# Intrinsic Analysis of Undirected Weighted Graphs Based on the Heat Kernel

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

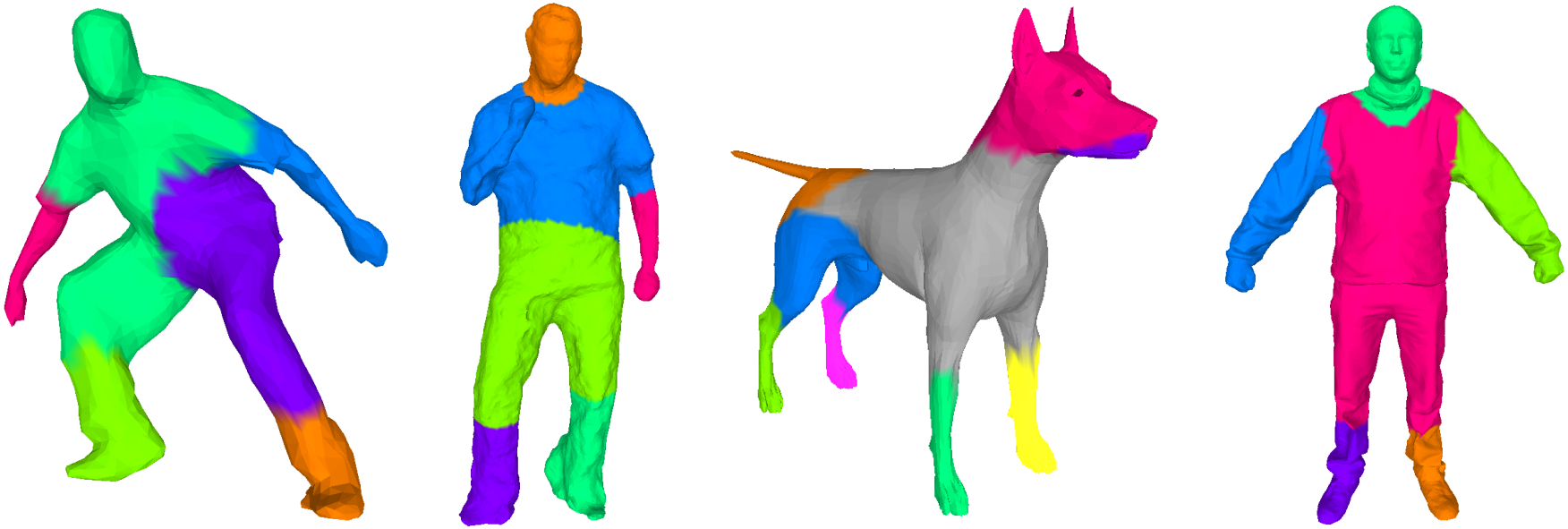
# Introduction

- Assume that the data (such as 3D shapes) lie on a closed Riemannian manifold $\mathcal{M} \subset \mathbb{R}^d$;
- The general idea is to characterize these data by embedding the manifold into a metric space;
- There is no *explicit* description of the manifold. Instead we have discrete data sampled from a continous surface, e.g., a *point cloud*.
- Embedding consists in two steps:
  1. Build an undirected weighted graph and
  2. Analyze the properties of the graph using the eigenvalues and eigenvectors of various operators or *graph matrices*.
- This will reveal the *intrinsic geometry* of the point-cloud/graph.

# The discrete heat operator

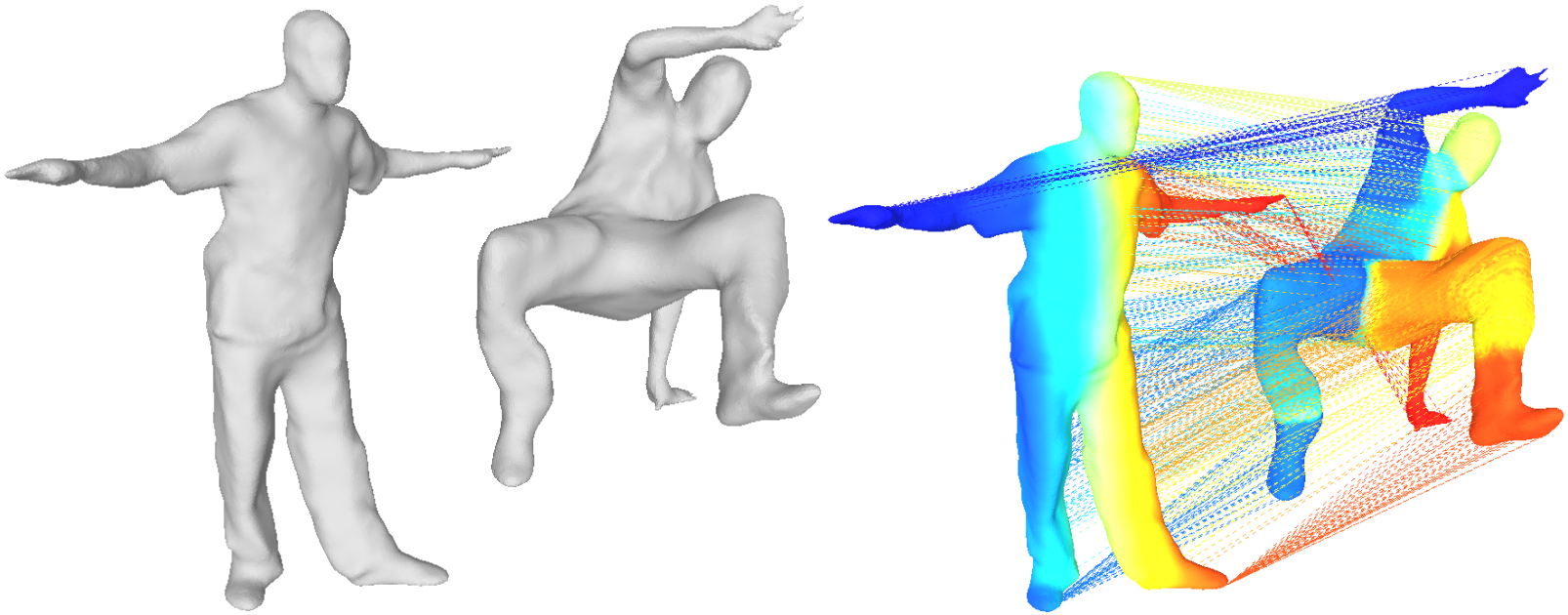
- We will base our analysis on the algebraic/spectral properties of the *discrete heat operator*, i.e., yet another **graph matrix**.
- This matrix can also be viewed as a *Gramm matrix* and hence each matrix entry can be viewed as a kernel, the *heat kernel* defining a dot product in the embedded space (or feature space).
- The heat kernel can be used in the framework of kernel methods.
- *Manifold embedding in a metric space with reasonable dimension* may be viewed as *data preprocessing* for many machine learning/vision tasks: clustering, dimensionality reduction, segmentation, matching, recognition, classification, etc.

