Eigenvectors, Heat Kernels, and Low Dimensional Representation of Data Sets

Thanks to:

Yale: R. Coifman, A. Osipov, V. Rokhlin

APPCOMSCI: D. Bassu, L. Ness

M. Maggioni, R.Schul, Amit Singer, J. Zhao

Topics

- Random Walks
- Transition Matrix
- Heat Kernels
- Representations of Data Sets
- Varadhan's Lemma
- Logarithms of Ratios of Heat Kernels
- Canonically Chosen Coordinates
- An example with Lidar

Background

1. Start with a data set embedded in $\mathbb{R}^d$,

   d >> 1. (For example ~ 1,000,000 pixel neighborhoods of size 9x9. Hyperspectral imaging: 20 wavelengths. Then 1 million points in d = 1620 = 20x81.)

2. Build the kernel $k = \exp\{ - |x_j - x_k|^2 / \sigma^2 \}$

3. Mission: Study the matrix $(k(x_j, x_k))$, adjusted for density.

4. Study Heat Kernel, eigenvectors.

Adjust the Matrix for Density

Define Density:

$W_{j,k} = k(x_j, x_k)$

$D_{j,j} = \sum_k W_{j,k}$ is a diagonal matrix

$L = Id - D^{-1/2} W D^{-1/2}$

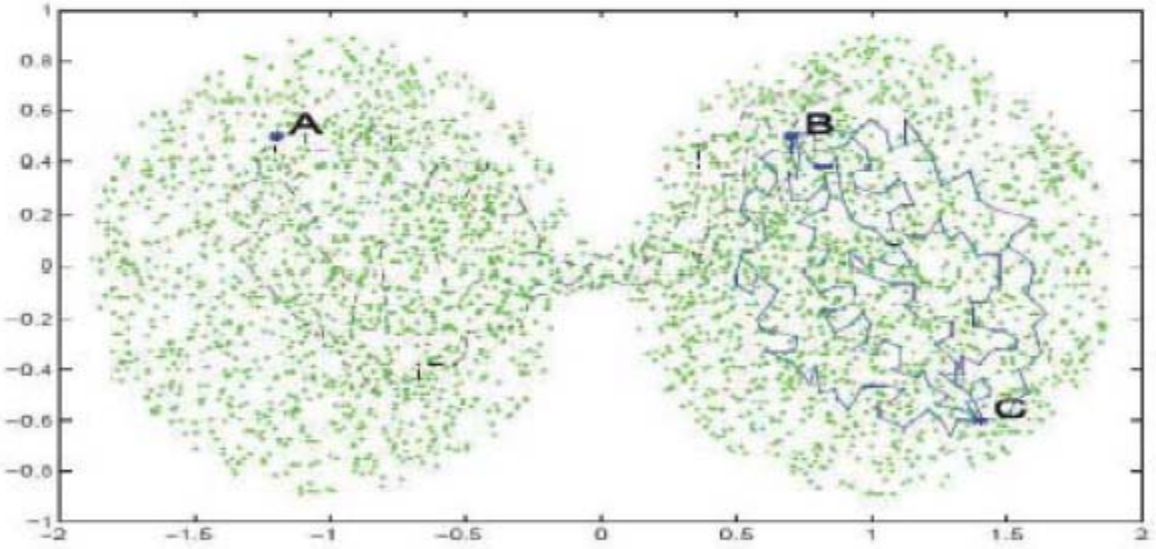
Then L is a Self-adjoint matrix whose eigenvectors are ϕ_k with eigenvalues λ_k. L is the normalized Laplacian.

Laplacian on Graphs

Given a weighted graph (G, W, E), the combinatorial Laplacian is defined by $L = D - W$, where $(D)_{ii} = \sum_j W_{ij}$, and the normalized Laplacian is defined by

$$\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}.$$

These are self-adjoint positive-semi-definite operators, let λ_i and ϕ_i be the eigenvalues and eigenvectors. Fourier analysis on graphs. The heat kernel is of course defined by $H_t = e^{-t\mathcal{L}}$; the natural random walk is $D^{-1}W$.



$d_{geod.}(A, B) \sim d_{geod.}(C, B)$, however $d^{(t)}(A, B) >> d^{(t)}(C, B)$.

Diffusion Geometry

The idea of using eigenfunctions as good local coordinate systems has appeared in the last ~ 15 years. Many authors in many communities. This goes under the general name of spectral geometry. One model is called Diffusion Geometry = special version of Δ on data sets.

Empirical observation: It often works in many situations where PCA does not, and is used for "dimensional reduction".

Question: Why does it "work"? (Indeed, why should it even "work" for manifolds?)

Let's first look at data sets.

Data Sets: Choose d Eigenvectors

$$\phi(x_j) = \{\phi i_1(x_j), \phi i_2(x_j), \ldots, \phi i_d(x_j)\}$$

This maps each point x_j to a point in $\mathbb{R}^d$. Note that the mapping is NONLINEAR. Many examples have shown that a proper selection of eigenvectors yields a good representation of the original data set. We will see examples, some of which have a clear choice of eigenvectors.

But why should this work? We will present a theorem in the continuous case that explains why one expects this. The upshot is that any method that uses spectral theory can be examined via the theorem.

A Theorem on Existence of good eigenfunction coordinates

1. **P. Jones, M. Maggioni, R. Schul, "Manifold parametrizations by eigenfunctions of the Laplacian and heat kernels". PNAS, vol. 105 no. 6 (2008), pages 1803-1808**

2. **P. Jones, M. Maggioni, R. Schul "Universal Local Parametrizations via Heat Kernels and Eigenfunctions of the Laplacian", To appear in Ann. Acad. Sci. Fennica.**

We now drop some subscripts.

Approximate Statement

Theorem (J,M,S) For a domain D or Manifold M of Volume = 1, the Laplace Eigenfunctions form a universal coordinate system on embedded balls.

d = dimension, B(x,r) an embedded ball

$\Rightarrow$ exist exactly d eigenf's $\{\phi_1, \phi_2, \ldots, \phi_d\}$, that BLOW UP B(x,$\epsilon$r) to AT LEAST Size 1.

