Analysis of Three-Dimensional Shapes

(IN2238, TU München, Summer 2014)

Functional Maps
(12.06.2014)

Dr. Emanuele Rodolà
rodola@in.tum.de
Room 02.09.058, Informatik IX

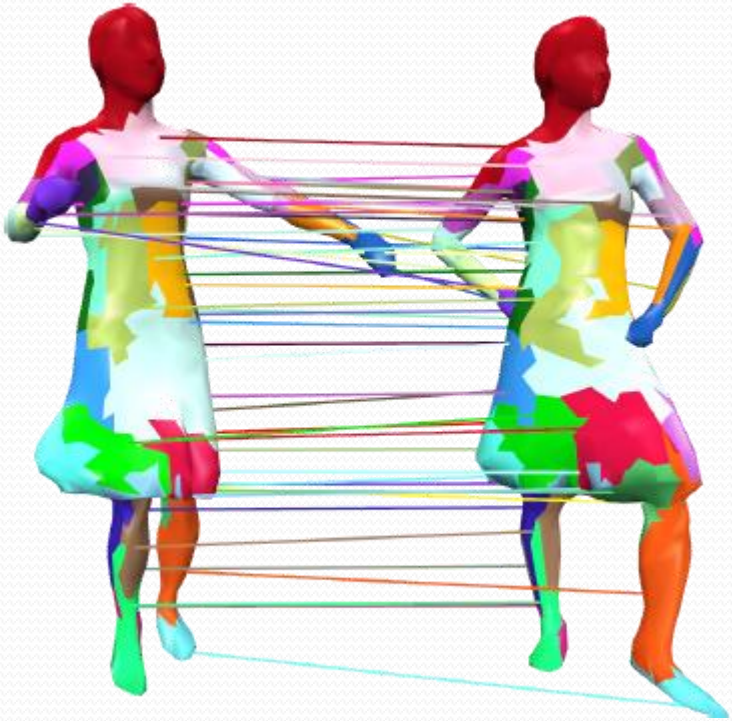
Seminar

«LP relaxation for elastic shape matching»

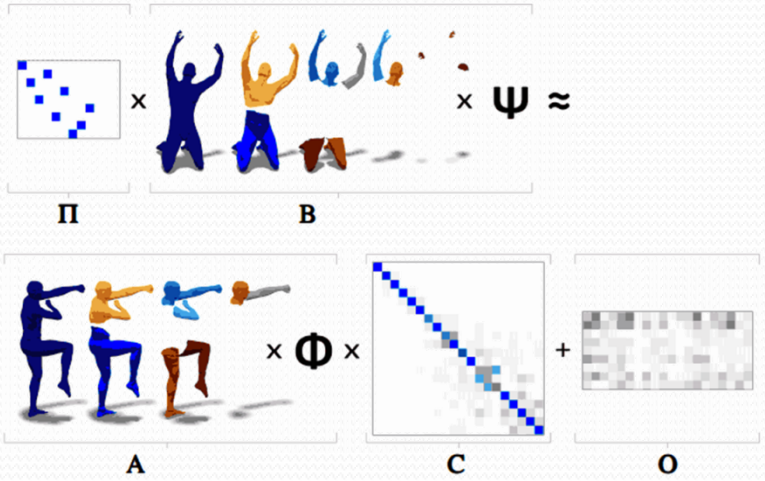
Fabian Stark

Wednesday, June 18th

14:00 Room 02.09.023



Seminar



«Sparse modeling of intrinsic correspondences»

