Quaternions (https://en.wikipedia.org/wiki/Quaternion)

Extension of the complex numbers:

a) Complex numbers: $\mathbf{c} = \mathbf{a} + \mathbf{b} * \mathbf{i};\ \boldsymbol{i^2} = \mathbf{-1}$

b) Quaternions:

$$\boldsymbol{q} = \boldsymbol{q_0} + \boldsymbol{q_1} * \boldsymbol{i} + \ \boldsymbol{q_2} * \boldsymbol{j} + \boldsymbol{q_3} * \boldsymbol{k}; \boldsymbol{i^2} = \boldsymbol{j^2} = \boldsymbol{k^2} = \mathbf{-1}; \boldsymbol{i} * \boldsymbol{j} = \boldsymbol{k}; \boldsymbol{j} * \boldsymbol{k} = \boldsymbol{i}; \boldsymbol{k} * \boldsymbol{i} = \boldsymbol{j}$$

Quaternion addition is simply the some of its parts:

$$\boldsymbol{q} + \boldsymbol{r} = (\boldsymbol{q_0} + \boldsymbol{r_0}) + (\boldsymbol{q_1} + \boldsymbol{r_1}) * \boldsymbol{i} + \ (\boldsymbol{q_2} + \boldsymbol{r_2}) * \boldsymbol{j} + (\boldsymbol{q_3} + \boldsymbol{r_3}) * \boldsymbol{k}$$

The product of quaternions is non-commutative, but distributive:

$$\boldsymbol{q} * \boldsymbol{r} = (q_0 + q_1 * i + \ q_2 * j + q_3 * k) * (r_0 + r_1 * i + \ r_2 * j + r_3 * k) = (\boldsymbol{q_0}, \vec{\boldsymbol{q}}) * (\boldsymbol{r_0}, \vec{\boldsymbol{r}})$$

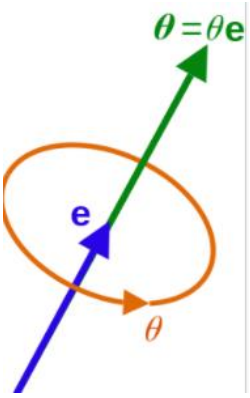
$$= (q_0 * r_0 - q_1 * r_1 - q_2 * r_2 - q_3 * r_3) + i * (q_0 * r_1 + q_1 * r_0 + q_2 * r_3 - q_3 * r_2) +$$

$$+ j * (q_0 * r_2 + q_2 * r_0 + q_3 * r_1 - q_1 * r_3) + k * (q_0 * r_3 + q_3 * r_0 + q_1 * r_2 - q_2 * r_1)$$

$$= (\boldsymbol{q_0} * \boldsymbol{r_0} - \vec{\boldsymbol{q}}.\vec{\boldsymbol{r}},\ \boldsymbol{q_0} * \vec{\boldsymbol{r}} + \boldsymbol{r_0} * \vec{\boldsymbol{q}} + \vec{\boldsymbol{q}} \times \vec{\boldsymbol{r}})$$

Unit Quaternions and 3d rotations

Unit Quaternions are quaternions q with $\boldsymbol{q_0^2} + \boldsymbol{q_1^2} + \boldsymbol{q_2^2} + \boldsymbol{q_3^2} = \mathbf{1}$. The relation to 3D rotations come from the axis-angle representation, $\boldsymbol{q} = [\mathbf{cos}\left(\frac{\boldsymbol{\theta}}{\mathbf{2}}\right), \vec{\boldsymbol{e}} * \mathbf{sin}\left(\frac{\boldsymbol{\theta}}{\mathbf{2}}\right)]$.



The quaternions representing a vector or a point is defined as $\boldsymbol{q}(\vec{\boldsymbol{v}}) = \mathbf{(0, \vec{v})}$, where the real part of the quaternion is $\boldsymbol{0}$, and the vector part of the quaternion corresponds to the coordinates of the vector or the point. The conjugate of the quaternion $\boldsymbol{q} = (\boldsymbol{q_0}, \vec{\boldsymbol{q}}_v)$ is the quaternion with the same real part and the opposite vector part, i.e. $\boldsymbol{q}^* = (\boldsymbol{q_0}, -\vec{\boldsymbol{q}}_v)$. The transformation of a vector by a matrix R, i.e $\vec{\boldsymbol{v}}_1 = \boldsymbol{R} * \vec{\boldsymbol{v}}$ is written with quaternions as: $\boldsymbol{q}(\vec{\boldsymbol{v}}_1) = \ \boldsymbol{q} * \boldsymbol{q}(\vec{\boldsymbol{v}}) * \boldsymbol{q}^*$.

