Samuel Zlota
CS 4610/5335 Final Report
April 18, 2023

Geometric Hashing

Abstract

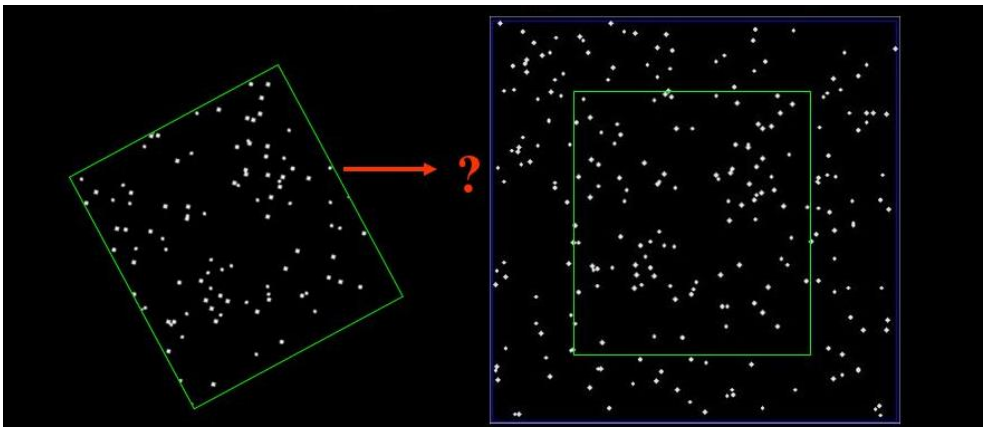
Point Cloud Registration (PCR) is a common problem faced in many vision and robotics problems. With accurate correspondences, PCR reduces to the trivial problem of solving a geometric transformation. In many applications, correspondences cannot be assumed; however, algorithms like ICP can perform quite well under sufficiently small geometric transformations without correspondences, assuming the image and model are sufficiently proximal with a high degree of overlap. In this report, we will explore the Geometric Hashing Algorithm (GHA), an algorithm that solves the PCR problem without correspondences and is not constrained by the size of the transformation. In fact, GHA can also operate under more complex transformations than rigid body motion. In this report, we will explore the affine-invariant GHA variant, comparing it with the SOTA for the PCR problem, ICP. In the end, we show under what circumstances GHA will outperform ICP and which cases ICP may be preferred.

