Facelab: A Facial Image Editing Language

Xin Chen (xc2409)
Kejia Chen (kc3136)
Tongfei Guo(tg2616)
Weiman Sun (ws2517)

September 26, 2017

1 Introduction and Motivation

Facelab aims to perform face detection, face recognition, filter applying and photo sticker adding among other features which enable the target users to manipulate their portrait photos with ease and accuracy.

The basic syntax of this language largely resembles that of C++, excluding some of the irrelevant and hard-to-implement details such as inheretance, template, etc. With the inclusion of the matrix data type that is common to many scientific programming languages, it not only facilitates image processing related computation, but also grants users the ability to manipulate photo on a pixel scale and allows users the freedom to define and tailor their own filter to individuals' preference. Moreover, by having OpenCV linked at the compiling stage, it provides access to some of the state-of-art face detection and face recognition algorithms without the hassle of having to install the whole libraries of OpenCV, learning its and its companying auxiliary libraries'(such as Eigen library) functionalities. A combination of these afore-mentioned features could considerably simplify real-life tasks such as adjusting photo brightness and contrast, batch-editing photos, auto-applying facial pixelization, and so on.

2 Syntax

Table 1: Data Structure

type name	description
int	32-bit signed integer
double	64-bit float-point number
bool	8-bit boolean variable
string	array of ASCII characters
matrix	data structure storing bool/ints/doubles of arbitrary size
image	a jpg image(data structure storing 3 matrices)

Table 2: Arithmetic Operator

Operator name	functionality
+	addition(scalar), addition(matrix)
-	subtraction(scalar), subtraction(matrix)
*	multiplication(scalar), multiplication(matrix)
.*	component-wise multiplication(matrix)
'	transpose(matrix)
/	division(scalar)
%	yields the remainder from the division of the first expression by the second(scalar)
!	not(scalar)
|	or(scalar)
&	and(scalar)
$	pre-defined filter identifier, followed by filter's name.(e.g. image $sharpen)

Table 3: Relational Operator

Operator name	functionality
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

Table 4: Flow Control

Operator name	functionality	Syntax
if/ else/ else if	Conditional expression	if (condition)
while/ for	Iteration	while(condition),for(count; condition; update)
continue/break	Branching control	break;continue;

Table 5: Built-in functions

Function name	functionality	Signiture
load	load image	load("filename.jpg")
save	save image	save("filename.jpg")
recognition	recognize the face in the filename.jpg by training images from the given folder	recognition("folder name","filename.jpg")
detection	detect whether filename.jpg includes faces	detection("filename.jpg")
drawRect	draw a rectangle of given size at given position	drawRect(image, x, y, width, height)
drawText	draw given string at given position	drawText(image, string, x, y)
pixelize	apply pixelization at given position	pixelize(image, x, y, width, height)

Indentation indentation does not affect syntax.

Trailing semicolon semicolons at the end of each statement perform no operation but signal the end of a statement.

Comment "// comments" for line comment, and "/* comments */" for block comments.

Function functions can return multiple values as shown below: func function_name(Parameter1, Parameter2,...) { //function body here. return variable1, variable2, ... }

3 Sample Program

3.1 GCD Algorithm

```
func gcd(int m, int n) {
//calculate gcd of two integer number
    while(m>0)
    {
        int c = n % m;
        n = m;
        m = c;
    }
    return n;
}
```

3.2 Apply a Filter

```
image pic1 = load(../"xxx.jpg"); // built in function to load an image
matrix sharpen =
[0,-1,0;
-1,5,-1;
0,-1,0];
//define a filter.
image pic2 = pic1 $sharpen; // apply filter to image.
save("../yyy.jpg", pic2); // built in func to save an image to file
```

3.3 Face Recognition

```
string label; // define label of the given person.
label, x, y, w, h = recognition("../folder", "../xxx.jpg");
// train images from a given folder, recognize images from target path,
// return the label of the recognized person and a rectangle around the
// face, (x, y, w, h) stand for coordinates, width and height of the rectangle.
image pic1 = load("../xxx.jpg");
pic1 = drawRect (pic1, x, y, w, h);
// draw a rectangle around the face
pic1 = drawText (pic1, identity, x + 10, y + 10);
// draw the label around the face
save("../yyy.jpg", pic1);// built in func to save an image to file
```

References

3.4 References

Face Recognition with OpenCV
http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html
CAL: Concise Animation Language
http://www.cs.columbia.edu/ sedwards/classes/2013/w4115-fall/index.html