This leads to the representation *R* as a function of the quaternions:

$$R(q) = \begin{bmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2 * (q_1 * q_2 - q_0 * q_3) & 2 * (q_1 * q_3 + q_0 * q_2) \\ 2 * (q_1 * q_2 + q_0 * q_3) & (q_0^2 + q_2^2 - q_1^2 - q_3^2) & 2 * (q_2 * q_3 - q_0 * q_1) \\ 2 * (q_1 * q_3 - q_0 * q_2) & 2 * (q_2 * q_3 + q_0 * q_1) & (q_0^2 + q_3^2 - q_1^2 - q_2^2) \end{bmatrix}$$

Dual Numbers

In order to understand dual quaternion, we have to understand first dual numbers. "In algebra, the dual numbers are a hypercomplex number system first introduced in the 19th century. They are expressions of the form a + b ε, where a and b are real numbers, and ε is a symbol where $\varepsilon^2 = 0$". (https://en.wikipedia.org/wiki/Dual_number).

$$(a + b\varepsilon)(c + d\varepsilon) = ac + (ad + bc)\varepsilon,$$

Dual quaternions

Dual quaternions are the same as quaternions, with the difference is that the 4 elements of the quaternions are dual numbers (https://en.wikipedia.org/wiki/Dual_quaternion). "A dual quaternion can be represented in the form: $\widehat{\boldsymbol{A}} = \mathbf{A} + \boldsymbol{\varepsilon}\mathbf{B} = (\mathbf{A} + \boldsymbol{\varepsilon}\mathbf{B})$, where A and B are ordinary quaternions and ε is the dual symbol".

The addition and multiplication properties are:

$$\widehat{\boldsymbol{A}} + \widehat{\boldsymbol{C}} = (\boldsymbol{A} + \boldsymbol{\varepsilon}\boldsymbol{B}) + (\boldsymbol{C} + \boldsymbol{\varepsilon}\boldsymbol{D}) = (\boldsymbol{A} + \boldsymbol{C} + \boldsymbol{\varepsilon}(\boldsymbol{B} + \boldsymbol{D}))$$

$$\widehat{\boldsymbol{A}} * \widehat{\boldsymbol{C}} = (\boldsymbol{A} + \boldsymbol{\varepsilon}\boldsymbol{B}) * (\boldsymbol{C} + \boldsymbol{\varepsilon}\boldsymbol{D}) = (\boldsymbol{A} * \boldsymbol{C} + \boldsymbol{\varepsilon}(\boldsymbol{A} * \boldsymbol{D} + \boldsymbol{B} * \boldsymbol{C}))$$

The multiplication table is:

Multiplication table for dual quaternion units

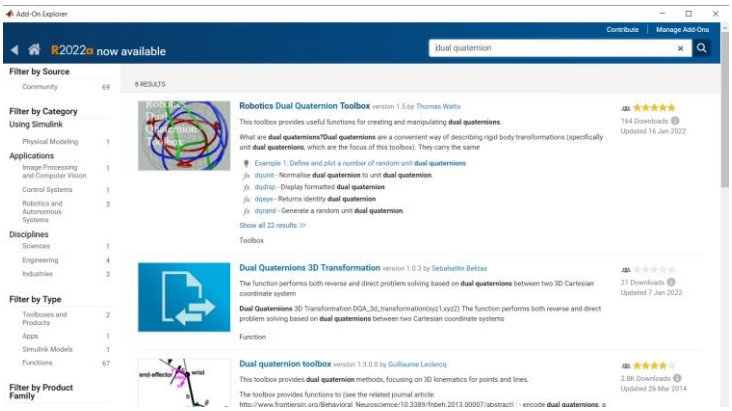
(Row x Column)	1	*i*	*j*	*k*	ε	ε*i*	ε*j*	ε*k*
1	1	*i*	*j*	*k*	ε	ε*i*	ε*j*	ε*k*
i	*i*	−1	*k*	−*j*	ε*i*	−ε	ε*k*	−ε*j*
j	*j*	−*k*	−1	*i*	ε*j*	−ε*k*	−ε	ε*i*
k	*k*	*j*	−*i*	−1	ε*k*	ε*j*	−ε*i*	−ε
ε	ε	ε*i*	ε*j*	ε*k*	0	0	0	0
ε*i*	ε*i*	−ε	ε*k*	−ε*j*	0	0	0	0
ε*j*	ε*j*	−ε*k*	−ε	ε*i*	0	0	0	0
ε*k*	ε*k*	ε*j*	−ε*i*	−ε	0	0	0	0

