**SUMMARY**

Our project focuses on volume or 3D visualization of human data which is large-scale and has to be interactive, hence we are motivated to use GPU techniques. GPU-based large-scale volume visualization techniques based on the notions of actual output-resolution visibility and the current working set of volume bricks the current subset of data that is minimally required to produce an output image of the desired display resolution.

Our methodology is consist four major steps :-

* + Preprocessing
  + Generating a View-Dependent Working Set
* View-Dependent Sorting of the Bricks
* Memory Management of the Working Set
* Brick Pool
* Macro-cell Pool
  + Proxy-Geometry Rasterization
  + Ray-casting

**Preprocessing :** The process of pre-processing is as follow –

* A large volume dataset is first pre-processed using blocking techniques (division of large dataset in blocks of data) and storing it in Octree.
* An **octree of bricks** is constructed where the actual resolution data is stored in leaf nodes.
* At each level or node a brick of same dimension Bres3 is made.
* Inner bricks are built using down sampling of the lower level nodes like averaging filter ,So the whole data is represented as multi-resolution hierarchy which is maintained on CPU (out of core).

**Generating a View-Dependent Working Set :** The main steps are as follow –

* For viewing different frames a cut is decided that include different resolutions and updated to the GPU brick pool for rendering.
* For deciding the cut BFS order octree traversal is used starting from the root node and is continued till the required node is found. Pointers to the traversed cut-nodes are stored in a STL list..
* A traversed node is always replaced by its children in front to back order during depth-first search.
* The cut represents the brick data and macrocell data of the nodes. This data is then transferred to GPU asynchronously.

**Proxy-Geometry Rasterization :** Process is as follow –

* The whole working set of bricks is rasterized in the first pass and capture all bricks that a ray penetrates into a per-pixel list.
* In second pass, we rasterize all non-empty macro-cell.
* This results in skipping of empty spaces of macro-cell and gives detailed process for the 2nd pass i.e. traversing the macro-cells

**Ray-casting :** Work flow is as follow –

* A ray is generated for each desired image pixel.
* In the ray-casting pass, depth interval is defined by the two end points of a ray-segment that performs the GPU ray-casting for the corresponding brick.

**Future Scope and Conclusion :** We have implemented is a fast, GPU based out-of-core volume ray- casting, which moves the branching intensive octree traversal out of the GPU ray-casting loop and after the traversing execute it on the GPU. It also offers more sophisticated empty space skipping, and which makes possible the interactive use of advanced features for the large out-of-core data sets. The method can also be used to improve the performance of the other visualization systems and as a future work we can also use it for time-varying data visualization