SURIM 2017 Project: Persistent Homology of Molecular Rotation Projections

Patrick O'Neil
Advisor: Gunnar Carlsson

September 1, 2017

1 Abstract

When dealing with data sets, computing the homology and Betti numbers is an extremely powerful tool for data analysis. This paper will examine the work I've done this summer on how images respond under this type of computation and what methods and metrics should be used when doing this type of work.

2 Introduction

Processing data is one of the main fundamental questions in modern computer science. Although data storage and management are also important, especially as we continue into the era of Big Data and the Internet of Things, here we will be principally concerned with how to process data in order to gain an understanding of it. In particular, we will be comparing use of the 'weighted graph metric' to the more traditional Euclidean metric when forming our simplicial complexes. The point cloud upon which I'll build my simplicial complexes will be made from rotations of a randomized molecule around some number of axes, with each partial rotation's xy-planar projection as a single data point. There is much work in this area describing different complex constructions; in order to minimize necessary computational power and to stay consistent I'll only be using the lazy witness construction (for more details on the merits of this construction, I point you to *Topological Estimation Using Witness Complexes* by G. Carlsson and V. de Silva).

I will be using some important groundwork done by V. de Silva and G. Carlsson on simplex building and pattern recognition, though I will be doing much more applied work. To this end, I will be using the publicly available Matlab program javaplex, developed by the Computational Topology workgroup at Stanford University.

I will begin with a review of homology and definitions of the various tools and variables used in computational homology, including persistence, filtration, and Betti num-

bers. Then I will move into a description of my work, and a discussion of my results. I will finish with an overview, and potential directions for future work in this area.

3 Background

Homology: When differentiating between surfaces, it is useful to compare fundamental groups. However, we cannot tell the difference between S^3 and S^4 using the fundamental groups. In order to overcome this difficulty, we define on each finite simplicial complex K the simplicial homology groups $H_q(K) \mid q \in \mathbb{Z}$. Although some spaces cannot be described with a simplicial complex or can be described in many different ways, the simplicial homology groups turn out to depend only on the polyhedron $|K|$.

To compute homology, then, we can use any specific simplicial complex K (usually a triangulation, if one exists) of our space X. The homology group $H_q(K)$ can be thought of, in a sense, as measuring the $(q+1)$ -dimensional holes in X. Thus, we are interested in closed paths which do *not* bound any portion of our surface.

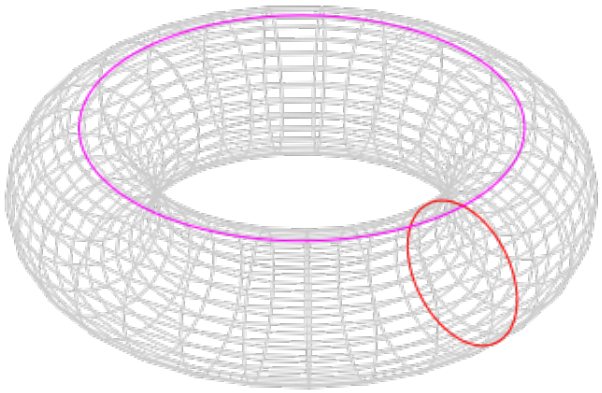


Fig. 1 - Examples of non-bounding curves on a Torus. Source: Wikipedia

We choose some specific simplicial complex K whose geometric realization is our space X. Now, we can construct a free abelian group $c_q(K)$ generated by the q-simplices of K (we can either decide a fixed orientation for K or simply think of operations on our simplices as being in the binary field), this group is called the *q-th chain group*. We shall often want to define homomorphisms on these chain groups, a particularly important homomorphism is the boundary homomorphism. We define the boundary homomorphism ∂ on a q-simplex to be the sum of all the $(q-1)$-simplices, defined over $\mathbb{F}_2$. Then we have

$$\partial_q : C_q(K) \to C_{q-1}(K) \mid C_{-1} = 0.$$

Lemma 3.1. *The composition* $C_{q+1} \xrightarrow{\partial_{q+1}} C_q \xrightarrow{\partial_q} C_{q-1}$ *is the zero homomorphism.*