The dual quaternions can be also expressed as a dual number and dual vector:

$$\hat{q}_1 * \hat{q}_2 = (\hat{s}_1, \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{1}}) * (\hat{s}_2, \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{2}}) = (\hat{s}_1 * \hat{s}_2 - \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{1}}^{\boldsymbol{T}} \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{2}}, \hat{s}_1 * \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{2}} + \hat{s}_2 * \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{1}} + \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{1}} \times \widehat{\vec{\boldsymbol{v}}}_{\boldsymbol{2}})$$

The conjugate of a dual quaternion $\hat{A} = (A1 + \boldsymbol{\varepsilon}\, A2)$ is the quaternion $\widehat{\boldsymbol{A}}^* = (\boldsymbol{A1}^* + \boldsymbol{\varepsilon}\ \boldsymbol{A2}^*)$, where $A1$, $A2$ are quaternions, and $A1^*$, $A2^*$, are the conjugate of the respective quaternions. We recall that the conjugate of the quaternion $q = (q_0, \vec{q}_v)$ is the quaternion $q^* = (q_0, -\vec{q}_v)$.

In Matlab, quaternions are part of the official packages. However, dual quaternions is not, so we add one of the proposed implementations, e.g. the “Robotics Dual Quaternion Toolbox”



Using this package, we define two dual quaternions:

```
clear all;
clc;
close all;

dq1 = [1 2 3 4 5 6 7 8];
dqdisp(dq1);
dqdisp(dqconj(dq1));

dq2 = [2 1 4 7 1 2 3 1];
dqdisp(dq2);

dq3 = dqmultiply(dq1, dq2);
dqdisp(dq3);

% scalar part = s1 * s2 - qv1' * qv2
% vector part = s1 * qv2 + s2 * qv1 + cross(qv1, qv2)
s1 = myDualScalarPart(dq1);
qv1 = myDualVectorPart(dq1);
s2 = myDualScalarPart(dq2);
qv2 = myDualVectorPart(dq2);

dq3Scalar = myDualSum(myDualProd(s1, s2), -myDualDotProd(qv1, qv2))

dq3Vect = myDualSumVect(myDualSumVect(myDualProdVect(s1, qv2), ...
                                      myDualProdVect(s2, qv1)), ...
                        myDualCrossVect(qv1, qv2))

dq31 = myDualQuat(dq3Scalar, dq3Vect);
dqdisp(dq31);
%%
function dq = myDualQuat(dqScalar, dqVector)
  dq = [dqScalar; dqVector];
  dq = dq(:);
end
%%
function d = myDualProd(d1, d2) % sum of dual numbers
  d = [d1(1) * d2(1), d1(1) * d2(2) + d2(1) * d1(2)];
end
%%
function d = myDualSum(d1, d2) % product of dual numbers
  d = d1 + d2;
end
%%
function dv = myDualProdVect(s1, dv1) % product of dual number by dual vector
```

```
  dv = zeros(3, 2);

  for i = 1:3
    dv(i, :) = myDualProd(s1, dv1(i, :));
  end
end
%%
function dv = myDualSumVect(dv1, dv2)
  dv = dv1 + dv2;
end
%%
function dv = myDualCrossVect(dv1, dv2)
  dv = zeros(3, 2);

  dv(1, :) = myDualSumVect(myDualProd(dv1(2, :), dv2(3, :)), ...
                           - myDualProd(dv1(3, :), dv2(2, :)));
  dv(2, :) = myDualSumVect(myDualProd(dv1(3, :), dv2(1, :)), ...
                           - myDualProd(dv1(1, :), dv2(3, :)));
  dv(3, :) = myDualSumVect(myDualProd(dv1(1, :), dv2(2, :)), ...
                           - myDualProd(dv1(2, :), dv2(1, :)));
end

%%
function dv = myDualDotProd(dv1, dv2)
  dv = zeros(1, 2);

  dv(1, 1) = dv1(:, 1)' * dv2(:, 1);
  dv(1, 2) = dv1(:, 1)' * dv2(:, 2) + dv1(:, 2)' * dv2(:, 1);
end

%%
function sd = myDualScalarPart(qd)
  sd = [qd(1), qd(5)];
end
%%
function vd = myDualVectorPart(qd)
  vd = [qd(2:4)', qd(6:end)'];
end
```

