Algebra

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 Algebra Namespace Reference

Is Linear Algebra library of required mathematics for, Kinetics, and Kinematics mechanics.

Functions

- bool Epsilon (const double val)

  Find if the value is negligible, or close to zero.
- Eigen::Matrix3d VecToso3 (const Eigen::Vector3d &omg)

  Get the skew symmetric matrix representation of a vector $v = \begin{vmatrix} v_1 & v_2 & v_3 \end{vmatrix}$ $skew(v) = \begin{vmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & 1v_1 \\ -v_2 & v_1 & 0 \end{vmatrix}$.
- Eigen::Vector3d so3ToVec (const Eigen::MatrixXd &so3mat)

  Returns vector represented by the skew symmetric matrix.
- Eigen::MatrixXd RpToTrans (const Eigen::Matrix3d &R, const Eigen::Vector3d &p)

  Combines a rotation matrix and position vector into a single Special Euclidian Group (SE3) homogeneous transformation matrix.
- std::vector< Eigen::MatrixXd > TransToRp (const Eigen::MatrixXd &T)

  Separates the rotation matrix and position vector from the transfomation matrix representation.
- Eigen::MatrixXd VecTose3 (const Eigen::VectorXd &V)

  Translates a spatial velocity vector into a transformation matrix.
- Eigen::VectorXd se3ToVec (const Eigen::MatrixXd &T)

  Translates a transformation matrix into a spatial velocity vector.
- Eigen::MatrixXd Normalize (Eigen::MatrixXd V)

  Returns a normalized version of the input vector.
- Eigen::Vector4d AxisAng3 (const Eigen::Vector3d &expc3)
- Eigen::Matrix3d MatrixExp3 (const Eigen::Matrix3d &so3mat)

  Translates an exponential rotation into a rotation matrix using rodrigues formula.
- Eigen::Matrix3d MatrixLog3 (const Eigen::Matrix3d &R)

  Function: Computes the matrix logarithm of a rotation matrix.
- Eigen::MatrixXd Adjoint (const Eigen::MatrixXd &T)

  Provides the adjoint representation of a transformation matrix Used to change the frame of reference for spatial velocity vectors.
- Eigen::MatrixXd MatrixExp6 (const Eigen::MatrixXd &se3mat)

  Rotation expanded for screw axis.

- Eigen::MatrixXd MatrixLog6 (const Eigen::MatrixXd &T)

  Computes the matrix logarithm of a homogeneous transformation matrix.
- Eigen::MatrixXd TransInv (const Eigen::MatrixXd &transform)

  Inverts a homogeneous transformation matrix.
- Eigen::MatrixXd RotInv (const Eigen::MatrixXd &rotMatrix)

  Inverts a rotation matrix.
- Eigen::VectorXd ScrewToAxis (Eigen::Vector3d q, Eigen::Vector3d s, double h)

  Takes a parametric description of a screw axis and converts it to a normalized screw axis.
- Eigen::VectorXd AxisAng6 (const Eigen::VectorXd &expc6)

  Converts a 6-vector of exponential coordinates into screw axis-angle form.
- Eigen::MatrixXd ProjectToSO3 (const Eigen::MatrixXd &M)

  Returns a projection of mat into SO(3)
- Eigen::MatrixXd ProjectToSE3 (const Eigen::MatrixXd &M)

  Returns a projection of mat into SE(3)
- double DistanceToSO3 (const Eigen::Matrix3d &M)

  Returns the Frobenius norm to describe the distance of mat from the SO(3) manifold.
- double DistanceToSE3 (const Eigen::Matrix4d &T)

  Returns the Frobenius norm to describe the distance of mat from the SE(3) manifold.
- bool TestIfSO3 (const Eigen::Matrix3d &M)
- bool TestIfSE3 (const Eigen::Matrix4d &T)
- Eigen::MatrixXd ad (Eigen::VectorXd V)

  Calculate the 6x6 matrix of the given 6-vector V.