# Segmentation



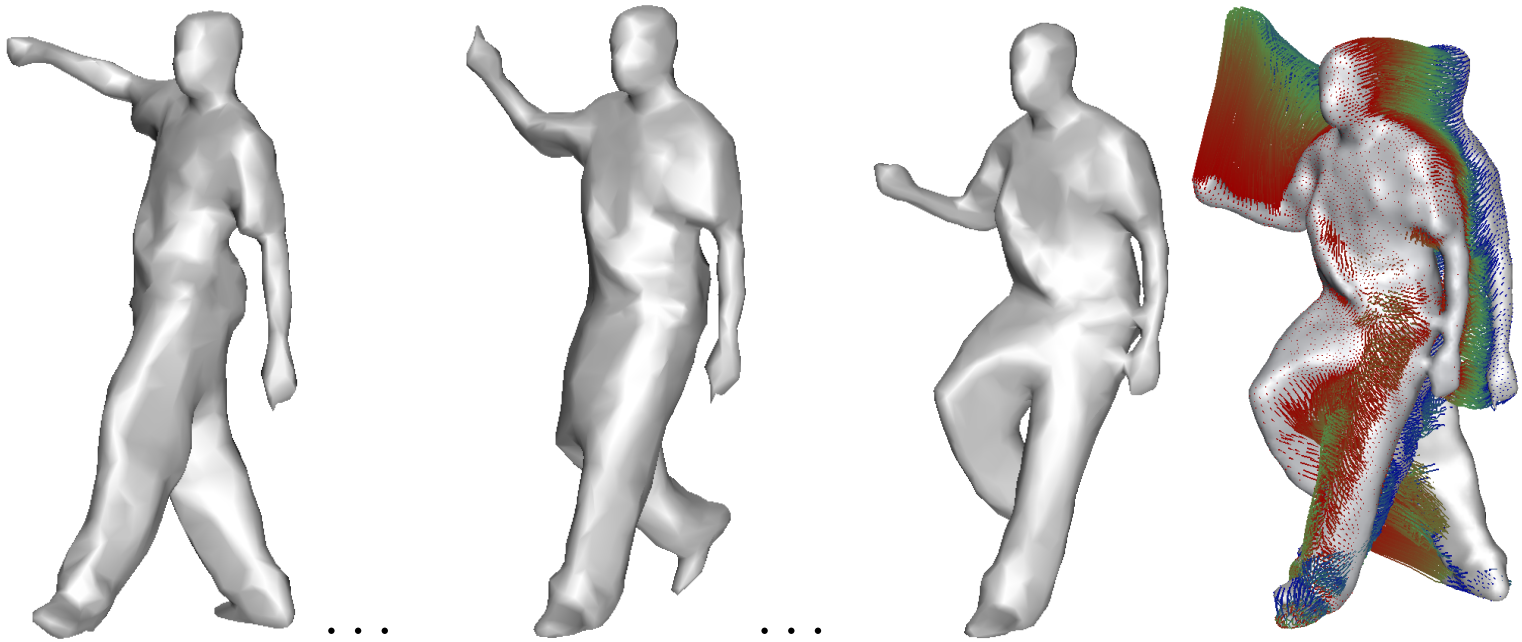
[Sharma et al 2009]

# Matching



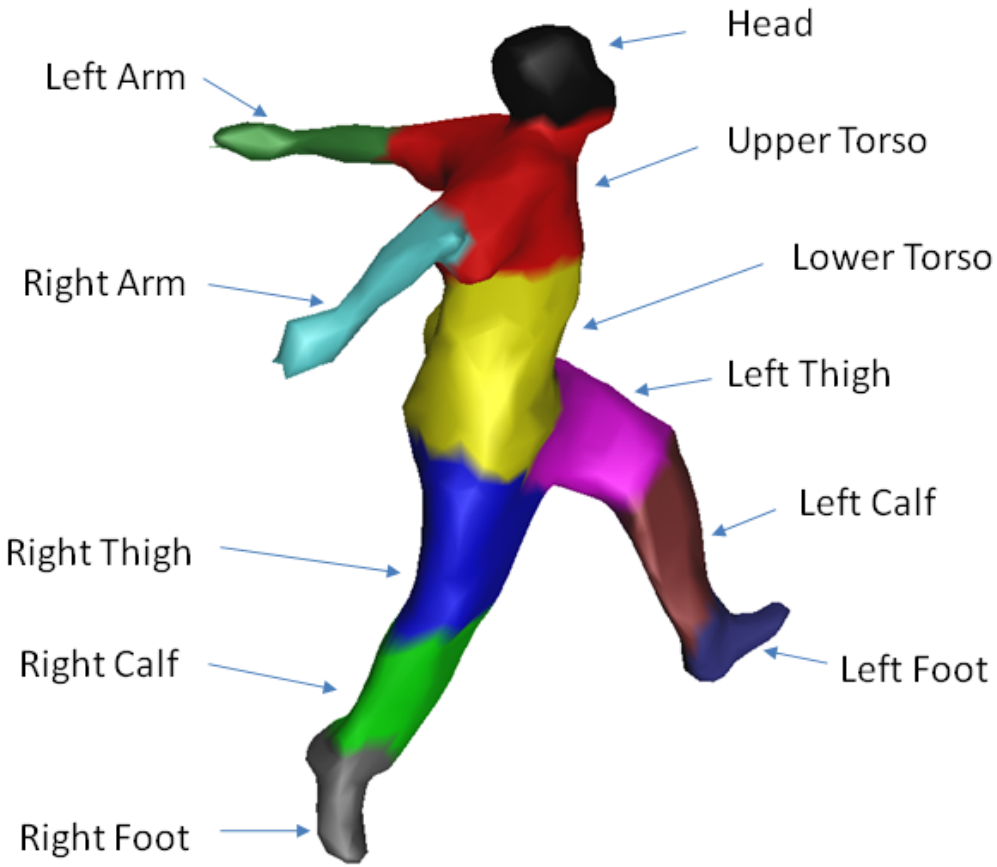
[Mateus et al 2008], [Knossow et al 2009]

# Tracking

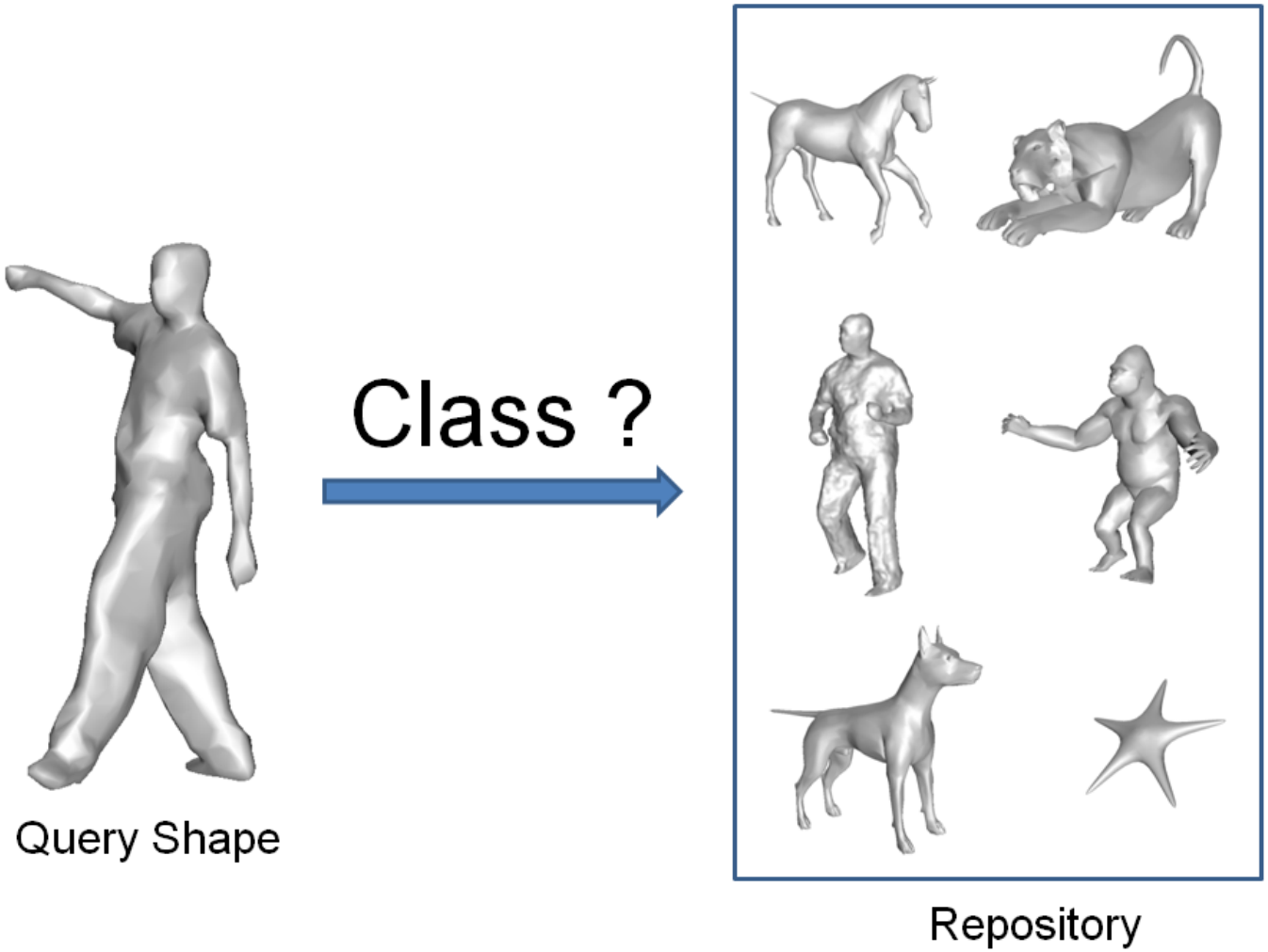


[Varanasi et al 2008]

# Recognition



# Classification



# Background

- *Algebraic/spectral graph theory* studies the eigenvalues and eigenvectors of the graph matrices (adjacency, Laplacian operators).
- *Kernel methods* study the data via the Gramm matrix, i.e., $G_{ij} =< \phi(x_i), \phi(x_j) >$, without making explicit the feature (embedded) space.
- Spectral methods for dimensionality reduction (PCA, MDS, LLE, Kernel PCA, Laplacian embedding, LTSA, etc.) search for a low-dimensional structure in high-dimensional data. However, we may end up in an embedded space with dimensionality **higher** than the initial data.

