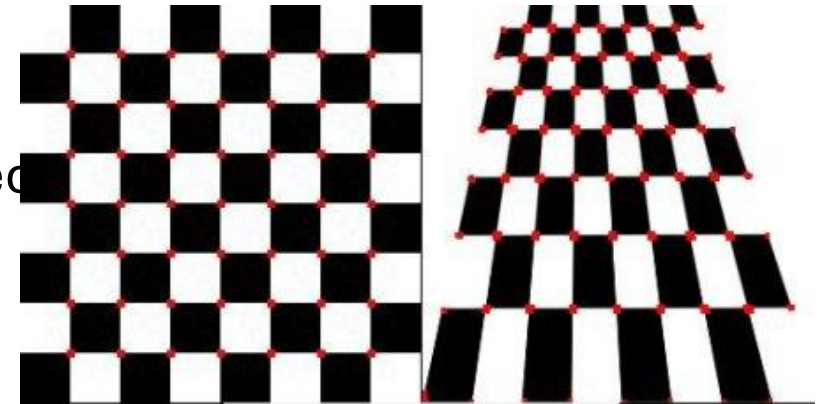
Harris Corner Detection

- Pada OpenCV, metode yang digunakan adalah:

`cv2.cornerHarris(src, dest, blockSize, kSize, freeParameter, borderType)`

dengan parameter:

- Src: citra input
- Dest: cistra untuk menyimpan hasil Harris detector
- blockSize: ukuran ketetanggaan
- ksize: kernel size untuk operator Sobel
- freeParameter: parameter bebas untuk Harris deted
- borderType: metode pixel extrapolation

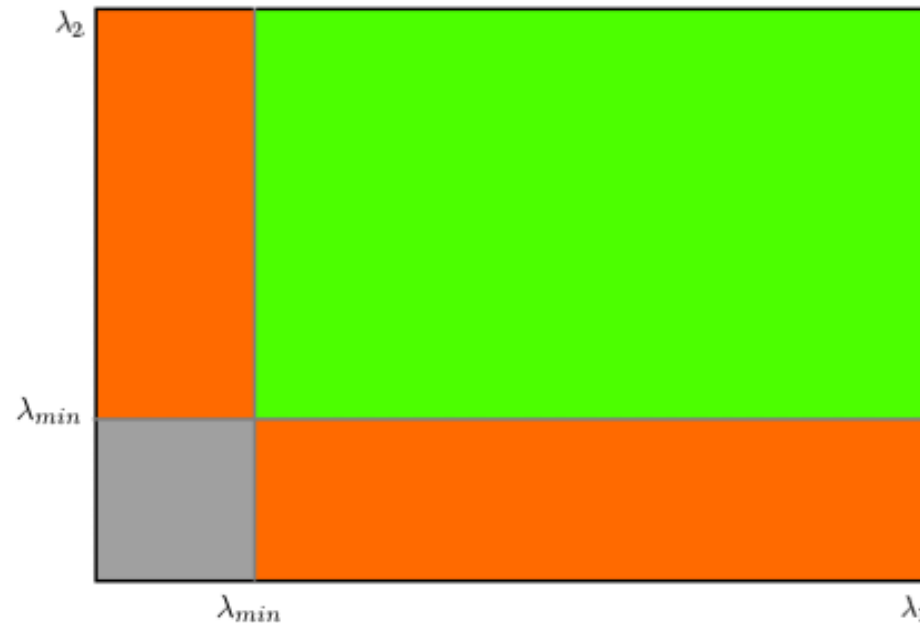


3. Corner Detection

Shi-Tomasi Corner Detection

- Shi-Tomasi Corner Detection menggunakan dasar bahwa sudut dapat dideteksi dengan mencari perubahan signifikan ke segala arah

