PYTHON OPEN

Modul #14 : Pemrosesan Citra Real-Time

Nama Kelompok : Kelompok 3

Anggota :

1. Habib Ainur Ma'ruf / V3923010
2. Puput Surya Ningtyas / V3923015
3. Rossi Dwi Cahyo / V3923018

D3 Teknik Informatika PSDKU

Madiun

Penangkapan Video Real-time

Penangkapan video real-time adalah proses pengambilan gambar bergerak (video) secara langsung dari perangkat penangkap (seperti kamera) dan mengalirkannya untuk pemrosesan atau penyimpanan segera tanpa jeda yang signifikan. Ini memerlukan perangkat keras yang cepat dan perangkat lunak yang efisien untuk menangani aliran data yang terus-menerus.

Tujuan penangkapan video real-time:

1. Mengawasi area untuk mencegah dan mendeteksi aktivitas mencurigakan.
2. Video konferensi atau panggilan video untuk komunikasi real-time.
3. Mengalirkan acara atau kegiatan secara langsung ke audiens online.
4. Memantau dan mengontrol proses manufaktur.
5. Menggunakan kamera untuk operasi atau diagnostik medis secara langsung.

Manfaat mempelajari penangkapan video real-time:

1. Membantu dalam mengambil tindakan cepat dalam situasi darurat.
2. Memungkinkan pengawasan terus-menerus dan deteksi pelanggaran.
3. Memfasilitasi komunikasi yang lebih baik dalam pengaturan bisnis dan pribadi.
4. Mengurangi biaya dengan meminimalisir kebutuhan untuk kehadiran fisik.
5. Menyediakan data yang dapat segera digunakan untuk analisis dan keputusan.

Metode Penangkapan Video Real-Time

- Kamera yang terhubung langsung ke komputer melalui USB atau yang sudah terpasang pada laptop atau perangkat lainnya.
- Kamera yang terhubung melalui jaringan internet, memungkinkan akses dari lokasi yang berbeda.
- Menggunakan aplikasi pada smartphone untuk menangkap dan mengalirkan video secara real-time.
- Menggunakan kamera yang dipasang pada drone untuk menangkap video dari udara.
- Menggunakan perangkat keras khusus yang menangkap sinyal video dari sumber eksternal seperti kamera profesional.

Proses Penangkapan Video Real-Time

1. Hubungkan kamera ke komputer atau perangkat melalui USB, jaringan, atau sinyal nirkabel.
2. Pastikan perangkat keras dikenali oleh sistem operasi dan perangkat lunak yang digunakan.
3. Gunakan pustaka atau API seperti OpenCV, GStreamer, atau perangkat lunak lain untuk menangkap video.
4. Setel parameter seperti resolusi, frame rate, dan codec kompresi jika diperlukan.
5. Sistem menangkap frame secara terus-menerus dari kamera.
6. Setiap frame diproses dalam urutan yang sangat cepat untuk memastikan tidak ada penundaan yang signifikan.
7. Pemrosesan dasar seperti konversi warna, penyesuaian kecerahan, dan kontras.
8. Pemrosesan lanjutan seperti deteksi objek, pengenalan wajah, atau analisis lainnya jika diperlukan.
9. Menampilkan frame yang diambil pada layar secara real-time.
10. Menyimpan frame ke file video atau streaming ke jaringan menggunakan protokol seperti RTSP atau WebRTC.
11. Mengelola kondisi kesalahan seperti koneksi kamera terputus atau frame hilang.

12. Membersihkan sumber daya dengan benar setelah selesai (melepaskan perangkat, menutup jendela, dll).

Contoh Kasus

1. Sistem CCTV yang dapat mendeteksi pergerakan mencurigakan atau intrusi dan mengirimkan peringatan secara real-time.
2. Mobil yang mampu mengenali dan bereaksi terhadap rambu lalu lintas, pejalan kaki, dan kondisi jalan lainnya.
3. Aplikasi yang menambahkan informasi atau objek digital ke dalam pemandangan dunia nyata secara real-time.
4. Penggunaan pemrosesan citra dalam prosedur medis seperti endoskopi, di mana gambar diolah dan ditampilkan secara real-time untuk membantu dokter.