# Outline of the tutorial

- Graph matrices and their spectral properties
- Random walks on undirected weighted graphs (not addressed)
- Heat diffusion on a Riemannian manifold
- The discrete heat operator and the heat kernel
- Spectral properties
- Principal component analysis and dimensionality reduction
- Normalizing the embedding
- Application to shape analysis: scale-space feature extraction, segmentation, matching.

# Basic graph notations

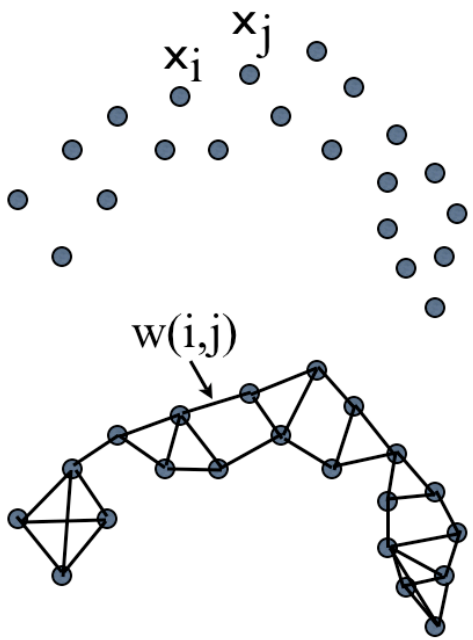
- We consider *undirected weighted graphs*:
  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with a node set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and an edge set $\mathcal{E} = \{e_{ij}\}$.
- Each edge $e_{ij}$ is weighted by $\omega_{ij}$.
- We consider real-valued functions $\boldsymbol{f} : \mathcal{V} \longrightarrow \mathbb{R}$.
- $\boldsymbol{f}$ is a vector indexed by the graph's vertices, hence $\boldsymbol{f} \in \mathbb{R}^n$

# Examples of graphs

- Electric networks
- Chemical structures
- Social networks
- Images
- Image databases
- Meshes (discretized surfaces)
- Shapes
- etc.

# The graph of a cloud of points

- $K$-nearest neighbor graph
- $\varepsilon$-radius graph
- A fully connected clique around each point [Weinberger & Saul 2006].
- KNN may guarantee that the graph is connected (depends on the implementation)
- $\varepsilon$-radius does not guarantee that the graph has one connected component
- $\boldsymbol{X}_i, \boldsymbol{X}_j \in \mathbb{R}^d$
- $w(i,j) > 0$
- Possible choice:
  $w(i,j) = \exp(-d^2(i,j)/\sigma^2)$



# The weighted adjacency matrix

- A real symmetric matrix defined by:

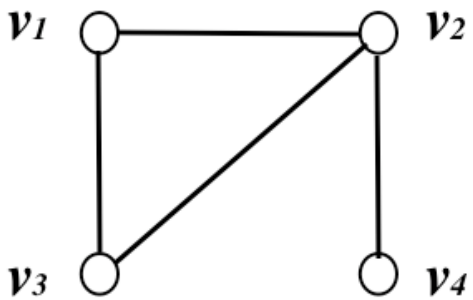
$$\mathbf{\Omega} = \begin{cases} \Omega(i,j) = \omega_{ij} & \text{if there is en edge } e_{ij} \\ \Omega(i,j) = 0 & \text{if there is no edge} \\ \Omega(i,i) = 0 \end{cases}$$

- The degree matrix: $\mathbf{D} = D_{ii} = \sum_{j=1}^{n} \omega_{ij}$.
- The graph volume: $\text{vol}(\mathcal{G}) = \sum_{i=1}^{n} D_{ii}$

# The Laplacian matrix of a graph

- $\mathbf{L} = \mathbf{D} - \mathbf{\Omega}$.
- Example: a binary-weighted graph and its Laplacian.

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$



# Other graph matrices

- The *normalized weighted adjacency matrix*

$$\mathbf{\Omega}_N = \mathbf{D}^{-1/2}\mathbf{\Omega}\mathbf{D}^{-1/2}$$

- The *transition* matrix of the associated time-reversible Markov chain:

$$\mathbf{\Omega}_R = \mathbf{D}^{-1}\mathbf{\Omega} = \mathbf{D}^{-1/2}\mathbf{\Omega}_N\mathbf{D}^{1/2}$$

# Other Laplacian matrices

- the *normalized Laplacian*:

$$\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}_C\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{\Omega}_N$$

- the *random-walk Laplacian* also referred to as the *discrete Laplace operator*:

$$\mathbf{L}_R = \mathbf{D}^{-1}\mathbf{L}_C = \mathbf{I} - \mathbf{\Omega}_R$$

# The Laplacian as an operator

- Consider real-valued functions $f : \mathcal{V} \longrightarrow \mathbb{R}$.
- $\boldsymbol{f} = (f_1 \dots f_n)$ is a vector indexed by the graph's vertices, hence $\boldsymbol{f} \in \mathbb{R}^n$.
- $\mathbf{L}$ is an operator, $\boldsymbol{g} = \mathbf{L}\boldsymbol{f}$, such that:

$$g_j = \sum_{v_j \sim v_k} w_{kj}(f_j - f_k)$$

- The associated quadratic form:

$$\boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} = \sum_{e_{ij}} w_{ij}(f_i - f_j)^2$$

