Module Interface Specification for IFDS

Gaofeng Zhou

Mar.24,2024

1 Revision History

Date	Version	Notes
Mar.24, 2024	1.0	First Version

2 Symbols, Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
DoG	Differential of Gaussian Transform of
G(x)	Gaussian Transform or Guassian Filtering of Image x
GoI	Gradients of Image
GUI	Graphical User Interface
I(x)	Image x
IFDS	Image Features Detection System
L(x)	Laplacian Transform
LoG	Laplacian Transform of Gaussian Transform of
Mat	Matrix
M	Module
MG	Module Guide
MIS	Module Interface Specification
OS	Operating System
R	Requirement
SC	Scientific Computing
SIFT	Scale Invariant Feature Transform
SRS	Software Requirements Specification
UC	Unlikely Change

Contents

3 Introduction

The following document details the Module Interface Specifications for Image Features Detection System(IFDS)

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at ,https://github.com/Zhou4truth/imageFeatureDetection/blob/main/docs/SRS/Software%20Requriement%20Specificati pdf, and https://github.com/Zhou4truth/imageFeatureDetection/blob/main/docs/Design/SoftArchitecture/MG_for_IFDS.pdf.

4 Notation

The structure of the MIS for modules comes from [**HoffmanAndStrooper1995**], with the addition that template modules have been adapted from [**GhezziEtAl2003**]. The mathematical notation comes from Chapter 3 of [**HoffmanAndStrooper1995**]. 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 .

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

The specification of 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, 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 Module	
Behaviour-Hiding Module	System Control Input Operation Image Processing Output Operation
Software Decision Module	GUI layout Display Setup Parameter Adjustment

Table 1: Module Hierarchy

6 MIS of Hardware-Hiding Module

6.1 Module

Hardware-Hiding

6.2 Uses

To run on different OS platforms.

6.3 Syntax

6.3.1 Exported Constants

Compiled program depending on different OS platforms.

6.3.2 Exported Access Programs

IFDS software installable format.

Name	**In**	**Out**	**Exceptions**
.exe	-	For windows	-
.pkg	-	For MacOS	-

6.4 Semantics

6.4.1 State Variables

No.

6.4.2 Environment Variables

No.

6.4.3 Assumptions

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

6.4.4 Access Routine Semantics

To be decided later.

7 MIS of Input Module

7.1 Module

Input

7.2 Uses

To process the input images and judge if these inputs are available to be processed within this system.

7.3 Syntax

Here we use the OpenCV library to do this input process function with "cv2.read()". Simple demo as bellow:

Listing 1: Image Input

```
import cv2
image =cv2.imread('path.....')
if image is None:
    print("Could not open or find the image")
else:
    cv2.imshow('Display window', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

And besides that, we also need to do the preprocessing of those input images. So the functions should be like:

Listing 2: Image Preprocessing

```
import cv2
import numpy as np

def preprocess_image(image_path):
    # Read the image from the specified path
    image = cv2.imread(image_path)

    # Check if image is loaded properly
    if image is None:
        print("Could not open or find the image.")
        return None
```

```
    # Convert the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Resize the image to a standard size (for example: 500x500)
    resized = cv2.resize(gray, (500, 500))

    # Apply Gaussian blur to remove noise
    blurred = cv2.GaussianBlur(resized, (5, 5), 0)

    # Threshold the image to obtain a binary image
    _, binary = cv2.threshold(blurred, 128, 255, cv2.THRESH_BINARY | cv2.THRE

    # Perform edge detection using Canny
    edges = cv2.Canny(binary, 100, 200)

    # Return the preprocessed image
    return edges

# Usage
preprocessed_image = preprocess_image('path_to_image.jpg')
```

7.3.1 Exported Constants

Preprocessed Image Files,Data Arrays,Parameters and Metrics,Configuration Files.

7.3.2 Exported Access Programs

Preprocessed images that will be continued to the feature detection algorithm module.

Name	In	Out	Exceptions
rgb2grey	RGB image	Gray image	-
resized	RGB image	Resized image	-
blurred	RGB image	Blurred image	-

7.4 Semantics

7.4.1 State Variables

image

7.4.2 Environment Variables

PYTHONPATH,PATH,
OPENCV-DIR,

LD-LIBRARAY-PATH.

7.4.3 Assumptions

Here, we use the methods in OpenCV library to do this step.

7.4.4 Access Routine Semantics

7.4.5 Local Functions

8 MIS of Image Processing Module

8.1 Module

Image Processing

8.2 Uses

To realize all those feature detection functions.

8.3 Syntax

Listing 3: Edge detection

```
import cv2

# Load an image
image = cv2.imread('path_to_image.jpg', cv2.IMREAD_GRAYSCALE)

# Use Canny edge detector
edges = cv2.Canny(image, threshold1=100, threshold2=200)

# Display the edges
cv2.imshow('Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Listing 4: Contour Detection

```
import cv2

# Load an image
image = cv2.imread('path_to_image.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
# Find edges using Canny
edges = cv2.Canny(gray, 100, 200)

# Find contours
contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Draw contours on the original image
cv2.drawContours(image, contours, -1, (0, 255, 0), 3)

# Display the image with contours
cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Listing 5: Corner Detection

