REPORT

Background Subtraction in Video Sequences

Group-6

Anjani K-SE22UCSE029

Ankita-SE22UCSE030

Anshu N-SE22UCSE031

Antara-SE22UCSE032

Anya-SE22UCSE033

1. Introduction

Background subtraction is a fundamental technique in computer vision for distinguishing moving foreground objects from the static background in video sequences. It is widely employed in surveillance systems, motion tracking, and activity recognition. The aim is to generate a dynamic understanding of the scene by isolating regions of interest (foreground) while maintaining an accurate background model.

This project utilizes OpenCV's background subtraction methods to process a video, generating separate videos for the moving foreground and the static background.

---

2. Code

```python
import cv2
import numpy as np

video_path = r"C:\Users\Antaraa\Downloads\moving_bg.mp4"
cap = cv2.VideoCapture(video_path)

if not cap.isOpened():
    print("Error: Cannot open video.")
    exit()

frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
foreground_output_path = r"C:\Users\Antaraa\Downloads\foreground_output.mp4"
background_output_path = r"C:\Users\Antaraa\Downloads\background_output.mp4"

foreground_writer = cv2.VideoWriter(foreground_output_path, fourcc, fps, (frame_width, frame_height))
background_writer = cv2.VideoWriter(background_output_path, fourcc, fps, (frame_width, frame_height))

subtractor = cv2.createBackgroundSubtractorKNN(history=500, dist2Threshold=400, detectShadows=True)

background_model = None
alpha = 0.01

while True:
    ret, frame = cap.read()
    if not ret:
        break

    fg_mask = subtractor.apply(frame)
    _, fg_mask = cv2.threshold(fg_mask, 200, 255, cv2.THRESH_BINARY)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)
    fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_CLOSE, kernel)
    fg_mask = cv2.erode(fg_mask, kernel, iterations=1)
    fg_mask = cv2.dilate(fg_mask, kernel, iterations=2)

    foreground = cv2.bitwise_and(frame, frame, mask=fg_mask)
    foreground = cv2.convertScaleAbs(foreground, alpha=1.2, beta=20)

    sharpening_kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
    foreground = cv2.filter2D(foreground, -1, sharpening_kernel)

    if background_model is None:
        background_model = frame.astype(np.float32)
    else:
        cv2.accumulateWeighted(frame, background_model, alpha)

    background = cv2.convertScaleAbs(background_model)

    foreground_writer.write(foreground)
    background_writer.write(background)

    cv2.imshow("Original Frame", frame)
    cv2.imshow("Foreground", foreground)
    cv2.imshow("Background", background)

    if cv2.waitKey(30) & 0xFF == 27:
        break

cap.release()
foreground_writer.release()
background_writer.release()
cv2.destroyAllWindows()

print(f"Foreground video saved at: {foreground_output_path}")
print(f"Background video saved at: {background_output_path}")
```

3. Explanation of Concepts

3.1 Background Subtraction

Background subtraction is a technique in video processing to distinguish moving objects (foreground) from static elements (background). It works by:

1. Building a background model, which represents the stationary or slowly changing parts of the scene.
2. Subtracting the background model from each frame to isolate moving objects.

BackgroundSubtractorKNN:

In this project, OpenCV's `BackgroundSubtractorKNN` is used, which implements:

- **K-Nearest Neighbors (KNN)**: Maintains a history of pixel intensities and uses a statistical approach to classify each pixel as part of the foreground or background.
- **Dynamic Background Update**: Adapts to changes such as lighting variations or minor environmental shifts by continuously updating the background model.
- **Shadow Detection**: Optionally identifies shadows in the foreground to avoid false positives. Shadows are treated as lighter regions of the moving objects.

3.2 Preprocessing

Preprocessing is crucial to refine the raw foreground mask obtained from background subtraction.

Thresholding:

Thresholding converts the grayscale foreground mask into a binary mask:

- **Grayscale mask**: Pixels have intensity values ranging from 0 (black) to 255 (white).
- **Binary mask**: Thresholding maps all values above a certain level (e.g., 200) to 255 (white) and below to 0 (black), isolating the foreground clearly.

