Automated Reconstruction of Three-dimensional Fish Motion, Forces, and Torques – Mathematical Background

Cees J. Voesenek1*, Remco P. M. Pieters1, Johan L. van Leeuwen1

1 Experimental Zoology, Department of Animal Sciences, Wageningen University, Wageningen, Netherlands

* cees.voesenek@wur.nl

Creating the parameterised fish model

Shape

We describe the fish in its undeformed state in a coordinate system ($\hat{x}, \hat{y}, \hat{z}$) attached to the snout of the fish, with $\hat{x}$ in caudal-rostral direction, $\hat{y}$ in lateral direction and $\hat{z}$ in ventral-dorsal direction such that they form a right-handed coordinate system.

A (super-)elliptic cross-sectional shape is used for the eyes, the fin fold and the region of the body without yolk sack is given by:

$$
\begin{align*}
\hat{y}(\psi) &= r\hat{y}\cos(\psi); \\
\hat{z}(\psi) &= \hat{z}_0 \pm r\hat{z}\left[1 - \left(\frac{|\hat{y}(\psi)|}{r\hat{y}}\right)^n\right]^{\frac{1}{n}},
\end{align*}
$$

with $r\hat{y}$ and $r\hat{z}$ the radii in respectively $\hat{y}$- and $\hat{z}$-direction and $\psi = [0, 2\pi]$ where the sign in the equation for $\hat{z}$ depends on the considered quadrant: $+$ in the positive $\hat{z}$-quadrant, $-$ in the negative $\hat{z}$-quadrant. For the fin fold and body cross-sections in the tail, the exponent $n = 2$, making it a regular ellipse. The elliptic exponent for the eyes is 2.3.

The slices of the body with a bulging yolk sack have their egg-like cross-sectional shape defined by a cubic spline fit (using MATLAB’s `spline`) through 4 points: $(x_0, r\hat{y}, \hat{z}_0 + h_{\text{shift}})$, $(x_0, 0, \hat{z}_0 + r\hat{z})$, $(x_0, -r\hat{y}, \hat{z}_0 + h_{\text{shift}})$, $(x_0, 0, \hat{z}_0 - r\hat{z})$. To prevent edge effects of the spline, we perform the fit on three repetitions of the points, and only use the middle section as a cross-section.

We build up the body with $N_{\text{lon}}$ cross-sections and $N_{\text{circ}}$ points along the circumference of each cross-section. We can connect each set of two cross-sections with $(N_{\text{lon}} - 1)N_{\text{circ}}$ quadrilateral faces, or $2(N_{\text{lon}} - 1)N_{\text{circ}}$ triangular faces. To close the surface, we cap the front- and aftmost cross-sections with $N_{\text{circ}}$ triangular faces each.

Parameterisation

The body curvature $\kappa$ is prescribed in a small number ($< N_{\text{lon}}$, 7 for the zebrafish larva) of control points along the length of the fish. We define a normalised parameter along the fish length $s = [0, 1]$ and use it to create a cubic spline (MATLAB’s `spline`) that interpolates the curvature control points. We analytically integrate each piecewise polynomial of the curvature spline to obtain the local deformation angle:

$$
\theta(s) = \ell \int_0^s \kappa(s) \, ds,
$$
with \( \ell \) the body length. We model the fish’s centreline as a sequence of straight segments \( i \) of length \( \Delta \ell = \ell / (N_{\text{ion}} - 1) \), and apply the local deformation angle to each:

\[
\begin{align*}
\hat{x}_{\text{centre},i} &= \begin{cases} 
0 & i = 1 \\
\hat{x}_{i-1} - \Delta \ell \cos \left( \frac{\theta_{i-1} + \theta_i}{2} \right) & \text{otherwise}
\end{cases} \\
\hat{y}_{\text{centre},i} &= \begin{cases} 
0 & i = 1 \\
\hat{y}_{i-1} + \Delta \ell \sin \left( \frac{\theta_{i-1} + \theta_i}{2} \right) & \text{otherwise}
\end{cases}
\end{align*}
\]