Implementasi kode

mengimpor semua pustaka, seperti pandas dan panda. Kemudian mengimpor fungsi cv2, time, dan DateTime dari modul DateTime.

```
# Importing the Pandas libraries
import pandas as panda

# Importing the OpenCV libraries
import cv2

# Importing the time module
import time

# Importing the datetime function of the datetime module
from datetime import datetime
```

menginisialisasi beberapa variabel yang akan digunakan lebih lanjut dalam kode. Mendefinisikan keadaan awal sebagai "Tidak Ada" dan akan menyimpan gerakan yang dilacak dalam variabel lain motionTrackList. Mendefinisikan daftar 'motionTime' untuk menyimpan waktu saat gerakan terdeteksi dan menginisialisasi daftar dataFrame menggunakan modul panda.

```
# Assigning our initial state in the form of variable initialState as None for initial frames
initialState = None

# List of all the tracks when there is any detected of motion in the frames
motionTrackList= [ None, None ]

# A new list 'time' for storing the time when movement detected
motionTime = []

# Initialising DataFrame variable 'dataFrame' using pandas libraries panda with Initial and Final column
dataFrame = panda.DataFrame(columns = ["Initial", "Final"])
```

Melakukan langkah-langkah deteksi gerakan utama. Mari kita pahami secara bertahap:

1. Pertama, kita akan mulai merekam video menggunakan modul cv2 dan menyimpannya dalam variabel video.
2. Kemudian kita akan menggunakan loop while tak terhingga untuk menangkap setiap frame dari video.
3. Kita akan menggunakan metode read() untuk membaca setiap frame dan menyimpannya ke dalam variabel masing-masing.
4. Kami mendefinisikan gerakan variabel dan menginisialisasinya ke nol.
5. Kami membuat dua variabel lagi grayImage dan grayFrame menggunakan fungsi cv2 cvtColor dan GaussianBlur untuk menemukan perubahan dalam gerakan.
6. Jika initialState kita adalah None maka kita tetapkan grayFrame saat ini ke initialState sebaliknya dan menghentikan proses selanjutnya dengan menggunakan kata kunci 'continue'.
7. Pada bagian selanjutnya, kami menghitung perbedaan antara frame awal dan frame skala abu-abu yang kami buat pada iterasi saat ini.
8. Kemudian kami akan menyorot perubahan antara frame awal dan frame saat ini menggunakan fungsi ambang batas cv2 dan fungsi dilate.
9. Kita akan menemukan kontur dari objek bergerak pada gambar atau bingkai saat ini dan menunjukkan objek bergerak tersebut dengan membuat batas hijau di sekitarnya dengan menggunakan fungsi persegi panjang.
10. Setelah ini, kita akan menambahkan motionTrackList dengan menambahkan elemen yang terdeteksi saat ini ke dalamnya.
11. Kami telah menampilkan semua bingkai seperti skala abu-abu dan bingkai asli, dll., dengan menggunakan metode imshow.
12. Selain itu, kami membuat kunci menggunakan metode witkey() dari modul cv2 untuk mengakhiri proses, dan kami dapat mengakhiri proses kami dengan menggunakan kunci 'm'.

```python
video = cv2.VideoCapture(0)

while True:
    # Read the current frame
    check, cur_frame = video.read()

    # Convert to grayscale and apply Gaussian blur
    gray_image = cv2.cvtColor(cur_frame, cv2.COLOR_BGR2GRAY)
    gray_frame = cv2.GaussianBlur(gray_image, (21, 21), 0)

    # Handle first frame
    if initialState is None:
        initialState = gray_frame
        continue

    # Calculate difference between initial and current frame
    differ_frame = cv2.absdiff(initialState, gray_frame)

    # Threshold the difference frame
    thresh_frame = cv2.threshold(differ_frame, 30, 255, cv2.THRESH_BINARY)[1]
    thresh_frame = cv2.dilate(thresh_frame, None, iterations=2)

    # Find contours in the thresholded frame
    cont, _ = cv2.findContours(thresh_frame.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Detect and mark moving objects
    for cur in cont:
        if cv2.contourArea(cur) < 10000:
            continue
        (cur_x, cur_y, cur_w, cur_h) = cv2.boundingRect(cur)
        cv2.rectangle(cur_frame, (cur_x, cur_y), (cur_x + cur_w, cur_y + cur_h), (0, 255, 0), 3)

    # Track motion history
    motionTrackList.append(1 if any(cv2.contourArea(c) > 10000 for c in cont) else 0)
    motionTrackList = motionTrackList[-2:]

    # Record motion start and end times
    if motionTrackList[-1] == 1 and motionTrackList[-2] == 0:
        motionTime.append(datetime.datetime.now())
    if motionTrackList[-1] == 0 and motionTrackList[-2] == 1:
        motionTime.append(datetime.datetime.now())

    # Display results
    cv2.imshow("Gray Frame", gray_frame)
    cv2.imshow("Difference", differ_frame)
    cv2.imshow("Thresholded Frame", thresh_frame)
    cv2.imshow("Color Frame", cur_frame)

    # Wait for 'm' key to exit
    wait_key = cv2.waitKey(1)
    if wait_key == ord('m'):
        break
```

Setelah menutup loop, kami akan menambahkan data kami dari dataFrame dan daftar motionTime ke dalam file CSV dan terakhir mematikan video.

