Module Interface Specification for Image Feature Correspondences for Camera Calibration

Kiran Singh

March 21, 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

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

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

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
string	str	a sequence of characters
boolean	$\mathbb{B}$	a boolean in {0,1}
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [0, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

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

This module addresses the functional requirements as follows.

- R1
- R2
- R3
- R4

6.1 Module

config

6.2 Uses

- specParams (Section 7)

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_head_directory	-	head_path as string	noHeadFound
get_active_functions	-	mthd_img_smoothing: $\mathbb{N}$, mthd_kp_detection: $\mathbb{N}$, mthd_kp_description: $\mathbb{N}$, mthd_ft_match: $\mathbb{N}$	-
get_chosen_parameters	-	kernel_sz: $\mathbb{N}$, bin_sz: $\mathbb{N}$, patch_sz: $\mathbb{N}$, FAST_threshold: $\mathbb{N}$, std_deviation $\mathbb{N}$	-
get_img_names	head_path as str	img_names as str^n	-
check_limits	kernel_sz: $\mathbb{N}$, bin_sz: $\mathbb{N}$, patch_sz: $\mathbb{N}$, FAST_threshold: $\mathbb{N}$, std_deviation $\mathbb{N}$	-	invalid_parameters

6.4 Semantics

6.4.1 State Variables

- kernel_sz: $\mathbb{Z}$
- std_deviation: $\mathbb{R}$
- FAST_threshold: $\mathbb{Z}$
- bin_sz: $\mathbb{Z}$
- patch_sz: $\mathbb{Z}$
- mthd_img_smoothing: $\mathbb{Z}$
- mthd_kp_detection: $\mathbb{Z}$
- mthd_kp_description: $\mathbb{Z}$
- mthd_ft_match: $\mathbb{Z}$

tuple of methods and parameters goes here.
set the state as the defaults,
then set the state as the user defined methods, if available

6.4.2 Environment Variables

- head_path as str

6.4.3 Assumptions

none

6.4.4 Access Routine Semantics

get_head_directory():

- output: head_path = Path(os.getcwd()) where head_path defined as a member of the Python Path Class.
- exception exc:= none

get_active_functions():

- output: out := [mthd_img_smoothing, mthd_kp_detection, mthd_kp_description, mthd_ft_match]

get_chosen_parameters():

- output: [kernel_sz, bin_sz, patch_sz, FAST_threshold, std_deviation]
- exception exc:= none

get_img_names(head_path as str):

```
img_path = Path(head_path + "Raw_Images")
img_dir = Path(img_path)
input_img = [(file.stem, file.suffix, file.name) for file in img_dir.iterdir() if file.is_file()]
num_images = len(input_img)
```

- output: input_img $\in str^n$, num_images $\in \mathbb{N}$
- exception: none

check_limits():

- output: out:= none
- exception: exc:=invalid_parameters

$\neg(kernel\_sz < 1)$	$\Rightarrow$ "badKernelSize"
$\neg(kernel\_sz > 15)$	$\Rightarrow$ "badKernelSize"
$\neg(kernel\_sz \ \% \ 2 \neq 0)$	$\Rightarrow$ "badKernelSize"
$\neg(0 < std\_deviation < 10)$	$\Rightarrow$ "badStdDeviation"
$\neg(2 \leq FAST\_threshold \leq 255)$	$\Rightarrow$ "badFASTThreshold"
$\neg(1 \leq bin\_sz \leq 2048)$	$\Rightarrow$ "badBinSize"
$\neg(5 \leq patch\_sz \leq 100)$	$\Rightarrow$ "badPatchSize"

7 MIS of Specification Parameters Module

This module addresses the functional requirements as follows.

- R5

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

7.1 Module

specParams (Section 6)

7.2 Uses

None.

