BGSW (Bosch) - AutoVisionX Hackathon

Problem Statement – Home Parking Assistant(HPA)

Team Name – Brothers

Team Members- Harsh Balwani , Aneesh Rijhwani

Project Components:

1 Command-line Arguments:

The project accepts two command-line arguments: <video_file> and <start_address>.

Example: ./Hackathon.mp4 0x12340000

2 Memory Management:

Efficient memory allocation is crucial for performance. The base address provided is used for all memory buffer allocations within the project.

3 Background Subtraction:

Utilizes OpenCV's **BackgroundSubtractorMOG2** to subtract the background from each frame.

4 Video Capture Setup:

Opens the specified video file for processing.

5 Video Properties Extraction:

Extracts width, height, and frames per second (fps) information from the input video.

6 VideoWriter Configuration:

Configures VideoWriter to save the output with a specified codec, fps, and dimensions.

7 Dynamic Object Detection:

Implements a robust algorithm for detecting dynamic objects, including vehicles, pedestrians, and other relevant objects.

8 Overlay Output:

Overlays the detected dynamic objects on the original video frame.

9 Main Processing Loop:

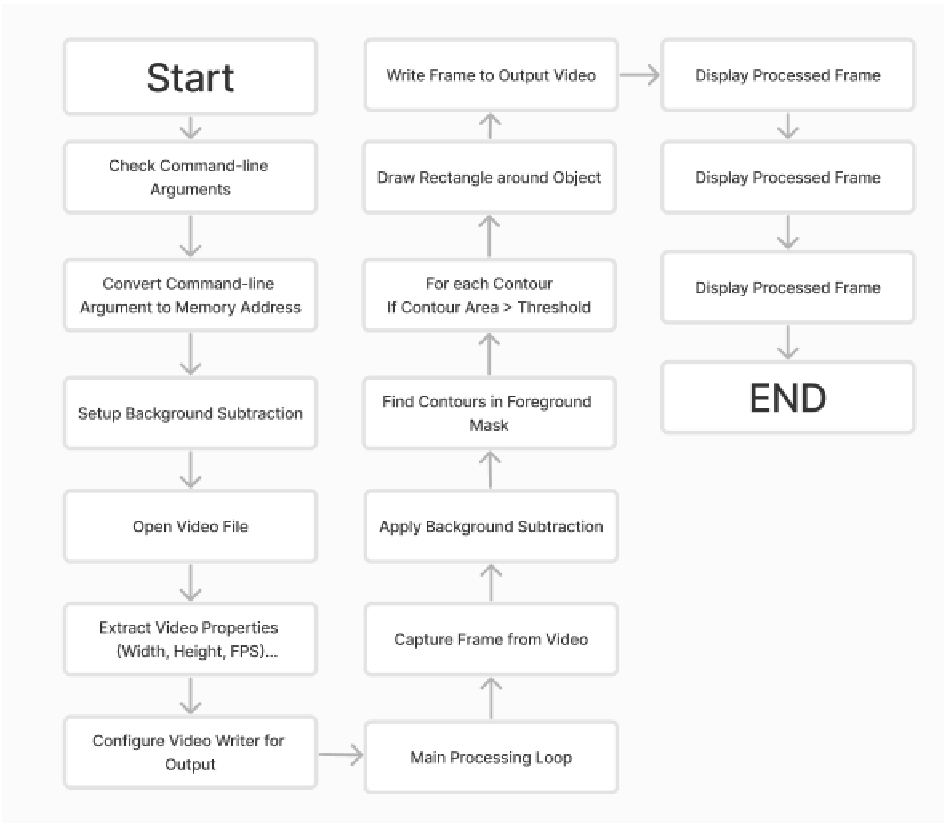
Captures each frame, applies background subtraction, identifies contours, and draws rectangles around moving objects.