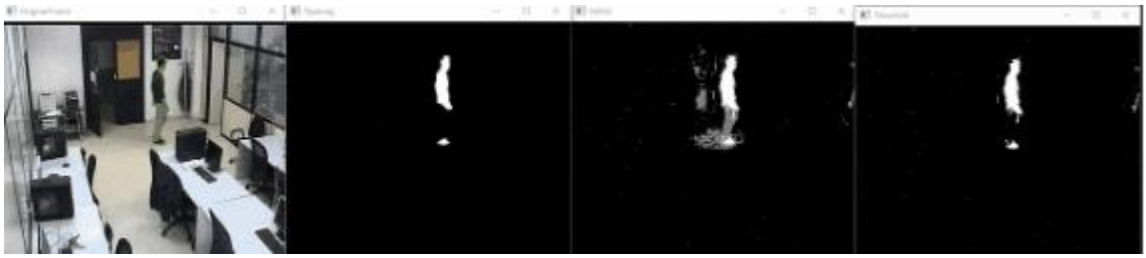
```python
# At last we are adding the time of motion or var_motion inside the data frame
for a in range(0, len(motionTime), 2):

    dataFrame = dataFrame.append({"Initial" : time[a], "Final" : motionTime[a + 1]}, ignore_index = True)


# To record all the movements, creating a CSV file
dataFrame.to_csv("EachMovement.csv")

# Releasing the video
video.release()

# Now, Closing or destroying all the open windows with the help of openCV
cv2.destroyAllWindows()
```

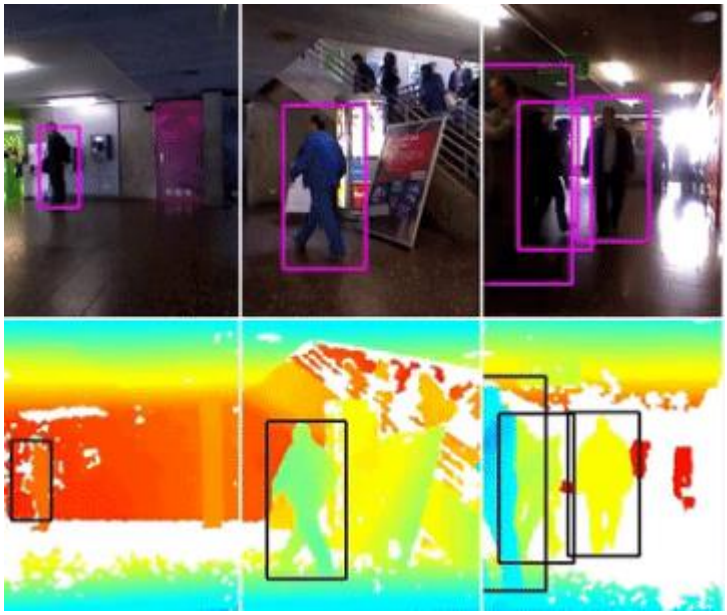
merekam video menggunakan webcam perangkat kami, kemudian mengambil frame awal dari video masukan sebagai referensi dan memeriksa frame berikutnya dari

waktu ke waktu. Jika ditemukan bingkai yang berbeda dari yang pertama, maka terdapat gerakan. Itu akan ditandai di kotak hijau.

Hasil

kita dapat melihat bahwa gerakan pria dalam video tersebut telah terlacak. Dengan demikian, outputnya bisa terlihat sesuai. Namun, dalam kode ini, pelacakan akan dilakukan dengan bantuan kotak persegi panjang di sekitar objek bergerak, mirip dengan yang bisa dilihat di bawah. Hal yang menarik untuk diperhatikan di sini adalah bahwa video tersebut merupakan rekaman kamera keamanan yang telah dilakukan deteksi.





Referensi

https://chatgpt.com/

https://www.kdnuggets.com/2022/08/perform-motion-detection-python.html

Aplikasi Real-Time Deteksi Gerakan

Deteksi gerakan real-time adalah teknologi yang memungkinkan sistem untuk secara otomatis mengenali dan merespons perubahan di lingkungannya berdasarkan pergerakan objek atau orang. Teknologi ini umumnya digunakan dalam keamanan, pemantauan, atau aplikasi yang memerlukan respons cepat terhadap perubahan dalam situasi yang dinamis.

