Data Analysis and Manifold Learning
Lecture 9: Diffusion on Manifolds and on Graphs

Radu Horaud
INRIA Grenoble Rhone-Alpes, France
Radu.Horaud@inrialpes.fr
http://perception.inrialpes.fr/

Outline of Lecture 9

- Kernels on graphs
- The exponential diffusion kernel
- The heat kernel of discrete manifolds
- Properties of the heat kernel
- The auto-diffusion function
- Shape matching

Material for this lecture

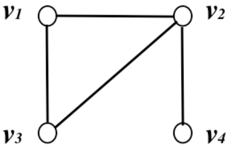
- P. Bérard, G. Besson, & G. Gallot. Embedding Riemannian manifolds by their heat kernel. Geometric and Functional Analysis (1994): A rather theoretical paper.
- J. Shawe-Taylor & N. Cristianini. Kernel Methods in Pattern Analysis (chapter 10): A mild introduction to graph kernels.
- R. Kondor and J.-P. Vert: Diffusion Kernels in "Kernel Methods in Computational Biology" ed. B. Scholkopf, K. Tsuda and J.-P. Vert, (The MIT Press, 2004): An interesting paper to read.
- J. Sun, M. Ovsjanikov, & L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. Symposium on Geometric Processing (2009).
- A. Sharma & R. Horaud. Shape matching based on diffusion embedding and on mutual isometric consistency. NORDIA workshop (2010).

The Adjacency Matrix of a Graph (from Lecture #3)

- For a graph with n vertices and with binary edges, the entries of the $n \times n$ adjacency matrix are defined by:

$$\mathbf{A} := \begin{cases} A_{ij} = 1 & \text{if there is an edge } e_{ij} \\ A_{ij} = 0 & \text{if there is no edge} \\ A_{ii} = 0 & \end{cases}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



The Walks of Length 2

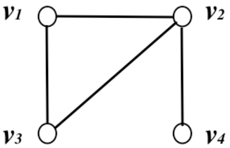
- The number of walks of length 2 between two graph vertices:

$$N_2(v_i, v_j) = A^2(i, j)$$

- Examples:

$$N_2(v_2, v_3) = 1; \;\; N_2(v_2, v_2) = 3$$

$$\mathbf{A}^2 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 3 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$



The Number of Walks of Length k

Theorem

The number of walks of length k joining any two nodes v_i and v_j of a binary graph is given by the (i, j) entry of the matrix $\mathbf{A}^k$:

$$N_k(v_i, v_j) = \mathbf{A}^k(i, j)$$

- A proof of this theorem is provided in:
  Godsil and Royle. (2001). Algebraic Graph Theory. Springer.

The Similarity Matrix of an Undirected Weighted Graph

- We consider *undirected weighted graphs*; Each edge e_{ij} is weighted by $\omega_{ij} > 0$. We obtain:

$$\mathbf{\Omega} := \begin{cases} \Omega(i,j) = \omega(v_i, v_j) = \omega_{ij} & \text{if there is an edge } e_{ij} \\ \Omega(i,j) = 0 & \text{if there is no edge} \\ \Omega(i,i) = 0 & \end{cases}$$

- We will refer to this matrix as the *base* similarity matrix

Walks of Length 2

- The weight of the walk from v_i to v_j through v_l is defined as the *product* of the corresponding edge weights:

$$\omega(v_i, v_l)\omega(v_l, v_j)$$

- The sum of weigths of all walks of length 2:

$$\omega_2(v_i, v_j) = \sum_{l=1}^{n} \omega(v_i, v_l)\omega(v_l, v_j) = \mathbf{\Omega}^2(i, j)$$

Associated Feature Space

- Let $\mathbf{\Omega} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, i.e., the spectral decomposition of a real symmetric matrix.
- We have $\mathbf{\Omega}^2 = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^\top$ is symmetric semi-definite *positive*, hence it can be interpreted as a kernel matrix, with:

$$\begin{aligned} \omega_2(v_i, v_j) &= \sum_{l=1}^{n} \omega(v_i, v_l)\omega(v_l, v_j) \\ &= \langle (\omega(v_i, v_l))_{l=1}^{n}, (\omega(v_j, v_l))_{l=1}^{n} \rangle = \kappa(v_i, v_j) \end{aligned}$$

- Associated feature space:

$$\phi : v_i \rightarrow \boldsymbol{\phi}(v_i) = (\omega(v_i, v_1) \dots \omega(v_i, v_l) \dots \omega(v_i, v_n))^\top$$

- It is possible to "enhance" the base similarity matrix by linking vertices along walks of length 2.

