Manifold Learning Techniques: *So which is the best?*

Todd Wittman

Math 8600: Geometric Data Analysis

Instructor: Gilad Lerman

Spring 2005

<u>Note</u>: This presentation does not contain information on LTSA, which was added to MANI after this presentation was given. Of the NLDR methods considered, LTSA seems to perform the best on the examples included in MANI.

Manifold Learning for Dimensionality Reduction

Goal: We want to map a D-dimensional data set X to a d-dimensional data set Y, preserving the local geometries on the original manifold as much as possible.

$$X \in \Re^D \rightarrow Y \in \Re^d$$

Methods we've learned

- MDS
- PCA
- ISOMAP
- LLE
- Hessian LLE
- Laplacian Eigenmap
- Diffusion Maps
- KNN Diffusion (Mauro)

So which is the best?
First let's make some predictions...

PCA

- Summary
  - Just SVD of covariance matrix XX'.
  - Take largest d eigenvectors.
- Predictions
  - Doesn't really infer the geometry of the manifold.
  - Assumes the data Y represents the projection onto a d-dimensional space.
  - But it's extremely fast, so it might be good to give it first crack at your data.

MDS

- Summary
  - Iterates to answer by learning the correct eigen decomposition.
- Predictions
  - Like PCA, does not seem to have any built-in mechanism for determining the geometry of the manifold.
  - Also much slower than PCA.
  - Requires parameters describing learning rate and number of iterations.

General Framework

- ISOMAP, LLE, Hessian, Laplacian, & KNN Diffusion are all based on KNN graphs.

- The graph-based algorithms have 3 basic steps.
  - 1. Find K nearest neighbors.
  - 2. Estimate local properties of manifold by looking at neighborhoods found in Step 1.
  - 3. Find a global embedding that preserves the properties found in Step 2.

ISOMAP *(Tennenbaum, DeSilva, & Langford)*

- Summary
  - Build graph from K Nearest Neighbors.
  - Run MDS.
- Predictions
  - Since MDS is slow, ISOMAP will be very slow.
  - Need estimate of K.
  - Assumes data set is convex (no holes).
  - ISOMAP removes outliers in pre-processing. So is it extra-sensitive to noise?

LLE *(Saul & Roweis)*

- Summary *(see Effie's talk)*
  - Build graph from K Nearest Neighbors.
  - Determine reconstruction weights by assuming neighborhoods are locally linear and insuring invariance.
  - Determine embedding.
- Predictions
  - If it's locally linear, can it handle curvature and corners?
  - What about noise?

Laplacian Eigenmap *(Belkin & Nyogi)*

- Summary
  - Build graph from K Nearest Neighbors.
  - Construct weighted adjacency matrix with Gaussian kernel.
  - Compute embedding from normalized Laplacian.
    - minimize $\int \|\nabla f\|^2 dx$ subject to $\|f\| = 1$
- Predictions
  - Assumes each point lies in the convex hull of its neighbors. So it might have trouble at the boundary.
  - Will have difficulty with non-uniform sampling.

Hessian LLE *(Donoho & Grimes)*

- Summary
  - Build graph from K Nearest Neighbors.
  - Estimate tangent Hessians.
  - Compute embedding based on Hessians.

$$f : X \rightarrow \Re \quad Basis\left(null\left(\int \left\| H_f(x) \right\| dx\right)\right) = Basis(X)$$

- Predictions
  - Specifically set up to handle non-convexity.
  - Slower than LLE & Laplacian.
  - Will perform poorly in sparse regions.
  - Only method with convergence guarantees.

Diffusion Map *(Coifman & Lafon)*

- Summary
  - Find Gaussian kernel $K(x, y) = \exp\left(-\frac{\|x-y\|^2}{\sigma} \right)$
  - Normalize kernel

$$K^{(\alpha)}(x, y) = \frac{K(x, y)}{p^{\alpha}(x)\, p^{\alpha}(y)} \quad \text{where} \quad p(x) = \int K(x, y) P(y) dy$$

  - Apply weighted graph Laplacian.

$$A(x, y) = \frac{K^{(\alpha)}(x, y)}{d^{(\alpha)}(x, y)} \quad \text{where} \quad d^{(\alpha)}(x, y) = \int K^{(\alpha)}(x, y) P(y) dy$$

  - Compute SVD of A
- Predictions
  - Doesn’t seem to infer geometry directly.
  - Need to set parameters alpha and sigma.

KNN Diffusion *(Mauro)*

- Summary
  - Build graph from K nearest neighbors.
  - Run Diffusion Map on graph.
- Predictions
  - Should infer geometry better than Diffusion Map.
  - Now we have to set the parameters alpha, sigma, and K.

How do we compare the methods?

- Speed
- Manifold Geometry
- Non-convexity
- Curvature
- Corners
- Noise
- Non-uniform Sampling
- Sparse Data
- Clustering
- High-Dimensional Data: *Can the method process image manifolds?*
- Sensitivity to Parameters
  - K Nearest Neighbors: *Isomap, LLE, Hessian, Laplacian, KNN Diffusion*
  - Sigma: *Diffusion Map, KNN Diffusion*

Testing Examples

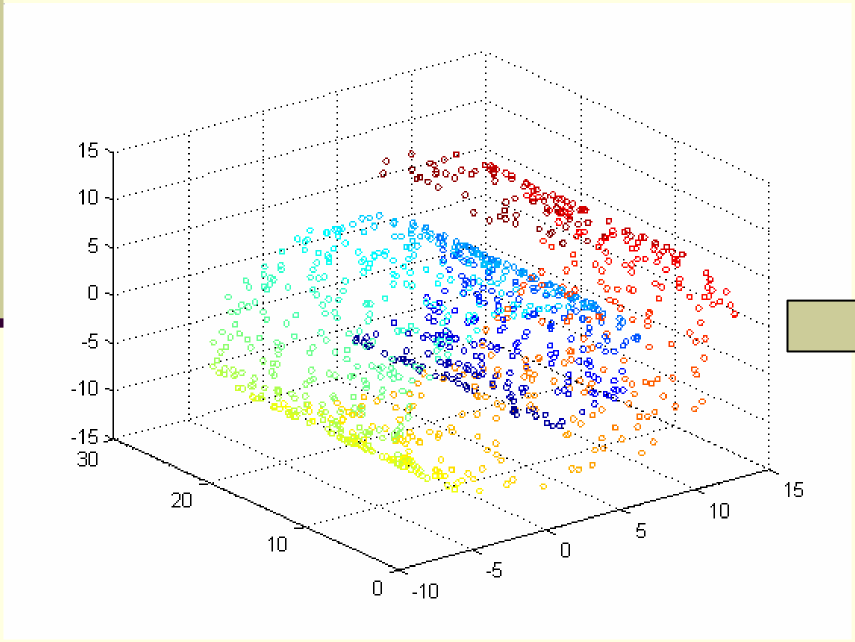
I've generated a set of examples to test the methods. Many of these examples were taken from the original papers.

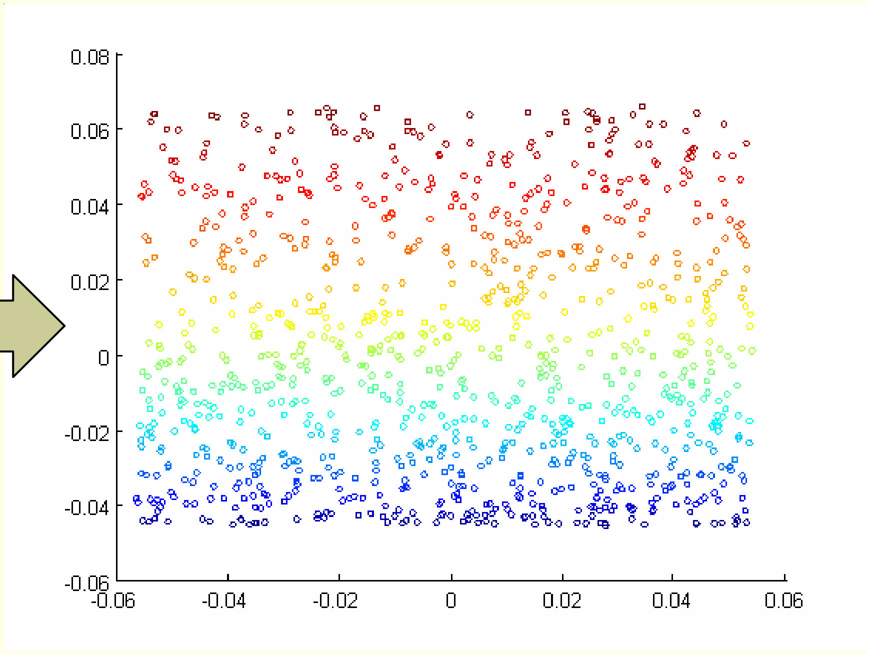
- Swiss Roll
- Swiss Hole
- Punctured Sphere
- Corner Planes
- 3D Clusters
- Twin Peaks
- Toroidal Helix
- Gaussian
- Occluded Disks

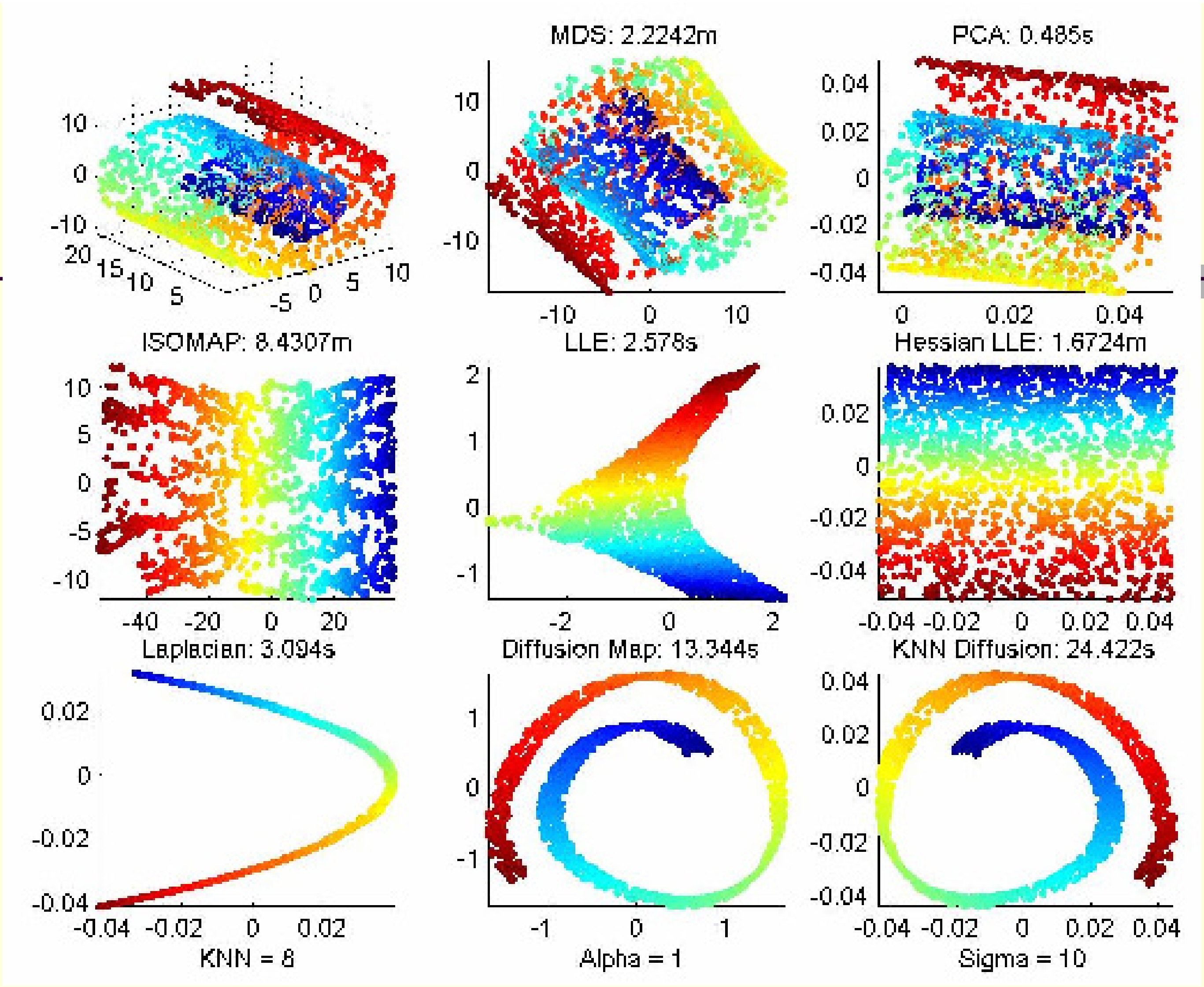
- Except for the last one, we map 3D to 2D data sets so we can visualize the results.
- We'll compare the speed and sensitivity to parameters throughout.

Manifold Geometry

- First, let’s try to unroll the Swiss Roll.
- We should see a plane.