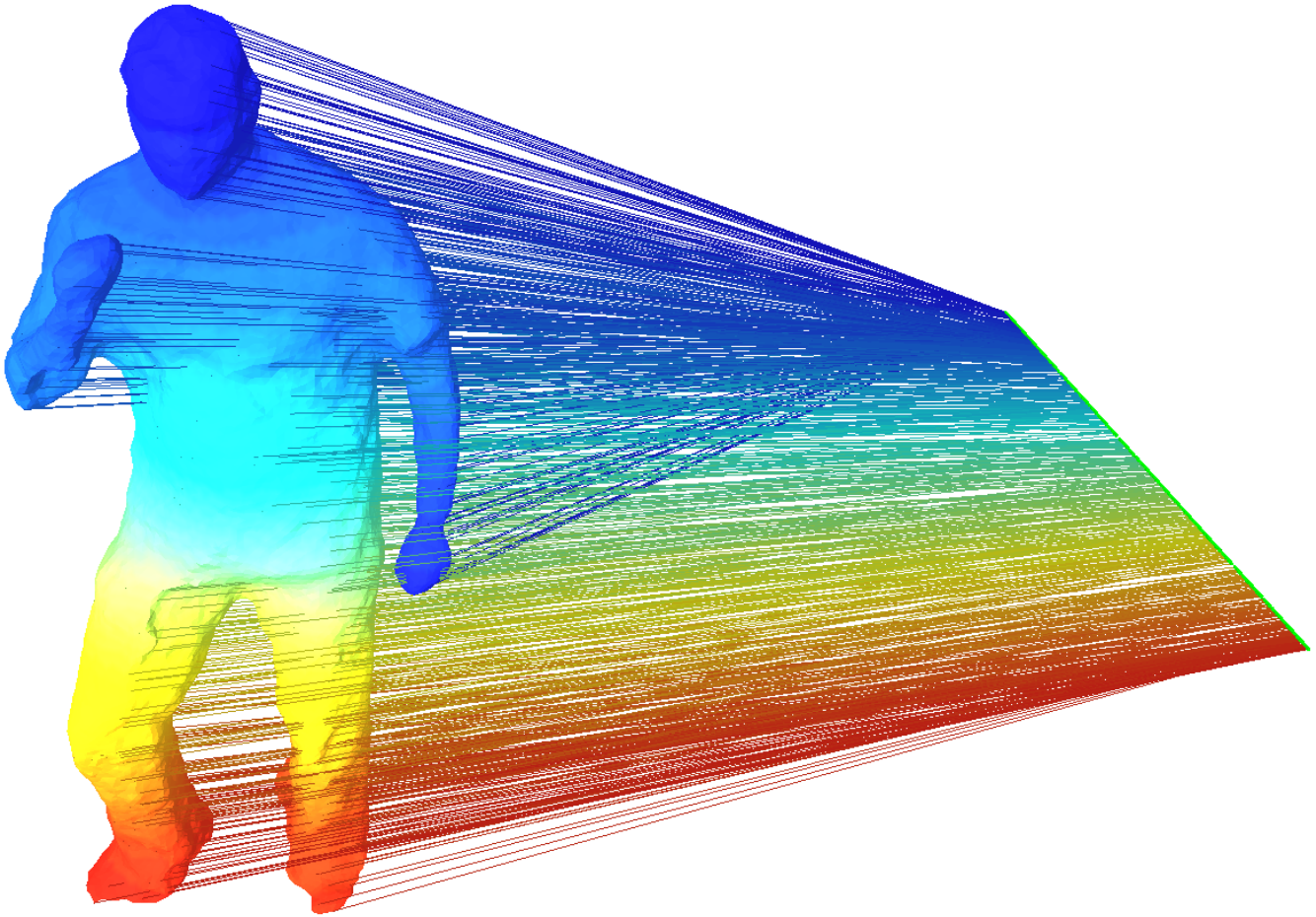
# Laplacian embedding: mapping a graph on an eigenvector

- Map a weighted graph onto a line such that connected nodes stay as close as possible, i.e., minimize $\sum_{i,j=1}^{n} W_{ij}(f_i - f_j)^2$, or:

$$\arg\min_{\boldsymbol{f}} \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} \text{ with: } \boldsymbol{f}^\top \boldsymbol{f} = 1 \text{ and } \boldsymbol{f}^\top \mathbf{1} = 0$$

- The solution is the eigenvector associated with the smallest nonzero eigenvalue of the eigenvalue problem: $\mathbf{L}\boldsymbol{u} = \lambda \boldsymbol{u}$, namely the Fiedler vector $\boldsymbol{u}_2$.

# Example of mapping a shape onto the Fiedler vector



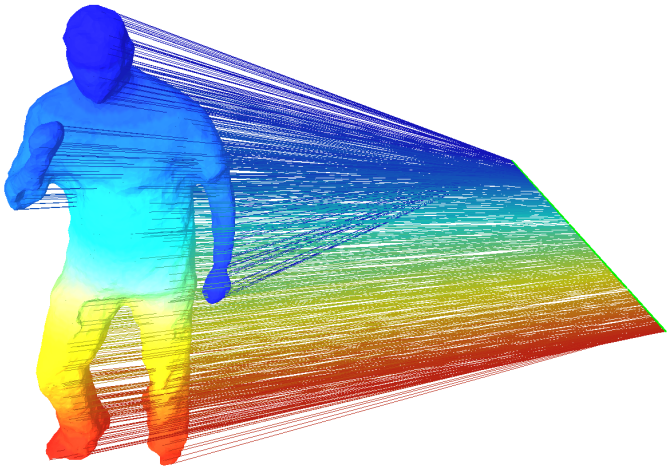
# Laplacian embedding

- Embed the graph in a $k$-dimensional Euclidean space. The embedding is given by the $n \times k$ matrix $\mathbf{F} = [\boldsymbol{f}_1 \boldsymbol{f}_2 \dots \boldsymbol{f}_k]$ where the $i$-th row of this matrix – $\boldsymbol{f}^i$ – corresponds to the Euclidean coordinates of the $i$-th graph node $v_i$.
- We need to minimize [Belkin & Niyogi '03]:

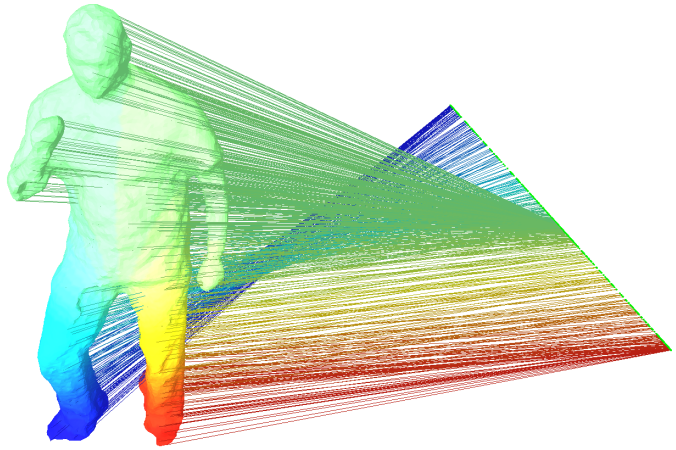
$$\arg \min_{\boldsymbol{f}_1 \cdots \boldsymbol{f}_k} \sum_{i,j=1}^{n} W_{ij} \|\boldsymbol{f}^i - \boldsymbol{f}^j\|^2 \text{ with: } \mathbf{F}^\top \mathbf{F} = \mathbf{I}.$$

- The solution is provided by the matrix of eigenvectors corresponding to the $k$ lowest nonzero eigenvalues of the eigenvalue problem $\mathbf{L}\boldsymbol{u} = \lambda \boldsymbol{u}$.

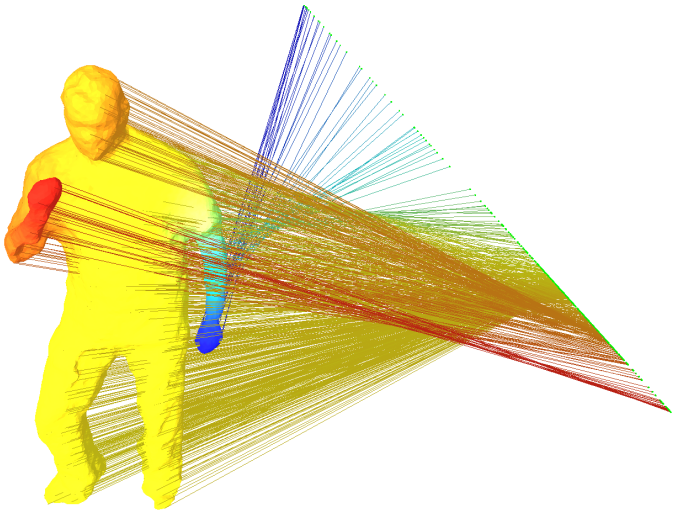
# Examples of one-dimensional mappings



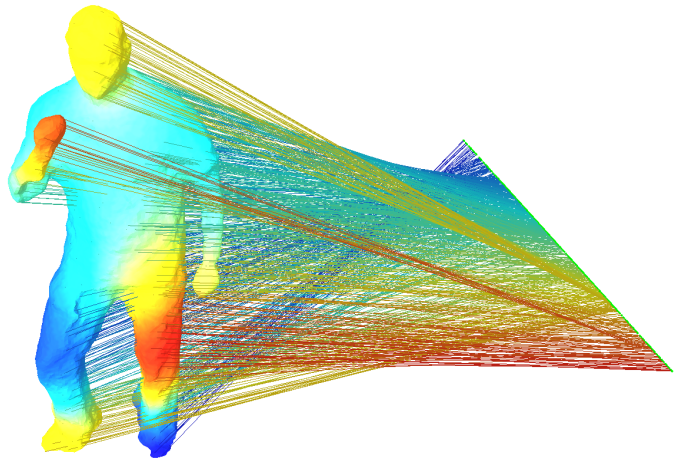
$\boldsymbol{u}_2$



$\boldsymbol{u}_3$



$\boldsymbol{u}_4$



$\boldsymbol{u}_8$

# Heat diffusion on a graph

- Diffusion 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 at time $t$ and at each manifold location.
- 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 $\boldsymbol{f}(t) = (f_1(t) \dots f_n(t))$ 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, see [Chung 1997], [Chung & Yau 2000].
- The solution in the discrete case is:

$$\boldsymbol{f}(t) = \mathbf{H}(t)\boldsymbol{f}(0)$$

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

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

- $\boldsymbol{f}(0)$ corresponds to the initial heat distribution:

$$\boldsymbol{f}(0) = (0 \dots f_i = 1 \dots 0)$$

- Starting with this distribution, the heat 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:

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

(More on this later)

# 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: $\boldsymbol{u}_1 = \mathbb{1}, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$
- $\lambda_2$ 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 matrix

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

with:

$$e^{-t\mathbf{\Lambda}} = \text{Diag}\left[e^{-t\lambda_1} \ldots e^{-t\lambda_n}\right]$$

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

$$Z(t) = \text{trace}\left(\mathbf{H}\right) = \sum_{k=1}^{n} e^{-t\lambda_k}$$

- The determinant:

$$\det(\mathbf{H}) = \prod_{k=1}^{n} e^{-t\lambda_k} = e^{-t\text{trace}\left(\mathbf{L}\right)} = e^{-t\text{vol}(\mathcal{G})}$$

# The heat matrix/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

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

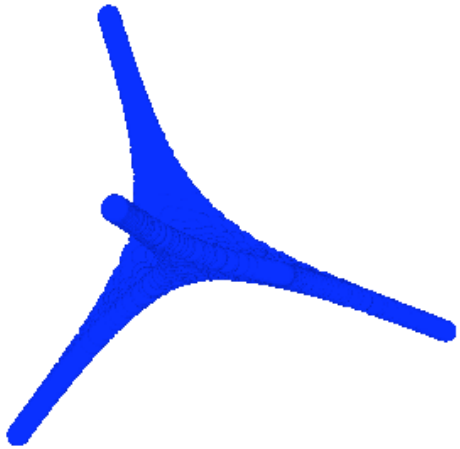
- Each row of the $n \times n$ matrix $\mathbf{U}e^{-t\mathbf{\Lambda}/2}$ can be viewed as the coordinates of a graph vertex in a feature space, i.e., the mapping $F : \mathcal{V} \rightarrow \mathbb{R}^{n-1}$, $\boldsymbol{x}_i = F(v_i)$:

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

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

$$h(i,j;t) = < F(v_i), F(v_j) >$$

# Example: Shape embedding



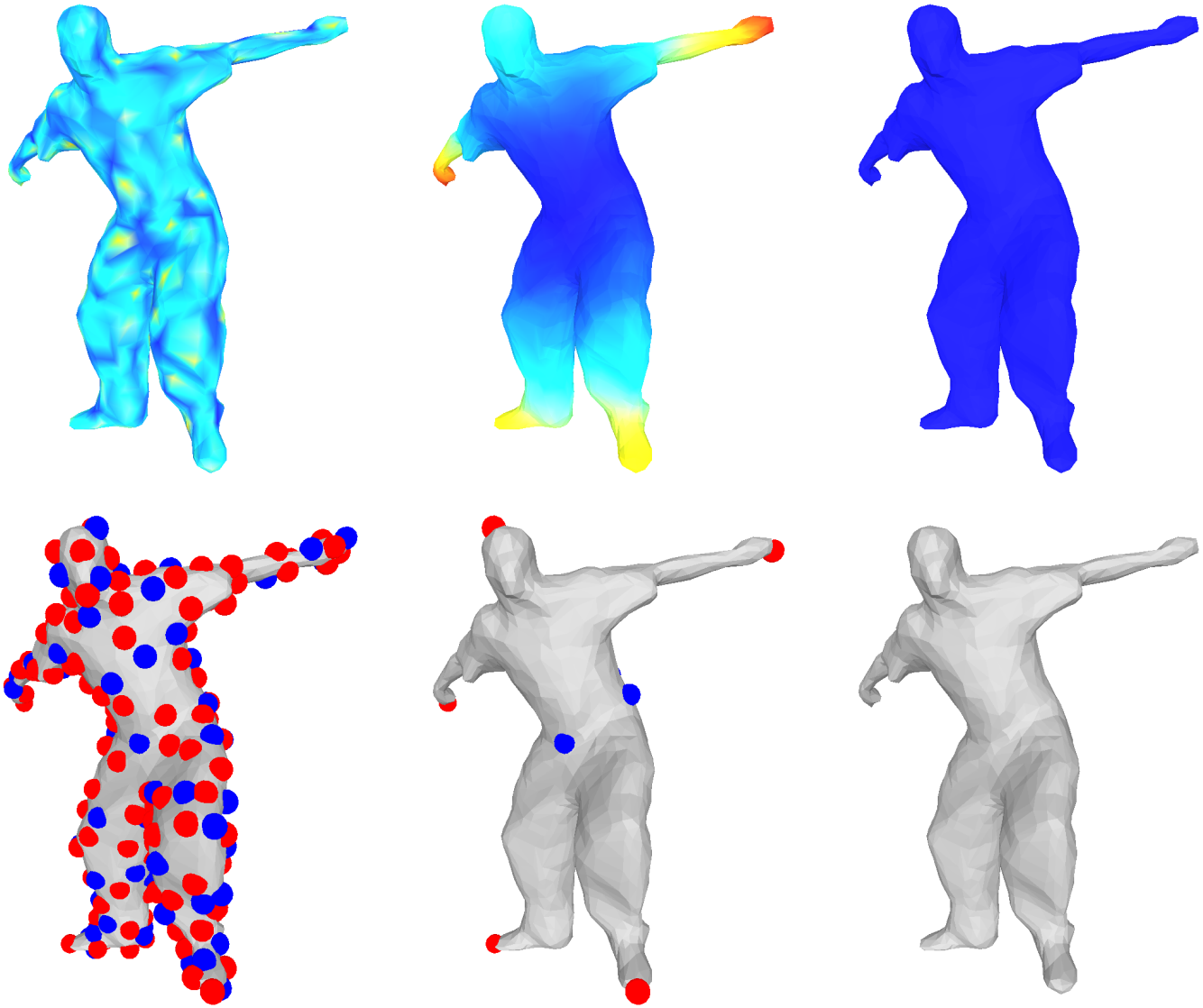
# The auto-diffusion function

- Each diagonal term of the heat matrix corresponds to the square Euclidean 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, 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.

# Shape description with the heat-kernel



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

# 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:

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

# Principal component analysis

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

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

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

# Result I: The PCA of a graph/shape

- The eigenvectors (Laplacian eigenvectors) are the principal components of the heat-kernel embedding: hence we obtain a maximum-variance embedding
- The associated "hyper-ellipsoid" has eccentricities $e^{-t\lambda_2}/n, \ldots, e^{-t\lambda_n}/n$.
- 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 $K$ such that (scree diagram)

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

- 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:

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

- This allows to choose $K$ for a scale $t$.