Thinking back to our discussion of bounding and non-bounding closed curves, we can see that the image of ∂_{q+1} is the group of bounding q-cycles of K (denoted as $B_q(K)$), and

the kernel of ∂_q is the group of q-cycles of K (denoted as $Z_q(K)$). The above lemma shows that, in fact, $B_q(K)$ is a subgroup of $Z_q(K)$. The q-th homology group is now defined as

$$H_q(K) = Z_q(K)/B_q(K).$$

Betti Numbers: When simply differentiating between spaces, we often aren't interested in what the homology groups are, but rather how many there are at different dimensions. The q-th Betti number β_q (after Enrico Betti) is defined as the rank of the q-th homology group. Informally, they can be thought of as the number of 'q-dimensional holes' in a space, with β_0 being the number of connected components. For example, the Betti numbers for a n-sphere are $\beta_0 = 1$ and $\beta_n = 1$. The Betti numbers do not uniquely determine a space, but they are nevertheless a powerful tool for recognizing spaces. Related to this, it is important to know a result of the Whitehead theorem, this theorem gives us a way to show that two spaces' homology groups are isomorphic.
Some Betti numbers for commonly known spaces (if an entry is empty or not shown then it is 0):

	β_0	β_1	β_2	β_3	β_n
S^n	1				1
P^n, n odd	1				1
P^n, n even	1				0
T^2	1	2	1	0	
Klein Bottle	1	1	0	0	
$SO(3)$	1	1	1	1	

Persistence: A point cloud doesn't come with any sort of 'natural topology', so we must instead use the points to construct one. To do this, we construct a series of *filtered simplicial complexes* from the data set. A filtration on a simplicial complex X is a collection of subcomplexes $\{X(f) \mid f \in \mathbb{R}\}$ such that $X(f) \subset X(f')$ when $f \leq f'$. The construction method I used is called a *lazy witness complex*. This is a less-computationally-intensive version of a witness complex construction (thus the 'lazy'). Given a point cloud Z, we pick a 'landmark subset' L that is evenly distributed on Z, and we pick a parameter $v \in \mathbb{N}$ (usually 0, 1, or 2). A *lazy witness complex* is defined as follows: From our point cloud Z we choose a subset of points L (called the 'landmark subset') that is evenly distributed on Z. Then, L is our vertex set, and for vetrices a, b the edge $[ab]$ is in our complex if there exists a 'witness point' $z \in Z$ such that

$$\max\{d(a,z), d(b,z)\} \leq f + m(z)$$

where f is our filtration value and $m(z)$ is defined as follows: If we chose our parameter $v = 0$ then $m(z) = 0$ for all z. Otherwise, $m(z)$ is the distance from z to the vth closest landmark point. A higher-dimensional simplex is in our complex if all of its edges are.
Note that this construction gives us a filtered simplicial complex; the 'lazy' refers to the fact that the 1-skeleton determines all the higher dimensional simplices.

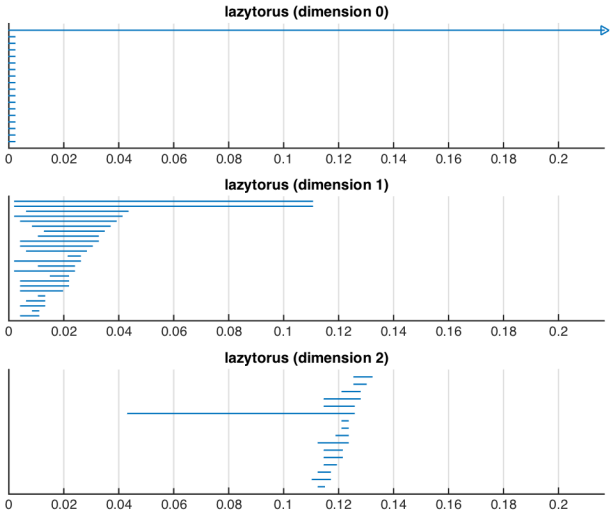


Fig. 2 - Example of persistent Betti numbers on a point cloud sampled from a torus. You can see that $\beta_0 = 1$, $\beta_1 = 2$, and $\beta_2 = 1$ since there's one bar in dimension zero going from 0 to infinity, and significantly larger bars than the others in dimensions 1 and 2. Ideally, we would have all relevant bars going from zero to infinity; in reality what we look for qualitatively is for bars that stand out from the noise.

