Vertex Array Object (VAO) based Visualization - A Better Method for ROV Stance Visualization for Real-Time Application

Annapurna Agrawal
Scientist 'D', R&DE (E)
DRDO, Pune, India

BaniHazra
Scientist 'D', R&DE (E)DRDO,
Pune, India

Alok Mukherjee
Scientist 'G', R&DE (E)
DRDO, Pune, India

ABSTRACT
Remotely Operated Vehicle (ROV) is an unmanned mobile platform which is operated remotely through the Control Station. ROVs generally have Manipulator Arm with multiple degrees of freedom for handling objects and application specific payloads such as sensors for navigation, surveillance, target tracking and destruction. The ROV Operator needs to be confident about the ROV's stability and safe remote and Non-Line of Sight (Non-LOS) Operations.

This paper presents the ROV stance visualization system which is developed to show the vehicle pose and manipulator pose to the operator remotely. Stereo Lithography (STL) model file of the ROV is used for this visualization. In this paper two approaches conventional method and Vertex Array Object (VAO) based method for STL Model visualization are presented, compared and concluded that, the Vertex Array Object (VAO) based method gives high rendering speed and is therefore suitable for real-time application. This work is generic for any ROV/ static platform mounted with Manipulator Arm with multiple Degrees of freedom (DOF).

General Terms
Remotely Operated Vehicle (ROV), Stance Visualization, Degrees of Freedom (DOF)

Keywords
Manipulator Arm, STL model, 3D Visualization, Vertex Array Object (VAO).

1. INTRODUCTION
Unmanned platforms for defense applications are key asset in the hands of any Army or Paramilitary forces. Unmanned Ground Vehicles (UGV) provide a new dimension to the fighting forces. In the near future, battle fields shall be populated with systems which are unmanned and have sufficient intelligence on-board to carry out the intended mission. UGV system consists of Remotely Operated Vehicle to perform some specific task and a Master Control Station (MCS) to control the ROV remotely.

Remotely Operated Vehicle (ROV) may have Manipulator arm with multiple Degrees of freedom (DOF). Manipulator arm has multiple rigid links, which are interconnected by joints. Joints provide motion to the links to reach the desire pose of the end effector for handling the object. Direct Kinematics [1] is used to find the pose and orientation of Manipulator Arm given the arm parameter e.g. Link lengths and joint angles. Figure 1 shows a ROV platform which has a Manipulator Arm with multiple Links and joints.

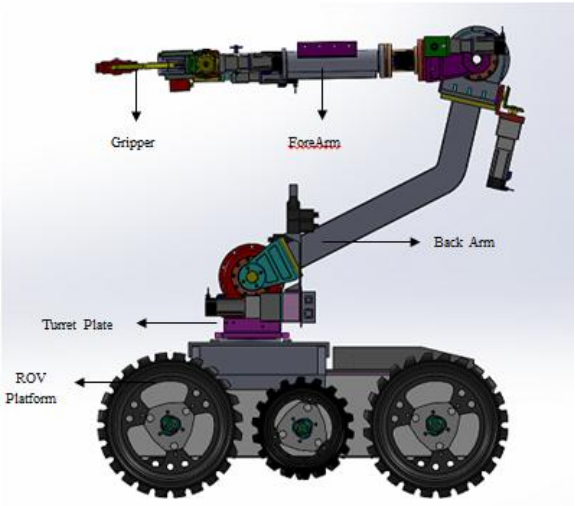

Fig 1. ROV with Manipulator Arm

ROV is operated remotely through a Control Station. Control station consists of Human Machine Interface (HMI) through which various control commands can be sent to the ROV and the Video/Sensor/Manipulator feedback can be visualized by the operator for safe navigation and object handling. Stance visualization software is a part of Human Machine Interface. It gets the Manipulator/Vehicle Pose feedback from the onboard controller of ROV.

To visualize the Stance of ROV and the Manipulator, Binary Stereo Lithography (STL) [2, 3] model files are used. STL file is a triangular representation of a 3D object. The surface of an object is broken into a logical series of triangles. Each triangle is uniquely defined by three points representing its vertices in clockwise direction and a normal of triangle pointing outward to the surface. The binary STL file format uses the IEEE integer and floating point numerical representation [4].