Combining Powers of the Base Similarity Matrix

- More generally one can take powers of the base similarity matrix as well as linear combinations of these matrices:

$$\alpha_1 \mathbf{\Omega} + \ldots + \alpha_k \mathbf{\Omega}^k + \ldots + \alpha_K \mathbf{\Omega}^K$$

- The eigenvalues of such a matrix must be nonnegative to satisfy the kernel condition:

$$\alpha_1 \mathbf{\Lambda} + \ldots + \alpha_k \mathbf{\Lambda}^k + \ldots + \alpha_K \mathbf{\Lambda}^K \succeq 0$$

The Exponential Diffusion Kernel

- Consider the following combination of powers of the base similarity matrix:

$$\mathbf{K} = \sum_{k=0}^{\infty} \frac{\mu^k}{k!} \mathbf{\Omega}^k$$

- This corresponds to:

$$\mathbf{K} = e^{\mu \mathbf{\Omega}}$$

- where μ is a *decay factor* chosen such that the influence of longer walks decreases, since they are less reliable.
- Spectral decomposition:

$$\mathbf{K} = \mathbf{U} e^{\mu \mathbf{\Lambda}} \mathbf{U}^\top$$

- Hence $\mathbf{K}$ is a kernel matrix since its eigenvalues $e^{\mu \lambda_i} \geq 0$.

Heat Diffusion on a Graph

- The heat-diffusion equation on a Riemannian manifold: $\left(\frac{\partial}{\partial t} + \Delta_{\mathcal{M}}\right) f(x;t) = 0$
- $\Delta_{\mathcal{M}}$ denotes the geometric Laplace-Beltrami operator.
- $f(x;t)$ is the distribution of heat over the manifold at time t.
- By extension, $\frac{\partial}{\partial t} + \Delta_{\mathcal{M}}$ can be referred to as the *heat operator* [Bérard et al. 1994].
- This equation can also be written on a graph:

$$\left(\frac{\partial}{\partial t} + \mathbf{L}\right) \boldsymbol{F}(t) = 0$$

where the vector $\boldsymbol{F}(t) = (F_1(t) \dots F_n(t))^\top$ is indexed by the nodes of the graph.

The Fundamental Solution

- The fundamental solution of *the (heat)-diffusion equation on Riemannian manifolds* holds in the discrete case, i.e., for undirected weighted graphs.
- The solution in the discrete case is:

$$\boldsymbol{F}(t) = \mathbf{H}(t)\boldsymbol{f}$$

- where $\mathbf{H}$ denotes the discrete heat operator:

$$\mathbf{H}(t) = e^{-t\mathbf{L}}$$

- $\boldsymbol{f}$ corresponds to the initial heat distribution:

$$\boldsymbol{F}(0) = \boldsymbol{f}$$

- Starting with a point-heat distribution at vertex v_i, e.g., $(0 \dots f_i = 1 \dots 0)^\top$, the distribution at t, i.e., $\boldsymbol{F}(t) = (F_1(t) \dots F_n(t))$ is given by the i-th column of the heat operator.

How to Compute the Heat Matrix?

- The exponential of a matrix:

$$e^{\mathbf{A}} = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!}$$

- Hence:

$$e^{-t\mathbf{L}} = \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \mathbf{L}^k$$

- It belongs to the "exponential diffusion" family of kernels just introduced with $\mu = -t, t > 0$.

Spectral Properties of $\mathbf{L}$

We start by recalling some basic facts about the combinatorial graph Laplacian:

- Symmetric semi-definite positive matrix: $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$
- Eigenvalues: $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$
- Eigenvectors: $\boldsymbol{u}_1 = \mathbb{1}, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$
- λ_2 and $\boldsymbol{u}_2$ are the Fiedler value and the Fiedler vector
- $\boldsymbol{u}_i^\top \boldsymbol{u}_j = \delta_{ij}$
- $\boldsymbol{u}_{i>1}^\top \mathbb{1} = 0$
- $\sum_{i=1}^n u_{ik} = 0, \forall k \in \{2, \ldots, n\}$
- $-1 < u_{ik} < 1, \forall i \in \{1, \ldots, n\}, \forall k \in \{2, \ldots, n\}$

$$\mathbf{L} = \sum_{k=2}^{n} \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^\top$$