10 User Interaction:

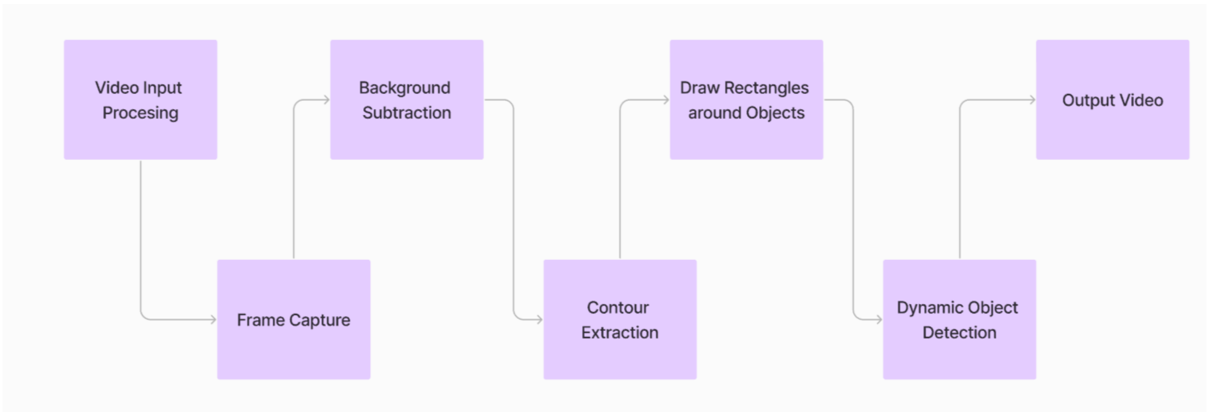
The processed video is displayed in a window named "Dynamic Object Detection".

Pressing 'Esc' exits the processing loop.

Flowchart:


Start
Check Command-line Arguments
Convert Command-line Argument to Memory Address
Setup Background Subtraction
Open Video File
Extract Video Properties (Width, Height, FPS)...
Configure Video Writer for Output
Main Processing Loop
Capture Frame from Video
Apply Background Subtraction
Find Contours in Foreground Mask
For each Contour If Contour Area > Threshold
Draw Rectangle around Object
Write Frame to Output Video
Display Processed Frame
Display Processed Frame
Display Processed Frame
END

Sequence Diagram:



Build and Run Instructions:

Step 0: Prerequisites

- Ensure you have Visual Studio installed.

- Install OpenCV 4.5.5 in `C:\` using the link.

- Unzip it in the project.

Step 1: Create an Empty C++ Project

1. Open Visual Studio.

2. Create a new project.

3. Select "Empty Project" and click Next.

4. Specify Project Name and Location.

5. Click "Create."

Step 2: Add a new C++ file to the project

1. Right-click on "Source Files" in Solution Explorer.

2. Choose "Add" -> "New Item."

3. Select "C++ File “**detect.cpp**" and specify the Name.

4. Click "Add."

Step 3: Setup Solution Platform

1. Right-click on the project in Solution Explorer.

2. Click on "**Properties**."

3. Set "Configuration" to "**x64**."

Step 4: Set Additional Include Directories

1. Set "Configuration" to "All Configuration" and "Platform" to "x64."

2. Go to "C/C++" -> "General" -> "Additional Include Directories" -> Edit.

3. Browse and select the OpenCV include folder.

4. Click "OK" and then "Apply."

Step 5: Set Additional Library Directories

1. Set "Configuration" to "All Configuration" and "Platform" to "x64."

2. Go to "Linker" -> "General" -> "Additional Library Directories" -> Edit.

3. Browse and select the OpenCV lib folder.

4. Click "OK" and then "Apply."

Step 6: Set Environment

1. Set "Configuration" to "All Configuration" and "Platform" to "x64."

2. Go to "Configuration Properties" -> "Debugging" -> "Environment" -> Edit.

3. Set the PATH to OpenCV bin folder:

```
PATH=C:\OpenCV\x64\vc16\bin;%PATH%
```

4. Click "Apply" and then "OK."

[Project HPA]