(1 + 2i + 3j + 4k) + ε(5 + 6i + 7j + 8k)
(1 + -2i + -3j + -4k) + ε(5 + -6i + -7j + -8k)
(2 + 1i + 4j + 7k) + ε(1 + 2i + 3j + 1k)
(-40 + 10i + 0j + 20k) + ε(-96 + 29i + 12j + 73k)

dq3Scalar =

-40 -96

dq3Vect =

10 29
0 12
20 73

(-40 + 10i + 0j + 20k) + ε(-96 + 29i + 12j + 73k)

Unit Dual quaternions and spatial rotations and translations

Unit dual quaternions are quaternions satisfying the property:

$$\hat{q} * \hat{q}^* = \hat{q}^* * \hat{q} = (q + \varepsilon q') * (q^* + \varepsilon q'^*) = (q^* + \varepsilon q'^*) * (q + \varepsilon q') = (1 + \varepsilon 0)$$

All spatial transformation (matrix R and translation t) can be associated to a unit dual quaternion. This results in:

$$q * q^* = q^* * q = 1 \text{ (Scalar part)}$$

$$q * q'^* + q' * q^* = q^* * q' + q'^* * q = 0 \text{ (Dual part)}$$

The representation of the 4x4 homogeneous transformation $T = \begin{bmatrix} R & \vec{t} \\ 0 & 1 \end{bmatrix}$ has a corresponding dual quaternion $dq(T) = \hat{q}$ (See [Daniilidis 99] for the proof)

$$\hat{q} = \left((q_0, \vec{q}_v) + \frac{1}{2} * \varepsilon * (-\vec{q}_v^T * \vec{t}, q_0 * \vec{t} + \vec{t} \times \vec{q}_v) \right)$$

Where $(q_0, \vec{q}_v)$ is the quaternion corresponding to the rotation matrix R.

If the transformation is a pure rotation, i.e. $\vec{t} = \vec{0}$, we end up with $\hat{q}$ =$((q_0, \vec{q}_v) + 0 * \varepsilon)$. If the transformation is a pure translation, that is, **θ = 0** and $(q_0, \vec{q}_v) = (1, \vec{0})$, we end up with $\hat{q} = ((1, \vec{0}) + \frac{1}{2} * \varepsilon * (0, \vec{t}))$.

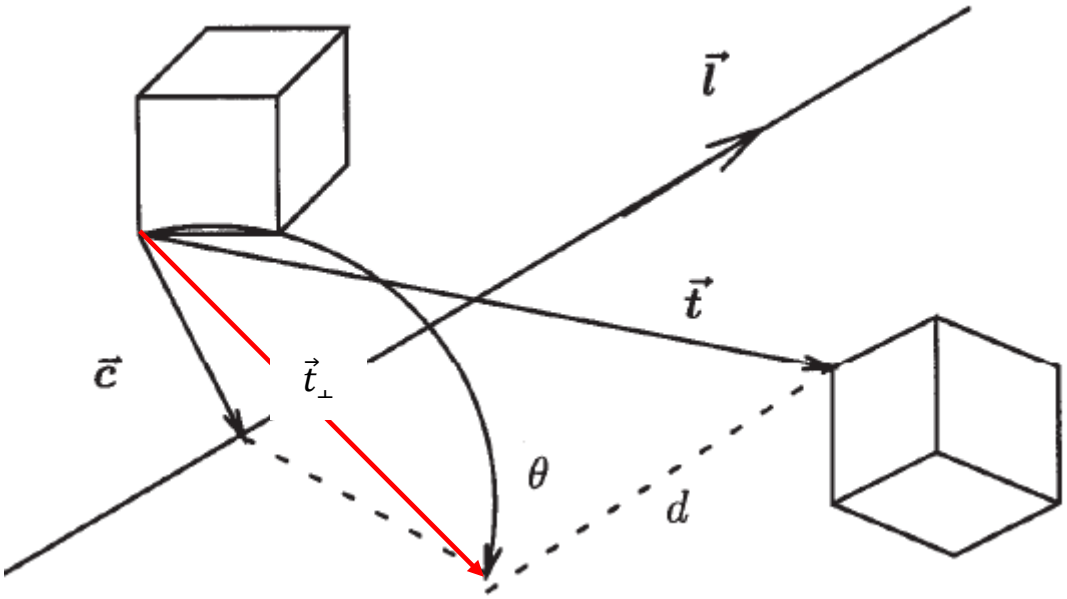
Let´s do an example with the Dual quaternion ToolBox:

```
% Define a Rotation
R = rotx(0.1) * roty(-0.2) * rotz(0.3);
qR = rotm2quat(R);
t = [100; 200; 300];

T = [R, t; 0 0 0 1]

dqR = [qR, 1/2 * [-qR(2:4) * t, qR(1) * t' + cross(t', qR(2:4)')]]

dqR1 = tform2dq(T) % Extract dq from T using the toolbox
dq2tform(dqR)      % Get T from dq using toolbox
```

