Final Project Report: Real-Time Hand Gesture Recognition and Special Effects

Objective

The goal of this project was to create a real-time hand gesture recognition system capable of identifying specific hand signals and triggering corresponding visual effects on the screen. The application uses a pre-trained neural network for gesture classification and OpenCV for image processing to apply effects like "thumbs up", "thumbs down", balloons, and fireworks.

Methodology

Gesture Recognition

We utilized the MediaPipe library, which offers pre-trained models for hand tracking and gesture recognition. The model classifies ten distinct hand gestures including 'okay', 'peace', 'thumbs up', 'thumbs down', 'call me', 'stop', 'rock', 'live long', 'fist', and 'smile'.

Visual Effects

The visual effects were implemented in two parts:

- **Image Overlay**: Functions were developed to overlay static images with transparent backgrounds (using alpha channels) onto the video stream, aligning with the recognized gestures.
- **Video Overlay**: A separate Python script extracted frames from `.mov` video files, added alpha channels to make the backgrounds transparent, and saved them as images. These frames were then sequenced and overlaid onto the video feed to simulate continuous fireworks and balloon effects.

Implementation

Extracting Video Frames

A function `extractFrames` was implemented to process video files and extract frames with added transparency where black pixels (below a specific threshold) were made transparent.

Gesture Recognition and Effect Application

In the main script, a `create_fading_effect` function was written to initialize fading effects based on the position of the hand landmarks. An `overlay_image_alpha` function was responsible for blending images with alpha transparency onto the video feed. The `apply_effect` function resized and positioned these images relative to the detected hand landmarks. The script also handled the looping of video frames for the continuous effects of fireworks and balloons through the `create_background_effect`.

```
def create_fading_effect(effects_list, landmarks, start_alpha=1.0, end_alpha=0.0, duration=30):
    if len(landmarks) > 5:
        # Add a new thumbs up effect with its start position at landmark 4
        effects_list.append({
            "position": landmarks[4],
            "alpha": start_alpha,
            "end_alpha": end_alpha,
            "duration": duration,
            "current_frame": 0
        })


#Function to overlay video frames on top of images
def overlay_image_alpha(img, img_overlay, pos, alpha_mask):
    """Overlay img_overlay on top of img at the position specified by
    pos and blend using alpha_mask.

    Alpha mask must contain values within the range [0, 1] and be the
    same size as img_overlay.
    """

    x, y = pos

    # Image ranges
    y1, y2 = max(0, y), min(img.shape[0], y + img_overlay.shape[0])
    x1, x2 = max(0, x), min(img.shape[1], x + img_overlay.shape[1])

    # Overlay ranges
    y1o, y2o = max(0, -y), min(img_overlay.shape[0], img.shape[0] - (y))
    x1o, x2o = max(0, -x), min(img_overlay.shape[1], img.shape[1] - (x))

    # Exit if nothing to do
    if y1 >= y2 or x1 >= x2 or y1o >= y2o or x1o >= x2o:
        return

    channels = img.shape[2]

    alpha = alpha_mask[y1o:y2o, x1o:x2o]
    alpha_inv = 1.0 - alpha

    for c in range(channels):
        img[y1:y2, x1:x2, c] = (alpha * img_overlay[y1o:y2o, x1o:x2o, c] +
                                alpha_inv * img[y1:y2, x1:x2, c])
```

```python
def apply_effect(frame, landmarks, effect_name, effects_images):
    if effect_name not in effects_images:
        return  # Invalid effect name

    # Calculate the height of the thumb in pixels
    thumb_height = int(np.linalg.norm(np.array(landmarks[4]) - np.array(landmarks[1])))

    # Resize the effect image to match the height of the thumb
    effect_image = cv2.resize(effects_images[effect_name]["image"], (thumb_height, thumb_height), interpolation = cv2.INTER_AREA)
    effect_alpha = cv2.resize(effects_images[effect_name]["alpha"], (thumb_height, thumb_height), interpolation = cv2.INTER_AREA)

    # Calculate current alpha based on the frame number
    alpha_scale = np.clip(effect_data["current_frame"] / effect_data["duration"], 0, 1)
    current_alpha = (1 - alpha_scale) * effect_data["alpha"] + alpha_scale * effect_data["end_alpha"]
    alpha_mask = effect_alpha * current_alpha

    # Add an offset to the position
    if effect_name == "thumbs up":
        offset = [180, -50]  # Adjust these values as needed
    elif effect_name == "thumbs down":
        offset = [180, -200]
    position_with_offset = [effect_data["position"][0] + offset[0], effect_data["position"][1] + offset[1]]

    overlay_image_alpha(frame, effect_image, position_with_offset, alpha_mask)

    # Increment the current frame counter
    effect_data["current_frame"] += 1
```

```python
def create_bacckground_effect(frame,effect_name):
    global firework_counter
    global balloon_counter
    if effect_name == "fireworks":
        #iterate through the list of fireworks images in the fireworks_frame folder and overlay them on the frame
        overlay_image=cv2.imread("fireworks_frames/frame"+str(firework_counter)+".png",-1)
        overlay_image=cv2.resize(overlay_image,(frame.shape[1],frame.shape[0]))
        #get the alpha channel of the overlay image
        alpha_mask=overlay_image[:,:,3]/255.0
        firework = overlay_image_alpha(frame, overlay_image, (0,0), alpha_mask)
        return firework

    if effect_name == "balloons":
        #iterate through the list of balloons images in the balloons_frame folder and overlay them on the frame
        overlay_image=cv2.imread("balloons_frames/frame"+str(balloon_counter)+".png",-1)
        overlay_image=cv2.resize(overlay_image,(frame.shape[1],frame.shape[0]))
        #get the alpha channel of the overlay image
        alpha_mask=overlay_image[:,:,3]/255.0
        balloon = overlay_image_alpha(frame, overlay_image, (0,0), alpha_mask)
        return balloon
```