Introduction and Background

Motivation

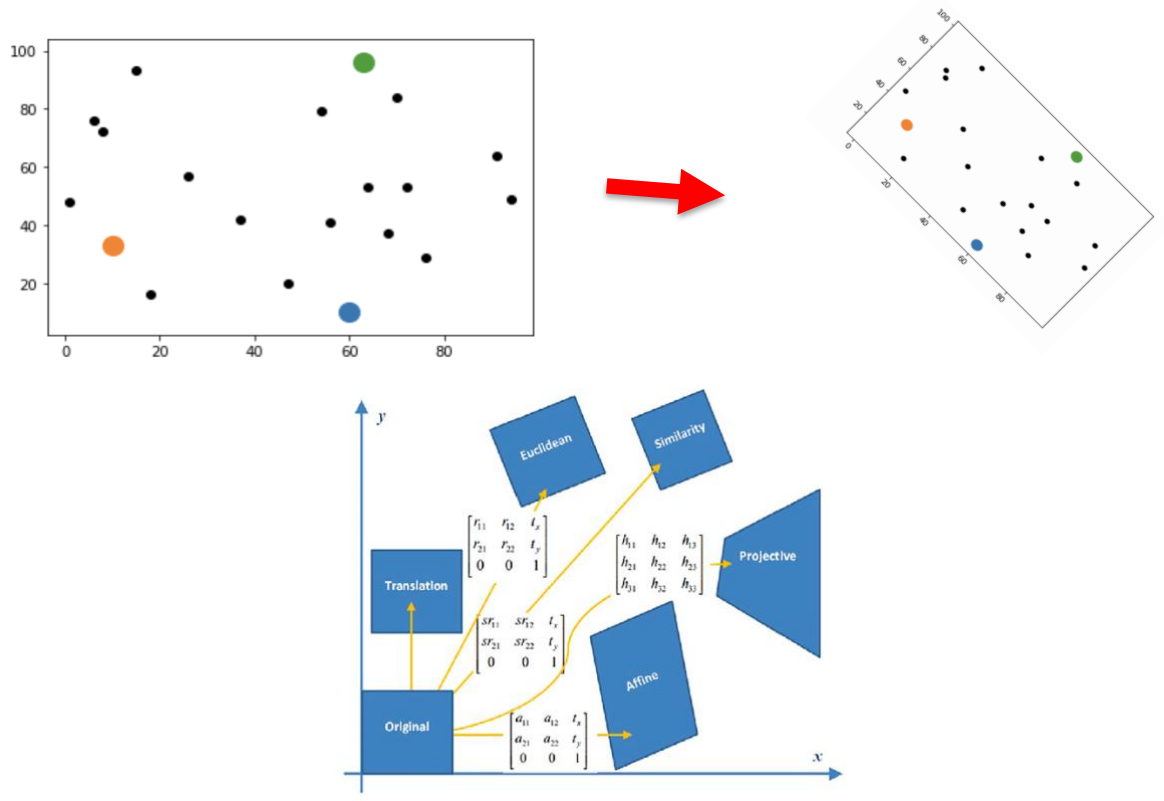
To motivate the usefulness of this algorithm, we will use a somewhat uncommon but easy to understand application of the algorithm, star tracking. This problem is sometimes referred to as the "Lost in Space" problem. Imagine you are on a spacecraft with no communication capabilities, and you want to localize yourself based solely on the stars you can see. The logic is sound, at any given point in space, the stars you can see and how they look are unique. Therefore, how can we exploit the uniqueness of our field of view to localize our position. The assumption here is that we have a map of where all the stars are, which is a strong assumption to make but required, nonetheless.

Another way to think of this problem is that you are a human being on earth that takes a picture of the night sky and wonders what stars they are looking at. GHA is one way to solve such problems. Many other applications exist for GHA for structure recognition and object recognition.



Problem Formulation

To recap, our problem formulation is as follows: we have some map/model of the world, we have some sensor reading or image of the world that differs from our world model by some known class of geometric transformation that we want to estimate without any correspondences. In this report, we will use the 2D affine-invariant GHA. The procedure, however, does not change when using 3D data or more complex transformations. The affine geometric transformation is quite robust to most 2D geometric transformations, having 6-DOF which is just short of the 8-DOF for a full projective transformation, the most complex 2D geometric transformation. Therefore, the affine-model is a great choice for many applications whether they be simple rigid-body motion or complex 2D transformations. If 3D data is sufficiently planer (i.e. points on the surface of the earth), then the affine model approximates the transformation well.



High Level Algorithm Overview

At a high level, the GHA works by uniquely encoding (hashing) substructures of the world model such that they are invariant under affine transformations. These encodings are used as hashes in a hash table and thus can be queried during online inference with some image data. We will use the word image to refer to the general class of sensor readings and observations which may or may not be from a camera.

The algorithm falls into two primary stages, offline and online, which we will discuss in more detail below.

Offline Stage

The offline stage works by enumerating all affine bases in our world model, a 2D point cloud. An affine basis is triplet of points that form a triangle (are not collinear). For each point, *D*, that is not part of that basis, *(A,B,C)* , we can encode *D* relative to *(A,B,C)* using the following formula:

$$D = x(B-A) + y(C-A) + A$$

The pair *(x,y)* is the "geometric hash" of *D* with respect to *(A,B,C)*. The key property here is that if *A, B, C,* and *D* undergo the same affine transformation, *(x,y)* will not change, hence they are affine-invariant. We can then use add these values to a hash table like so:

$$(x, y) \rightarrow (A, B, C)$$

The pair *(x,y)* is the key and the tuple *(A,B,C)* is the value. The expectation here is that in our image we will find *A', B', C' ,D'* and be able to compute *(x,y).* We thus have a correspondence between our image and the world and solve for the geometric transformation. Because we are using the affine model, a three-point correspondence is sufficient to solve the 6-DOF linear system of equation to recover the transformation.