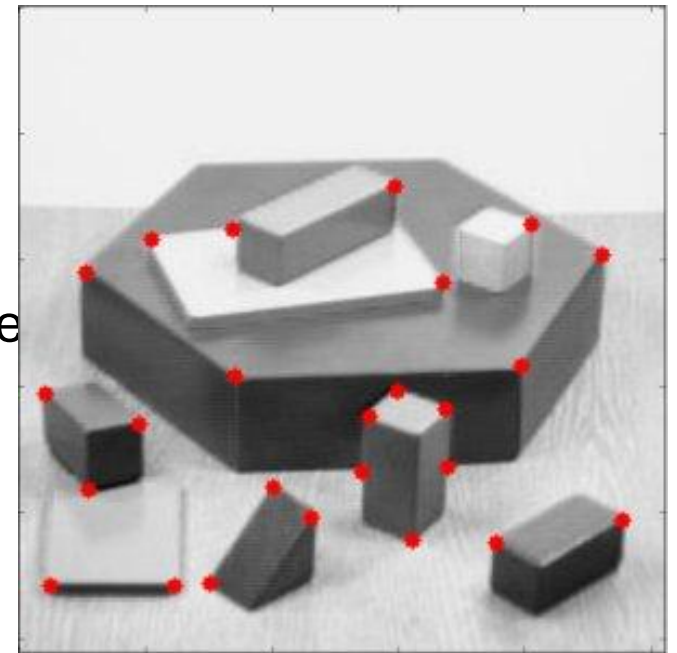
$$R = \min(\lambda_1, \lambda_2)$$



3. Corner Detection

Shi-Tomasi Corner Detection

- OpenCV memiliki fungsi, **cv.goodFeaturesToTrack()**.
- Fungsi ini akan menemukan N sudut terkuat pada gambar dengan metode Shi-Tomasi
- Parameter
 - (a) jumlah sudut yang ingin ditemukan
 - (b) tingkat kualitas, yang merupakan nilai antara 0-1,
 - (c) jarak euclidean minimum antar sudut yang terdeteksi



4. Grid Detection

- Grid detection dapat dilakukan dengan metode Hough Transform
- Tahapan:
 - Menerapkan Canny Edge Detection
 - Dilasi gambar tepi (Canny bisa menemukan kedua tepi pemisah pada grid sebagai tepi yang berbeda, dilatasi dapat membuat kedua tepi ini bergabung lagi)
 - Erosi (setelah proses dilasi, garis border menjadi tebal sehingga Hough akan mendeteksi banyak garis)
 - Menerapkan HoughLines dengan fungsi `cv2.HoughLines()` pada OpenCV
 - Menggabungkan garis yang mirip (similar)

4. Grid Detection



5. Contour Detection

- Kontur adalah kurva tertutup yang menghubungkan semua titik kontinu yang memiliki beberapa warna atau intensitas.
- Kontur mewakili bentuk objek yang ditemukan dalam gambar.
- Tahapan:
 - Ubah citra input menjadi citra biner (bisa didapat dengan mengimplementasikan thresholding atau canny edge detection).
 - Menemukan kontur menggunakan fungsi `findContours()` pada OpenCV.
 - Menggambarkan kontur pada citra input menggunakan fungsi `drawContours()`.

5. Contour Detection

- Fungsi findContours menerima tiga buah parameter, yaitu :
 - Parameter pertama adalah Image yang akan dianalisis konturnya
 - Parameter kedua menunjukkan metode retrieval.
 - Parameter ketiga menunjukkan metode yang digunakan untuk proses approximation dari shape



Dokumentasi:

https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html

Aspek Pemindaian dalam Deteksi Objek

- Pemindaian dilakukan untuk menelusuri seluruh area citra.
- Dapat dilakukan per baris, blok, atau region of interest (ROI).
- Jenis-jenis Pemintaian:
 - 1. Pemindaian spasial (spatial scanning):**
 - Pemindaian dilakukan per baris atau blok untuk mendeteksi perubahan nilai intensitas.
 - Metode sliding window: template dipindai ke seluruh area citra dengan langkah tertentu.
 - Prinsip pyramid scaling → citra diubah ukurannya untuk mendeteksi objek berbagai ukuran.
 - 2. Pemindaian multiskala dan orientasi:**
 - Pada algoritma SIFT dan ORB, sistem mendeteksi objek meskipun ukuran dan rotasi berubah.
 - Setiap skala menghasilkan keypoints berbeda → diseleksi yang paling signifikan.
 - 3. Pemindaian berbasis region (region proposal):**
 - Digunakan pada deteksi modern seperti R-CNN, YOLO, dan SSD.
 - Citra dibagi menjadi beberapa region of interest (ROI) untuk mempercepat proses deteksi.

