SIPL: Simple Image Processing Language

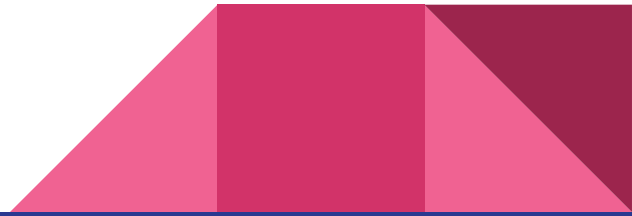
Shanshan Zhang, Yihan Zhao, Yuedong Wang, Ci Chen, Simon Zhai
COMS W4115: Programming Languages and Translators
Columbia University

December 20, 2017

Introduction

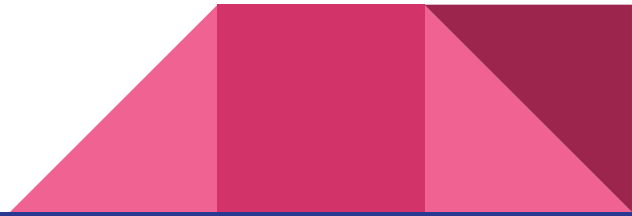
Simple Image Processing Language

- Targeted for image processing
- Deal with images in an effective and fast way.
- Simplify the implementation of image processing algorithm
  - define primitive data types - Matrix, Image, Pixel



Introduction

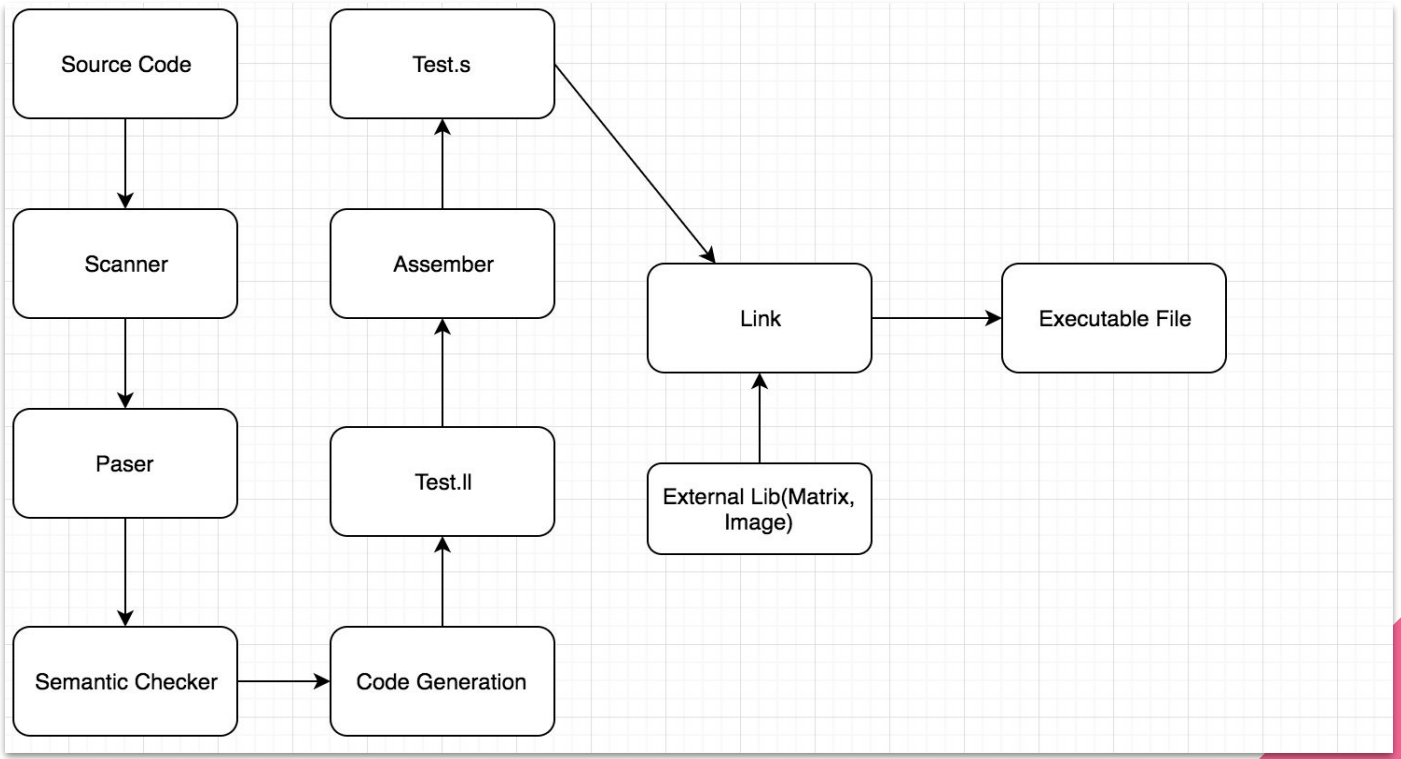
- Supports basic calculation for images
- Supports more advanced image manipulations
- For example,
  - Resize the image
  - Flip and rotate the images
  - Change image into grey
  - Detect the edge of an image
  - Support pixel operation
  - Easy convolution