3.1.1 Detailed Description

Is Linear Algebra library of required mathematics for, Kinetics, and Kinematics mechanics.

Build on top of Kevin M. Lynch's, and Frank C. Park 'Modern Robotics' Book http://hades.mech.↩northwestern.edu/index.php/Modern_Robotics, and courses from Princeton.

3.1.2 Function Documentation

3.1.2.1 ad()

```
Eigen::MatrixXd Algebra::ad (
            Eigen::VectorXd V )  [inline]
```

Calculate the 6x6 matrix of the given 6-vector V.

Parameters

V	Eigen::VectorXd (6x1), V=[omg, v]

Returns

Eigen::MatrixXd (6x6) Note: Can be used to calculate the Lie bracket [V1, V2] = [adV1]V2

3.1.2.2 Adjoint()

```
Eigen::MatrixXd Algebra::Adjoint (
            const Eigen::MatrixXd & T )  [inline]
```

Provides the adjoint representation of a transformation matrix Used to change the frame of reference for spatial velocity vectors.

Parameters

T	4x4 Transformation matrix SE(3) $\begin{vmatrix} R & p \\ 0 & 1 \end{vmatrix}$.

Returns

6x6 Adjoint Representation of the matrix T $adj(T) = \begin{vmatrix} R & 0 \\ [p]R & R \end{vmatrix}$

3.1.2.3 AxisAng3()

```
Eigen::Vector4d Algebra::AxisAng3 (
            const Eigen::Vector3d & expc3 )  [inline]
```

3.1.2.4 AxisAng6()

```
Eigen::VectorXd Algebra::AxisAng6 (
            const Eigen::VectorXd & expc6 )  [inline]
```

Converts a 6-vector of exponential coordinates into screw axis-angle form.

Parameters

expc6	A 6-vector of exponential coordinates for rigid-body motion S$*$theta

Returns

S The corresponding normalized screw axis

theta The distance traveled along/about S

3.1.2.5 DistanceToSE3()

```
double Algebra::DistanceToSE3 (
            const Eigen::Matrix4d & T )  [inline]
```

Returns the Frobenius norm to describe the distance of mat from the SE(3) manifold.

Parameters

mat	A 4x4 matrix

Returns

A quantity describing the distance of mat from the SE(3) manifold Computes the distance from mat to the SE(3) manifold using the following method: Compute the determinant of matR, the top 3x3 submatrix of mat. If det(matR) $<=$ 0, return a large number. If det(matR) $>$ 0, replace the top 3x3 submatrix of mat with mat↩ R^T.matR, and set the first three entries of the fourth column of mat to zero. Then return norm(mat - I).

3.1.2.6 DistanceToSO3()

```
double Algebra::DistanceToSO3 (
            const Eigen::Matrix3d & M )  [inline]
```

Returns the Frobenius norm to describe the distance of mat from the SO(3) manifold.

Parameters

M	A 3x3 matrix

Returns

A quantity describing the distance of mat from the SO(3) manifold Computes the distance from mat to the SO(3) manifold using the following method: If det(mat) $<=$ 0, return a large number. If det(mat) $>$ 0, return norm(mat^T.mat - I).

3.1.2.7 Epsilon()

```
bool Algebra::Epsilon (
            const double val )  [inline]
```

Find if the value is negligible, or close to zero.

Parameters

val	double value to be checked.

Returns

Boolean of true-ignore or false-can't ignore.

3.1.2.8 MatrixExp3()

```
Eigen::Matrix3d Algebra::MatrixExp3 (
            const Eigen::Matrix3d & so3mat )  [inline]
```

Translates an exponential rotation into a rotation matrix using rodrigues formula.

Parameters

so3mat	exponenential representation of a rotation in so3 $[w]\theta$.

Returns

Rotation matrix $e^{[w]\theta}$.