1. Pemindaian spasial (spatial scanning): **Sliding Window Technique**

- Teknik jendela geser digunakan untuk mencari kecocokan pola di seluruh area citra.
- Digunakan pada deteksi wajah, plat nomor, atau objek kecil.

2. Pemindaian multiskala dan orientasi: **Multi-Scale Scanning**

- Citra diubah ukurannya agar sistem dapat mendeteksi objek dari berbagai ukuran.
- Digunakan dalam SIFT dan ORB.

Feature Matching

- Feature matching melakukan ekstraksi ekstraksi fitur penting dari sebuah citra menggunakan ide dasar dari corner, edge, dan contour detection. Selanjutnya akan dilakukan perhitungan distance untuk mencari kecocokan antara image sumber dengan image template.
- Jika objek pada citra template tidak berukuran sama dengan citra sumber, deteksi tetap bisa dilakukan dengan baik.

<https://www.pyimagesearch.com/2015/01/26/multi-scale-template-matching-using-python-opencv/>

<https://laptrinhx.com/opencv-template-matching-cv2-matchtemplate-2183411893/>

Feature Matching

1. Brute-Force Matching menggunakan ORB Descriptor (Orient FASE and Rotated BRIEF)
2. Brute-Force Matching menggunakan SIFT Descriptor dan Test Ratio (Scale-Invariant Feature Transform)

ORB (Orient Fast and Rotated BRIEF)

ORB adalah berasal dari "OpenCV Labs.

Diajukan oleh Ethan Rublee, Vincent Rabaud, Kurt Konolige dan Gary R. Bradski dalam artikel mereka berjudul “*ORB: An efficient alternative to SIFT or SURF*” pada tahun 2011.

Seperti yang tertulis pada judulnya, ORB adalah alternatif pengganti dari SIFT dan SURF dalam hal biaya komputasi, kinerja matching dan terutama adalah paten. SIFT dan SURF dipatenkan dan kita harus membayar untuk menggunakannya, beda dengan ORB yang tidak perlu membayar untuk penggunaannya.