7.3 Syntax

7.3.1 Exported Constants

- $kernel\_sz := 5$
- $std\_deviation := 1$
- $FAST\_threshold := 15$
- $bin\_sz := 2000$
- $patch\_sz := 31$
- $mthd\_img\_smoothing := 1$
- $mthd\_kp\_detection := 1$
- $mthd\_kp\_description := 1$
- $mthd\_ft\_match := 1$

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_default_parameters	-	$kernel\_sz : \mathbb{Z}$ $std\_deviation : \mathbb{R}$ $FAST\_threshold : \mathbb{Z}$ $bin\_sz : \mathbb{Z}$ $patch\_sz : \mathbb{Z}$	-
get_default_methods	-	$mthd\_img\_smoothing : \mathbb{Z}$ $mthd\_kp\_detection : \mathbb{Z}$ $mthd\_kp\_description : \mathbb{Z}$ $mthd\_ft\_match : \mathbb{Z}$	-

7.4 Semantics

7.4.1 State Variables

$kernel\_sz : \mathbb{Z}$
$std\_deviation : \mathbb{R}$
$FAST\_threshold : \mathbb{R}$
$bin\_sz : \mathbb{Z}$
$patch\_sz : \mathbb{Z}$

$mthd\_img\_smoothing : \mathbb{Z}$
$mthd\_kp\_detection : \mathbb{Z}$
$mthd\_kp\_description : \mathbb{Z}$
$mthd\_ft\_match : \mathbb{Z}$

7.4.2 Environment Variables

none

7.4.3 Assumptions

none

7.4.4 Access Routine Semantics

get_default_parameters():

- output: out:=
  - kernel_sz: $\mathbb{Z}$
  - std_deviation: $\mathbb{R}$
  - FAST_threshold: $\mathbb{Z}$
  - bin_sz: $\mathbb{Z}$
  - patch_sz: $\mathbb{Z}$
- exception: none

get_default_methods():

- output:
  - mthd_img_smoothing: $\mathbb{Z}$
  - mthd_kp_detection: $\mathbb{Z}$
  - mthd_kp_description: $\mathbb{Z}$
  - mthd_ft_match: $\mathbb{Z}$
- exception: none

8 MIS of Output Format Module

- R14
- R15

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

8.1 Module

formatOutput

8.2 Uses

- OpenCVLib (Section 16)

8.3 Syntax

8.3.1 Exported Constants

Not applicable.

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
output_keypoints	img_id: str, parent_dir: str, keypoints: OpenCV Keypoint Class	-	-
output_features	img_id: str, parent_dir: str, descriptors: OpenCV DMatch Class	-	-
output_matches	query_id: str, train_id: str, parent_dir: str, matches: BFMatcher Class	-	-

8.4 Semantics

8.4.1 State Variables

- keypoint_fldr: str
- feature_fldr: str
- match_fldr: path

8.4.2 Environment Variables

- keypoint_path: str
- feature_path: str

- match_path: path

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

output_keypoints(img_id, parent_id, keypoints):

- transition: tran:= keypoint_path = parent_dir + keypoint_fldr + img_id + “kp” + “.csv”, where keypoint_path specifies the path to the output CSV file for the identified keypoints. This file will output the keypoint properties as follows per the OpenCV Keypoint Class.
  - horizontal pixel position
  - vertical pixel position
  - size
  - angle
  - response
- output: none
- exception: none

output_features(img_id, parent_dir, features):

- transition: tran:= feature_path = parent_dir + img_id + feature_fldr + “fd” + “.csv”, where descriptor_path specifies the path to the output CSV file for the identified feature descriptors. This file will output the descriptor properties as follows per the OpenCV Feature2D Class.
  - horizontal pixel position: $\mathbb{N}$
  - vertical pixel position: $\mathbb{N}$
  - size: $\mathbb{N}$
  - angle: $\mathbb{R}^+$
  - response: $\mathbb{N}$
  - descriptor: $\mathbb{N}_{256}^{32}$, where each bit is a 32-byte vector, and $\mathbb{N}_{256}$ represents unsigned 8-bit numbers [0, 255]
- output: none

