

Analysis of Three-Dimensional Shapes

(IN2238, TU München, Summer 2015)

Functional Maps
(08.06.2015)

Dr. Emanuele Rodolà

rodola@in.tum.de

Room 02.09.058, Informatik IX

Wrap-up

We introduced the Laplace-Beltrami operator Δ for regular surfaces and noted it is an **isometry invariant** quantity of the shape.

In the last few lectures, we used quantities derived from Δ in order to construct **descriptors** as well as **metrics** for our manifolds.

The general idea is being able to formulate and solve matching problems of the general form:

$$\min_C \text{dis}(C) + \alpha \text{dis}(C \times C)$$

The diagram shows two arrows originating from the terms in the equation above. One arrow points from $\min_C \text{dis}(C)$ to the text 'descriptor similarity'. The other arrow points from $\alpha \text{dis}(C \times C)$ to the text 'metric similarity'.

descriptor similarity **metric similarity**

Wrap-up

We alternatively denoted our intrinsic descriptors as **isometry-invariant Euclidean embeddings** of the surface. In particular, we considered the mappings:

$$p \mapsto \left(\frac{\varphi_1(p)}{\sqrt{\lambda_1}}, \frac{\varphi_2(p)}{\sqrt{\lambda_2}}, \frac{\varphi_3(p)}{\sqrt{\lambda_3}}, \dots \right) \quad \text{GPS}$$

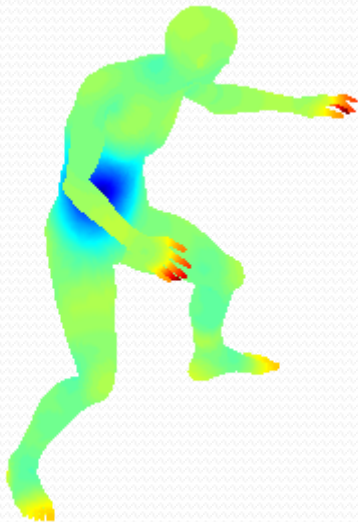
$$p \mapsto (e^{-\lambda_1 t} \phi_1(p), e^{-\lambda_2 t} \phi_2(p), e^{-\lambda_3 t} \phi_3(p) \dots) \quad \text{diffusion map}$$

$$p \mapsto \left(\sum_i e^{-\lambda_i t_1} \phi_i(p)^2, \sum_i e^{-\lambda_i t_2} \phi_i(p)^2, \sum_i e^{-\lambda_i t_3} \phi_i(p)^2 \dots \right) \quad \text{HKS}$$

Wrap-up

We introduced a new intrinsic metric known as **diffusion distance**:

$$d_t^2(x, y) = \int_S (k_t(x, z) - k_t(y, z))^2 dz = \sum_i e^{-2\lambda_i t} (\phi_i(x) - \phi_i(y))^2$$



$t = 1$



$t = 5$



$t = 10$

Wrap-up

The diffusion distance admits an equivalent expression in terms of the heat kernel:

$$d_t^2(x, y) = k_{2t}(x, x) + k_{2t}(y, y) - 2k_{2t}(x, y)$$

Integrating the heat kernel over all times, we came to the definition of the **commute-time kernel**, which equals the Green's function for the manifold:

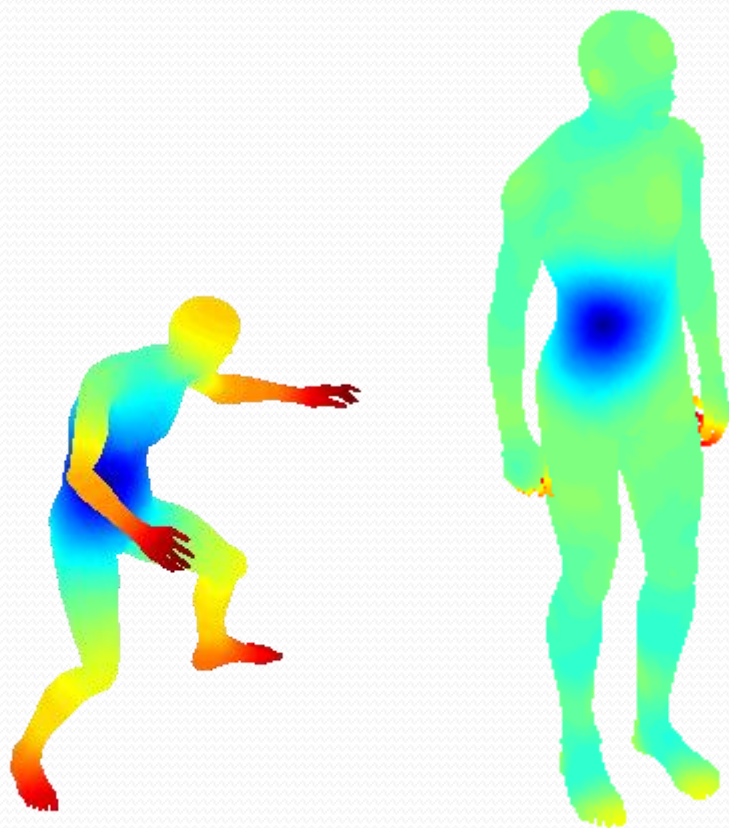
$$\int_0^\infty k_t(x, y) dt = \int_0^\infty \sum_k e^{-\lambda_k t} \phi_k(x) \phi_k(y) dt = \sum_k \frac{1}{\lambda_k} \phi_k(x) \phi_k(y) = g(x, y)$$