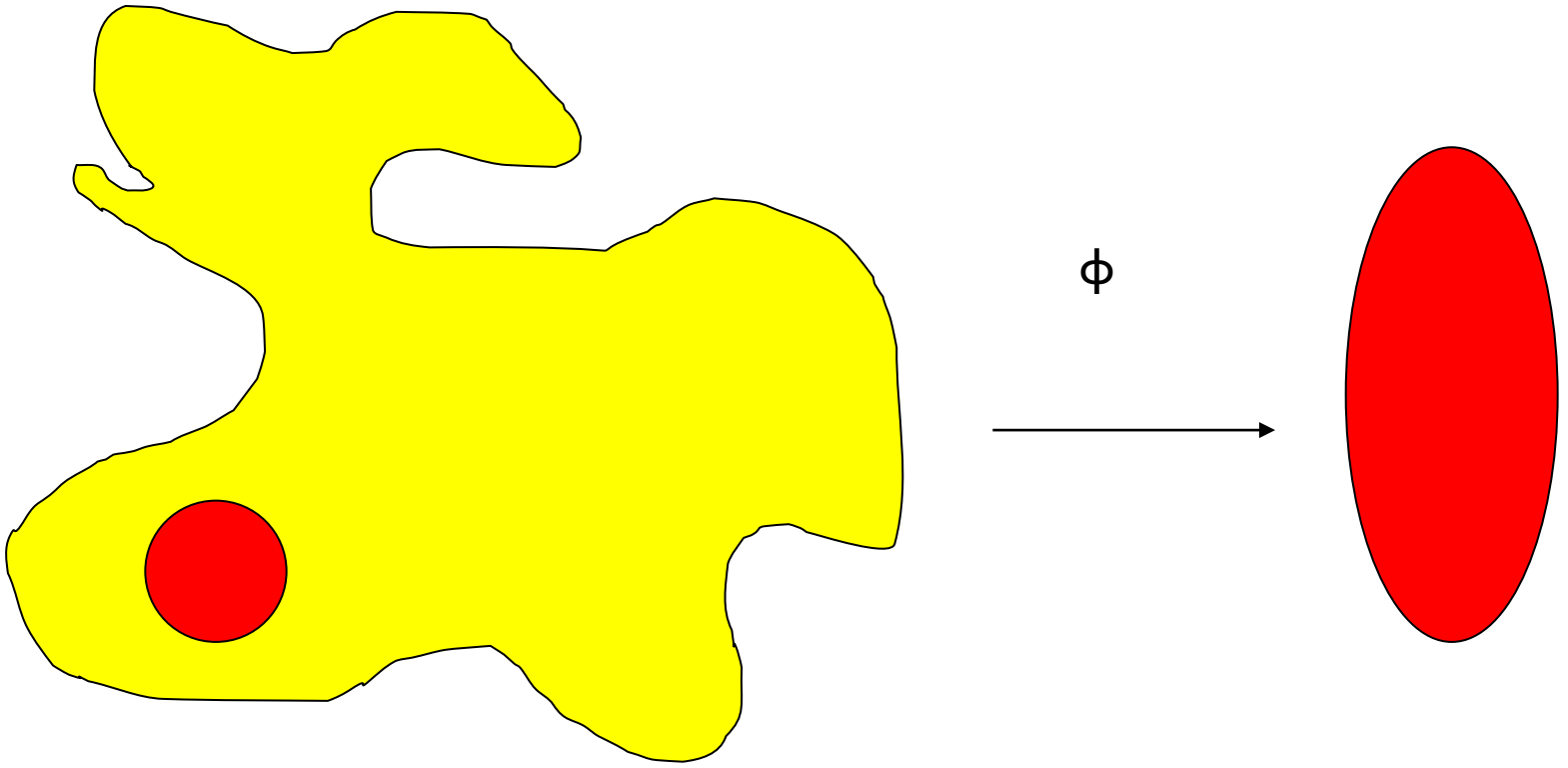
(and with LOW DISTORTION)

The eigenfunctions here are NOT the first d eigenfunctions (subscripts dropped).

Eigen Rule: $\lambda_k \sim r^{-2}$

In other words, the eigenfunctions "find the manifold", and the epsilon (in "B(x,ϵr)") is universal.

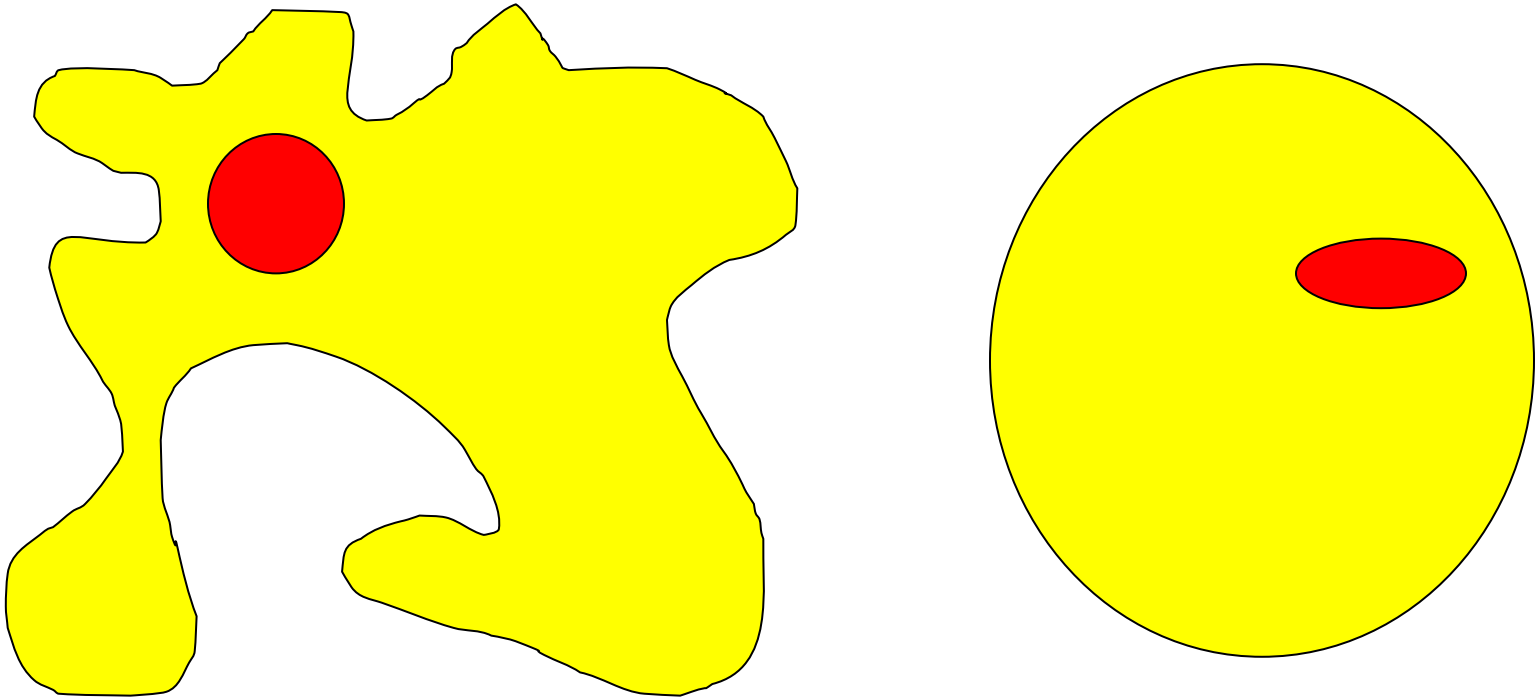
$x \to \{\phi_1(x), \phi_2(x)\} \ldots, \phi_d(x)\}$
on "1/2" of an embedded red ball