# Normalizing the embedding

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 $\lim_{t\to\infty} h(i,j;t) = 0$
- 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:

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

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):

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

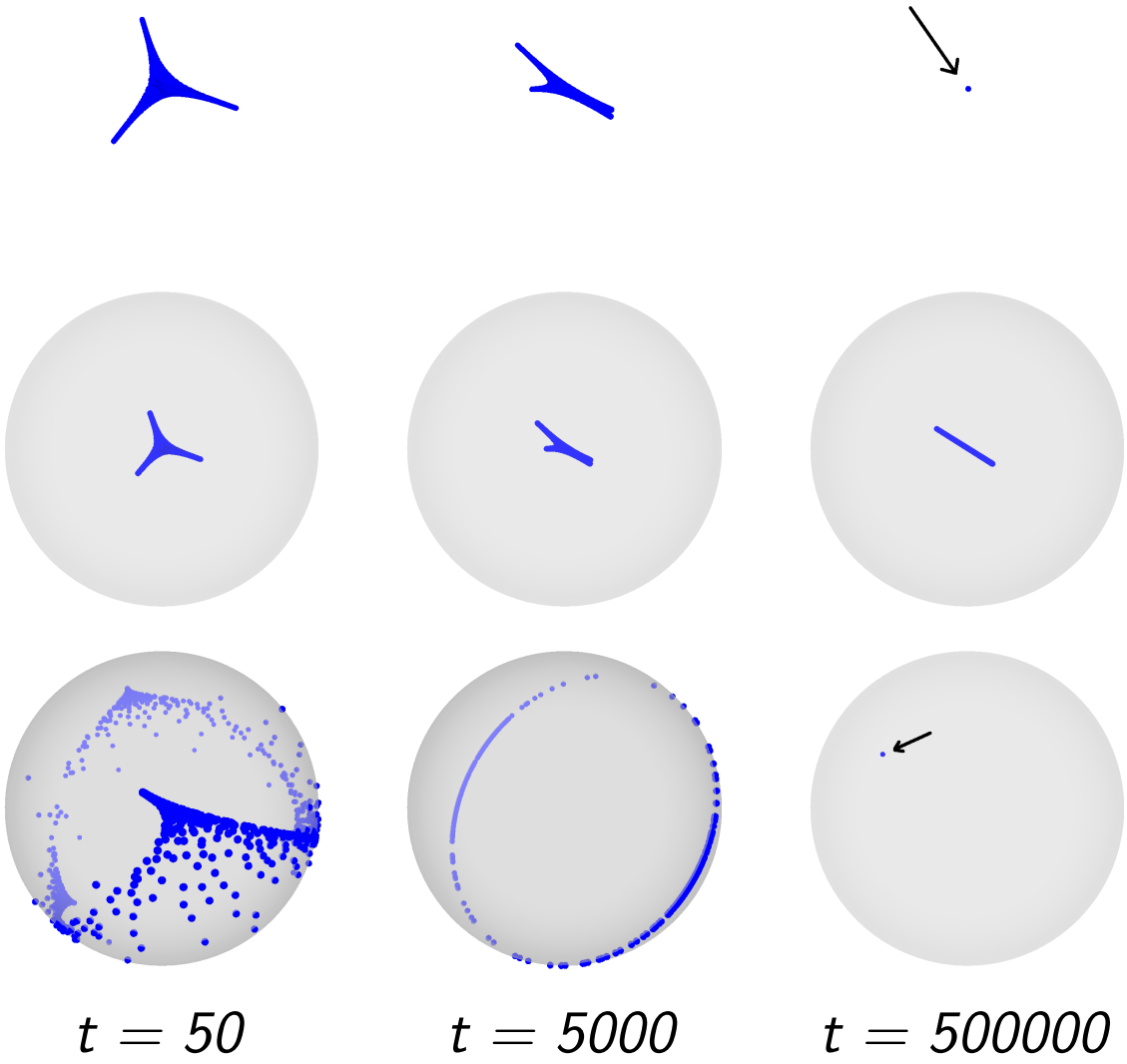
# Time-invariant embedding

- Integration over time:

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

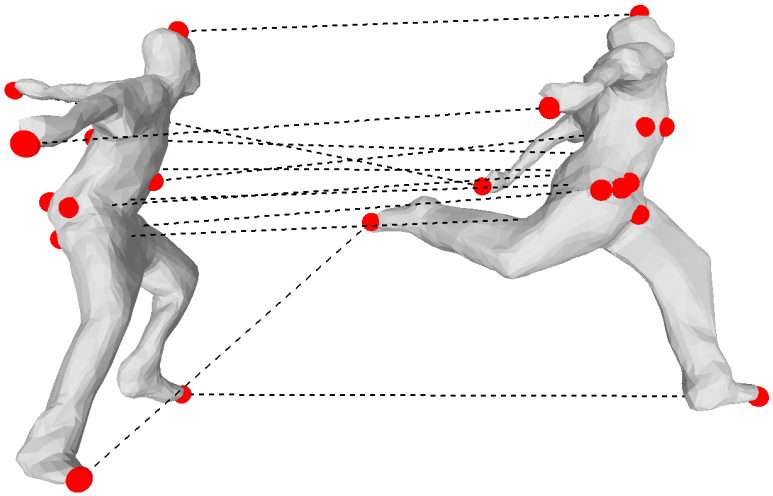
- with: $\boldsymbol{\Lambda}^\dagger = \text{Diag}\,[\lambda_2^{-1}, \ldots, \lambda_n^{-1}]$.
- Matrix $\mathbf{L}^\dagger$ is called the *discrete Green's function* [ChungYau2000], the Moore-Penrose pseudo-inverse of the Laplacian.
- Embedding: $\boldsymbol{x}_i = \left(\lambda_2^{-1/2} u_{i2} \;\ldots\;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