Tobias Gurdan

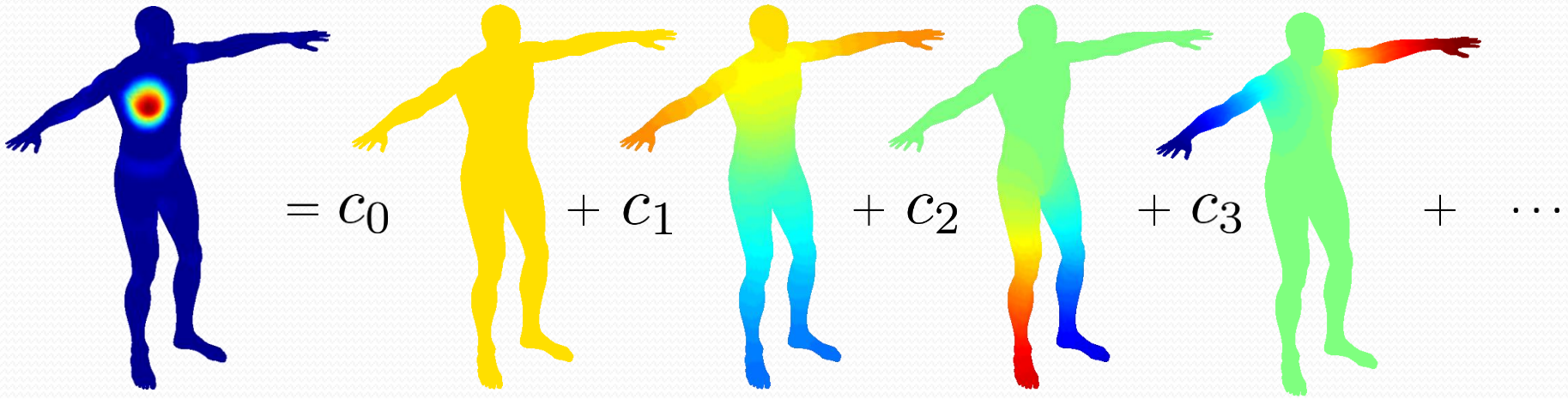
Wednesday, June 18th

14:45 Room 02.09.023

Wrap-up

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

In particular, we saw how its eigenfunctions provide a **basis** for square-integrable functions defined on the surface.

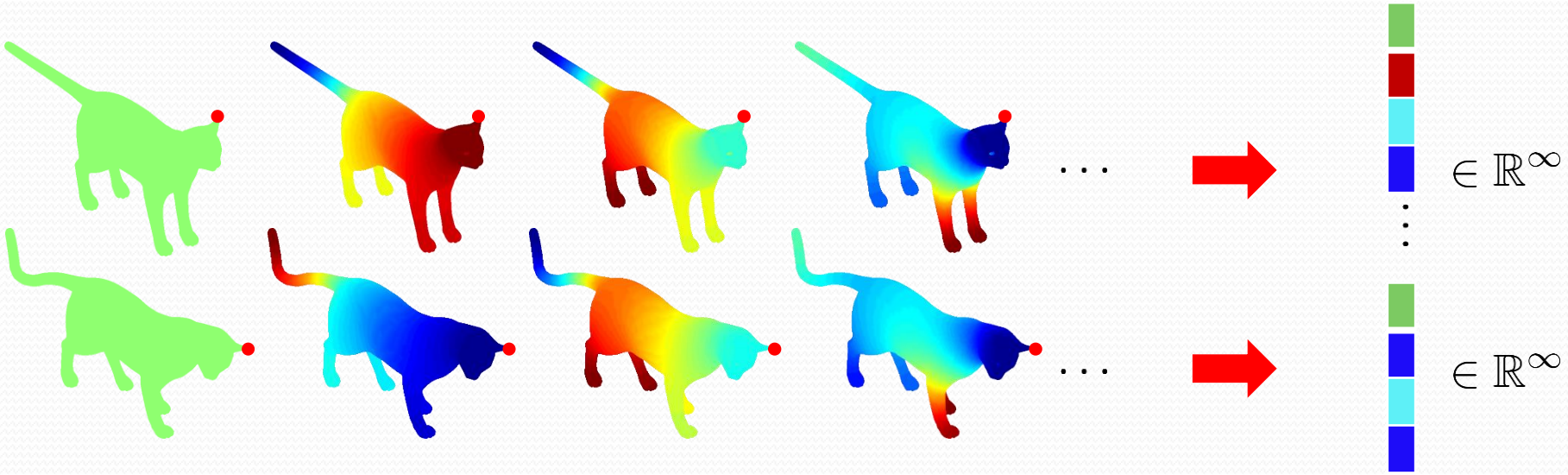


Wrap-up

Based on the intrinsic properties of this operator, we considered an isometry-invariant Euclidean **embedding** of the surface, namely via the mapping:

scale-invariant!

$$p \mapsto \left(\frac{\varphi_1(p)}{\sqrt{|\lambda_1|}}, \frac{\varphi_2(p)}{\sqrt{|\lambda_2|}}, \frac{\varphi_3(p)}{\sqrt{|\lambda_3|}}, \dots \right) \qquad p \in S$$



Wrap-up

We studied **heat diffusion** on surfaces, modeled according to the heat equation:

$$\frac{\partial u(x,t;u_0)}{\partial t} = \Delta u(x,t;u_0)$$

$$u(x,0) = u_0(x)$$

A solution to the heat equation is given by:

$$u(x,t;u_0) = \int_S k_t(x,y) u_0(y) dy$$

heat kernel

Wrap-up

Interesting properties of the heat kernel include:

Informative property
A surjective $T : S \to S'$ is an isometry iff $k_t^S(x, y) = k_t^{S'}(T(x), T(y))$

Restricting our attention to $k_t(x, x)$ still gives us a similar property under mild conditions (non-repeating eigenvalues).