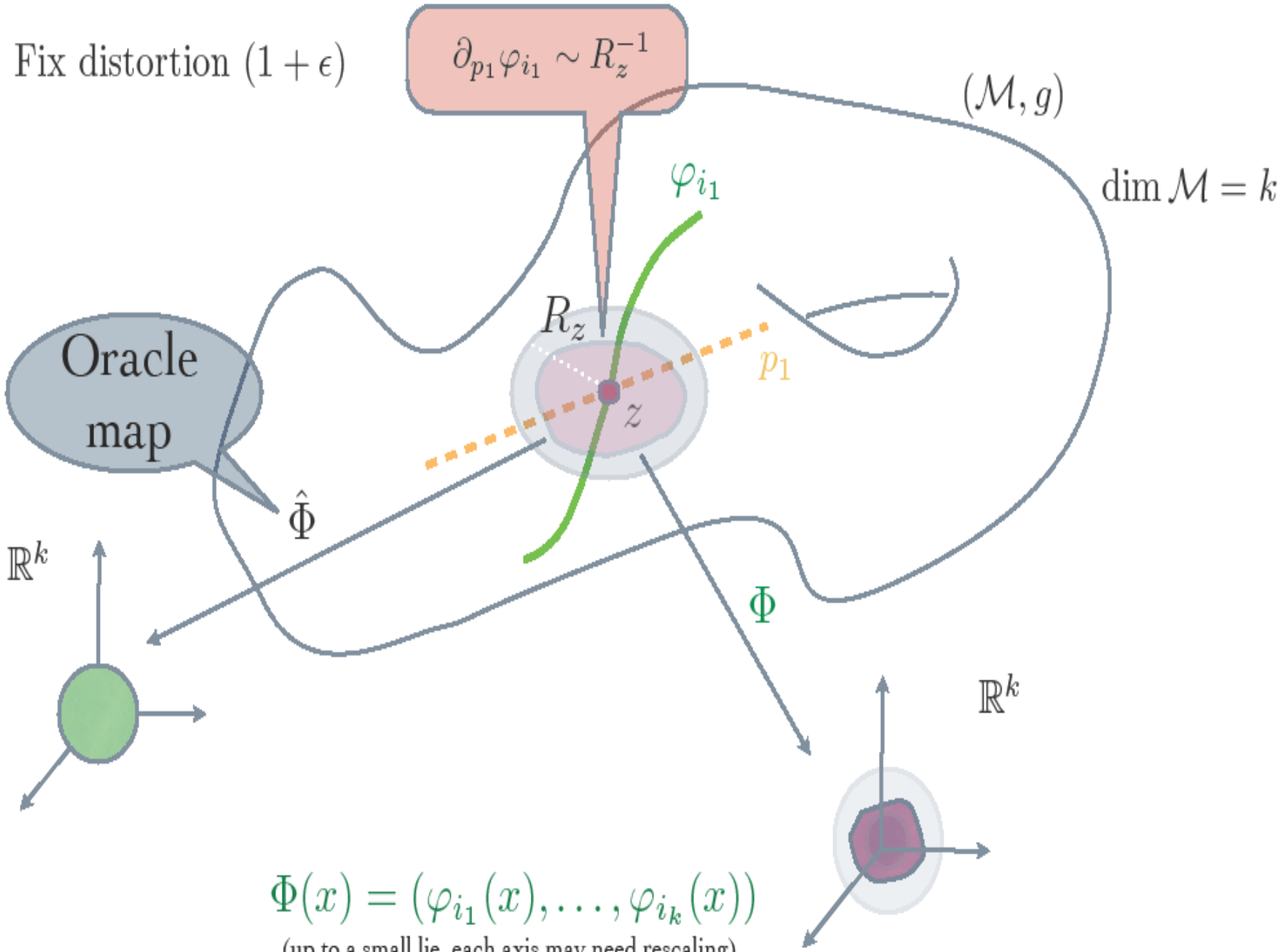


The ellipse has size at least 1. This has an analogy with the Riemann Mapping Theorems.

The Riemann Map F sends “good disks” to “almost disks”

F: Domain → Unit Disk




Fix distortion $(1+\epsilon)$
$\partial_{p_1}\varphi_{i_1} \sim R_z^{-1}$
$(\mathcal{M}, g)$
$\dim \mathcal{M} = k$
φ_{i_1}
R_z
p_1
z
Oracle map
$\hat{\Phi}$
Φ
$\mathbb{R}^k$
$\mathbb{R}^k$
$\Phi(x) = (\varphi_{i_1}(x), \ldots, \varphi_{i_k}(x))$
(up to a small lie, each axis may need rescaling)

Varadhan's Lemma: Heat Kernels

On $\mathbb{R}^d$:

$K(t,w,z) = K(t,w,z) = (4\pi t)^{-d/2} \exp\{ -|w-z|^2/4t\}$

Short Time behavior of the Heat Kernel(Varadhan)

Take limits as $t \to 0$:

$\text{Lim} \quad -4t\text{Log}(K(t,w,z)) = d(w,z)^2$

This is true on any smooth manifold.

Algebraic Corollary of Varadhan's Lemma:

$$-4t\log(K(x,y_1,t)/K(x,y_2,t))$$

$$\longrightarrow \quad d(x, y_1)^2 - d(x, y_2)^2$$

Call the limit $\Phi(x, Y)$ where Y corresponds to the ordered two points (y_1, y_2).