in coordinates relative to the fish’s head. Every cross-section \( i \) consists of \( N_{\text{circ}} \) circumferential points to which we assign the index \( j \). We rotate every point \( i,j \) on the cross-section with the rotation of the centreline, such that it remains perpendicular:

\[
\begin{bmatrix}
\hat{x}_{\text{def},i,j} \\
\hat{y}_{\text{def},i,j} \\
\hat{z}_{\text{def},i,j}
\end{bmatrix} = 
\begin{cases} 
\hat{x}_{\text{centre},i} + \hat{y}_{i,j} \sin \theta_i \\
\hat{y}_{\text{centre},i} + \hat{y}_{i,j} \cos \theta_i \\
\hat{z}_{i,j}
\end{cases}
\]

(4)

We now have a deformed fish in the \((\hat{x}, \hat{y}, \hat{z})\) coordinate system, which we place in world coordinates with the position \( x_{\text{snout}} \) and rotation \( \phi \) at the snout. The angles \( \phi_{\text{roll}}, \phi_{\text{pitch}}, \phi_{\text{yaw}} \) (abbreviated as \( \phi_r, \phi_p, \phi_y \) for reasons of clarity) specifying the rotation are converted to a rotation matrix:

\[
R = 
\begin{bmatrix}
\cos \phi_p \cos \phi_y & -\sin \phi_p & \cos \phi_r \sin \phi_p \\
\sin \phi_r \sin \phi_y + \cos \phi_r \sin \phi_y & \cos \phi_r \cos \phi_y & -\sin \phi_r \sin \phi_y \\
-\cos \phi_r \sin \phi_y + \sin \phi_r \cos \phi_y & \cos \phi_r \cos \phi_y & \cos \phi_r \sin \phi_y
\end{bmatrix}
\]

(5)

We compute the final position of each point on the fish with:

\[
x_{\text{def},i,j} = R\hat{x}_{\text{def},i,j} + x_{\text{snout}}.
\]

(6)

Finally, all points are reassembled into a three-dimensional surface of quadrilaterals during tracking and of triangles during post-processing.

**Fitting the body model**

**Objective function**

The objective function that we aim to minimise in order to achieve an optimal fit is defined as follows:

\[
f_{\text{tot}}(\Omega) = f_{\text{GoF}}(\Omega) + f_{\text{reg}}(\Omega).
\]

(7)

Here, \( \Omega \) is the list of optimised parameters, \( f_{\text{GoF}}(\Omega) \) is a term indicating how accurately the fish model approaches the high-speed video images, and \( f_{\text{reg}}(\Omega) \) is a term that penalises “unsmoothness”. The two terms are discussed in detail below.

**Goodness of fit**

To compare the virtual images to the segmented high-speed video images, we project the three-dimensional shape of the fish onto virtual cameras. We define a left-handed camera coordinate system \((\xi, \eta, \theta)\), respectively rows, columns, and a coordinate perpendicular to the image plane in the viewing direction of the camera. A transformation matrix from camera to world coordinate system is given by:

\[
T_{\text{cam}} = 
\begin{bmatrix}
\omega_{x,\xi} & \omega_{x,\eta} & \omega_{x,\theta} \\
\omega_{y,\xi} & \omega_{y,\eta} & \omega_{y,\theta} \\
\omega_{z,\xi} & \omega_{z,\eta} & \omega_{z,\theta}
\end{bmatrix}
\]

(8)
Since we are considering a camera system with parallel light, the distance of an object from the camera has no influence on its projected size. The light is collimated to within $10^\circ$ divergence, so perspective effects are negligible at the millimetre scales of zebrafish motion. Projecting objects onto the camera is therefore only a coordinate transformation to camera coordinates (i.e., a rotation, translation and rescaling):

$$\xi = \frac{1}{d_{\text{scale}}} T^T_{\text{cam}} (x_{\text{obj}} - x_{\text{cam}}) + 1,$$  \hspace{1cm} (9)

where $x_{\text{object}}$ are the object coordinates, $x_{\text{cam}}$ the camera coordinate system origin (i.e., the top left corner of the image), $d_{\text{scale}}$ a scaling factor in unit distance per pixel. The 1 is added because by convention, the top left corner is denoted as having row-column (i.e., $\xi, \eta$) coordinates (1, 1). Because the transformation matrix $T_{\text{cam}}$ is orthogonal, we can invert the matrix by taking its transpose. The image plane coordinates of the projected point are given by $(\xi, \eta)$; the third coordinate $\vartheta$ can be ignored.

