

DISCUSSION LOG

Sikang Yan

University of Kaiserslautern

yan@rhrk.uni-kl.de

March 2, 2019

THEORY

In our cloth simulation, we follow the procedure written by Rohmer et al.. We consider to begin with the *Deformation 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 Deformation 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) characrizе 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:

- $faces(i, j)$ is the j th vertex of the i th triangle, here we choose the $face(0, 0)$, $face(0, 1)$, $face(0, 2)$ as examples and calculate the $VecT$ and $VecR$ for $face(0, 0)$.

$$VecT = e_{l_1} VertT_1 - e_{l_1} VertT_2; e_{l_1} VertT_1 - e_{l_1} VertT_3 \quad (3)$$

$$VecR = e_{l_1} VertR_1 - e_{l_1} VertR_2; e_{l_1} VertR_1 - e_{l_1} VertR_3 \quad (4)$$

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

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

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

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

$$(9)$$

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} ,

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

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

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

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

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

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

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

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

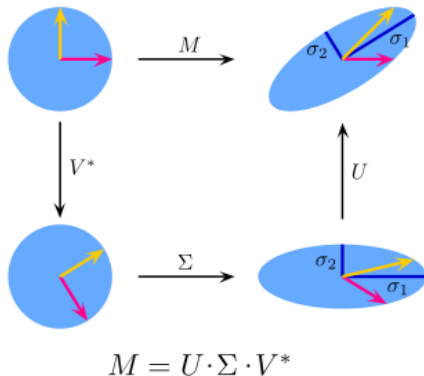


Figure: Singular Value Decomposition SVD

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.

Kabsch 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 $\mathbf{E} = \mathbf{\Omega P} - \mathbf{Q}$ is a minimum

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

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} .

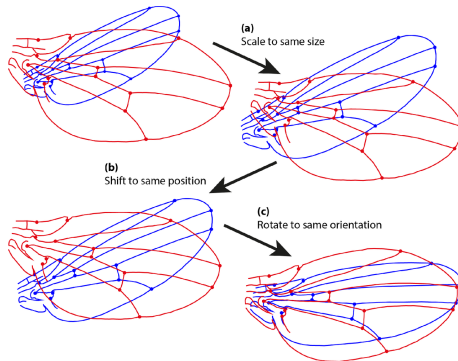


Figure: Procrustes superimposition

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

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

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

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

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

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

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), then

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

$$D_{22} = \frac{\dot{\lambda}_2}{\lambda_2} \quad (35)$$

$$D_{33} = \frac{\dot{\lambda}_3}{\lambda_3} \quad (36)$$

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

save the backside 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 matrial

please refer "click me!!!" p68-70 λ and **D** relationship
remedy Neighbor2x

matrix_example.cpp

add func to calculate D and L ,think timestep is to small