Nhận diện cử chỉ tay thời gian thực sử dụng MediaPipe và Random Forest

Nguyễn Thanh Hòa
SE183091
Nguyễn Quý Toàn
SE182785
Lê Minh Hùng
SE182706

July 2024

1 Giới Thiệu

Nhận diện cử chỉ tay đã thu hút được sự chú ý đáng kể do tiềm năng ứng dụng trong tương tác người với máy tính,giao tiếp bằng ngôn ngữ ký hiệu và thực tế ảo. Báo cáo này trình bày về hệ thống nhận diện cử chỉ tay theo thời gian thực ,sử dụng MediaPipe để theo dõi bàn tay và bộ phân loại Random Forest cho việc phân loại cử chỉ, không dựa vào các kỹ thuật học sâu.

2 Phạm vi

2.1 Phạm vi của chủ đề

Phạm vi của báo cáo này là khám phá và triển khai hệ thống nhận dạng cử chỉ tay theo thời gian thực sử dụng MediaPipe để theo dõi bàn tay và Random Forest để phân loại cử chỉ.

2.2 Định nghĩa vấn đề

Nhận dạng cử chỉ tay là một lĩnh vực nghiên cứu quan trọng với nhiều ứng dụng thực tế khác nhau, bao gồm tương tác giữa người và máy tính, giao tiếp ngôn ngữ ký hiệu và thực tế ảo. Vấn đề chính được đề cập trong báo cáo này là phát triển một hệ thống đáng tin cậy và hiệu quả để nhận dạng cử chỉ tay trong thời gian thực.

3 Kiến Trúc Hệ Thống

Hệ thống của chúng tôi xây dựng theo kiến trúc hai giai đoạn:tích hợp khả năng theo dõi tay mạnh mẽ của MediaPipe với sức mạnh phân loại của bộ phân loại Random Forest.

3.1 Giai đoạn theo dõi bàn tay: Sử dụng MediaPipe Hands để theo dõi chuyển động của bàn tay.Framework của MediaPipe Hands

MediaPipe Hands là một công cụ hiệu quả và chính xác để theo dõi bàn tay,cung cấp 21 điểm mốc 2.5D đại diện cho các khớp xương của bàn tay. (sau khi MediaPipe Hands tìm thấy bàn tay sẽ tự tạo một khung xương 2.5D giúp mô tả chính xác cấu trúc và vị trí của bàn tay trong không gian).Sử dụng hai mô hình:

1. Bộ phát hiện lòng bàn tay: xác định vị trí ban đầu của tay và vẽ ra một hình hộp bao quanh tay và xoay theo hướng của bàn tay.
2. Mô hình dự đoán điểm khớp: 21 điểm khớp chi tiết của tay trong vùng lòng bàn tay đã phát hiện.

MediaPipe Hands cho phép theo dõi tay hiệu quả và chính xác và được ứng dụng rất nhiều như điều khiển máy tính bằng cử chỉ ,chơi game và điều khiển robot bằng cử chỉ.

3.2 Random Forest Classifier

Random Forest là một phương pháp học tổng hợp xây dựng nhiều cây quyết định trong quá trình huấn luyện. Bộ phân loại sau đó dự đoán bằng cách xem xét số phiếu đa số được bầu từ tất cả các cây quyết định. Random Forest có độ bền, độ chính xác cao, và khả năng xử lý dữ liệu có chiều cao.

4 Thu thập và Xử lý Dữ liệu

4.1 Thu thập Dữ liệu

Để huấn luyện bộ phân loại Random Forest, nhóm đã thu thập một bộ dữ liệu các cử chỉ tay :

1. Capture Hand Gestures:Ghi lại các khung hình video của các cá nhân thực hiện các cử chỉ tay khác nhau.
2. Labeling:Gán nhãn thủ công cho từng khung hình với loại cử chỉ tương ứng.
3. Extract Landmarks:Sử dụng MediaPipe Hands để trích xuất các điểm khớp 2.5D của tay từ mỗi khung hình. Giải thích thuật ngữ :(Khung hình là từng ảnh tĩnh riêng được tạo ra từ video)

4.1.1 Code: 'create_dataset.py'

```python
import os
import pickle

import mediapipe as mp
import cv2
import matplotlib.pyplot as plt


mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

DATA_DIR = './data'

data = []
labels = []
for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []

        x_ = []
        y_ = []

        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        results = hands.process(img_rgb)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y

                    x_.append(x)
                    y_.append(y)

                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    data_aux.append(x - min(x_))
                    data_aux.append(y - min(y_))

            data.append(data_aux)
            labels.append(dir_)

f = open('data.pickle', 'wb')
pickle.dump({'data': data, 'labels': labels}, f)
f.close()
```

4.2 Xử lý Dữ liệu

Các điểm đặc trưng đã được trích xuất, sau đó được xử lý trước để chuẩn bị cho việc huấn luyện: (các điểm đặc trưng trên bàn tay, ví dụ như đầu ngón tay, cổ tay, khớp ngón tay,...)