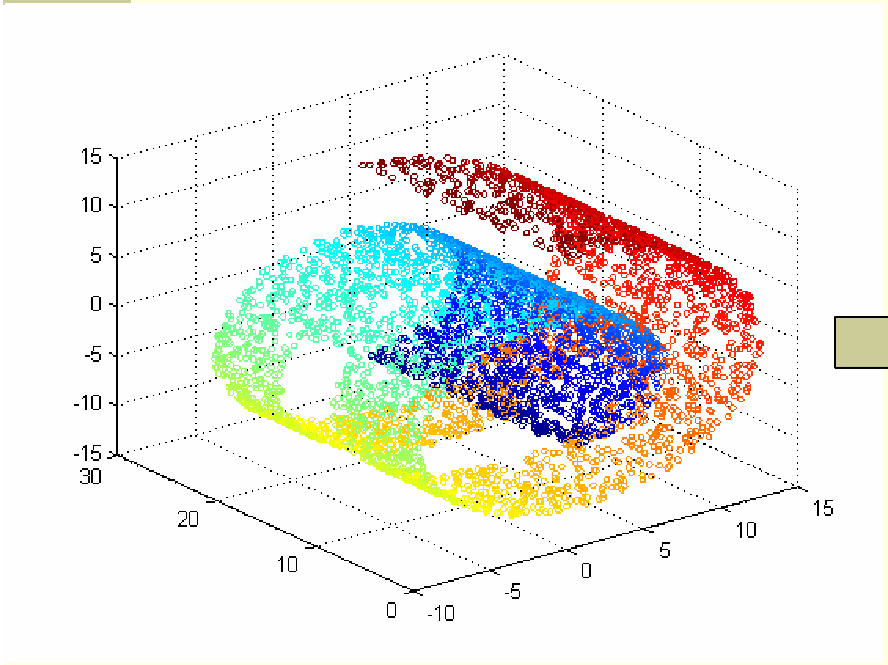


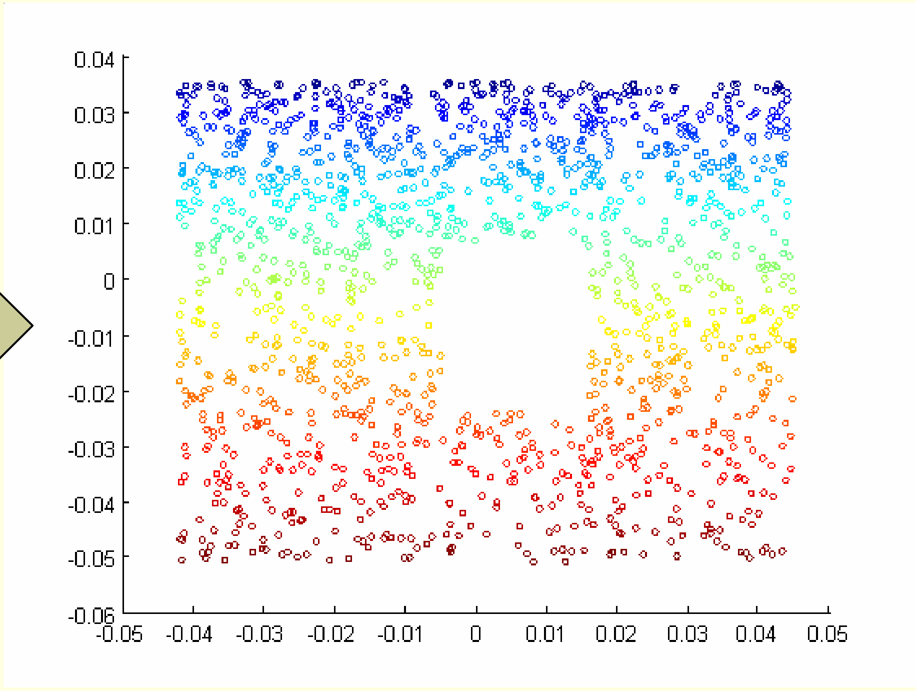


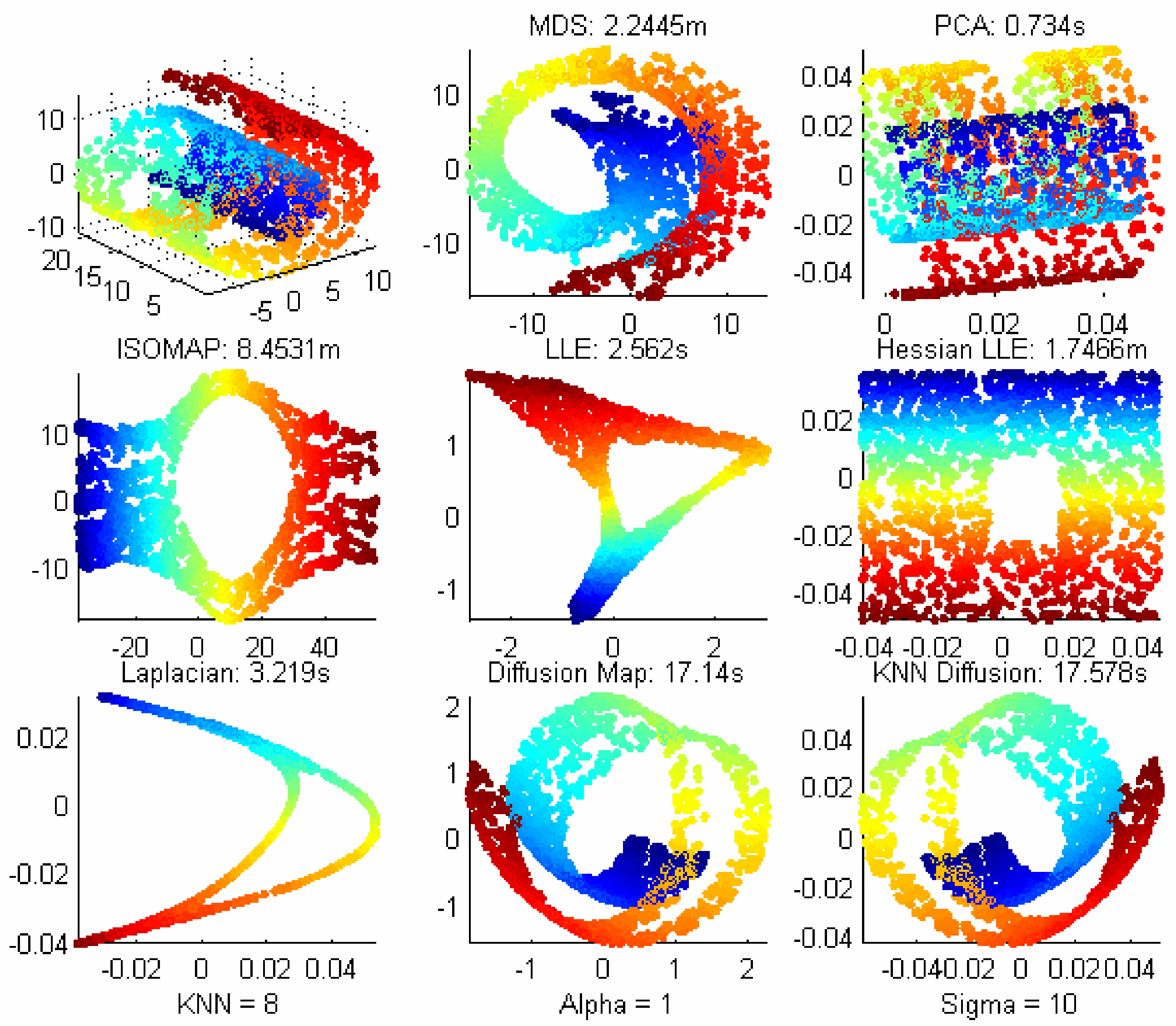
Hessian LLE is pretty slow, MDS is very slow, and ISOMAP is extremely slow.
MDS and PCA don't can't unroll Swiss Roll, use no manifold information.
LLE and Laplacian can't handle this data.
Diffusion Maps could not unroll Swiss Roll for any value of Sigma.

Non-Convexity

- Can we handle a data set with a hole?
- Swiss Hole: Can we still unroll the Swiss Roll when it has a hole in the middle?



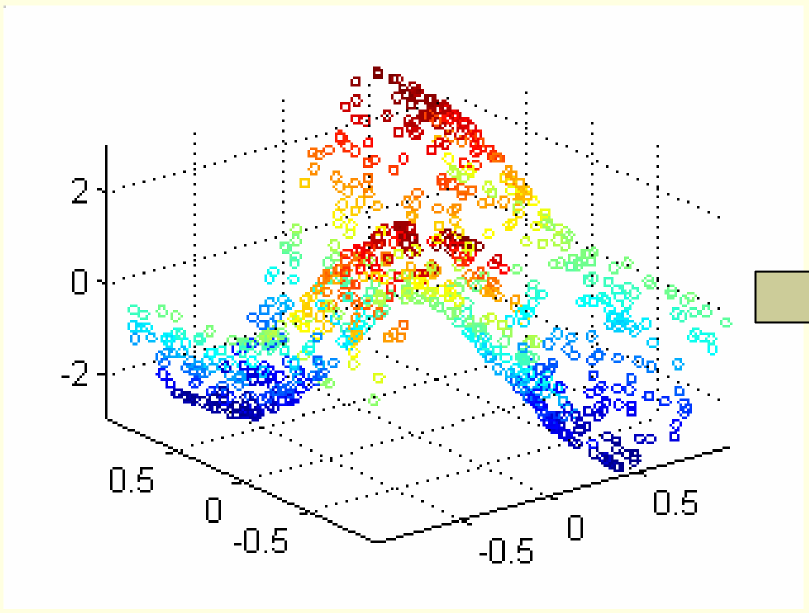


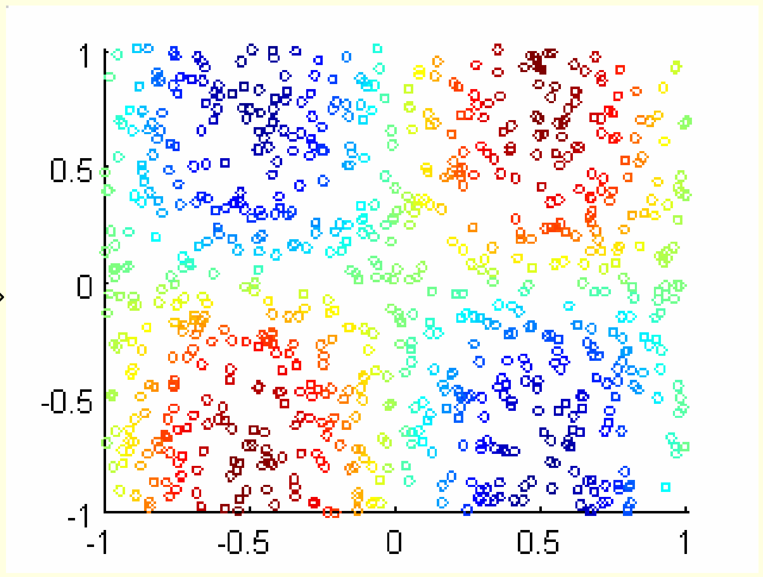


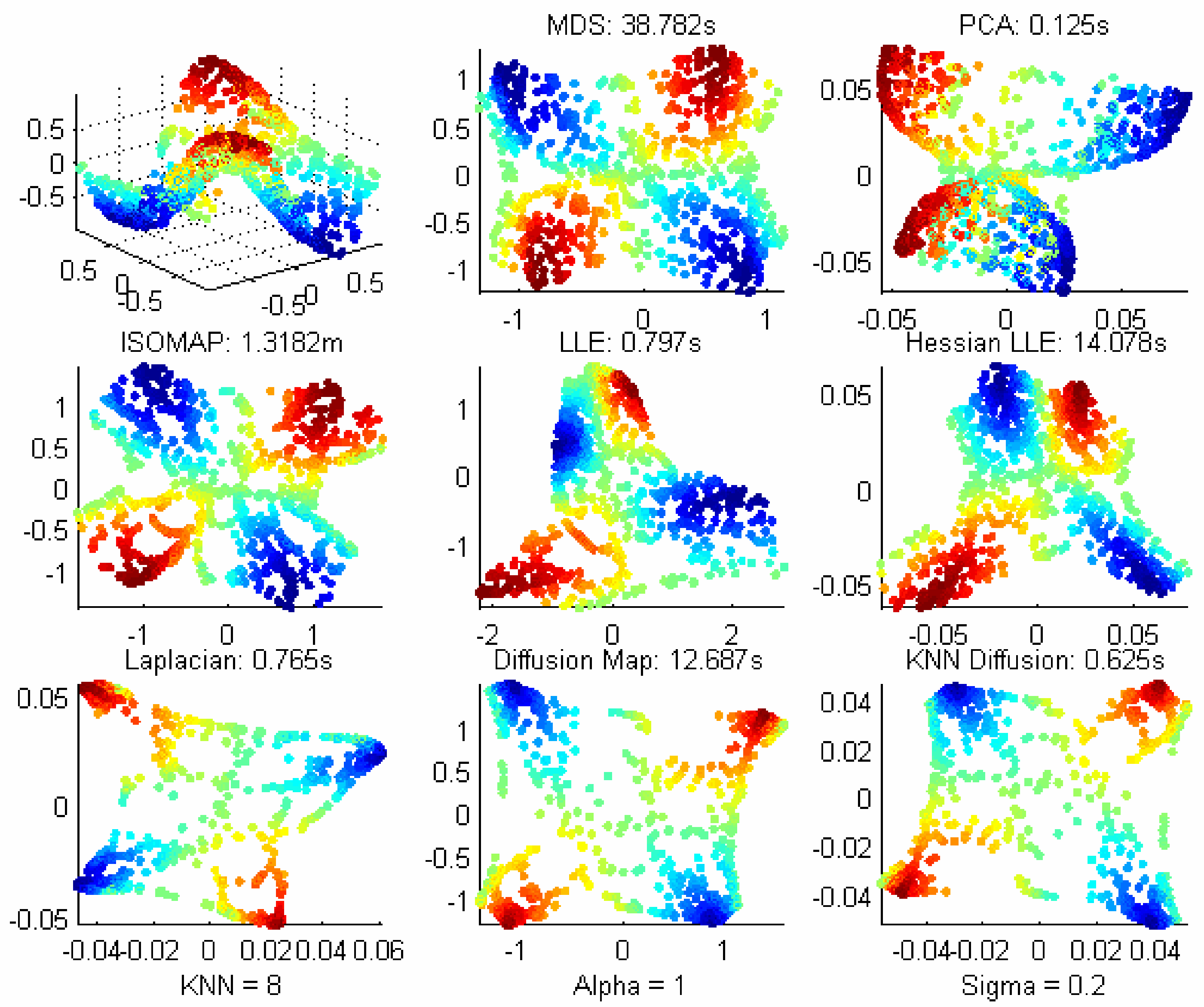
Only Hessian LLE can handle non-convexity.
ISOMAP, LLE, and Laplacian find the hole but the set is distorted.