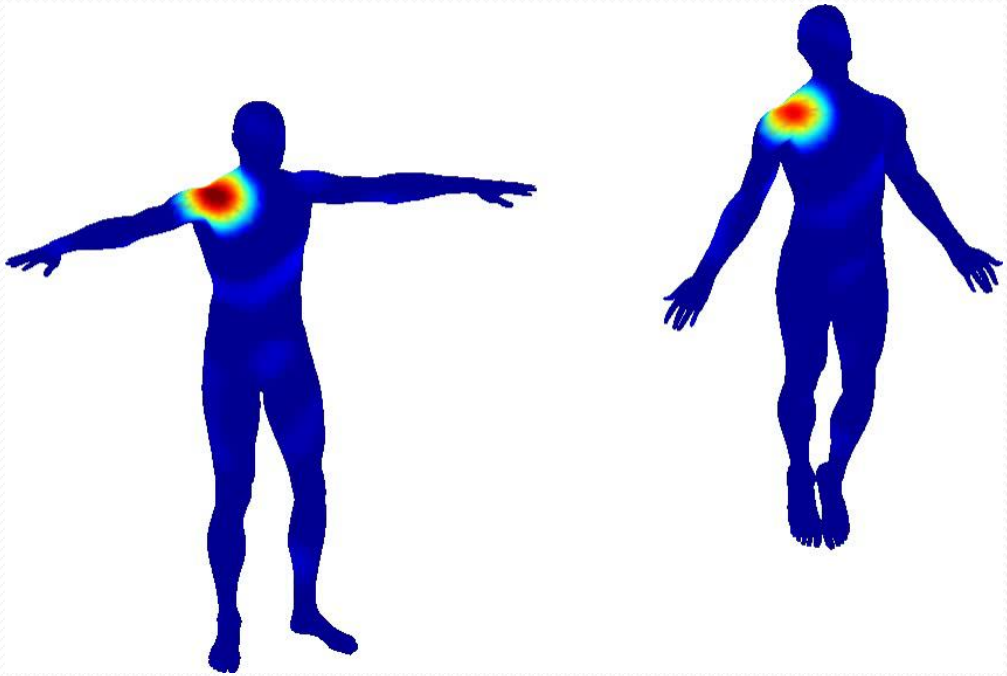
We then defined the **heat kernel signature** as the time-dependent function

$$HKS(x, t) = k_t(x, x) = u(t, x; \delta_x)$$

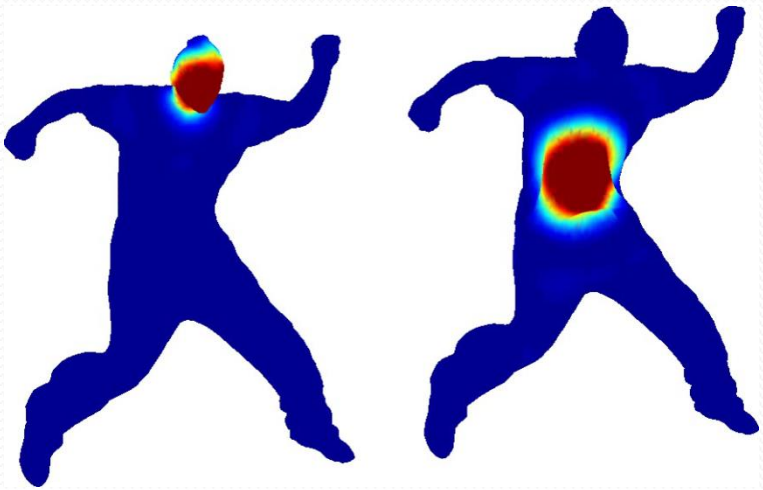
Wrap-up

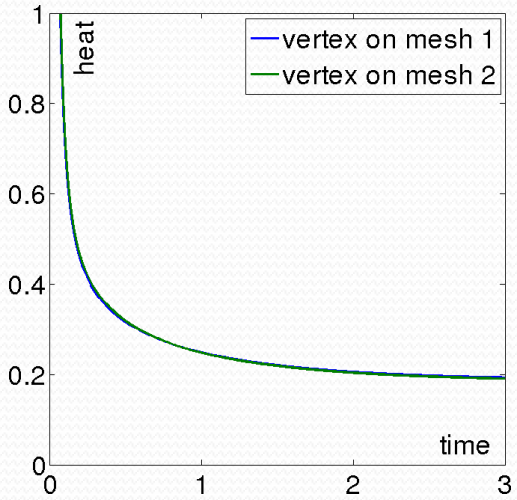
The heat kernel signature can be expressed in terms of the eigenfunctions of Δ as:

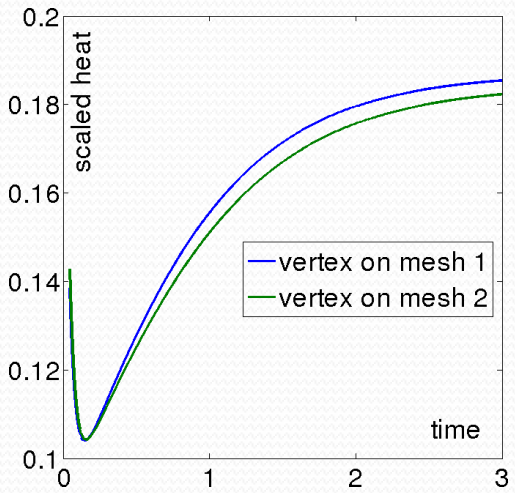
$$HKS(x,t) = \sum_{k=0}^{n-1} e^{\lambda_k t} \psi_k^2(x)$$

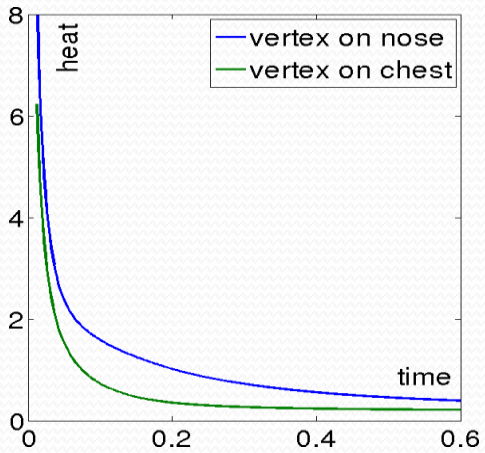


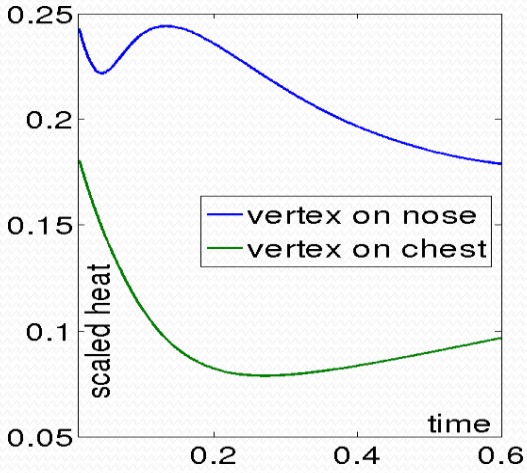
Wrap-up




heat
vertex on mesh 1
vertex on mesh 2
time
0
0.2
0.4
0.6
0.8
1
0
1
2
3


scaled heat
vertex on mesh 1
vertex on mesh 2
time
0.1
0.12
0.14
0.16
0.18
0.2
0
1
2
3

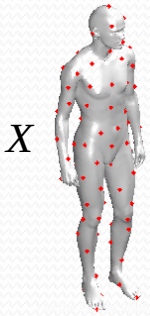

heat
vertex on nose
vertex on chest
time
0
2
4
6
8
0
0.2
0.4
0.6


scaled heat
vertex on nose
vertex on chest
time
0.05
0.1
0.15
0.2
0.25
0.2
0.4
0.6

