

# DISCUSSION LOG

Sikang Yan

University of Kaiserslautern

*yan@rhrk.uni-kl.de*

May 14, 2019

# THEORY

In our cloth simulation, we follow the procedure written by Rohmer et al.. We consider to begin with the *Defomation gradient*  $\mathbf{F}$ , which is defined by

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}, \quad (1)$$

where the  $\mathbf{x}$  denotes the deformed vector and the  $\mathbf{X}$  denotes the reference vector.

Since we have no further information about the mapping from  $\mathbf{X}$  to  $\mathbf{x}$ , we use the vector  $(\mathbf{u}_1, \mathbf{u}_2)$  and  $(\overline{\mathbf{u}_1}, \overline{\mathbf{u}_2})$  to approximate the Defomation gradient, which is defined as

$$\mathbf{F} = [\mathbf{u}_1, \mathbf{u}_2] [\overline{\mathbf{u}_1}, \overline{\mathbf{u}_2}]^{-1}, \quad (2)$$

Attention should be paid especially:

- eq.(2) characrizie only the 2D deformation of each triangle.
- in Rohmer et al.  $\mathbf{F}$  is symbolised as  $\mathbf{T}$ .

We provide here our code preceed with concrete data:

- $\text{faces}(i, j)$  is the  $j$ th vertex of the  $i$ th triangle, here we choose the  $\text{face}(0, 0)$ ,  $\text{face}(0, 1)$ ,  $\text{face}(0, 2)$  as examples and calculate the  $\text{Vec}T$  and  $\text{Vec}R$  for  $\text{face}(0, 0)$ .

## *cloth\_vec*

$$VecT = el_1 VertT_1 - el_1 VertT_2; el_1 VertT_1 - el_1 VertT_3 \quad (3)$$

$$VecR = el_1 VertR_1 - el_1 VertR_2; el_1 VertR_1 - el_1 VertR_3 \quad (4)$$

## *cloth\_vec*

$$VecT = [0.771842 \ -0.0144887 \ 6.39045] - [0.780121 \ -0.0186188 \ 6.37318] \\ (5)$$

$$[0.771842 \ -0.0144887 \ 6.39045] - [0.737177 \ -0.00912791 \ 6.39292] \\ (6)$$

$$VecR = [0.759919 \ -0.015194 \ 6.38401] - [0.767822 \ -0.0212492 \ 6.3669]; \\ (7)$$

$$[0.759919 \ -0.015194 \ 6.38401] - [0.726977 \ -0.00749985 \ 6.38692] \\ (8)$$

$$(9)$$

such that

*cloth\_vec*

$$VecT = [-0.0079 \ 0.0061 \ 0.0171 \ 0.0329 \ -0.0077 \ -0.0029]; \quad (10)$$

$$VecR = [-0.0083 \ 0.0041 \ 0.0173 \ 0.0347 \ -0.0054 \ -0.0025]; \quad (11)$$

$$(12)$$

where the first 3 entries of  $VecR$  is the vector  $\mathbf{u}_1$  and the last 3 entries of  $VecR$  is the vector  $\mathbf{u}_2$ .  $VecT$  analogously.

## *cloth\_eig\_2D*

we use here *Eigen :: Map* to transform the vector *VecT* and *VecR* to  $2 * 2$  2D deformation gradient  $\mathbf{F}$ .

$$\mathbf{F} = \begin{bmatrix} -0.0083 & 0.0347 \\ 0.0041 & -0.0054 \end{bmatrix} \begin{bmatrix} -0.0079 & 0.0329 \\ 0.0061 & -0.0077 \end{bmatrix}^{-1} \quad (13)$$

hence the  $\mathbf{F}$  has a rotation information  $\mathbf{R}$  and a stretch information  $\mathbf{U}$ ,

$$\mathbf{F} = \mathbf{RU}. \quad (14)$$

we use  $\mathbf{F}^T \mathbf{F}$  to eliminate the rotation information to obtain  $\det \mathbf{R} = 1$

$$\mathbf{F}^T \mathbf{F} = (\mathbf{RU})^T \mathbf{RU} \quad (15)$$

$$= \mathbf{U}^T \mathbf{R}^T \mathbf{R} \mathbf{U} \quad (16)$$

$$= \mathbf{U}^2 \quad (17)$$

$$= \mathbf{C} \quad (18)$$

and using decomposition, if we have the form

$$\mathbf{C} = \lambda_1^2 \mathbf{v}_1 \mathbf{v}_1^T + \lambda_2^2 \mathbf{v}_2 \mathbf{v}_2^T \quad (19)$$

then we could obtain the  $\mathbf{U}$  by applying

$$\mathbf{U} = \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{v}_2 \mathbf{v}_2^T \quad (20)$$