Manifold Geometry

- Twin Peaks: fold up the corners of a plane.
- LLE will have trouble because it introduces curvature to plane.



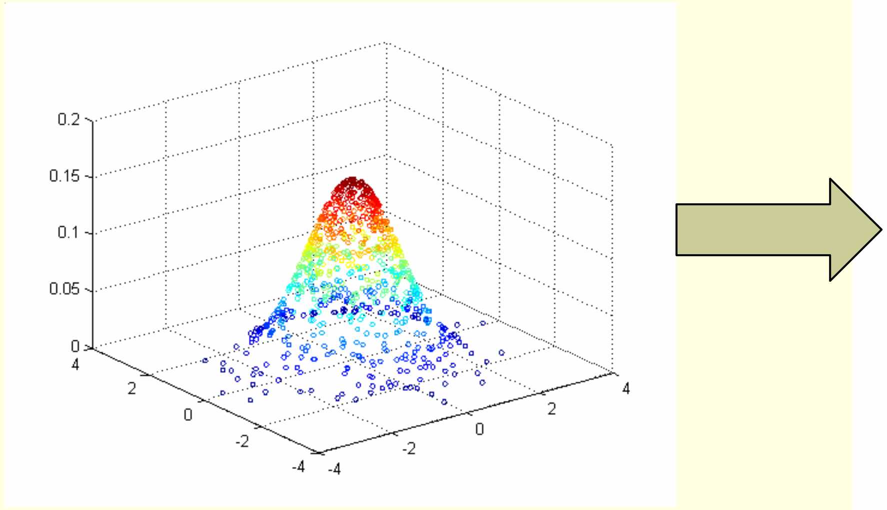


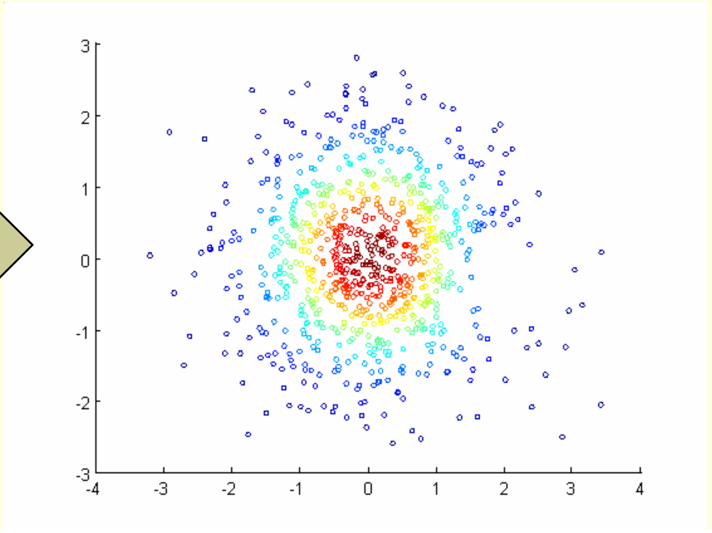


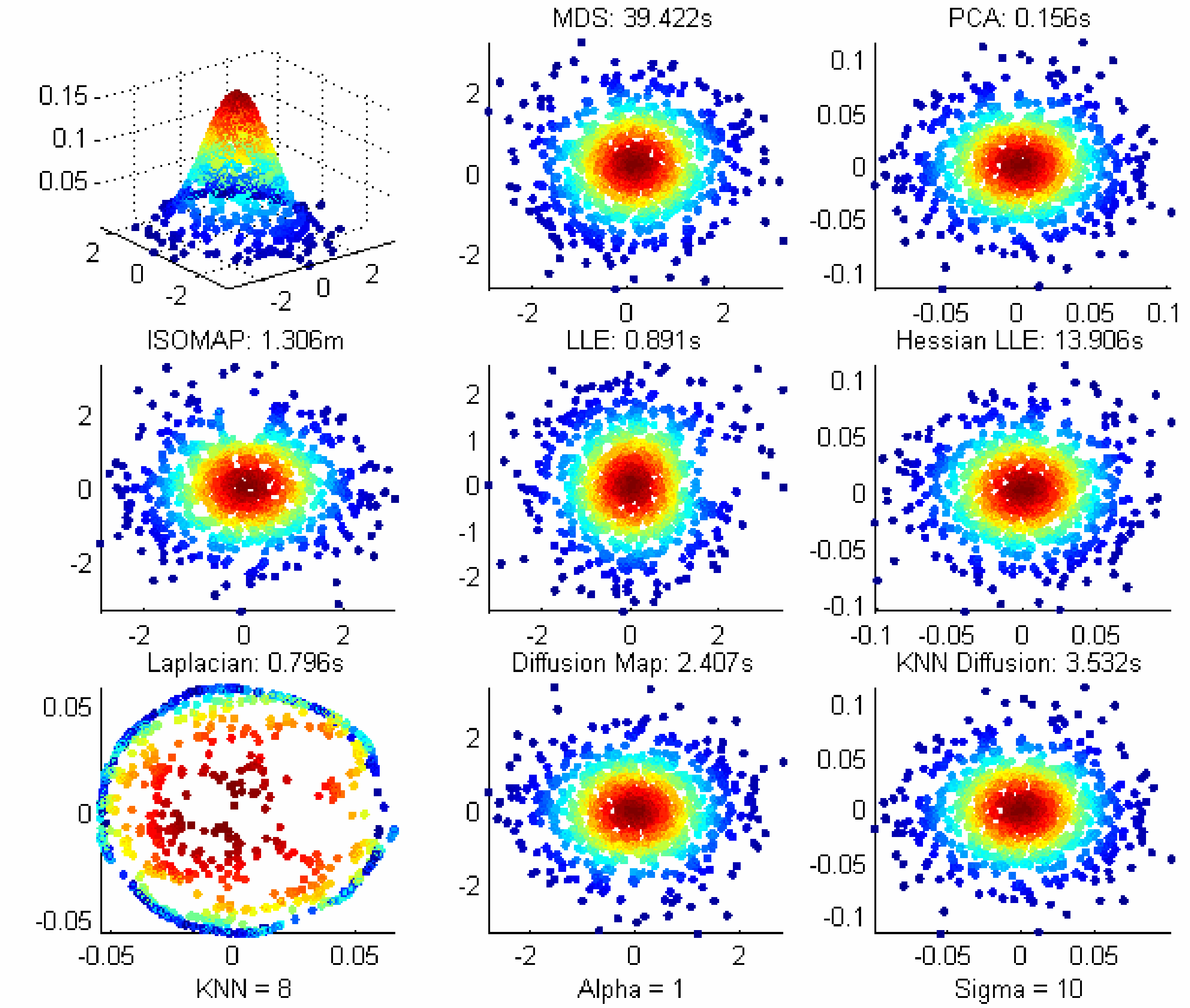
PCA, LLE, and Hessian LLE distort the mapping the most.

Curvature & Non-uniform Sampling

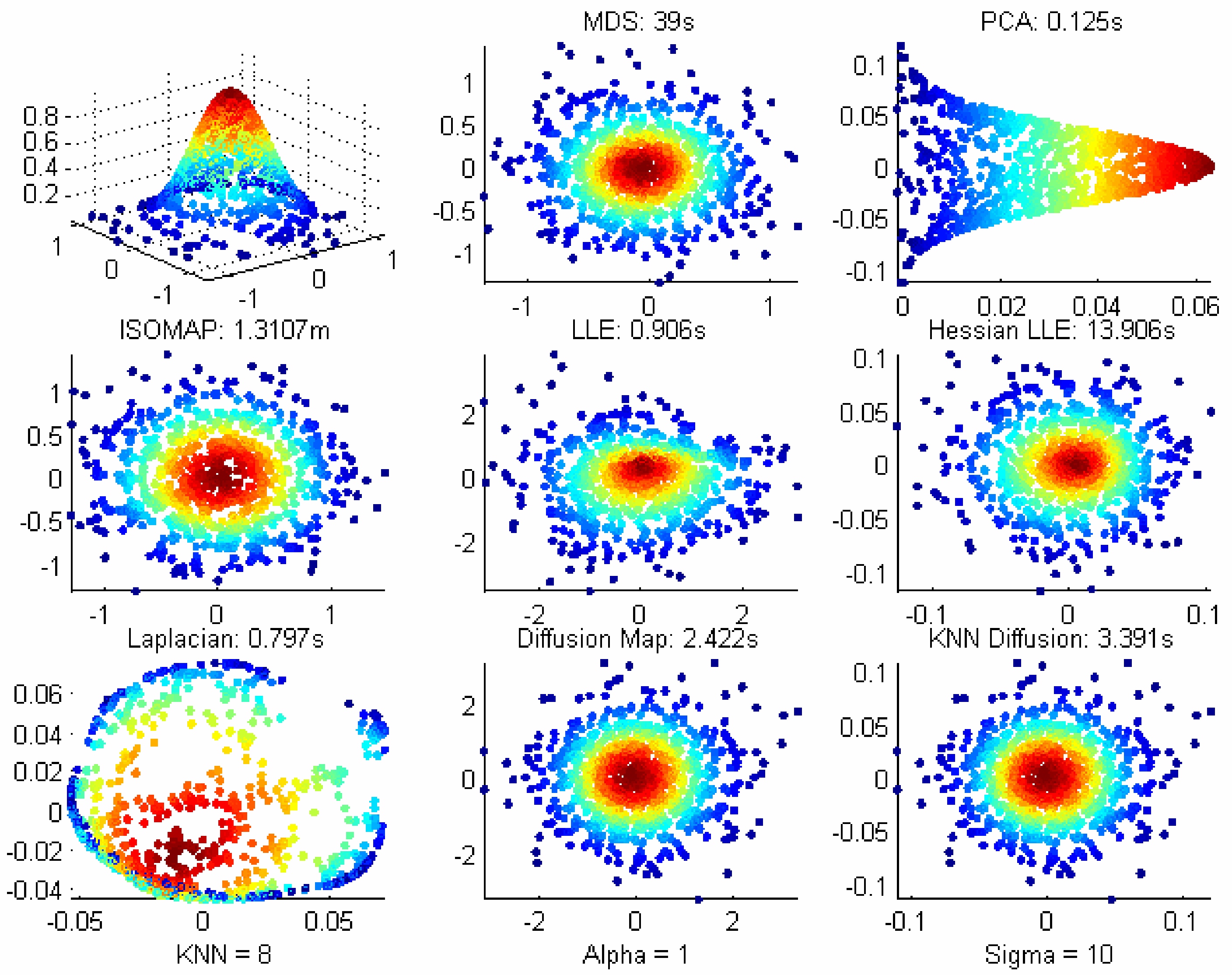
- Gaussian: We can randomly sample a Gaussian distribution.
- We increase the curvature by decreasing the standard deviation.
- Coloring on the z-axis, we should map to concentric circles.



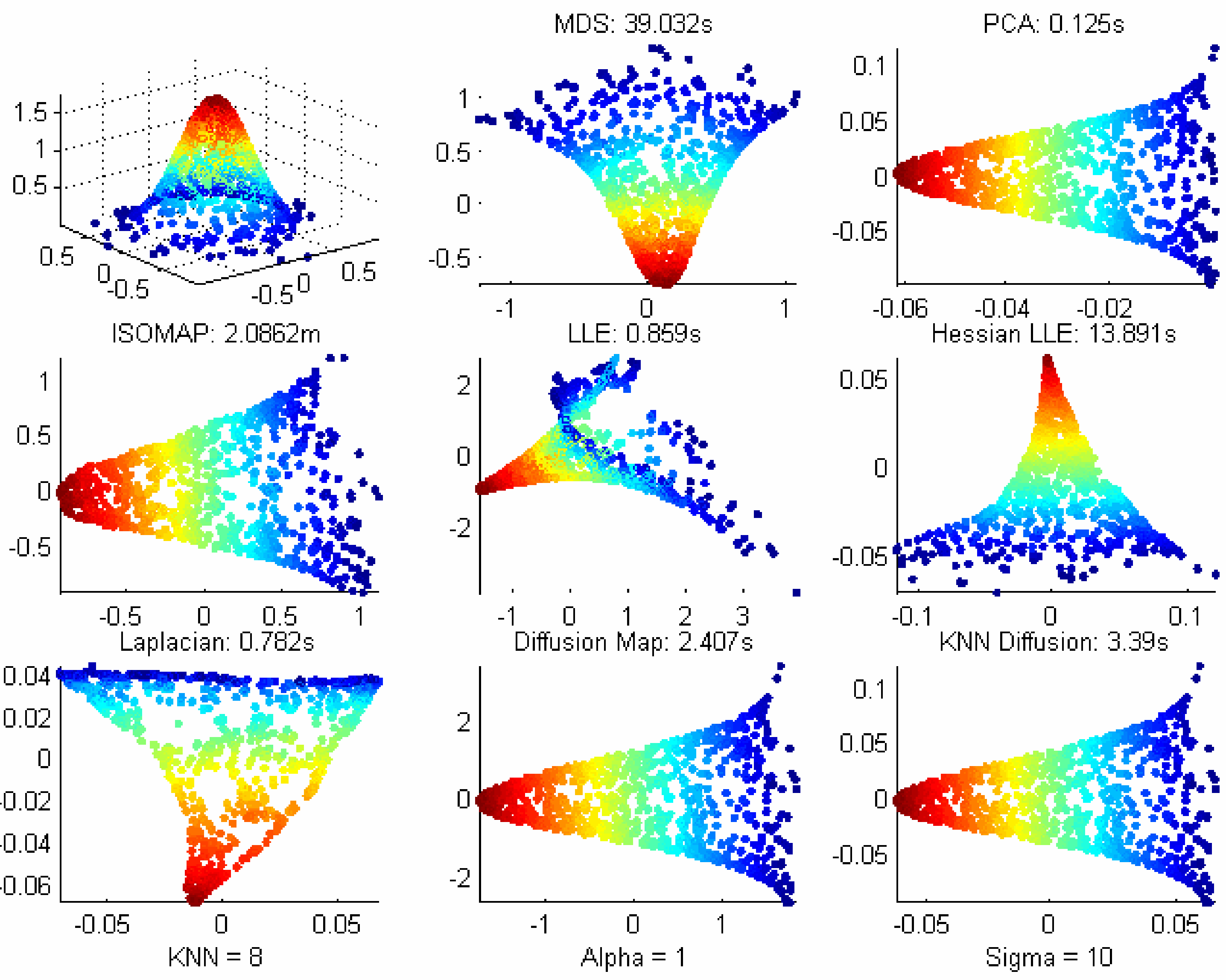




For std = 1 (low curvature), MDS and PCA can project accurately.
Laplacian Eigenmap cannot handle the change in sampling.



