MODUL 11 – Metode Deteksi Objek: *Template Matching, Edge Detection, Corner Detection, Grid Detection, Contour Detection*

A. TUJUAN

- Mahasiswa dapat memahami konsep deteksi objek
- Mahasiswa mampu memahami beberapa metode yang dapat digunakan dalam proses deteksi objek
- Mahasiswa dapat mengimplementasikan beberapa metode dalam proses deteksi objek menggunakan Python pada Google Colab

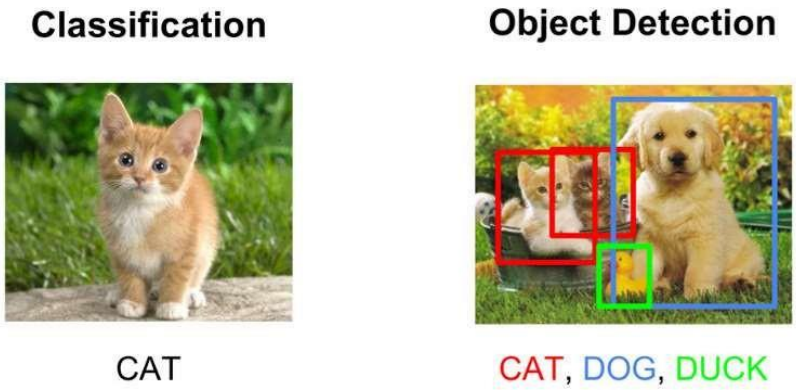
B. ALAT DAN BAHAN

1. PC/LAPTOP
2. Github
3. *Google Colaborator*

C. DASAR TEORI

C.1 Konsep Deteksi Objek

Deteksi objek (object detection) merupakan salah satu teknik visi computer (computer vision) yang berfungsi untuk mengidentifikasi dan menemukan objek dalam sebuah gambar atau video. Secara khusus, proses deteksi objek akan menggambar kotak pembatas di sekitar objek yang terdeteksi, sehingga dapat diketahui dimanakah lokasi objek tersebut pada gambar / video. Tujuan dari deteksi objek adalah untuk mendeteksi semua *instance* objek dari kelas yang telah diketahui, seperti orang, mobil atau wajah pada gambar. Umumnya, hanya sedikit objek yang terdapat pada sebuah gambar, tetapi terdapat banyak sekali jumlah kemungkinan lokasi dan skala di mana objek tersebut dapat muncul. Berikut ini adalah ilustrasi dari konsep deteksi objek:



Dalam sistem computer vision, 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. Setelah objek terdeteksi (misalnya, wajah), bisa didapatkan informasi lebih lanjut, diantaranya: (i) mengenali contoh spesifik (misalnya, untuk mengidentifikasi subjek dari wajah yang terdeteksi), (ii) melacak

objek pada urutan gambar (misalnya, untuk melacak wajah dalam video), dan (iii) mengekstrak informasi lebih lanjut tentang objek (misalnya, untuk menentukan jenis kelamin subjek). Beberapa contoh implementasi deteksi objek antara lain:

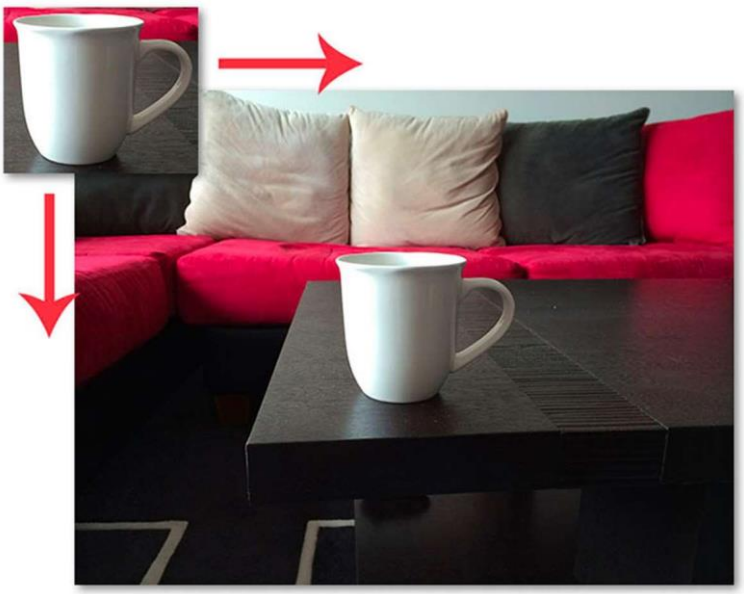
1. Video surveillance (CCTV), model deteksi objek mampu melacak banyak orang sekaligus, secara real-time
2. Crowd counting (Penghitungan kerumunan), untuk area padat pengunjung seperti taman hiburan, mall, atau alun-alun kota, deteksi objek dapat membantu pemilik bisnis dan pemerintahan kota untuk mengukur dengan lebih efektif berbagai jenis lalu lintas, dan beberapa rencana seperti jam operasional toko, shift karyawan, dan pengalokasian sumber daya.
3. Anomaly detection, misal di bidang pertanian, model pendeteksian objek dapat secara akurat mengidentifikasi dan menemukan lokasi potensi penyakit tanaman, sehingga memungkinkan petani untuk mendeteksi ancaman terhadap hasil panen mereka yang tidak dapat dilihat dengan mata telanjang.
4. Self-driving cars, deteksi objek merupakan Teknik yang mendasari terciptanya mobil tanpa pengemudi. Sistem ini harus dapat mengidentifikasi, menemukan, dan melacak objek di sekitarnya agar dapat bergerak di jalan raya dengan aman dan efisien.