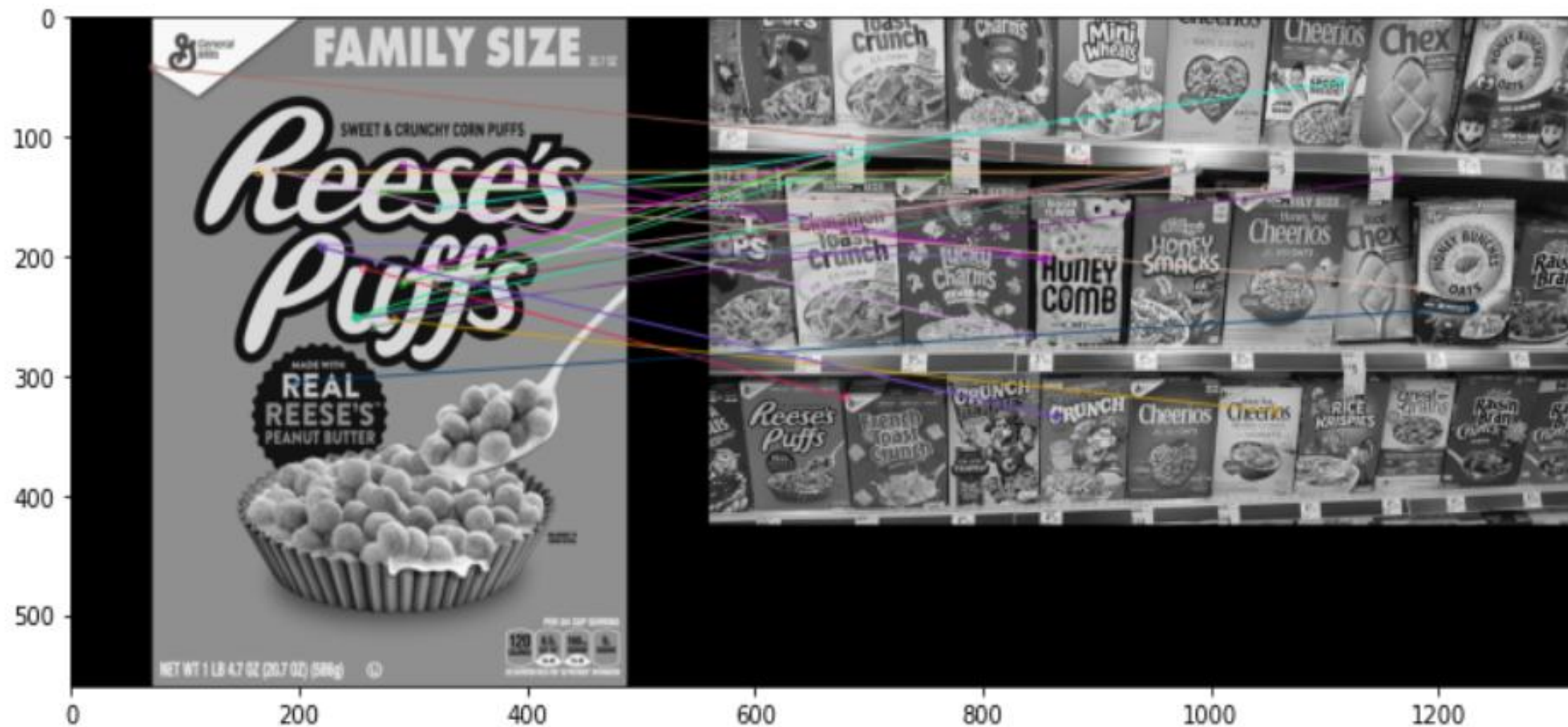
ORB

ORB pada dasarnya adalah perpaduan dari detektor keypoint FAST dan deskriptor BRIEF dengan banyak modifikasi untuk meningkatkan kinerjanya. Sebagai langkah awal, FAST digunakan untuk menemukan keypoint, lalu Harris Corner detection digunakan untuk mencari N-point tertinggi. Akan tetapi FAST tidak menghitung orientasinya. Modifikasi dilakukan agar algoritma ini invariant terhadap orientasi.

ORB



ORB



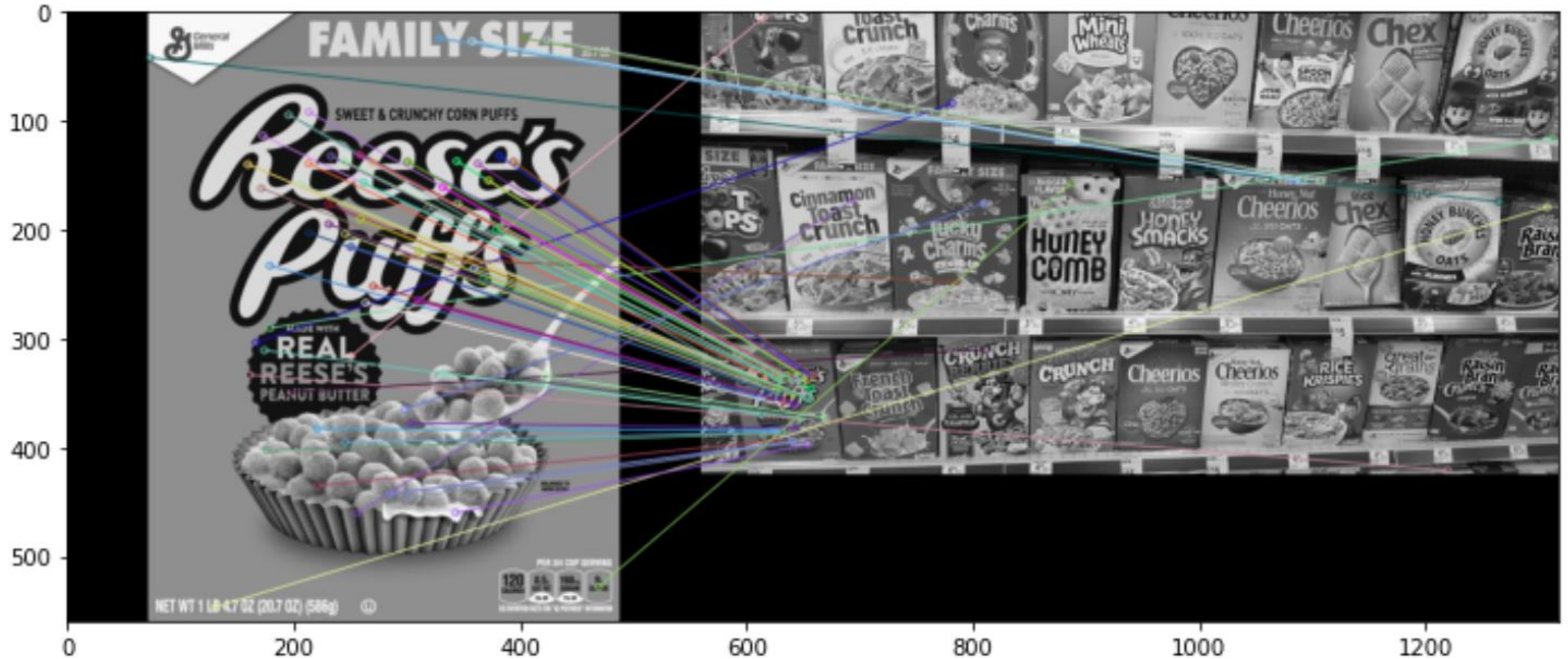
SIFT (Scale-Invariant Feature Transform)

