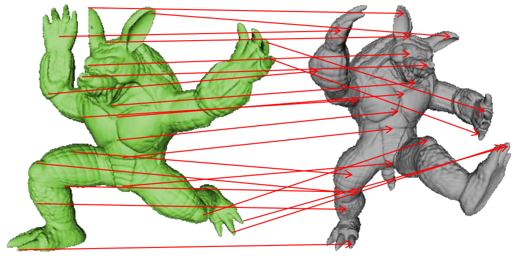
Analysis of 3D Shapes (IN2238)

Frank R. Schmidt
Matthias Vestner

Summer Semester 2016

Shape matching



Diffeomorphism

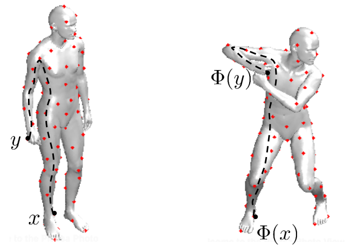
A mapping $\Phi : M \to N$ between two shapes M and N is a diffeomorphism if it is bijective and Φ and Φ^{-1} are C^1. If such a mapping exists the shapes are called diffeomorphic.

If two compact surfaces are diffeomorphic they have the same **Euler characteristic** (i.e. the same genus).

If M and N are diffeomorphic, there are coordinate maps (x_j, U_j) and (y_j, U_j) uch that $M = \cup x_j(U_j)$ and $N = \cup y_j(U_j)$.

Isometry

A mapping $\Phi : M \to N$ between two shapes M and N is an isometry if $d_M(x,y) = d_N(\Phi(x), \Phi(y))$ for all points $x, y \in M$. If such a maping exists M and N are called isometric.



$d_M(x,y)$ $d_N(\Phi(x), \Phi(y))$

Many shape matching approaches assume that the shapes to be matched are (nearly) isometric. The task then becomes to find the (almost-)isometry Φ.

Intrinsic symmetry

Most of the shapes we consider come with an intrinsic symmetry $S : M \to M$, such that

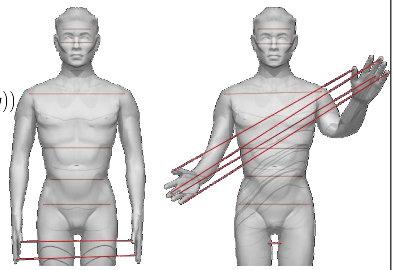
$$d_M(x,y) = d_M(S(x), S(y))$$

A consequence is that $\Phi : M \to N$ is not unique:

Φ isometry, S intrinsic symmetry:

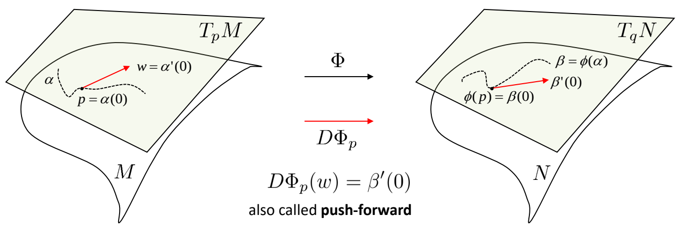
$$d_M(x,y) = d_M(S^{-1}(x), S^{-1}(y))$$
$$= d_M(\Phi \circ S^{-1}(x), \Phi \circ S^{-1}(y))$$

$\Rightarrow \Phi \circ S^{-1}$ is also an isometry.



Push forward

We can define the differential of a map between manifolds as we did with coordinate maps. Given a map $\Phi : M \to N$ the differential is a linear map $D\Phi_p : T_pM \to T_qN$ which maps tangent vectors at $p \in M$ to tangent vectors at $q = \Phi(p) \in N$.



$$D\Phi_p(w) = \beta'(0)$$

also called **push-forward**

Equivalent definition

A diffeomorphism $\Phi : M \to N$ is an isometry iff it preserves angles:

$$\langle v, w \rangle_{T_pM} = \langle D\Phi_p v, D\Phi_p w \rangle_{T_qN}$$

for all $v, w \in T_pM$ and $q = \Phi(p)$.