C.2 Template Matching

Pencocokan template (template matching) merupakan bentuk yang paling dasar dari deteksi objek. 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:

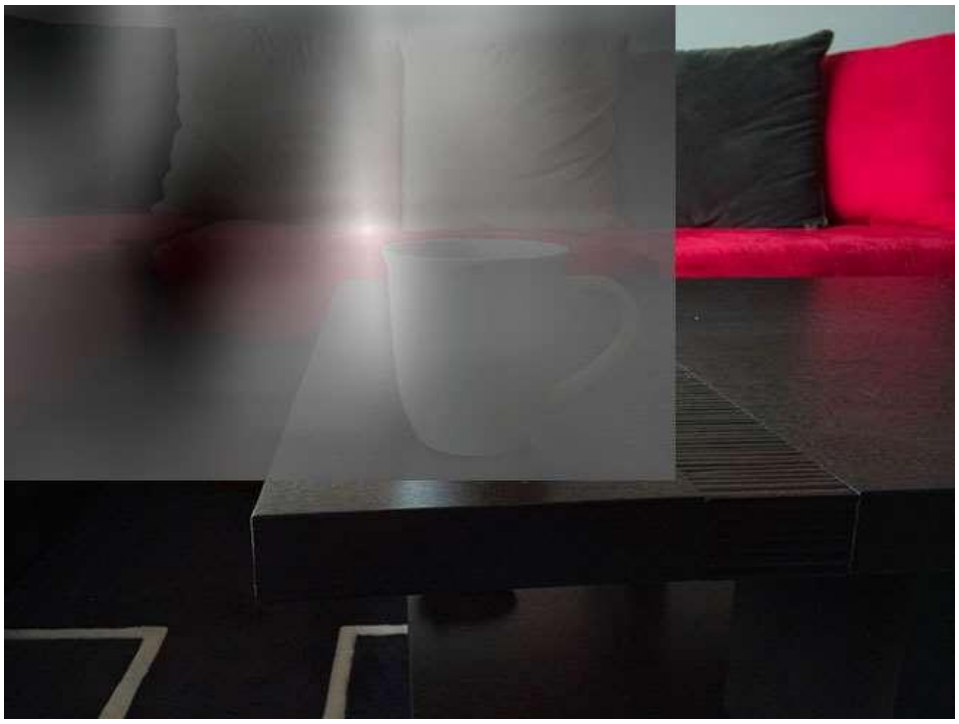
- Source image (I): gambar masukan yang diharapkan cocok dengan template.
- Template image (T): potongan objek yang akan dicari pada source image.

Untuk menemukan template di source image sumber, dilakukan perrgeseran template dari kiri ke kanan dan dari atas ke bawah:



Di setiap lokasi (x, y), sebuah metrik dihitung untuk merepresentasikan seberapa "baik" atau "buruk" kecocokan tersebut. Perhitungan koefisien korelasi dilakukan untuk

menentukan seberapa mirip intensitas piksel dari dua potongan gambar tersebut. Untuk setiap lokasi ***T*** pada ***I***, metrik hasil perhitungan akan disimpan dalam matriks hasil ***R***. Setiap koordinat (x, y) pada source image akan berisi elemen di matriks hasil ***R***:



Lokasi cerah dari matriks hasil R menunjukkan nilai kecocokan yang terbaik, sedangkan daerah gelap menunjukkan korelasi yang sangat kecil antara gambar sumber dan template.

OpenCV mengimplementasikan template matching menggunakan fungsi **matchTemplate()**. Terdapat 6 metode yang tersedia pada matchTemplate(), yaitu:

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

1. method=TM_SQDIFF
2. method=TM_SQDIFF_NORMED

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

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

3. method=TM_CCORR
4. method=TM_CCORR_NORMED

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

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

C.3 Edge Detection

Deteksi tepi (Edge detection) adalah operasi yang dijalankan untuk mendeteksi garis tepi (edges) yang membatasi dua wilayah citra yang memiliki tingkat kecerahan yang berbeda. Deteksi tepi sebuah citra digital merupakan proses untuk mencari perbedaan intensitas yang menyatakan batas-batas suatu objek (sub-citra) dalam keseluruhan citra digital yang dimaksud. Suatu titik (x,y) dikatakan sebagai tepi dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Deteksi tepi banyak dipakai untuk mengidentifikasi suatu objek dalam sebuah gambar. Terdapat 3 metode edge detection yang disediakan oleh OpenCV, yaitu:

1. Sobel Edge Detection
   Sobel adalah salah satu detektor tepi yang paling umum digunakan. Filter Sobel bekerja dengan menghitung gradien intensitas citra pada setiap piksel dalam sebuah citra. Saat menggunakan filter ini, gambar dapat diproses dalam arah X dan Y secara terpisah atau bersamaan. Sobel didasarkan pada konvolusi gambar dalam arah horizontal dan vertical menggunakan filter 3x3, berikut:

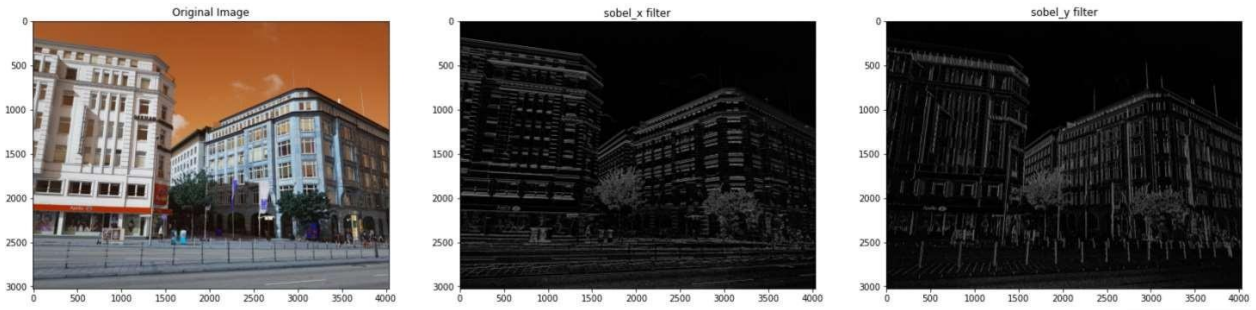
-1	0	+1
-2	0	+2
-1	0	+1

Gx

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

Gy

Contoh hasil dari sobel edge detection adalah sebagai berikut:



2. Laplacian Edge Detection

   Kernel 3x3 yang umumnya digunakan pada Laplacian Edge Detection adalah sebagai berikut:

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

   Metode ini sangat sensitive terhadap noise, maka dari itu untuk memperbaikinya, sebuah gambar umumnya akan dihaluskan terlebih dahuku menggunakan Gaussian filter sebelum menerapkan Laplacian. Contoh hasil dari laplacian edge detection adalah sebagai berikut:

3. Canny Edge Detection

   Canny Edge Detection merupakan metode yang paling umum digunakan dan paling efektif. Metode ini bekerja dengan 4 tahap, yaitu:

   a. Menghaluskan gambar dengan filter Gaussian untuk mengurangi noise
   b. Menghitung gradien menggunakan salah satu operator gradien Sobel
   c. Mengekstraksi titip tepi: Non-maximum suppression
   d. Menghubungkan dan thresholding: Hysteresis

Canny edge detection didasarkan pada konsep bahwa intensitas gambar akan bernilai tinggi di tepinya. Permasalahan pada konsep ini adalah jika gambar memiliki noise, maka noise tersebut juga akan terdeteksi sebagai tepi. Sehingga langkah pertama dalam Canny edge detection adalah menghilangkan noise, salah satu caranya dengan menerapkan filter Gaussian untuk menghaluskan gambar sebelum melanjutkan pemrosesan.