When considering cases with significant magnification effects (e.g., when no collimated light setup is used), we use a simple linear pinhole model, where image magnification is linear with distance from the camera. In that case, we use the $\vartheta$ coordinate to determine the scaling factor for the current projected point and thus compute its projection on the image plane.

To compute the goodness of fit, we project the three-dimensional surface of the fish onto the camera plane using the procedure outlined above. We then overlay the fish silhouette in image coordinates onto the segmented high-speed video frames. All pixels in the combined image are counted, and the pixels that overlap between the model fish and the actual fish are subtracted from this number. The final result is the number of mismatching pixels – an expression for the goodness of fit.

**Penalty function**

The regularising term in the objective function is of the form

$$f_{\text{reg}} = \int_0^1 w(s) \left( \frac{d^2 s(s)}{ds^2} \right)^2 ds,$$  \hspace{1cm} (10)

where $w(s)$ is a weighting function that allows stronger suppression of curvature near the head than near the tail. For the zebrafish, we prescribe $w(s)$ as a cubic spline fit (MATLAB’s `spline`) in logarithmic space of control points along the body. In our case of the larval zebrafish, we prescribe two points at $s = \{0, 1\}$. Since a cubic spline fit of two points is linear, the result is a function of the form $w(s) = c_0 e^{-c_1 s}$.

**Initialisation**

The initial set of model parameters for the first frame $\omega_0$ is computed from two user-clicked points per camera: the snout and the tail. Rewriting Eqn. (9) gives, for every camera,

$$\frac{1}{d_{\text{scale}}} T^T_{\text{cam}} x_{\text{obj}} = \xi + \frac{1}{d_{\text{scale}}} T^T_{\text{cam}} x_{\text{cam}} - 1.$$  \hspace{1cm} (11)

Manually indicating a point in an image will give us the first two components of $\xi$, so we can use the first two equations of this system for each camera. This results in an overdetermined linear system (for $N_{\text{cam}} > 2$): we have 3 unknowns $(x, y, z)$ and $2N_{\text{cam}}$ equations. We solve this system with least-squares (MATLAB’s `\` operator) to obtain the snout and tail positions in 3D.

The snout position can be directly used to initialise the position parameter for the optimisation. We set the initial roll angle to 0, the pitch angle to

$$\varphi_{\text{pitch}} = \arctan \left( \frac{z_{\text{snout}} - z_{\text{tail}}}{(x_{\text{snout}} - x_{\text{tail}})^2 + (y_{\text{snout}} - y_{\text{tail}})^2} \right)^{\frac{1}{2}}$$  \hspace{1cm} (12)
and the yaw angle to
\[ \varphi_{\text{yaw}} = \arctan \left( \frac{y_{\text{snout}} - y_{\text{tail}}}{x_{\text{snout}} - x_{\text{tail}}} \right). \] (13)

The arctangents are computed using MATLAB’s \texttt{atan2} function, that takes into account the sign of the nominator and denominator to compute the angle in the correct quadrant.

We approximate the fish length \( \ell \) as follows:
\[ \ell_{\text{init}} = \sqrt{(x_{\text{snout}} - x_{\text{tail}})^2 + (y_{\text{snout}} - y_{\text{tail}})^2 + (z_{\text{snout}} - z_{\text{tail}})^2} \] (14)

This approximation is used as an initial condition for an optimisation in the first frame of the sequence, where we allow the fish length to vary. The final, optimised fish length \( \ell \) is then kept fixed in the rest of the video sequence.