The offline stage has combinatorial space and time complexity (enumerating triplets), so it can be prohibitive for large point cloud models, but its high cost is amortized over the arbitrary number of constant-time queries done during online inference. Sparsifying large point cloud models may also work fine to bring down space and time complexity with similar registration capabilities – we are only aiming to find a three-point correspondence.

Online Stage

Under sufficiently controlled conditions, which we will discuss in more detail later, the online stage has constant time complexity. The online stage works similarly to the offline stage by enumerating all affine bases. For every other point that is not part of the basis, the table built in the offline stage is queried. Again, under optimal conditions, we will have a perfect correspondence. However, there are two situations we will encounter under most if not all conditions, noise, and hash collisions. The first mitigation here is that we verify our correspondence somehow, so that we do not proceed unless we have high confidence in our final registration. The second mitigation is to not query on exact matches but also nearest neighbors.

Verifying the Registration

Once we reach the point in the algorithm where we think we have a correspondence between the world and our image, we must verify the match is good. One simple way to do this is to reproject the

world into the image using the estimated transformation via the correspondence and calculate the error. This verification procedure is quite sound to detect hashing collisions, but it is still quite susceptible towards local minima in the error function. This is one of the biggest drawbacks to the algorithm. Moreover, certain types of data and noise models are more susceptible to these drawbacks, which we will discuss later. For example, if your sensor reading has high false positive or negative detections, then using a simple sum-of-squared-distances error will lead to local minima.

Methods, Results, and Discussion

Experimental Setup and Backdrop

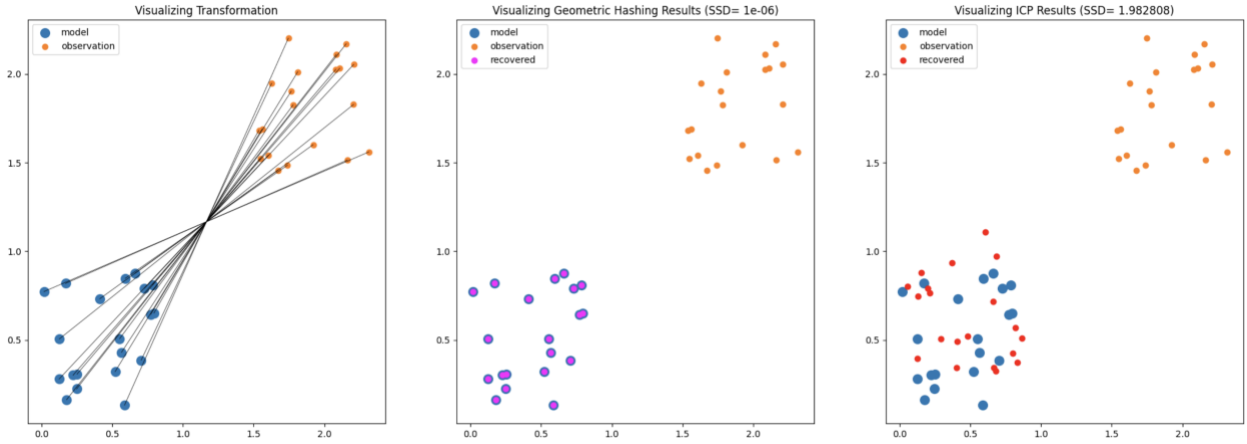
The following experiments serve to explore the various advantages and disadvantages of GHA. We will explore various transformations and noise models to see how well GHA performs against ICP. Unless otherwise specified, we will be using small, random, synthetic data. These controls seem are best to demonstrate various properties of the algorithm, but it is important to note that real-world data is not always randomly distributed. We will try to show that, in fact, GHA is very sensitive to the structure and distribution of the data. Moreover, it is expected that performance will degrade the less random the structure of the data is.

Additionally, due to the complexity of the offline stage, we elected to use only 20 points for the experiments. This size yields a runtime of a couple of seconds for the offline stage. The online stage runs in sub-second time.