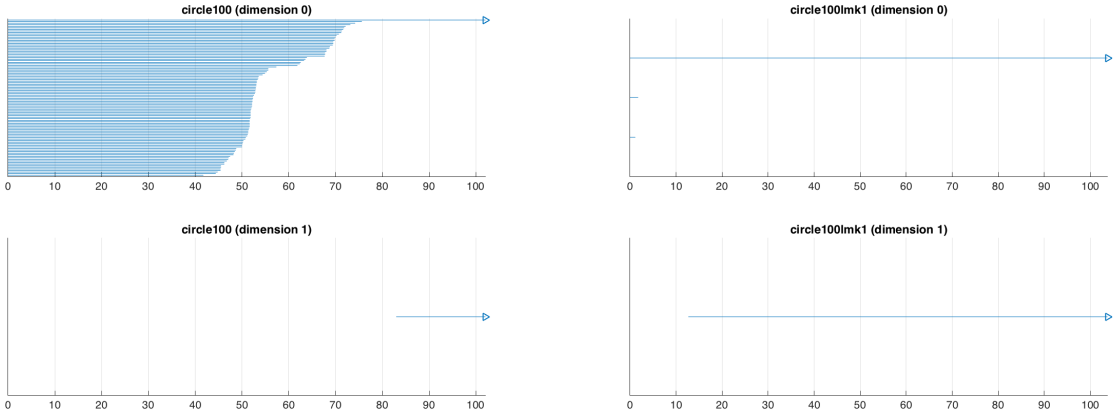


Fig. 3 - Example of $v = 0$ (left) and $v = 1$ (right) for a point cloud obtained by rotating a set of points around a single axis. In both cases, we can see $\beta_0 = \beta_1 = 1$, because the bars go off to infinity. However, you can see the right picture is much cleaner.

Lazy Witness Example: We'll now briefly walk through a construction of a lazy witness complex on a simple data set, a 'house' of points. So our points $(0, 3)$, $(-1, 2)$, $(1, 2)$, $(-1, 0)$, and $(1, 0)$; this is our point cloud Z from which we'll select our landmark points.

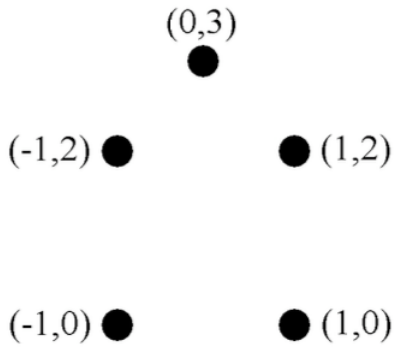


Fig. 4 - The 'house' point cloud.

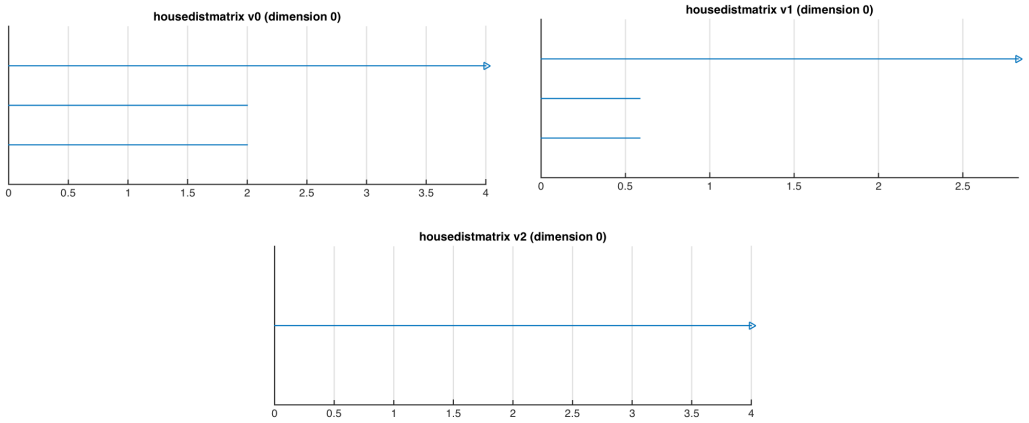
We select the points $(0, 3)$, $(-1, 0)$, and $(1, 0)$ to be our landmark subset L. Now we can calculate by hand the filtration values for each simplex in our simplicial complex. We begin with our lazy witness construction parameter $v = 0$. Then, the equation to determine when a simplex is included in the simplicial complex is just