- Pada pembahasan sebelumnya tentang Harris corner detection misal, metode-metodenya adalah rotation-invariant, artinya walaupun image dalam kondisi terrotasi, corner dapat dideteksi dengan baik.
- Karena corner akan tetap berupa corner walaupun objek dirotasikan. Berbeda kasus dengan scaling. Corner bisa jadi bukan lagi corner jika citra discaling.
- Hal ini menunjukkan bahwa Harris corner merupakan algoritma yang scale-invariant.

SIFT (Scale-Invariant Feature Transform)

- Pada tahun 2004, D. Lowe dari University of British Columbia, menemukan algoritma baru, Scale Invariant Feature Transform (SIFT) dalam artikelnya yang berjudul “*Distinctive Image Features from Scale-Invariant Keypoints*”, yang mengekstrak keypoint dan menghitung deskriptornya.
- Untuk menggunakan SIFT descriptor, harus digunakan OpenCV Versi 4.4.0.44 ke atas. Langkah awal adalah melakukan instalasi menggunakan pip terlebih dahulu sebelum menjalankan code SIFT.

SIFT (Scale-Invariant Feature Transform)



3. Pemindaian berbasis region (region proposal): **Region-Based Scanning (ROI)**

- Metode modern seperti R-CNN, YOLO, SSD membagi citra ke area potensial (region of interest) untuk efisiensi.

Integrasi Fitur dan Pemindaian

- Kombinasi antara fitur (edge, corner, contour) dan pemindaian menghasilkan sistem deteksi yang efisien dan akurat.

FACE DETECTION

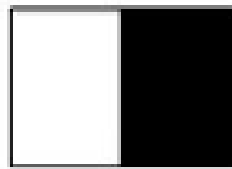
Face Detection

- Pada pertemuan ini dijelaskan cara kerja face detection menggunakan **Haar Cascades**, yaitu komponen kunci dari framework deteksi object Viola-Jones.
- Metode ini dapat mendeteksi wajah dengan cepat pada sebuah image dan mengetahui lokasinya dengan tepat. Metode ini digunakan sebagai tahap awal dalam mengenali wajah (Facial Recognition).

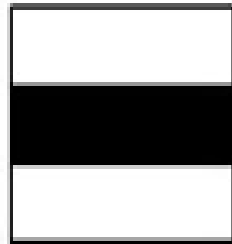
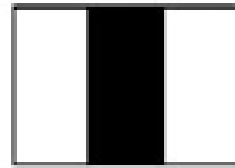
Face Detection

- Pada tahun 2001, Paul Viola dan Michael Jones mempublikasikan metode face detection berdasarkan pada konsep sederhana dari beberapa fitur kunci. Ide lain yang digunakan adalah melakukan perhitungan awal dari integral image untuk mempercepat waktu komputasi.

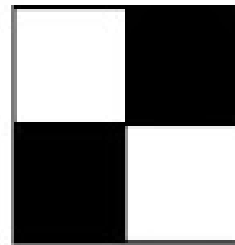
Face Detection



(a) Edge Features



(b) Line Features



(c) Four-rectangle features

Face Detection

- Tiap fitur adalah merupakan nilai tunggal yang didapatkan dengan mengurangkan rata-rata pixel pada area kotak putih dengan rata-rata pixel pada area kotak hitam.