Proof (only one direction):
Let $c : [0,1] \to M$ be a shortest curve connecting $p \in M$ and $q \in M$:
$d(p,q) = L(c) = \int_0^1 \|\dot{c}(t)\| \, dt$

Then the curve $d : \Phi \circ c : [0,1] \to N$ has length

$$L(d) = \int_0^1 \left\| \frac{d}{dt}(\Phi \circ c(t)) \right\| dt = \int_0^1 \left\| D\Phi_{c(t)} \dot{c}(t) \right\| dt = \int_0^1 \| \dot{c}(t) \| \, dt = L(c)$$

Since there is no shorter curve connecting $\Phi(p)$ and $\Phi(q)$ (why?), it follows

$$d(p,q) = d(\Phi(p), \Phi(q))$$

Intrinsics

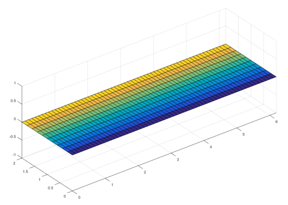
If M and N are given by coordinate maps (x_j, U_j) and (y_j, U_j) and $\Phi : M \to N$ is an isometry then $g^x_j(x^{-1}(p)) = g^y_j(y^{-1}(q))$ for all $q = \Phi(p)$.

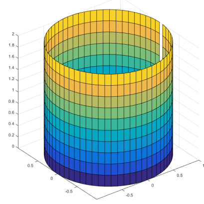
Thus **intrinsic** quantities are invariant under isometries:

- lenght of curves: $L(c) = L(\Phi(c))$
- angles between curves: $\langle \dot{c}_1, \dot{c}_2 \rangle_{T_qN} = \langle D\Phi\dot{c}_1, D\Phi\dot{c}_2 \rangle_{T_pM}$
- gradient operator: $D\Phi \nabla_M f(p) = \nabla_N (f \circ \Phi^{-1})(q)$
- divergence operator: $\text{div}_N(D\Phi \circ \vec{V} \circ \Phi^{-1}) = D\Phi \, \text{div}_M(\vec{V})$
- gaussian curvature: $\kappa(p) = \kappa(q)$
- ...

Example

$U = (0, 2\pi) \times (0, 1)$





$$x(u) = \begin{pmatrix} u_1 \\ 2u_2 \\ 0 \end{pmatrix} \qquad y(u) = \begin{pmatrix} \cos u_1 \\ \sin u_1 \\ 2u_2 \end{pmatrix}$$

Shapes as graphs

In practice we are working with surfaces discretized as triangular meshes (V, F).

Meshes can be seen as **undirected graphs** (V, E) with $E \subset V \times V$

For **adjacent** vertices we define the length function $L : E \to \mathbb{R}$ as the euclidean distance between the vertices $L(v_i, v_j) = \|v_i - v_j\|_2$

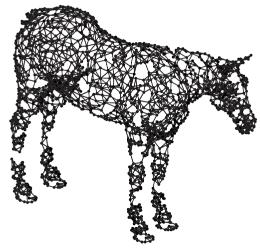
A **path** between $v_i, v_j \in V$ is an ordered set of connected edges

$\Gamma(v_i, v_j) = \{e_1, \ldots, e_k\} = \{(v_{i_1}, v_{i_2}), \ldots, (v_{i_k}, v_{i_{k+1}})\}$

with $v_{i_1} = v_i$ and $v_{i_{k+1}} = v_j$

The length of a path $\Gamma(v_i, v_j) = \{e_1, \ldots, e_k\}$ is then given by

$$L(\Gamma) = \sum_{n=1}^{k} L(e_n) = \sum_{n=1}^{k} L((v_{i_n}, v_{i_{n+1}}))$$



Distance in graph

Shortest path between $v_i, v_j \in V$

$$\Gamma^*(v_i, v_j) = \mathrm{argmin}_{\Gamma(v_i,v_j)} L(\Gamma(v_i, v_j))$$

Length metric in graph

$$d_L(v_i, v_j) = \min_{\Gamma(v_i,v_j)} L(\Gamma(v_i, v_j))$$

Approximates the geodesic distance on the shape.

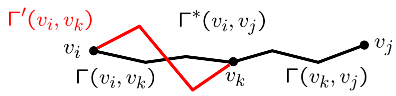
Shortest path problem: compute $\Gamma^*(v_i, v_j)$ and $d_L(v_i, v_j)$ between any $v_i, v_j \in V$.

Distance map problem: given a source point $v_0 \in V$, compute $d(v_i) = d_L(v_0, v_i)$.