Real-Time Video Processing

Using a webcam, the system captures live video, processes each frame for gesture recognition, and applies the appropriate effects based on the detected gestures.

```python
# Extract images frames from Video
def extractFrames(pathIn, pathOut):
    count = 0
    vidcap = cv.VideoCapture(pathIn)
    success, image = vidcap.read()
    while success:
        # Add an alpha channel to the frame
        image = cv.cvtColor(image, cv.COLOR_BGR2BGRA)
        print('Read a new frame: ', success)

        # If pixel of frame is black make it transparent
        # Define a threshold below which the pixel is considered 'black'
        threshold = 10  # You can adjust this threshold as needed
        black_indices = np.all(image[:, :, :3] < threshold, axis=2)
        image[black_indices] = [0, 0, 0, 0]  # Set alpha to 0 (transparent)

        cv.imwrite(pathOut + "/frame%d.png" % count, image)  # save frame as PNG file
        count = count + 1

        # Read the next frame
        success, image = vidcap.read()

#Change the names according to the video file name and folder to save the extracted frames
extractFrames("balloons.mov", "balloons_frames")
```

Results

The system successfully recognizes the predefined hand gestures in real-time and applies the corresponding effects on the screen. The 'thumbs up' and 'thumbs down' gestures trigger static image overlays, while the 'peace' and 'rock' gestures initiate dynamic video overlays that simulate balloons and fireworks, respectively.

Conclusions

The project demonstrates the effective integration of hand gesture recognition with visual effects, showcasing the potential for interactive applications. While the static image overlays perform well, the video overlays present challenges in maintaining frame rate and synchronization. Future work could explore optimization techniques and additional gesture-based interactions.

Acknowledgements

I would like to thank the developers of the MediaPipe library for providing robust models for hand tracking and gesture recognition, which were instrumental in the success of this project.