Hence we defined the **commute-time distance** as:

$$d^2(x, y) = g(x, x) + g(y, y) - 2g(x, y) = \sum_i \frac{1}{\lambda_i} (\phi_i(x) - \phi_i(y))^2$$

diffusion

$$d_t^2(x, y) = \sum_i e^{-2\lambda_i t} (\phi_i(x) - \phi_i(y))^2$$

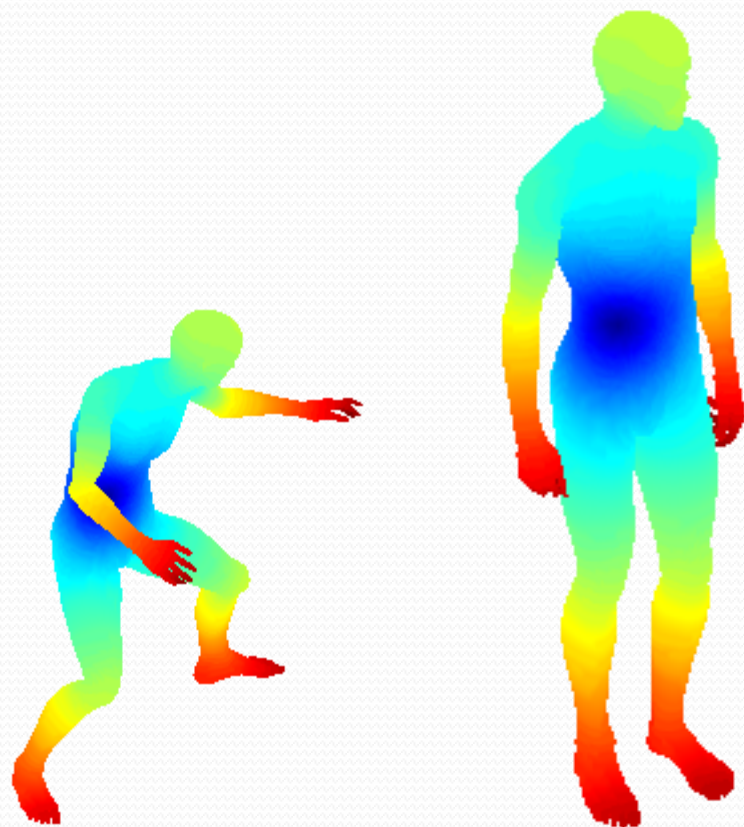


S

$\alpha S'$

commute-time

$$d^2(x, y) = \sum_i \frac{1}{\lambda_i} (\phi_i(x) - \phi_i(y))^2$$

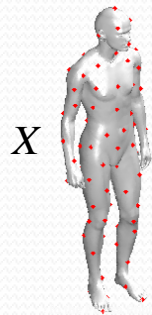


S

$\alpha S'$

Representing correspondences

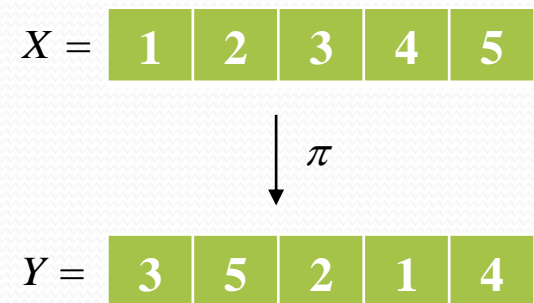
We have already seen that a correspondence can be represented by a matrix $R \in \{0,1\}^{n \times n}$



0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	0	0	1
0	0	1	0	0

Asking for a bijection corresponds to require R to be a *permutation matrix*.

In other words, we are optimizing over all permutations of $\{1, \dots, n\}$



Does not scale well
with the size of the
shapes

Representing correspondences

$$d_{\varphi}(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} \min_{\pi \in P_n} \max_{1 \leq i, j \leq n} |d_{\mathbf{X}}(x_i, x_j) - d_{\mathbf{Y}}(y_{\pi i}, y_{\pi j})|$$

We defined the cost matrix $\mathbf{C} \in \mathbf{R}^{n^2 \times n^2}$ such that:

$$C_{(i\ell)(jm)} = |d_{\mathbf{X}}(x_i, x_j) - d_{\mathbf{Y}}(y_{\ell}, y_m)|$$

(x_1, y_1)	0	13.5	23.4	104.6	7.64
(x_1, y_2)	13.5	0	13.52	11.2	71.1
(x_1, y_3)	23.4	13.52	0	0.22	23.44
\vdots	104.6	11.2	0.22	0	16.5
	7.64	71.1	23.44	16.5	0