Langkah kedua dalam proses deteksi tepi dengan Canny adalah komputasi gradien yang dilakukan dengan menghitung tingkat perubahan intensitas (gradien) pada gambar di sepanjang arah gradien. Citra yang telah dihaluskan pada langkah pertama kemudian difilter dengan kernel Sobel, baik horizontal maupun vertikal untuk mendapatkan turunan pertama dalam arah horizontal (Gx) dan vertikal (Gy). Dari kedua gambar tersebut, dapat dicari gradien tepi dan arah untuk setiap piksel sebagai berikut:

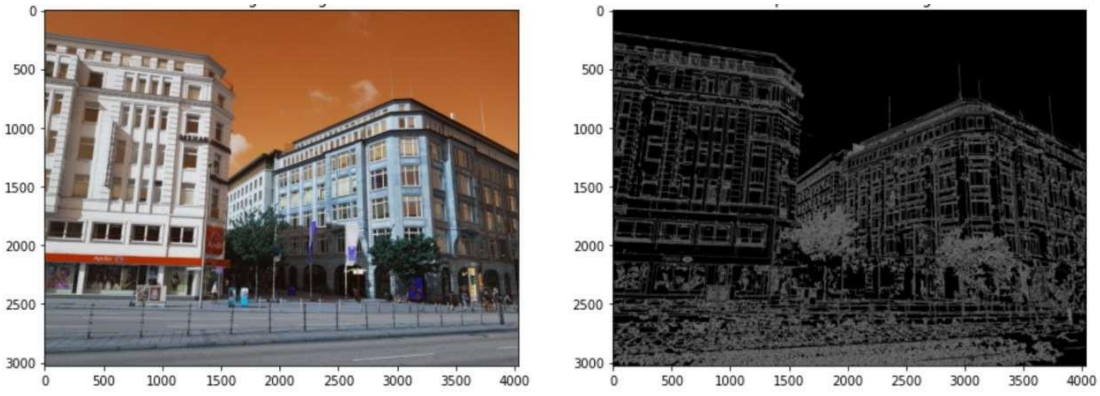
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Intensitas gambar berada pada titik tertinggi di tepinya, tetapi pada kenyataannya, intensitas tidak mencapai puncaknya pada satu piksel; sebaliknya, ada piksel bersebelahan dengan intensitas tinggi. Di setiap lokasi piksel, Canny edge detection akan membandingkan piksel dan memilih local maximal di 3X3 ketetanggaan ke arah gradien. Proses ini dikenal sebagai Non-maximum suppression.

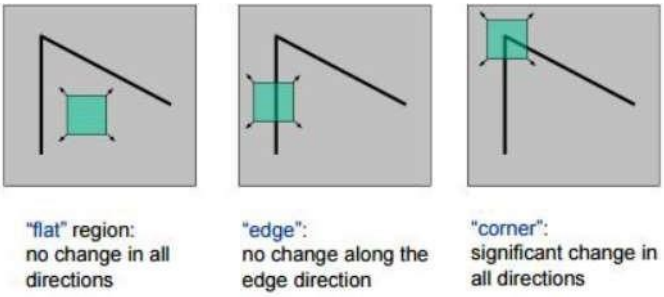
Langkah ketiga menghasilkan garis tepi berukuran tipis tetapi patah. Langkah terakhir adalah memperbaiki / menyambungkan tepi-tepi yang patah tersebut dengan menggunakan teknik hysteresis thresholding. Pada hysteresis thresholding, terdapat dua ambang batas: ambang batas tinggi dan ambang rendah (high and low threshold). Setiap piksel dengan nilai gradien lebih tinggi dari high threshold secara otomatis disimpan sebagai tepi. Sedangkan piksel yang gradiennya berada di antara high threshold dan low threshold akan ditangani dengan dua cara. Piksel akan diperiksa apakah memiliki kemungkinan koneksi dengan tepi; piksel akan disimpan jika tersambung dan akan dibuang/diabaikan jika sebaliknya. Piksel dengan gradien lebih rendah dari low threshold akan dibuang secara otomatis.

Contoh hasil dari canny edge detection adalah sebagai berikut:



C.4 Corner Detection

Sudut dapat diartikan sebagai persimpangan dua sisi. Titik sudut tidak bisa didefinisikan pada piksel tunggal, karena di sana hanya terdapat pada satu gradien di setiap titik. Gradien adalah arah perubahan intensitas kecerahan dalam suatu citra. Sudut adalah fitur penting dalam sebuah gambar, dan umumnya disebut sebagai interest point yang tidak berubah apabila dilakukan translasi, rotasi, dan iluminasi.



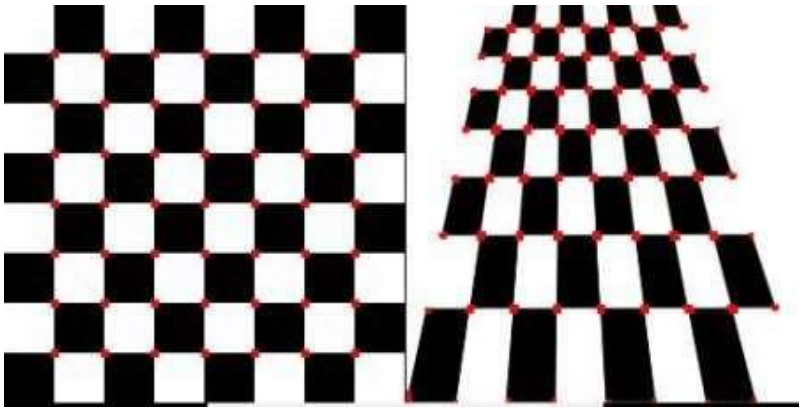
Terdapat 2 metode corner detection yang disediakan oleh OpenCV, yaitu:

1. Harris corner detection
   Harris Corner Detection adalah system pendeteksi sudut yang sering digunakan karena mampu menghasilkan nilai yang konsisten pada citra yang mengalami rotasi, penskalaan, variasi pencahayaan, maupun memiliki banya noise pada gambar. Deteksi sudut dengan metode Harris 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:

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

   Window function dapat berupa filter Gaussian yang memberi bobot pada piksel di bawahnya. Fungsi E (u, v) ini harus dimaksimalkan untuk deteksi sudut.
   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, ukurannya sama dengan citra input
   - blockSize: ukuran ketetanggaan
   - ksize: kernel size untuk operator Sobel
   - freeParameter: parameter bebas untuk Harris detector
   - borderType: metode pixel extrapolation

   Contoh hasil Harris corner detection adalah sebagai berikut:



Dokumentasi lengkap dapat dibaca pada link berikut:

https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html

2. Shi-Tomasi Corner Detection

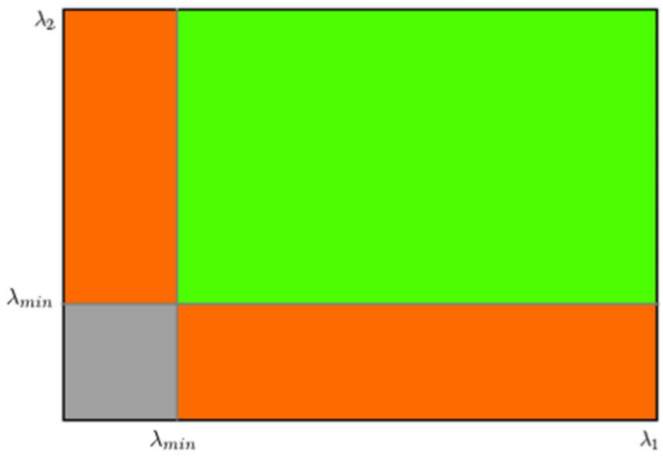
Shi-Tomasi Corner Detection menggunakan dasar bahwa sudut dapat dideteksi dengan mencari perubahan signifikan ke segala arah. Misal terdapat sebuah window kecil pada gambar kemudian memindai/scan seluruh gambar untuk mencari sudut. Menggeser window kecil ini ke segala arah akan menghasilkan perubahan besar pada tampilan, jika window tersebut terletak di sudut.
Pada ametode Harris, penilaian sebuah titik dikatakan sudut atau bukan dilakukan dengan memberikan nilai R

$$R = det(M) - k(\mathrm{Tr}(M))^2$$

Shi-Tomasi mengajukan ide untuk penilaian R hanya dengan menentukan nilai Eigen dari M saja

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

Jika nilainya lebih besar dari nilai threshold, maka dianggap sebagai sudut. Jika diplot di ruang λ1 − λ2, akan menjadi seperti berikut:

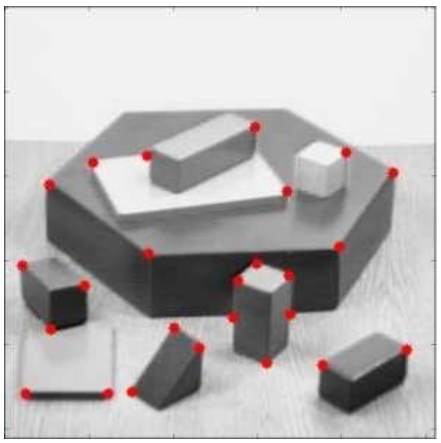


Dari gambar tersebut dapat dilihat bahwa hanya jika λ1 dan λ2 di atas nilai minimum, λmin dianggap sebagai sudut (area berwarna hijau).

OpenCV memiliki fungsi, cv.goodFeaturesToTrack(). Fungsi ini akan menemukan N sudut terkuat pada gambar dengan metode Shi-Tomasi. Citra input harus berupa gambar grayscale. Pengaturan
/ parameter dari metode ini antara lain: (a) jumlah sudut yang ingin ditemukan, (b) tingkat kualitas, yang merupakan nilai antara 0-1, yang menunjukkan kualitas minimum dari sudut, dan
(c) jarak euclidean minimum antar sudut yang terdeteksi. Dengan parameter-parameter tersebut, fungsi akan menemukan sudut-sudut pada gambar. Semua sudut di bawah tingkat kualitas akan ditolak. Sudut yang tersisa akan diurutkan berdasarkan kualitas dalam urutan menurun. Kemudian fungsi akan mengambil sudut terkuat pertama,

membuang semua sudut terdekat dalam kisaran jarak minimum dan mengembalikan N sudut terkuat.

Contoh hasil Harris corner detection adalah sebagai berikut:



Dokumentasi lengkap dapat dibaca pada link berikut:

https://docs.opencv.org/master/d4/d8c/tutorial_py_shi_tomasi.html

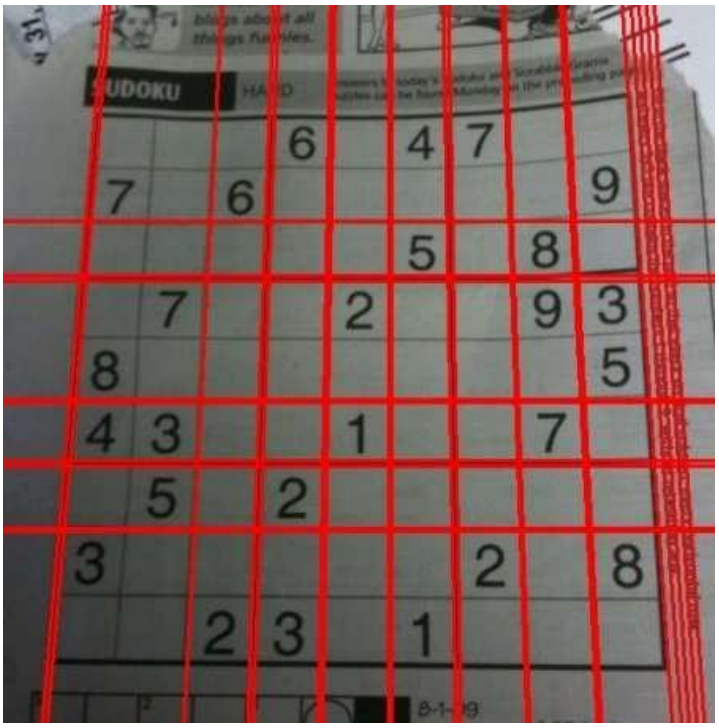
C.5 Grid Detection

Grid detection dapat dilakukan dengan metode Hough Transform. Hough Transform adalah teknik populer untuk mendeteksi bentuk apa pun, jika bentuk tersebut dapat direpresentasikan dalam bentuk matematika. Hough Transform dapat mendeteksi bentuk bahkan jika bentuknya rusak atau sedikit terdistorsi.

