**Obstacle detection**

Obstacle detection and calculation is performed by this ECU where it detects the vehicle and calculates its distance from the truck. Based on these inputs from Vehicle dynamic sensor ECU the vehicle performs autonomous functions like braking, acceleration, and so on. This part of the implementation is calculated in the OpenCL platform by our system where all these necessary performance calculations are done. OpenCL provides the facility for the user to compute kernel using a limited subset of C programming on a GPU. In our system, we have implemented an obstacle detection module where the sensors (RADAR) data in centimeters is converted into meter units considering the real-time scenario. Sensor data received from the environment is stored in an array on which kernel performs the computation for conversion. The sensor data and the conversion factor are allocated to the GPU memory and read operation is performed using clCreateBuffer(). The output is also allocated to the GPU memory where the write operation is performed using clCreateBuffer().Later the output from the GPU memory is copied to CPU memory using clEnqueueReadBuffer() function. A distance condition is checked in the main function in order to determine if obstacle is detected or not.

Steps followed in OpenCL are:

1. **Do the setup**
2. Get the OpenCl platform using clGetPlatformIDs()
3. Create context to share between devices and even for execution using clCreateContextFromType().
4. Create command queues for submitting work to different devices using clCreateCommandQueue()

Find all the available devices, put them together into a context, that create queues For the device in the

context, so that one can submit work to different devices.

1. **Compilation**
2. Create the program using clCreateProgramWithSource()
3. Build the program using clBuildProgram()
4. Create Kernel using clCreateKernel()
5. Create the memory objects
6. **Enqueue writes to copy data to GPU**
7. **Set Kernel arguments using clSetKernelArg()**
8. **Enqueue Kernel execution using clEnqueueNDRangeKernel()**

In execution take kernel object and put it into the queue and if the queue is pointing to a particular device it will execute on the device