$(x_1, y_1)(x_1, y_2)(x_1, y_3) \dots \ddots$

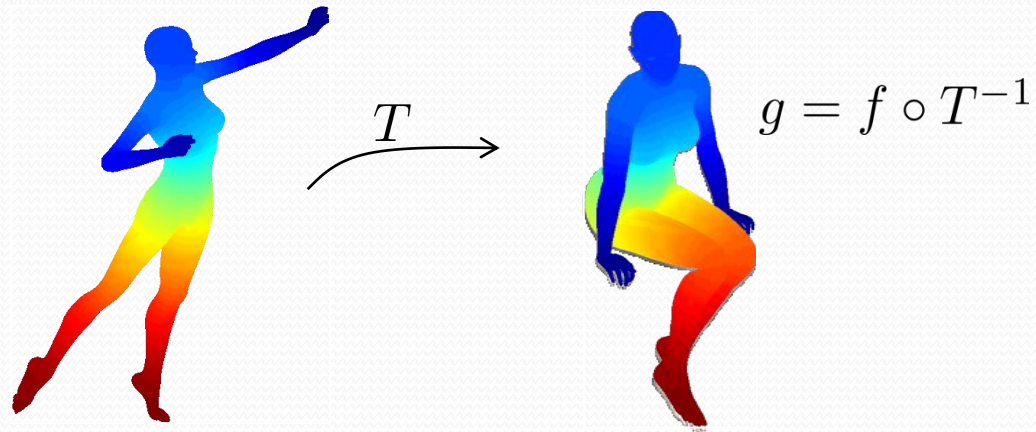
Quite big!

Can we do better than this?

A map between functions

Let $T : M \rightarrow N$ be a bijection between two regular surfaces M and N .

Given a scalar function $f : M \rightarrow \mathbb{R}$ on shape M , we can induce a function $g : N \rightarrow \mathbb{R}$ on the other shape by composition:



We can denote this transformation by a functional T_F , such that

$$T_F(f) = f \circ T^{-1}$$

We call T_F the **functional representation** of T .

Functional maps are linear

Note that T_F is a *linear* map between function spaces:

$$\begin{aligned} T_F(\alpha_1 f_1 + \alpha_2 f_2) &= (\alpha_1 f_1 + \alpha_2 f_2) \circ T^{-1} = \alpha_1 f_1 \circ T^{-1} + \alpha_2 f_2 \circ T^{-1} \\ &= \alpha_1 T_F(f_1) + \alpha_2 T_F(f_2) \end{aligned}$$

by linearity of the composition
operator \circ

This means that we can give a matrix representation for it, after we choose a basis for the two function spaces on M and N .

The key observation here is that, while T can in general be a very complex transformation between the two shapes, T_F always acts *linearly*.

It remains to see whether knowledge of T is equivalent to knowledge of T_F , and vice versa.

Recovering T from T_F

If we know T , we can obviously construct T_F by its definition $T_F(f) = f \circ T^{-1}$

Let's see if we can also do the contrary. That is, assume we know how to map functions to functions, and we want to be able to map points to points.

We first construct an indicator function $f : M \rightarrow \mathbb{R}$ on the first shape, such that $f(a) = 1$, and $f(x) = 0 \ \forall x \neq a \in M$.

Then if we call $g = T_F(f)$, it must be $g(y) = f \circ T^{-1}(y) = 0$ whenever $T^{-1}(y) \neq a$, and $g(y) = 1$ otherwise. Since T is a bijection, this happens only once, and $T(a)$ is the unique point $y \in N$ such that $g(y) = 1$. In other words, we have reconstructed T from T_F .

At this point it is still a bit unclear why we introduced functional mappings in the first place. Let's see!



Matrix notation (1/2)

Let $\{\phi_i^M\}$ be a basis for functions f on M , so that $f = \sum_i a_i \phi_i^M$. Then we can write:

$$T_F(f) = T_F\left(\sum_i a_i \phi_i^M\right) = \sum_i a_i T_F(\phi_i^M)$$

Similarly, if $\{\phi_j^N\}$ is a basis for functions on N , we can write:

$$T_F(\phi_i^M) = \sum_j c_{ji} \phi_j^N$$

Putting the two equations together, we get:

$$T_F(f) = \sum_i a_i \sum_j c_{ji} \phi_j^N = \sum_j \sum_i a_i c_{ji} \phi_j^N$$

Matrix notation (2/2)

$$\begin{aligned} T_F(f) &= \sum_j \sum_i a_i c_{ji} \phi_j^N \\ &= \sum_j b_j \phi_j^N \end{aligned}$$

So we can represent each function f on M by its coefficients a_i , and similarly function $T_F(f)$ on N by its coefficients b_j .

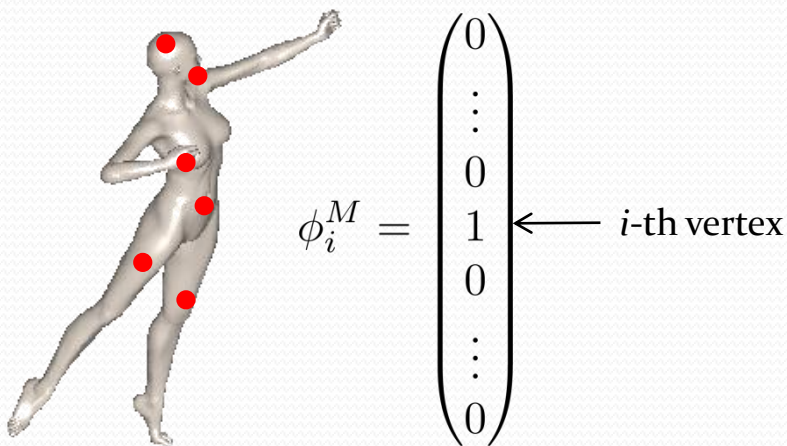
If the basis for functions f on M is orthogonal with respect to some inner product $\langle \cdot, \cdot \rangle$, then we can simply write $a_i = \langle f, \phi_i^M \rangle$. Similarly on N , we can write $c_{ji} = \langle T_F(\phi_i^M), \phi_j^N \rangle$.

