Module Interface Specification for Image Feature Correspondences for Camera Calibration

Kiran Singh

March 22, 2025

1 Revision History

Date	Version	Notes
2025-03-21	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. Additional symbols and abbreviations are outlined in below.

symbol	type	description
b	$\mathbb{N}$	bin size
bf_matcher_object	-	Instance of a Brute Force Matcher object as BFMatcher Class
colour	$\mathbb{N}^3$	2D array of RGB values to assign red, green, and blue pixels
crosscheck flag	$\mathbb{B}$	flag used to assign how features are compared and matched
descriptors, fd1, fd2	$\mathbb{N}$	Feature Descriptors as OpenCV Feature2D Class
head_dir	str	A str that indicates the local directory of the IFC program within the file path system
image_IDs	str	List of strings that contains all IDs of saved images
img, img_in, img_out	$\mathbb{N}^{h \times w}$	Instance of an image object with height h and width w in pixels
img_obj_1, img_obj_2	$\mathbb{N}^{h \times w}$	Instances of image objects with height h and width w in pixels
img_idx_1, img_idx_1	$\mathbb{N}$	Indices of image names within the list **image_IDs**
img_kp	$\mathbb{N}^{h \times w}$	Instance of an image with displayed keypoints
img_fd	$\mathbb{N}^{h \times w}$	Instance of an image with size keypoints to account for feature descriptors
img_fm	$\mathbb{N}^{h \times w}$	Instance of two combined images with corresponding matches between keypoints
k	$\mathbb{N}$	kernel size
keypoints, kp1, kp2	-	Keypoints as OpenCV Keypoint Class
matches	-	instance of matched features as OpenCV DMatch Class
mthd_img_smoothing	$\mathbb{N}$	method employed to perform image smoothing, where values range from 1 to n methods.
mthd_kp_detection	$\mathbb{N}$	method employed to perform keypoint detection, where values range from 1 to n methods.
mthd_kp_description	$\mathbb{N}$	method employed to identify feature descriptors, where values range from 1 to n methods.
mthd_ft_match	$\mathbb{N}$	method employed to perform feature matching, where values range from 1 to n methods.
norm_method	$\mathbb{N}$	method used to determine the norm between two features per OpenCV NORM HAMMING
num_images	$\mathbb{N}$	number of images to be processed
orb_object	-	instance of an OpenCV ORB Class

p	$\mathbb{N}$	patch size
path_keypoints	str	Relative path to save keypoint data and imagery
path_descriptors	str	Relative path to save feature descriptor data and imagery
path_matches	str	Relative path to save feature match data and imagery
query_img_id	str	name of the query image
σ	$\mathbb{R}$	standard deviation of the Gaussian kernel
sel_read_path	str	user-defined read path to import imagery data
sel_save_path	str	user-defined read path to export imagery data
sorted_matches	-	instance of matched features as OpenCV DMatch Class
t	$\mathbb{N}$	FAST Intensity Threshold
train_img_id	str	name of the training image

Contents

3 Introduction

The following document details the Module Interface Specifications for the Image Feature Correspondence Software. The software identifies regions with similar feautures and pixel intensities amongst images, and returns a set of correspondences between these images to support downstream perception for applications in robotics.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/KiranSingh15/CAS-741-Image-Correspondences.

4 Notation

The structure of the MIS for modules comes from **?**, with the addition that template modules have been adapted from **?**. The mathematical notation comes from Chapter 3 of **?**. 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. Examples of common class definitions in OpenCV follow.

- brute force matcher objects as BFMatcher Class
- feature descriptor objects as OpenCV DMatch Class
- image keypoints objects as OpenCV ORB Class

- match objects as OpenCV DMatch Class
- ORB objects as OpenCV ORB Class
- OpenCV Norm objects as OpenCV NORM HAMMING

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	Control Module
	Input Format Module
	Specification Parameters Module
	Output Format Module
	Output Verification Module
	Image Smoothing Module
	Keypoint Detection Module
	Feature Descriptor Module
	Feature Matching Module
	Image Plot Module
Software Decision	OpenCV Library