**Solution extrapolation**

We extrapolate the solution from previous time steps to initialise the next time step. We use the following:

\[
\Omega_{\text{init,}i_t} = \begin{cases} 
\Omega_0 & i_t = 1 \\
\Omega_{\text{opt,}i_t-1} & i_t = 2 \\
-\Omega_{\text{opt,}i_t-2} + 2\Omega_{\text{opt,}i_t-1} & i_t = 3 \\
\Omega_{\text{opt,}i_t-3} - 3\Omega_{\text{opt,}i_t-2} + 3\Omega_{\text{opt,}i_t-1} & i_t > 3 
\end{cases} 
\] (15)

where \( i_t \in \mathbb{N} \) denotes the time step, \( \Omega_{\text{init,}i_t} \) is the set of parameters used to initialise the optimisation at time step \( i_t \) and \( \Omega_{\text{init,}i_t} \) is the final, optimised set of parameters at time step \( i_t \).

**Post-processing and inverse dynamics**

**Smoothing**

In general, the solution from the optimisation procedure is insufficiently smooth in time to accurately compute (second) derivatives. We perform post hoc smoothing to remove spurious high-frequency components, but retain (most of) the physical signal. We use a regularised data fit similar to [1], where smoothing is presented as a minimisation problem:

\[
\arg \min_{\phi_{\text{sm}}} \left\{ \int_0^{t_{\text{max}}} \left| \phi_{\text{data}}(t) - \phi_{\text{sm}}(t) \right|^2 dt + \lambda_{\text{sm}} \int_0^{t_{\text{max}}} \left| \frac{d^n \phi_{\text{sm}}(t)}{dt^n} \right|^2 dt \right\} 
\] (16)

where \( \phi_{\text{sm}} \) are the smoothed data points, \( \phi_{\text{data}} \) are the measured data points, \( \lambda_{\text{sm}} \) is a parameter controlling the amount of regularisation and \( n \) is the order of the derivative used for the penalty function. It has been shown [1] that the solution for a discrete number of points is given by

\[ \phi_{\text{sm}} = (I + \lambda_{\text{sm}} D^T D)^{-1} \phi_{\text{data}}, \] (17)

where \( D \) is a matrix that approximates the required \( n^{\text{th}} \)-order derivative. We assume here that our fitted points are located at the same time instants as the data points. For the \( D \) matrix, we use 4 subsequent one-sided differences (using MATLAB’s \texttt{diff}) to approximate the fourth order derivative, ensuring that our second derivatives are smooth.

To combat edge effects of the smoothing, we separate two cases: starts, where the animal accelerates from a stationary position, and “continuous” swimming, where it enters and exits the field of view while swimming. In the case of starts, we know that the animal is stationary before the time series starts. We therefore add 25 time points before \( t = 0 \) which all have the value at \( t = 0 \), thus forcing the solution to have a realistic zero gradient at the start. In the case of “continuous” swimming, we can simply cut off the first few (in our case 5) points, where the edge effects are strongest. For both cases, we cut off the same number of points off the end of the time series.
Derivatives

Differentiation is required to compute velocity and acceleration from the centre of mass position, and torque from the angular momentum vector. We use the same procedure for any time series, for first derivatives:

\[
\frac{\partial \phi}{\partial t} \approx \begin{cases} 
\frac{-3\phi_i + 4\phi_{i+1} - \phi_{i+2}}{2\Delta t} & i_t = 1 \\
\frac{-\phi_{i-1} + \phi_{i+1}}{2\Delta t} & 1 < i_t < N_t \\
\frac{\phi_{N_t} - 2\phi_{N_t-1} + 3\phi_{N_t-2}}{2\Delta t} & i_t = N_t
\end{cases}
\]

for second derivatives:

\[
\frac{\partial^2 \phi}{\partial t^2} \approx \begin{cases} 
\phi_i - 2\phi_{i+1} + \phi_{i+2} \Delta t & i_t = 1 \\
\phi_{i-1} - 2\phi_i + \phi_{i+1} \Delta t & 1 < i_t < N_t \\
\phi_{N_t} - 2\phi_{N_t-1} + 3\phi_{N_t-2} \Delta t & i_t = N_t
\end{cases}
\]