Rewriting in matrix notation the equations above, we have:

$$T_F(\mathbf{a}) = \mathbf{b} = C\mathbf{a}$$

Choice of a basis

Up until now we have been assuming the presence of a **basis** for functions defined on the two shapes. The first possibility is to consider the *standard basis* on each shape, that is the set of indicator functions defined at each vertex:



$$f = \sum_i a_i \phi_i^M$$

$$c_{ji} = \langle T_F(\phi_i^M), \phi_j^N \rangle$$

The two terms in the inner product are indicator functions

$$C\mathbf{a} = \mathbf{b} \Rightarrow P\mathbf{a} = \mathbf{b}$$

permutation matrix

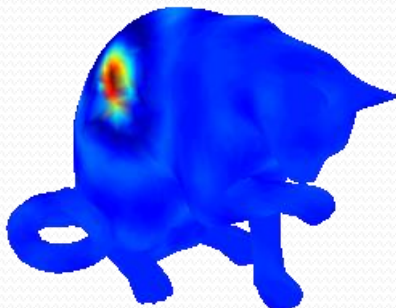
Choice of a basis

We already know another possibility!

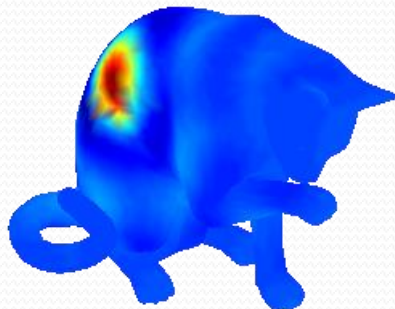
The eigenfunctions of the Laplace-Beltrami operator form an orthogonal basis (w.r.t. weighted inner product $\langle \cdot, \cdot \rangle_S$) for ℓ^2 functions on each shape.

In particular, we have seen that we can approximate:

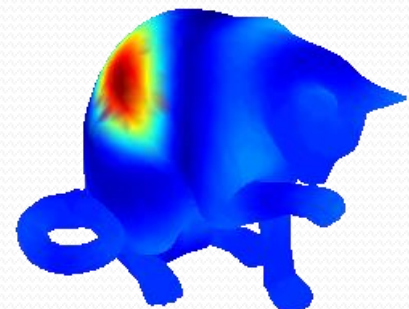
$$f = \sum_{i=0}^{\infty} a_i \phi_i^M \approx \sum_{i=0}^m a_i \phi_i^M$$



$m = 200$



$m = 100$



$m = 50$

Choice of a basis

This means that we can also approximate:

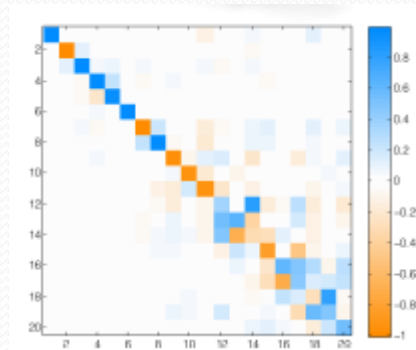
$$T_F(f) = \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} a_i c_{ji} \phi_j^N \approx \sum_{j=0}^m \sum_{i=0}^m a_i c_{ji} \phi_j^N$$

And then, going back to matrix notation we find out that we are reducing the size of matrix C quite a lot.

$$C\mathbf{a} = \mathbf{b}$$

Matrix C , which represents our correspondence, is a $m \times m$ matrix. *Its size does not depend on the size of the shapes!*

Typical values for m are 50 or 100



Moreover, matrices associated to correct correspondences tend to be sparse.

From P to C

Given a correspondence (bijection) $T : M \rightarrow N$ (in matrix notation, it can be written as a permutation matrix P), we can construct the associated functional map as follows:

$$T_F(f) = f \circ T^{-1} \implies C\mathbf{a} = \mathbf{b}$$

$$\text{We know it must be: } P\mathbf{e}_x = \mathbf{e}_{T(x)}$$

indicator function for vertex $x \in M$

indicator function for vertex $T(x) \in N$

$$\mathbf{a} = \Phi_M^{-1} \mathbf{e}_x \quad \text{indicator function in the } \{\phi_i^M\} \text{ basis}$$

$$C\mathbf{a} \quad \text{indicator function mapped to the } \{\phi_j^N\} \text{ basis}$$

$$\Phi_N C\mathbf{a} \quad \text{indicator function on the other shape}$$

$$\underbrace{\Phi_N C \Phi_M^{-1}}_P \mathbf{e}_x = \mathbf{e}_{T(x)}$$

$$\Phi_N C \Phi_M^{-1} = P \Rightarrow C = \Phi_N^{-1} P \Phi_M$$

Projecting onto the eigenbasis

Given a correspondence, we now know how to construct a functional map out of it. More than that, we have a way to *approximate* the correspondence and thus reduce the space taken by the representation.