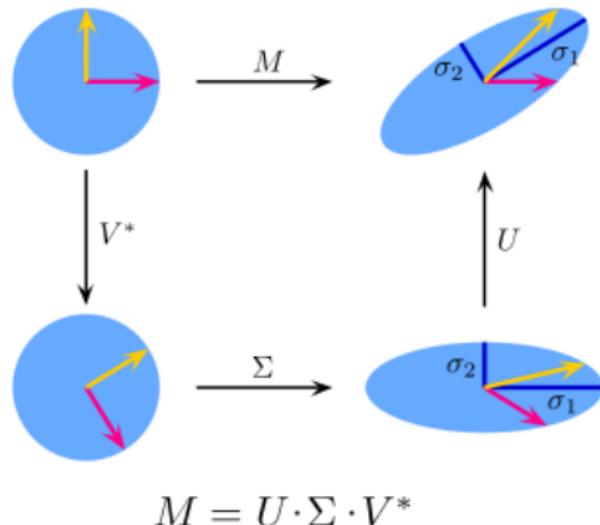


Figure: Singular Value Decomposition SVD

the main drawback of Rohmer et al. is that this is only applied for 2D problem, thus we need a new algorithm, which can also take the consideration for the vertical deformation. Therefore, the *Kabisch Algorithm* is introduced.

## Kabisch Algorithm

Let  $\mathbf{P}$  denotes the points set of template frame, and  $\mathbf{Q}$  denotes the points set of reference frame. The *optimal rotation matrix*  $\mathbf{R}$  can be calculated as

$$\mathbf{R} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} \mathbf{U}^T \quad (21)$$

where  $\mathbf{V}$  and  $\mathbf{U}$  and the *SVD* of *cross-covariance matrix*  $\mathbf{H}$ , which is determined by

### Theorem (cross-covariance matrix)

$$\mathbf{H} = \mathbf{P}^T \mathbf{Q} \quad (22)$$

and  $d = \det \mathbf{V} \mathbf{U}^T$

The matrix  $\mathbf{H}$  is derived from the *orthogonal Procrustes problem*[?]. It is defined as the least-squares problem of transforming a given matrix  $\mathbf{P}$  into a given matrix  $\mathbf{Q}$  by an orthogonal transformation matrix  $\mathbf{R}$ , such that the sums of squares of the residual matrix  $\mathbf{E} = \Omega\mathbf{P} - \mathbf{Q}$  is a minimum

$$\mathbf{R} = \arg \min_{\Omega} \|\Omega\mathbf{P} - \mathbf{Q}\|_F \quad \text{subject to} \quad \Omega^T \Omega = \mathbf{I}, \quad (23)$$

where  $\|\cdot\|_F$  is the Frobenius norm. This problem is equivalent to find the nearest orthogonal matrix  $\mathbf{R}$  to a given matrix  $\mathbf{H} = \mathbf{P}^T \mathbf{Q}$  [?], which most closely maps  $\mathbf{P}$  to  $\mathbf{Q}$ . Thus, we use  $\mathbf{H}$  to approximate the deformation gradient  $\mathbf{F}$  in Eq. (??)

$$\tilde{\mathbf{F}} = \mathbf{H}. \quad (24)$$

The cross-covariance matrix describe the covariance of one process with the other at pairs of time points and measure of similarity. We use therefore the cross-covariance matrix  $\mathbf{H}$  to approximate our deformation gradient  $\mathbf{F}$ .

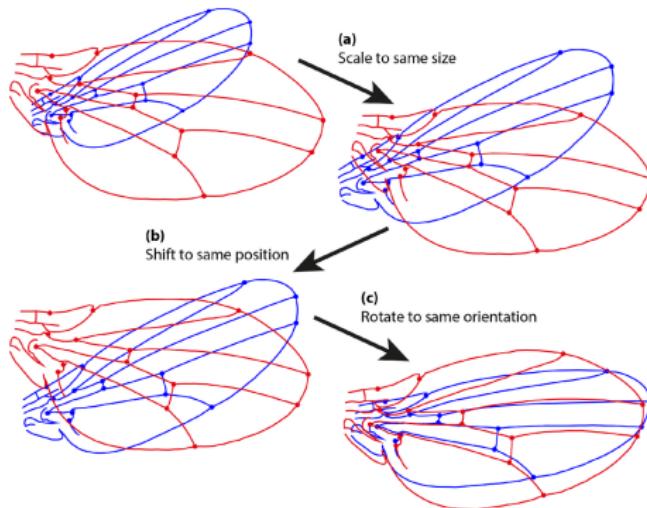


Figure: Procrustes superimposition

Let  $i$  be the all index set of the nearest vertex of vertex  $l$  in the template frame and  $\bar{i}$  be the all index set of the nearest vertex of vertex  $l$  in the template frame. Process define