Idea: $\Phi(x, Y)$ corresponds to a "vector" that points in the "direction" $y_1 - y_2$.

We now use this to give a Prescription for a local coordinate system "anywhere". Example: $\mathbb{R}^d$.

Formula due to Formula (2008??)

In Euclidean Space this can be used to recover the usual coordinates. **There are no limits.** Here is how to recover the usual x,y coordinates in $\mathbb{R}^2$ for any point (x,y).

$Z_0 = (0,0)$, $Z_1 = (1,0)$. (Later use $Z_2 = (0,1)$). Let $t = 1/2$.

Y_1 is the (ordered) triple of points (Z_0 , Z_1 , Z_2).

$\Phi((x,y), Y_1) = \text{LOG}((K(t, (x,y), Z_1)/ K(t, (x,y), Z_0))$

Then $x = ½ + \Phi((x,y), Y_1)$ Our definition of Φ.

The same reasoning gives the y coordinate:

$y = ½ + \text{LOG}((K(t, (x,y), Z_2)/ K(t, (x,y), Z_0))$

Prescription for a Canonical"Varadhan-like" local coordinate system "anywhere".

We canonically choose d pairs of points $Y_j = (y_{j,1}, y_{j,2})$.

Define $\Phi_j(x) = \Phi(x, Y_j)$ for a suitable time t.

A "local coordinate chart" is then given by the vector:

$$V(x) = \{\Phi(x, Y_1) , \ldots\ldots, \Phi(x, Y_d)\}$$

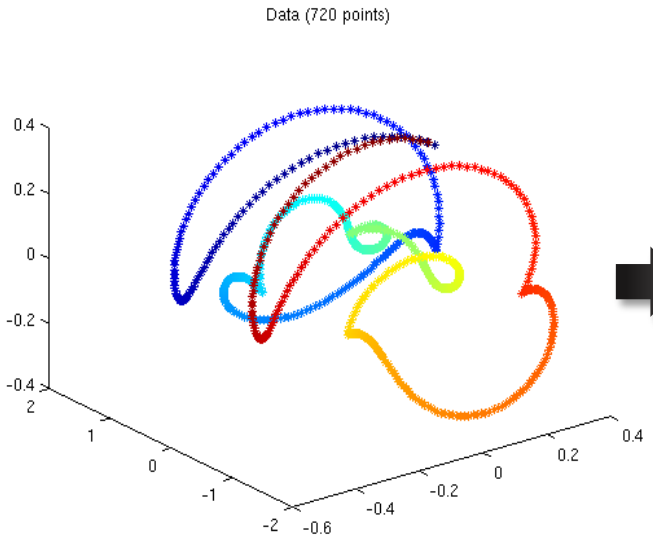
Note this is always globally defined and guaranteed to be "good" near some base point (if one chooses suitable pairs Y_j.) All we need here is some kind of heat kernel. (There is a version that is a theorem for manifolds: J,M,S.) But we can also do this for <u>any</u> data set.

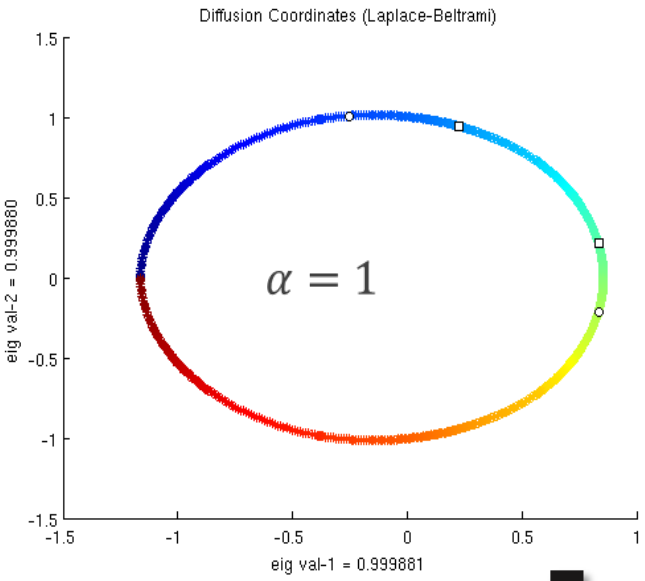
N.B. The heat kernel is much more stable that individual eigenfunctions (eigenvectors in numerical examples).

- Guaranteed robust coordinate system that looks like an Euclidean coordinate system a) locally on a manifold, b) globally on a flat manifold
  - Utilizes “heat triangulation” for construction of coordinates; much more stable than eigenfunction coordinates
- Works on hyperbolic spaces (Michal Tryniecki).
- Locally looks the same as diffusion maps
  - HKC: canonical way to pick coordinates in an automated fashion
  - Diffusion Maps: diffusion coordinates have to be picked manually

- References
  - “*Manifold parametrizations by eigenfunctions of the Laplacian and heat kernels*” by Peter W. Jones, Mauro Maggioni, Raanan Schul, PNAS vol. 105, no. 6, pp. 1803-1808, 2008.
  - Ph.D. Thesis by Michal Tryniecki, Yale University, 2013.

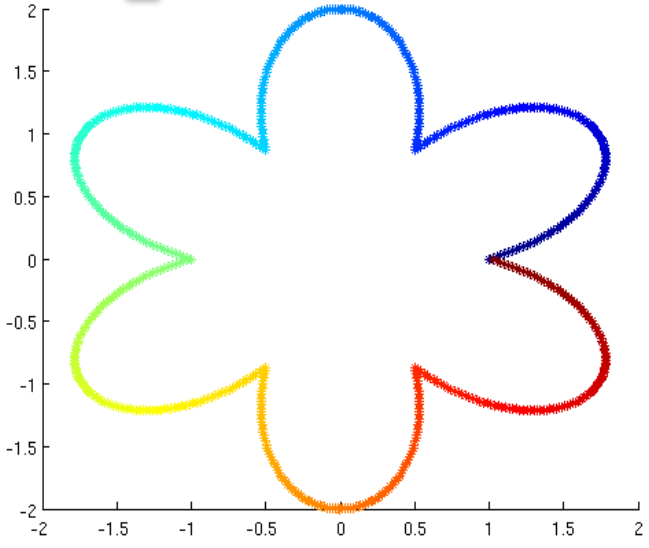
Example: r(θ) = 1 + ← |sin(3θ)| (Then "Swiss Roll")

