

Module Interface Specification for Image Feature Correspondences for Camera Calibration

Kiran Singh

March 12, 2025

1 Revision History

Date	Version	Notes
2025-03-19	1.0	Initial Release

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at <https://github.com/KiranSingh15/CAS-741-Image-Correspondences/blob/main/docs/SRS/SRS.pdf>.

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	MIS of Input Format Module	3
6.1	Module	3
6.2	Uses	3
6.3	Syntax	3
6.3.1	Exported Constants	3
6.3.2	Exported Access Programs	3
6.4	Semantics	3
6.4.1	State Variables	3
6.4.2	Environment Variables	3
6.4.3	Assumptions	3
6.4.4	Access Routine Semantics	4
6.4.5	Local Functions	4
7	MIS of Specification Parameters Module	4
7.1	Module	4
7.2	Uses	4
7.3	Syntax	4
7.3.1	Exported Constants	4
7.3.2	Exported Access Programs	4
7.4	Semantics	5
7.4.1	State Variables	5
7.4.2	Environment Variables	5
7.4.3	Assumptions	5
7.4.4	Access Routine Semantics	5
7.4.5	Local Functions	5
8	MIS of Output Format Module	5
8.1	Module	6
8.2	Uses	6
8.3	Syntax	6
8.3.1	Exported Constants	6
8.3.2	Exported Access Programs	6

8.4	Semantics	6
8.4.1	State Variables	6
8.4.2	Environment Variables	6
8.4.3	Assumptions	6
8.4.4	Access Routine Semantics	6
8.4.5	Local Functions	7
9	MIS of Output Verification Module	7
9.1	Module	7
9.2	Uses	7
9.3	Syntax	7
9.3.1	Exported Constants	7
9.3.2	Exported Access Programs	7
9.4	Semantics	7
9.4.1	State Variables	7
9.4.2	Environment Variables	8
9.4.3	Assumptions	8
9.4.4	Access Routine Semantics	8
9.4.5	Local Functions	8
10	MIS of Control Module	8
10.1	Module	8
10.2	Uses	8
10.3	Syntax	9
10.3.1	Exported Constants	9
10.3.2	Exported Access Programs	9
10.4	Semantics	9
10.4.1	State Variables	9
10.4.2	Environment Variables	9
10.4.3	Assumptions	9
10.4.4	Access Routine Semantics	9
10.4.5	Local Functions	9
11	MIS of Image Smoothing Module	10
11.1	Module	10
11.2	Uses	10
11.3	Syntax	10
11.3.1	Exported Constants	10
11.3.2	Exported Access Programs	10
11.4	Semantics	10
11.4.1	State Variables	10
11.4.2	Environment Variables	10
11.4.3	Assumptions	10

11.4.4	Access Routine Semantics	10
11.4.5	Local Functions	11
12	MIS of Keypoint Detection Module	11
12.1	Module	11
12.2	Uses	11
12.3	Syntax	11
12.3.1	Exported Constants	11
12.3.2	Exported Access Programs	11
12.4	Semantics	11
12.4.1	State Variables	11
12.4.2	Environment Variables	12
12.4.3	Assumptions	12
12.4.4	Access Routine Semantics	12
12.4.5	Local Functions	12
13	MIS of Feature Descriptor Module	12
13.1	Module	12
13.2	Uses	12
13.3	Syntax	13
13.3.1	Exported Constants	13
13.3.2	Exported Access Programs	13
13.4	Semantics	13
13.4.1	State Variables	13
13.4.2	Environment Variables	13
13.4.3	Assumptions	13
13.4.4	Access Routine Semantics	13
13.4.5	Local Functions	13
14	MIS of Feature Matching Module	14
14.1	Module	14
14.2	Uses	14
14.3	Syntax	14
14.3.1	Exported Constants	14
14.3.2	Exported Access Programs	14
14.4	Semantics	14
14.4.1	State Variables	14
14.4.2	Environment Variables	14
14.4.3	Assumptions	14
14.4.4	Access Routine Semantics	14
14.4.5	Local Functions	15

15 MIS of Image Data Structure Module	15
15.1 Module	15
15.2 Uses	15
15.3 Syntax	15
15.3.1 Exported Constants	15
15.3.2 Exported Access Programs	15
15.4 Semantics	16
15.4.1 State Variables	16
15.4.2 Environment Variables	16
15.4.3 Assumptions	16
15.4.4 Access Routine Semantics	16
15.4.5 Local Functions	16
16 MIS of Image Plot Module	16
16.1 Module	17
16.2 Uses	17
16.3 Syntax	17
16.3.1 Exported Constants	17
16.3.2 Exported Access Programs	17
16.4 Semantics	17
16.4.1 State Variables	17
16.4.2 Environment Variables	17
16.4.3 Assumptions	17
16.4.4 Access Routine Semantics	17
16.4.5 Local Functions	18
17 MIS of Feature Match Data Module	18
17.1 Module	18
17.2 Uses	18
17.3 Syntax	18
17.3.1 Exported Constants	18
17.3.2 Exported Access Programs	18
17.4 Semantics	18
17.4.1 State Variables	18
17.4.2 Environment Variables	19
17.4.3 Assumptions	19
17.4.4 Access Routine Semantics	19
17.4.5 Local Functions	19
18 MIS of Dataframe Structure Module	19
18.1 Module	19
18.2 Uses	19
18.3 Syntax	20