Tahapan untuk melakukan grid detection adalah sebagai berikut:

1. Menerapkan Canny Edge Detection
2. Dilasi gambar tepi (Canny bisa menemukan kedua tepi pemisah pada grid sebagai tepi yang berbeda, dilatasi dapat membuat kedua tepi ini bergabung lagi)
3. Erosi (setelah proses dilasi, garis border menjadi tebal sehingga Hough akan mendeteksi banyak garis)
4. Menerapkan HoughLines dengan fungsi cv2.HoughLines() pada OpenCV
5. Menggabungkan garis yang mirip (similar)

Contoh hasil Hough Transform adalah sebagai berikut:



Dokumentasi lengkap mengenai HoughTransform dapat dibaca pada link berikut:

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

C.6 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. Deteksi kontur adalah teknik yang berguna untuk analisis bentuk serta deteksi dan pengenalan objek. Kontur adalah kumpulan abstrak dari titik dan segmen yang sesuai dengan bentuk objek pada gambar. Kita dapat memanipulasi kontur dalam program kita seperti menghitung jumlah kontur, menggunakannya untuk mengkategorikan bentuk objek, atau memotong objek dari gambar (segmentasi gambar).

Untuk akurasi yang lebih baik, berikut ini merupakan tahapan untuk mendeteksi kontur pada gambar:

1. Ubah citra input menjadi citra biner (bisa didapat dengan mengimplementasikan thresholding atau canny edge detection).
2. Menemukan kontur menggunakan fungsi findContours() pada OpenCV.
3. Menggambarkan kontur pada citra input menggunakan fungsi

drawContours(). Fungsi findContours menerima tiga buah parameter, yaitu

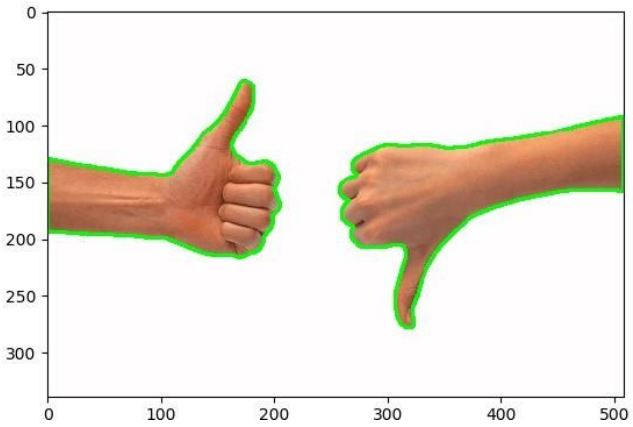
:

- Parameter pertama adalah Image yang akan dianalisis konturnya
- Parameter kedua menunjukkan metode retrieval. Metode retrieval tersebut dapat digunakan untuk melihat hierarcy dari kontur dalam sebuah citra. Hierarcy menunjukkan hubungan parent-child. Parents merupakan shape/contour yang lebih luar dibandingkan dengan child atau dengan kata

lain, parents merupakan shape/contour yang melingkupi child.

- Parameter ketiga menunjukkan metode yang digunakan untuk proses approximation dari shape. Methode tersebut dapat berupa metode cv2.CHAIN_APPROX_SIMPLE atau metode cv2.CHAIN_APPROX_NONE. Pada flag cv2.CHAIN_APPROX_SIMPLE, class cv2.findContours akan mengembalikan seminimum mungkin jumlah koordinat yang menunjukkan contours sedangkan pada flag cv2.CHAIN_APPROX_SIMPLE, class cv2.findContours akan mengembalikan sebanyak mungkin jumlah koordinat yang menunjukkan contours.

Contoh hasil contour detection adalah sebagai berikut:



Dokumentasi lengkap dapat dibaca pada link berikut:

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

D. LATIHAN PRAKTIKUM

Catatan: Untuk gambar pada praktikum ini menggunakan gambar pada link berikut:

https://drive.google.com/drive/folders/1d4U8FVnQ0Hq_K1Sy4XJvQsgq12ZjvmgK?usp=sharing

1 Buka https://colab.research.google.com/. Setelah dipastikan bahwa google Colab terhubung dengan Github Anda, buat notebook baru dan beri nama "Week11.ipynb". Kemudian import beberapa library dan akses folder yang ada di Drive Anda dengan cara sebagai berikut.

Week11.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

Object Detection

Metode yang digunakan:

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

```
[ ] #import library yang dibutuhkan
    import cv2 as cv
    import numpy as np
    from matplotlib import pyplot as plt
```

```
[ ] #akses drive
    from google.colab import drive

    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

2 Implementasikan 6 metode template matching pada OpenCV dengan menggunakan gambar cats_and_bunnies.jpg dan cat2_templatejpg.jpg sebagai templatenya.

Template Matching

Menggunakan library openCV:

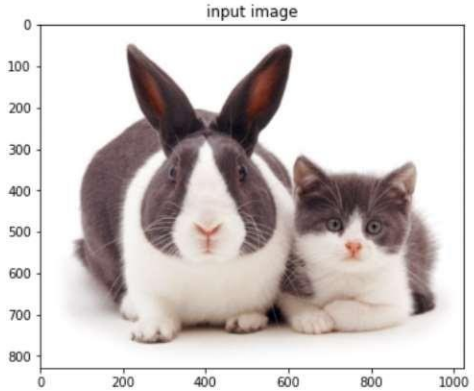
cv.matchTemplate(), dengan parameter:

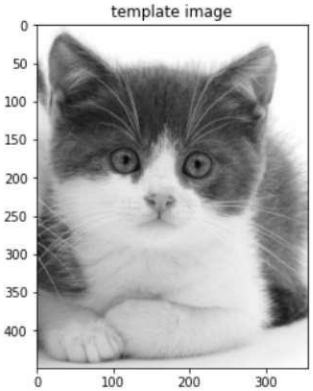
- **image**: citra input
- **templ**: template yang dicari, ukurannya tidak boleh lebih besar dari citra input
- **method**: metode dari template matching

Jenis metode template matching di OpenCV:

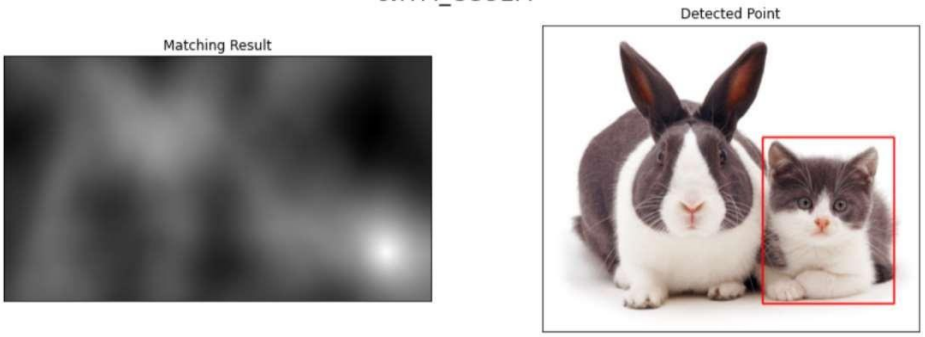
1. TM_SQDIFF
2. TM_SQDIFF_NORMED
3. TM_CCORR
4. TM_CCORR_NORMED
5. TM_CCOEFF
6. TM_CCOEFF_NORMED

Sehingga menghasilkan luaran seperti berikut:

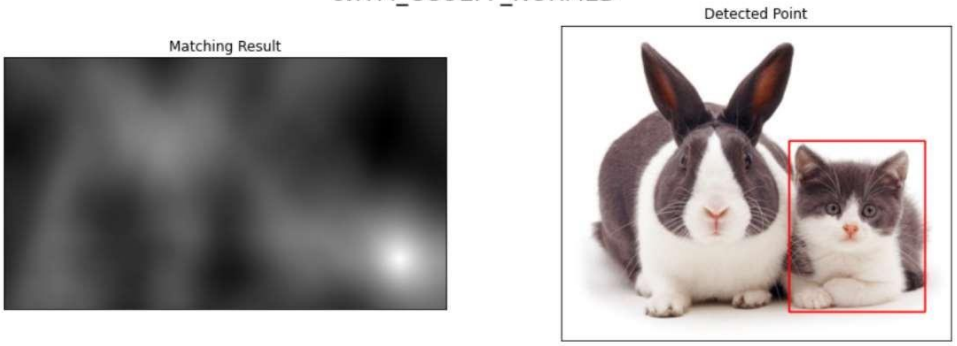




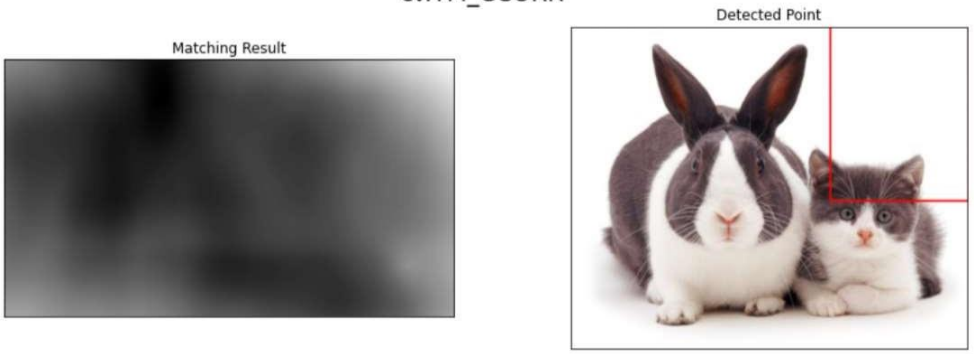
cv.TM_CCOEFF



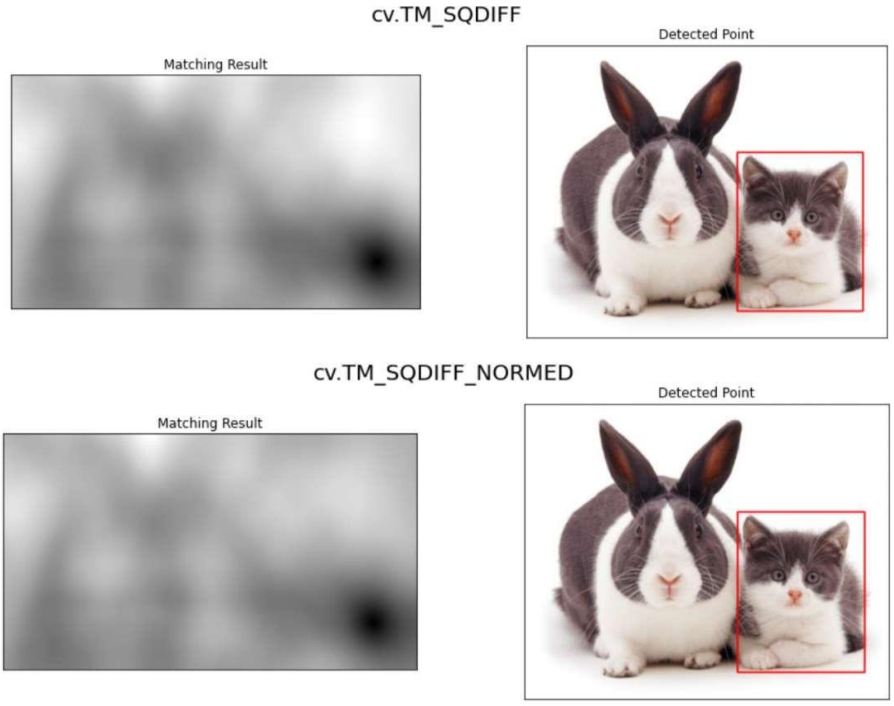
cv.TM_CCOEFF_NORMED



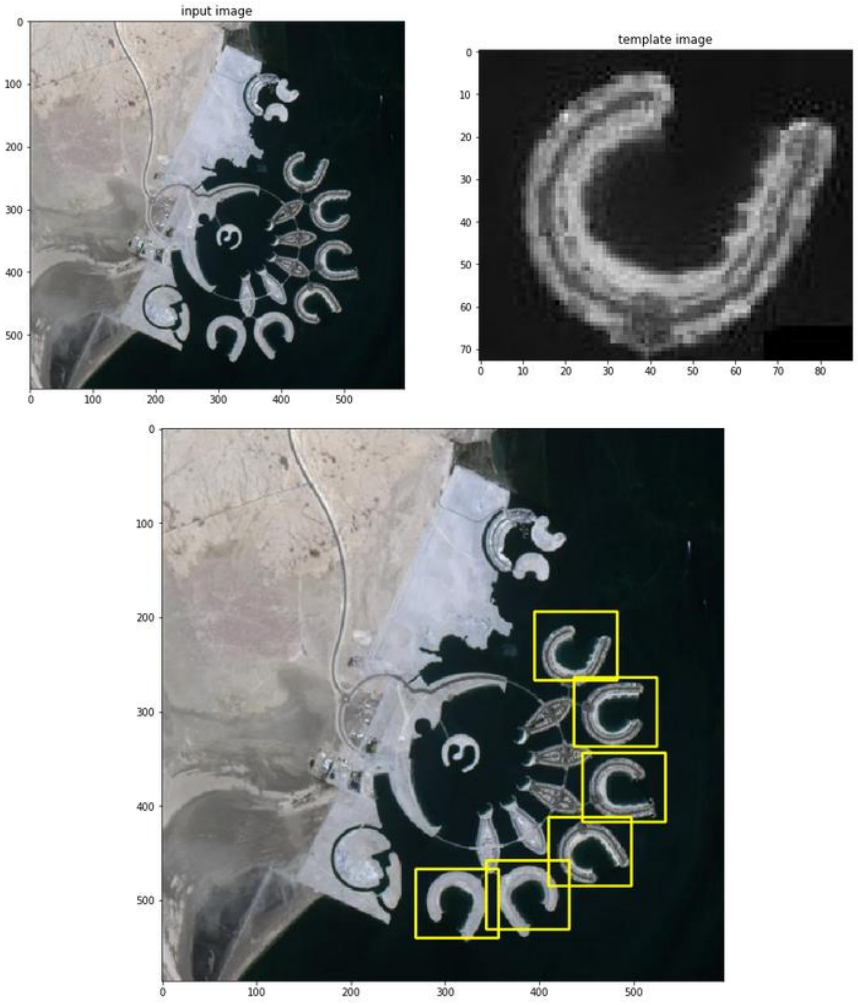
cv.TM_CCORR



cv.TM_CCORR_NORMED

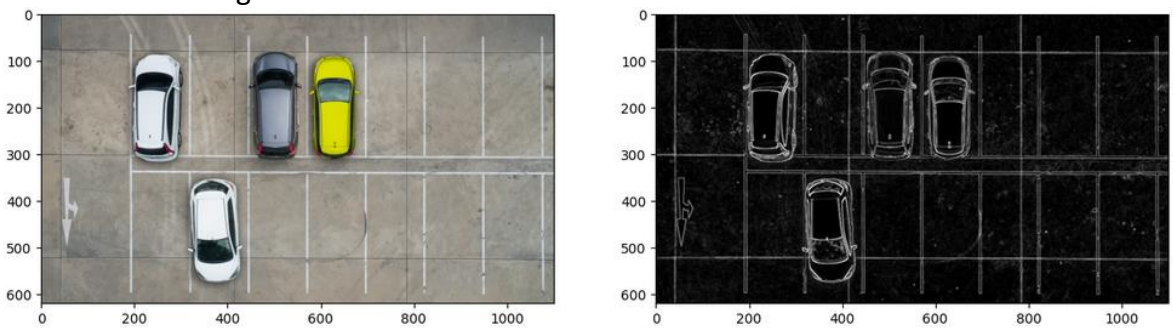


3 Implementasikan konsep template matching tanpa menggunakan library OpenCV untuk multiple object, menggunakan gambar bahrain.jpg untuk citra masukan dan bahrain-template.jpg sebagai citra template, sehingga menghasilkan output sebagai berikut:

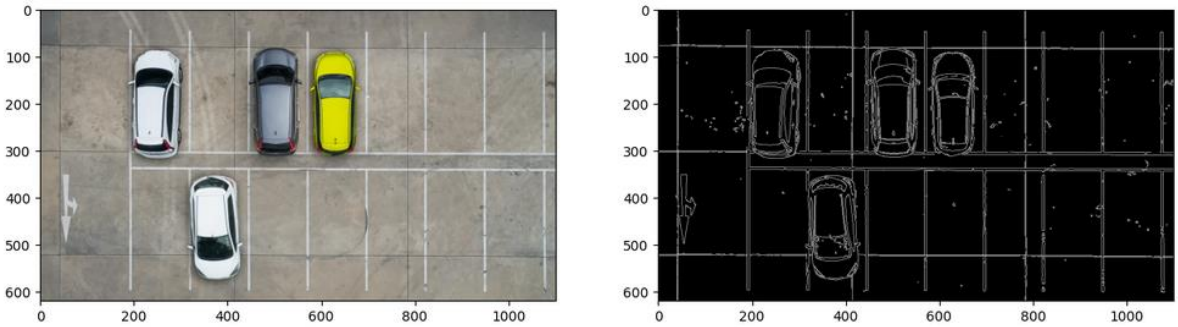


