2017 Fall COMS 4115 PLT Language Reference Manual

SIPL: Simple Image Processing Language

Simon Zhai, Ci Chen, Shanshan Zhang, Yihan Zhao, Yuedong Wang

{yz3116, cc4192, sz2648, yz2996, yw2931}@columbia.edu

Contents

1 Introduction

The SIPL language is a linear algebra manipulation language specially targeted for image processing. It uses an efficient way to express complex image manipulation algorithms to complete matrix operations.

2 Lexical Conventions

2.1 Tokens

There are five kinds of tokens in SIPL: Keywords, identifiers, constants, control characters and operators. Blanks, tabs, newlines and comments are ignored except that they serve to separate tokens.

2.2 Comments

The characters /* introduces a comments which terminates with */.

2.3 Keywords

The reserved keywords in SIPL are:

Boolean
False
Float
Kernel
Int
Matrix
True
Uint8

2.4 Identifiers

Identifiers are composed of a lower or upper-case letter immediately followed by any number of additional letters and/or digits. Identifiers are case sensitive, “Simon” and “simon” are different identifiers. Keywords and underscores are not allowed to be identifiers.

2.5 Constants

2.5.1 Numeric constants

Integers are represented by a series of number characters. Floats are represented by a series of number character with an optional period character, followed by a lower case “f”.

3 Meaning of Identifiers

3.1 Data Types

3.1.1 Basic Types

Int: 32 bit integer
Uint8: 8 bit unsigned integer
Float: 32 bit real number
Boolean: A constants with either **True** or **False**

3.1.2 Built-in Types

Matrix: n × m size of Unit8 matrix. We use one matrix to represent gray images and an array of three matrices to represent colored images.

4 Object and Definition

An object in SIPL is either an array of matrices or one single matrix. Each colored image consists of three matrices, each matrix represents a channel(Red Green Blue) while each gray image consists of one matrix.

5 Expressions

5.1 Primary Expression

5.1.1 Identifier

An Identifier is a primary expression

5.1.2 constant

An integer or a floating number is a primary expression

5.1.3 (expr)

An expression that is parenthesized is a primary expression, with the same type of original expression.

5.2 Object Creation Expression

5.2.1 [expr]

An expression that is bracketed is an object creation expression, it will create a matrix. We use ”,” to separate different elements in one row, and ”;” to separate different rows.

6 Operators

6.1 Assignment Operators: ”=”

Assign the value of right to the left, even if it is a matrix, the data would be copied by value. So the space should be pre-allocated.

6.2 Arithmetic Operatos: ”+”, ”-”, ”*”, ”/”

Use between Int, Float or Unit8. It will group from left to right, ”*” and ”/” have higher priority than ”+” and ”-”.

6.3 Matrix Operators: ”.+”, ”.-”, ”.*”

The two matrix should have the same shape, the output matrix would have the same shape with the imput matrix.
”.+”: Add the two elements in the same position. ”.-”: Minus the two elements in the same position. ”.*”: Multiply the two elemtns in the same position.

6.4 Relational Operators: ”>”, ”<”, ”>=”, ”<=”, ==

The value of right expression and left expression should be Int of Float or Uint8.

7 Statements

7.1 Control Flow Statements

7.1.1 Loop

”While” will execute a block that defined by user, if a specific expression is **True**.
while(expr){
Statement
}

8 Build-in Functions

8.1 Gray Conversion

img.gray := grayImg(Img)
Change colored image into gray image.

8.2 Image Rotation

rotatedImg = rotateImg(img, n)
Counter-clockwise rotate an image by $\pm\frac{\pi}{2}(n = 1$ for $\frac{\pi}{2}$, $n = -1$ for $-\frac{\pi}{2}$, otherwise cast an error)

8.3 Image Flipping

flippedImg = flipImg(img, n)
Flip an image($n = 1$ for horizontal flip and $n = -1$ for vertical flip, otherwise cast an error).

8.4 Matrix Construction

Use **Kernel** to construct a matrix

8.5 Image Filtering

filteredImage = convImg(img, kernel) Convolve a specific kernel with an image, different kernels will have different convolutions.

8.6 RGB Channelling

Get one of the channel from a colored image(red, blue or green), return a matrix.

8.7 Image Resizing

resizedImage = resizeImg(img, width, height)
Resize the original image, get an image with different width and height. The shape can be changed.

9 Sample Code

```
Img = readImg(‘img1.png’)
/* import an image for operation*/
Img = rotateImg(Img, 1)
/* rotate this image*/
Img = img + 2;
/* enhance the brightness of the rotated image.*/
Matrix kernel = Kernel(1,2)
/* create a gaussian kernel*/
Img = convImg(img, kernel)
/* we convolve the image using gaussian kernel, which will blur the image.*/
showImg(Img)
/* check the modified img*/
writeImg(Img,’ /MyDirectory/modified image.png’)
/* save the modified image into a directory named ’modified image.png’ */
```

10 References

1. B. W. Kernighan and D. Ritchie. The C Programming Language
2. Dennis M. Ritchie. C Reference Manual