Table 2: Module Hierarchy

6 MIS of Control Module

6.1 Module

main

6.2 Uses

- config (Section 7)
- formatOutput (Section 9)
- verifyOutput (Section 10)
- detectKeypoints (Section 12)
- assignDescriptors (Section 13)
- matchFeatures (Section 14)
- plotImage (Section 15)

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

6.4 Semantics

6.4.1 State Variables

- k: $\mathbb{N}$
- σ: $\mathbb{R}$
- : $\mathbb{N}$
- b: $\mathbb{N}$
- p: $\mathbb{N}$
- 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: $\mathbb{N}^{h \times w}$
- img_kp, img_fd, img_fm: $\mathbb{N}^{h \times w}$
- orb_object: OpenCV ORB Class
- brute_force_obj: BFMatcher Class
- kp1, kp2: OpenCV Keypoint Class
- fd1, fd2: OpenCV Feature2D Class
- matches: OpenCV DMatch Class
- image_IDs: str^n

6.4.2 Environment Variables

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

6.4.3 Assumptions

none

6.4.4 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()
set_parent_directory(head_dir)

[mthd_img_smoothing, mthd_kp_detection, mthd_kp_descriptors, mthd_ft_matching] = get_active_methods()

[k, σ, t, b, patch_s] = get_chosen_parameters()

check_limits(k, b, p, t, σ)

[image_IDs, num_images] = get_img_IDs(head_dir)

For images img_1 and img_2 in image_IDs, where $img_1 \neq img_2$ and $img_1 < img_2$
 # Smooth the image as a preprocessing step to keypoint detection
img_obj_1 = smooth_image(img_obj_1, k, σ)

 # 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, t, b, p)
brute_force_object = create_BF_matcher(mthd_ft_match, 6, TRUE) where 6 is the enumerated value of the Hamming Norm
kp1 = detect_keypoints(orb_object, img_1)
kp2 = detect_keypoints(orb_object, img_2)

 # export keypoints to csv
output_keypoints(img_IDs(*img_idx_1*), head_dir, kp1), where the image index corresponds to the selected keypoints
output_keypoints(img_IDs(*img_idx_2*), head_dir, kp2)

 # Assign descriptors to keypoints
fd1 = compute_descriptors(img_1, kp1)
fd2 = compute_descriptors(img_2, kp2)

 # export descriptors to csv
output_features(img_IDs(*img_idx_1*), head_dir, fd1), where the image index corresponds to the selected descriptors
output_features(img_IDs(*img_idx_2*), head_dir, fd1)

 # generate and save image with keypoints
img_kp = gen_kp_img(img_IDs(*img_idx_1*), kp1, 0)
save_image(img_kp, “”kpDetection”, img_IDs(*img_idx_1*))

 # generate and save image with scaled keypoints
img_kp = gen_kp_img(img_IDs(*img_idx_2*), kp2, 0)
save_image(img_kp, “”kpDetection”, img_IDs(*img_idx_2*))

 # generate and save image with scaled keypoints

img_fd = gen_kp_img(img_IDs(img_idx_1), kp1, 4), where flag = 4 representes a flag to draw rich keypoints
save_image(img_kp, "fDescriptors", img_IDs(img_idx_1))

img_fd = gen_kp_img(img_IDs(img_idx_2), kp2, 4)
save_image(img_kp, "fDescriptors", img_IDs(img_idx_2))

##

 # Compare features between differing images
matches = match_features(brute_force_obj, fd1, fd2)
matches = sort_matches(brute_force_obj, matches)

 # verify that the match structure conforms to the conditions in the Output Verification Module
check_match_uniqueness(img_IDs(img_idx_1), img_IDs(img_idx_2, matches)

 # export matches to csv
output_matches(img_IDs(img_idx_1), img_IDs(img_idx_2), head_dir, matches)

 # generate and save images with corresponding matches
img_fm = gen_matched_features(img_obj_1, kp1, img_obj_2, kp2, matches, 200, 2), where 200 of the best matches are shown, and the flag of 2 indicates that unmatched keypoints will not be displayed.
save_image(img_fm,"fMatches", img_IDs(img_idx_1 + img_IDs(img_idx_2))

7 MIS of Input Format Module

This module addresses the functional requirements as follows.