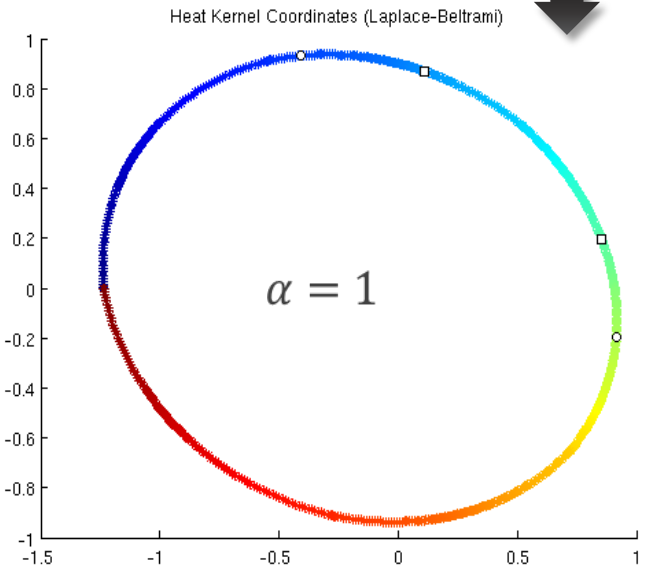




←

Diffusion Geometry





←

Heat Kernel Coordinates

The Algorithm

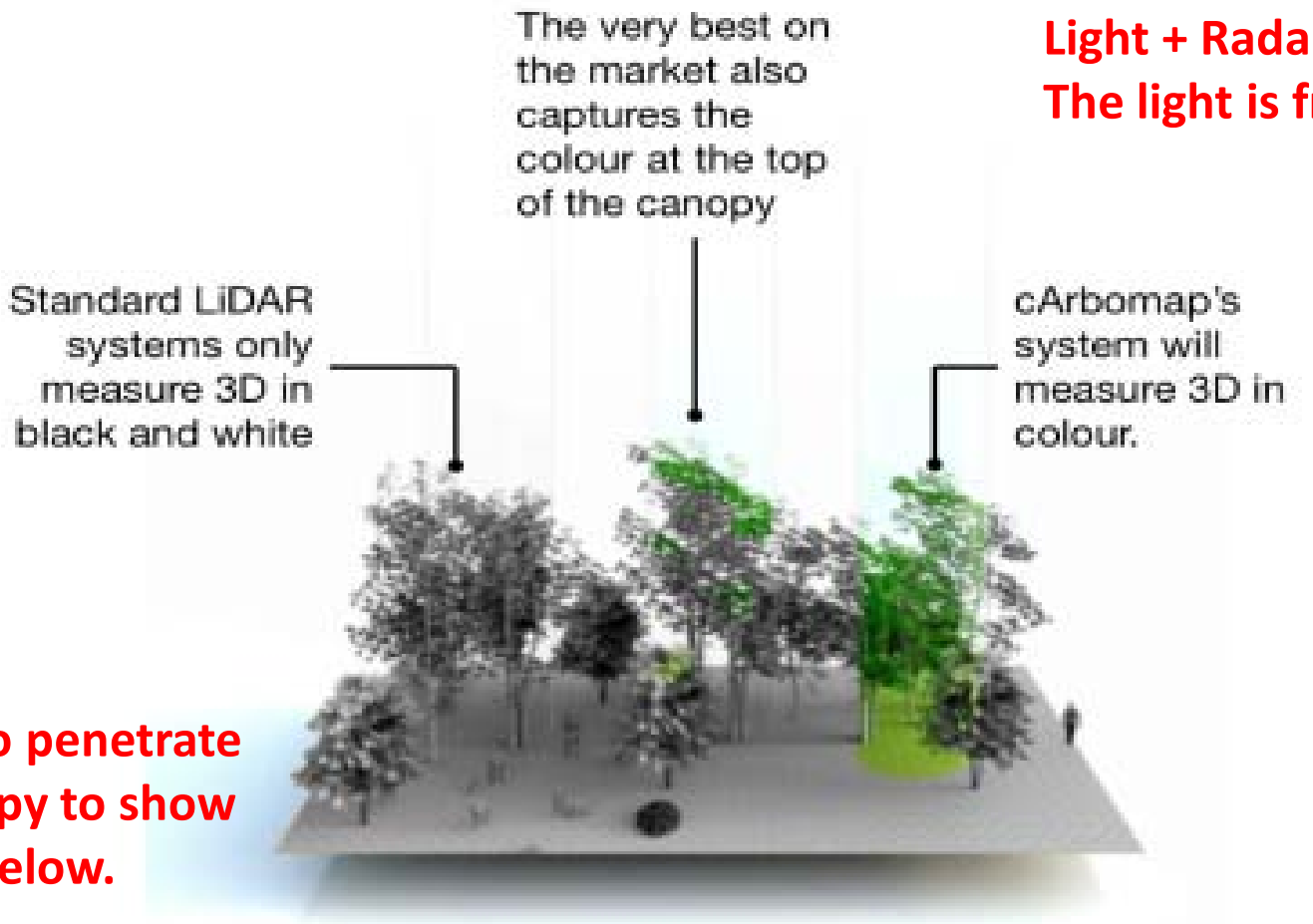
- HKC Algorithm
  1. Compute diffusion coordinates y_i in k dimensions for a given set of normalized points $x_i \in \mathbb{R}^d$; $i = 1,2,\dots,N$ at scale s
  2. Normalize $\{y_i\}$ to have zero mean and unit variance. Denote z_i as the normalized point corresponding to y_i
  3. For each heat kernel coordinate (l-th dimension), select a pair of points $z_{ndx(l,1)}$ and $z_{ndx(l,2)}$ from $\{z_i\}$ such that they are closest to the points $e_{l,1}$ and $e_{l,2}$ respectively
     - Here, $e_{l,1}$ and $e_{l,2}$ are both k dimensional vectors with all entries set to 1 expect for the l-th entry that is set to $-\frac{1}{4}$ or $\frac{1}{4}$ respectively
  4. For each x_i, compute the heat kernel coordinates $hkc(x_i)$ given by

$$hkc(x_i,t) = \left(\log\frac{p_t(x_i,x_{ndx(1,2)})}{p_t(x_i,x_{ndx(1,1)})},\dots,\log\frac{p_t(x_i,x_{ndx(k,2)})}{p_t(x_i,x_{ndx(k,1)})}\right)$$