Representing correspondences

We have already seen that a correspondence can be represented by a matrix

$R \in \{0,1\}^{n \times n}$

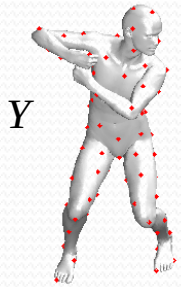


0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	0	0	1
0	0	1	0	0

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

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

Does not scale well with the size of the shapes



$X =$ | 1 | 2 | 3 | 4 | 5 |

π

$Y =$ | 3 | 5 | 2 | 1 | 4 |

Representing correspondences

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

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

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

(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)	$\cdots$	$\ddots$

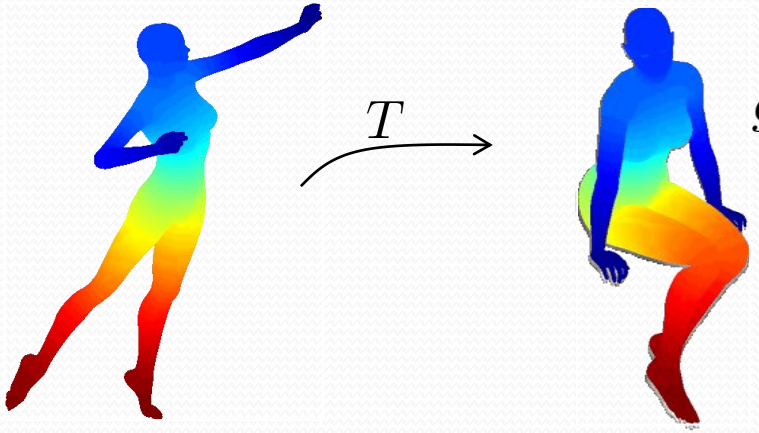
Quite big!

Can we do better than this?

A map between functions

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

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



$$g = f \circ T^{-1}$$

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:

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

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.

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 $f : M \to \mathbb{R}$ 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)

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

$$= \sum_j b_j \phi_j^N$$

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

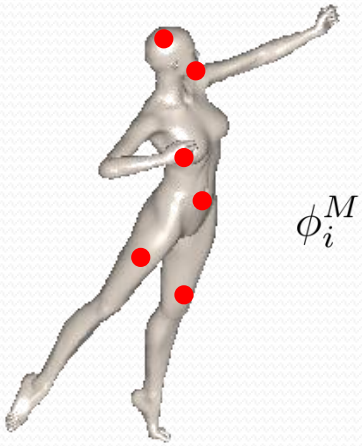
If the basis for functions *f* on *M* is orthogonal with respect to some inner product $\langle \cdot, \cdot \rangle$, then we can simply write $a_i = \langle f, \phi_i^M \rangle$. Similarly on *N*, we can write $c_{ji} = \langle T_F(\phi_i^M), \phi_j^M \rangle$.

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:



$$\phi_i^M = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i\text{-th vertex}$$

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

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

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

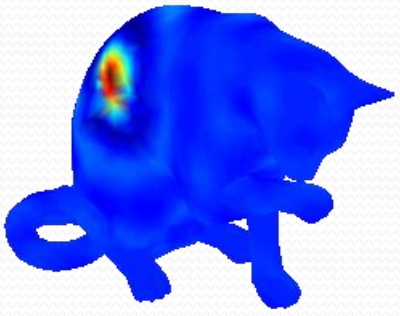
We already know another possibility!

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

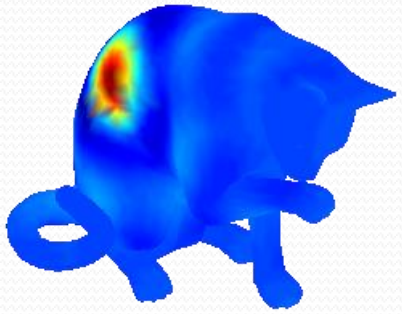
here S is the mass matrix

In particular, we have seen that we can approximate:

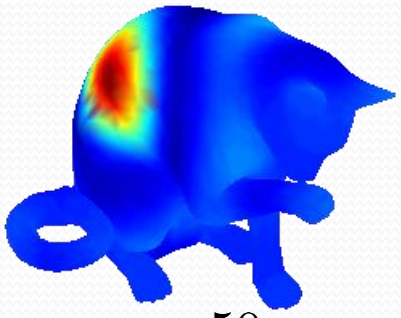
$$f = \sum_{i=0}^{\infty} a_i \phi_i^M \approx \sum_{i=0}^{m} a_i \phi_i^M$$



$m = 200$



$m = 100$



$m = 50$

Choice of a basis

This means that we can also approximate:

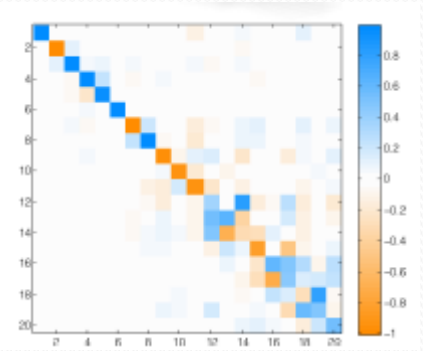
$$T_F(f) = \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} a_i c_{ji} \phi_j^N \approx \sum_{j=0}^{m} \sum_{i=0}^{m} a_i c_{ji} \phi_j^N$$