The Heat-kernel Matrix

$$\mathbf{H}(t) = e^{-t\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top} = \mathbf{U}e^{-t\mathbf{\Lambda}}\mathbf{U}^\top \succeq 0$$

with:

$$e^{-t\mathbf{\Lambda}} = \mathsf{Diag}[e^{-t\lambda_1} \dots e^{-t\lambda_n}]$$

- Eigenvalues: $1 = e^{-t0} > e^{-t\lambda_2} \geq \dots \geq e^{-t\lambda_n}$
- Eigenvectors: same as the Laplacian matrix with their properties (previous slide).
- The heat trace (also referred to as the partition function):

$$Z(t) = \mathsf{tr}(\mathbf{H}) = \sum_{k=1}^{n} e^{-t\lambda_k}$$

- The determinant:

$$\det(\mathbf{H}) = \prod_{k=1}^{n} e^{-t\lambda_k} = e^{-t\mathsf{tr}(\mathbf{L})} = e^{-t\mathsf{vol}(\mathcal{G})}$$

The Heat-kernel

- Computing the heat matrix:

$$\mathbf{H}(t) = \sum_{k=2}^{n} e^{-t\lambda_k} \boldsymbol{u}_k \boldsymbol{u}_k^\top$$

where we applied a *deflation* to get rid of the constant eigenvector: $\mathbf{H} \longrightarrow \mathbf{H} - \boldsymbol{u}_1 \boldsymbol{u}_1^\top$

- The heat kernel (en entry of the matrix above):

$$h(i,j;t) = \sum_{k=2}^{n} e^{-t\lambda_k} u_{ik} u_{jk}$$

Feature-space Embedding Using the Heat Kernel

$$\mathbf{H}(t) = \left(\mathbf{U}e^{-\frac{1}{2}t\mathbf{\Lambda}}\right)\left(\mathbf{U}e^{-\frac{1}{2}t\mathbf{\Lambda}}\right)^\top$$

- Each row of the $n \times n$ matrix $\mathbf{U}e^{-t\mathbf{\Lambda}/2}$ can be viewed as the coordinates of a graph vertex in a feature space, i.e., the mapping $\phi : \mathcal{V} \to \mathbb{R}^{n-1}$, $\boldsymbol{x}_i = \boldsymbol{\phi}(v_i)$:

$$\begin{aligned} \boldsymbol{x}_i &= \left(\begin{array}{ccccc} e^{-\frac{1}{2}t\lambda_2}u_{i2} & \dots & e^{-\frac{1}{2}t\lambda_k}u_{ik} & \dots & e^{-\frac{1}{2}t\lambda_n}u_{in} \end{array} \right)^\top \\ &= (x_{i2} \dots x_{ik} \dots x_{in})^\top \end{aligned}$$

- The heat-kernel computes the inner product in feature space:

$$h(i, j; t) = \langle \boldsymbol{\phi}(v_i), \boldsymbol{\phi}(v_j) \rangle$$

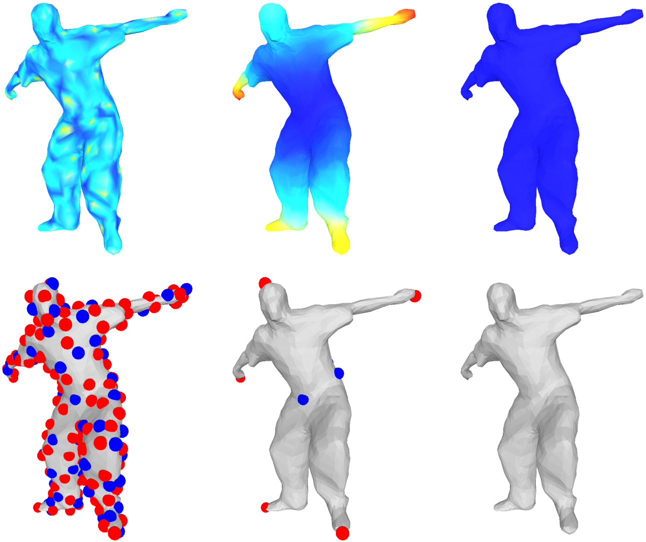
The Auto-diffusion Function

- Each diagonal term of the heat matrix corresponds to the norm of a feature-space point:

$$h(i,i;t) = \sum_{k=2}^{n} e^{-t\lambda_k} u_{ik}^2 = \|\boldsymbol{x}_i\|^2$$

- This is also known as the *auto-diffusion function* (ADF), or the amount of heat that remains at a vertex at time t.
- The local maxima/minima of this function have been used for a feature-based scale-space representation of shapes.
- Associated shape descriptor: $v_i \rightarrow h(i,i;t)$ hence it is a scalar function defined over the graph.