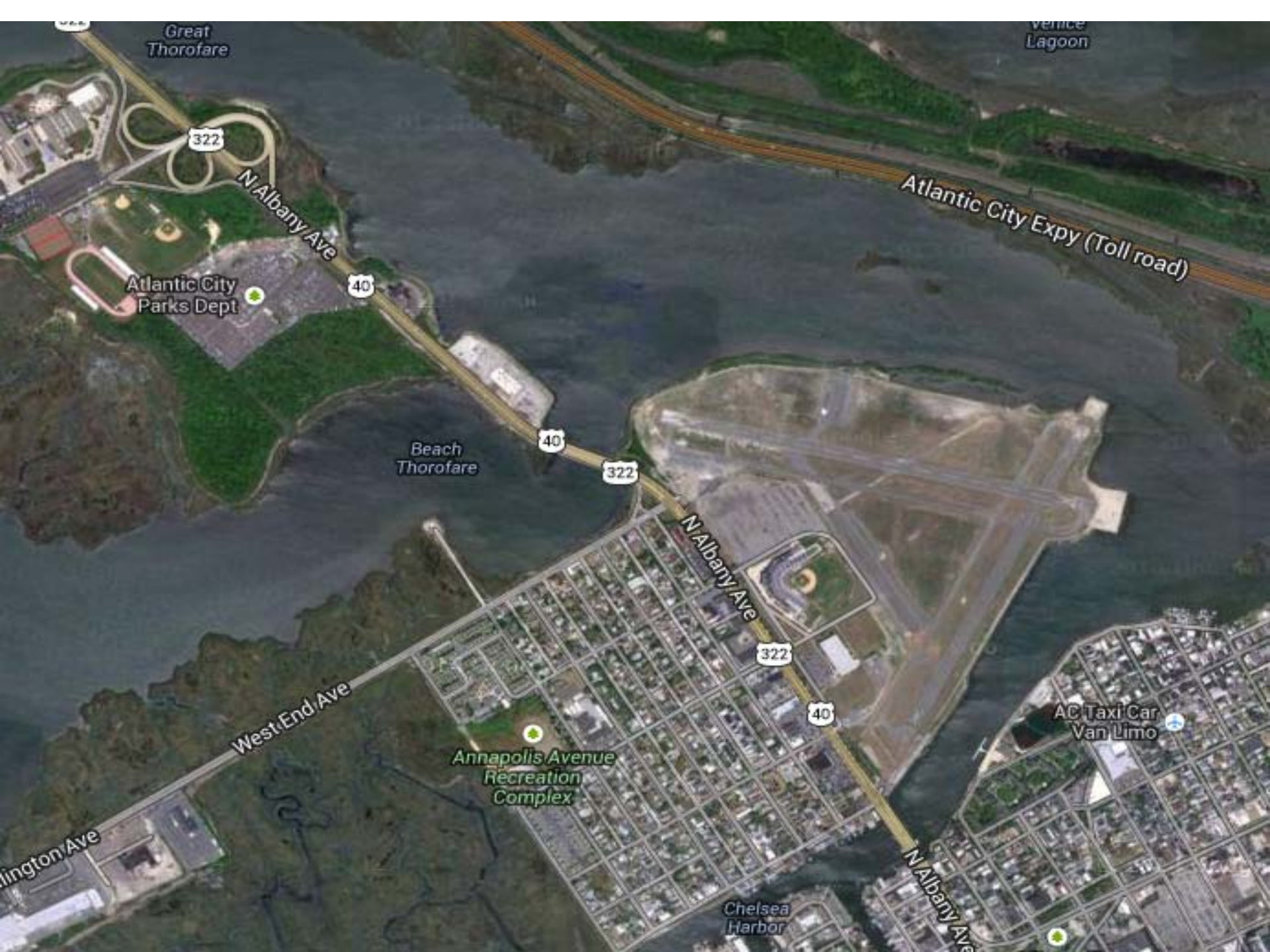
  5. Finally, normalize $\{hkc(x_i,t)\}$ for visualization and further analysis
- Cost(HKC) = Cost(diffusion coordinates) + $O(k^2 \cdot N)$

Multi-Spectral LIDAR and work with D. Bassu, L. Ness, and V. Rokhlin

Portmanteau of
Light + Radar
The light is from a Laser



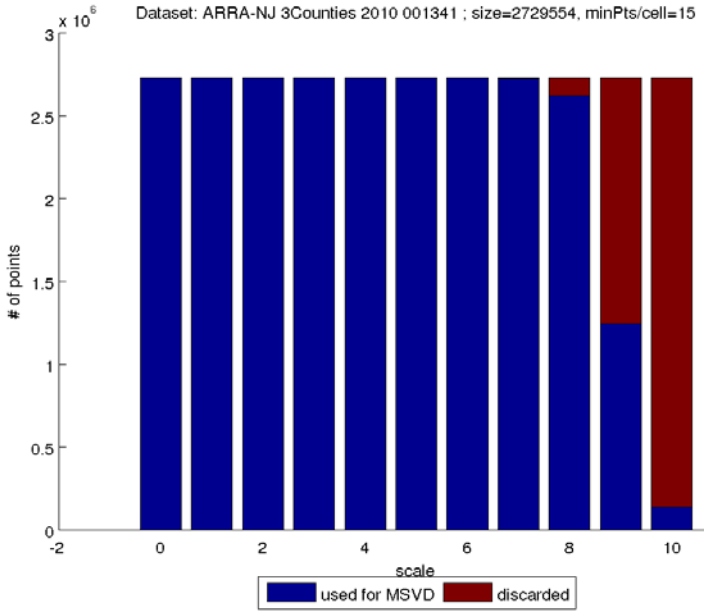
This can also penetrate Forest Canopy to show structures below.