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

0	0.1	0.8	1
0.3	0.1	0.7	0.8
0.1	0.2	0.8	0.8
0.2	0.2	0.8	0.8

$\text{mean}(\text{area gelap}) - \text{mean}(\text{area terang})$.

$\text{Sum}([0.8, 1, 0.7, 0.8, 0.8, 0.8, 0.8, 0.8]) = 6.5$

$\text{Sum}([0, 0.1, 0.3, 0.1, 0.1, 0.2, 0.2, 0.2]) = 1.2$

$\text{Mean} = 6.5 / 8 = 0.8125$

$\text{Mean} = 1.2 / 8 = 0.15$

$\text{Delta} = 0.8125 - 0.15 = 0.6625$

Nilai delta tersebut kemudian di threshold sesuai kebutuhan kita.

Semakin tinggi nilai delta ($=1$) maka akan semakin cocok fitur tersebut sebagai edge. Jika dimisalkan diberi nilai Threshold = 0.5, sedangkan nilai delta adalah 0.6625, maka fitur tersebut dianggap sebagai edge.

Face Detection

- Menghitung nilai delta untuk keseluruhan citra akan membuat komputasi menjadi sangat mahal. Algoritma Viola-Jones mengajukan metode integral image.

31	2	4	33	5	36
12	26	9	10	29	25
13	17	21	22	20	18
24	23	15	16	14	19
30	8	28	27	11	7
1	35	34	3	32	6

31	33	37	70	75	111
43	71	84	127	161	222
56	101	135	200	254	333
80	148	197	278	346	444
110	186	263	371	450	555
111	222	333	444	555	666

$$15 + 16 + 14 + 28 + 27 + 11 =$$
$$101 + 450 - 254 - 186 = 111$$

Face Detection

- Algoritma ini juga dijalankan dengan cepat dengan menggunakan **cascade classifier**.
- **Cascade classifier** adalah menghitung nilai fitur image **secara seri** berdasarkan pada fitur sederhana yang telah ditunjukkan sebelumnya.
- Misalkan dilakukan ekstraksi fitur edge terlebih dahulu, dan jika tidak termasuk sebagai fitur edge, maka fitur line dan rectangle tidak dilakukan dan bergeser ke pencarian berikutnya.

Face Detection

- Langkah awal yang dilakukan adalah dengan mengubah citra masukan ke dalam Grayscale.
- Fitur awal yang digunakan sebagai pencarian adalah edge feature yang menunjukkan adanya mata dan pipi.
- Jika di image tidak menunjukkan adanya fitur ini, dianggap bahwa tidak ada wajah pada image tersebut. Jika terindikasi ada, maka dilanjutkan dengan mencari fitur berikutnya, seperti nose bridge (bagian tengah hidung), lips (bibir), alis. Pencarian terus dilakukan menggunakan cascading, yang berarti akan terdapat ribuan fitur yang terkumpul.

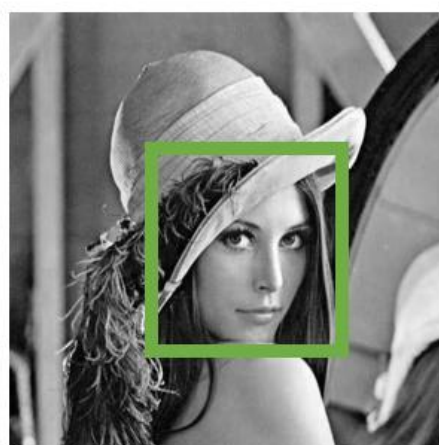
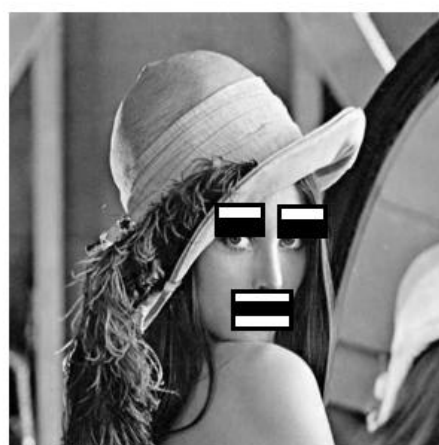
Face Detection

- Secara teori, pendekatan ini dapat digunakan pada berbagai objek dan deteksi. Sebagai contoh, dapat digunakan untuk mendeteksi mata, bibir, hidung dan lain sebagainya. Atau dapat juga digunakan untuk mendeteksi objek lain seperti mobil, motor, pesawat, dan lain sebagainya.