The ADF as a Shape Descriptor



$t = 0$ $t = 50$ $t = 500000$

Spectral Distances

- The heat distance:

$$\begin{aligned} d_t^2(i,j) &= h(i,i;t) + h(j,j;t) - 2h(i,j;t) \\ &= \sum_{k=2}^{n} (e^{-\frac{1}{2}t\lambda_k}(u_{ik} - u_{jk}))^2 \end{aligned}$$

- The commute-time distance:

$$\begin{aligned} d_{\mathsf{CTD}}^2(i,j) &= \int_{t=0}^{\infty} \sum_{k=2}^{n} (e^{-\frac{1}{2}t\lambda_k}(u_{ik} - u_{jk}))^2 dt \\ &= \sum_{k=2}^{n} \left(\lambda_k^{-1/2}(u_{ik} - u_{jk})\right)^2 \end{aligned}$$

Principal Component Analysis

- The covariance matrix in feature space:

$$\mathbf{C}_X = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^\top$$

- With:

$$\mathbf{X} = \left(\mathbf{U}e^{-\frac{1}{2}t\boldsymbol{\Lambda}}\right)^\top = [\boldsymbol{x}_1 \ldots \boldsymbol{x}_i \ldots \boldsymbol{x}_n]$$

- Remember that each column of $\mathbf{U}$ sums to zero.
- $-1 < -e^{-\frac{1}{2}t\lambda_k} < x_{ik} < e^{-\frac{1}{2}t\lambda_k} < 1, \forall 2 \leq k \leq n$

Principal Component Analysis: The Mean

$$
\begin{aligned}
\overline{\boldsymbol{x}} &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \\
&= \frac{1}{n}e^{-\frac{1}{2}t\boldsymbol{\Lambda}}\begin{pmatrix} \sum_{i=1}^{n} u_{i2} \\ \vdots \\ \sum_{i=1}^{n} u_{in} \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}
\end{aligned}
$$

Principal Component Analysis: The Covariance

$$
\begin{aligned}
\mathbf{C}_X &= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \\
&= \frac{1}{n}\mathbf{X}\mathbf{X}^\top \\
&= \frac{1}{n}\left(\mathbf{U}e^{-\frac{1}{2}t\boldsymbol{\Lambda}}\right)^\top \left(\mathbf{U}e^{-\frac{1}{2}t\boldsymbol{\Lambda}}\right) \\
&= \frac{1}{n}e^{-t\boldsymbol{\Lambda}}
\end{aligned}
$$

Result I: The PCA of a Graph

- The eigenvectors (of the combinatorial Laplacian) are the principal components of the heat-kernel embedding: hence we obtain a maximum-variance embedding
- The associated variances are $e^{-t\lambda_2}/n, \ldots, e^{-t\lambda_n}/n$.
- The embedded points are strictly contained in a hyper-parallelepipedon with volume $\prod_{i=2}^{n} e^{-t\lambda_i}$.

Dimensionality Reduction (1)

- Dimensionality reduction consists in selecting the K largest eigenvalues, $K < n$, conditioned by t, hence the criterion: choose K and t, such that (scree diagram):

$$\alpha(K) = \frac{\sum_{i=2}^{K+1} e^{-t\lambda_i}/n}{\sum_{i=2}^{n} e^{-t\lambda_i}/n} \approx 0.95$$

- This is not practical because one needs to compute all the eigenvalues.

Dimensionality Reduction (2)

- An alternative possibility is to use the determinant of the covariance matrix, and to choose the first K eigenvectors such that (with $\alpha > 1$):

$$\alpha(K) = \ln \frac{\prod_{i=2}^{K+1} e^{-t\lambda_i}/n}{\prod_{i=2}^{n} e^{-t\lambda_i}/n}$$

- which yields:

$$\alpha(K) = t\left(\text{tr}(\mathbf{L}) - \sum_{i=2}^{K+1} \lambda_i\right) + (n-K)\ln n$$

- This allows to choose K for a scale t.