$$max(d(a, z), d(b, z)) \leq f.$$

We take $a = (0, 3)$, $b = (1, 0)$, and $c = (-1, 0)$. Then the filtration value $f(ab) = 2$, and it is witnessed by $(1, 2)$; similarly $f(ac) = 2$ and is witnessed by $(-1, 2)$. The other edge bc also appears at $f = 2$ and is witnessed by either b or c. The lone 2-simplex abc appears automatically when all 3 edges are present, so $f(abc) = 2$.

We now repeat the process for $v = 1$ and $v = 2$. For $v = 1$, we get $f(ab) = 2 - \sqrt{2}$ witnessed by $(1, 2)$, $f(ac) = 2 - \sqrt{2}$ witnessed by $(-1, 2)$, and $f(abc) = f(bc) = \sqrt{2}$ with bc witnessed by $(\pm 1, 2)$.

For $v = 2$, $f(ab) = 0$ witnessed by $(1, 2)$ or by a itself, $f(ac) = 0$ witnessed by $(-1, 2)$ or by a, $f(bc) = 0$ witnessed by $(\pm 1, 0)$, and thus $f(abc) = 0$.



Barcodes for the house with $v = 0, 1, 2$.

4 Results

In order to determine what methods work well with images, we need to be working with a known space. I chose to work with all possible rigid rotations of a randomized 'molecule', a set of 20 or so small balls in 3d space. The previously-mentioned Whitehead theorem gives us the helpful result that the homology group of SO(3) is isomorphic to the homology group of these rotations. This means that we know what Betti numbers we're looking for, so we have a way to verify what results are 'good'.

Methodology: In order to analyze data from an image set, we need to first create or find the images. For this project, I decided to capture images of a 'molecule' that I randomly created in Matlab, and then to use the built-in Matlab 'scatter' and 'getframe' functions to get an image. The molecule was created by choosing a random value between -1 and 1, and then iteratively choosing more values (with some bounds to ensure the 'atoms' weren't too far from or too close to each other). Since we're concerned with the homology of molecular rotations, I chose a number of axes (points uniformly distributed on the unit sphere) around which to rotate and a radian value to determine how many images I'd capture per rotation axis. For example, choosing 3 axes and a value of $\pi/4$ would give 12 images, one for each $\pi/4$ radians around each of the axes.