Resultant forces and torques

The resultant force \( F_{\text{net}} \) on the fish can be computed from Newton’s second law:

\[
F_{\text{net}} = ma_{\text{CoM}} = \rho V \frac{d^2 r_{\text{CoM}}}{dt^2},
\]

where \( m \) is fish mass, \( \rho \) is fish density, and \( a_{\text{CoM}} \) denotes the acceleration of the centre of mass; the volume \( V \) and position of the centre of mass \( r_{\text{CoM}} \) can be computed using the method by [2] from the triangulated surface of the body model. Every triangular face is made into a tetrahedron by taking the coordinate system origin as a fourth vertex, see Fig 1A. Calculations can be performed separately on each of these tetrahedra, and subsequently summed to yield the result for the complete fish.

The method by [2] only computes the centre of mass and the moment of inertia matrix; we extend their method to compute angular momentum for an arbitrary triangulated polyhedron of constant density. Angular momentum is defined as:

\[
L = \rho \int \int \int_V r^* (x) \times v^* (x) \, dV,
\]

where the asterisks denotes quantities measured with respect to the centre of mass. The integrand is linear, so we can evaluate Eqn. [21] for each tetrahedron separately and then sum the results. To simplify this computation, we transform every tetrahedron into a unit tetrahedron in the origin of a new coordinate system \((x', y', z')\), see Fig 1B. This is a linear transformation, with the following transformation matrix from unit to world tetrahedron:

\[
T_{\text{tet}} = \begin{bmatrix}
x_{d,k} & x_{e,k} & x_{f,k} \\
y_{d,k} & y_{e,k} & y_{f,k} \\
z_{d,k} & z_{e,k} & z_{f,k}
\end{bmatrix}
\]

The transformed integral for every tetrahedron becomes

\[
\Delta L = \rho \det T_{\text{tet}} \int_0^1 \int_0^{1-x'} \int_0^{1-x'-y'} r^* (x') \times v^* (x') \, dz' \, dy' \, dx',
\]

Transformed to a unit tetrahedron, linear interpolation of quantities known at its vertices reduces to a single matrix multiplication. The interpolating matrices for the position and velocity are given by:

\[
\Gamma_r = \begin{bmatrix}
x_{d,k} & x_{e,k} & x_{f,k} \\
y_{d,k} & y_{e,k} & y_{f,k} \\
z_{d,k} & z_{e,k} & z_{f,k}
\end{bmatrix}
\]

\[
\Gamma_v = \begin{bmatrix}
v_{x,d,k} & v_{x,e,k} & v_{x,f,k} \\
v_{y,d,k} & v_{y,e,k} & v_{y,f,k} \\
v_{z,d,k} & v_{z,e,k} & v_{z,f,k}
\end{bmatrix}
\]
With interpolation, Eqn. 23 can be written as:

$$\Delta L_k = \rho \det T_{\text{tet}} \int_0^1 \int_0^1 \int_0^1 \Gamma_\rho \cdot \mathbf{X}' \times \Gamma_\rho \cdot \mathbf{x}' \, dz \, dy \, dx'. \quad (26)$$

Evaluating this integral (using Maple 18, Maplesoft, Waterloo, Ontario, Canada) gives the following expressions for the $x$-, $y$- and $z$-components of the angular momentum vector:

$$\Delta L_{x,k} = \frac{\rho \Delta V_k}{20} \left[ + y_{d,k} (2v_{x,d,k} + v_{y,e,k} + v_{z,f,k}) + y_{e,k} (v_{x,d,k} + 2v_{y,e,k} + v_{z,f,k}) + y_{f,k} (v_{x,d,k} + v_{y,e,k} + 2v_{z,f,k}) - z_{d,k} (2v_{y,d,k} + v_{y,e,k} + v_{y,f,k}) - z_{e,k} (v_{y,d,k} + 2v_{y,e,k} + v_{y,f,k}) - z_{f,k} (v_{y,d,k} + v_{y,e,k} + 2v_{y,f,k}) \right]; \quad (27)$$