Normalizing the Feature-space

Observe that the heat-kernels collapse to 0 at infinity: $\lim_{t\to\infty} h(i,j;t) = 0$. To prevent this problem, several normalizations are possible:

- Trace normalization
- Unit hyper-sphere normalization
- Time-invariant embedding

Trace Normalization

- Observe that $\lim_{t\rightarrow\infty} h(i,j;t) = 0$
- Use the trace of the operator to normalize the embedding:

$$\widehat{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i}{\sqrt{Z(t)}}$$

with: $Z(t) \approx \sum_{k=2}^{K+1} e^{-t\lambda_k}$

- the k-component of the i-coordinate writes:

$$\hat{x}_{ik}(t) = \frac{\left(e^{-t\lambda_k} u_{ik}^2\right)^{1/2}}{\left(\sum_{l=2}^{K+1} e^{-t\lambda_l}\right)^{1/2}}$$

- At the limit:

$$\widehat{\boldsymbol{x}}_i(t \rightarrow \infty) = \begin{pmatrix} \frac{u_{i2}}{\sqrt{m}} & \dots & \frac{u_{i\,m+1}}{\sqrt{m}} & 0 & \dots & 0 \end{pmatrix}^\top$$

where m is the multiplicity of the first non-null eigenvalue.

Unit Hyper-sphere Normalization

- The embedding lies on a unit hyper-sphere of dimension K:

$$\widetilde{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|}$$

- The heat distance becomes a geodesic distance on a spherical manifold:

$$d_{\mathcal{S}}(i,j;t) = \arccos \widetilde{\boldsymbol{x}}_i^\top \widetilde{\boldsymbol{x}}_j = \arccos \frac{h(i,j;t)}{(h(i,i;t)h(j,j;t))^{1/2}}$$

- At the limit (m is the multiplicity of the largest non-null eigenvalue):

$$\widetilde{\boldsymbol{x}}_i(t \rightarrow \infty) = \left(\frac{u_{i2}}{\left(\sum_{l=2}^{m+1} u_{il}^2\right)^{1/2}} \quad \dots \quad \frac{u_{i\,m+1}}{\left(\sum_{l=2}^{m+1} u_{il}^2\right)^{1/2}} \quad 0 \quad \dots \quad 0 \right)^\top$$

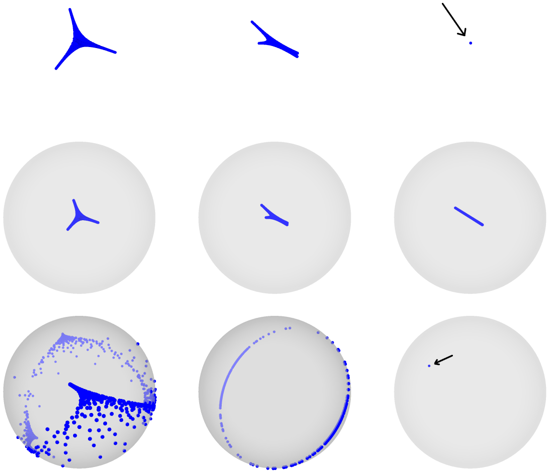
Time-invariant Embedding

- Integration over time:

$$\begin{aligned}\mathbf{L}^{\dagger} &= \int_0^{\infty} \mathbf{H}(t) = \int_0^{\infty} \sum_{k=2}^{n} e^{-t\lambda_k} \boldsymbol{u}_k \boldsymbol{u}_k^{\top} dt \\ &= \sum_{k=2}^{n} \frac{1}{\lambda_k} \boldsymbol{u}_k \boldsymbol{u}_k^{\top} = \mathbf{U}\boldsymbol{\Lambda}^{\dagger}\mathbf{U}^{\top}\end{aligned}$$

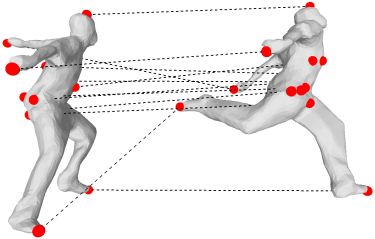
- with: $\boldsymbol{\Lambda}^{\dagger} = \text{Diag}[\lambda_2^{-1}, \ldots, \lambda_n^{-1}]$.
- Matrix $\mathbf{L}^{\dagger}$ is called the *discrete Green's function* [ChungYau2000], the Moore-Penrose pseudo-inverse of the Laplacian.
- Embedding: $\boldsymbol{x}_i = \left(\lambda_2^{-1/2} u_{i2} \;\ldots\; \lambda_{K+1}^{-1/2} u_{i\,K+1}\right)^{\top}$
- Covariance: $\mathbf{C}_X = \frac{1}{n}\text{Diag}[\lambda_2^{-1}, \ldots, \lambda_{K+1}^{-1}]$