Timeline

Date	Milestone
September 26	Complete Language proposal
October 16	Complete Language Reference Manual
October 31	Compiler front end (lexer and parser) complete
November 16	Semantics / type checking complete
November 25	Code generation complete
December 6	**Hello World runs**
December 17	**Finalize the test suite and bug fix**
December 19	Demo
December 20	Final report complete

Architecture



Features

- **Data Types** (Image, Matrix, Pixel)
- **Operators** (e.g convolution, image addition and multiplication)
- **Built-in functions** (imgread, imgwrite, imgCopy, ...)
- **Specific features** (e.g Auto Clamping, type conversion, bounding)

Data Types

- **Image**

  var = img[i][j]["g"] access green channel
  img[i][j]["r"] = val assign to red channel
  Image.width get width of image
  Image.height get height of image

- **Matrix**

  mat = [0.1, 0.2; 0.3, 0.4; 0.5, 0.6];
  var = mat[i][j] access
  mat[i][j] = var assign
  mat.width get width of matrix
  mat.height get height of matrix

- **Pixel**

  pixel = img[i][j]["a"] access Pixel (3 channels)
  Img[i][j]["a"] = pix; assign Pixel to img

type conversion

int -> float int2float() auto cast
float -> int mannual cast

- float
- string
- int

Operators

- Image operators:
  - img + img
  - img + int
  - img - int
  - img * float
- Pixel operators:
  - pixel + pixel
  - pixel + int
  - pixel - int
  - pixel * float
- Basic operators:
  - + - * /
- Convolution:
  - img ** mat
  - mat ** img

Built-in functions

- **img = imgCreate(height, width);**
- **img = imgread(“myPicture.png”);**
- **imgwrite(img, “picture.png”);**
- **img = imgCopy(img);**
- img = changeGrey(img);
- img = rotateImage(img);
- img = flipImage(img);

- **mat = matCreate(height, width);**
- mat = kernel(“edge”);
- int2float()
- float2int()
- power(x,y);

SIPL Lib

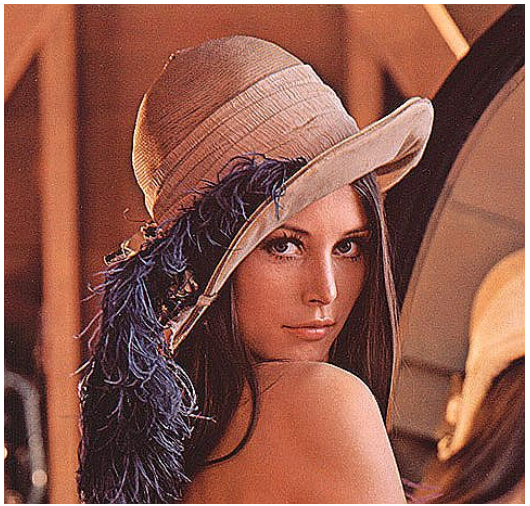
- img = resize(img, 2.0, 2.0); // twice original size.
- img = sliceImg(img, left, up, width, height);
- mat = sliceMat(mat, left, up, width, height);
- mat = gaussian(size, sigma);