Bellman's principle of optimality

Let $\Gamma^*(v_i, v_j)$ be the **shortest path** between $v_i, v_j \in V$ and $v_k \in \Gamma^*(v_i, v_j)$

Then $\Gamma(v_i, v_k)$ and $\Gamma(v_k, v_j)$ are **shortest sub-paths** between v_i, v_k and v_k, v_j.



Suppose there exists a **shorter** path $\Gamma'(v_i, v_k)$. Then

$$\begin{aligned} L(\Gamma'(v_i, v_j)) &= L(\Gamma'(v_i, v_k)) + L(\Gamma(v_k, v_j)) \\ &< L(\Gamma(v_i, v_k)) + L(\Gamma(v_k, v_j)) = L(\Gamma^*(v_i, v_j)) \end{aligned}$$

This is a **contradiction** to Γ^* being the shortest path.

Dynamic programming

How to compute the shortest path between source v_0 and v_i?

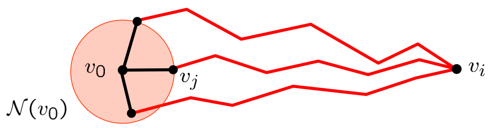
Bellman principle: there exists $v_j \in \mathcal{N}(v_0)$ such that

$$d_L(v_0, v_i) = L(v_0, v_j) + d_L(v_j, v_i)$$

v_j has to minimize the path length

$$d_L(v_0, v_i) = \min_{v_j \in \mathcal{N}(v_i)} \{L(v_0, v_j) + d_L(v_j, v_i)\}$$

Recursive **dynamic programming equation**



Dijkstras algorithm

Initialize $d(v_0) = 0$ and $d(v_i) = \infty$ for the rest of the graph.

Initialize **queue of unprocessed vertices** $Q = V$.

While $Q \neq \emptyset$

Find vertex with smallest value of d: $v = \mathrm{argmin}_{v \in Q}\, d(v)$

For each **unprocecced adjacent vertex** $v' \in \mathcal{N}(v) \cap Q$

$d(v') = \min\{d(v'), d(v) + L(v, v')\}$

Remove v from Q

Return **distance map** $d(v_i)$.

Dijkstra - complexity

While there are still unprocessed vertices

Find and remove minimum

For each **unprocecced adjacent vertex**

perform update

Every vertex is processed exactly once: $n = |V|$ outer iterations.

Naive **minimum extraction complexity:** $O(n)$

Can be reduced to $O(\log n)$ using **heap data structure**

Updating adjacent vertices is in general $O(|\mathcal{N}|) = O(|E|)$

In our case, graph is **sparsely connected**, update in $O(1)$

Total complexity: $O(n \log n)$

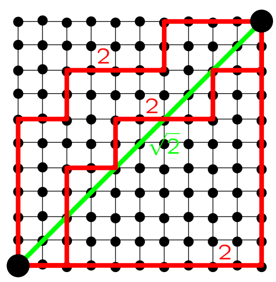
Trouble (in the neighborhood)

Grid with **4-neighbor** connecitivity

True euclidean distance: $d_{\mathbb{R}^2} = \sqrt{2}$

Shortest path in graph (not unique): $d_L = 2$

Increasing sampling density does not help



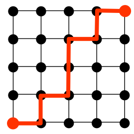
Consistent approximation

How to approximate the metric **consistently**?

$$\lim_{h \to 0} d_L = d_{\mathbb{R}^2}$$

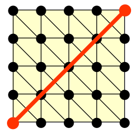
Solution 1

Stick to **graph** representation
Change connectivity and sampling
Under certain conditions consistency is guaranteed



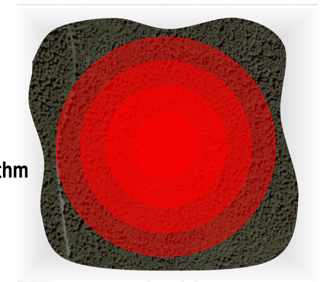
Solution 2

Stick to given **sampling** and **connectivity**
Compute distance map **on a surface** in some representation (e.g. **mesh**)
Requires a new algorithm



Fast Marching methods (FMM)

- A family of methods
- finds the **distance map**
- Simulates **wavefront propagation** from a source set
- A continiuius variant of **Dijkstra's algorithm**
- **Consistent approximation** of geodesic distance on surface
- Our picture: Fire *marching* through a forest



Dijkstras algorithm

Initialize $d(v_0) = 0$ and $d(v_i) = \infty$ for the rest of the graph.