$$\Delta L_{y,k} = \frac{\rho \Delta V_k}{20} \left[ - x_{d,k} (2v_{x,d,k} + v_{y,e,k} + v_{z,f,k}) - x_{e,k} (v_{x,d,k} + 2v_{y,e,k} + v_{z,f,k}) - x_{f,k} (v_{x,d,k} + v_{y,e,k} + 2v_{z,f,k}) + z_{d,k} (2v_{x,d,k} + v_{y,e,k} + v_{z,f,k}) + z_{e,k} (v_{x,d,k} + 2v_{y,e,k} + v_{z,f,k}) + z_{f,k} (v_{x,d,k} + v_{y,e,k} + 2v_{z,f,k}) \right]; \quad (28)$$

$$\Delta L_{z,k} = \frac{\rho \Delta V_k}{20} \left[ + x_{d,k} (2v_{y,d,k} + v_{y,e,k} + v_{z,f,k}) + x_{e,k} (v_{y,d,k} + 2v_{y,e,k} + v_{z,f,k}) + x_{f,k} (v_{y,d,k} + v_{y,e,k} + 2v_{z,f,k}) - y_{d,k} (2v_{x,d,k} + v_{y,e,k} + v_{z,f,k}) - y_{e,k} (v_{y,d,k} + 2v_{y,e,k} + v_{z,f,k}) - y_{f,k} (v_{y,d,k} + v_{y,e,k} + 2v_{z,f,k}) \right]. \quad (29)$$

The total angular momentum is then given by the sum over all tetrahedra $k$:

$$\mathbf{L} = \sum_k \Delta \mathbf{L}_k. \quad (30)$$

With the angular momentum known in each time step, we can compute the net torque by taking its time derivative:

$$\mathbf{\tau}_{\text{net}} = \frac{d\mathbf{L}}{dt}. \quad (31)$$

**Fish reference system**

To make interpretation of the forces and torques more intuitive, we transform them to a local fish coordinate system $(x_{\text{fish}}, y_{\text{fish}}, z_{\text{fish}})$ that rotates with the changing orientation of the fish. To create this fish axis system, we first rotate the world coordinate system so it is aligned with the fish’s head, using the rotation matrix in Eqn. 34. This transforms the fish to the $(x, y, z)$ coordinate system defined in section [Creating the parameterised fish model](#) where the $\hat{z}$-plane is aligned with the deformation plane of the fish, and $\hat{z}$ is perpendicular to it. The $\hat{x}$-coordinate is aligned with the head, which rotates with a large amplitude and makes interpretation of the time-variation of the forces and torques difficult.

In order to define an angle that is related to the entire body, we use a “body angle” $\varphi$, defined as:

$$\varphi_{\text{body}} = \frac{\int_0^1 I_{\text{CoM}}(s) \alpha(s) \, ds}{\int_0^1 I_{\text{CoM}}(s) \, ds}, \quad (32)$$

where $I_{\text{CoM}}(s)$ is the moment of inertia about the centre of mass per unit length of the cross-section at $s$ and $\alpha(s)$ is the angle between a line projected from the centre of mass to the centre of segment and horizontal. The body angle is effectively a moment of inertia-weighted average of the local angles of each segment.
where \( i \) is the index of the point just right of the discontinuity. The entire sequence to the right of the discontinuity is shifted, retaining gradient information. We iterate over all discontinuities in the sequence, making sure we get the correct sign depending on the quadrant we are in by using MATLAB’s \texttt{atan2} function.

We compute the body angle by evaluating the integral from Eqn. \( 32 \) using MATLAB’s \texttt{trapz} function. This results in a series of body angles over time that may still have discontinuities where the centre of mass crosses the centreline; in these points, the angle “reflects” around an unknown line since all the projected lines flip direction. To remove these discontinuities, we average a one-sided derivative left and right of the discontinuity. We then extrapolate the solution from the point left of the discontinuity:

\[
\alpha_i = \alpha_{i-1} + \frac{1}{2} \left( \frac{\alpha_{i-3} - 4\alpha_{i-2} + 3\alpha_{i-1} - 3\alpha_i + 4\alpha_{i+1} - \alpha_{i+2}}{2} \right),
\]

where \( i \) is the index of the point just right of the discontinuity. The entire sequence to the right of the discontinuity is shifted, retaining gradient information. We iterate over all discontinuities in the sequence, yielding a body angle that is continuous from the first angle onwards. Since we start with a straight fish in the \( \hat{x}, \hat{y}, \hat{z} \)-coordinates, we define this first angle as 0.