$$\mathbf{P} = \begin{bmatrix} \vec{x}_1 & \vec{y}_1 & \vec{z}_1 \\ \vec{x}_2 & \vec{y}_2 & \vec{z}_2 \\ \vdots & & \\ \vec{x}_N & \vec{y}_N & \vec{z}_N \end{bmatrix} \quad 1, 2, \dots, N \in i \quad (25)$$

$$\mathbf{Q} = \begin{bmatrix} \bar{\vec{x}}_1 & \bar{\vec{y}}_1 & \bar{\vec{z}}_1 \\ \bar{\vec{x}}_2 & \bar{\vec{y}}_2 & \bar{\vec{z}}_2 \\ \vdots & & \\ \bar{\vec{x}}_N & \bar{\vec{y}}_N & \bar{\vec{z}}_N \end{bmatrix} \quad 1, 2, \dots, N \in \bar{i} \quad (26)$$

and we apply to obtain the deformation gradient  $\mathbf{F}$  of vertex  $l$

$$\mathbf{F} = \mathbf{P}^T \mathbf{Q} \quad (27)$$

Whereas the deformation gradient measures the local deformation, the *spacial velocity gradient*  $\mathbf{L}$  describes the rate of deformation, given by

$$\mathbf{L} = \text{grad } \mathbf{v} = \dot{\mathbf{F}}\mathbf{F}^{-1}, \quad (28)$$

where  $\mathbf{v}$  denotes the *velocity* of the material point  $\mathbf{X}$ , and  $\dot{\mathbf{F}}$  denotes the material time derivative of deformation gradient  $\mathbf{F}$ .

Analogically, we apply the polar decomposition on the spacial velocity gradient  $\mathbf{L}$ [?], and obtain

$$\mathbf{L} = \mathbf{D} + \mathbf{W}. \quad (29)$$

Thus, we could decompose the tensor  $\mathbf{L}$  into its symmetric part  $\mathbf{D}$  and skew-symmetric part  $\mathbf{W}$

$$\mathbf{D} = \frac{1}{2}(\mathbf{L} + \mathbf{L}^T), \quad (30)$$

$$\mathbf{W} = \frac{1}{2}(\mathbf{L} - \mathbf{L}^T). \quad (31)$$

where  $\mathbf{D}$  is called the *rate of strain tensor* and  $\mathbf{W}$  is called the *rate of rotation tensor*.

Assuming  $\mathbf{dx}$  is a material line element in the current configuration, the rate of change of its length  $\dot{\epsilon}_{ii}$  and angle  $\dot{\gamma}_{ij}$  is measured by means of  $\mathbf{D}[?]$ .

$$\dot{\epsilon}_{ii} = \mathbf{e}_i \mathbf{D} \mathbf{e}_i = D_{ii} \quad (32)$$

$$\dot{\gamma}_{ij} = 2 \mathbf{e}_i \mathbf{D} \mathbf{e}_j = D_{ij} \quad (33)$$

where  $D_{ij}$  are the  $ij$ -entries of  $\mathbf{D}$ . Additionally, if  $\mathbf{D}$  has the same base  $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$  as in eq.(20), than

$$D_{11} = \frac{\dot{\lambda}_1}{\lambda_1} \quad (34)$$

$$D_{22} = \frac{\dot{\lambda}_2}{\lambda_3} \quad (35)$$

$$D_{33} = \frac{\dot{\lambda}_2}{\lambda_3} \quad (36)$$

from 0-2 EigNorm1 8759 NaN, Problem:: 26278 has -0 eigenvalues, can't be square. So I use a if(isnan) and define idx=0.  
add lambda1,2,3 for all, which is in debug

save the backsite of clothes using 1-2, 1-3, 1-4. Not much differences from colormap, need a new color map???

save lambda1 for 1,74, they are different, but slightly.

lambda2 has more differences

do also for lambda3

check color run application in loop consider the kr to ring 1-2 1-3 ... 1-4  
2-5 ... 72-75

tensor flow

add neighbor2x

Neo Hook material

please refer

[http://web.mit.edu/abeyaratne/Volumes/RCA\\_Vol\\_II.pdf](http://web.mit.edu/abeyaratne/Volumes/RCA_Vol_II.pdf) p68-70

$\lambda$  and  $D$  relationship

remedy Neighbor2x

*matrix\_example.cpp*

add func to calculate D and L ,think timestep is too small 2-3

<http://www.continuummechanics.org/velocitygradient.html>

0.006 1/200

think should use less 2% points for KDtree

it is because the date set has isolated points: but I thought it doesn't matter, it influence only the colormap.

thoese points can be deleted during the optimazation.

calculate D using kd-tree.