Initialize **queue of unprocessed vertices** $Q = V$.

While $Q \neq \emptyset$

Find vertex with smallest value of d: $v = \text{argmin}_{v \in Q} d(v)$

For each **unprocecced adjacent vertex** $v' \in \mathcal{N}(v) \cap Q$

$$d(v') = \min\{d(v'), d(v) + L(v, v')\}$$

Remove v from Q

Return **distance map** $d(v_i)$.

Fast marching algorithm

Initialize $d(v_0) = 0$ and mark it as **black**.
Initialize $d(v_i) = \infty$ for the rest of the vertices and mark them as **green**.
Initialize **queue** of **red** vertices $Q = \emptyset$.

While there are still **green** vertices

Mark **green** neighbors of **black** vertices as **red** and assign edgelength to d.

For each **red vertex** $v \in Q$

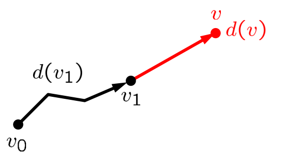
For each **triangle** sharing the vertex v

Update v from the triangle.

Mark v with **minimum** value of d as **black** (remove from Q)

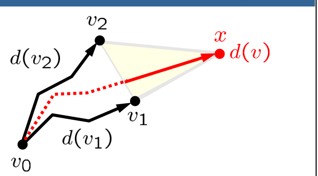
Return **distance map** $d(v_i)$.

Update step



Dijkstra update

- Vertex v updated from adjecent vertex v_1
- distance $d(v)$ computed from $d(v_1)$
- Path restricted to **graph edges**

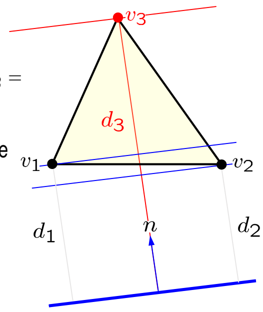


Fast Marching update

- Vertex v updated from triangle (v_1, v_2, v)
- distance $d(v)$ computed from $d(v_1)$ and $d(v_2)$
- Path can pass on **mesh faces**

Update step

- Vertex v_3 updated from triangle (v_1, v_2, v_3)
- distance $d(v_3)$ computed from $d_1 = d(v_1)$ and $d_2 = d(v_2)$
- model wave front propagating from planar source
- front hits v_1 at time d_1 and v_2 at time d_2
- **when does the front hit** v_3**?**



Planar source

$\langle v, n \rangle + p = 0$

Update step

We begin with a simplified setup: $v_1, v_2, v_3 \in \mathbb{R}^2$, $v_3 = 0$

- d_3 is given by the **point-to-plane distance**

$$d_3 = \langle v_3, n \rangle + p = p$$

- We can solve for n and p using the known distances d_1 and d_2

$$\langle v_1, n \rangle + p = d_1$$
$$\langle v_2, n \rangle + p = d_2$$

- With $V = (v_1, v_2)$, $d = (d_1, d_2)$ and $1 = (1, 1)^T$:

$$V^T n + p \cdot 1 = d \Leftrightarrow n = V^{-T}(d - p \cdot 1)$$

- Using $\|n\| = 1$ and substituting $Q = (V^T V)^{-1}$ we obtain

$$d_3^2 \cdot 1^T Q 1 - 2 d_3 \cdot 1^T Q d + d^T Q d - 1 = 0$$

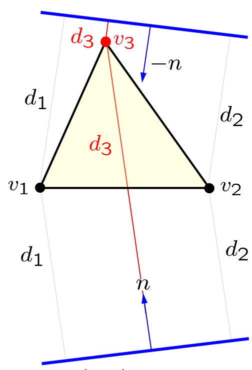
Causality condition

Quadratic equation has two solutions.

Causality: Front can only move forward in time.

$$d_3 > d_1, d_2$$
$$d_3 \cdot 1 > V^T n + p \cdot 1$$
$$d_3 \cdot 1 > V^T n + d_3 \cdot 1$$
$$0 > V^T n \quad \text{(Component wise)}$$

$\langle v, -n \rangle + p = 0$



$\langle v, n \rangle + p = 0$

Causality condition

Quadratic equation has two solutions.