Experiment 1: *Small Rigid Transformation*

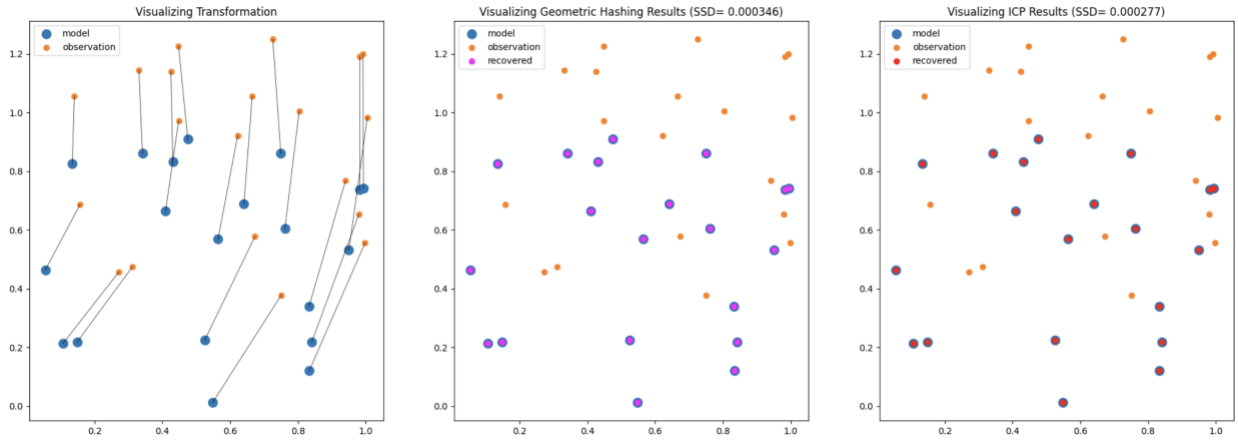
In the left most column you can see the model (world) and observation (image) points. You can see the correspondence between the model and the observation via the lines drawn. In the second and third columns, you can see the results of applying GHA and ICP respectively. Under a small, rigid transformation, we expect both algorithms to be performant. These results indicate that under the ideal conditions for ICP, GHA performs equally as well.

Experiment 2: *Large Rigid Transformation*



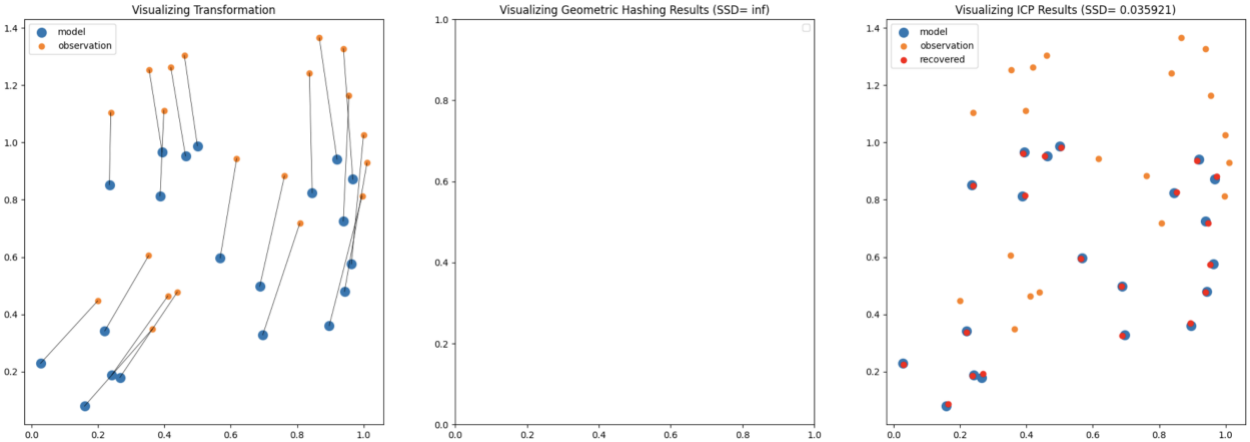
In this experiment, we used a 180-degree rotation and a larger translation. Although we have not changed the class/complexity of the transformation, we have severely degraded from the ideal conditions for ICP. However, because GHA is fully affine invariant, we expect its performance to not be degraded. The results demonstrate this hypothesis, where clearly ICP failed to register the correspondence.