- exception: none

output_matches(query_id, train_id, parent_dir, matches):

- transition: tran:= match_path = parent_dir + query_id + tran_id + match_fldr + "fm" + ".csv", where match_path specifies the path to the output CSV file for the identified matches. This file will output the properties for each keypoint as follows per the BFMatcher Class.
  - query index: ℕ
  - query horizontal position: ℕ
  - query vertical position: ℕ
  - train index: ℕ
  - train horizontal position: ℕ
  - train vertical position: ℕ
  - Distance: ℕ

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

- R13

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

9.1 Module

verifyOutput

9.2 Uses

None.

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
check_match_uniqueness	query_id: str, train_id: str, matches: BFMatcher Class	-	same_image, same_descriptor

9.4 Semantics

9.4.1 State Variables

none

9.4.2 Environment Variables

none

9.4.3 Assumptions

none

9.4.4 Access Routine Semantics

check_match_uniqueness (query_id, train_id, matches):

- output: none
- exception:
  - exc := "same_image" | (query_id == train_id), where the query and training images share the same name.
  - exc := "same_descriptor" | (matches.query_x == matches.train_x && matches.query_y == matches.train_y), where the coordinates of the matched features match between both query and training images.

9.4.5 Local Functions

none

10 MIS of Control Module

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

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

10.1 Module

main

10.2 Uses

- matchFeatures (Section 14)
- plotImage (Section 15)
- formatOutput (Section 8)
- verifyOutput (Section 9)

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

10.4 Semantics

10.4.1 State Variables

- kernel_sz: $\mathbb{N}$
- std_deviation: $\mathbb{R}$
- FAST_threshold: $\mathbb{N}$
- bin_sz: $\mathbb{N}$
- patch_sz: $\mathbb{N}$
- mthd_img_smoothing: $\mathbb{N}$
- mthd_kp_detection: $\mathbb{N}$
- mthd_kp_description: $\mathbb{N}$

- mthd_ft_match: $\mathbb{N}$
- img_obj_1, img_obj_2: $\mathbb{N}^{h \times w}$
- orb_obj1, orb_obj2 as OpenCV Keypoint Class
- brute_force_obj as BFMatcher Class

10.4.2 Environment Variables

- head_dir as str
- path_input_img as str
- path_keypoints as str
- path_descriptors as str
- path_feature_matches as str

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

main():

- transition: Modify the state of the Specification Parameters Module and the environment variables for the Image Plot Module and Output Format Module.

```
[head_dir as str] = get_head_directory()

[mthd_img_smoothing, mthd_kp_detection, mthd_kp_descriptors, mthd_ft_matching] = get_chosen_methods(

[kern_sz, std_deviation, FAST_threshold, bin_sz, patch_s] = get_chosen_parameters()

## For each image, i
# Smooth the image as a preprocessing step to keypoint detection
img_obj_1 = smooth_image(img_obj_1, kernel_sz, std_deviation)

# Identify the keypoints. Note that if the methods for keypoint detection and descriptors are both == 1, then ORB is the selected method, and the keypoint and descriptor modules should use the same ORB object, which likely will come from the OpenCV library
orb_object = initialize_orb(mthd_kp_detection
# Assign descriptors to keypoints
# export keypoints to csv
# export descriptors to csv

# generate and save image with keypoints
# generate and save image with scaled keypoints
##

# Compare features between differing images

# verify that the match structure conforms to the conditions in the Output Verification Module

# export matche tuples to csv
# generate and save images with corresponding matches
```

11 MIS of Image Smoothing Module

- R9

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

smoothImage

11.1 Uses

- config (Section 10)

11.2 Syntax

11.2.1 Exported Constants

None.

