Abstract
A hypergraph is a generalization of an ordinary graph in which edges can connect any number of vertices. The process of determining correspondence between the nodes in two hypergraphs, called hypergraph matching, helps solve problems such as image matching and object recognition. However, it is computationally demanding, making it impractical in real-world applications. Our main contribution is to accelerate problems such as image matching and object recognition.

Hypergraphs
Hyperedges may connect any number of vertices rather than just two. This allows hypergraphs to model complex relationships such as relationships of users in a social network website, or the reactions between compounds in a complex chemical reaction.

Algorithm
The algorithm based on a probabilistic approach of hypergraph matching [2]:
1. Take weight matrices of two hypergraphs and calculate the similarity scores of every pair of edges. Each score indicates how similar a pair of edges are.
2. Calculate similarity scores of every pair of nodes and the resulting matrix, called soft matching, contains the probabilities that any given pair of nodes match.
3. Convert the soft matching result matrix into a permutation matrix that contains only 1s and 0s, known as hard matching, giving a deterministic answer to the matching problem.

Implementation on GPU
- For hypergraphs with N nodes, launch a grid of $N^2$ threads on the GPU.
- Each thread fetches data from edge weight matrices, calculates edge similarity score, and stores the result in temporary matrices.
- Each thread then calculate vertex matching similarities using data in temporary matrices.

Testing Method
- Generate random points on a 2D-plane.
- Represent their distances as edge weights to construct a hypergraph.
- Obtain the second hypergraph by rotating and distorting the first one.

Future Directions
- Implement the algorithm on multi-GPU platforms to further enhance performance.
- Sample and test using real images instead of randomly generated datasets.
- Utilize tensors to extend the algorithm to support higher degree hypergraphs.

Preliminary Result
Our result shows that the parallel implementation on GPUs achieves a 8x to 10x speed up for moderate test sizes compared to the serial implementation on CPUs.

Reference

Acknowledgement
This research was supported by:
- CUDA Teaching Center Program, Nvidia Research
- Student Research Program, Trinity College