3.1.2.9 MatrixExp6()

```
Eigen::MatrixXd Algebra::MatrixExp6 (
            const Eigen::MatrixXd & se3mat )  [inline]
```

Rotation expanded for screw axis.

Parameters

se3mat	se3 matrix representation of exponential coordinates $[s]\theta$

Returns

6x6 Matrix exponential Transformation $T = e^{[s]\theta}$

3.1.2.10 MatrixLog3()

```
Eigen::Matrix3d Algebra::MatrixLog3 (
            const Eigen::Matrix3d & R )  [inline]
```

Function: Computes the matrix logarithm of a rotation matrix.

Parameters

R	Rotation matrix $[w]\theta$.

Returns

matrix logarithm of a rotation $e^{[w]\theta}$.

3.1.2.11 MatrixLog6()

```
Eigen::MatrixXd Algebra::MatrixLog6 (
            const Eigen::MatrixXd & T )  [inline]
```

Computes the matrix logarithm of a homogeneous transformation matrix.

Parameters

T	A matrix in SE3.

Returns

The matrix logarithm of $e^{[s]\theta}$.

3.1.2.12 Normalize()

```
Eigen::MatrixXd Algebra::Normalize (
            Eigen::MatrixXd V )  [inline]
```

Returns a normalized version of the input vector.

Parameters

V	Eigen::MatrixXd

Returns

Eigen::MatrixXd normalized V Note: MatrixXd is used instead of VectorXd for the case of row vectors Requires a copy Useful because of the MatrixXd casting

3.1.2.13 ProjectToSE3()

```
Eigen::MatrixXd Algebra::ProjectToSE3 (
            const Eigen::MatrixXd & M )  [inline]
```

Returns a projection of mat into SE(3)

Parameters

M	A 4x4 matrix to project to SE(3)

Returns

The closest matrix to T that is in SE(3) Projects a matrix mat to the closest matrix in SE(3) using singular-value decomposition (see http://hades.mech.northwestern.edu/index.php/Modern_Robotics_Linear_Algebra_Review). This function is only appropriate for matrices close to SE(3).

3.1.2.14 ProjectToSO3()

```
Eigen::MatrixXd Algebra::ProjectToSO3 (
            const Eigen::MatrixXd & M )  [inline]
```

Returns a projection of mat into SO(3)

Parameters

M	A matrix near SO(3) to project to SO(3)

Returns

The closest matrix to R that is in SO(3) Projects a matrix mat to the closest matrix in SO(3) using singular-value decomposition (see http://hades.mech.northwestern.edu/index.php/Modern_Robotics_Linear_Algebra_Review). This function is only appropriate for matrices close to SO(3).

3.1.2.15 RotInv()

```
Eigen::MatrixXd Algebra::RotInv (
            const Eigen::MatrixXd & rotMatrix )  [inline]
```

Inverts a rotation matrix.

Parameters

R	A rotation matrix

Returns

: The inverse of R

3.1.2.16 RpToTrans()

```
Eigen::MatrixXd Algebra::RpToTrans (
            const Eigen::Matrix3d & R,
            const Eigen::Vector3d & p )  [inline]
```

Combines a rotation matrix and position vector into a single Special Euclidian Group (SE3) homogeneous transformation matrix.

Parameters

R	Rotation Matrix
p	Position Vector

See also

TransToRp

Returns

Transformation Matrix $\begin{vmatrix} R & p \\ 0 & 1 \end{vmatrix}$.

3.1.2.17 ScrewToAxis()

```
Eigen::VectorXd Algebra::ScrewToAxis (
            Eigen::Vector3d q,
            Eigen::Vector3d s,
            double h )  [inline]
```

Takes a parametric description of a screw axis and converts it to a normalized screw axis.

Parameters

q	A point lying on the screw axis
s	A unit vector in the direction of the screw axis
h	The pitch of the screw axis

Returns

A normalized screw axis described by the inputs

3.1.2.18 se3ToVec()

```
Eigen::VectorXd Algebra::se3ToVec (
            const Eigen::MatrixXd & T )  [inline]
```

Translates a transformation matrix into a spatial velocity vector.