4 Implementasikan metode Sobel Edge Detection, Canny Edge Detection, dan Laplacian Edge Detection pada OpenCV dengan menggunakan gambar car-park.jpg, sehingga menghasilkan luaran sebagai berikut:

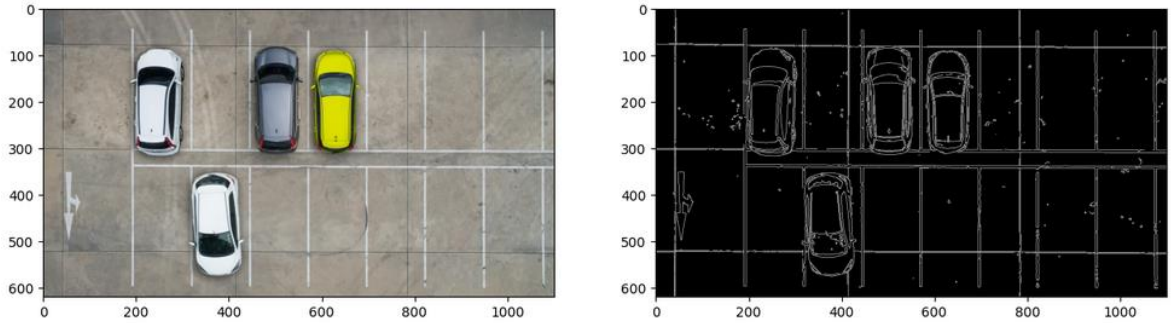
a. Sobel Edge Detection



b. Canny Edge Detection



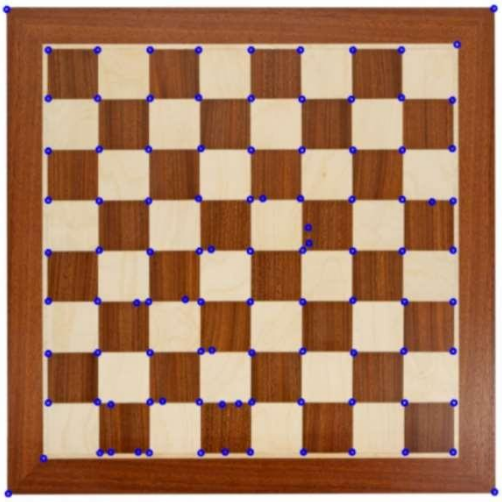
c. Laplacian Edge Detection



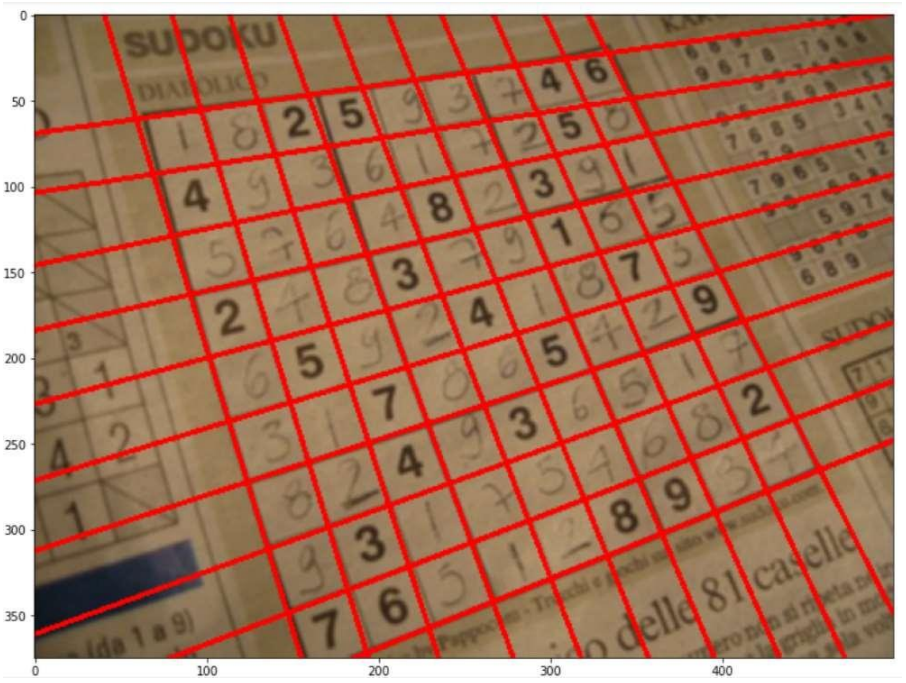
5 Implementasikan metode Harris Corner Detection dan Shi-Tomasi Detection pada OpenCV dengan menggunakan gambar chess-board.jpg, sehingga menghasilkan luaran sebagai berikut:

a. Harris Corner Detection

b. Shi-Tomasi Detection



6 Implementasikan metode Hough Transform pada OpenCV dengan menggunakan gambar sudoku.jpg. Tahapan proses grid detection sesuai yang terdapat pada ulasan teori, sehingga menghasilkan luaran sebagai berikut:



7. Implementasikan fungsi findContours() pada OpenCV untuk contour detection dengan menggunakan gambar laptop.jpg, sehingga menghasilkan luaran sebagai berikut:

E. Pengayaan Materi KTP

1. Siapkan file dataset, contoh pada praktik ini menggunakan Maftuchin Tuban


PROVINSI JAWA TIMUR
KABUPATEN TUBAN

NIK : 3523160606800003

Nama : MAFTUCHIN
Tempat/Tgl Lahir : TUBAN, 06-06-1980
Jenis Kelamin : LAKI-LAKI Gol. Darah : -
Alamat : JL. MASJID AL FALAH
RT/RW : 001 / 002
Kel/Desa : SUGIHARJO
Kecamatan : TUBAN
Agama : ISLAM
Status Perkawinan : KAWIN
Pekerjaan : WIRASWASTA
Kewarganegaraan : WNI
Berlaku Hingga : 06-06-2017

TUBAN
31-10-2012

2. Muat 2 library berikut sebagai tahapan persiapan