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



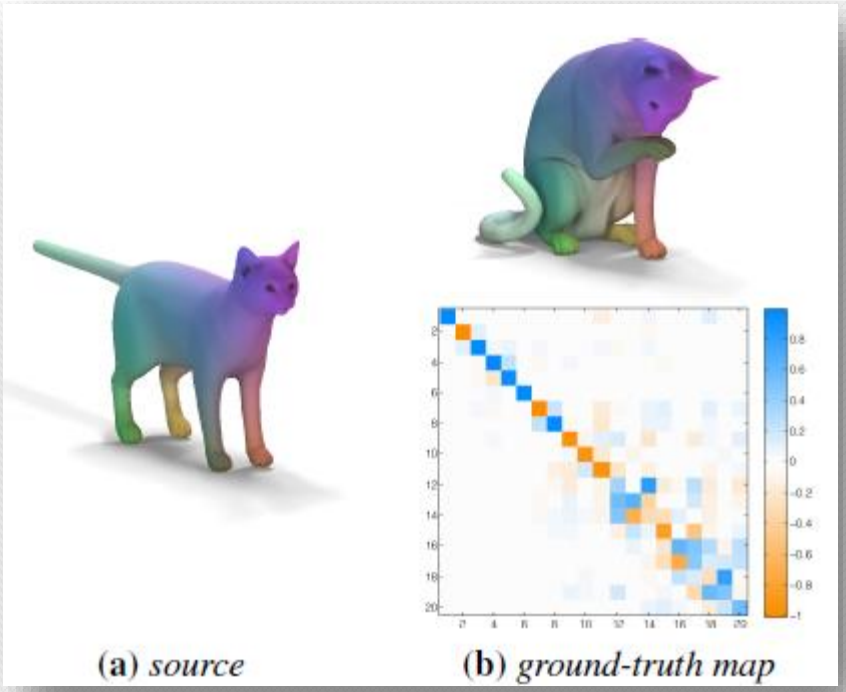
Moreover, matrices associated to correct correspondences tend to be sparse.

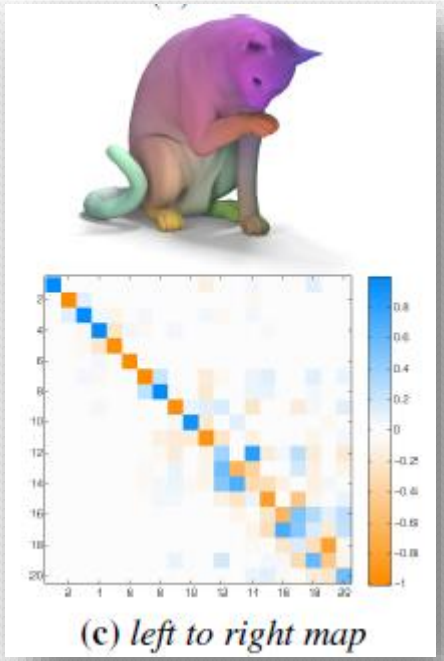
Examples

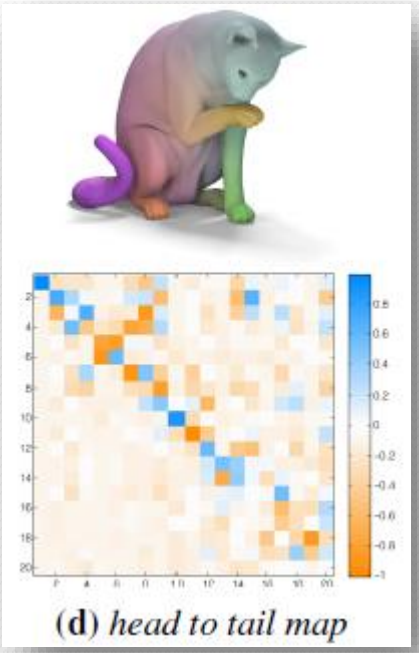
$$C\mathbf{a} = \mathbf{b}$$

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!