SharpGL [5] library which allows to use Open Graphics Library (OpenGL)[6, 7] in .net Framework, is used for the STL Visualization

2. STANCE VISUALIZATION ALGORITHM APPROACH I
In this approach conventional method is used for drawing STL Model of Manipulator Arm. As STL file contains a large number of triangles to form a 3D object. The sample code of drawing elementary graphics primitive is shown below.

glBegin(GL_TRIANGLE)

glNormal3f (n1, n2, n3);

glVertex3f(x1, y1,z1);

glVertex3f(x2, y2,z2);

glVertex3f(x3, y3,z3);

glEnd();

Main drawback of this method is that driver cannot tell the GPU to start rendering before glEnd, because it does not know when will the data submission be finished, and it needs to transfer that data too. This came out as a very slow method for drawing as graphics card was directly linked to the program flow. Every time single triangle parameters were passing to GPU for rendering. This approach is very simple to implement, but not suitable to render realistic model in real time application.

3. STANCE VISUALIZATION ALGORITHM VAO BASED APPROCH

During the development of stance visualization system, main challenge was to achieve high rendering speed so that it can be integrated with Human Machine Interface (HMI) of Remotely Operated Vehicle (ROV) for real-time Stance Visualization. This goal is achieved using Vertex Array Object (VAO) based method to draw the Manipulator STL Model. In this method array of vertices and array of normals are created and those arrays (all vertices and normal) are passed to the Video RAM at Graphics Processing Unit (GPU) and the render program has no longer to process this data. The render thread and GPU run asynchronously and parallel, which yields better performance and high rendering rate. Vertex arrays also reduce the number of function calls and redundant usage of shared vertices. Therefore it increases the performance of rendering.

Following steps are used to draw the STL model in this approach.

1. glEnableClientState(GL_VERTEX_ARRAY);// Enable the Arrays (vertex, normal, texture etc)

2. glVertexPointer(3, GL_FLOAT, sizeof(vertex), vertices); // the array pointer to pass the vertices/ normal/ textures array to openGL object.

3. glDrawArrays(GL_TRIANGLES, 0, num_indices);//Draw the object from Array data.

The flow diagram for complete Stance visualization software using approach 1 or Approach 2 is shown in figure 2.

Comparison of both the approaches have been done and the comparison is presented in table 1.

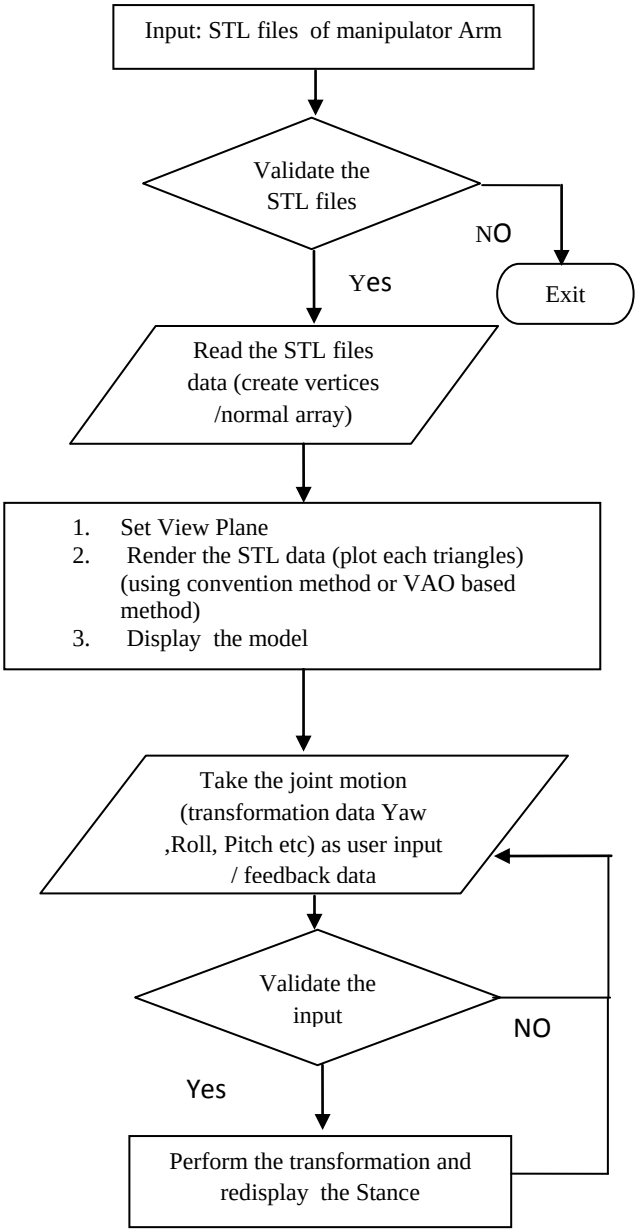

Fig 2 :Stance Visualization Algorithm - Approach I / Approach II:

Table 1: Comparison Table Approach I and VAO method

Criteria	Conventional Method	VAO based method
Processing method	Vertices based	Array based
Redundant Vertices Processing	No, process all the vertices	Yes
Implementation	Easy to implement, Well documented	Advance method, With less documentation
Performance	Slow rendering speed, not suitable for RT application	High rendering speed for Real-timeapplication

4. EXPERIMENTS AND RESULTS

This work has been initiated for our ongoing project CBRNe ROV which is a six wheeled robot with a six axis Manipulator Arm which provides joint angle feedback of each joints to visualize the stance of Manipulator Arm.