This is simply done by choosing the number of eigenfunctions to use when performing the projection $\mathbf{a} = \Phi_M^{-1} \mathbf{f}$, that is $f = \sum_{i=0}^{\infty} a_i \phi_i^M \approx \sum_{i=0}^m a_i \phi_i^M$

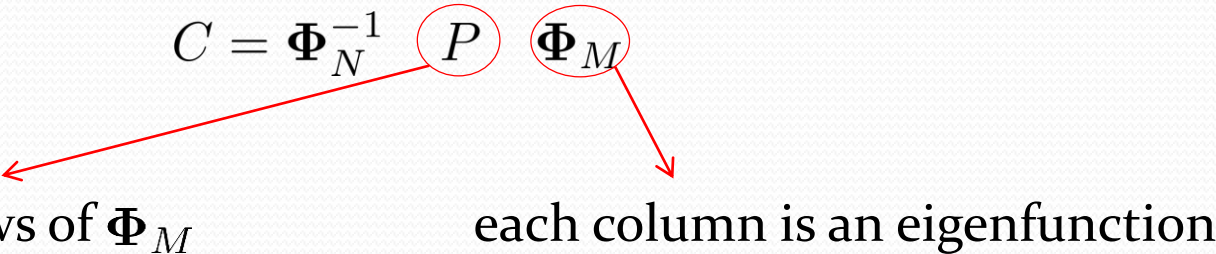
Note that in general we cannot say $\Phi_M^{-1} = \Phi_M^T$, because our eigenbasis is **not** orthogonal with respect to the **standard** inner product $\langle \cdot, \cdot \rangle$.

Indeed, our basis is orthogonal w.r.t. the mass-weighted inner product $\langle \cdot, \cdot \rangle_S$

Since $\langle \phi_i^M, f \rangle_S = \phi_i^T S \mathbf{f}$, we simply have $\Phi_M^{-1} = \Phi_M^T S$

Another look

Let us have another look at the formula relating the standard permutation matrix with the functional map:

$$C = \Phi_N^{-1} P \Phi_M$$


permutes the rows of Φ_M

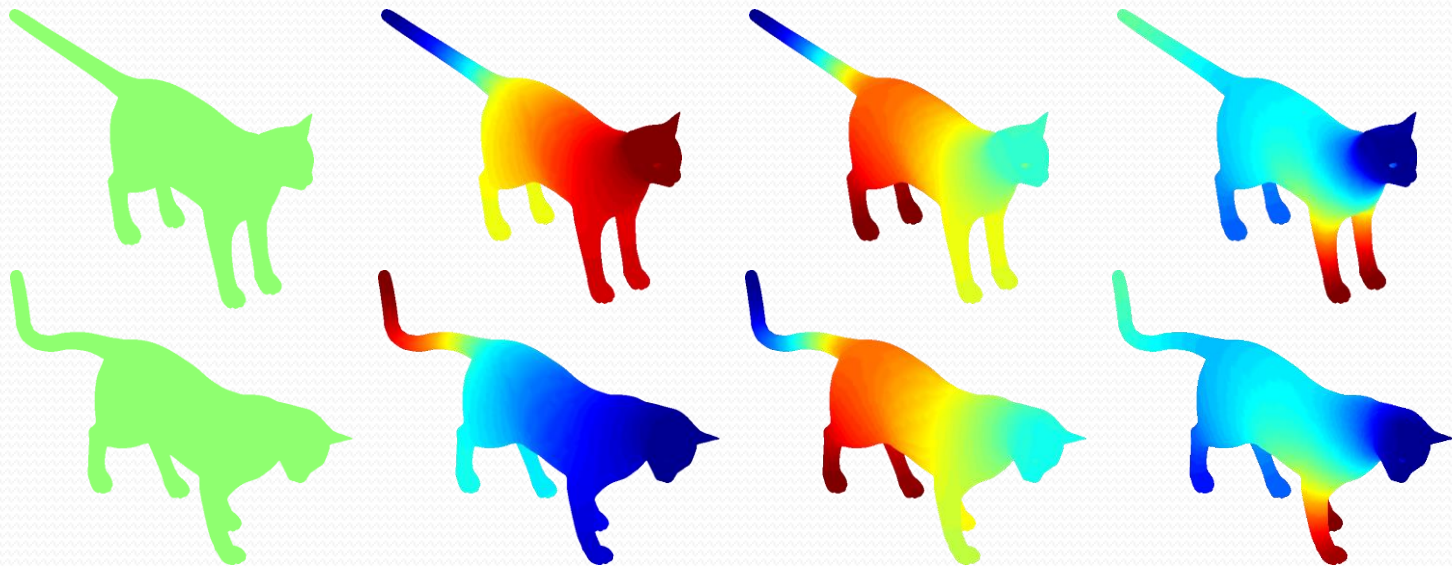
each column is an eigenfunction

Simply put, the functional map matrix **C** contains all the inner products between the eigenfunctions of the two shapes, after vertex ordering has been disambiguated by the known bijection P .

