**BOSCH HACKATHON**

**Title:** Dynamic object detection with Custom memory allocation using Linear Allocator

This document includes the approach for the problem statement followed by the instructions to run the project in windows ×64 architecture.

**Approach:**

The object detection process involves the following steps:

**1. Dynamic Object Detection**

**a)** **Read Video Stream**

* The program starts by reading a video file specified as a command-line argument by the user. This video stream will be processed frame by frame.
* We use OpenCV library to read the frames and perform further processing.

**b) Frame Preprocessing**

* Each frame is resized to a standard size (640x480) for uniform processing.
* The resized frames are converted to grayscale simplify further processing steps and reduce memory consumption.
* Experiments verified that the accuracy of the final output barely depends on this conversion.

**c)** **Object Detection Logic**

* We applied a simple yet effective method to detect dynamic obstacles.
* Absolute difference between consecutive frames is calculated to identify changes (potential dynamic objects) using OpenCV diff () function.
* A binary threshold is applied to the difference image to highlight significant changes.
* Contours are then identified in the binary image using OpenCV's contour detection functions.
* Detected contours are drawn on the original frame, outlining potential dynamic objects.

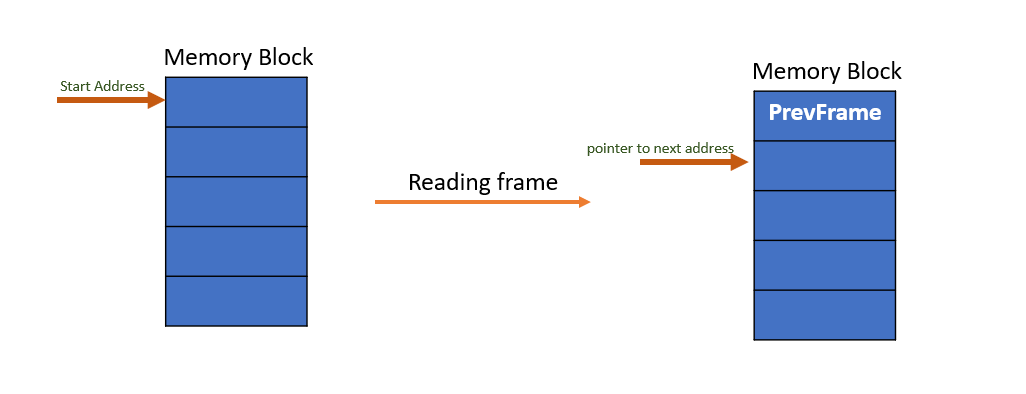
**2. Custom Memory Allocation using Linear Allocator**

**a)** **Linear Allocator**

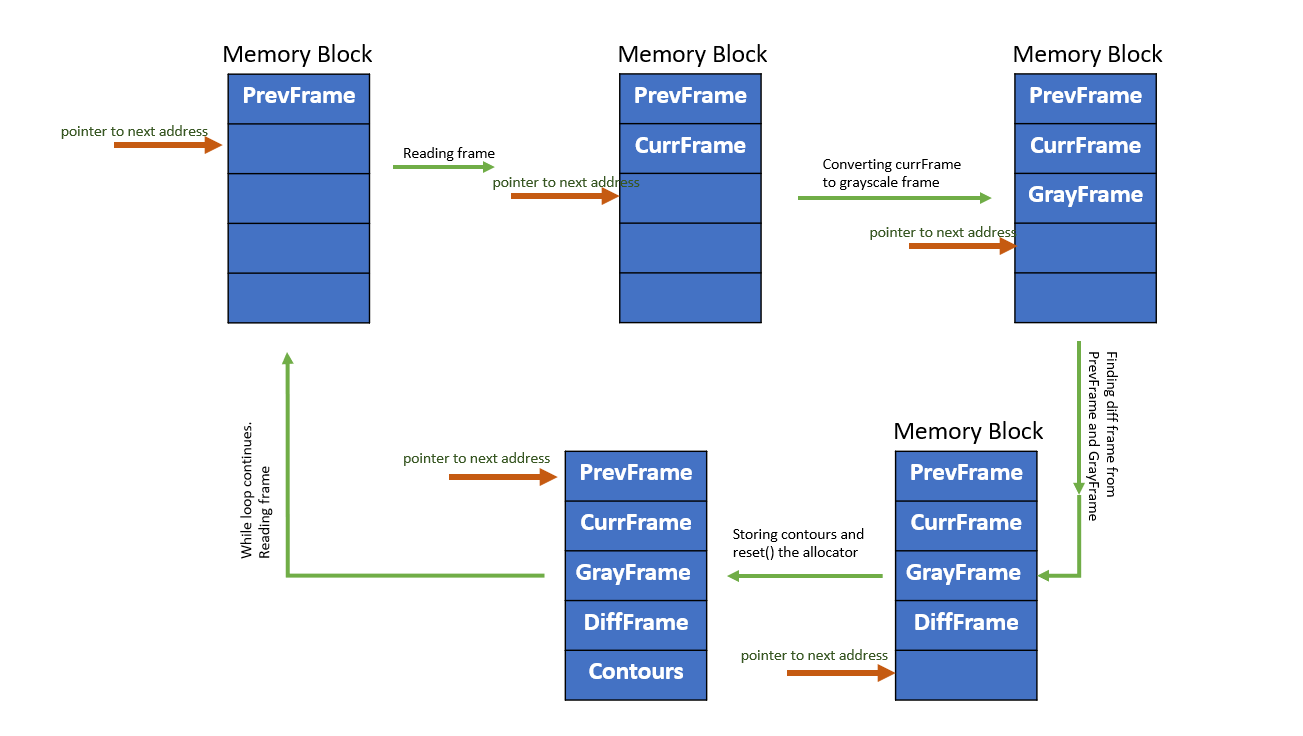
* A custom Linear Allocator is implemented to efficiently manage memory for frame storage. The allocator is initialized with a specified memory block, and memory is allocated as needed during runtime.
* To perform the object detection, the algorithm we used needs 4 frames to store and 2D vector to store the resultant contours (max size would be the size of frame). So, we allocate the total size to be 5\*640\*480(to store 5 frames).
* We allocate memory only at the start of program. It speeds up the program by decreasing the number of times the OS shifts from kernel to user mode.

**b)** **Memory Allocation for Frames**

* Just by using the 5 frames of allocated memory, we can process and find dynamic objects in the video file of any size.
* We do this by adjusting the pointer to the start address once the frame is processed and objects are overlayed. LinearAllocator.Reset() function makes this happen by setting the memory allocated and offset to 0.
* So, for every frame the memory allocation starts from the address specified by the user.
* This method becomes very efficient in case of memory constrained situations.



**Memory and Process flow of the project**



**3. Instructions to run the project**

* The OpenCV library, LinearAllocator library is included in the zip file specified along with the source code of the project.
* DetectObstacles.cpp is the main file for program execution. It is where the memory allocation and object detection happen.
* Build the project before starting to run it.

1. **Use CMake to configure and build the project.**

cmake -G "MinGW Makefiles" -S . -B build

1. **Compile the project**

cmake --build build

1. **Run the Project**

.\DetectObstacles.exe “path/to/video.mp4” memory\_address