Both the approach Conventional method and Vertex Array Object (VAO) based method are implemented in C# using SharpGL classes for 3D visualization. Stance visualization software is developed to visualize the pose of CBRNe Manipulator Arm. STL model files of CBRNe Manipulator are taken as input. This Manipulator is multi-axis, with five rotary joints and one prismatic joint thus six degrees of freedom. 7 STL files are used to display the complete Manipulator Arm. Table 1 shows details of STL files.

Table 2: STL files Description for Manipulator Arm

S. No.	File name	Size	Remark
1.	Cbrne1.STL	2.72 MB	Turret of manipulator
2.	Cbrne2.STL	7.47 MB	Back Arm Bracket
3.	Cbrne3.STL	2.16 MB	Back Arm of Manipulator
4.	Cbrne4.STL	1.12 MB	Fore Arm Bracket
5.	Cbrne5.STL	304 KB	Fore Arm of Manipulator
6.	Cbrne6.STL	2.45 MB	Tilt mechanism for Gripper
7.	Cbrne7.STL	3.7 MB	Gripper
Total Data for manipulator Visualization		**~20 MB**	

About 20 MB of data has to be rendered to visualize the complete model. The Manipulator Arm and its different poses visualize by Stance visualization Software is shown in figure 4, 5, 6.

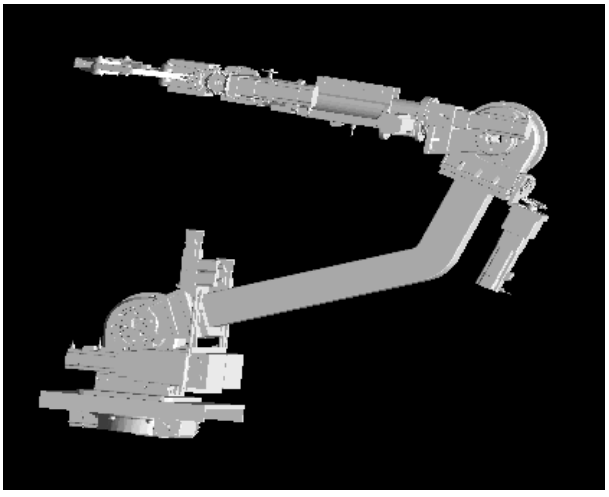

Fig 3. Manipulator Arm visualization (Home position)

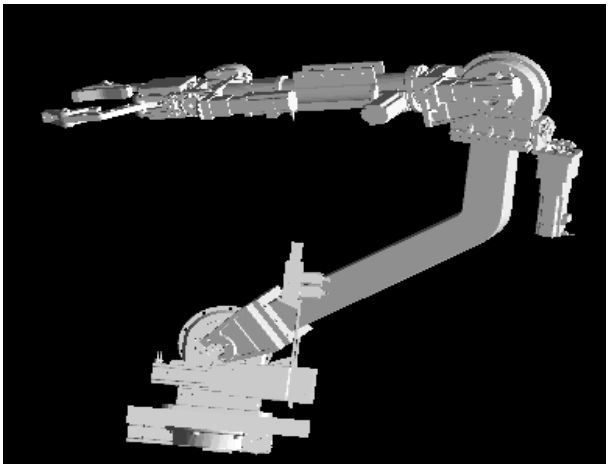

Fig 4.BackArm 30 Degree Up.

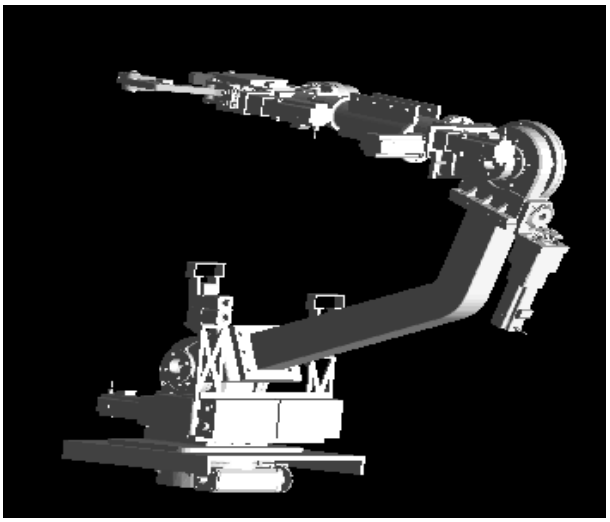

Fig 5Turret 45 Degree left.