```
import cv2
import os
```

3. Tambahkan kode lokalisasi dataset dan lakukan pembacaan file name di dalam direktori dataset tersebut

```
# Lokasi hasil pelat
path_plate = "dataset/sliced"

# Looping file di direktori
for name_file in sorted(os.listdir(path_plate)):
  src = cv2.imread(os.path.join(path_plate, name_file))
  blurred = src.copy()
  gray = blurred.copy()
```

4. Masih pada looping file direktori, terapkan Gaussian Blur untuk setiap file supaya noise pickle dapat direduksi

```
# Filtering
for i in range(10):
  blurred = cv2.GaussianBlur(src, (5, 5), 0.5)
```

5. Lakukan konversi grayscale untuk setiap hasil citra yang tereduksi noise dan lanjutkan dengan binerisasi citra (thresholding)

```
# Ubah ke grayscale
gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY)

ret, bw = cv2.threshold(gray.copy(), 0, 255, cv2.THRESH_BINARY
+ cv2.THRESH_OTSU)
print(ret, bw.shape)
```

Hasil

6. Terapkan morfologi untuk setiap citra yang telah mengalami binerisasi supaya setiap karakter atau obyek tidak berhimpitan.

Hasil

7. Terapkan ekstraksi kontur untuk untuk mendapatkan semua kontur setiap karakter/obyek. Kontur harus diseleksi berdasarkan ukuran dan rasio sebuah karakter. Karakter/obyek di dalam KTP ini bervariasi ukuran dan rasionya, sehingga setiap karakter/obyek perlu penanganan yang berbeda. Contoh pada praktik ini berfokus pada karakter yang seukuran dengan NIK. Jika hasil ekstraksi kontur memperlihatkan hasil lain yaitu karakter lain terdeteksi, hal tersebut wajar karena perlu optimize code dan preprocessing yang lebih baik (karena hasil akuisisi setiap citra dapat berbeda).

Hasil

F. Tugas Praktikum

Setiap citra memiliki karakteristik yang berbeda. Terapkan praktik pada materi pengayaan KTP untuk setiap dataset KTP yang tersedia. Perhatikan kondisi awal dataset, terapkan preprocessing yang sesuai jika kualitas visual citra kurang memadai untuk segera diolah. Pengerjaan dilakukan secara berkelompok dengan aturan sebagai berikut:

Kelompok 1 – KTP Riyanto Sleman
Kelompok 2 – KTP Galang Raka Bengkulu
Kelompok 3 – KTP Arief Wijaya Cimahi
Kelompok 4 – KTP Edo Nias
Kelompok 5 – KTP Widiarso Bekasi
Kelompok 6 – KTP Abdurrauf Soppeng

--- SEMANGAT BELAJAR ---