From *P* to *C*

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

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

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

indicator function for vertex $x \in M$

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

$\mathbf{a} = \mathbf{\Phi}_M^{-1}\mathbf{e}_x$ indicator function in the $\{\phi_i^M\}$ basis

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

$\mathbf{\Phi}_N C\mathbf{a}$ indicator function on the other shape

$$\underbrace{\mathbf{\Phi}_N C \mathbf{\Phi}_M^{-1}}_{P}\mathbf{e}_x = \mathbf{e}_{T(x)}$$

$$\mathbf{\Phi}_N C \mathbf{\Phi}_M^{-1} = P \Rightarrow C = \mathbf{\Phi}_N^{-1} P \mathbf{\Phi}_M$$

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 $\mathbf{a} = \mathbf{\Phi}_M^{-1}\mathbf{f}$, 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 $\mathbf{\Phi}_M^{-1} = \mathbf{\Phi}_M^{\mathrm{T}}$, 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 $\langle \cdot, \cdot \rangle_S$

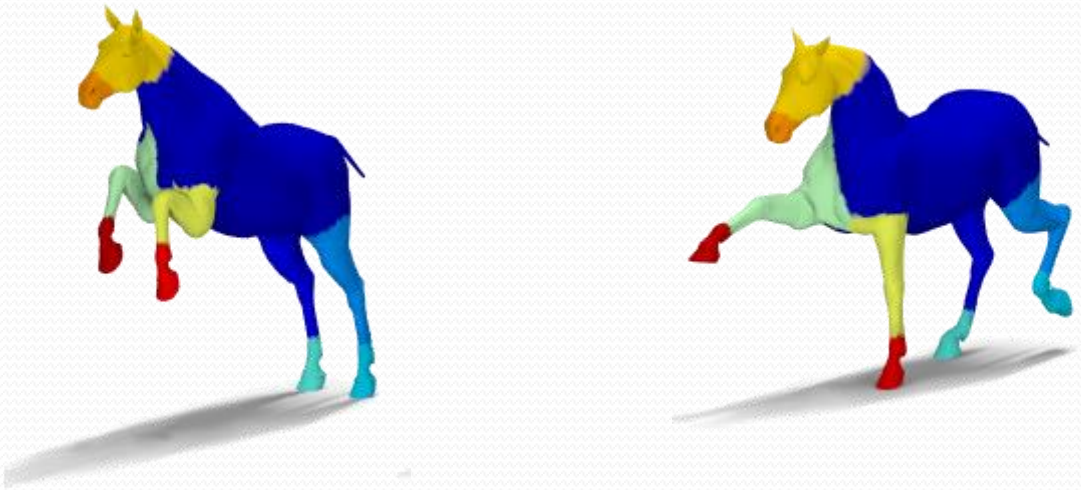
Since $\langle \phi_i^M, f \rangle_S = \phi_i^{\mathrm{T}} S \mathbf{f}$, we simply have $\mathbf{\Phi}_M^{-1} = \mathbf{\Phi}_M^{\mathrm{T}} S$

Function transfer

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

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

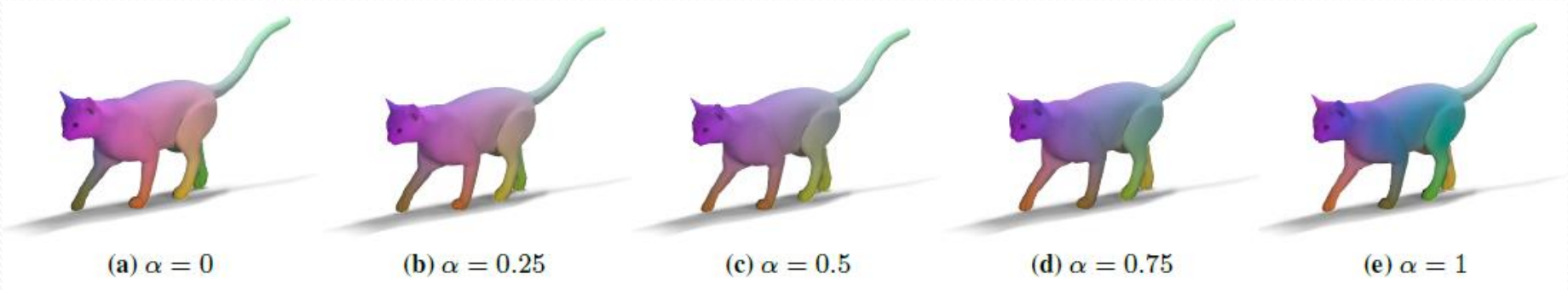
A simple example is segmentation transfer:



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:

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

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:

$$C \begin{pmatrix} | & & | \\ \mathbf{a}_1 & \cdots & \mathbf{a}_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \mathbf{b}_1 & \cdots & \mathbf{b}_n \\ | & & | \end{pmatrix}$$

$m \times m$ $m \times n$ $m \times n$

$n < m$ under-determined

$n = m$ full rank

$n > m$ over-determined

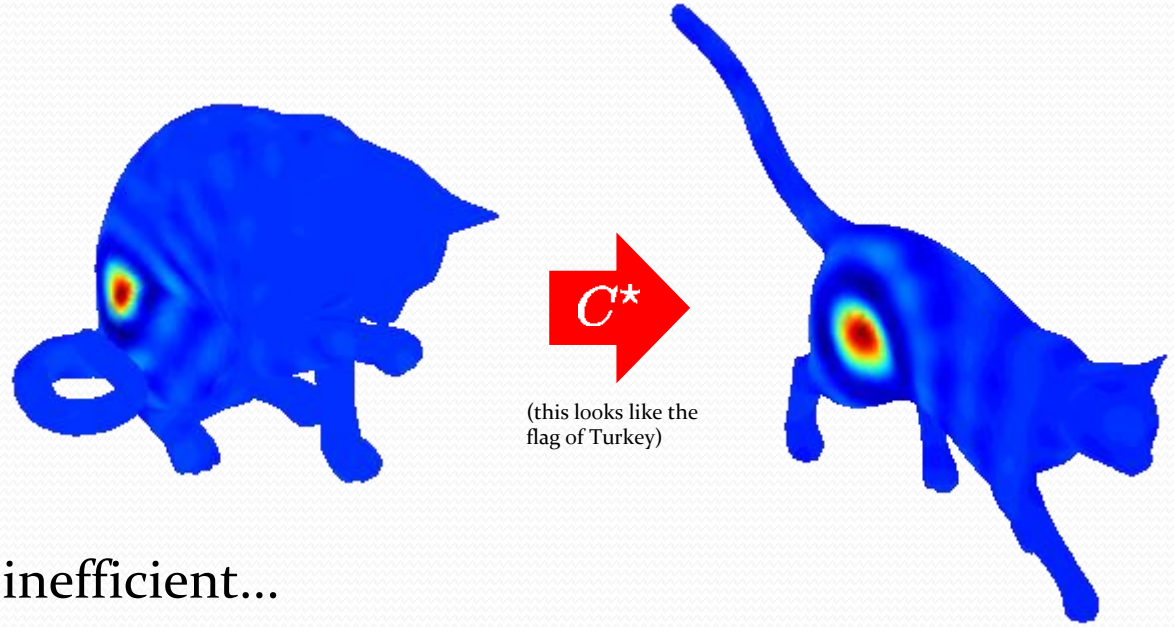
In the common case in which $n > m$, we can solve the resulting linear system in the least-squares sense:

$$CA = B \Longrightarrow C^\star = \arg\min_C \|CA - B\|^2$$

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 is quite inefficient…

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 $\mathbf{\Phi}^{\mathrm{T}} S$, where *k* is the index of point $x \in M$ and *S* is the mass matrix.

Thus, images for *all* the delta functions can be computed simply as $C\mathbf{\Phi}_M^{\mathrm{T}} S$

Let g_1 and g_2 be two functions on *N*, with spectral coefficients $\mathbf{b}_1$ and $\mathbf{b}_2$.

$$\|\mathbf{b}_1 - \mathbf{b}_2\|_2^2 = \int_N (g_1(y) - g_2(y))^2 dy$$ **Plancherel / Parseval's theorem**

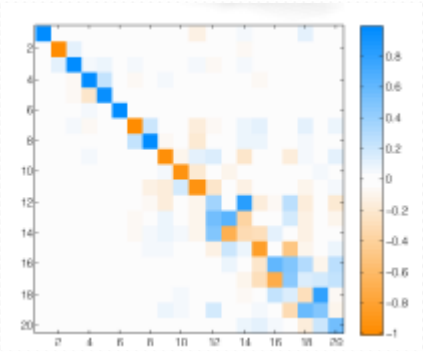
This means that for every point (i.e. column) of $C\mathbf{\Phi}_M^{\mathrm{T}} S$ we can look for its nearest neighbor in $\mathbf{\Phi}_N^{\mathrm{T}} S$. These are *m*-dimensional points.

This process can be accelerated using efficient search structures (kd-trees)!

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.

Tobias will present one of these approaches on Wednesday, don't miss it! ☺

Suggested reading

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