- R??
- R??
- R??
- R??

7.1 Module

config

7.2 Uses

- specParams (Section 8)

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_head_directory	-	head_dir 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	-	k: $\mathbb{N}$, b: $\mathbb{N}$, p: $\mathbb{N}$, t: $\mathbb{N}$, σ: $\mathbb{N}$	-
get_img_IDs	head_dir as str	img_IDs as str^n	-
check_limits	k: $\mathbb{N}$, b: $\mathbb{N}$, p: $\mathbb{N}$, t: $\mathbb{N}$, σ $\mathbb{N}$	-	invalid_parameters

7.4 Semantics

7.4.1 State Variables

- k: $\mathbb{Z}$
- σ: $\mathbb{R}$
- t: $\mathbb{Z}$
- b: $\mathbb{Z}$
- p: $\mathbb{Z}$
- 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

7.4.2 Environment Variables

- head_dir as str

7.4.3 Assumptions

none

7.4.4 Access Routine Semantics

get_head_directory():

- output: head_dir = Path(os.getcwd()) where head_dir 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: out:= [k, b, p, t, σ]
- exception exc:= none

get_img_IDs(head_dir as str):

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

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

check_limits():

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

$\neg(k < 1)$	$\Rightarrow$ "badKernelSize"
$\neg(k > 15)$	$\Rightarrow$ "badKernelSize"
$\neg(k \,\%\, 2 \neq 0)$	$\Rightarrow$ "badKernelSize"
$\neg(0 < \sigma < 10)$	$\Rightarrow$ "badStdDeviation"
$\neg(2 \leq t \leq 255)$	$\Rightarrow$ "badFASTThreshold"
$\neg(1 \leq b \leq 2048)$	$\Rightarrow$ "badBinSize"
$\neg(5 \leq p \leq 100)$	$\Rightarrow$ "badPatchSize"

8 MIS of Specification Parameters Module

This module addresses the functional requirements as follows.

- R??

8.1 Module

specParams (Section 7)

8.2 Uses

None.

8.3 Syntax

8.3.1 Exported Constants

- $k := 5$
- $\sigma := 1$
- $t := 15$
- $b := 2000$
- $p := 31$
- $mthd\_img\_smoothing := 1$
- $mthd\_kp\_detection := 1$
- $mthd\_kp\_description := 1$
- $mthd\_ft\_match := 1$

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_default_parameters	-	k: $\mathbb{Z}$ σ: $\mathbb{R}$ t: $\mathbb{Z}$ b: $\mathbb{Z}$ p: $\mathbb{Z}$	-
get_default_methods	-	mthd_img_smoothing: $\mathbb{Z}$ mthd_kp_detection: $\mathbb{Z}$ mthd_kp_description: $\mathbb{Z}$ mthd_ft_match: $\mathbb{Z}$	-

8.4 Semantics

8.4.1 State Variables

k: $\mathbb{Z}$
σ: $\mathbb{R}$
t: $\mathbb{R}$
b: $\mathbb{Z}$
p: $\mathbb{Z}$
mthd_img_smoothing: $\mathbb{Z}$
mthd_kp_detection: $\mathbb{Z}$
mthd_kp_description: $\mathbb{Z}$
mthd_ft_match: $\mathbb{Z}$

8.4.2 Environment Variables

none

8.4.3 Assumptions

none

8.4.4 Access Routine Semantics

get_default_parameters():

- output: out:= [k, σ t, b, p]
- 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

9 MIS of Output Format Module

- R??
- R??

9.1 Module

formatOutput

9.2 Uses

- OpenCVLib (Section 16)

9.3 Syntax

9.3.1 Exported Constants

Not applicable.

9.3.2 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_img_id: str, train_img_id: str, parent_dir: str, matches: BFMatcher Class	-	-

9.4 Semantics

9.4.1 State Variables

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

9.4.2 Environment Variables

- keypoint_path: str
- feature_path: str
- match_path: path

9.4.3 Assumptions

none

9.4.4 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_img_id, train_img_id, parent_dir, matches):

- transition: tran:= match_path = parent_dir + query_img_id + train_img_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: $\mathbb{N}$
  - query horizontal position: $\mathbb{N}$
  - query vertical position: $\mathbb{N}$

- train index: $\mathbb{N}$
- train horizontal position: $\mathbb{N}$
- train vertical position: $\mathbb{N}$
- Distance: $\mathbb{N}$

- output: none
- exception: none

9.4.5 Local Functions

none

10 MIS of Output Verification Module

- R??

