# Optimized Geometric Image Hashing

## Introduction

Geometric hashing is a technique for recognizing objects by mapping their features into a standardized coordinate system [1][2].
It uses pairs of model points to define a basis, transforming other features relative to that basis. These transformed features are stored in a hash table for quick lookup.
When analyzing a scene, the same transformations are applied to its features, allowing fast matching against the model.
This method is robust to several changes such as position, orientation, and scale, making it effective for object detection.

## Project Goal

The goal of the project is to implement optimized versions of the two key phases of geometric hashing: the pre-processing phase (hash table construction) and the matching phase. We will focus on translation, scaling, and rotation transformations, though additional transformations are welcome. For 2D images, these transformations exhibit $O(n^3)$ complexity, where n is the number of features used. Teams should focus on 2D images rather than 3D scenes.

## Example Code

In papers [1][2] some pseudocode is provided to implement the two key operations mentioned above. In this GitHub repository [3] there is a Python implementation that supports the operations mentioned above (plus some other things that can be ignored). Moreover in this repository [4] we implement another simple Python implementation that can be used as inspiration.

## Dataset and Notes

You may use either synthetic datasets or real images. For synthetic datasets, generate stylized point sets that represent objects and ensure that your models have a sufficient number of features (do not make N too small or it might be hard to see the speedup from optimizations) to effectively demonstrate performance improvements and a realistic example. If you choose to use real images, please use off-the-shelf feature extraction tools (such as OpenCV's SIFT, SURF, or ORB) and datasets so that you can concentrate on optimizing the geometric hashing process rather than on feature extraction. Additionally, include multiple model images to show that your algorithm is robust across different object representations and perspectives.
Evaluation will focus on the performance and correctness of your optimized hashing algorithms,

not on the intricacies of feature extraction or anything strictly unrelated to the performance modelling. Feel free also to tune things such as the voting thresholds and other parameters to get the best results.

# References

[1] H. J. Wolfson and I. Rigoutsos, "Geometric hashing: an overview," in IEEE Computational Science and Engineering, vol. 4, no. 4, pp. 10-21, Oct.-Dec. 1997, doi: 10.1109/99.64160
https://graphics.stanford.edu/courses/cs164-10-spring/Handouts/paper_geohash.pdf
[2] Y. Lamdan and H. J. Wolfson, "Geometric Hashing: A General And Efficient Model-based Recognition Scheme," [1988 Proceedings] Second International Conference on Computer Vision, Tampa, FL, USA, 1988, pp. 238-249, doi: 10.1109/CCV.1988.589995.
https://www.cs.utexas.edu/~grauman/courses/spring2007/395T/395T/papers/Lamdan88.pdf
[3] https://github.com/ahakouz17/Geometric-Hashing-For-Protein-Interface-Recognition
[4] https://github.com/tommasobo/GeoHashExample