11.2.2 Exported Access Programs

Name	In	Out	Exceptions
smooth_image	mthd_img_smoothing: $\mathbb{N}$ raw_img: $\mathbb{N}^{h \times w}$ kernel_sz: $\mathbb{N}$ std_deviation: $\mathbb{N}$	img_blur: $\mathbb{N}^{h \times w}$	-
get_orb_object	-	orb_object as OpenCV ORB Class	-
detect_keypoints	orb_object as OpenCV ORB Class, img $\in \mathbb{N}^{h \times w}$	keypoints as OpenCV Keypoint Class	-

11.2.3 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]

11.2.4 Assumptions

- Exceptions on input limits are handled in specParams module.

11.2.5 Access Routine Semantics

smooth_image(mthd_img_smoothing, raw_img, kernel_sz, std_deviation | mthd_img_smoothing == 1):

- output: out:= img_blur = gaussianBlur(noisy_img, kernel_sz, std_deviation)
- exception: None

12 MIS of Keypoint Detection Module

- R6
- R10

12.1 Module

detectKeypoints

12.2 Uses

- config (Section 6)
- smoothImage (Section 11)
- OpenCVLib (Section 16)

12.3 Syntax

12.3.1 Exported Constants

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
initalize_orb	mthd_kp_detection: $\mathbb{N}$, FAST_Threshold: $\mathbb{N}$, bin_sz: $\mathbb{N}$, patch_sz: $\mathbb{N}$	orb_object: OpenCV ORB Class	-
get_orb_object	-	orb_object: OpenCV ORB Class	-
detect_keypoints	orb_object: OpenCV ORB Class, img: $\mathbb{N}^{h \times w}$	keypoints: OpenCV Keypoint Class	-

12.4 Semantics

12.4.1 State Variables

- orb_object as OpenCV ORB Class

12.4.2 Environment Variables

none

12.4.3 Assumptions

none

12.4.4 Access Routine Semantics

initialize_orb(mthd_kp_detection, FAST_Threshold, bin_sz, patch_sz | mthd_kp_detection == 1, mthd_kp_description == 1):

- transition: tran:= orb_object = ORB.create(FAST_Threshold, bin_sz, patch_sz)
- output: none

- exception: none

get_orb_object():

- output: out:= orb_object
- exception: none

detect_keypoints(orb_object, img):

keypoints = orb_object.detect(img)

- output: out:= keypoints
- exception: none

13 MIS of Feature Descriptor Module

- R7
- R11

13.1 Module

assignDescriptors

13.2 Uses

- config (Section 10)
- detectKeypoints (Section 12)
- OpenCVLib (Section 16)

13.3 Syntax

13.3.1 Exported Constants

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
compute_descriptors	img: $\mathbb{Z}^{h \times w}$, keypoints: OpenCV Keypoint Class	descriptors: OpenCV Feature2D Class	-

13.4 Semantics

13.4.1 State Variables

- orb_object: OpenCV ORB Class

13.4.2 Environment Variables

None.

13.4.3 Assumptions

- ORB object is instatiated in the Keypoint Detector Module.

13.4.4 Access Routine Semantics

compute_descriptors(orb_obj, img, keypoints):
orb_object = get_orb_object()

- output: desc := orb_object.compute(img, keypoints)
- exception: None

14 MIS of Feature Matching Module

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

- R8
- R12

14.1 Module

matchFeatures

14.2 Uses

- config (Section 10)
- detectKeypoints (Section 12)
- assignDescriptors (Section 13)
- OpenCVLib (Section 16)

14.3 Syntax

14.3.1 Exported Constants

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
create_BF_matcher	mthd_fm_match: ℕ, norm_method: ℕ, crosscheck_flag: 𝔹	bf_matcher_object: BFMatcher Class	-
get_bfm_object	-	bf_matcher_object: BFMatcher Class	-
match_features	bf_matcher_object: BFMatcher Class, desc1, desc2: OpenCV Feature2D Class	matches: OpenCV DMatch Class	-
sort_matches	bf_matcher_object: BFMatcher Class, matches: OpenCV DMatch Class,	sorted_matches: OpenCV DMatch Class	-

14.4 Semantics

14.4.1 State Variables

- bf_matcher_object: BFMatcher Class

14.4.2 Environment Variables

None.