Parameters

T	Transformation matrix $\begin{vmatrix} [w] & v \\ 0 & 0 \end{vmatrix}$

See also

VecTose3

Returns

Spatial velocity vector $\begin{vmatrix} w \\ v \end{vmatrix}$

3.1.2.19 so3ToVec()

```
Eigen::Vector3d Algebra::so3ToVec (
            const Eigen::MatrixXd & so3mat )  [inline]
```

Returns vector represented by the skew symmetric matrix.

$$v = \begin{vmatrix} v_1 & v_2 & v_3 \end{vmatrix} \; skew(v) = \begin{vmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & 1v_1 \\ -v_2 & v_1 & 0 \end{vmatrix}$$

Parameters

so3mat	Eigen::MatrixXd 3x3 skew symmetric matrix in so3

See also

VecToso3

Returns

v_ret Eigen::Vector3d 3x1 vector

3.1.2.20 TestIfSE3()

```
bool Algebra::TestIfSE3 (
            const Eigen::Matrix4d & T )  [inline]
```

3.1.2.21 TestIfSO3()

```
bool Algebra::TestIfSO3 (
            const Eigen::Matrix3d & M )  [inline]
```

3.1.2.22 TransInv()

```
Eigen::MatrixXd Algebra::TransInv (
            const Eigen::MatrixXd & transform )  [inline]
```

Inverts a homogeneous transformation matrix.

Parameters

T	A homogeneous transformation matrix

Returns

The inverse of T Uses the structure of transformation matrices to avoid taking a matrix inverse, for efficiency.

3.1.2.23 TransToRp()

```
std::vector<Eigen::MatrixXd> Algebra::TransToRp (
            const Eigen::MatrixXd & T )  [inline]
```

Separates the rotation matrix and position vector from the transfomation matrix representation.

Parameters

T	Homogeneous transformation matrix $\begin{vmatrix} R & p \\ 0 & 1 \end{vmatrix}$.

See also

RpToTrans

Returns

Rp_ret std::vector of rotation matrix (R), and position vector (p).

3.1.2.24 VecTose3()

```
Eigen::MatrixXd Algebra::VecTose3 (
            const Eigen::VectorXd & V )  [inline]
```

Translates a spatial velocity vector into a transformation matrix.

Parameters

V	Spatial velocity vector $\begin{vmatrix} w \\ v \end{vmatrix}$

See also

se3ToVec

Returns

linear representation of Twist in special Euclidian group se3 $\begin{vmatrix} [w] & v \\ 0 & 0 \end{vmatrix}$

3.1.2.25 VecToso3()

```
Eigen::Matrix3d Algebra::VecToso3 (
            const Eigen::Vector3d & omg )  [inline]
```

Get the skew symmetric matrix representation of a vector $v = \begin{vmatrix} v_1 & v_2 & v_3 \end{vmatrix}$ $skew(v) = \begin{vmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & 1v_1 \\ -v_2 & v_1 & 0 \end{vmatrix}$.

Parameters

omg	Eigen::Vector3d 3x1 angular velocity vector

See also

so3ToVec

Returns

Eigen::MatrixXd 3x3 skew symmetric matrix in so3

Chapter 4

File Documentation

4.1 algebra.hpp File Reference

```
#include <Eigen/Dense>
#include <cmath>
#include <vector>
#include <iostream>
```

Include dependency graph for algebra.hpp:

4.2 algebra_test.cpp File Reference

```
#include <iostream>
#include <Eigen/Dense>
#include "gmock/gmock.h"
#include "algebra.hpp"
```

Include dependency graph for algebra_test.cpp:

Functions

- TEST (ALGEBRA, Epsilon)
- TEST (ALGEBRA, So3ToVecTest)
- TEST (ALGEBRA, VecToSO3Test)
- TEST (ALGEBRA, RpToTrans)
- TEST (ALGEBRA, TransToRp)
- TEST (ALGEBRA, VecToso3)
- TEST (ALGEBRA, so3ToVec)
- TEST (ALGEBRA, AxisAng3)
- TEST (ALGEBRA, Adjoint)
- TEST (ALGEBRA, Adjoint2)
- TEST (ALGEBRA, MatrixLog3)
- TEST (ALGEBRA, TransInv)
- TEST (ALGEBRA, RotInv)
- TEST (ALGEBRA, ScrewToAxis)
- TEST (ALGEBRA, AxisAng6)
- TEST (ALGEBRA, MatrixLog6)
- TEST (ALGEBRA, DistanceToSO3Test)
- TEST (ALGEBRA, DistanceToSE3Test)
- TEST (ALGEBRA, TestIfSO3Test)
- TEST (ALGEBRA, TestIfSE3Test)
- TEST (ALGEBRA, adTest)
- int main (int argc, char **argv)

4.2.1 Function Documentation

4.2.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

4.2.1.2 TEST() [1/21]

```
TEST (
            ALGEBRA ,
            Adjoint  )
```

4.2.1.3 TEST() [2/21]

```
TEST (
            ALGEBRA ,
            Adjoint2  )
```

4.2.1.4 TEST() [3/21]

```
TEST (
            ALGEBRA ,
            adTest  )
```

4.2.1.5 TEST() [4/21]

```
TEST (
            ALGEBRA ,
            AxisAng3  )
```

4.2.1.6 TEST() [5/21]

```
TEST (
            ALGEBRA ,
            AxisAng6  )
```

4.2.1.7 TEST() [6/21]

```
TEST (
            ALGEBRA ,
            DistanceToSE3Test  )
```

4.2.1.8 TEST() [7/21]

```
TEST (
            ALGEBRA ,
            DistanceToSO3Test  )
```

4.2.1.9 TEST() [8/21]

```
TEST (
            ALGEBRA ,
            Epsilon  )
```

4.2.1.10 TEST() [9/21]

```
TEST (
            ALGEBRA ,
            MatrixLog3  )
```

4.2.1.11 TEST() [10/21]

```
TEST (
            ALGEBRA ,
            MatrixLog6  )
```

4.2.1.12 TEST() [11/21]

```
TEST (
            ALGEBRA ,
            RotInv  )
```

4.2.1.13 TEST() [12/21]

```
TEST (
            ALGEBRA ,
            RpToTrans  )
```

4.2.1.14 TEST() [13/21]

```
TEST (
            ALGEBRA ,
            ScrewToAxis  )
```

4.2.1.15 TEST() [14/21]

```
TEST (
            ALGEBRA ,
            so3ToVec  )
```

4.2.1.16 TEST() [15/21]

```
TEST (
            ALGEBRA ,
            So3ToVecTest  )
```

4.2.1.17 TEST() [16/21]

```
TEST (
            ALGEBRA ,
            TestIfSE3Test  )
```

4.2.1.18 TEST() [17/21]

```
TEST (
            ALGEBRA ,
            TestIfSO3Test  )
```

4.2.1.19 TEST() [18/21]

```
TEST (
            ALGEBRA ,
            TransInv  )
```

4.2.1.20 TEST() [19/21]

```
TEST (
            ALGEBRA ,
            TransToRp  )
```

4.2.1.21 TEST() [20/21]

```
TEST (
            ALGEBRA ,
            VecToso3  )
```

4.2.1.22 TEST() [21/21]

```
TEST (
            ALGEBRA ,
            VecToSO3Test  )
```

4.3 c1-c3.cpp File Reference

```
#include <iostream>
#include "algebra.hpp"
```

Include dependency graph for c1-c3.cpp:

Functions

- int main (int argc, char **args)

4.3.1 Function Documentation

4.3.1.1 main()

```
int main (
            int argc,
            char ** args )
```

4.4 c1w3.cpp File Reference

Index