Tujuan

1. Mendeteksi intrusi atau aktivitas mencurigakan dalam area tertentu.
2. Pemantauan lalu lintas, penggunaan fasilitas umum, atau aktivitas di tempat kerja.
3. Memicu respons otomatis seperti pemberitahuan atau tindakan berdasarkan deteksi gerakan.
4. Mengumpulkan data tentang perilaku dan pola gerakan untuk analisis lebih lanjut.

Manfaat

1. Sistem dapat merespons secara instan terhadap perubahan situasi.
2. Memungkinkan pengawasan yang terus-menerus tanpa perlu kehadiran manusia secara konstan.
3. Mencegah kejadian negatif dengan mendeteksi gerakan mencurigakan.
4. Mengotomatiskan proses berdasarkan pergerakan atau kehadiran orang atau objek.
5. Mengumpulkan data untuk analisis perilaku dan kebutuhan perencanaan lebih lanjut.

Metode

1. Menggunakan teknik seperti pengubahan warna ke skala abu-abu, Gaussian Blur, dan thresholding untuk memproses gambar.
2. Mencari objek bergerak dengan mengidentifikasi kontur dalam frame gambar.
3. Seperti algoritma perbedaan frame (frame differencing) atau penggunaan model machine learning untuk deteksi objek.

Proses

1. Mulai penangkapan video dari sumber (misalnya webcam atau kamera CCTV).
2. Membaca setiap frame, mengubahnya ke skala abu-abu, dan menerapkan operasi seperti Gaussian Blur untuk mengurangi noise.
3. Membandingkan frame saat ini dengan frame awal (initialState), mengekstraksi perbedaan, dan menerapkan threshold untuk menyoroti perubahan signifikan.
4. Menggunakan metode deteksi kontur untuk mengidentifikasi dan menandai objek bergerak.
5. Menerapkan respons seperti menampilkan kotak di sekitar objek bergerak atau memicu tindakan lainnya.
6. Opsional, menyimpan data atau waktu terjadinya gerakan untuk analisis lebih lanjut.

Contoh Kasus

Salah satu studi kasus yang umum adalah penggunaan deteksi gerakan untuk keamanan rumah pintar. Sistem ini menggunakan kamera CCTV untuk mendeteksi gerakan di sekitar rumah dan memberi notifikasi ke pemilik rumah melalui aplikasi jika ada aktivitas mencurigakan.

Implementasi Kode

```python
import cv2

# Inisialisasi cascade classifier
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Inisialisasi webcam
cap = cv2.VideoCapture(0)

while True:
    # Ambil frame dari kamera
    ret, frame = cap.read()

    # Konversi ke citra keabuan
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Deteksi wajah dalam frame
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, minSize=(30, 30))

    # Gambar kotak di sekitar wajah yang terdeteksi
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Tampilkan hasil
    cv2.imshow('PEOPLE DETECTED!!!', frame)

    # Keluar dari loop jika tombol 'q' ditekan
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Bebaskan sumber daya
cap.release()
cv2.destroyAllWindows()

```

Hasil

Referensi

https://youtu.be/kSUl4tJAvwY?si=QrC-DPs2L-zxL0Xd

Aplikasi Real-Time Pelacakan Objek

Aplikasi real-time pelacakan objek adalah sistem yang menggunakan teknologi pemrosesan citra atau video untuk mengidentifikasi dan mengikuti pergerakan objek secara langsung saat objek tersebut bergerak dalam suatu ruang.

Tujuan utama dari aplikasi real-time pelacakan objek adalah untuk secara akurat dan efisien melacak pergerakan objek dalam berbagai konteks, seperti pengawasan keamanan, analisis perilaku, navigasi kendaraan otonom, dan lain sebagainya.

Manfaat dari aplikasi real-time pelacakan objek antara lain:

- Meningkatkan kemampuan pengawasan dan deteksi intrusi dalam sistem keamanan.
- Mendukung otomatisasi dan pengambilan keputusan berdasarkan lokasi dan perilaku objek.
- Mengurangi waktu yang diperlukan untuk pencarian dan identifikasi objek secara manual.
- Memungkinkan untuk menganalisis pola dan perilaku objek yang bergerak.

