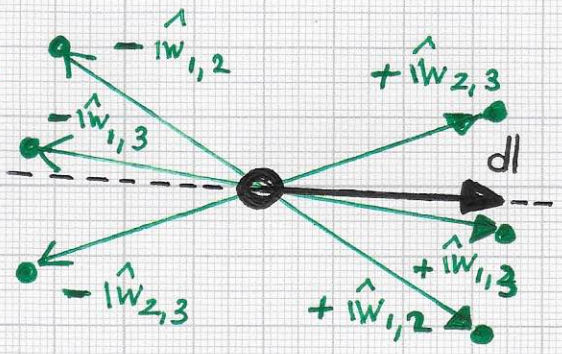
OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

- Laplacian eigenfunctions and neural networks: ...



In other words, for each calculated direction vector $w_{i,j}$ we compute its normalized version, called $\hat{w}_{i,j} = w_{i,j} / \| w_{i,j} \|$, and establish the vector pair $+\hat{w}_{i,j}$ and $-\hat{w}_{i,j}$, see the Left figure. We can now use principal component analysis (PCA) and apply it to the set of 'o' points with the vectors +/- $\hat{w}_{i,j}$ being their positional vectors. (The figure shows a scenario resulting from three planes P_1, P_2 and P_3.) The unit vector $dl = (d_1, d_2, d_3)^T$ is the normalized PCA eigenvector associated with the largest eigenvalue. THE VECTOR dl DEFINES THE UNIT ROTATION AXIS DIRECTION VECTOR FOR THE QUATERNION NOTATION OF $q' = (c\frac{\alpha}{2} + s\frac{\alpha}{2}\langle i, dl \rangle) \cdot q \cdot (c\frac{\alpha}{2} - s\frac{\alpha}{2}\langle i, dl \rangle)$. Only the value of the rotation angle α must be determined via a single-parameter optimization step. As before, we must minimize the sum of squared distances between point pairs $\hat{y}_i$ and x_i, $i = 1 \ldots n$, where $\hat{y}_i$ is the result of rotating y_i around the axis with direction dl and passing through the origin by α. ...

OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:... We review the calculation of an image point using quaternion representation for a rotation in 3D space around an axis through the origin. First, we do not consider indices of points to keep the calculation and notation simple. We rotate a point $\mathbf{x} = (x, y, z)^T$ by an angle α around the directed rotation axis defined by the unit direction vector $\mathbf{u} = (u, v, w)^T$, $\|\mathbf{u}\| = 1$. Further, we use the notation $\hat{c} = \cos\frac{\alpha}{2}$ and $\hat{s} = \sin\frac{\alpha}{2}$. Thus, the rotation is given as

$$(\hat{c} + \hat{s}(iu + jv + kw)) \cdot (0 + ix + jy + kz) \cdot (\hat{c} - \hat{s}(iu + jv + kw)).$$

Pages 11-17 (12/2/2023 – 12/8/2023) provide the detailed calculations involved in computing this quaternion product, i.e., the image point $(0 + ix' + jy' + kz') \triangleq \mathbf{x}' = (x', y', z')^T$. It is

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} u^2(1-c)+c & uv(1-c)-ws & uw(1-c)+vs \\ uv(1-c)+ws & v^2(1-c)+c & vw(1-c)-us \\ uw(1-c)-vs & vw(1-c)+us & w^2(1-c)+c \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R\mathbf{x}.$$

Here, the matrix entries use the notation $c = \cos\alpha$ and $s = \sin\alpha$. This mapping of $\mathbf{x}$ to $\mathbf{x}'$ is needed for the computation of the squared distances of all point pairs $\mathbf{y}_i$ and $\mathbf{x}_i$, $i = 1 \ldots n$.

...

OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks:... Since our goal is the minimization of the sum of these squared distances, i.e., $\sum_{i=1}^{n} \| \hat{w}_i - x_i \|^2$, we must perform differentiation with respect to the angle α that must be optimized. Thus, we only differentiate the individual terms, obtaining $\frac{d}{d\alpha}\left(\| \hat{w}_i - x_i \|^2 \right)$. First, we only consider a point $x = (x, y, z)^T$ and a point $\bar{a}_i = (\bar{a}, \bar{b}, \bar{c})^T$ to determine the general result for $\frac{d}{d\alpha}\left(\| x' - \bar{a}_i \|^2 \right) = \frac{d}{d\alpha}\left(\| Rx - \bar{a}_i \|^2 \right) = \frac{d}{d\alpha}\left(\| R \cdot (x, y, z)^T - (\bar{a}, \bar{b}, \bar{c})^T \|^2 \right)$. By avoiding the index i in this differentiation, the calculations to be done are simpler. One obtains the following result:

$$\frac{d}{d\alpha}\left(\| R \cdot (x, y, z)^T - (\bar{a}, \bar{b}, \bar{c})^T \|^2 \right) = \dots$$

$$\| v - w \|^2 = (v_1 - w_1)^2 + (v_2 - w_2)^2 + (v_3 - w_3)^2 = (v - w)^T (v - w) = \langle v - w, v - w \rangle$$

$$= \frac{d}{d\alpha}\left((Rx - \bar{a}_i)^T (Rx - \bar{a}_i) \right) = \frac{d}{d\alpha}\left(((Rx)^T - \bar{a}_i^T)(Rx - \bar{a}_i) \right) =$$

$$= \frac{d}{d\alpha}\left((x^T R^T - \bar{a}_i^T)(Rx - \bar{a}_i) \right) = \frac{d}{d\alpha}\left(x^T R^T R x - x^T R^T \bar{a}_i - \bar{a}_i^T R x + \bar{a}_i^T \bar{a}_i \right) =$$

$$= \dots \quad R \text{ rotation matrix} \Rightarrow R^T = R^{-1} \Rightarrow R^T R = R^{-1} R = I$$

$$= \frac{d}{d\alpha}\left(x^T I x - x^T R^T \bar{a}_i - \bar{a}_i^T R x + \bar{a}_i^T \bar{a}_i \right) =$$

$$= \frac{d}{d\alpha}\left(x^T x - x^T R^T \bar{a}_i - \bar{a}_i^T R x + \bar{a}_i^T \bar{a}_i \right) =$$

$$= \frac{d}{d\alpha}(x^T x) - \frac{d}{d\alpha}(x^T R^T \bar{a}_i) - \frac{d}{d\alpha}(\bar{a}_i^T R x) + \frac{d}{d\alpha}(\bar{a}_i^T \bar{a}_i) = \dots$$

$= 0$ $= 0$

OBJECT AND MATERIAL EIGENFUNCTIONS - Cont'd.

• Laplacian eigenfunctions and neural networks: ...

$$= -\left({}^{d}/_{d\alpha}\left(x^T R^T \bar{a}_i \right) + {}^{d}/_{d\alpha}\left(\bar{a}_i^T R x \right) \right) =$$

$$= -\left(x^T \left({}^{d}/_{d\alpha} R^T \right) \bar{a}_i + \bar{a}_i^T \left({}^{d}/_{d\alpha} R \right) x \right) = \ldots$$

$$R = \begin{bmatrix} u^2 & uv & uw \\ uv & v^2 & vw \\ uw & vw & w^2 \end{bmatrix} + c \begin{bmatrix} 1-u^2 & -uv & -uw \\ -uv & 1-v^2 & -vw \\ -uw & -vw & 1-w^2 \end{bmatrix} + s \begin{bmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{bmatrix} \equiv$$