```
import cv2
import numpy as np

# Load image
image = cv2.imread('path_to_image.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Find Harris corners
gray = np.float32(gray)
dst = cv2.cornerHarris(gray, blockSize=2, ksize=3, k=0.04)

# Dilate to mark the corners, not important to detection
dst = cv2.dilate(dst, None)

# Threshold for an optimal value, marking the corners in red
image[dst > 0.01 * dst.max()] = [0, 0, 255]

# Display the corners
cv2.imshow('Harris Corners', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Listing 6: SIFT Detection

```
import cv2

# Load an image
```

```
image = cv2.imread('path_to_image.jpg')

# Create a SIFT object
sift = cv2.SIFT_create()

# Detect SIFT features (keypoints and descriptors)
keypoints, descriptors = sift.detectAndCompute(image, None)

# Draw keypoints on the image
image_with_keypoints = cv2.drawKeypoints(image, keypoints, None)

# Display the image with keypoints
cv2.imshow('SIFT KeyPoints', image_with_keypoints)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.3.1 Exported Constants

Detected feature points of those input images.

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
cv2.Canny	image	edges	-
cv2.findContours	image	contours	-
cv2.cornerHarris	image	harris points	-
cv2.SIFT-create()	image	SIFT configure	-
sift.detectAndCompute	image	SIFT descriptors	-

8.4 Semantics

8.4.1 State Variables

Gradient magnitude and angle matrices calculated during the edge detection process.

The output binary image.

The hierarchy of contours and the binary image after applying thresholding or edge detection, which is used to find contours.

The response matrix generated by the Harris corner detection algorithm, indicating corner strength at each pixel.

Image pyramids and gradient images used internally by the SIFT algorithm to detect key-points across scale space.

8.4.2 Environment Variables

PYTHONPATH,
PATH,
OPENCV-FFMPEG-CAPTURE-OPTIONS,
CUDA-VISIBLE-DEVICES,
LD-LIBRARAY-PATH.

8.4.3 Assumptions

Here, we would use libraries in OpenCV.

8.4.4 Access Routine Semantics

cv2.canny():

cv2.findcontours():

cv2.cornerHarrris():

cv2.SIFT-creat().detectAndCompute():

9 MIS of Output Module

9.1 Module

Output

9.2 Uses

To show the results of those detected image feature points.

9.3 Syntax

Listing 7: to show the image

```
cv2.imshow('Processed Image', image)
cv2.waitKey(0)  # Wait for a key press to close the window
cv2.destroyAllWindows()  # Close all OpenCV windows
```

Listing 8: the save those output

```
cv2.imwrite('output.jpg', image)
```

Listing 9: the display the outputs

```
import numpy as np

# Assuming 'original' and 'processed' are images of the same height
combined = np.hstack((original, processed))
cv2.imshow('Original vs Processed', combined)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

9.3.1 Exported Constants

Display Windows Names

Image File Formats

Feature Data File Formats

Window Wait Time

Path and Filename Conventions

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
display-image	image	cv2.imshow	-
save-image	image	cv2.imwrite	-
display-multiple-images	images	cv2.imshow	-
save-feature-data	data	file	-
save-numerical-data	data	file	

9.4 Semantics

9.4.1 State Variables

Window Names

Recent Save Path

Supported Formats

Export Paths

Serialization Format

9.4.2 Environment Variables

OUTPUT-PATH

DISPLAY

PYTHONPATH

OMP-NUM-THREADS

9.4.3 Assumptions

Only selected features can be displayed.

9.4.4 Access Routine Semantics

display-image()

save-image()

display-multiple-images()

save-feature-data()

9.4.5 Local Functions

```
def _is_valid_extension(file_path):
    valid_extensions = ['.jpg', '.jpeg', '.png', '.bmp', '.tiff']
    _, ext = os.path.splitext(file_path)
    return ext.lower() in valid_extensions

def _prepare_window(window_name):
    # Ensure the window name is unique or reset properties
    # This is a placeholder for any necessary window preparation
    pass



def _concatenate_images(images, axis='horizontal'):
    if axis == 'horizontal':
        return np.hstack(images)
    else:  # 'vertical'
        return np.vstack(images)



def _serialize_keypoints(keypoints):
    # Convert keypoints to a serializable format
    serialized_kps = [{'pt': kp.pt, 'size': kp.size, 'angle': kp.angle,
                       'response': kp.response, 'octave': kp.octave,
                       'class_id': kp.class_id} for kp in keypoints]
    return serialized_kps


def _is_supported_format(format):
    supported_formats = ['pkl', 'json', 'xml']
    return format in supported_formats
```

References

SRS,https://github.com/Zhou4truth/imageFeatureDetection/blob/main/docs/SRS/Software
MG,https://github.com/Zhou4truth/imageFeatureDetection/blob/main/docs/SRS/Software

10 Appendix

11 Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)
   The biggest limitation is only to do the features detection before further image reconstruction. If given unlimited resources, I would like to build a system to do the 3D image reconstruction.

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)

   I think it's a good idea to do medical image processing systems using knowledge obtained from this course. It can be used for future further development rather than only a practice of those procedures.