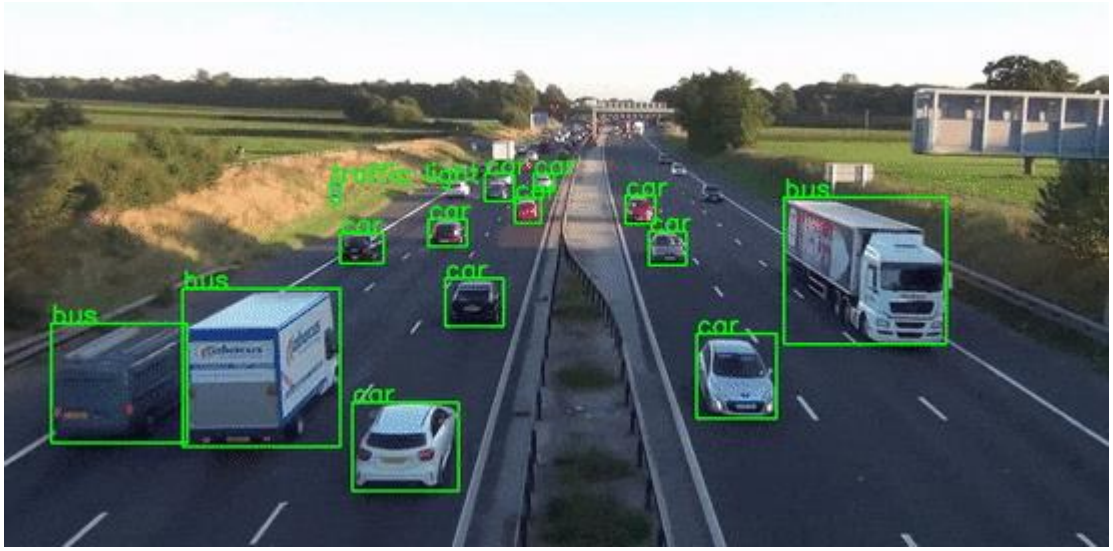
Contoh Kasus

Sebagai contoh, aplikasi real-time pelacakan objek digunakan dalam sistem pengawasan bandara untuk melacak tas yang ditinggalkan atau pergerakan yang mencurigakan, dalam industri otomotif untuk mengawasi proses produksi dan pergerakan barang, serta dalam navigasi kendaraan otonom untuk menghindari tabrakan dan menjaga jarak dengan objek lain di jalan.

Implementasi kode

```python
import cv2
import pafy
import torch
import numpy as np
from time import time

# URL video YouTube
URL = "https://www.youtube.com/watch?v=dQw4w9WgXcQ"

# Mengambil video dari YouTube
video = pafy.new(URL)
best = video.getbest(preftype="mp4")

# Membuat aliran video dengan OpenCV
cap = cv2.VideoCapture(best.url)

# Menggunakan model YOLOv5
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

def score_frame(frame, model):
    device = 'cuda' if torch.cuda.is_available() else 'cpu'
    model.to(device)
    frame = [torch.tensor(frame).to(device)]
    results = model(frame)
    labels = results.xyxyn[0][:, -1].cpu().numpy()
    cords = results.xyxyn[0][:, :-1].cpu().numpy()
    return labels, cords

def plot_boxes(results, frame):
    labels, cords = results
    n = len(labels)
    x_shape, y_shape = frame.shape[1], frame.shape[0]
    for i in range(n):
        row = cords[i]
        if row[4] < 0.2:  # Jika skor kurang dari 0.2, abaikan prediksi
            continue
        x1 = int(row[0] * x_shape)
        y1 = int(row[1] * y_shape)
        x2 = int(row[2] * x_shape)
        y2 = int(row[3] * y_shape)
        bgr = (0, 255, 0)  # Warna kotak
        class_name = model.names[int(labels[i])]  # Nama kelas
        label_font = cv2.FONT_HERSHEY_SIMPLEX  # Font untuk label
        cv2.rectangle(frame, (x1, y1), (x2, y2), bgr, 2)  # Plot kotak
        cv2.putText(frame, class_name, (x1, y1 - 10), label_font, 0.9, bgr, 2)  # Letakkan label di atas kotak
    return frame

# Video writer untuk menyimpan output
out_file = "output.avi"
x_shape = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
y_shape = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
four_cc = cv2.VideoWriter_fourcc(*"MJPG")
out = cv2.VideoWriter(out_file, four_cc, 20, (x_shape, y_shape))

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    start_time = time()
    results = score_frame(frame, model)  # Skor frame
    frame = plot_boxes(results, frame)  # Plot kotak
    end_time = time()
    fps = 1 / np.round(end_time - start_time, 3)  # Mengukur FPS
    print(f"Frames Per Second : {fps}")
    out.write(frame)  # Tulis frame ke output
    cv2.imshow('Object Tracking', frame)  # Tampilkan hasil

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Bebaskan sumber daya
cap.release()
out.release()
cv2.destroyAllWindows()
```

Hasil



Referensi

https://towardsdatascience.com/implementing-real-time-object-detection-system-using-pytorch-and-opencv-70bac41148f7