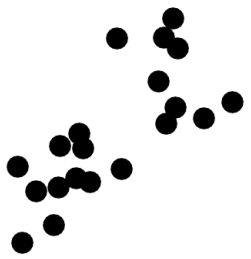
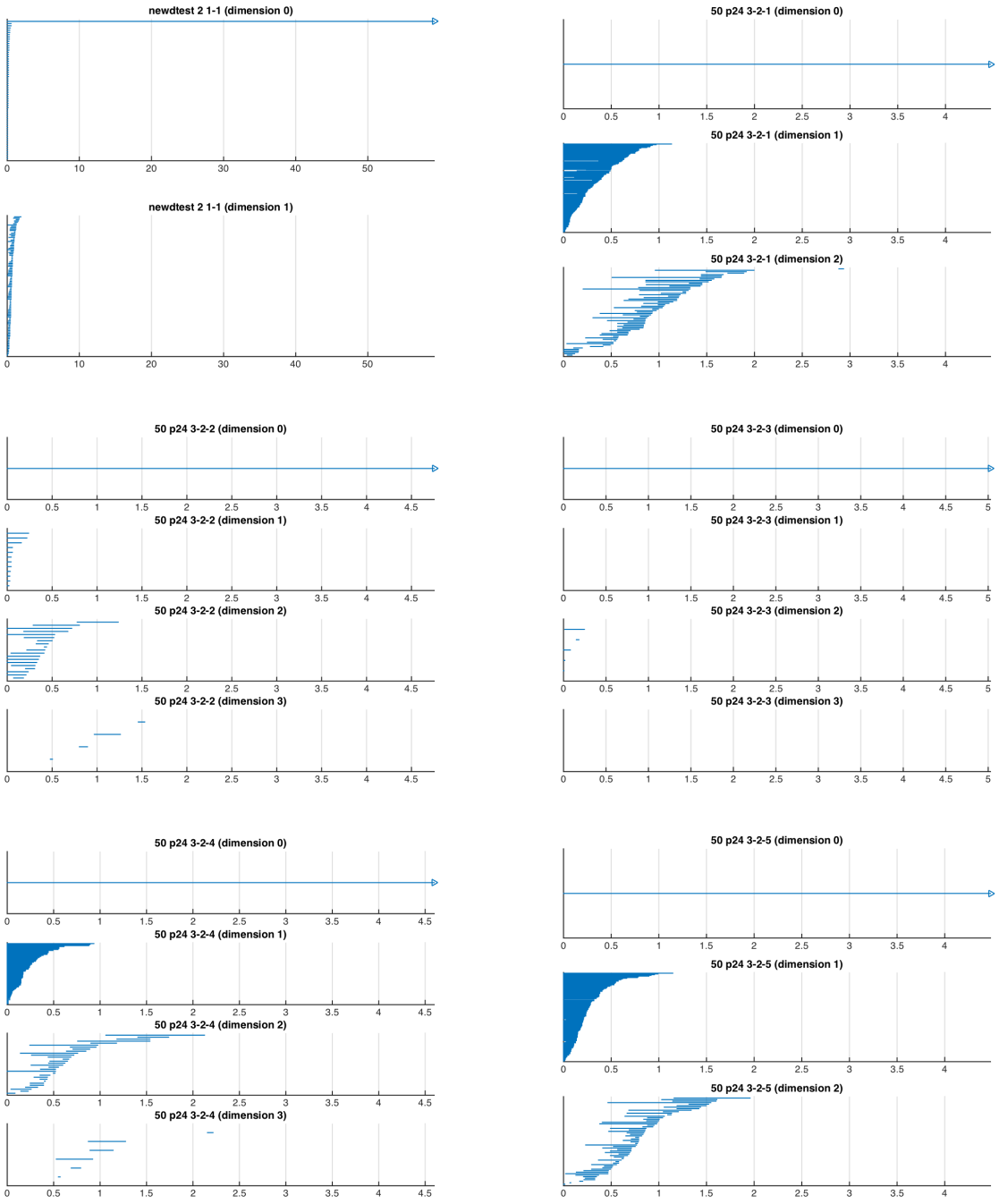


Fig. 5 - Example of a captured image of a molecule with 20 'atoms'.

Rather than saving the image each time and re-loading it as a matrix of color values, I converted the image directly into a matrix and then from a matrix into a column vector. Once I had produced such a vector for all the rotations, I used them to create a distance matrix D where $D(i, j)$ is the L2 vector distance between the ith and jth image vectors. Creating this distance matrix is necessary - in order to create simplicial complexes, we need a notion of distance between the data points. In this case, this L2 distance gives us a metric on our space. Once D is calculated, we put it into the javaplex along with chosen values for the simplicial complex construction, and javaplex calculates the persistent homology.

After verifying that the code was running correctly and giving out the correct homology for rotation around one axis (see figure 3), I moved on to the whole data set. Due to computational constraints, I could only have about 2400 projections total. The L2 metric didn't give great results - remember, we're looking for strongly persistent $\beta_0 = \beta_1 = \beta_2 = \beta_3 = 1$.



Note that not all of them show dimensions 0-3, this is due to either memory restraints or failure to get promising results at lower dimensions or both. While some of these look promising, there's nothing we can say other than that $\beta_0 = 1$ and maybe that $\beta_1 = 1$ in images 2, 5, and 6. By changing the lazy witness parameter, the number of landmark points, and the filtration value, these pictures can become a bit nicer; these 6 above are a characteristic sample of some of the nice- and less nice-looking results from the L2 distance metric.