We rotate the head-aligned coordinate system along its vertical axis by the computed body angle. We represent the rotation as an axis and angle: respectively \( \omega_{z_{\text{head}}} \) (abbreviated in Eqn. \( 36 \) as \( \omega \)), the unit orientation vector of the \( z_{\text{head}} \)-axis and \( \varphi_{\text{body}} \) (abbreviated in Eqn. \( 36 \) as \( \varphi \)). This results in a rotation matrix of the following form:

\[
R_{\text{body}} = \begin{bmatrix}
\cos \varphi & +\omega_y (1 - \cos \varphi) & +\omega_z (1 - \cos \varphi) \\
-\omega_z \sin \varphi & +\omega_y (1 - \cos \varphi) & +\omega_z (1 - \cos \varphi) \\
+\omega_y \sin \varphi & \omega_x \omega_z (1 - \cos \varphi) & -\omega_x \sin \varphi
\end{bmatrix}
\]

Figure 1. Tetrahedra used for computation of derived quantities: centre of mass position and instantaneous forces and torques. (A) To compute volume properties from a surface description, a tetrahedron is created for each surface triangle \( \text{d} \ \text{e} \ \text{f} \), with a volume \( \Delta V \) and a centre of mass position \( \Delta r_{\text{CoM}} \). The sign of its contribution is determined by the direction of the normal \( \text{n} \). (B) The same tetrahedron is transformed into a unit tetrahedron for easier interpolation and integration, allowing us to compute the contribution of each tetrahedron to the angular momentum separately with a single analytical expression as a function of the values at the vertices.
Applying this rotation matrix to the axes of the head coordinate system yields our final fish coordinate system \((x_{\text{fish}}, y_{\text{fish}}, z_{\text{fish}})\), in which we express our forces and torques. We transform the force vector by projecting it onto each unit orientation vector of the axes, e.g.,

\[
F_{x,\text{fish}} = F_x \cdot \omega_{x_{\text{fish}}}. \tag{37}
\]

We define an additional axis in the direction of the velocity vector:

\[
\omega_{\text{fwd}} = \frac{v_{\text{CoM}}}{|v_{\text{CoM}}|}, \tag{38}
\]

where \(v_{\text{CoM}}\) is the instantaneous velocity vector of the centre of mass. We project the force to this axis to have an indication of the effective force propelling the fish.

**References**

1. Stickel JJ. Data smoothing and numerical differentiation by a regularization method. Comput Chem Eng. 2010;34(4):467–475. Available from: [http://dx.doi.org/10.1016/j.compchemeng.2009.10.007](http://dx.doi.org/10.1016/j.compchemeng.2009.10.007).

2. Dobrovolskis AR. Inertia of any polyhedron. Icarus. 1996;124(2):698–704. Available from: [http://dx.doi.org/10.1006/icar.1996.0243](http://dx.doi.org/10.1006/icar.1996.0243).

3. Van Leeuwen JL, Voesenek CJ, Müller UK. How body torque and Strouhal number change with swimming speed and developmental stage in larval zebrafish. J R Soc Interface. 2015;12(110):20150479. Available from: [http://dx.doi.org/10.1098/rsif.2015.0479](http://dx.doi.org/10.1098/rsif.2015.0479).

4. Liggett JA. Exact formulae for areas, volumes and moments of polygons and polyhedra. Commun Appl Numer M. 1988;4(6):815–820. Available from: [http://dx.doi.org/10.1002/cnm.1630040616](http://dx.doi.org/10.1002/cnm.1630040616).