A repetition test was performed to measure the performance of both the approaches based on their draw time and rendering rate.

Average draw time and Rendering Rate for Visualization is calculated for comparison. Results are provided in the table 2.

Table 3: Repetition test result

S. No.	Repetition test	Conventional Method		VAO based Method	
		Draw Time (ms)	Rendering Rate (FPS)	Draw Time (ms)	Rendering Rate (FPS)
1.	Exp 1	204.1	4.9	28.98	34.5
2.	Exp 2	208.3	4.8	28.5	35.1
3.	Exp 3	204.1	4.9	27.93	35.8
4.	Exp4	212.7	4.7	28.9	34.6
5.	Exp 5	222.2	4.5	27.6	36.2

Average draw time and rendering rate for both the method are shown in table 4

Table 4: Comparison of approach I and approach II

Method	Average Draw time (ms)	Average Rendering Rate (FPS)
Conventional Method	210.28	4.75
VAO based method	**28.38**	**35.2**

It is concluded that VAO based method for Stance visualization is **approximately 7 times faster** than the conventional method of visualization. This method can be used for real-time stance visualization.

In addition to that this software has zoom in/out and view change capability for better visualization.

This software has been integrated with the Human Machine Interface (HMI) of CBRNe MCS. Presently it is under testing and trial.

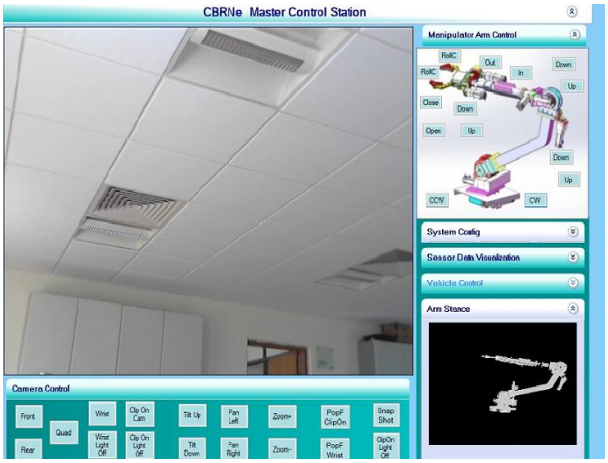

Figure 6: Stance visualization Software with HMI

4. CONCLUSION AND FUTURE WORK

The rendering rate achieved by VAO based method for Stance visualization is 35 FPS, which is suitable for Real-time Application. This work has been carried out for CBRNe ROV which provides the feedback of joint angles at 50 ms rate. This is a very important module of Human Machine Interface (HMI) for the operator to operate the ROV and payloads safely and efficiently. It may prevent the accidents by generating early warning.

Presently this module is tested in Software In Loop (SIL) mode using ROV feedback simulator, which generates the feedback massages for various pose of Manipulator Arm. This module has been integrated with the Human Machine Interface (HMI) of Master Control Station (MCS) and willbe tested in real-time with feedback. The module has been developed generically and can be easily adapted in any HMI code for further projects to be under taken.

5. ACKNOWLEDGMENT

The authors would like to thank Dr. S Guruprasad, Director, R&DE(E) and Mr. A K Patel, Group Director ASG-Robotics, R&DE(E) for all the encouragement and support in carrying out this work.

Also the help of Robotics Group is deeply acknowledged. The authors are thankful to R&DE(E) where this work is carried out. The authors are grateful to the anonymous referees for their valuable comments and suggestions by which paper becomes much clearer.

6. REFRENCES

[1] Direct Kinematics, Fundamental of Robotics, "Chapter 2",Robert J. Schilling.

[2] STL, URL. http://www.wikipidiea.com/STL/

[3] Annapurna Agrawal, "STL model Visualization and motion simulation for Manipulator Arm using OpenGL", ICCSIT – 2012, Pune, India, April 2012

[4] IEEE745-1985, URL, Http://en.wikipedia.org/wiki/IEEE 745-1985..

[5] SharpGL, URL, Http://sharpgl.codeplex.com

[6] R.S. Wright, L. Benjamin and H. Nicholas (2007)OpenGL(R) SuperBible: Comprehensive Tutorial and Reference, 4th Edition, Addison-Wesley.

[7] OpenGL Reference Manual.pdf

[8] Tom Mcreynolds and David Blythe (2010), "Advanced Graphics Programming using OpenGL"