14.4.3 Assumptions

Exception handling on user-selected methods and parameters are handled in the Parameter Specification Module.

14.4.4 Access Routine Semantics

create_BF_matcher(mthd_fm_match, norm_method, crosscheck_flag | mthd_fm_match == 1):

- output: out:= bf_matcher_object = BFMatcher(norm_method, crosscheck_flag)
- exception: None

matches = match_features(bf_matcher_object, desc1, desc2)

- **output:** out:= matches = bf_matcher_object.match(desc1, desc2)
- **exception:** None

sort_matches(bf_matcher_object, matches):

- **output:** out:= sorted _matches = bf_matcher_object.sorted(matches), where , such that the entries are organized in ascending order of the distance attribute
- **exception:** None

15 MIS of Image Plot Module

15.1 Module

plotImage

15.2 Uses

- OpenCVLib (Section 16)

15.3 Syntax

15.3.1 Exported Constants

none

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
set_parent_directory	dir: str	-	-
gen_kp_img	img_in: $\mathbb{N}^{h \times w}$, keypoints: OpenCV Keypoint Class, flags: $\mathbb{N}$	img_kp $\in \mathbb{N}^{h \times w}$	-
gen_matched_features	img_1, img_2: $\mathbb{N}^{h \times w}$, kp_1, kp_2: OpenCV Keypoint Class, matches: OpenCV DMatch Class max_matches: $\mathbb{N}$, flags: $\mathbb{N}$	img_matches: $\mathbb{N}^{h \times w}$	-
save_image	img_in: $\mathbb{N}^{h \times w}$, target_folder: str, img_name: str	png_out: png image	-

15.4 Semantics

15.4.1 State Variables

- DrawMatchesFlag: $\mathbb{N}$
- colours:= $\mathbb{N}^3$

15.4.2 Environment Variables

- parent_dir:= str
- img_output_path:= str

15.4.3 Assumptions

- gen_kp_img has been initialized with keypoints

15.4.4 Access Routine Semantics

set_parent_directory(dir)

- transition: tran:= parent_dir = dir

gen_kp_img(img_in, keypoints, flags):
img_keypoints = drawKeypoints(img_in, keypoints, colour, flags)

- output: img_keypoints $\in \mathbb{Z}^{h \times w}$

gen_matched_features(img_1, img_2,kp_1, kp_2, matches, max_matches):
img_matches = cv.drawMatches(img_1, kp_1, img_2, kp_2, matches[:max_matches], flags)

- output: img_matches $\in \mathbb{Z}^{h \times w}$, where the displayed matches range from 1:max_matches have the smallest distance attribute

save_image(img_in, target_folder, img_name):

- transition: img_output_path = join(parent_dir, target_folder, img_name)

- output: out:= png_out = imwrite(img_in, img_output_path)

16 MIS of OpenCV Module

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

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

16.1 Module

OpenCVLib

16.2 Uses

- config (Section 6)

16.3 Syntax

16.3.1 Exported Constants

None.