Then, if we use the standard bases in the two shapes, we simply get:

$$C = \mathbf{I}^{-1} P \mathbf{I} = P$$

Exact isometries: eigenbases



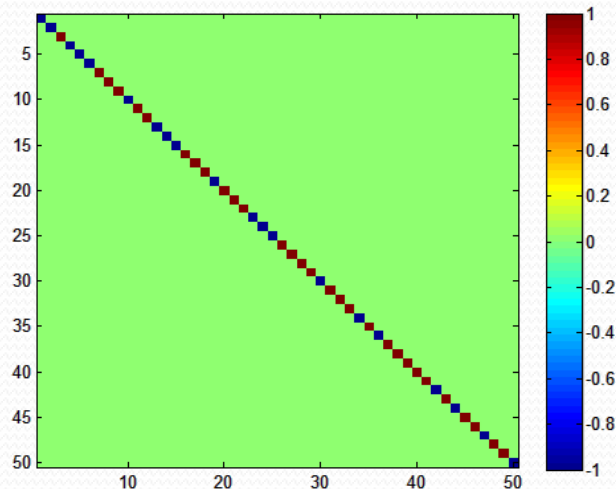
sign flip

Assume we are given two *exactly* isometric shapes. Then, we expect the eigenvectors of the two Laplacians to be identical **up to sign**, and **up to vertex ordering**.

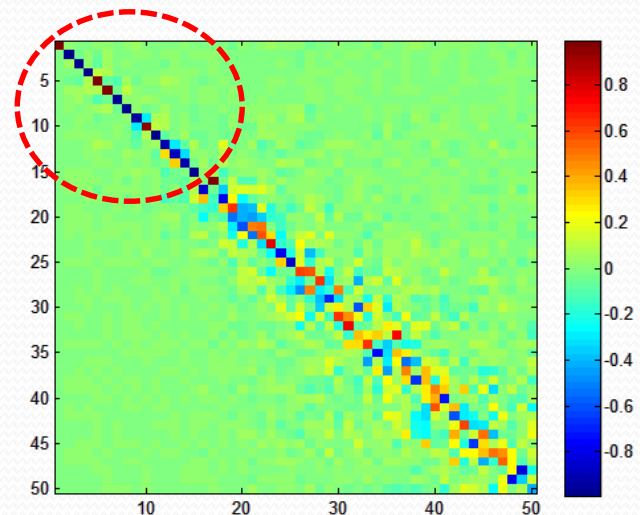
Exact isometries: functional map

$$C = \Phi_N^{-1} P \Phi_M$$

Since the permutation P is taking care of the vertex ordering, we are left with inner products that yield either 0, 1, -1.



exact isometry



near-isometry

Examples

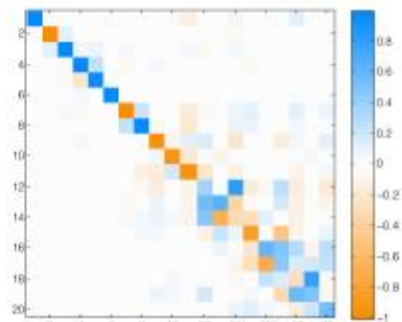
$$Ca = b$$

Fully encodes the original map T .
Note that this is a linear mapping!

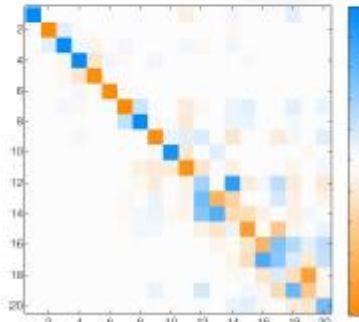
Note also that **not**
every linear map
corresponds to a
point-to-point
correspondence!