$$\equiv R_0 + c R_1 + s R_2 .$$

$$\Rightarrow R^T = R_0^T + c R_1^T + s R_2^T .$$

$$\Rightarrow {}^{d}/_{d\alpha} R = R_\alpha = -s R_1 + c R_2 .$$

$$\Rightarrow {}^{d}/_{d\alpha} R^T = R_\alpha^T = -s R_1^T + c R_2^T .$$

$$= -\left(x^T \left(-s R_1^T + c R_2^T \right) \bar{a}_i + \bar{a}_i^T \left(-s R_1 + c R_2 \right) x \right) =$$

$$= \left(s x^T R_1^T - c x^T R_2^T \right) \bar{a}_i + \left(s \bar{a}_i^T R_1 - c \bar{a}_i^T R_2 \right) x =$$

$$= s \left(x^T R_1^T \bar{a}_i + \bar{a}_i^T R_1 x \right) - c \left(x^T R_2^T \bar{a}_i + \bar{a}_i^T R_2 x \right) .$$

This result can now be used to define the derivative, with respect to the rotation angle α_1, of $\| \hat{y}_i - x_i \|^2$, i.e., ${}^{d}/_{d\alpha}\left(\| \hat{y}_i - x_i \|^2 \right) = {}^{d}/_{d\alpha}\left(\| R y_i - x_i \|^2 \right)$, $i = 1 \ldots n$. This derivative's value is

$${}^{d}/_{d\alpha}\left(\| R y_i - x_i \|^2 \right) = s\left(y_i^T R_1^T x_i + x_i^T R_1 y_i \right) - c\left(y_i^T R_2^T x_i + x_i^T R_2 y_i \right).$$

Since the goal is the minimization of the sum of the squared distances of all point pairs $\hat{y}_i$ and x_i, the sum of the respective derivatives must be 0, i.e.,

$$s \sum_{i=1}^{n} \left(y_i^T R_1^T x_i + x_i^T R_1 y_i \right) - c \sum_{i=1}^{n} \left(y_i^T R_2^T x_i + x_i^T R_2 y_i \right) = 0.$$

$$\Rightarrow \frac{s}{c} = \tan\alpha = \frac{\sum_{i=1}^{n} y_i^T R_2^T x_i + x_i^T R_2 y_i}{\sum_{i=1}^{n} y_i^T R_1^T x_i + x_i^T R_1 y_i} \neq 0 \ldots$$

■ OBJECT AND MATERIAL EIGENFUNCTIONS – Cont'd.

• Laplacian eigenfunctions and neural networks:... The crucial "trick" used in the derivation of the formula for $\tan\alpha$ on the previous page is the representation of the rotation matrix as the sum $R = R_0 + \cos\alpha\, R_1 + \sin\alpha\, R_2$. Since differentiation must be done with respect to the angle α, using $\cos\alpha$ and $\sin\alpha$ as factors of R_1 and R_2 makes it simpler to calculate $d/d\alpha$ of the function of interest.

At this point, we have computed the needed unit axis direction vector in a first step and optimal rotation angle α in a second step — completing the desired "optimal" quaternion-based rotation, see page 1 (3-28-2024). We can now apply the rotation to all points $\mathbb{y}_i$, $i=1\ldots n$; the resulting distance between $\hat{\mathbb{y}}_i$ and $\mathbb{x}_i$ is $\| \hat{\mathbb{y}}_i - \mathbb{x}_i \|$. It is possible to use the following overall approximation errors, for example:

- mean squared distance error $\frac{1}{n} \sum_{i=1}^{n} \| \hat{\mathbb{y}}_i - \mathbb{x}_i \|^2$;
- mean distance error $\frac{1}{n} \sum_{i=1}^{n} \| \hat{\mathbb{y}}_i - \mathbb{x}_i \|$;
- root mean squared distance error $\left(\frac{1}{n} \sum_{i=1}^{n} \| \hat{\mathbb{y}}_i - \mathbb{x}_i \|^2 \right)^{1/2}$;
- maximal distance error $\max \{ \| \hat{\mathbb{y}}_i - \mathbb{x}_i \| \}_{i=1}^{n}$.

...