1. Normalization:Chuẩn hóa các tọa độ điểm đặc trưng để loại bỏ sự khác biệt về kích thước và vị trí của tay.

2. Feature Vector:Chuyển đổi các điểm đặc trưng đã chuẩn hóa thành một vector đặc trưng, thường là một mảng 1D.Chúng tôi sử dụng các đặc trưng này để phân biệt giữa các cử chỉ khác nhau.

5 Huấn luyện phân loại (Classifier Training)

Dữ liệu được xử lý trước khi được sử dụng để huấn luyện bộ phân loại Random Forest:

1. Lựa chọn mô hình: Chọn các tham số thích hợp cho Random Forest, các tham số tối ưu cho Random Forest Classifier như số lượng cây, độ sâu tối đa của cây, ...

2. Huấn luyện: Huấn luyện bộ phân loại về các đặc điểm mốc được gắn nhãn bằng cách sử dụng kỹ thuật xác thực chéo để đánh giá mô hình.

3. Đánh giá: Đánh giá hiệu suất của mô hình bằng cách sử dụng các số liệu như độ chính xác(Tỷ lệ dự đoán chính xác trên tổng số dự đoán),Precision(Tỷ lệ dự đoán dương tính chính xác trong số tất cả các dự đoán)...

5.0.1 Code: ‘collect_data.py‘

```python
import cv2
import os

def extract_frames(video_paths, output_dir, num_frames=100):
    for idx, video_path in enumerate(video_paths, start=1):
        # Tạo thư mục con trong thư mục data
        video_folder = os.path.join(output_dir, f"{idx}")
        if not os.path.exists(video_folder):
            os.makedirs(video_folder)

        # Đối tượng capture video từ file
        cap = cv2.VideoCapture(video_path)

        # Kiểm tra xem video có mở thành công hay không
        if not cap.isOpened():
            print(f"Không thể mở video: {video_path}")
            continue

        frame_count = 0
        while frame_count < num_frames:
            # Đọc từng khung hình
            ret, frame = cap.read()

            # Kiểm tra nếu không còn khung hình nào để đọc
            if not ret:
                break

            # Lưu khung hình thành file ảnh
            frame_filename = os.path.join(video_folder, f"frame_{frame_count:04d}.png")
            cv2.imwrite(frame_filename, frame)

            frame_count += 1

        # Giải phóng tài nguyên
        cap.release()

# Đường dẫn đến các video đầu vào và thư mục đầu ra
video_paths = [
    r'C:\Users\Junn\Desktop\FPT_study\CPV\Hand_gesture\bao.mp4',
    r'C:\Users\Junn\Desktop\FPT_study\CPV\Hand_gesture\keo.mp4',
    r'C:\Users\Junn\Desktop\FPT_study\CPV\Hand_gesture\bua.mp4'
]
output_dir = 'data'  # Thư mục để lưu các thư mục con của từng video

# Số lượng frame muốn lấy cho mỗi video
num_frames_per_video = 100

# Gọi hàm để tách video và lưu các frame vào các thư mục con
extract_frames(video_paths, output_dir, num_frames=num_frames_per_video)
```

6 Suy luận và nhận dạng cử chỉ

Sau khi huấn luyện bộ phân loại Rừng ngẫu nhiên, nó có thể được sử dụng để nhận dạng cử chỉ theo thời gian thực:

1. Theo dõi bàn tay (Hand Tracking): MediaPipe Hands liên tục theo dõi bàn tay của người dùng trong thời gian thực.
2. Trích xuất mốc (Landmark Extraction): MediaPipe trích xuất các mốc từ bàn tay được theo dõi.
3. Trích xuất đặc điểm (Feature Extraction): Các mốc được xử lý trước để tạo ra một vectơ đặc trưng.
4. Dự đoán cử chỉ (Gesture Prediction): Trình phân loại Random Forest dự đoán cử chỉ có khả năng xảy ra nhất dựa trên vectơ đặc trưng.