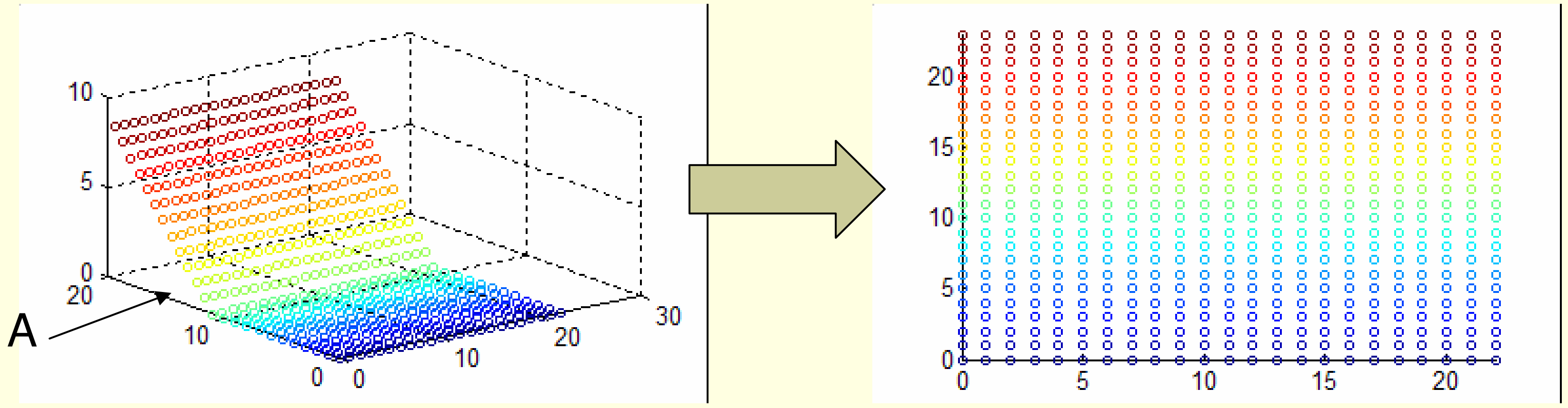
For std = 0.4 (higher curvature), PCA projects from the side rather than top-down. Laplacian looks even worse.



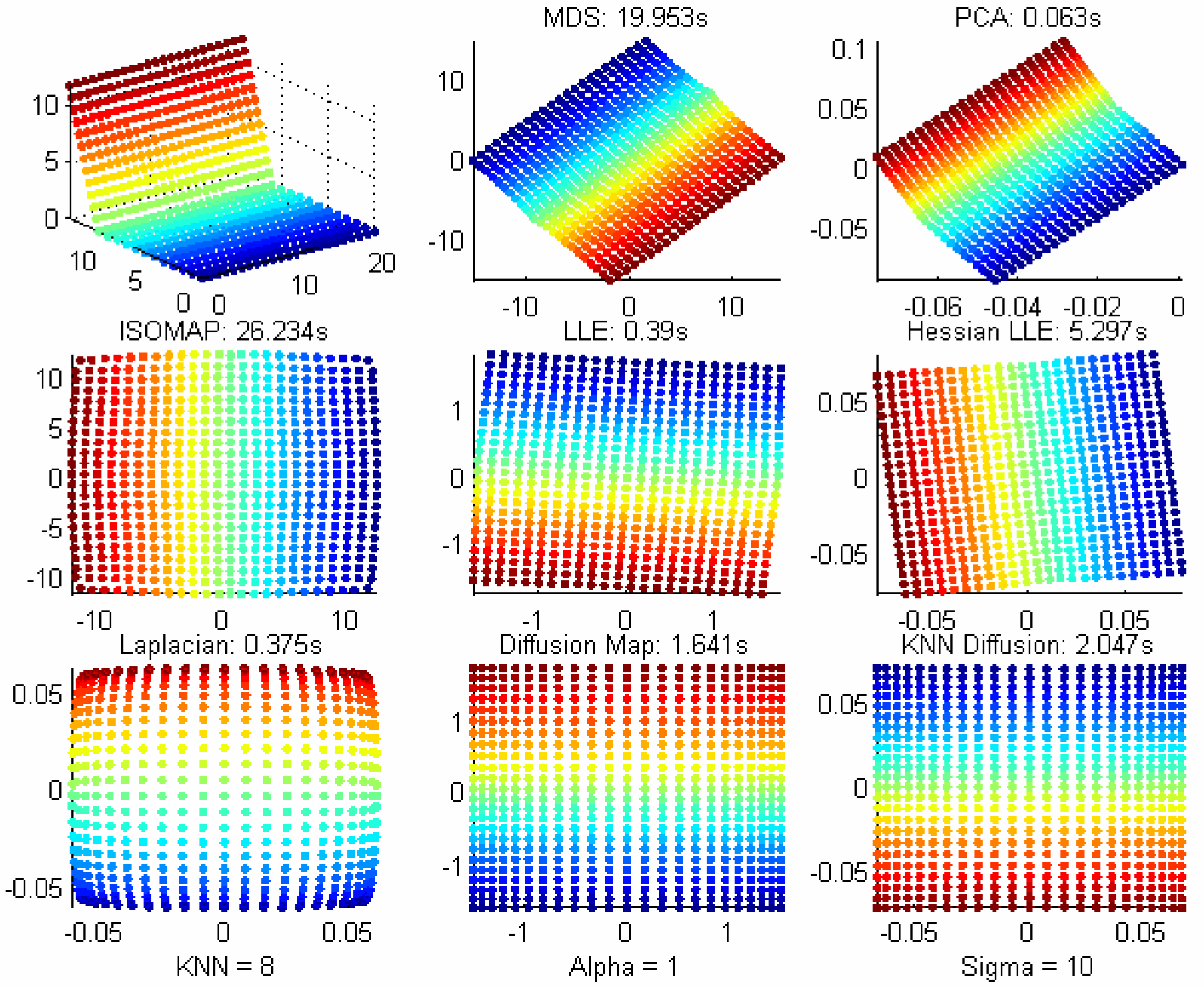
For std = 0.3 (high curvature), none of the methods can project correctly.

Corners

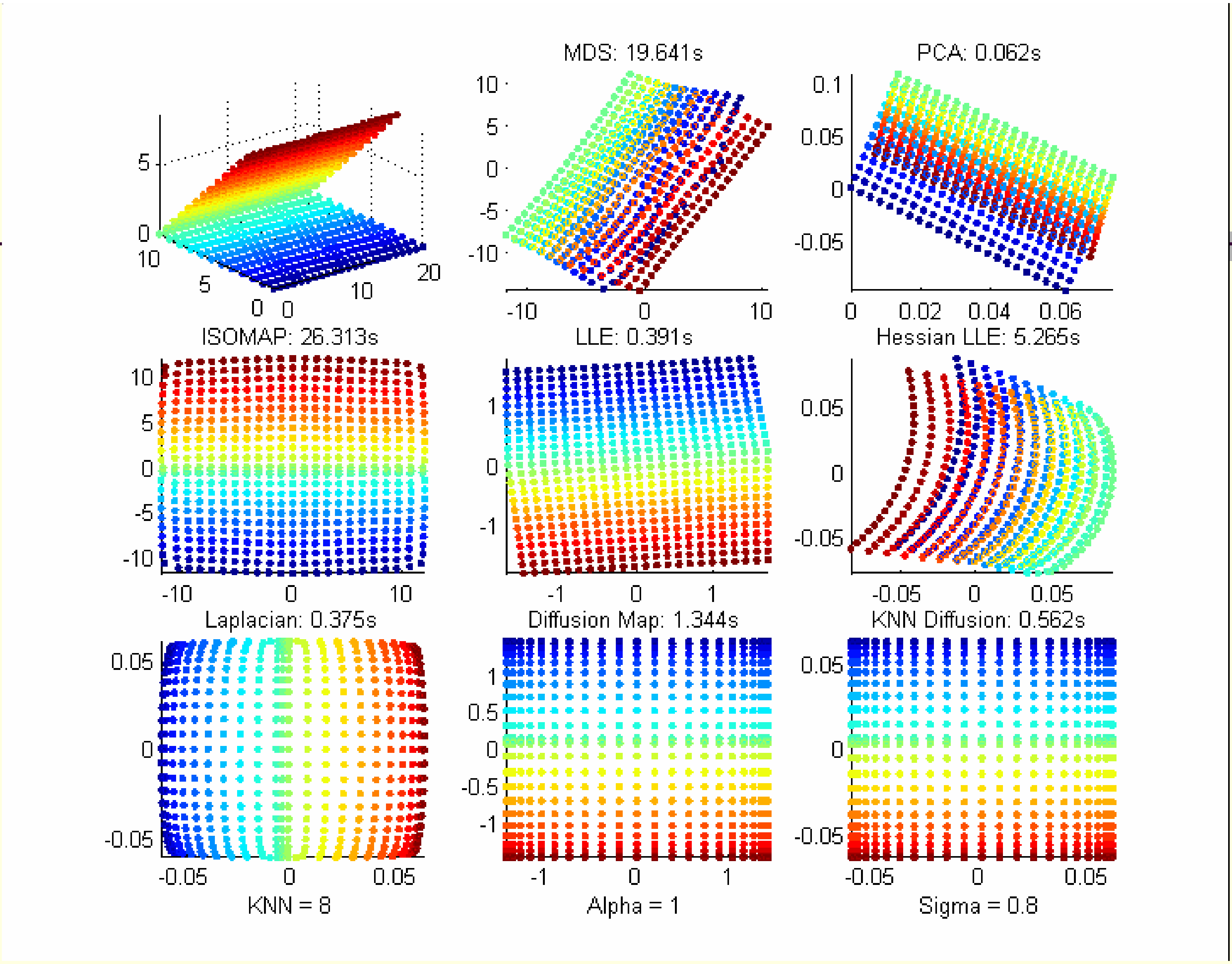
- Corner Planes: We bend a plane with a lift angle A.
- We want to bend it back down to a plane.



- If A > 90, we might see the data points written on top of each other.



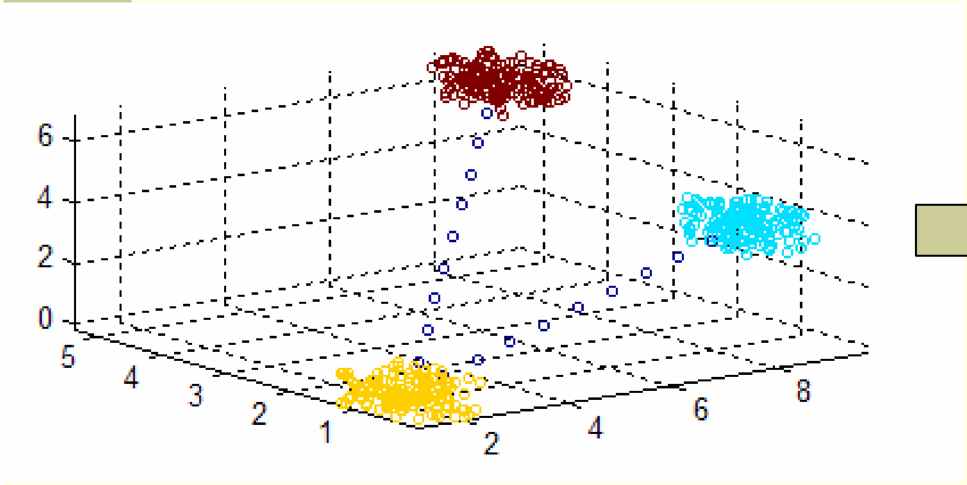
For angle A=75, we see some disortions in PCA and Laplacian.