16.3.2 Exported Access Programs

General OpenCV Access Programs

Name	**Input**	**Output**	**Exceptions**
imread	sel_read_path: **str**	img $\in \mathbb{N}^{h \times w}$	inValidImgPath
imwrite	sel_save_path: **str**, out_img: $\mathbb{N}^{h \times w}$	img_png as .png	inValidImgPath
gaussianBlur	img: $\mathbb{N}^{h \times w}$, kernel_sz: $\mathbb{N}$, std_deviation: $\mathbb{R}$	smooth_img: $\mathbb{N}^{m \times n}$	-
drawKeypoints	img_in: $\mathbb{N}^{h \times w}$, keypoints: OpenCV Keypoint Class, colour: $\mathbb{N}^3$, flags: $\mathbb{N}$	img_kp: $\mathbb{N}^{h \times w}$	-
drawMatches	img1_in: $\mathbb{N}^{h_1 \times w_1}$, img2_in: $\mathbb{N}^{h_2 \times w_2}$ kp1, kp2: OpenCV Keypoint Class, flags: $\mathbb{N}$	img_matches: $\mathbb{N}^{(h_1+h_2) \times (w_1+w_2)}$	-
ORB.create	bin_sz: $\mathbb{N}$, patch_sz: $\mathbb{N}$, FAST_threshold: $\mathbb{N}$	orb_object: OpenCV ORB Class	-
BFMatcher	match_method: $\mathbb{N}$, cross_check_flag: $\mathbb{B}$	brute_force_object: OpenCV Brute Force Matcher Class	-

ORB Object Member Functions

Name	**In**	**Out**	**Exceptions**
detect	img: $\mathbb{Z}^{h \times w}$	keypoints: OpenCV Keypoint Class	invalidImg
compute	img $\in \mathbb{Z}^{h \times w}$, K where K: OpenCV Keypoint Class	descriptors: OpenCV Feature2D Class	invalidImg, invalidKeypoints

Brute Force Matcher Object Functions

Name	In	Out	Exceptions
match	D_1, D_2: OpenCV Feature2D Class	matches: OpenCV DMatch Class	Raises an error if descriptors are invalid or empty.
sorted	unsorted_matches: OpenCV DMatch Class	sorted_matches: OpenCV DMatch Class	-

16.4 Semantics

16.4.1 State Variables

- orb_object: OpenCV ORB Class
- bf_matcher_object: BFMatcher Class

16.4.2 Environment Variables

None.

16.4.3 Assumptions

- The input image is a valid grayscale or color image.
- Keypoints are detected before computing descriptors.
- ORB objects are initialized prior to use.
- BFMatcher objects are initialized prior to use.

16.4.4 Access Routine Semantics

imread(sel_read_path as str):

- output: out:= img $\in \mathbb{N}^{h \times w}$
- exception: if no image identified, flag as inValidImgPath

imwrite(sel_save_path as **str**, out_img $\in \mathbb{N}^{h \times w}$):

- output: out:= img_png as .png file
- exception: exc:= invalidImage

gaussianBlur(img, kernel_sz, std_deviation):

- output: out:= img_blur

- exception: none

drawKeypoints(img_in, keypoints,colour, flags):

- output: out:= img_kp
- exception: none

drawMatches(img1_in, img2_in, kp1, kp2, matches, flags):

- output: img_matches $\in \mathbb{N}^{(h_1+h_2)\times(w_1+w_2)}$

ORB.create(bin_sz, patch_sz, FAST_threshold):

- output: out:= orb_object as OpenCV ORB Class
- exception: None.

detect(img):

- output: out:= keypoints as OpenCV Keypoint Class
- exception: invalidImage

compute(img, K):

- output: out:= D as OpenCV Feature2D Class
- exception: exc:=
  - image not found ⇒ invalidImg
  - keypoints not found ⇒ invalidKeypoints

match(D_1, D_2):

- output: out:= matches M as OpenCV DMatch Class.
- exception: exc:= Raises an error if the descriptors are invalid or empty.
  - descriptors are invalid

sorted(unsorted_matches):

- output: out:= sorted_matches, where matches are sorted from unsorted_matches in ascending order of the distance attribute of the OpenCV DMatch Class
- exception: Raises an error if the match set is empty.

17 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 LaTeXfor hypperlinks to external documents. —SS]

17.1 Module

[Short name for the module —SS]

17.2 Uses

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]

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.

18 Appendix

[Extra information if required —SS]