Step 7: Implement the Project

Approach:

a. **Command-Line Argument:**
   The program takes a memory address as a command-line argument. This address is used as a starting point for memory buffer allocations within the project.
b. **Background Subtraction:**
   It creates a background subtractor using the **MOG2 algorithm**. This subtractor helps identify moving objects by detecting changes in the video frames.
c. **Video Capture Setup:**
   Opens a video file named "**Hackathon.mp4**" using OpenCV's VideoCapture. If the video file is not found or cannot be opened, an error message is displayed.
d. **Video Properties:**
   Extracts the width, height, and frames per second (fps) of the input video. This information is crucial for configuring the output video.
e. **VideoWriter Configuration:**
   Sets up a VideoWriter to create an output video file named "**output_video.avi**". The video codec used is **MJPG**, matching the properties of the input video.
f. **Main Processing Loop:**
   i. Enters a loop to process each frame of the input video.
   ii. Applies background subtraction to obtain a foreground mask highlighting moving objects.
   iii. Finds contours in the foreground mask to identify separate objects.
   iv. Draws **bounding rectangles** around moving objects based on the contours, only considering those above a certain area threshold.
   v. Saves the processed frame to the output video file, displays it, and continues to the next frame.
   vi. The loop stops when the video ends or the '**Esc**' key is pressed.
g. **VideoWriter Release:**
   Releases the VideoWriter after processing all frames, finalizing the output video.

Use the provided code in your detect.cpp file.

```cpp
#include <opencv2/opencv.hpp>

using namespace cv;
```

```cpp
int main(int argc, char* argv[]) {
    if (argc != 2) {
        std::cerr << "Usage: " << argv[0] << " <start_address>" << std::endl;
        return -1;
    }

    // Convert the command-line argument to a memory address
    void* startAddress;
    sscanf(argv[1], "%p", &startAddress);

    // Create a background subtractor
    Ptr<BackgroundSubtractorMOG2> bgSubtractor =
createBackgroundSubtractorMOG2();

    // Open the video file
    VideoCapture cap("Hackathon.mp4");
    if (!cap.isOpened()) {
        std::cerr << "Error: Could not open video file." << std::endl;
        return -1;
    }

    // Get video properties
    int width = static_cast<int>(cap.get(CAP_PROP_FRAME_WIDTH));
    int height = static_cast<int>(cap.get(CAP_PROP_FRAME_HEIGHT));
    int fps = static_cast<int>(cap.get(CAP_PROP_FPS));

    // Define the codec and create a VideoWriter object
    VideoWriter videoWriter("output_video.avi", VideoWriter::fourcc('M', 'J',
'P', 'G'), fps, Size(width, height));

    while (true) {
        Mat frame;
        cap >> frame;

        // Break the loop if the video ends
        if (frame.empty())
            break;

        // Apply background subtraction
        Mat fgMask;  // Foreground mask
        bgSubtractor->apply(frame, fgMask);

        // Find contours in the foreground mask
        std::vector<std::vector<Point>> contours;
        findContours(fgMask, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

        // Draw bounding rectangles around moving objects
        for (const auto& contour : contours) {
            if (contourArea(contour) > 100) {  // Adjust the threshold as
needed
                Rect boundingBox = boundingRect(contour);
                rectangle(frame, boundingBox, Scalar(0, 255, 0), 2);
            }
        }

        // Save the frame to the video
        videoWriter.write(frame);

        // Display the processed frame
        imshow("Motion Detection", frame);

        // Press 'Esc' to exit the loop
        if (waitKey(30) == 27)
            break;
    }

    // Release the VideoWriter
    videoWriter.release();

    return 0;
}
```

Step 8: Build and Run the Project

1. Select your desired **Solution Configuration**.

2. Build the project (**Ctrl+B** or Build -> Build project_name).

3. Run the project with or without debugging (**F5 or Debug** -> Start Debugging).