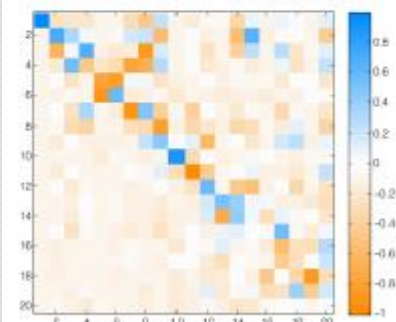
(a) source



(b) ground-truth map



(c) left to right map



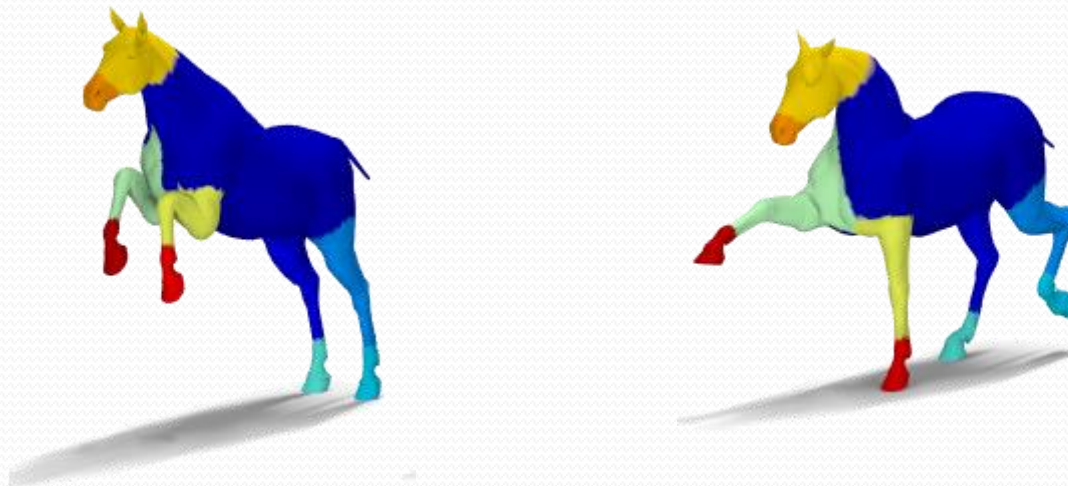
(d) head to tail map

Function transfer

Functional maps provide us a *compact* way to transfer functions between surfaces.

$$Ca = b$$

A simple example is segmentation transfer:



Cat algebra

Since we are now dealing with matrices (i.e. linear transformations), we can use all the tools from linear algebra to deal with our functional maps (e.g. sum or composition).



As a simple example, here we map the colors from a source shape (left) to a target shape (down), using an interpolation of “direct” and “symmetric” ground-truth maps according to $C = \alpha C_1 + (1 - \alpha)C_2$



(a) $\alpha = 0$



(b) $\alpha = 0.25$



(c) $\alpha = 0.5$



(d) $\alpha = 0.75$



(e) $\alpha = 1$

Imposing linear constraints

Interestingly, many common constraints that are used in shape matching problems also become *linear* in the functional map formulation.

Descriptor preservation

$$C\mathbf{a} = \mathbf{b}$$

function on M function on N

For instance, consider *curvature* or other descriptors.

If we are given a k -dimensional descriptor, we can just phrase k such equations, one for each dimension:

$$\begin{array}{l} C\mathbf{a}_1 = \mathbf{b}_1 \\ \vdots \\ C\mathbf{a}_k = \mathbf{b}_k \end{array} \quad \Rightarrow \quad C \begin{pmatrix} | & & | \\ \mathbf{a}_1 & \cdots & \mathbf{a}_k \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \mathbf{b}_1 & \cdots & \mathbf{b}_k \\ | & & | \end{pmatrix}$$

Landmark matches

Assume we know that $T(x) = y$. We can define two distance maps:

$d_x^M(x') = d_M(x, x')$, $d_y^N(y') = d_N(y, y')$, and then preserve the two functions as in the previous case.

Matching with functional maps

The functional maps representation can be employed to determine a correspondence between two shapes.

Using the ideas from the previous slide, we can just set up a linear system:

$$\begin{array}{ccc} C \begin{pmatrix} | & & | \\ \mathbf{a}_1 & \cdots & \mathbf{a}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \mathbf{b}_1 & \cdots & \mathbf{b}_n \\ | & & | \end{pmatrix} & \begin{array}{ll} n < m & \text{under-determined} \\ n = m & \text{full rank} \\ n > m & \text{over-determined} \end{array} \\ m \times m & m \times n & m \times n \end{array}$$

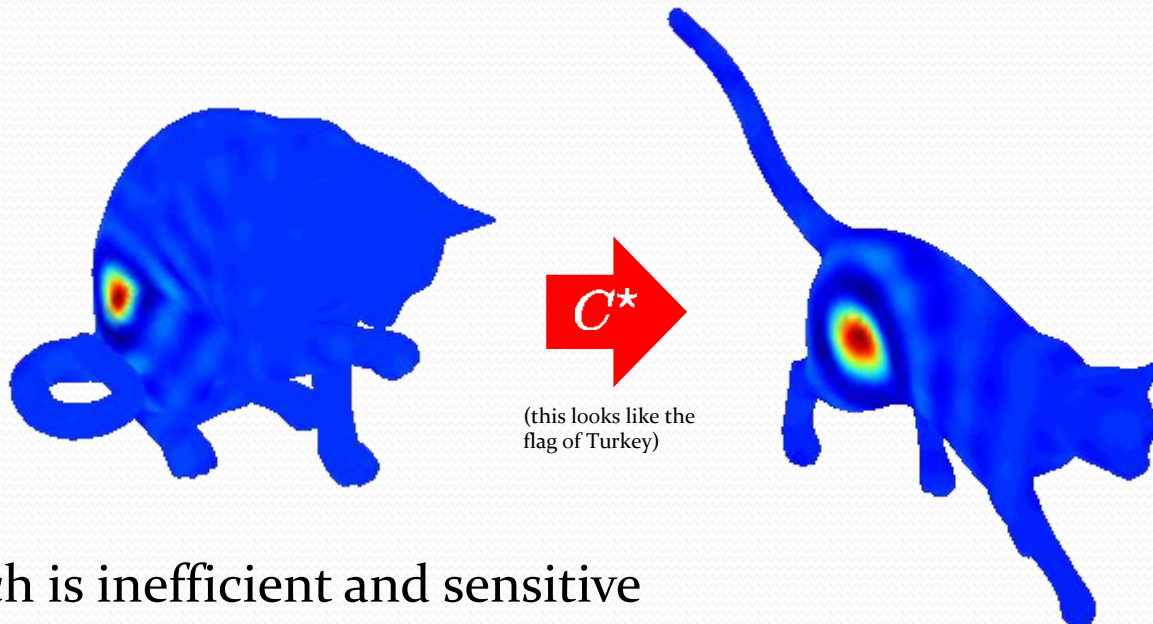
In the common case in which $n > m$, we can solve the resulting linear system in the least-squares sense:

$$CA = B \implies C^* = \arg \min_C \|CA - B\|^2$$

Convert C back to T

Once we have found an optimal functional map C^* , we may want to convert it back to a point-to-point correspondence.

Simplest idea: Use C^* to map indicator functions *at each point*.



This approach is inefficient and sensitive to truncation effects.

Convert C back to T

Observe that the delta function δ_x centered around $x \in M$, when represented in the eigenbasis $\{\phi_i^M\}$, has as coefficients the k -th column of matrix Φ^T , where k is the index of point $x \in M$.

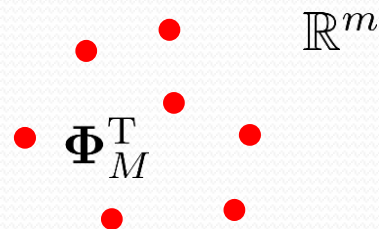
$$\Phi_M^T \mathbf{e}_k \in \mathbb{R}^m$$

representation of an indicator function in the eigenbasis

$$\Phi_M^T \in \mathbb{R}^{m \times n}$$

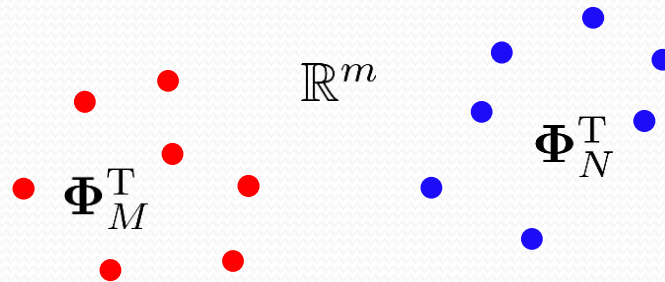
representation of *all* the indicator functions

Φ_M^T can then be regarded as a set of n points in \mathbb{R}^m

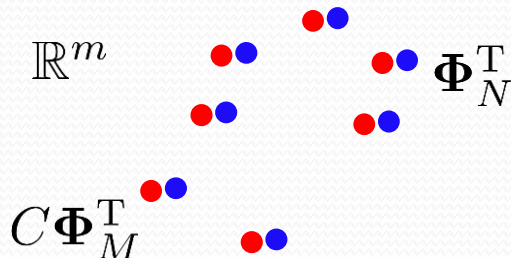


Convert C back to T

Clearly, the same can be said for the eigenfunctions of the second shape, Φ_N^T

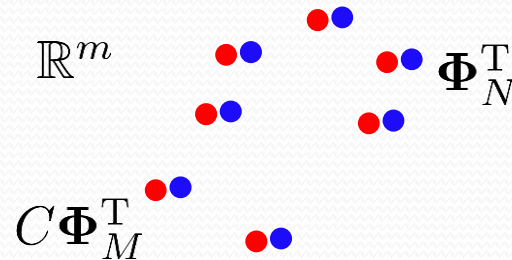


Applying the functional map C to the columns of Φ_M^T is equivalent to aligning the two point sets:



In fact, recall that by mapping indicator functions we were expecting to have $C\Phi_M^T \mathbf{e}_k = \mathbf{e}_h$

Convert C back to T



Thus, the problem of converting a functional map back to a point-wise correspondence can be phrased as a simple **nearest-neighbors** search in \mathbb{R}^m

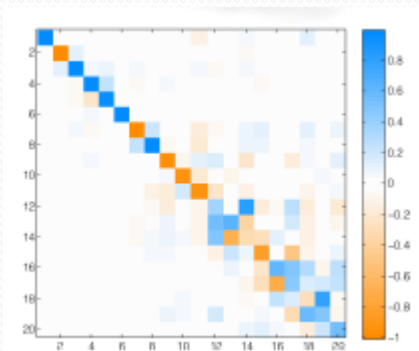
Specifically, the algorithm consists in seeking, for each column of Φ_N^T , the nearest column of $C\Phi_M^T$ in the L2 sense (the standard metric in \mathbb{R}^m).

This is especially convenient, since efficient data structures exist for this kind of problems (e.g. kd-trees, octrees, etc.).

Main issues

The functional maps representation provides a very convenient framework for shape matching, but most of its power derives from the availability of an appropriate basis.

Laplace-Beltrami eigenbasis: robust to nearly-isometric deformations only!



Recent approaches get state-of-the-art results by explicitly requiring C to have a diagonal structure.

Other approaches try to define an optimal basis given two shapes undergoing general deformations.

Suggested reading

- *Functional maps: A flexible representation of maps between surfaces.* Ovsjanikov et al. SIGGRAPH 2012.