# Examples of normalized embeddings

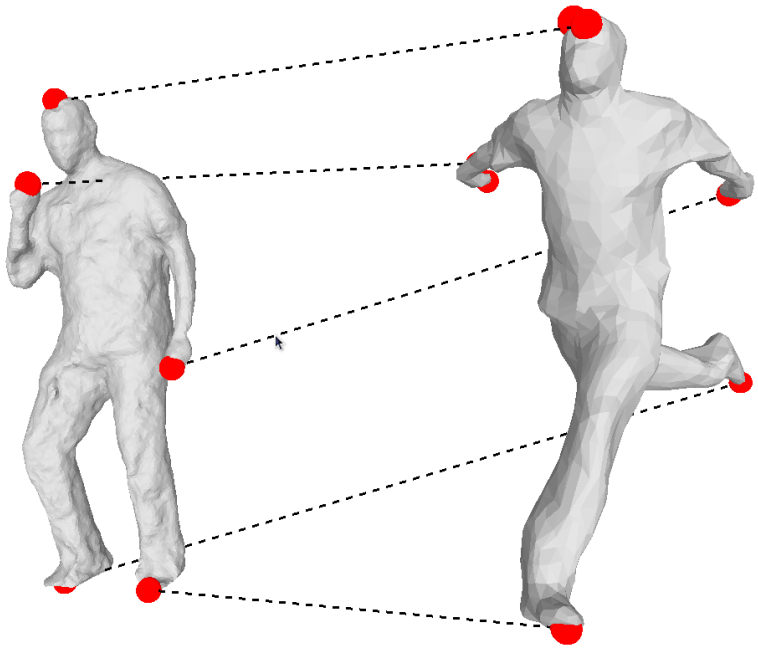


$t = 50$ $t = 5000$ $t = 500000$

# Shape matching (1)

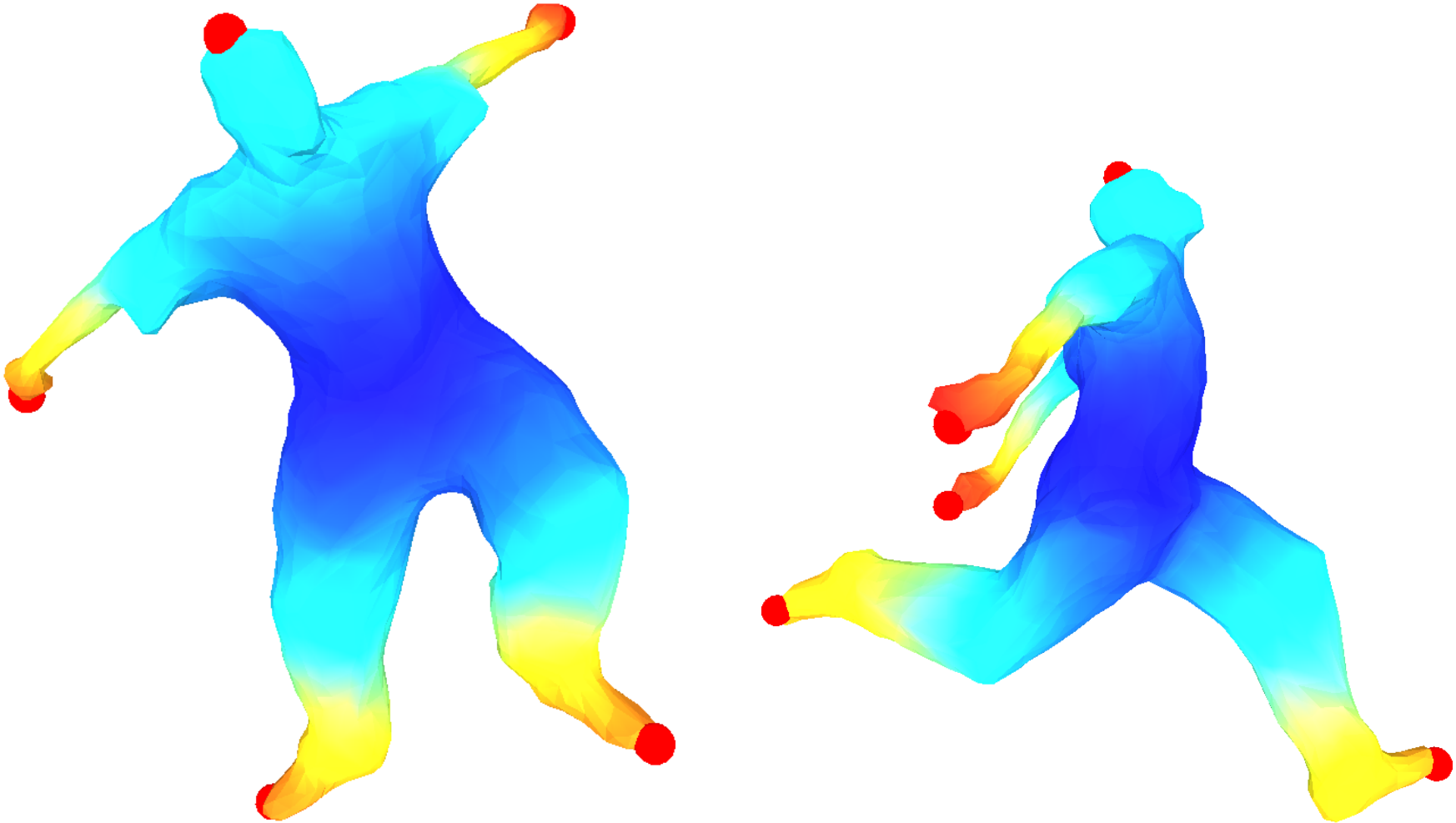


$t = 200,\ t' = 201.5$

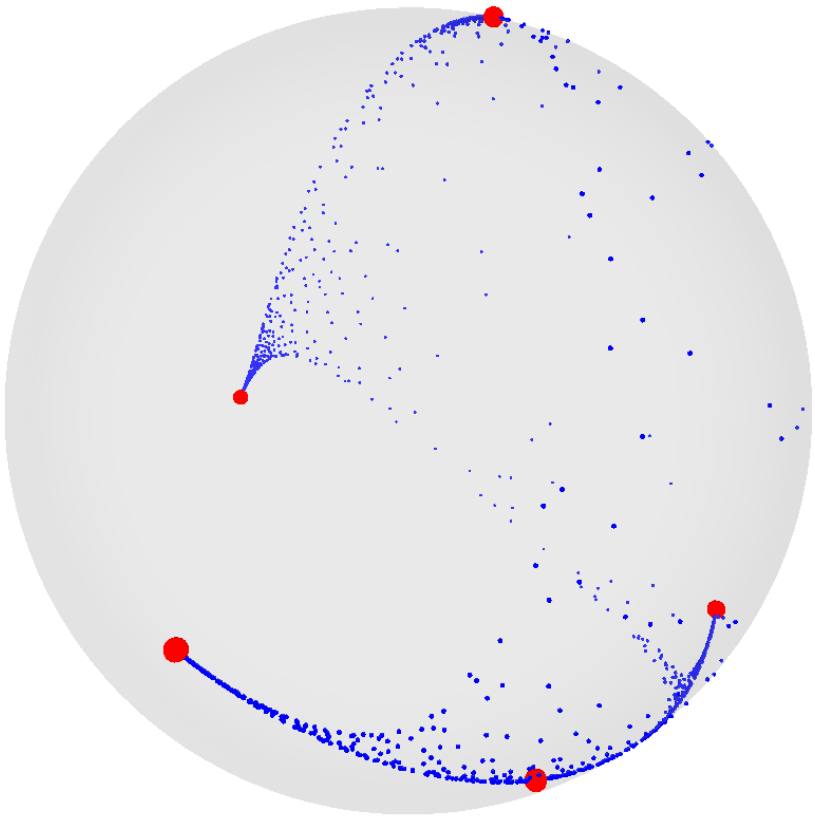


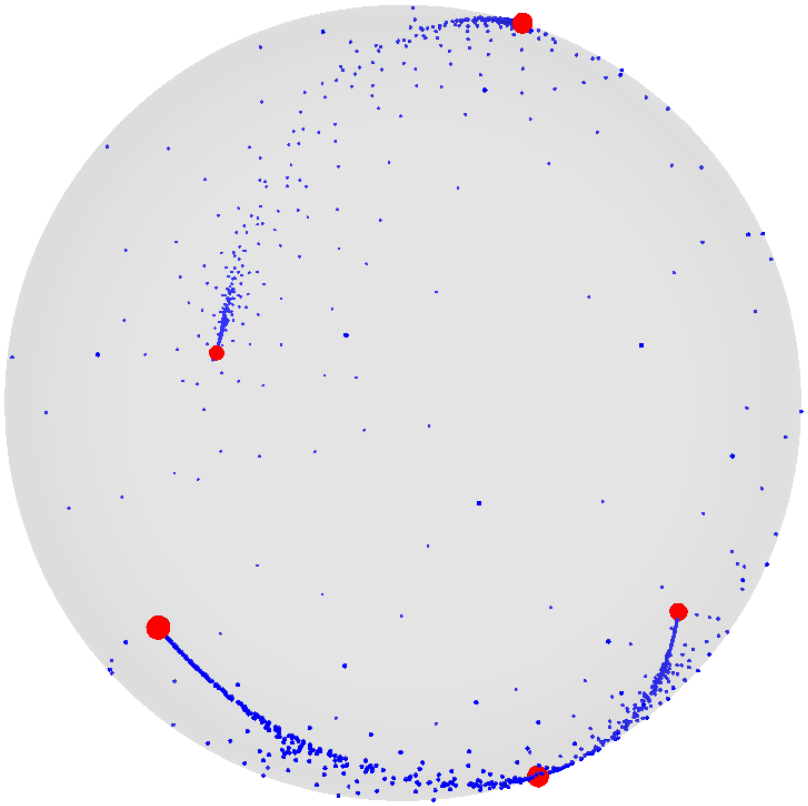
$t = 90,\ t' = 1005$

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

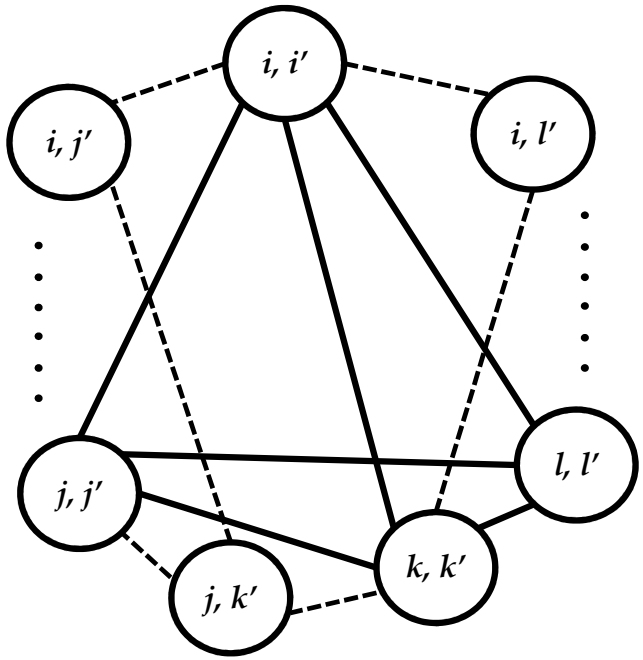
$$\det(\mathbf{S}_X) = \det(\mathbf{S}_{X'})$$

- $\det(\mathbf{S}_X$ 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:

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

# Robust matching

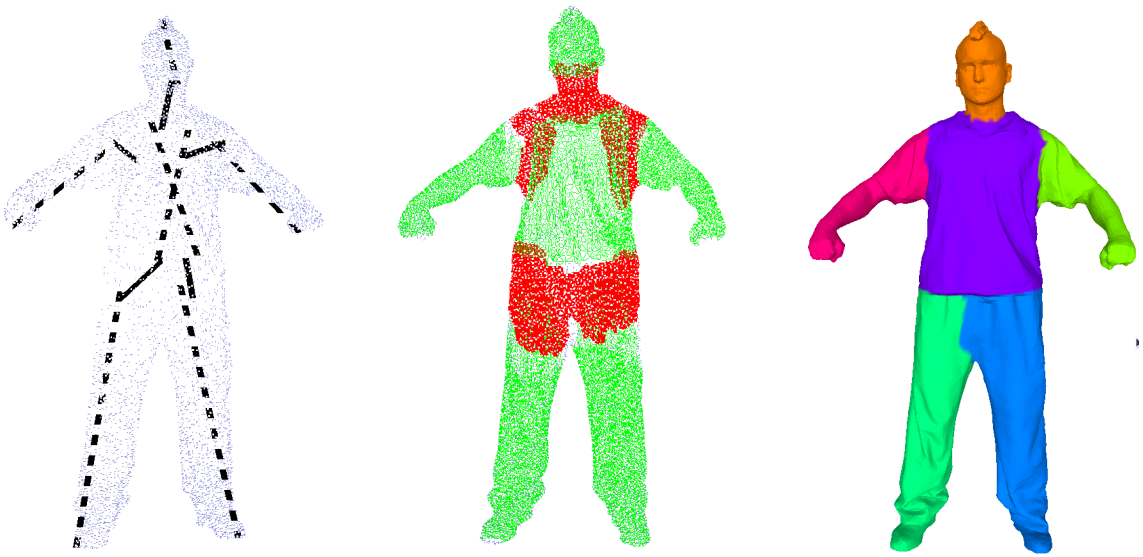
- Build an association graph.
- Search for the largest set of mutually compatible nodes (maximal clique finding).
- See [Sharma and Horaud 2010] (Nordia workshop) for more details.



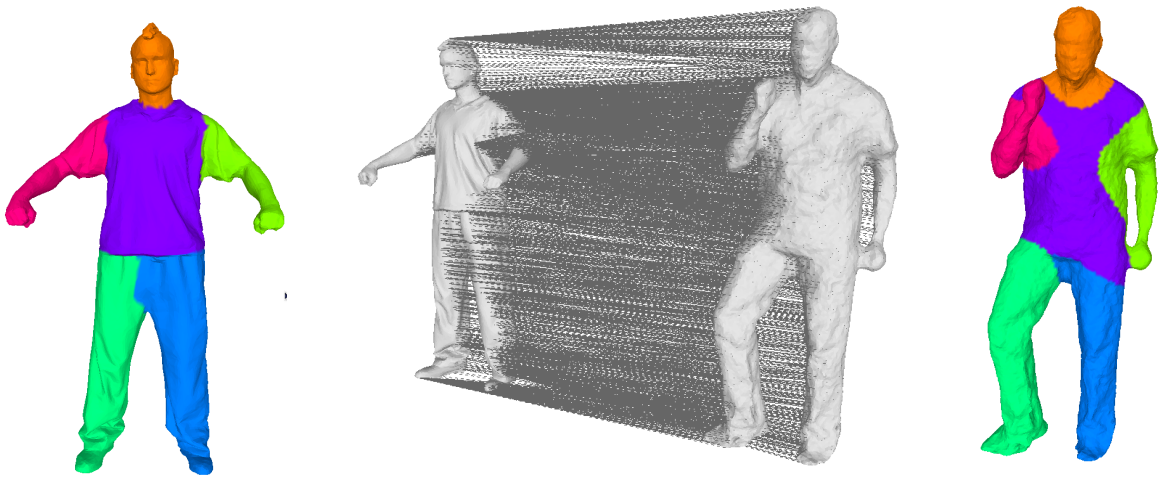
# Other uses of heat diffusion

- Graph partitioning / spectral clustering [Lafon and Coifman 2006], [Fouss et al. 2007]
- Constrained spectral clustering (using the time-invariant embedding and the commute-time distance) [Sharma et al. 2010] (ECCV).
- Supervised and semi-supervised kernel-based learning [Shawe-Taylor and Christianini 2004].

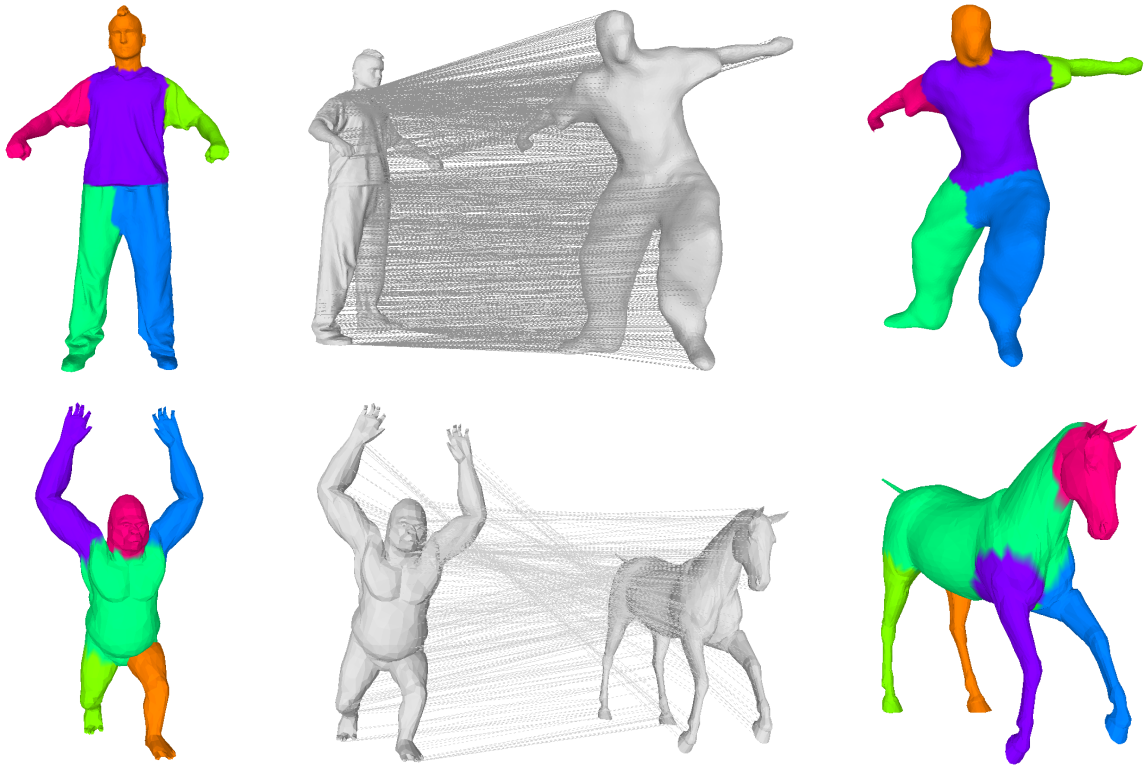
# Segmentation and matching: (1) constrained spectral clustering



# Segmentation and matching: (2) probabilistic label transfer



# Other examples



# Conclusions

- A 3D shape can be viewed as a graph whose nodes are sampled from a Riemannian manifold.
- Laplacian embedding is the standard way of representing such types of data in a metric (feature) space.
- Diffusion embedding is a more general principle that allows an elegant interpretation of the embedded space: heat kernel, spectral distances, principal component analysis, dimensionality reduction, data normalization, etc.
- It also allows a scale-space representation of the graph/shape based on the auto-diffusion function.
- It is an *intrinsic* representation/analysis that does not depend on the dimensionality of the space in which the data are observed.

# Thank you!

Radu.Horaud@inrialpes.fr
(ask for the latest version of the slides.)

`http://perception.inrialpes.fr/people/Horaud/Talks/ECCV10-Tutorial4-Horaud.pdf`