Examples of Normalized Embeddings

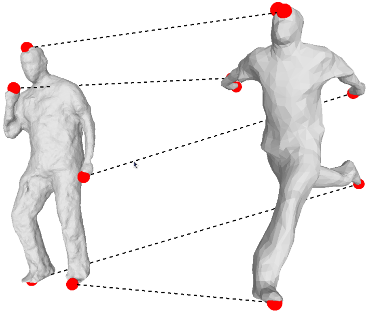


$t = 50$ $t = 5000$ $t = 500000$

Shape Matching (1)

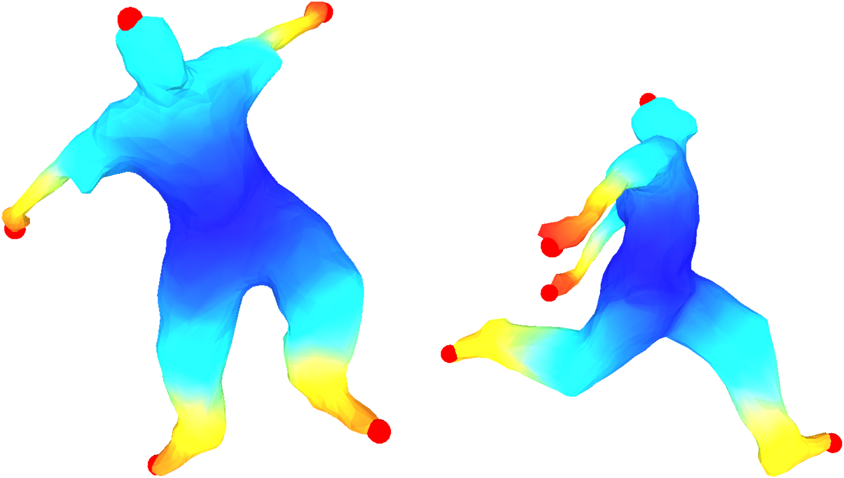


$t = 200,\ t' = 201.5$

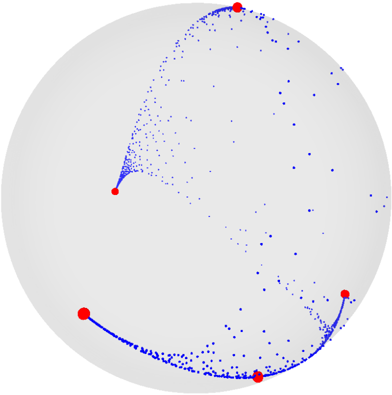


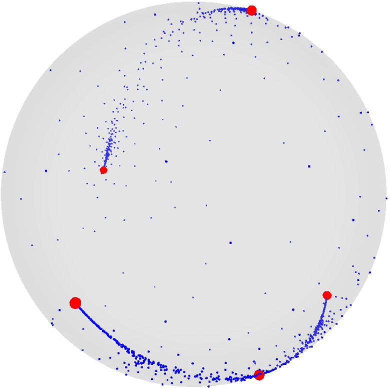
$t = 90,\ t' = 1005$

Shape Matching (2)



Shape Matching (3)





Sparse Shape Matching

- Shape/graph matching is equivalent to matching the embedded representations [Mateus et al. 2008]
- Here we use the projection of the embeddings on a unit hyper-sphere of dimension K and we apply rigid matching.
- How to select t and t', i.e., the scales associated with the two shapes to be matched?
- How to implement a robust matching method?

Scale Selection

- Let $\mathbf{C}_X$ and $\mathbf{C}_{X'}$ be the covariance matrices of two different embeddings $\mathbf{X}$ and $\mathbf{X}'$ with respectively n and n' points:

$$\det(\mathbf{C}_X) = \det(\mathbf{C}_{X'})$$

- $\det(\mathbf{C}_X$ measures the volume in which the embedding X lies. Hence, we impose that the two embeddings are contained in the same volume.
- From this constraint we derive:

$$t' \operatorname{tr}(\mathbf{L}') = t \operatorname{tr}(\mathbf{L}) + K \log n/n'$$

Robust Matching

- Build an association graph.
- Search for the largest set of mutually compatible nodes (maximal clique finding).
- See [Sharma and Horaud 2010] (Nordia workshop) for more details.