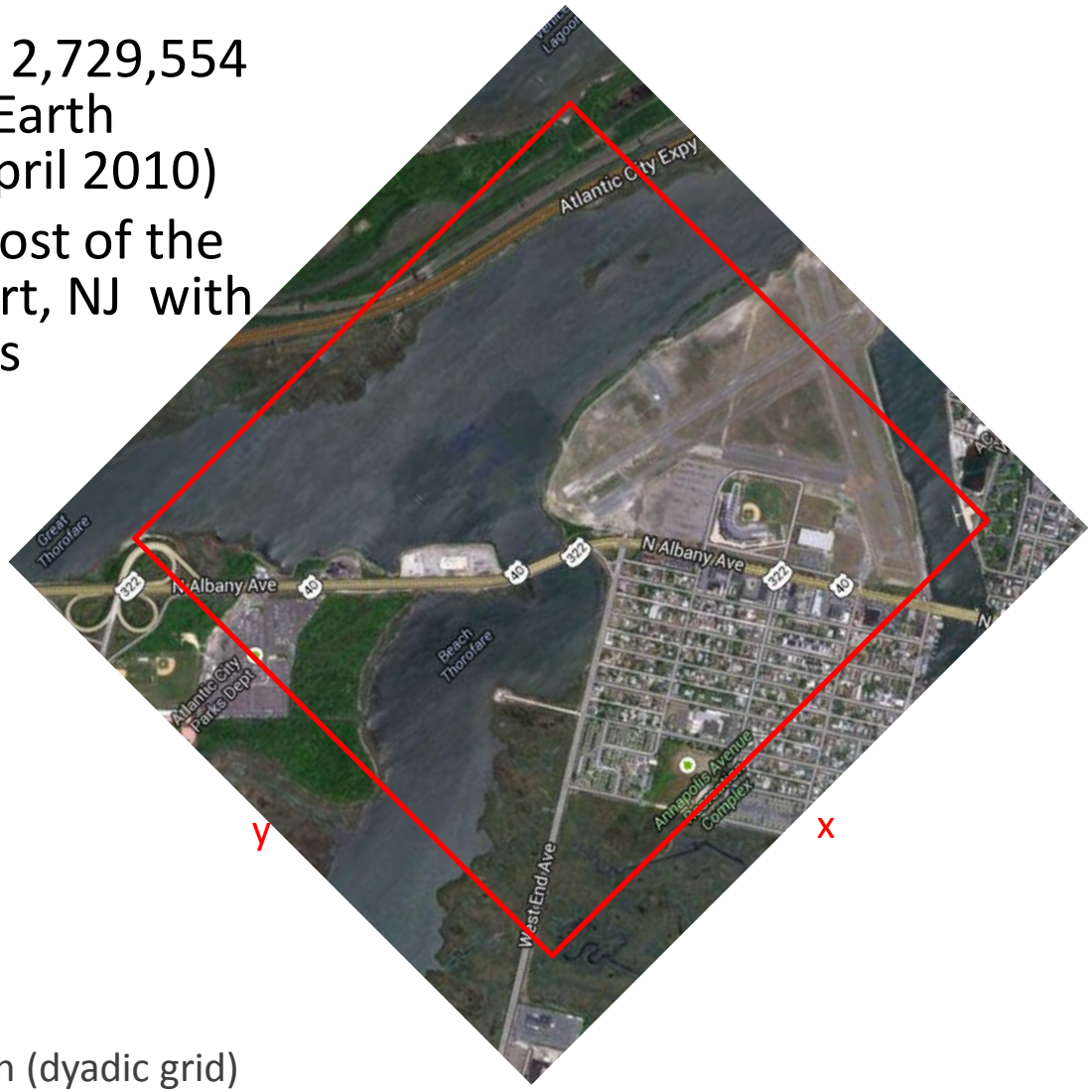

Great Thorofare
Venice Lagoon
322
N Albany Ave
Atlantic City Expy (Toll road)
Atlantic City Parks Dept
40
Beach Thorofare
322
N Albany Ave
322
40
West End Ave
AC Taxi Car Van Limo
Annapolis Avenue Recreation Complex
Chelsea Harbor
N Albany Ave

Ex.1: Former Atlantic City International Airport, NJ

An Example by D. Bassu

- LIDAR dataset comprising 2,729,554 points provided by USGS Earth Explorer tool (collected April 2010)
- Square region covering most of the former Atlantic City Airport, NJ with side length ~ 5,000 meters