18.3.1	Exported Constants	20
18.3.2	Exported Access Programs	20
18.4	Semantics	20
18.4.1	State Variables	20
18.4.2	Environment Variables	20
18.4.3	Assumptions	20
18.4.4	Access Routine Semantics	20
18.4.5	Local Functions	20
19	MIS of ORB Data Structure Module	21
19.1	Module	21
19.2	Uses	21
19.3	Syntax	21
19.3.1	Exported Constants	21
19.3.2	Exported Access Programs	21
19.4	Semantics	21
19.4.1	State Variables	21
19.4.2	Environment Variables	21
19.4.3	Assumptions	21
19.4.4	Access Routine Semantics	22
19.4.5	Local Functions	22
20	MIS of [Module Name —SS]	22
20.1	Module	22
20.2	Uses	22
20.3	Syntax	22
20.3.1	Exported Constants	22
20.3.2	Exported Access Programs	22
20.4	Semantics	23
20.4.1	State Variables	23
20.4.2	Environment Variables	23
20.4.3	Assumptions	23
20.4.4	Access Routine Semantics	23
20.4.5	Local Functions	23
21	Appendix	25

3 Introduction

The following document details the Module Interface Specifications for [Fill in your project name and description —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at [provide the url for your repo —SS]

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

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 $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by the Image Feature Correspondences for Camera Calibration software.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

The specification of Image Feature Correspondences for Camera Calibration 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, Image Feature Correspondences for Camera Calibration 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
Hardware-Hiding	
Behaviour-Hiding	Input Parameters Input Format Module Specification Parameters Output Format Module Output Verification Module Control Module Image Smoothing Module Keypoint Detection Module Feature Descriptor Module Feature Matching Module
Software Decision	Sequence Data Structure Image Data Structure Module Image Plot Module Feature Match Data Module Dataframe Structure Module ORB Data Structure Module

Table 1: Module Hierarchy

6 MIS of Input Format Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use \LaTeX for hyperlinks to external documents. —SS]

6.1 Module

config

6.2 Uses

- smoothImage (Section 11)
- detectKeypoints (Section 12)
- assignDescriptors (Section 13)
- matchFeatures (Section 14)

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

6.4 Semantics

6.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

6.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

6.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

6.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

6.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

7 MIS of Specification Parameters Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

7.1 Module

specParams (Section 6)

7.2 Uses

- config (Section 6)

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

7.4 Semantics

7.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

7.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

7.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

7.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

7.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

8 MIS of Output Format Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

8.1 Module

formatOutput

8.2 Uses

- main (Section 10)

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

8.4 Semantics

8.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

8.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

8.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

8.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

8.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

9 MIS of Output Verification Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

9.1 Module

verifyOutput

9.2 Uses

- main (Section 10)

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

9.4 Semantics

9.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

9.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

9.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

9.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

9.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

10 MIS of Control Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use \LaTeX for hyperlinks to external documents. —SS]

10.1 Module

main

10.2 Uses

Not applicable.

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

10.4 Semantics

10.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

10.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

10.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

10.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

10.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

11 MIS of Image Smoothing Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

11.1 Module

smoothImage

11.2 Uses

- detectKeypoints (Section 12)

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

11.4 Semantics

11.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

11.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

11.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

11.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

11.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

12 MIS of Keypoint Detection Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hypperlinks to external documents. —SS]

12.1 Module

detectKeypoints

12.2 Uses

- assignDescriptors (Section 13)

12.3 Syntax

12.3.1 Exported Constants

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

12.4 Semantics

12.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

12.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

12.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

12.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

12.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

13 MIS of Feature Descriptor Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use \LaTeX for hyperlinks to external documents. —SS]

13.1 Module

assignDescriptors

13.2 Uses

- matchFeatures (Section 14)

13.3 Syntax

13.3.1 Exported Constants

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

13.4 Semantics

13.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

13.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

13.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

13.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

13.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

14 MIS of Feature Matching Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hypperlinks to external documents. —SS]

14.1 Module

matchFeatures

14.2 Uses

- main (Section 10)

14.3 Syntax

14.3.1 Exported Constants

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

14.4 Semantics

14.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

14.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

14.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

14.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

14.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

15 MIS of Image Data Structure Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hypperlinks to external documents. —SS]

15.1 Module

imageStruct

15.2 Uses

- smoothImage (Section 11)
- detectKeypoints (Section 12)
- assignDescriptors (Section 13)
- matchFeatures (Section 14)

15.3 Syntax

15.3.1 Exported Constants

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

15.4 Semantics

15.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

15.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

15.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

15.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

15.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

16 MIS of Image Plot Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

16.1 Module

plotImage

16.2 Uses

- main (Section 10)

16.3 Syntax

16.3.1 Exported Constants

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

16.4 Semantics

16.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

16.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

16.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

16.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

16.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

17 MIS of Feature Match Data Module

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use \LaTeX for hyperlinks to external documents. —SS]

17.1 Module

matchStruct

17.2 Uses

- formatOutput (Section 8)
- matchFeatures (Section 14)

17.3 Syntax

17.3.1 Exported Constants

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

17.4 Semantics

17.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

17.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

17.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

17.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

17.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

18 MIS of Dataframe Structure Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

18.1 Module

dataframeStruct

18.2 Uses

- formatOutput (Section 8)

18.3 Syntax

18.3.1 Exported Constants

18.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

18.4 Semantics

18.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

18.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

18.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

18.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

18.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

19 MIS of ORB Data Structure Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hypperlinks to external documents. —SS]

19.1 Module

orbStruct

19.2 Uses

- detectKeypoints (Section 12)
- assignDescriptors (Section 13)
- matchFeatures (Section 14)

19.3 Syntax

19.3.1 Exported Constants

19.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

19.4 Semantics

19.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

19.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

19.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

19.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

19.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

20 MIS of [Module Name —SS]

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

20.1 Module

[Short name for the module —SS]

20.2 Uses

20.3 Syntax

20.3.1 Exported Constants

20.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

20.4 Semantics

20.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

20.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

20.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

20.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

20.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

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>.

21 Appendix

[Extra information if required —SS]