ask Ali the offical video about clothing deforamtion.

need to be emphasised that we focus on the CR(next time step)

Libreoffice Draw A1

Dornisch Friday. monday presentation

paper send to him

add results

add smooth not finish yet. need to connect opengl and calculation

# KW15

setcolor at end  
correct colormap

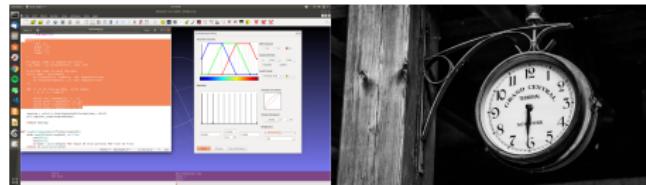


Figure: colormap

the error is by interpolation, how to change it? because meshlab using a unknown function?

don't draw before calculation

glMapBuffer

gluPMapBUFFER

add new results using new colormap, have seen red

add default color as pink for all vertices

add smooth: problem negative values cannot be root square...

add colormap for D, based on t not on t+deltaT

tried to pass calculated data directly to OpenGL class -*gl*? , so I directly read the ply file from output. no color? didn't find the glMapBuffer.

IT is actu. a normalization (Rohmer)

change Gamma to 1/2.5

0.1, 0.2, 0.5, 0.75, 1, 1.5, 2, 3

new evaluate

add python for pic, need rotation

- tensor density <https://www.revolvy.com/page/Tensor-density>
- openFoam limited

visply

slerp

https:

//github.com/cmu462/Scotty3D/wiki/Linear-Blend-Skinning

The screenshot shows a presentation slide with the following content:

**Multi-pass: Objective Function**

min <sub>$\alpha_{ij}, \omega_{ij}, \mathbf{H}_{ij}, \mathbf{T}_{ij}$</sub>   $E = E_D + \omega E_S + \lambda E_J$

Where:

$$E_D = \frac{1}{N} \sum_{i=1}^n \sum_{j=1}^M \sum_{k=1}^K \left\| \left[ \sum_{l=1}^{n+1} \alpha_{kl} [\mathbf{H}_{kl}] [\mathbf{T}_{kl}] \begin{bmatrix} \mathbf{v}_l \\ 1 \end{bmatrix} - \mathbf{v}_r \right] \right\|_2^2$$
$$E_S = \sum_{j=1}^M \mathbf{w}_j^\top \mathbf{L} \mathbf{w}_j$$
$$E_J = \frac{1}{2} \sum_{(i,j) \in E} \sum_{k=1}^n \left\| \begin{bmatrix} [\mathbf{H}_{kj}] [\mathbf{T}_{kj}] - [\mathbf{H}_{kj}] [\mathbf{T}_{kj}] \end{bmatrix} \begin{bmatrix} \mathbf{o}_{jk} \\ 1 \end{bmatrix} \right\|^T$$

Subject to:

$$\alpha_{ij} \geq 0, \sum_{j=1}^M \alpha_{ij} = 1, \|\mathbf{w}_i\|_2 \leq \mathcal{K} (= 4), \forall i, j$$
$$\mathbf{H}_{kj}^\top \mathbf{H}_{kj} = \mathbf{I}, \det \mathbf{H}_{kj} = 1, \forall j, k$$

The right side of the slide shows a 3D rendering of a dark, atmospheric forest scene with glowing lanterns hanging from trees.

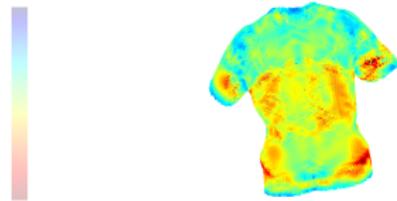
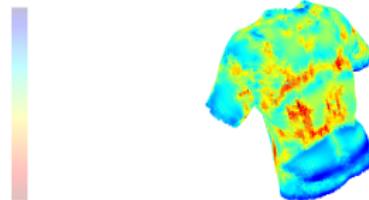
**Transition Cost Function** After laying out the requisite terminology, we have the ingredients to define the transition cost function  $D(i, j)$  between frame  $i$  and  $j$ . We use a squared state-space distance metric,

$$D(i, j) = \|\mathbf{q}_i - \mathbf{q}_j\|_2^2 + \alpha \|\dot{\mathbf{q}}_i - \dot{\mathbf{q}}_j\|_2^2 + \beta \|\mathbf{r}_i - \mathbf{r}_j\|_2^2, \quad (5)$$



cannot use QPPP because there are 0 acnnot be inversed, could use  
kdtree but not good  
need binary datas  
still negative entries

## matrix log / exp



add mapNeighbor method

add update vertices using kdtree because of 0 of trimesh

question about the t 2 cases  
instable for update  
calculation times