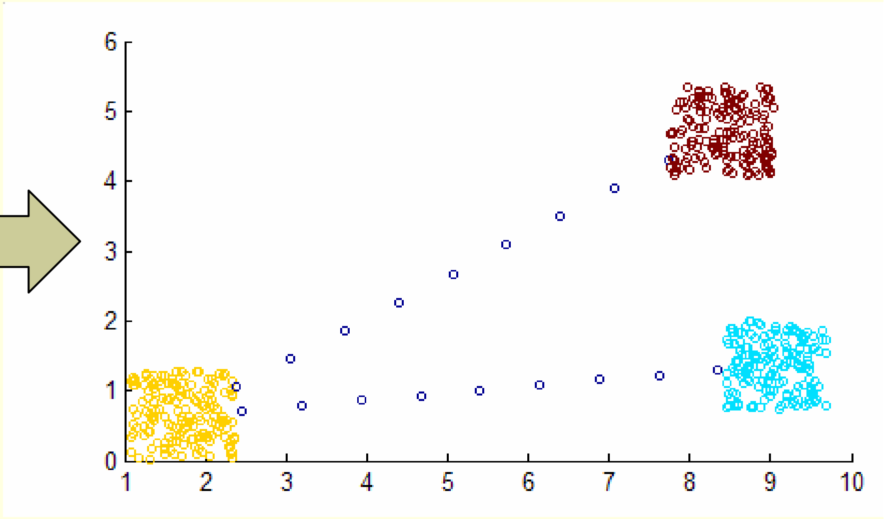


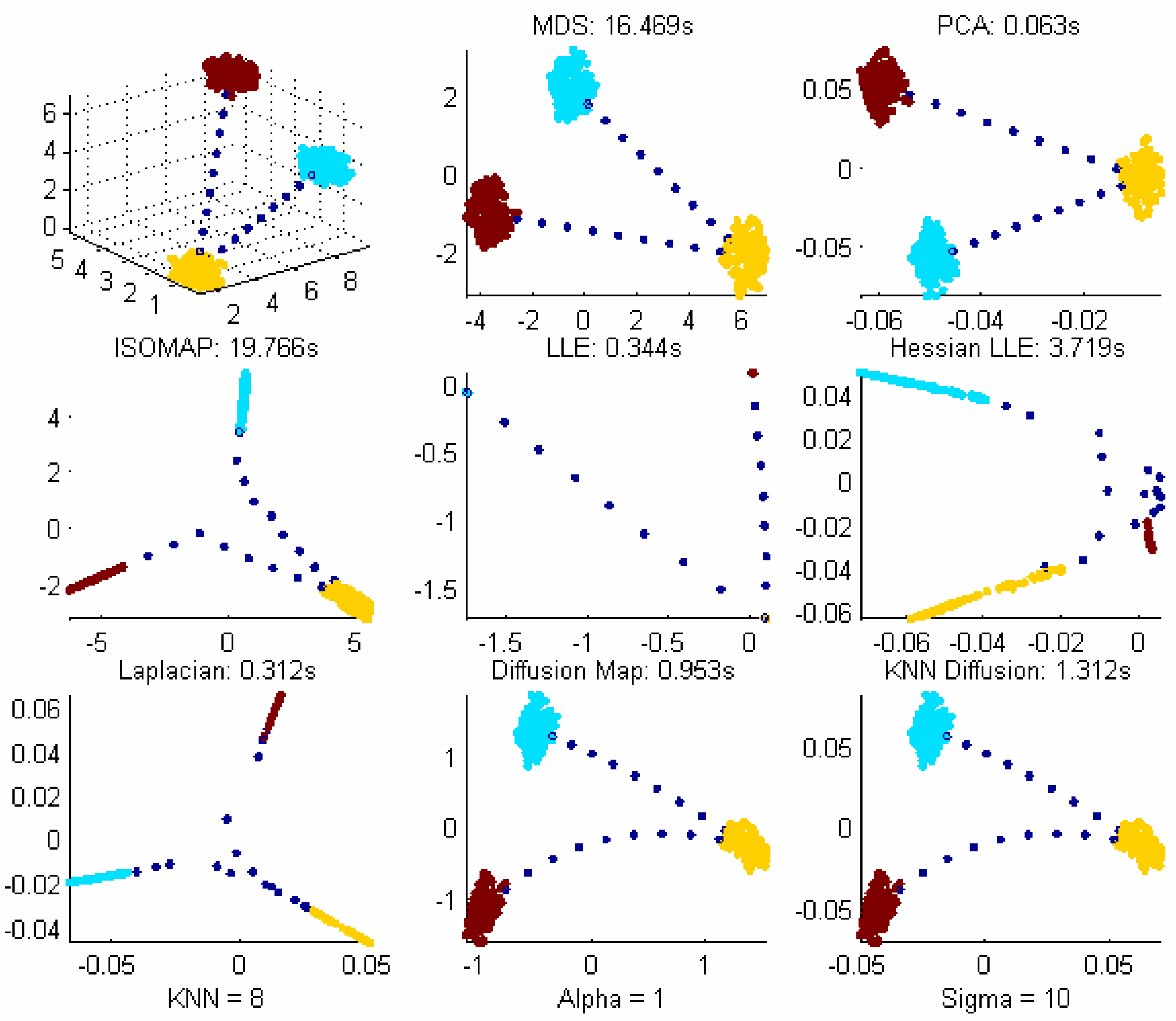
For A = 135, MDS, PCA, and Hessian LLE overwrite the data points.
Diffusion Maps work very well for Sigma < 1.
LLE handles corners surprisingly well.

Clustering

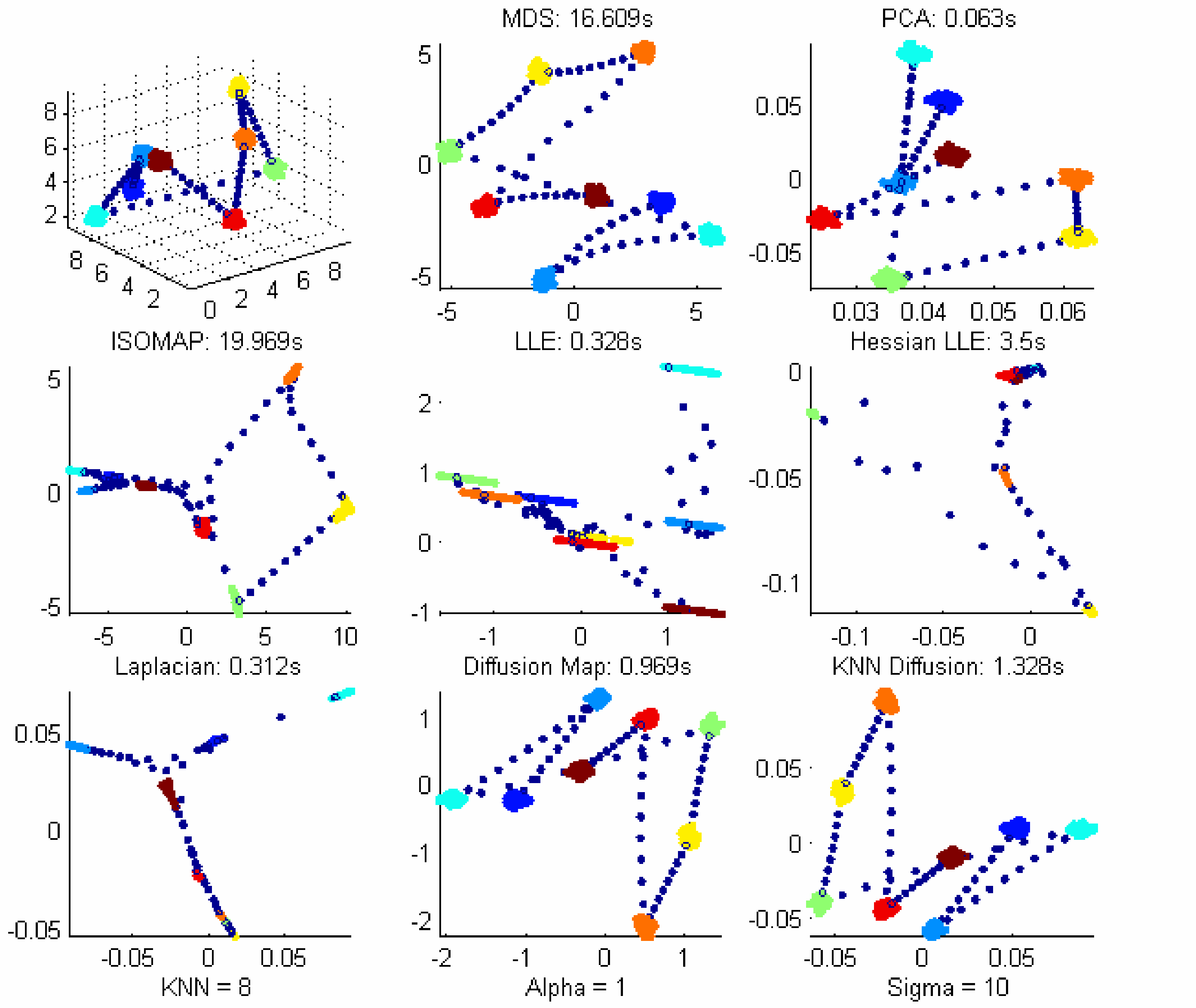
- A good mapping should preserve clusters in the original data set.
- 3D Clusters: Generate M non-overlapping clusters with random centers. Connect the clusters with a line.







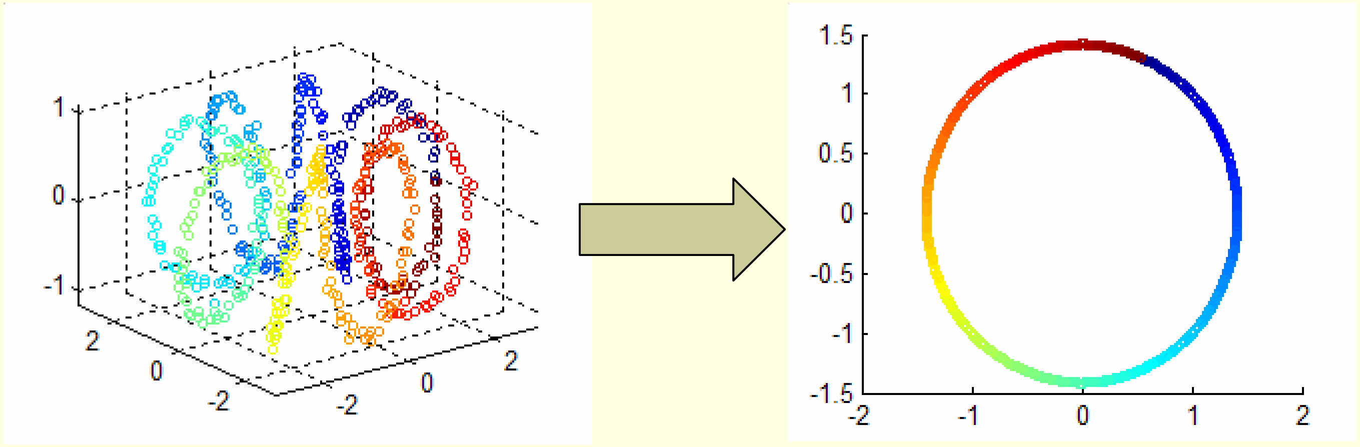
For M = 3 clusters, MDS and PCA can project correctly.
Diffusion Maps work well with large Sigma.
LLE compresses each cluster into a single point.
Hessian LLE has trouble with the sparse connecting lines.



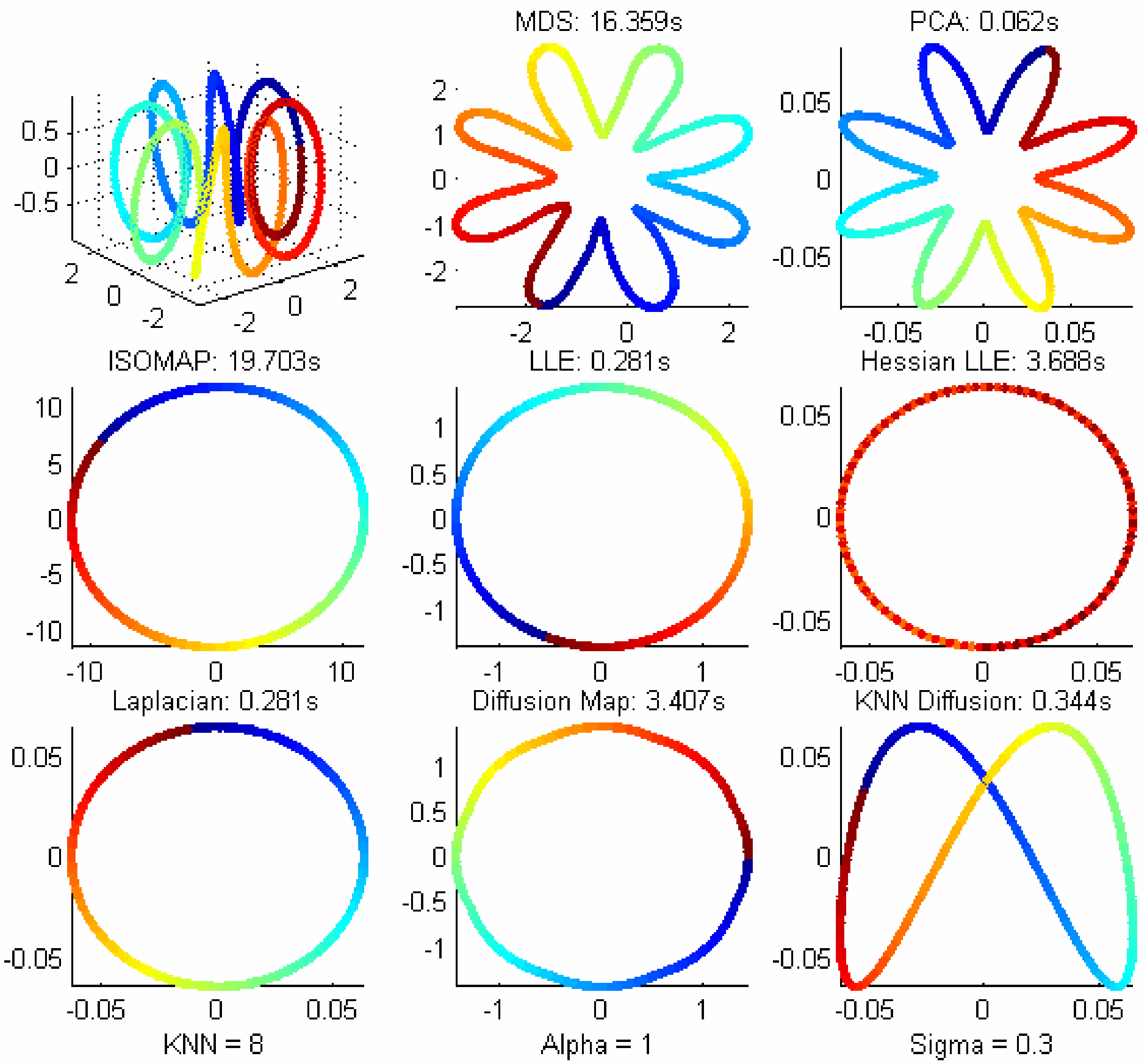
For M=8 clusters, MDS and PCA can still recover.
Diffusion Maps do quite well.
LLE and ISOMAP are decent, but Hessian and Laplacian fail.