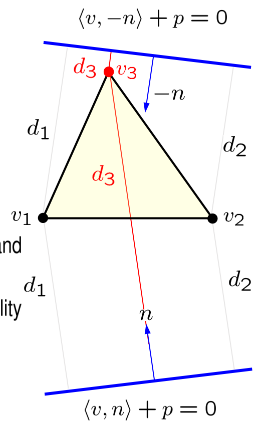
Causality: Front can only move forward in time.

$$d_3 > d_1, d_2$$
$$d_3 \cdot 1 > V^\top n + p \cdot 1$$
$$d_3 \cdot 1 > V^\top n + d_3 \cdot 1$$
$$0 > V^\top n \quad \text{(Component wise)}$$

n has to form obtuse angles with both edges (v_3, v_1) and (v_2, v_1).

Smallest solution of quadratic equation violates causality $\Rightarrow$ **discard**

If largest solution satisfies causality $\Rightarrow$ **done**



Monotonicity condition

d_3 must increase when d_1 or d_2 increase:

$$\nabla_d d_3 = \left(\frac{\partial d_3}{\partial d_1}, \frac{\partial d_3}{\partial d_2}\right)^\top = \frac{Q(d - d_3 \cdot 1)}{1^T Q(d - d_3 \cdot 1)} > 0$$

Substitute $n = V^{-T}(d - d_3 \cdot 1)$

$$\nabla_d d_3 = \frac{QV^T n}{1^T QV^T n} > 0$$

Monotonicity satisfied when both coordinates of $QV^T n$ have the **same sign.**

Q is **positive definit**
Causality condition: $V^T n < 0$
} At least one coordinate of $QV^T n$ is negative

Monotonicity condition: $QV^T n < 0$

One sided update

Since $Q = (V^T V)^{-1}$ we have $QV^T V = I$

Rows of QV^T are orthogonal to triangle edges

Monotonicity condition: $QV^T n < 0$

Interpretation:

n must form obtuse angles with normals to triangle edges

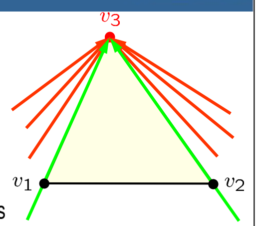
n must come from within the triangle

One sided update:

If n comes from outside the triangle, project it to one of the edges

Update reduces to Dikstra update:

$$d_3 = d_1 + \|v_1 - v_3\|_2 \quad \text{or} \quad d_3 = d_2 + \|v_2 - v_3\|_2$$



Fast marching update

Solve quadratic equation and select algest solution

$$d_3^2 \cdot 1^\top Q 1 - 2 d_3 \cdot 1^\top Q d + d^\top Q d - 1 = 0$$

Compute propagation direction

$$n = V^{-\top}(d - d_3 \cdot 1)$$

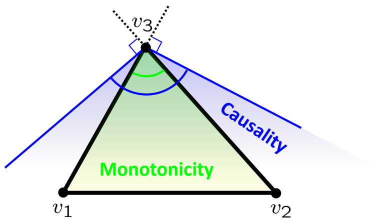
If monotonicity condition $QV^T n < 0$ is violated

$$d_3 = \min\{d_1 + \|x_1 - x_3\|_2, d_2 + \|x_2 - x_3\|_2\}$$

Set

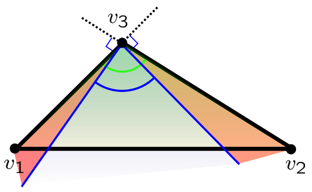
$$d(x_3) = \min\{d(x_3), d_3\}$$

Causality and monotonicity



Acute triangle

All directions in the triangle satisfy causality and monotonicity conditions.



Obtuse triangle

Some directions in the triangle violate causality condition!

FMM outlook

Inconsistent solution of mesh contains **obtuse** triangles

Remeshing is costly

Solution: split obtuse triangles by adding **virtual connections** to **non-adjecent** vertices

Done as **pre-processing step** in $O(n)$

Homework

Update step has to be slightly modified for general triangles with vertices in $\mathbb{R}^3$ (Translation, Rotation).

The planar source model can be replaced by a point source model.

Example