Morphological Operations:

Morphological transformations improve the accuracy of the foreground mask by removing noise and closing gaps:

1. **Opening**: Removes small, isolated noise by eroding and then dilating the mask.
2. **Closing**: Fills small holes in the foreground by dilating and then eroding the mask.
3. **Erosion/Dilation**: Refines object boundaries:
    - **Erosion** shrinks the mask, removing noise along the edges.
    - **Dilation** expands the mask, reconnecting fragmented parts of the foreground.

Kernel:

Morphological operations rely on a **kernel** (structuring element), which determines the shape and size of the operation (e.g., rectangular, circular). The project uses a 5x5 rectangular kernel.

3.3 Foreground Enhancement

Enhancing the extracted foreground improves its visual quality for better interpretation:

1. **Brightness and Contrast Adjustment**:
    - Brightness (β\betaβ) shifts pixel intensity values.
    - Contrast (α\alphaα) scales the intensity range.
    - Formula: output=α·input+β\text{output} = \alpha \cdot \text{input} + \betaoutput=α·input+β

    This ensures the foreground is visibly distinct.

2. **Sharpening**: Enhances edges in the image by emphasizing high-frequency components using a convolution kernel:
    - A sharpening kernel (e.g., [[0,−1,0],[−1,5,−1],[0,−1,0]][[0, -1, 0], [-1, 5, -1], [0, -1, 0]][[0,−1,0],[−1,5,−1],[0,−1,0]]) highlights boundaries, making moving objects more prominent.

3.4 Background Model Update

To maintain a robust background model, the project uses:

- **Weighted Average Update**: Incorporates the current frame into the background model with a small blending factor (α=0.01\alpha = 0.01α=0.01): background_model=(1−α)·background_model+α·current_frame\text{background_model} = (1 - \alpha) \cdot \text{background_model} + \alpha \cdot \text{current_frame}background_model=(1−α)·background_model+α·current_frame
    - This gradual update adapts the model to slow scene changes (e.g., lighting transitions) while preserving stability.

---

4. Code Workflow

4.1 Initialization

- Video is loaded using `cv2.VideoCapture()`.
- Video parameters like frame dimensions, FPS, and codec are fetched to initialize output writers for foreground and background videos.

4.2 Processing Frames

- Each frame is processed to extract the foreground mask using BackgroundSubtractorKNN.
- The mask undergoes preprocessing to ensure accurate segmentation.
- Foreground and background images are enhanced or updated accordingly.

4.3 Writing Output

- Foreground and background images are saved to video files using `cv2.VideoWriter`.

4.4 Visualization

- The original, foreground, and background frames are displayed in real-time for user feedback.

---

5. Report of Working Code

1. Video Setup

The video file is loaded using cv2.VideoCapture. Essential properties such as frame width, height, and frame rate (FPS) are retrieved to ensure the output matches the input dimensions. Two cv2.VideoWriter objects are set up to save the processed foreground and background videos.

```
cap = cv2.VideoCapture(video_path)
foreground_writer = cv2.VideoWriter(foreground_output_path, fourcc, fps, (frame_width, frame_height))
background_writer = cv2.VideoWriter(background_output_path, fourcc, fps, (frame_width, frame_height))
```

2. Background Subtraction Initialization

A KNN-based background subtractor is initialized using cv2.createBackgroundSubtractorKNN. This technique learns the background dynamically by analysing the input frames over time. A background_model variable is used to maintain a running average for a smoother, updated background. The alpha value (0.01) controls how quickly the model adapts to changes.

```
subtractor = cv2.createBackgroundSubtractorKNN(history=500, dist2Threshold=400, detectShadows=True)

background_model = None
alpha = 0.01
```