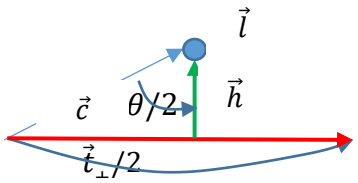
```
T =
    0.9363   -0.2896   -0.1987  100.0000
    0.2751    0.9564   -0.0978  200.0000
    0.2184    0.0370    0.9752  300.0000
         0         0         0    1.0000

dqR =
    0.9833    0.0343   -0.1060    0.1436  -12.6473   79.4277   96.2968
138.7740

dqR1 =
    0.9833    0.0343   -0.1060    0.1436  -12.6473   79.4277   96.2968
138.7740


    0.9363   -0.2896   -0.1987  100.0000
    0.2751    0.9564   -0.0978  200.0000
    0.2184    0.0370    0.9752  300.0000
         0         0         0    1.0000
```

Relationship between dual quaternions and screw theory

We saw that there is a relationship between normal quaternions and the axis-angle representation, as $\boldsymbol{q} = [\mathbf{cos}\left(\frac{\boldsymbol{\theta}}{2}\right), \vec{\boldsymbol{e}} * \mathbf{sin}\left(\frac{\boldsymbol{\theta}}{2}\right)]$. The normal question is if there exists a similar relationship related to the dual quaternions. The answer is yes, and the relationship is to the screw representing a rotation and a translation (In forward kinematics, the screw correspond to the direction of the axis of rotation and the moment of the axis of rotation). Instead, we use a theorem, called Chasles' theorem, or Mozzi–Chasles' theorem, that says that the most general rigid body displacement can be produced by a translation along a line (called its screw axis or Mozzi axis) followed (or preceded) by a rotation about an axis parallel to that line." (https://en.wikipedia.org/wiki/Chasles%27_theorem_(kinematics)).



In the Figure, the cube is rotated around the line $\vec{\boldsymbol{l}}$ **of an angle** $\boldsymbol{\theta}$ followed by a translation $\boldsymbol{d} * \vec{\boldsymbol{l}}$, $\boldsymbol{d} = \vec{\boldsymbol{l}}^{\boldsymbol{T}} * \vec{\boldsymbol{t}}$. The line $\vec{\boldsymbol{l}}$ **and angle** $\boldsymbol{\theta}$ correspond to the **axis-angle representation of the rotation**. In the Figure, the vector $\vec{c}$ corresponds to the perpendicular of the origin of the first frame to the line $\vec{l}$. Also, $\vec{\boldsymbol{t}}_{\perp} = \vec{\boldsymbol{t}} - \boldsymbol{d} * \vec{\boldsymbol{l}} = \vec{\boldsymbol{t}} - (\vec{\boldsymbol{l}}^{\boldsymbol{T}} * \vec{\boldsymbol{t}}) * \vec{\boldsymbol{l}}$. If we represent the rotation around the axis $\vec{\boldsymbol{l}}$ as it follows:



Then, we get that:

$$\mathbf{cot}\left(\frac{\boldsymbol{\theta}}{2}\right) = \frac{\|\vec{\boldsymbol{t}}_{\perp}\|}{2 * \|\vec{\boldsymbol{h}}\|}; \quad \|\vec{\boldsymbol{h}}\| = \frac{\mathbf{1}}{\mathbf{2}} * \mathbf{cot}\left(\frac{\boldsymbol{\theta}}{2}\right) * \|\vec{\boldsymbol{t}}_{\perp}\|; \; \vec{\boldsymbol{h}} = \frac{\mathbf{1}}{\mathbf{2}} * \mathbf{cot}\left(\frac{\boldsymbol{\theta}}{2}\right) * (\vec{\boldsymbol{l}} \times \vec{\boldsymbol{t}})$$

$$\vec{c} = \frac{\vec{t}_{\perp}}{2} + \vec{h} = \frac{1}{2} * \left(\vec{t} - \left(\vec{l}^{T} * \vec{t} \right) * \vec{l} + \cot\left(\frac{\theta}{2} \right) * \left(\vec{l} \times \vec{t} \right) \right)$$

$$\vec{c} = \frac{1}{2} * \left(\vec{t} - d * \vec{l} + \cot\left(\frac{\theta}{2} \right) * \left(\vec{l} \times \vec{t} \right) \right)$$

The screw corresponding to the Plucker coordinates of the line is given by $\boldsymbol{S} = \left(\vec{l}, \quad \vec{m} \right) = \left(\vec{l}, \quad \vec{c} \times \vec{l} \right)$. More about screw theory in (https://en.wikipedia.org/wiki/Screw_theory).

The relationship to the precious screw is given by (see [Daniilidis 99]):

$$\hat{q}(\boldsymbol{R}, \boldsymbol{T}) = \left[\cos\left(\frac{\theta + \varepsilon * d}{2} \right), \sin\left(\frac{\theta + \varepsilon * d}{2} \right) * \left(\vec{l} + \varepsilon * m \right) \right] = \left[\cos\frac{\left(\hat{\theta} \right)}{2}, \hat{\vec{l}} * \sin\frac{\left(\hat{\theta} \right)}{2} \right].$$

Where: $\cos\left(\frac{\theta + \varepsilon * d}{2} \right) = \cos\left(\frac{\theta}{2} \right) - \varepsilon * \frac{d}{2} * \sin\left(\frac{\theta}{2} \right)$, $\sin\left(\frac{\theta + \varepsilon * d}{2} \right) = \sin\left(\frac{\theta}{2} \right) + \varepsilon * \frac{d}{2} * \cos\left(\frac{\theta}{2} \right)$

$\hat{\vec{l}} = \left(\vec{l}, \quad \vec{m} \right)$ and $\hat{\theta} = (\theta + \varepsilon * d)$.

Example in Matlab for the previous T = [R, t; 0 0 0 1] value

```
% Get the axis-angle of R
axAng = rotm2axang(R);
theta = axAng(4);
axis = axAng(1:3)';

d = axis' * t;
c = 1/2 * (t - d * axis + cot(theta / 2) * cross(axis, t));
m = cross(c, axis);

dqrealQuadRT = [cos(theta / 2), - d/2 * sin(theta / 2)];
dqvectQuadRT = myDualProdVect([sin(theta / 2), + d/2 * cos(theta /
2)], ...
                              [axis, m]);
dqdisp(myDualQuat(dqrealQuadRT, dqvectQuadRT))
```

(0.98335 + 0.034271i + -0.10602j + 0.14357k) + ε(-12.6473 + 79.4277i + 96.2968j + 138.774k)

We can realize that we get exactly the previous value!!!

```
dqR =
    0.9833    0.0343   -0.1060    0.1436  -12.6473   79.4277   96.2968
138.7740
```

Hand-Eye estimation with Dual Quaternions

In order to solve the hand-eye with quaternions, it is necessary to solve the equation:

$$\boldsymbol{T_A * T_X = T_X * T_B}$$

Where $T_A = \begin{bmatrix} R_A & P_A \\ 0 & 1 \end{bmatrix}, T_B = \begin{bmatrix} R_B & P_B \\ 0 & 1 \end{bmatrix}, T_X = \begin{bmatrix} R_X & P_X \\ 0 & 1 \end{bmatrix}$.

