Module Interface Specification for Attitude Check

Adrian Sochaniwsky

March 15, 2024

1 Revision History

Date	Version	Notes
March 15, 20241	1.0	Initial document

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/adrian-soch/attitude_check/blob/main/docs/SRS/SRS.pdf.

Contents

3 Introduction

The following document details the Module Interface Specifications for Attitude Check.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/adrian-soch/attitude_check.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Attitude Check.

Data Type	**Notation**	**Description**
real	$\mathbb{R}$	any number in (-∞, ∞)
boolean	$\mathbb{B}$	value in $[\text{false} = 0, \text{true} = 1]$
matrix	$\mathbb{R}^{m \times n}$	matrix of any number in (-∞, ∞)
vector	$\mathbb{R}^m$	column vector of any number in (-∞, ∞)
quaternion	$\mathbf{q}$	a quaternion $\in \mathbb{R}^4$, see SRS for details

The specification of Attitude Check uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Attitude Check uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2
Behaviour-Hiding Module	Control Module
	Input Verification Module
	Initial Quaternion Estimator w/o Mag Module
	Initial Quaternion Estimator w Mag Module
	Estimate w/o Mag Module
	Estimate w Mag Module
Software Decision Module	Matrix Math Module
	Quaternion Module

Table 1: Module Hierarchy

6 MIS of Control Module

6.1 Module

6.2 Uses

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
	-	-	-

6.4 Semantics

6.4.1 State Variables

6.4.2 Environment Variables

6.4.3 Assumptions

6.4.4 Access Routine Semantics

():

- transition:
- output:
- exception:

6.4.5 Local Functions

7 MIS of Estimate w/o Mag Module

7.1 Module

7.2 Uses

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
	-	-	-

7.4 Semantics

7.4.1 State Variables

7.4.2 Assumptions

7.4.3 Access Routine Semantics

():

- transition:
- output:
- exception:

7.4.4 Local Functions

8 MIS of Estimate w Mag Module

8.1 Module

8.2 Uses

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
	-	-	-

8.4 Semantics

8.4.1 State Variables

8.4.2 Assumptions

8.4.3 Access Routine Semantics

():

- transition:
- output:
- exception:

8.4.4 Local Functions

9 MIS of Initial Quaternion Estimator w/o Mag Module

9.1 Module

9.2 Uses

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
	-	-	-

9.4 Semantics

9.4.1 State Variables

9.4.2 Assumptions

9.4.3 Access Routine Semantics

():

- transition:
- output:
- exception:

9.4.4 Local Functions

10 MIS of Initial Quaternion Estimator w Mag Module

10.1 Module

10.2 Uses

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
	-	-	-

10.4 Semantics

10.4.1 State Variables

10.4.2 Assumptions

10.4.3 Access Routine Semantics

():

- transition:
- output:
- exception:

10.4.4 Local Functions

11 MIS of Input Verification Module

11.1 Module

Input

11.2 Uses

Quaternion Module

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
	-	-	-

11.4 Semantics

11.4.1 State Variables

11.4.2 Assumptions

11.4.3 Access Routine Semantics

():

- transition:
- output:
- exception:

11.4.4 Local Functions

12 MIS of Quaternion Module

12.1 Module

Quaternion

12.2 Uses

Matrix Math Module

12.3 Syntax

12.3.1 Exported Access Programs

Name	In	Out	Exceptions
create_quat	w:=$\mathbb{R}, x := \mathbb{R}, y := \mathbb{R}, z := \mathbb{R}$	-	ValueError
quat_prod	$P := \mathbf{q}, q := \mathbf{q}$	$q_{\text{out}} := \mathbf{q}$	-
normalize	-	-	-
assert_is_norm	w:=$\mathbb{R}, x := \mathbb{R}, y := \mathbb{R}, z := \mathbb{R}$	out:= $\mathbb{B}$	-

12.4 Semantics

12.4.1 State Variables

quat : $\mathbf{q}$

12.4.2 Assumptions

12.4.3 Access Routine Semantics

create_quat(w, x, y, z):

- transition: quat := $\mathbf{q}$ where $\mathbf{q} = [w, x, y, z]$
- exception: ValueError when $|\text{quat}| \neq 1$

quat_prod(p, q):

- output:

$$q_{\text{out}} := \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix}$$

- exception: none

normalize():

- transition: $\text{quat} := \left[\frac{\text{quat}_w}{d}, \frac{\text{quat}_x}{d}, \frac{\text{quat}_y}{d}, \frac{\text{quat}_z}{d}\right]$ where $d = \sqrt{\text{quat}_w^2 + \text{quat}_x^2 + \text{quat}_y^2 + \text{quat}_z^2}$
- exception: none

assert_is_norm():

- output: out:= $(1 == \sqrt{w^2 + x^2 + y^2 + z^2})$
- exception: none

13 MIS of Matrix Math Module

13.1 Module

Math

13.2 Uses

N/A

13.3 Syntax

13.3.1 Exported Access Programs

Name	In	Out	Exceptions
$*$	$\mathbb{R}^{m\times n} \times \mathbb{R}^{n\times m}$	$m := \mathbb{R}^{n\times n}$	EIGEN_STATIC_ASSERT_ERROR
$*$	$\mathbb{R}^{m\times n} \times \mathbb{R}$	$m := \mathbb{R}^{m\times n}$	EIGEN_STATIC_ASSERT_ERROR
$+$	$\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n}$	$m := \mathbb{R}^{m\times n}$	EIGEN_STATIC_ASSERT_ERROR
transpose	$\mathbb{R}^{m\times n}$	$m := \mathbb{R}^{n\times m}$	EIGEN_STATIC_ASSERT_ERROR

13.4 Semantics

13.4.1 State Variables

None.

13.4.2 Access Routine Semantics

transpose($\mathbb{R}^{m\times n}$):

- output: $m := \mathbb{R}^{m\times n}$
- exception: EIGEN_STATIC_ASSERT_ERROR

$[\mathbf{A}^T]_{i,j} = [\mathbf{A}]_{j,i}$

$\mathbb{R}^{m\times n} * \mathbb{R}^{n\times m}$:

- output: $m := \mathbb{R}^{n\times n}$
- exception: EIGEN_STATIC_ASSERT_ERROR

Let $\mathbf{A} = [a_{i,j}]_{m\times n}$ and $\mathbf{B} = [b_{i,j}]_{n\times m}$. Then $\mathbf{C} = \mathbf{A} * \mathbf{B}$ with $c_{i,j} = a_{i,0}b_{0,j} + a_{i,1}b_{1,j} ... a_{i,n}b_{n,j}$.

$\mathbb{R}^{m\times n} * \mathbb{R}$:

- output: $m := \mathbb{R}^{m\times n}$

- exception: EIGEN_STATIC_ASSERT_ERROR

Let $\mathbf{A} = [a_{i,j}]_{m\times n}$ and $k = \mathbb{R}$. Then $\mathbf{C} = \mathbf{A} * k$ with $c_{i,j} = ka_{i,j}$.

$\mathbb{R}^{m\times n} + \mathbb{R}^{m\times n}$:

- output: $m := \mathbb{R}^{m\times n}$
- exception: EIGEN_STATIC_ASSERT_ERROR

Let $\mathbf{A} = [a_{i,j}]_{m\times n}$ and $\mathbf{B} = [b_{i,j}]_{m\times n}$. Then $\mathbf{A} + \mathbf{B} = [a_{i,j} + b_{i,j}]_{m\times n}$.

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.