Noise & Non-uniform Sampling

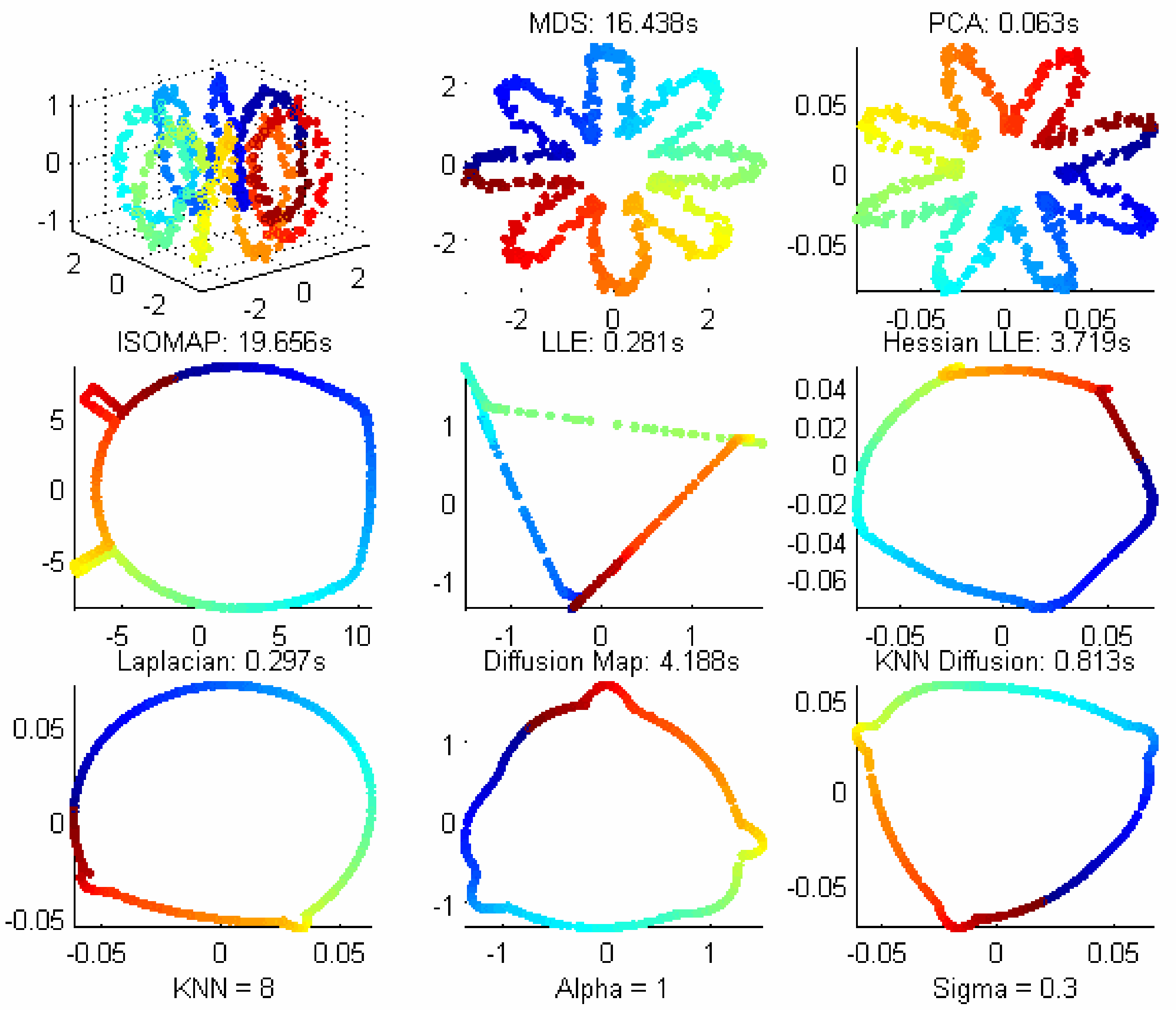
- Can the method handle changes from dense to sparse regions?
- Toroidal Helix should be unraveled into a circle parametrized by t.



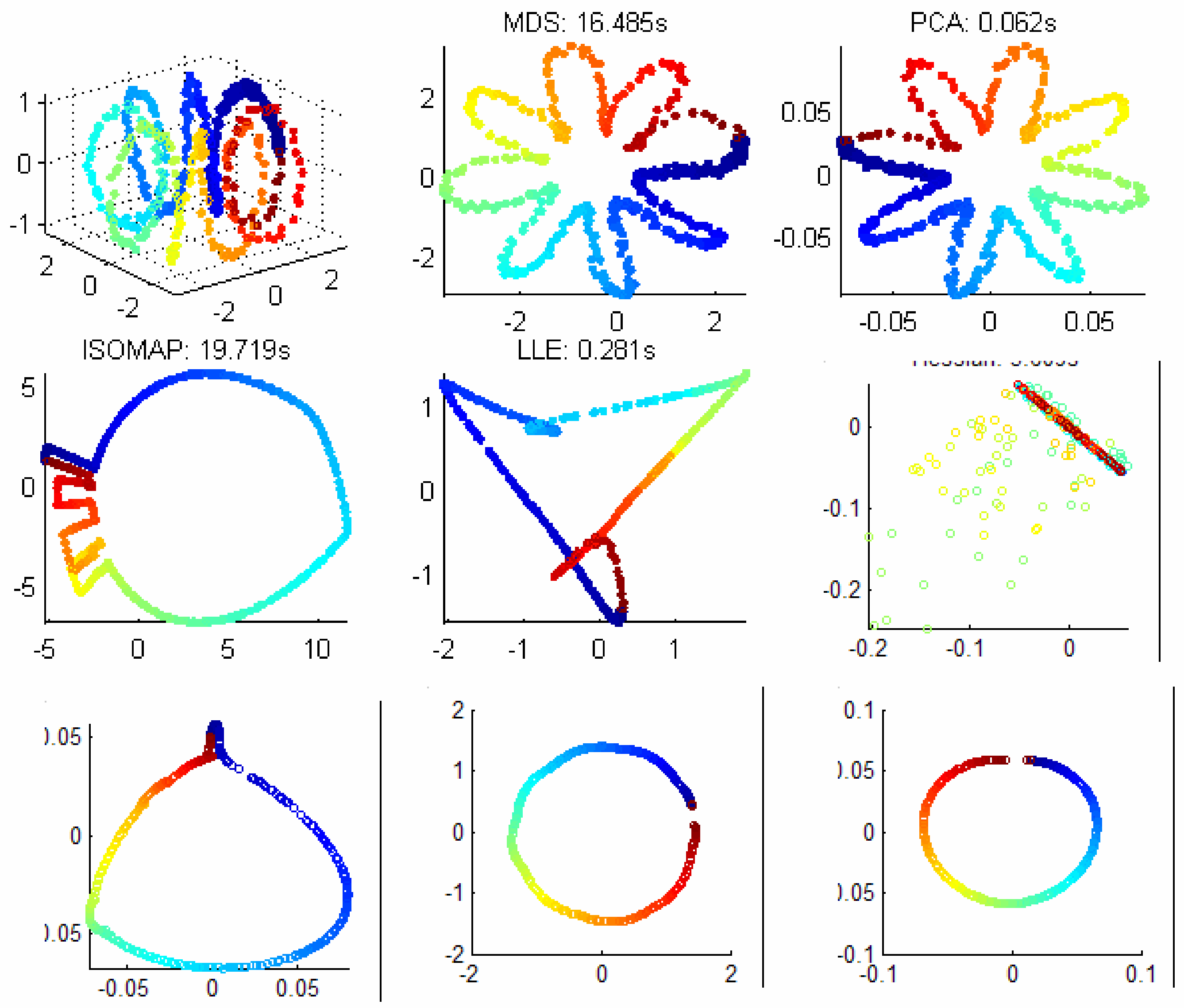
- We can change the sampling rate along the helix by changing the exponent R on the parameter t and we can add some noise.



With no noise added, ISOMAP, LLE, Laplacian, and Diffusion Map are correct.
MDS and PCA project to an asterisk.
What's up with Hessian and KNN Diffusion?



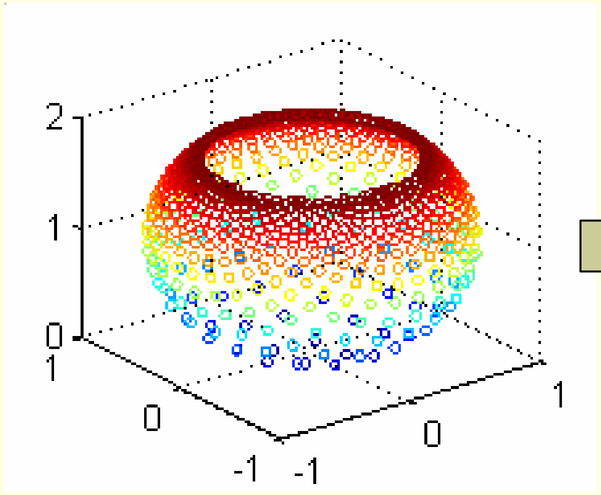
Adde noise to the Helix sampling.
LLE cannot recover the circle.
ISOMAP emphasizes outliers more than the other methods.

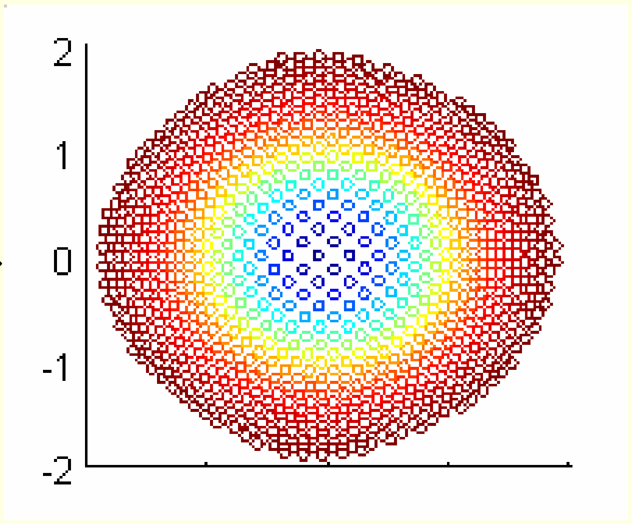


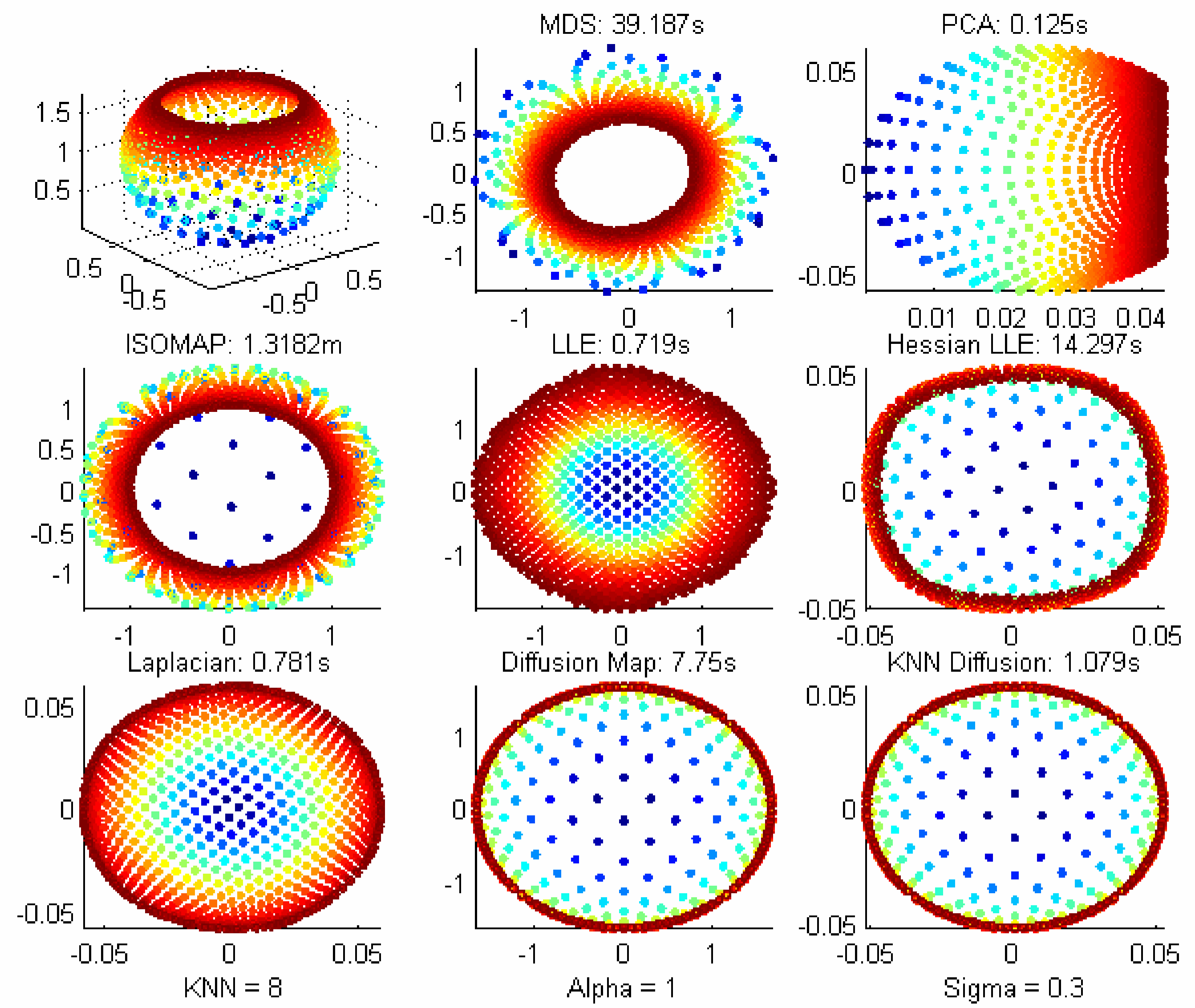
When the sampling rate is changed along the torus, Laplacian starts to mess up and Hessian is completely thrown off.
Hessian LLE code crashed frequently on this example.
Diffusion maps handle it quite well for carefully chosen Sigma=0.3.