The Weighted Graph Metric: There is a significant problem with the L2 metric, which I will now explain. Consider a black dot moving from the top left corner of an image to the bottom right corner, capturing images along the way:

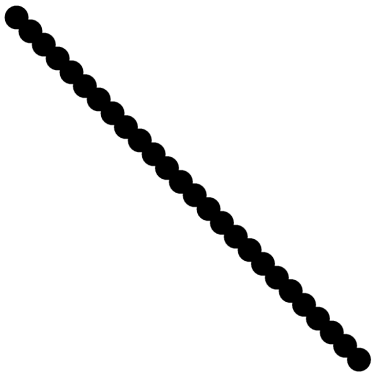


Fig. 6 - The total path of a dot moving from $(-1, 1)$ to $(1, -1)$.

Now consider these separate images of the dot along its path:

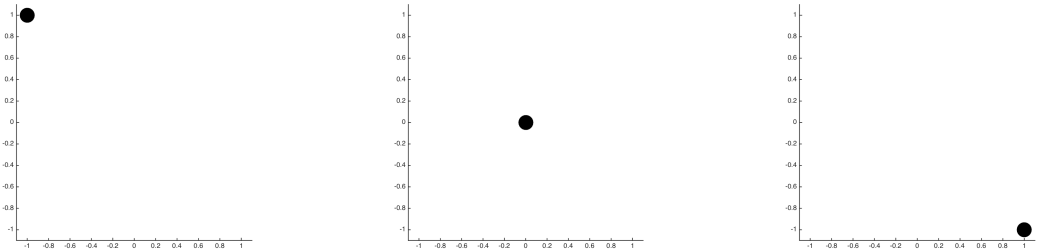


Fig. 7 - 3 separate locations of the dot.

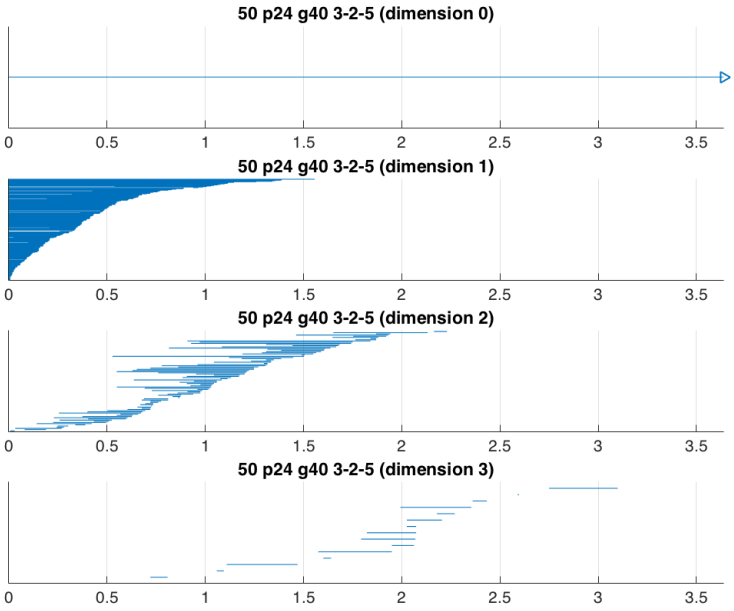
According to the L2 distance, these 3 pictures are all equidistant. That gives us some strange results - while we want the start and end points to be the farthest apart, our metric processes them as being relatively close. In computing homology, this could potentially create a different space than we would like - what we want is for our metric to include the position of the dot in the overall path, with it being closer to adjacent dots and farther from dots that are multiple steps away.