Using dual unit quaternions, this equation can be expressed as:

$$\boldsymbol{\hat{q}_A * \hat{q}_X = \hat{q}_X * \hat{q}_B}$$

Similar properties as the quaternions follow, e.g. the opposite of a dual quaternion corresponds to the same transformation T.

Let´s do one example in Matlab:

```
% Give Tx, TA
TX = T;
RA = rotx(0.5) * roty(1.2) * rotz(0.1);
TA = [RA, [-200; 300; -500]; 0 0 0 1];
TB = inv(TX) * TA * TX;

dqA = tform2dq(TA);
dqB = tform2dq(TB);

dqdisp(dqmultiply(dqA, dqR))
dqdisp(dqmultiply(dqR, dqB))
```

```
(0.80167 + 0.35057i + 0.41628j + 0.24727k) + ε(-70.4932 + 178.0779i + 108.9505j + -207.3512k)
(0.80167 + 0.35057i + 0.41628j + 0.24727k) + ε(-70.4932 + 178.0779i + 108.9505j + -207.3512k)
```

```
% Give the transform from dqR and -dqR
dq2tform(dqR)
dq2tform(- dqR)
ans =
    0.9363   -0.2896   -0.1987  100.0000
    0.2751    0.9564   -0.0978  200.0000
    0.2184    0.0370    0.9752  300.0000
         0         0         0    1.0000
ans =
    0.9363   -0.2896   -0.1987  100.0000
    0.2751    0.9564   -0.0978  200.0000
    0.2184    0.0370    0.9752  300.0000
         0         0         0    1.0000
```

This equation is linear on the dual quaternion parameters of $\hat{q}_X$. As in the case of the single quaternions there are the possibilities of having a dual quaternion or the opposite one for $\hat{q}_A$ and $\hat{q}_B$. We choose the sign in order to have as close values for the non-dual q_0 part of $\hat{q}_A$ and $\hat{q}_B$. Further simplifications (see [1]), lead to the solution of the equation:

$$\begin{bmatrix} \overrightarrow{q_A} - \overrightarrow{q_B} & [\overrightarrow{q_A} + \overrightarrow{q_B}]_\times & \mathbf{0}_{3x1} & \mathbf{0}_{3x3} \\ \overrightarrow{q'_A} - \overrightarrow{q'_B} & [\overrightarrow{q'_A} + \overrightarrow{q_{'B}}]_\times & \overrightarrow{q_A} - \overrightarrow{q_B} & [\overrightarrow{q_A} + \overrightarrow{q_B}]_\times \end{bmatrix} * \begin{bmatrix} q_{0X} & \overrightarrow{q_X} & q'_{0X} & \overrightarrow{q'_X} \end{bmatrix}^T = \boldsymbol{C * \widehat{q_x} = \vec{0}}$$

Where $\widehat{\boldsymbol{q}}_{\boldsymbol{A}} = (\boldsymbol{q}_{\boldsymbol{0A}}, \overrightarrow{\boldsymbol{q}_{\boldsymbol{A}}}) + \boldsymbol{\varepsilon}\,(\boldsymbol{q'}_{\boldsymbol{0A}}, \overrightarrow{\boldsymbol{q'}_{\boldsymbol{A}}})$, $\widehat{\boldsymbol{q}}_{\boldsymbol{B}} = (\boldsymbol{q}_{\boldsymbol{0B}}, \overrightarrow{\boldsymbol{q}_{\boldsymbol{B}}}) + \boldsymbol{\varepsilon}\,(\boldsymbol{q'}_{\boldsymbol{0B}}, \overrightarrow{\boldsymbol{q'}_{\boldsymbol{B}}})$, $\widehat{\boldsymbol{q}}_{\boldsymbol{X}} = (\boldsymbol{q}_{\boldsymbol{0X}}, \overrightarrow{\boldsymbol{q}_{\boldsymbol{X}}}) + \boldsymbol{\varepsilon}\,(\boldsymbol{q'}_{\boldsymbol{0X}}, \overrightarrow{\boldsymbol{q'}_{\boldsymbol{X}}})$.