6.0.1 Code: ‘inference_classifier.py‘

```python
import pickle

import cv2
import mediapipe as mp
import numpy as np

model_dict = pickle.load(open('./model.p', 'rb'))
model = model_dict['model']

cap = cv2.VideoCapture(0)

mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)

labels_dict = {1: 'bao', 2: 'keo', 3: 'bua'}

while True:
    ret, frame = cap.read()

    # Check if the frame was successfully read
    if not ret:
        print("Error: Unable to read frame from camera. Check camera connection or permissions.")
        break  # Exit the loop if there's an error

    H, W, _ = frame.shape  # Now frame.shape will work correctly

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = hands.process(frame_rgb)
    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing.draw_landmarks(
                frame,  # image to draw
                hand_landmarks,  # model output
                mp_hands.HAND_CONNECTIONS,  # hand connections
                mp_drawing_styles.get_default_hand_landmarks_style(),
                mp_drawing_styles.get_default_hand_connections_style())

            # Initialize x_ and y_ for each hand
```

```
            # Initialize x_ and y_ for each hand
            x_ = []
            y_ = []

            data_aux = []

            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y

                x_.append(x)
                y_.append(y)

            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y
                data_aux.append(x - min(x_))
                data_aux.append(y - min(y_))

            x1 = int(min(x_) * W) - 10
            y1 = int(min(y_) * H) - 10

            x2 = int(max(x_) * W) - 10
            y2 = int(max(y_) * H) - 10

            prediction = model.predict([np.asarray(data_aux)])

            predicted_character = labels_dict[int(prediction[0])]

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)
            cv2.putText(frame, predicted_character, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 0), 3,
                        cv2.LINE_AA)

    cv2.imshow('frame', frame)
    cv2.waitKey(1)


cap.release()
cv2.destroyAllWindows()
```

7 Kết quả và thảo luận

- Các ưu điểm:
- Độ chính xác: Hệ thống đạt tỷ lệ chính xác cao trong việc nhận dạng cử chỉ tay, thể hiện tính hiệu quả của việc kết hợp MediaPipe với Random Forest.
- Hiệu suất thời gian thực: Hệ thống có thể xử lý các khung hình video trong thời gian thực, cho phép tương tác liền mạch với người dùng.
- Khả năng mở rộng: Khung có thể dễ dàng thích ứng với các bộ dữ liệu cử chỉ tay khác nhau và có thể được mở rộng để nhận dạng nhiều cử chỉ hơn.
- Nhược điểm:
- Hiệu suất của hệ thống có thể bị ảnh hưởng bởi các yếu tố như điều kiện ánh sáng, cách che tay và sự thay đổi kích thước bàn tay.

8 Kết luận và kế hoạch phát triển trong tương lai

Báo cáo này trình bày một hệ thống nhận dạng cử chỉ tay theo thời gian thực, tận dụng hiệu quả MediaPipe Hands và bộ phân loại Random Forest. Hệ thống này thể hiện độ chính xác cao và hiệu suất theo thời gian thực, khiến nó phù hợp với nhiều ứng dụng khác nhau.

Kế hoạch trong tương lai sẽ tập trung vào:

1. Mở rộng bộ dữ liệu cử chỉ: Thu thập bộ dữ liệu lớn hơn và đa dạng hơn về cử chỉ tay để cải thiện độ bền và tính tổng quát của mô hình.
2. Cải tiến về độ bền: Triển khai các kỹ thuật nhằm giải quyết tác động của các yếu tố môi trường như ánh sáng và chướng ngại vật.
3. Nhận dạng cử chỉ động: Kết hợp khả năng nhận dạng cử chỉ động để cho phép hệ thống nhận dạng các chuyển động tay liên tục.

9 Source Code

GitHub:`https://github.com/Jikay-070203/ASS_CPV301.git`

10 Nguồn

1. Fan Zhang và và những người khác. MediaPipe Hands: On-device Real-time Hand Tracking. `https://arxiv.org/pdf/2006.10214v1.pdf`
2. Kumar, R. et al. Mediapipe and CNNs for Real-Time ASL Gesture Recognition. `https://arxiv.org/pdf/2305.05296.pdf`