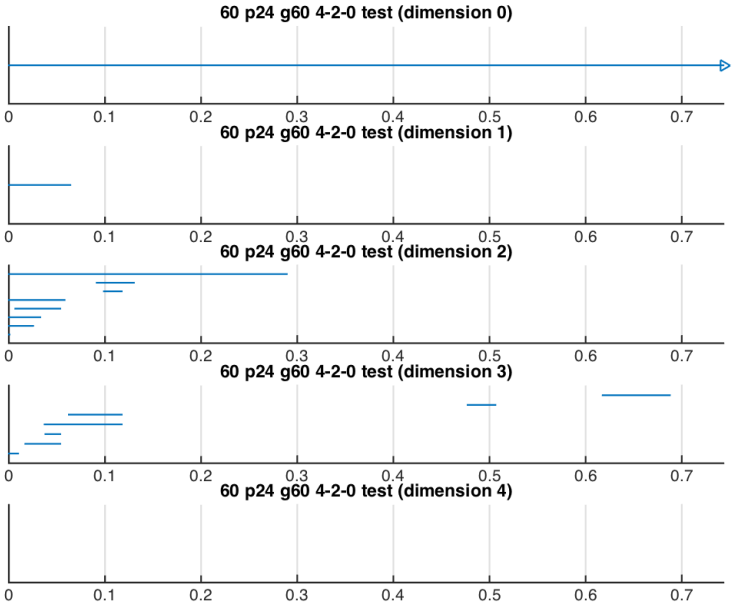
To do this, we use what is called a weighted graph metric. We see in our example above that the separate dots in figure 6 are not all equidistant, since some of them have overlap then immediately adjacent dots are closer than others. We use this knowledge in the following construction of our new metric.

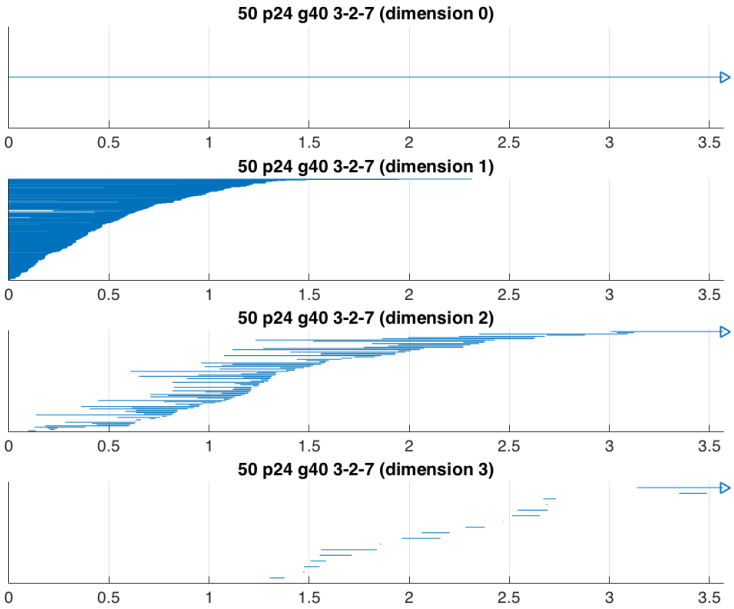
We compute the L2 distance matrix D on our point cloud and take some threshold t. We then construct an undirected, weighted graph G from D. If $D(i, j) \geq t$, then we take away the edge (ij) in G. Then, we create a new distance matrix D_g from this graph, where $D_g(i, j)$ is the minimum distance to travel on the graph from vertex i to vertex j (and since it's a distance matrix, it's of course symmetrical). Returning to the dot example, the threshold to choose would be the distance between the dots in figure 6. Since there's

a little bit of pairwise overlap between the images in figure 6, the distance is going to be smaller than this threshold. Thus the distance between the 3 dots in figure 6 becomes the additive distance between all of the slight overlaps between them, and our metric reflects that the middle dot is between the other two.

It's clear why this would be preferable to the L2 distance for our purposes. Indeed, through similar variance of the parameters as before, we get much better results.







Though only one of these gives us a clear $\beta_0 = \beta_1 = \beta_2 = \beta_3 = 1$, they all give much better results than we achieved with L2. Additionally, with better computation power and memory, we could further refine the results to obtain an even better picture. The most important parts of this process are to take images that are close enough together to have overlap, and to select a proper threshold for the weighted graph metric computation. Coming up with a 'proper threshold' is difficult, and involves a bit of experimenting with variables; you want to select one that is large enough to allow for a fully-connected graph (otherwise, you get values of infinity in your distance matrix) yet small enough to only leave extremely similar/'close' data points unaffected. Thinking back to the dot-path example, the threshold we would choose would be a bit below the maximum distance between 2 points in the L2 metric. Thus, 2 images with dots that overlapped would be unaffected, but if the dots didn't overlap then the distance between them would increase dramatically. I followed a similar idea in choosing the threshold by having Matlab output the maximum, mean, and minimum values in the columns of the L2 distance matrix, and taking a threshold a bit below the mean (I also took other thresholds near the max and min distances to compare the results). One important note on the weighted graph metric is that for my particular data, I had to forego a high number of axes for smaller rotation steps around each axis - I needed each partial rotation to be small enough that images in adjacent rotations would overlap more than average.