3. Main Processing Loop

This loop processes each frame from the video:

a. Foreground Mask Creation

The background subtractor generates a binary mask where moving objects are identified as the foreground. Thresholding is applied to refine the mask, ensuring only strong motion signals are preserved.

```
    fg_mask = subtractor.apply(frame)
    _, fg_mask = cv2.threshold(fg_mask, 200, 255, cv2.THRESH_BINARY)
```

b. Morphological Refinements

Morphological operations such as opening (removes noise), closing (fills small holes), and combinations of erosion and dilation further refine the mask, improving the separation of foreground from the background.

```
    fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN, kernel)
    fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_CLOSE, kernel)
```

c. Foreground Extraction and Enhancement

Using the refined mask, the foreground is extracted by applying the mask to the current frame. The extracted foreground is then enhanced for better visual clarity by adjusting brightness, contrast, and applying sharpening filters.

```
    foreground = cv2.bitwise_and(frame, frame, mask=fg_mask)
    foreground = cv2.convertScaleAbs(foreground, alpha=1.2, beta=20)
```

d. Background Update

The background model is updated dynamically using a weighted average. This ensures that stationary parts of the scene are captured accurately over time.

```
    if background_model is None:
        background_model = frame.astype(np.float32)
    else:
        cv2.accumulateWeighted(frame, background_model, alpha)
```

4. Write and Display

The processed foreground and background frames are saved using the respective cv2.VideoWriter objects. Simultaneously, the original frame, extracted foreground, and updated background are displayed in separate windows for real-time feedback.

```
foreground_writer.write(foreground)
background_writer.write(background)

cv2.imshow("Original Frame", frame)
cv2.imshow("Foreground", foreground)
cv2.imshow("Background", background)
```

5. Cleanup

Once all frames are processed, resources such as the video capture and writer objects are released, and all display windows are closed.

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

5.2 Key Highlights

- **Robust** foreground extraction using `BackgroundSubtractorKNN`.
- Smooth, adaptive background modelling.
- Enhanced foreground visualization for better clarity.

---

6. Results

The results have been saved in a folder called "results" since the videos cannot be stored here.

7. Comments on Results

- **Foreground Video**: Clear segmentation of moving objects with enhanced sharpness and contrast. Small noise may still persist in challenging scenarios.
- **Background Video**: Smoothly modeled background with gradual adaptation to changes in lighting.
- **Real-Time Visualization**: Effective for debugging and verifying the segmentation process.

8. Challenges

- **Noise in Foreground Masks**:The foreground masks generated often include small artefacts or noise, requiring additional morphological operations like opening, closing, erosion, and dilation for accurate refinement.

- **Shadow Misclassification:** Shadows detected by BackgroundSubtractorKNN can be incorrectly classified as part of the moving objects, affecting the quality of foreground segmentation.

- **Dynamic Background Handling**: Background changes, such as moving water or trees, are challenging for the background model to adapt, leading to inaccuracies in segmentation.

---

9. Usability and Deployment

Surveillance Systems

Usability: Monitors real-time activities to detect intruders or unusual movements.

- **Deployment**: Processes video feeds to detect moving objects and generates alerts.

Traffic Monitoring

- **Usability**: Tracks vehicle flow and provides traffic congestion data.
- **Deployment**: Detects vehicles by separating them from the background for real-time analysis.

Robotics and Autonomous Vehicles

- **Usability**: Detects obstacles and aids in navigation.
- **Deployment**: Uses cameras and sensors for real-time obstacle avoidance.

Video Analytics and Editing

- **Usability**: Isolates moving objects for motion tracking or special effects.
- **Deployment**: Processes video to extract foreground objects for editing or virtual reality.

Smart Retail and Customer Behaviour Analysis

- **Usability**: Tracks customer movement and analyses behaviour patterns.
- **Deployment**: Real-time analysis of customer movements for store optimization

10. Conclusion

This project successfully demonstrates background subtraction using OpenCV for video sequences. The approach effectively isolates moving objects while maintaining a dynamically updating background model. The enhanced foreground video provides sharper and clearer object boundaries, making it suitable for applications like surveillance and motion tracking. The combination of pre-processing, adaptive modelling, and output visualization ensures reliable results, although further refinement may be needed for highly dynamic or complex scenes.