Face Detection

- Kelemahan dari algoritma ini adalah kebutuhan akan referensi dataset yang besar untuk mendapatkan fitur yang diinginkan.
- Pada OpenCV juga disediakan fitur-fitur tertentu yang sering digunakan dan dipake oleh banyak peneliti. Fitur-fitur tersebut disediakan OpenCV dengan format XML.

Face Detection



Face Detection

Why GitHub? ▾TeamEnterpriseExplore ▾MarketplacePricing ▾

Search

opencv / opencv

<> Code

Issues 1.9k

Pull requests 83

Wiki

Security

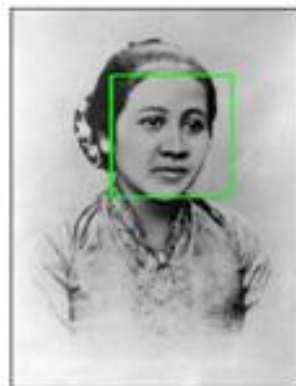
Insights

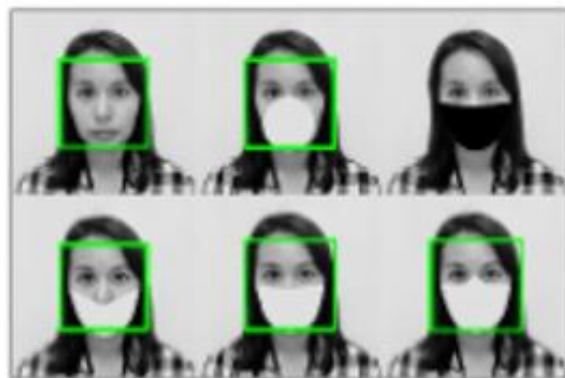
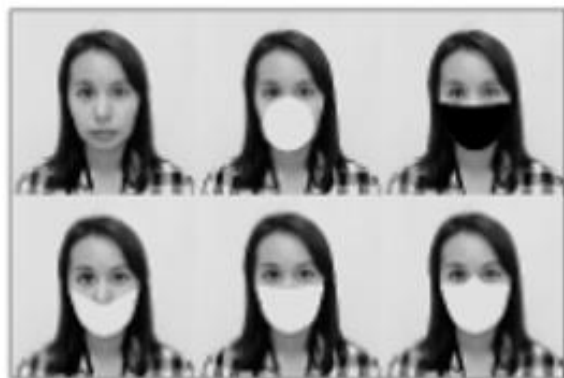
master ▾opencv / data /

alalek fix files permissions		
..		
haarcascades		fix files permissions
haarcascades_cuda		Some mist. typo fixes
hogcascades		Added HOG features to cascade detection algorithm.
lbpcascades		fix files permissions
vec_files		fix files permissions
CMakeLists.txt		Fixed mangled install layout on unix machines
readme.txt		"atomic bomb" commit. Reorganized OpenCV directory structure

master ▾opencv / data / haarcascades /

alalek fix files permissions		
..		
haarcascade_eye.xml		some attempts to tune the performance
haarcascade_eye_tree_eyeglasses.xml		some attempts to tune the performance
haarcascade_frontalcatface.xml		fix files permissions
haarcascade_frontalcatface_extended.xml		fix files permissions
haarcascade_frontalface_alt.xml		some attempts to tune the performance
haarcascade_frontalface_alt2.xml		some attempts to tune the performance
haarcascade_frontalface_alt_tree.xml		some attempts to tune the performance
haarcascade_frontalface_default.xml		some attempts to tune the performance
haarcascade_fullbody.xml		Some mist. typo fixes
haarcascade_lefteye_2splits.xml		some attempts to tune the performance
haarcascade_licence_plate_rus_16stages.xml		Added Haar cascade for russian cars licence plate detection
haarcascade_lowerbody.xml		Some mist. typo fixes
haarcascade_profileface.xml		some attempts to tune the performance





 **TERIMA KASIH!** 