Experiment 3: *Small Perturbation/Deformation Noise with Small Rigid Transformation*



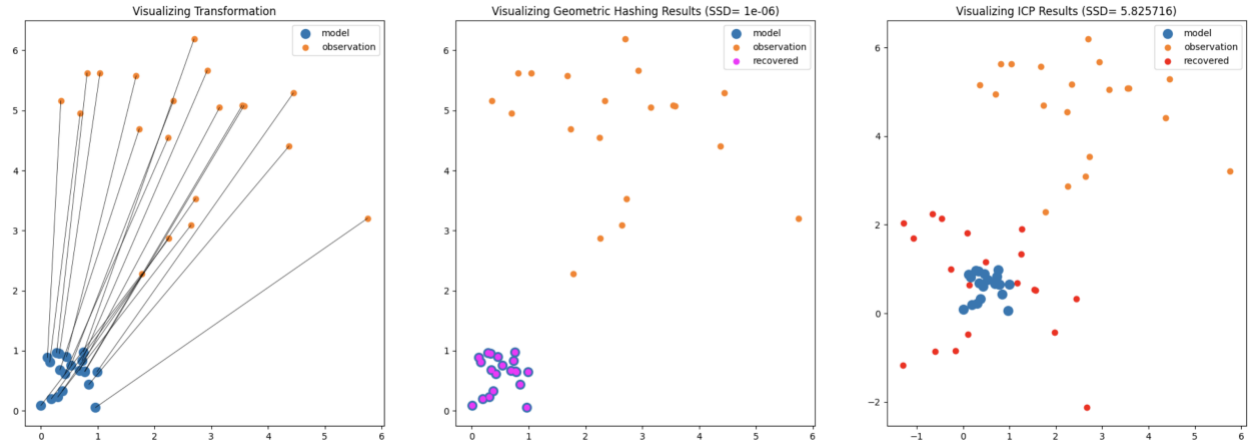
We expect the performance of GHA to severely degrade with deformations in the point cloud structure as we expect the algorithm to be highly sensitive to structural differences. In this example, all the points lie in the unit square, and we added a random perturbation noise of +/- 1e-4. We can see this amount of noise was not enough to throw off either algorithm.

Experiment 4: *Larger Perturbation/Deformation Noise with Small Rigid Transformation*



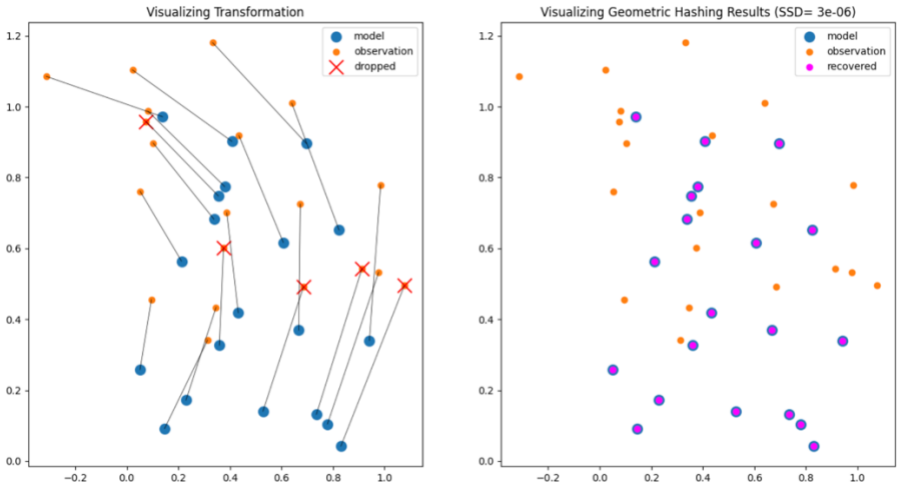
In this experiment, we used a perturbation value of +/- 1e-2. Ultimately, the GHA procedure failed. In such sub-optimal conditions, GHA is not able to approximate or find a good-enough solution. However, ICP is robust to these deformations for a sufficiently small transformation. Although, this is expected and understandable, it may be undesirable for certain applications.