Future Work: This weighted graph metric has only been shown to work better compared to the L2 distance metric for highly similar images. In general, the weighted graph metric seems well-suited for dealing with highly similar data, for instance when taking many measurements in quick succession of a changing situation. It would be interesting to see other metrics applied to image processing, or to see the weighted graph metric in different situations (for instance compare it to the 'image patches' used in Carlsson et al. "On

the Local Behavior of Spaces of Natural Images").

5 Code

I used the Matlab plugin Javaplex, available for download on Github at:
`http://appliedtopology.github.io/javaplex/`

The following are the main code snippets I used. The 'computeHomology' function is an edited version of code included in the javaplex download file.

```
% Creates a molecule with npoints points, rotates it and
% reads all the projection images in as vectors.


function vectors = createImgProjections(npoints, naxes, theta)

nrotations = 2*pi/theta;
molecule = createRandomMolecule(npoints);
axes = createRandomAxes(naxes);
vectors = [];

for i = 1:naxes
   raxis = axes(i,:);
   m = molecule;
   for j = 1:nrotations - 1
      m = rotateMolecule(raxis, theta, m);
      scatter(m(:,1),m(:,2),220,'filled','black');
      axis([-1.6 1.6 -1.6 1.6]);
      axis square;
      axis off;
      [img, map] = frame2im(getframe(gcf));
      img = img(:,:,1);
      img = reshape(im2double(img),[numel(img),1]);
      vectors = [vectors, img];
      delete(gca);
   end
close all;
end
--------------------

% Computes and displays the persistent homology of our rotation projections,
% using a Lazy Witness construction with the distance matrix as the metric.

function computeHomology(distanceMatrix, max_dimension, num_divisions, nu, filename);
```

```
   import edu.stanford.math.plex4.*;

   num_landmark_points = 100;
m_space = metric.impl.ExplicitMetricSpace(distanceMatrix);

   % create a sequential maxmin landmark selector
landmark_selector = api.Plex4.createMaxMinSelector(m_space, num_landmark_points);
R = landmark_selector.getMaxDistanceFromPointsToLandmarks()
max_filtration_value = 2*R;

   % create a lazy witness stream
stream = streams.impl.LazyWitnessStream(landmark_selector.getUnderlyingMetricSpace(),
landmark_selector, max_dimension, max_filtration_value, nu, num_divisions);
stream.finalizeStream();

   % print out the size of the stream
num_simplices = stream.getSize()
if num_simplices > 2500000
error('Num_simplices too high, likely to freeze/crash matlab.  Try again with
lower bounds.');
end

   % get persistence algorithm over Z/2Z
persistence = api.Plex4.getModularSimplicialAlgorithm(max_dimension, 2);

   % compute the intervals
intervals = persistence.computeIntervals(stream)

   % create the barcode plots
options.filename = filename;
options.max_filtration_value = max_filtration_value;
options.max_dimension = max_dimension - 1;
plot_barcodes(intervals, options);
```

6 Resources

M. A. Armstrong, *Basic Topology,* pp.173-178.
G. Carlsson, T. Ishkanov, V. de Silva, and A. Zomorodian, *On the Local Behavior of Spaces of Natural Images*.
G. Carlsson, *Toplogical Pattern Recognition for Point Cloud Data*.
G. Carlsson, *Topology and Data*.
V. de Silva and G. Carlsson, *Topological Estimation Using Witness Complexes*.

7 Acknowledgements

I'd like to thank Gunnar Carlsson for working with me this summer and teaching me an incredible amount, and to thank George Schaeffer and the Stanford Undergraduate Research Institute in Mathematics (SURIM) for making this all possible.