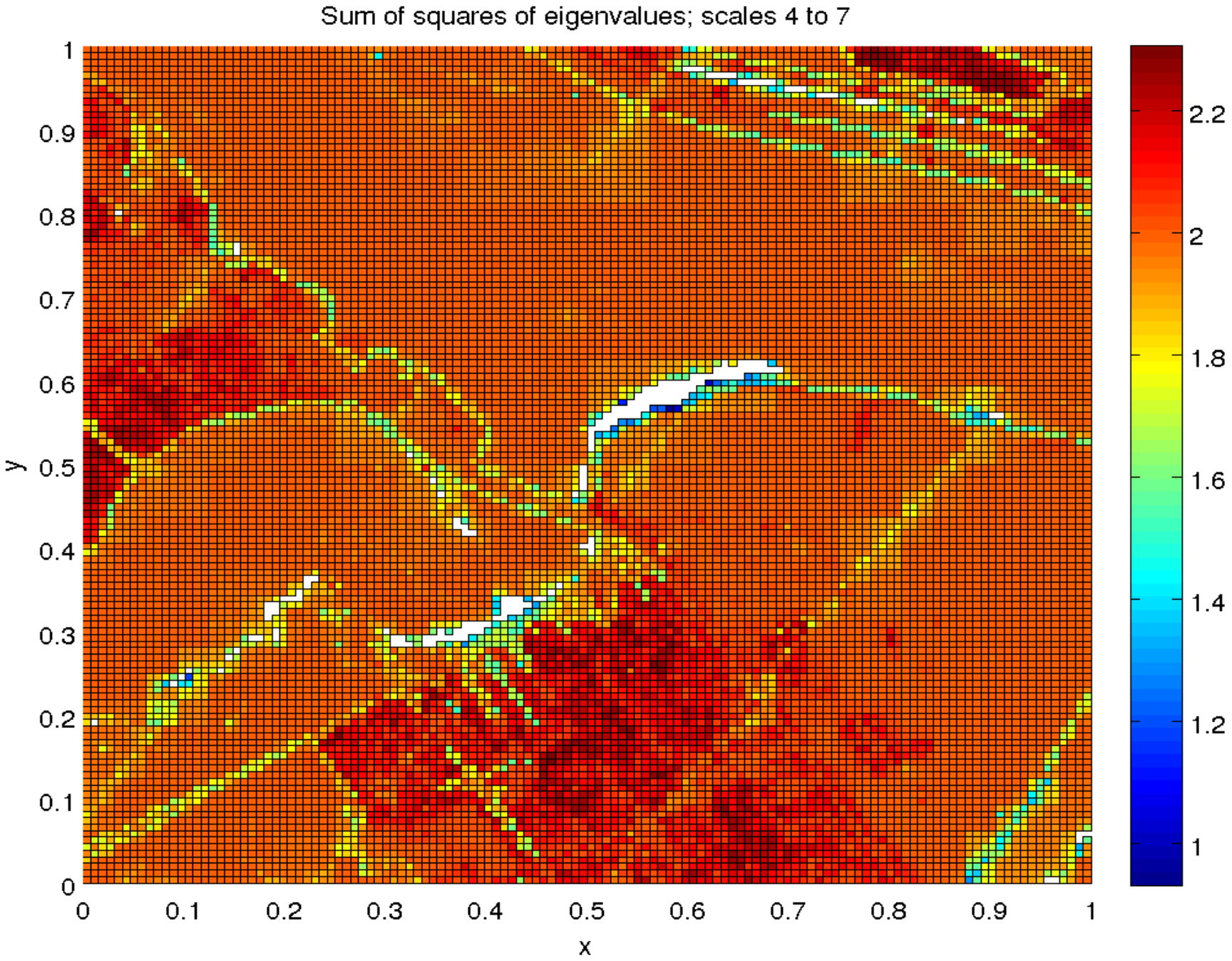


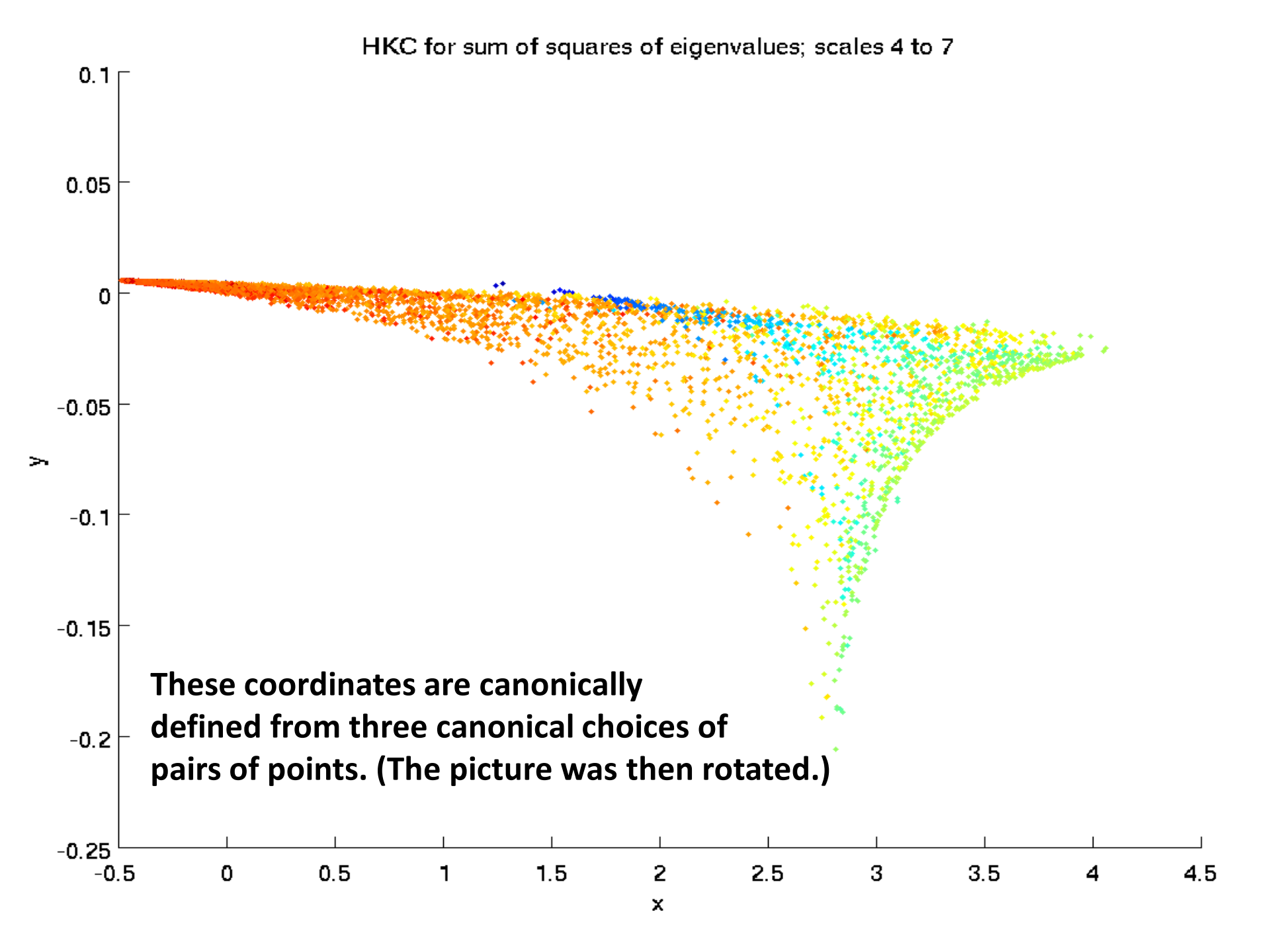


Points get isolated beyond 2^{-7} resolution (dyadic grid)

Multiscale SVD

On the Lidar data (which does not give us a 3D data set!) we compute SVD on four scales (for each point). For each scale we keep the top three (normalized!) Eigenvalues and we square . This gives us 4x3 = 12 numbers for each point, which we add (Call this number V.). For each planar point (x,y) we get a 3D vector: (x,y, V(x,y)). Now make canonical heat coordinates.


Sum of squares of eigenvalues; scales 4 to 7
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1
1.2
1.4
1.6
1.8
2
2.2


HKC for sum of squares of eigenvalues; scales 4 to 7
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
y
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
x
These coordinates are canonically
defined from three canonical choices of
pairs of points. (The picture was then rotated.)

Numerical Issues for Large Data sets in High Dimensions

- (Andre Osipov's Talk: Joint work with PJ and V. Rokhlin) In order to begin setting up the transition matrix (RW), one needs to know how to efficiently find approximate nearest neighbors for each data point.
- Then one needs to find the "heat kernel" and find certain eigenvectors. (Vladimir Rokhlin)
- In our numerical example today, one needs fast algorithms for multiscale SVD. (Vladimir Rokhlin)