Experiment 5: *More Complex Transformations (shearing, scaling)*



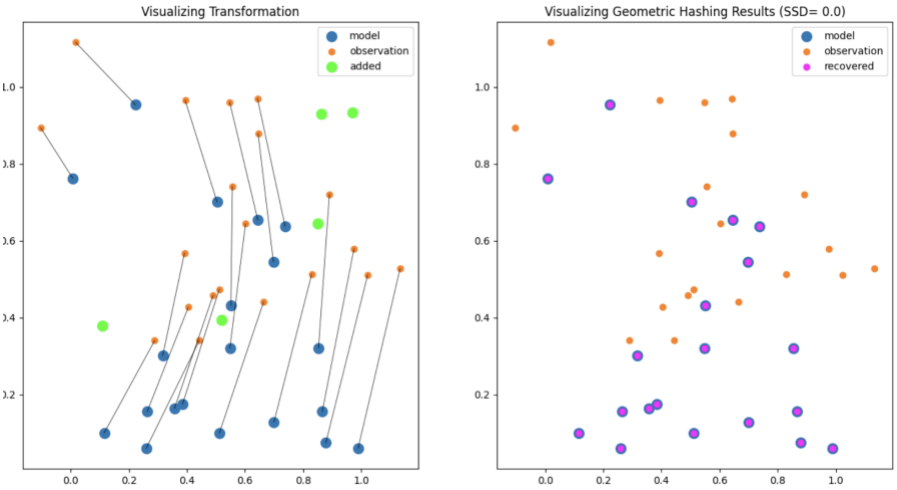
In this experiment, we demonstrate another way to degrade the performance of ICP, whilst preserving the performance of GHA. In this example, we introduce a small scale and shear on top of rotation and translation. Because shearing and scaling fall within the affine class of transformations, we expect the GHA to perform well. It is evident, however, that ICP could not perform under these circumstances due to its optimization for rigid transformations.

Experiment 6: *Random drop-out (false negative sensor readings)*



In this experiment, we performed a random drop out of five points from the observation point cloud. These points are labelled with X's. Because GHA exploits local features, we expect it to be robust to false negative observations. In real world data it is highly likely to observe occlusion. For example, in the night sky, weather and other variables can make certain stars appear invisible. The implementation for ICP used cannot handle different sized point clouds, so we omitted any results.

Experiment 6: *Random new points (false positive sensor readings)*



In this experiment, we simulate another noise model in which random new points appear in our observation. This, again, is very realistic because often sensors make false positive detections. For example, in an image of the night sky, there could be many things like planes or satellites that appear to be stars but are not. We demonstrate here that GHA is robust to false positive sensor readings.

Conclusions and Future Work

In this report we have demonstrated that GHA is a valid option to solve PCR problems. We have demonstrated its strengths in handling various noise models and complex transformations like inaccurate observations and full affine transformations. Moreover, we have elucidated where ICP performs sub-optimally and GHA does not. We, however, have also shown that GHA is very sensitive to structural deformation in point clouds, especially in comparison to ICP. Contexts and applications where deformation is likely may yield suboptimal results from GHA. Not only does GHA not recognize deformations, but it also fails catastrophically because some local structure may be deformed such that it resembles another local structure, or one never seen all together; ICP is better able to approximate transformations under deformations.

To fully understand the limitations of GHA, further research must be done to understand the sensitives of the algorithm. Exploring various properties of shapes like regularity and self-similarity will yield a deeper understanding of the algorithm. Additionally, there is a lot of room for improvement on the space complexity of the algorithm. Although the online stage has constant time complexity, it may be prohibitively expensive to store the entire database of model features.

Overall, GHA gives a unique perspective on the correspondence and a registration problem, which are critical foundational components of most vision and robotics applications.

References

Wolfson, Haim & Rigoutsos, Isidore. (1997). Geometric Hashing: An Overview. Computational Science & Engineering, IEEE. 4. 10 - 21. 10.1109/99.641604.

Link to Code: https://colab.research.google.com/drive/1i8tz-0-ul446jaUZsoXspDioW2fE76c9?usp=sharing