10.1 Module

verifyOutput

10.2 Uses

None.

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
check_match_uniqueness	query_img_id: str, train_img_id: str, matches: BFMatcher Class	-	same_image, same_descriptor

10.4 Semantics

10.4.1 State Variables

none

10.4.2 Environment Variables

none

10.4.3 Assumptions

none

10.4.4 Access Routine Semantics

check_match_uniqueness (query_img_id, train_img_id, matches):

- output: none
- exception:
  - exc := "same_image" | (query_img_id == train_img_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.

10.4.5 Local Functions

none

11 MIS of Image Smoothing Module

- R??

11.1 Module

smoothImage

11.2 Uses

- config (Section 6)

11.3 Syntax

11.3.1 Exported Constants

None.

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
smooth_image	mthd_img_smoothing: $\mathbb{N}$ img_in: $\mathbb{N}^{h \times w}$ k: $\mathbb{N}$ σ: $\mathbb{N}$	img_out: $\mathbb{N}^{h \times w}$	-
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	-

11.3.3 Environment Variables

none

11.3.4 Assumptions

- Exceptions on input limits are handled in specParams module.

11.3.5 Access Routine Semantics

smooth_image(mthd_img_smoothing, img_in, k, σ | mthd_img_smoothing == 1):

- output: out:= img_out = gaussianBlur(img_in, k, σ)
- exception: None

12 MIS of Keypoint Detection Module

- R??
- R??

12.1 Module

detectKeypoints

12.2 Uses

- config (Section 7)
- 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: $\mathbb{N}$, t: $\mathbb{N}$, b: $\mathbb{N}$, p: $\mathbb{N}$	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, t, b, p | mthd_kp_detection == 1, mthd_kp_description == 1):

- transition: tran:= orb_object = ORB.create(t, b, p)
- 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

- R??
- R??

13.1 Module

assignDescriptors

13.2 Uses

- config (Section 6)
- 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

- R??
- R??

14.1 Module

matchFeatures

14.2 Uses

- config (Section 6)
- 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: $\mathbb{N}$, norm_method: $\mathbb{N}$, crosscheck_flag: $\mathbb{B}$	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: $\mathbb{N}^{h \times w}$, kp1, kp2: 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: $\mathbb{N}$
- colour:= $\mathbb{N}^3$

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
- output: none

- exception: none

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

- output: img_keypoints $\in \mathbb{N}^{h \times w}$
- exception: none

gen_matched_features(img_1, img_2,kp1, kp2, matches, max_matches):
img_matches = cv.drawMatches(img_1, kp1, img_2, kp2, matches[:max_matches], flags)

- output: img_matches $\in \mathbb{N}^{h \times w}$, where the displayed matches range from 1:max_matches have the smallest distance attribute
- exception: none

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)
- exception: none

16 MIS of OpenCV Module

16.1 Module

OpenCVLib

16.2 Uses

- config (Section 7)

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: $\mathbb{N}^{h \times w}$, k: $\mathbb{N}$, σ: $\mathbb{R}$	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	b: $\mathbb{N}$, p: $\mathbb{N}$, t: $\mathbb{N}$	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 $\in \mathbb{Z}^{h \times w}$, keypoints: OpenCV Keypoint Class	descriptors: OpenCV Feature2D Class	invalidImg, invalidKeypoints

Brute Force Matcher Object Functions

Name	In	Out	Exceptions
match	fd1, fd2: 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, k, σ):

- output: out:= img_out

- 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(b, p, t):

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

detect(img):

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

compute(img, keypoints):

- output: out:= descriptors as OpenCV Feature2D Class
- exception: exc:=
  - image not found $\Rightarrow$ invalidImg
  - keypoints not found $\Rightarrow$ invalidKeypoints

match(fd1, fd2):

- 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 Appendix

[Extra information if required —SS]