Sparse Data & Non-uniform Sampling

- Of course, we want as much data as possible. But can the method handle sparse regions in the data?
- Punctured Sphere: the sampling is very sparse at the bottom and dense at the top.



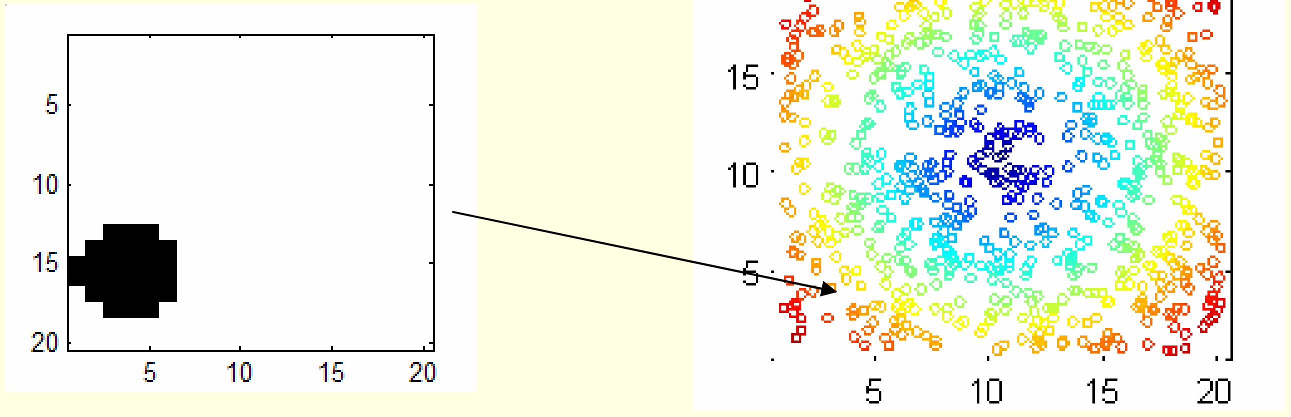


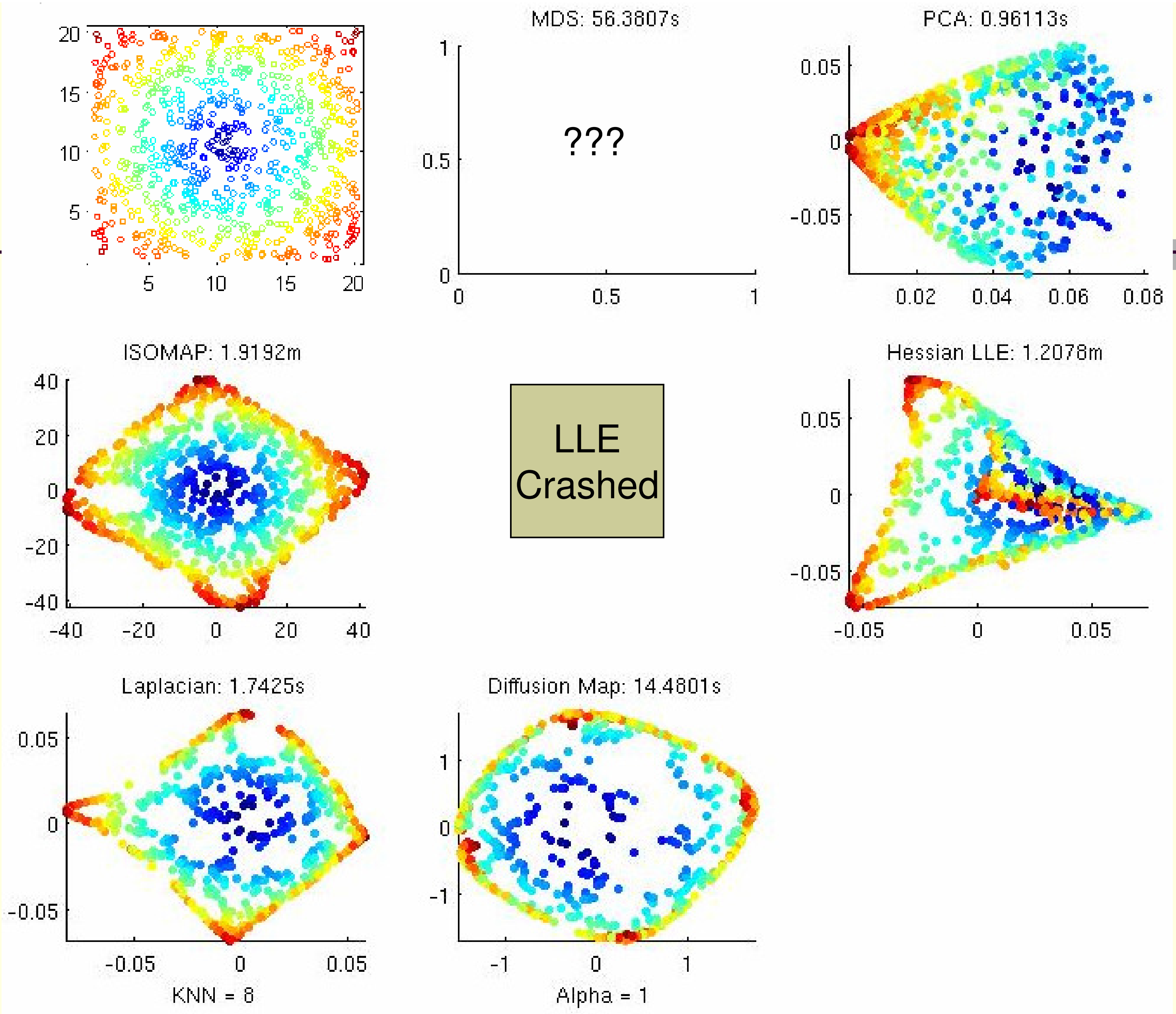


Only LLE and Laplacian get decent results.
PCA projects the sphere from the side. MDS turns it inside-out.
Hessian and Diffusion Maps get correct shape, but give too much emphasis to the sparse region at the bottom of the sphere.

High-Dimensional Data

- All of the examples so far have been 3D.
- But can the data handle high-dimensional data sets, like images?
- Disks: Create 20x20 images with a disk of fixed radius and random center.
- We should recover the centers of the circles.

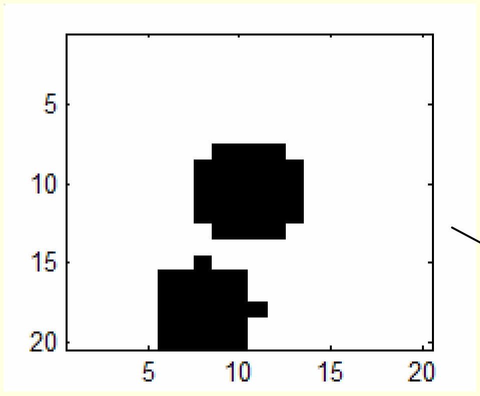


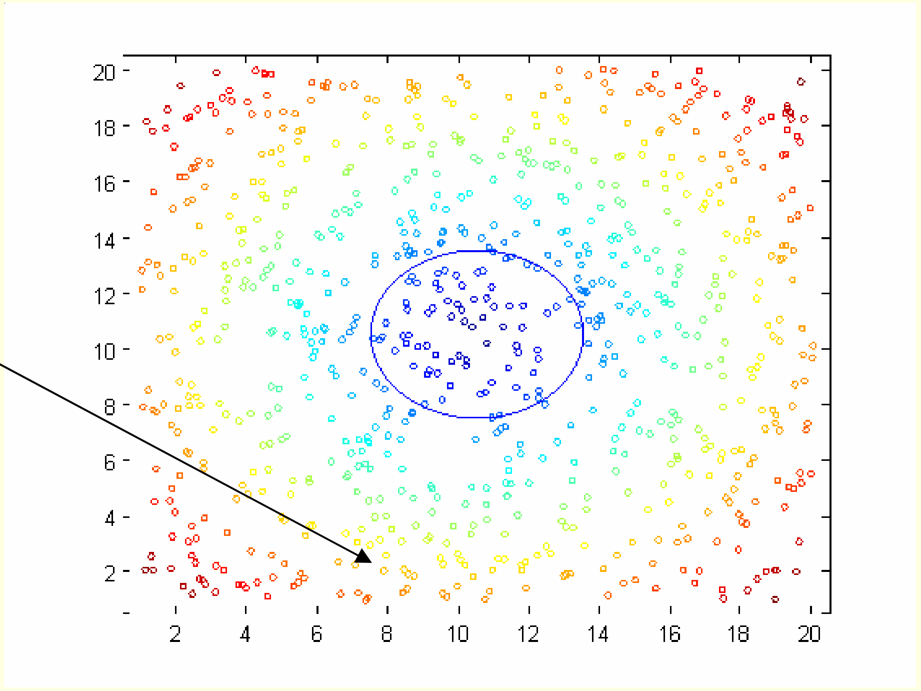


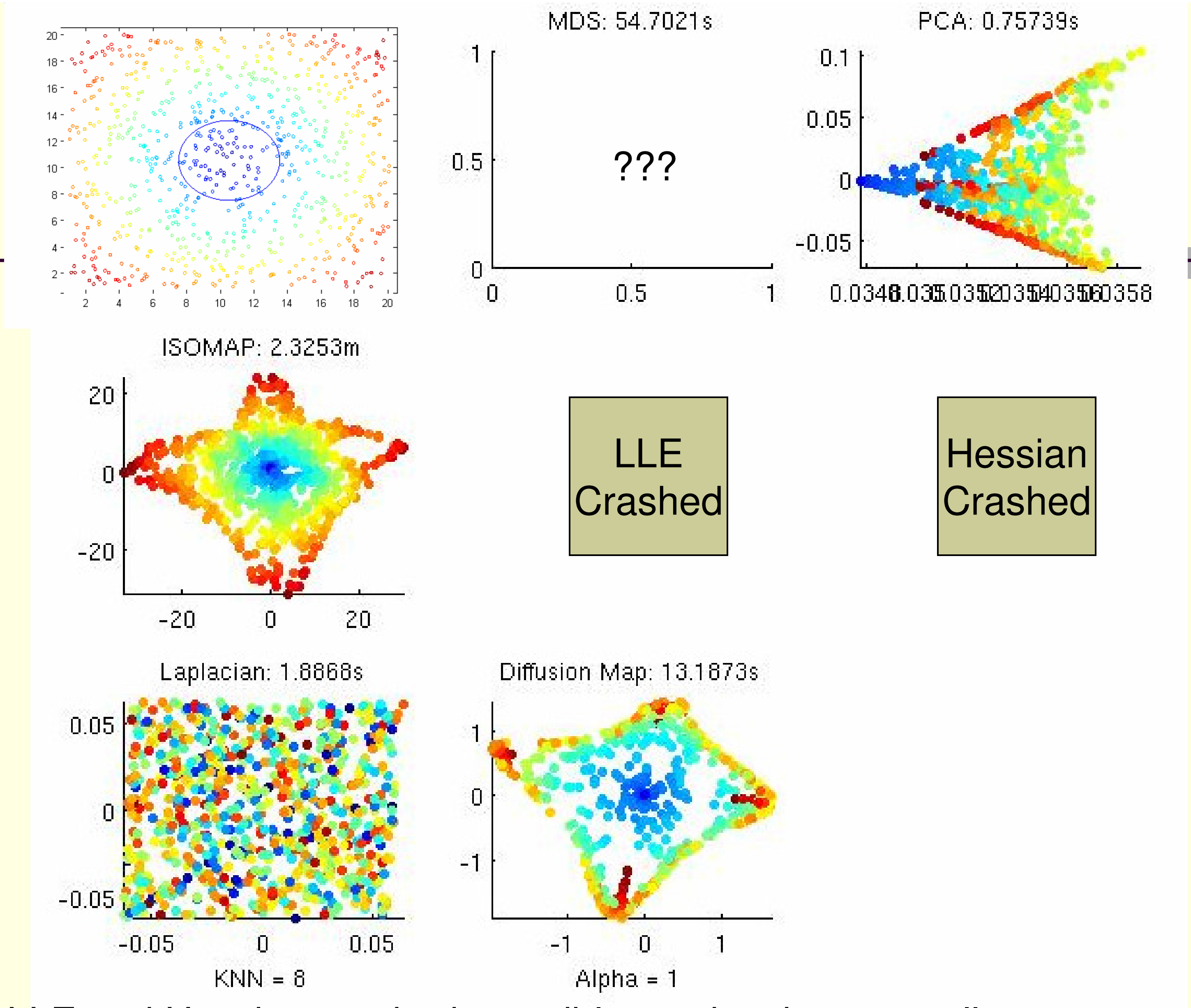
LLE crashed on high-dimensional data set.
Number of images was not high enough, but ISOMAP did a very good job.

Occluded Disks

- We can add a second disk of radius R in the center of every image.
- As Joe Kenney pointed out, this creates a non-convex data set by occlusion.







