3D Object Rasterization and Rendering with CUDA

Minyu Cai, Jinyan Zhu

Introduction

We implemented 3D object rasterization to render a 360-degree camera view of the object. We demonstrated how we can achieve great speedup by parallelizing the task among multiple GPUs with CUDA and OpenMP. In our approach, we applied per-vertex approach in projection task, and take advantage of per-pixel method in rasterization task, which yield huge performance gain on GPU. Moreover, we used OpenMP to enable task distribution among multiple GPUs, further promoting speedup on computation-intensive scenes.

System Design and Approach

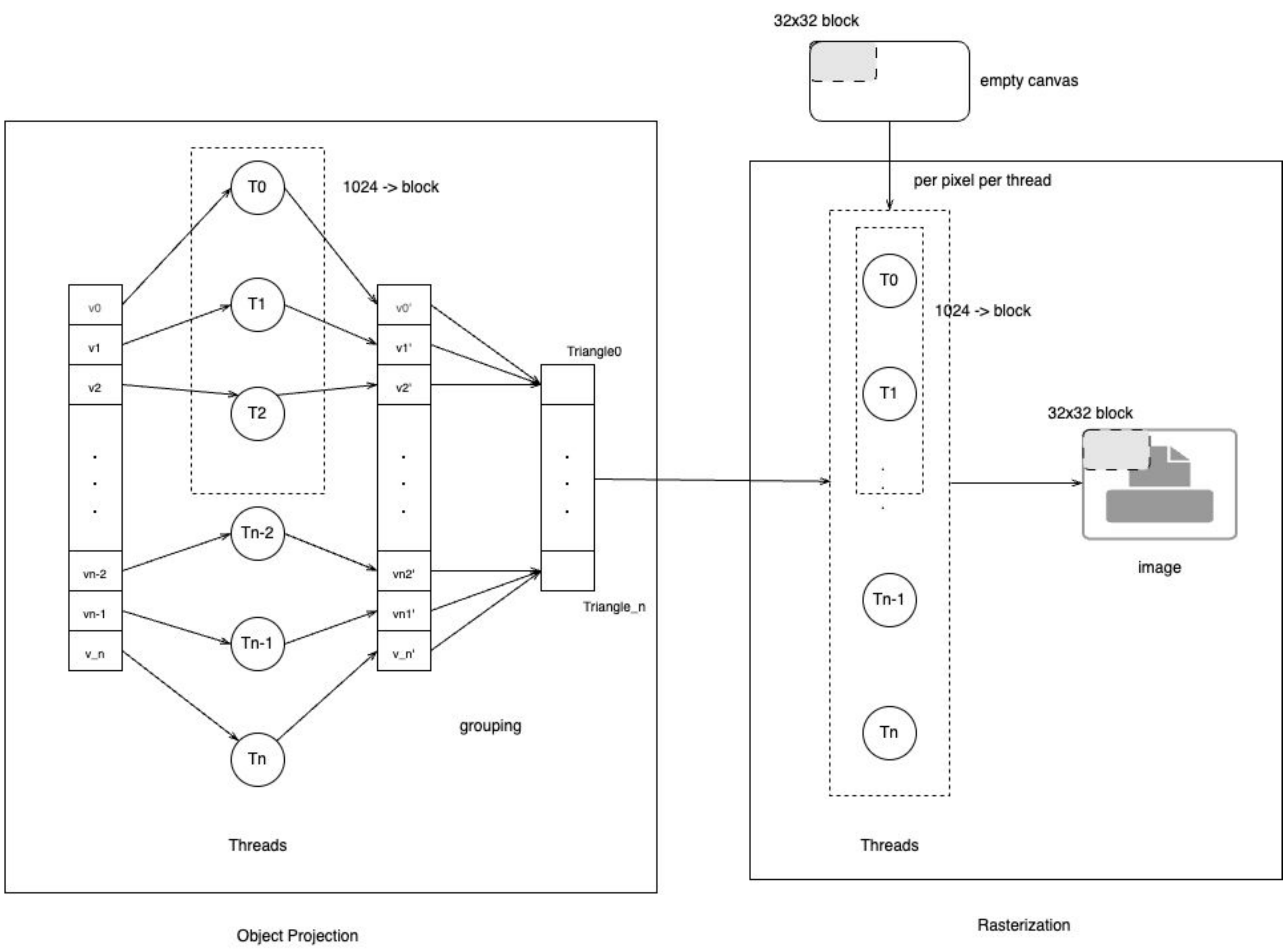


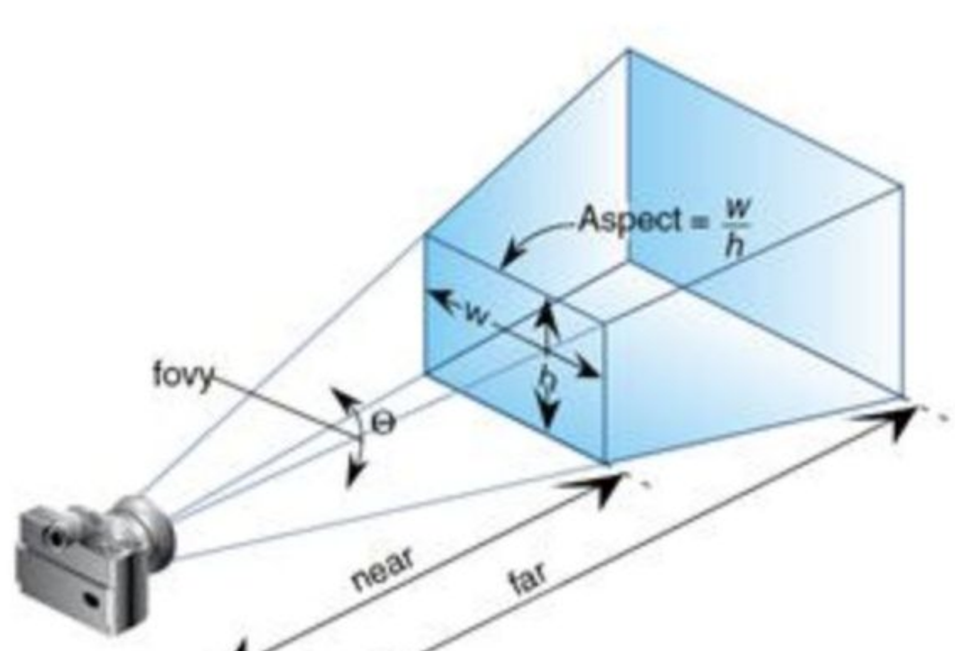
Fig.1: System Framework

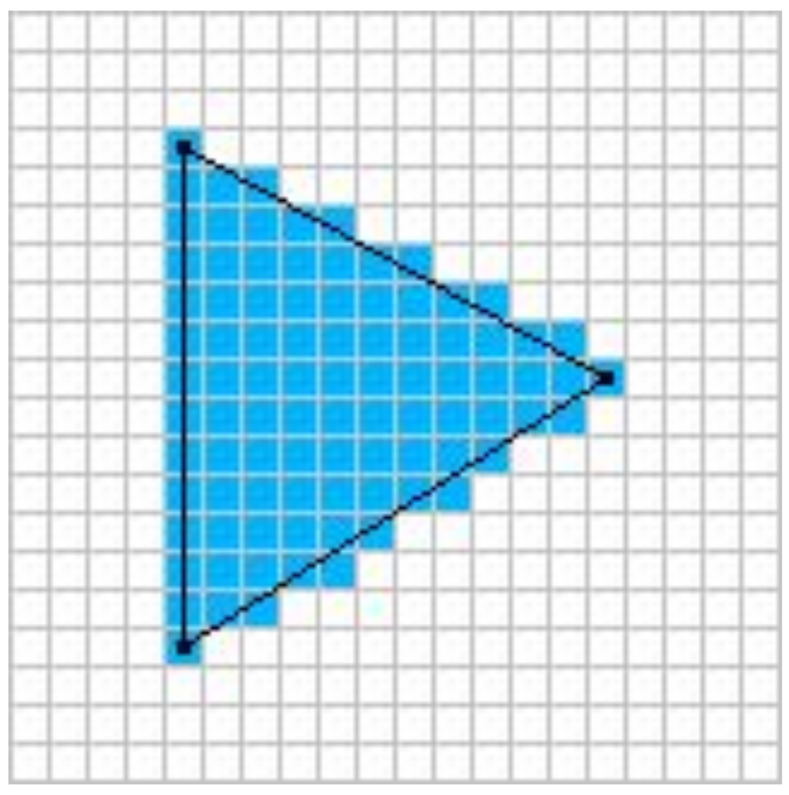
Projection:

- Input: Vector3 (x, y, z)
- Thread:
  - Per Vector3 per thread
  - 1024 threads -> block
  - Convert Vector3 to camera perspective
  - Group every 3 Vector3 as a Triangle

Rasterization:

- Input: Triangles from projection, pixels on empty canvas
- Thread:
  - Per pixel per thread
  - 1024 threads -> block (i.e. 32x32 pixels on canvas)
  - Block mapping





$$Mortho = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective matrix

$$R = I + (\sin\theta)K + (1-\cos\theta)K^2$$

Camera rotation matrix

3D Scenes Demo

Fig.2: basic 3D Objects (cube, tetrahedron)

Fig.3: 3D Course Title

Fig.4: “Cube” Visual Illusion

Performance Evaluation

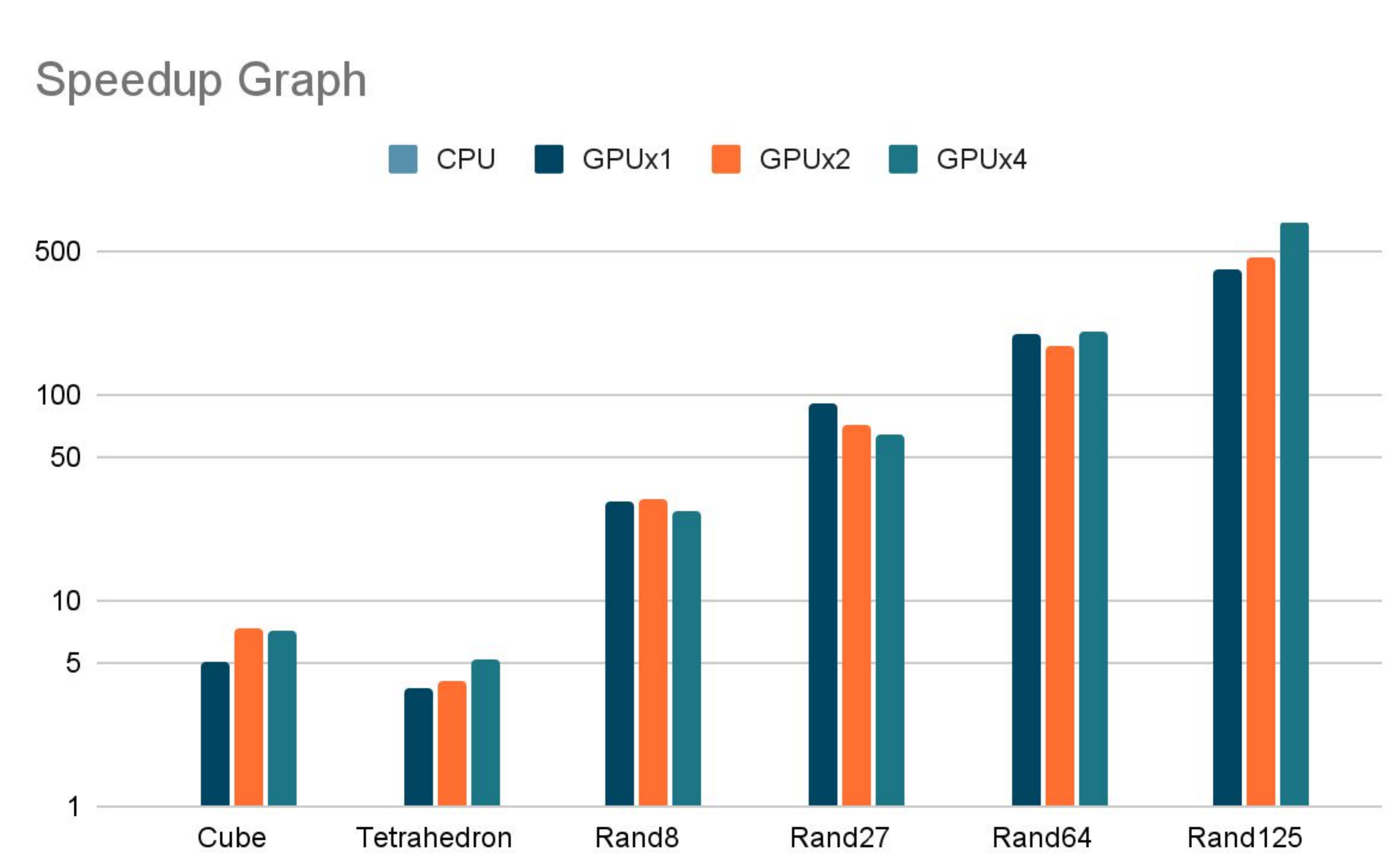


Fig.5: Speedup Graph (tested on Tesla V100-SXM3-32GB)