Besides the previous equation, the solution $\widehat{q_x}$ should be a unit dual quaternion. In fact, the solution of the equation $C * \widehat{q_x} = \vec{0}$ has the last singular values equal to zero, implying the restriction to be a unit dual quaternion. In the paper of Daniilidis [1], this problem is solved with the following algorithm:

Algorithm to compute the solution of unit dual quaternion $\widehat{q_x}$

1. Compute the SVD of the matrix C, i.e. [U, S, V] = svd(C), and check if only two singular values are almost equal to zero (due to noise), applying a threshold. Take the corresponding vectors v_7 = V(:, 7) and v_8 = V(:, 8). The solution of $\widehat{q_x} = \lambda_1 * v_7 + \lambda_2 * v_8$. The values of λ_1 and λ_2 are chosen in order to make the solution a unit dual quaternion. The equations of the unit dual quaternion constraints are:

$$\lambda_1^2 * u_1^T * u_1 + 2 * \lambda_1 * \lambda_2 * u_1^T * u_2 + \lambda_2^2 * u_2^T * u_2 = 1$$
$$\lambda_1^2 * u_1^T * v_1 + \lambda_1 * \lambda_2 * (u_1^T * v_2 + u_2^T * v_1) + \lambda_2^2 * u_2^T * v_2 = 0$$

   Where $v_7 = \begin{bmatrix} u_1 \\ v_1 \end{bmatrix}$ and $v_8 = \begin{bmatrix} u_2 \\ v_2 \end{bmatrix}$

2. This equation is solved by defining $\lambda_1 = s * \lambda_2$. Substituting in the previous equation, it results a second order equation in s, giving two possible values of s:

$$s^2 * u_1^T * v_1 + s * (u_1^T * v_2 + u_2^T * v_1) + u_2^T * v_2 = 0$$

3. The first equation can be put in the form:

$$\lambda_2^2 * (s^2 * u_1^T * u_1 + 2 * s * u_1^T * u_2 + u_2^T * u_2) = 1$$

From the two possible values of s, the values of $(s^2 * u_1^T * u_1 + 2 * s * u_1^T * u_2 + u_2^T * u_2)$ can be 0 and a positive value in absence of noise. Thus we choose the value of s that maximize the value of $(s^2 * u_1^T * u_1 + 2 * s * u_1^T * u_2 + u_2^T * u_2)$.

$$\lambda_2 = \sqrt{1/(s^2 * u_1^T * u_1 + 2 * s * u_1^T * u_2 + u_2^T * u_2)}$$
$$\lambda_1 = s * \lambda_2$$

4. Make $\widehat{q_x} = \lambda_1 * v_7 + \lambda_2 * v_8$

Relationship quaternions and Dual quaternions

$$q = q_0 + q_1 * i + q_2 * j + q_3 * k$$

$$\hat{q} = (q_0 + q_1 * i + q_2 * j + q_3 * k) + \varepsilon * (q_{d0} + q_{d1} * i + q_{d2} * j + q_{d3} * k)$$

$q = (q_0, \vec{q}_v)$ $\quad$ $\hat{q} = (\hat{q}_0, \vec{\hat{q}}_v)$

$q^* = (q_0, -\vec{q}_v)$ $\quad$ $\hat{q}^* = (\hat{q}_0, -\vec{\hat{q}}_v)$

$q * q^* = q^* * q = (1, \vec{0})$ (unit q) $\quad$ $\hat{q} * \hat{q}^* = \hat{q}^* * \hat{q} = (\hat{1}, \vec{\hat{0}})$ (unit dual q)

$Ra * Rb => q_a * q_b$ $\quad$ $Ta * Tb => \hat{q}_a * \hat{q}_b$

$q = (\cos\left(\frac{\theta}{2}\right), \vec{v} * \sin(\frac{\theta}{2})$ $\quad$ $\hat{q} = (\cos\left(\frac{\hat{\theta}}{2}\right), \vec{\hat{v}} * \sin(\frac{\hat{\theta}}{2})$

Conclusions

According to [1] and other research works in Robotics, Computer Vision and Computer Graphics, the usage of dual quaternions for estimation of the pose (rotation and translations) results in smaller identification errors that using separate representations of the rotation and translations. As an example, the commercial software Halcon for image processing and computer Vision, utilizes dual quaternions for the identification of the hand-eye.

Bibliography

[Daniilidis 99] “Hand-Eye Calibration Using Dual Quaternions”, Kostas Daniilidis, 1 March 1999, The International Journal of Robotics Research.