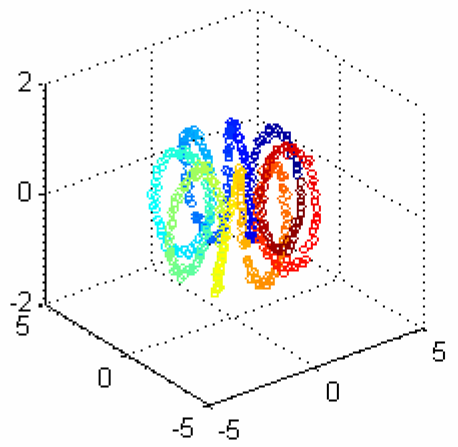
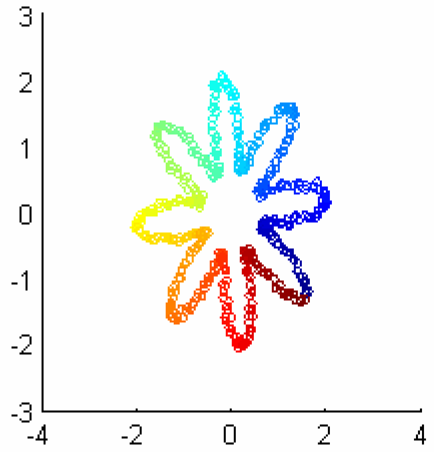
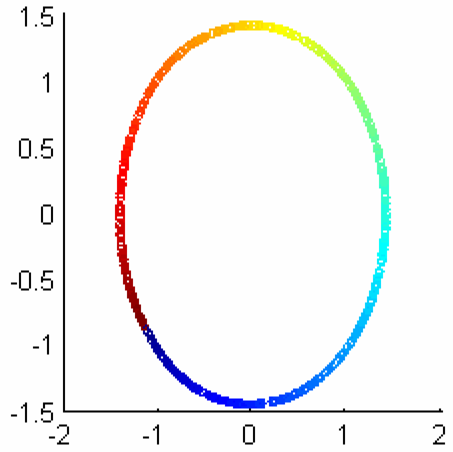
Both LLE and Hessian crashed, possibly # points is too small.
Laplacian failed completely.
Is ISOMAP the best for high-dimensional data?

Sensitivity to Parameters

- When the number of points is small or the data geometry is complex, it is important to set K appropriately, neither too big nor small.
- But if the data set is dense enough, we expect K around 8 or 10 to suffice.
- For Diffusion Maps, the method is very sensitive to the Sigma in Gaussian kernel. Varies from example to example.

Diffusion Map Sigma depends on manifold.

Helix

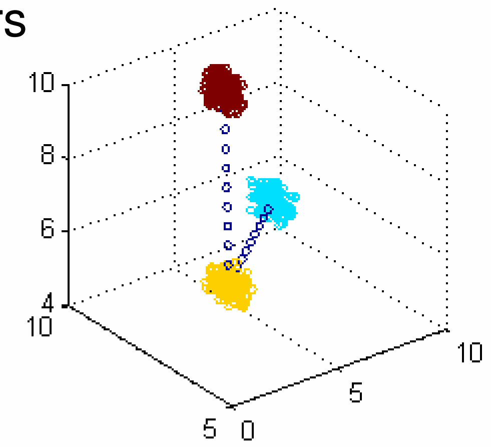
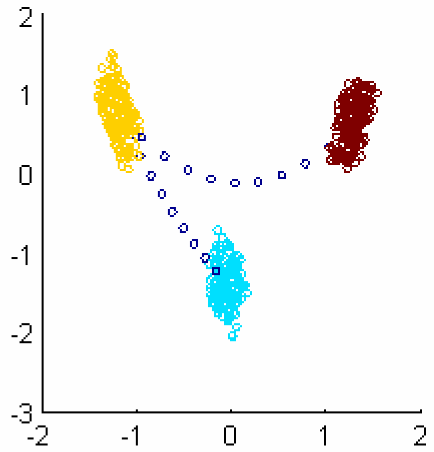
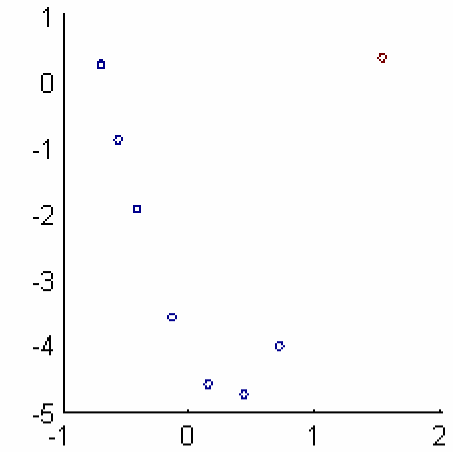




X

Sigma = 10

Sigma = 0.2

Clusters

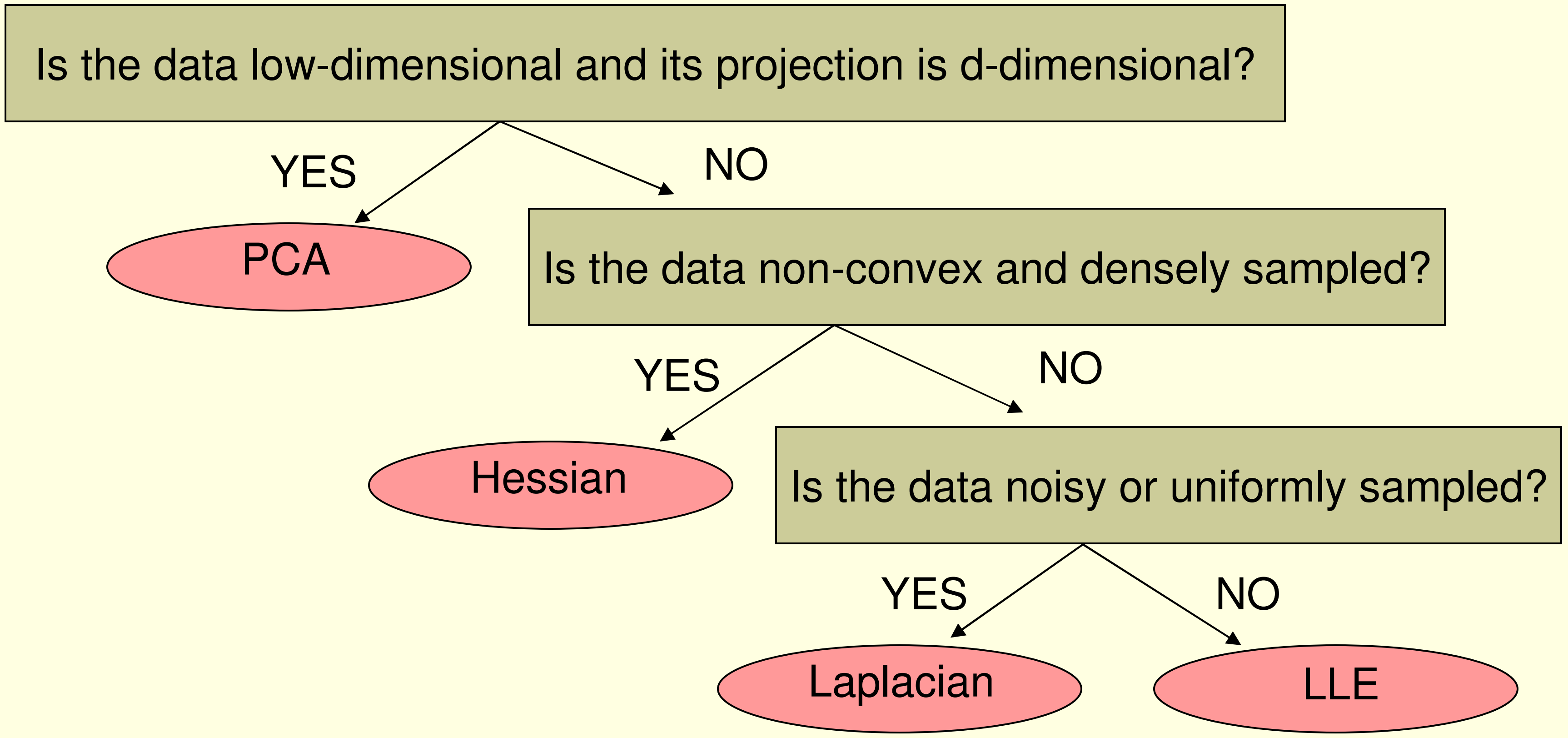




So what have you learned, Dorothy?

	MDS	PCA	ISOMAP	LLE	Hessian	Laplacian	Diffusion Map	KNN Diffusion
Speed	Very slow	Extremely fast	Extremely slow	Fast	Slow	Fast	Fast	Fast
Infers geometry?	NO	NO	YES	YES	YES	YES	MAYBE	MAYBE
Handles non-convex?	NO	NO	NO	MAYBE	YES	MAYBE	MAYBE	MAYBE
Handles non-uniform sampling?	YES	YES	YES	YES	MAYBE	NO	YES	YES
Handles curvature?	NO	NO	YES	MAYBE	YES	YES	YES	YES
Handles corners?	NO	NO	YES	YES	NO	YES	YES	YES
Clusters?	YES	YES	YES	YES	NO	NO	YES	YES
Handles noise?	YES	YES	MAYBE	NO	YES	YES	YES	YES
Handles sparsity?	YES	YES	YES	YES	NO *may crash*	YES	NO	NO
Sensitive to parameters?	NO	NO	YES	YES	YES	YES	VERY	VERY

So which is the best? Depends...

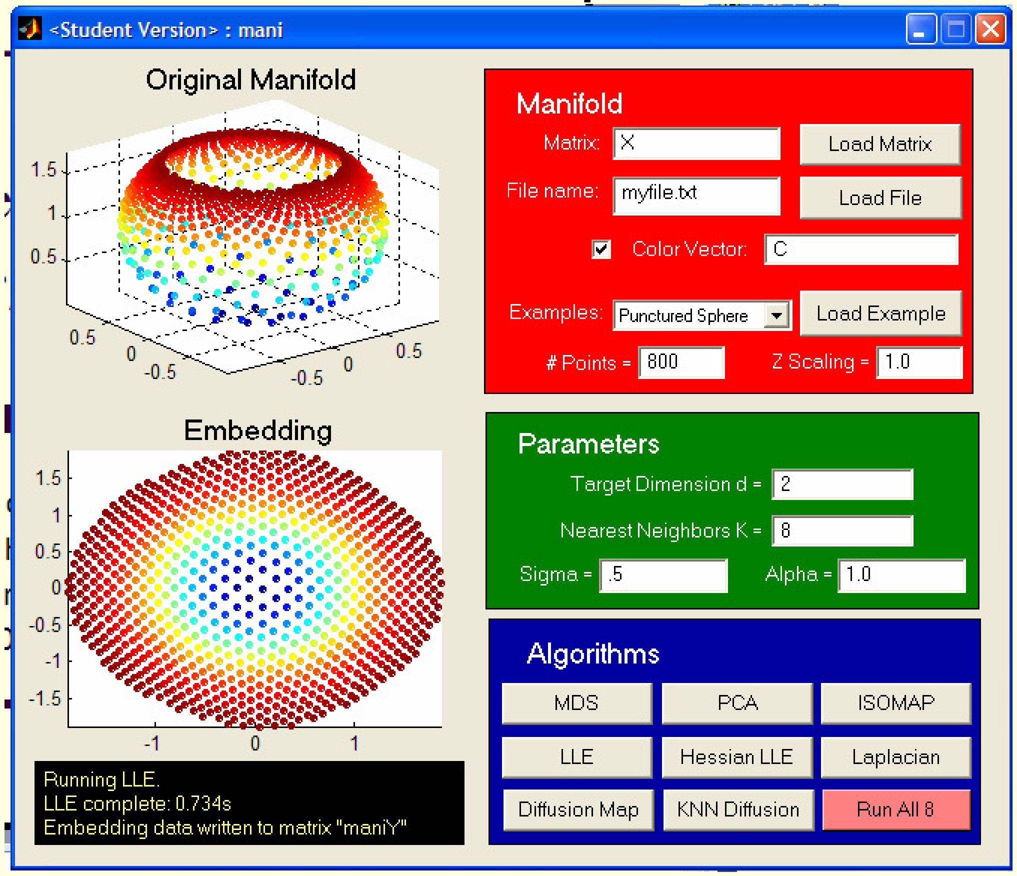
- ISOMAP is just too slow, so don't use it unless the data is high-dimensional or you've got a whole lot of time.
- It's too hard to set the parameters for Diffusion Maps.


Is the data low-dimensional and its projection is d-dimensional?
YES
NO
PCA
Is the data non-convex and densely sampled?
YES
NO
Hessian
Is the data noisy or uniformly sampled?
YES
NO
Laplacian
LLE

MANI

- MANIfold learning demonstration GUI
- Contains the methods and examples I've shown you.

Now for a demo!



Matrix: X

File name: myfile.txt

✓ Color Vector: C

Examples: Punctured Sphere

Points = 800 Z Scaling = 1.0

Target Dimension d = 2

Nearest Neighbors K = 8

Sigma = .5 Alpha = 1.0

```
Running LLE.
LLE complete: 0.734s
Embedding data written to matrix "maniY"
```

Some Notes on using MANI

- Hard to set K and Sigma just right.
- MDS and ISOMAP are very slow.
- Hessian LLE is pretty slow. Since Hessian needs a dense data set, this means it takes even longer when the # points is large.
- Occluded Disks is 400-dimensional data, which takes a long time and a lot of data points to correctly map.
- To run KNN Diffusion, requires unzipping Mauro’s EigenFunctions Package into the same directory as mani.m
- Matlab GUIs seem to run better on PC than Linux.

That's All Folks!

- The mani.m file will be e-mailed to you.
- If you try it out, please let me know what you think. Drop me an e-mail at:

  wittman@math.umn.edu

- *Thanks for listening!*