Examples

Copy

```
Image img; Image imgC;
img = imgread(“lena.png”);
imgC = imgCopy(img);
imgwrite(imgC, 0, “lena2.png”)
```

Sharpen



```
Matrix mat;
mat = kernel(“sharpen”);
img = img ** mat;
```

Edge

```
Matrix mat;
mat = kernel(“edge”);
img = img ** mat;
```

Specific features

```
newVal = img[i][j]["r"] + val;
if (newVal < 0) {
    img2[i][j][a"r"] = 0;
} else if (newVal > 255) {
    img2[i][j]["r"] = 255;
} else {
    img2[i][j]["r"] = newVal;
}
```

Auto-clamping

```
img2[i][j]["r"] = img[i][j]["r"] + val;
```

```
img2[i][j]["r"] = img[i][j]["r"] + val;
img2[i][j]["g"] = img[i][j]["g"] + val;
img2[i][j]["b"] = img[i][j]["b"] + val;
```

Pixel-manipulate

```
img2[i][j]["a"] = img[i][j]["a"] + val;
```

```
if( (indexx + 1 < width) &&
(indexy + 1 < height))
```

Auto-inbound

Example-resize

Matlab

```
function resize_image(img, scaleX, scaleY)
%Usage: directly call resize_img(img, scaleX, scaleY),
%e.g. resize('lena.png', 1.5, 1.5)
img = im2double(img);
[height,width,~] = size(img);
rows = height * scaleY;
cols = width * scaleX;
output_img = zeros(ceil(rows), ceil(cols), 3);
for i = 1:ceil(rows)
    for j = 1:ceil(cols)
        for k = 1:3
        bp_row = i/scaleY;
        bp_col = j/scaleX;
        indexx = ceil(bp_col);
        indexy = ceil(bp_row);

        delx = bp_col - indexx;
        dely = bp_row - indexy;

        if( (indexx + 1 < width) && (indexy + 1 < height))
            tmp = img(indexy,indexx,k) * (1.0 - delx) * (1.0 - dely);
            tmp = tmp + img(indexy + 1,indexx,k) * delx * ( 1.0 - dely);
            tmp = tmp + img(indexy + 1,indexx + 1,k) * dely * delx;
            tmp = tmp + img(indexy,indexx + 1,k) * (1.0 - delx) * dely;
            if(tmp > 1)
                tmp = 1;
            end
            output_img(i,j,k) = tmp;
        else
            output_img(i,j,k) = 0;
        end
        end
    end
end
imwrite(output_img, 'test_output.png');
```

without auto bounding

without clamping

SIPL

```
Image resize (Image img, float scaleX, float scaleY){
        float cols;float rows;float bp_row;float bp_col;int i;int j;
        float delx;float dely;
        Pixel tmp;int tmp2;int indexy;int indexx;Image res;
        cols = img.width * scaleX;
        rows =  img.height * scaleY;
        res = imgCreate(float2int(rows), float2int(cols));

        for(i = 0; i < rows; i = i + 1){
            for(j = 0; j < cols; j = j + 1){
                bp_row =  i / scaleY;
                bp_col =  j / scaleX;
                indexx = float2int(bp_col);
                indexy = float2int(bp_row);
                delx = bp_col -  indexx;
                dely = bp_row -  indexy;
                tmp = img[indexy][indexx]["a"] * (1.0 - delx) * (1.0 - dely);
                tmp = tmp +  img[indexy + 1][indexx]["a"] * delx * ( 1.0 - dely) ;
                tmp = tmp +  img[indexy + 1][indexx + 1]["a"] * dely * delx;
                tmp = tmp +  img[indexy][indexx + 1]["a"] * (1.0 - delx) * dely;
                res[i][j]["a"] = tmp;     }       }
        return res;
}
```

get image's attribute

auto type conversion

pixel manipulation

Test Plan

- **Syntax Verification :**

  the parser accepts all valid strings and rejects all invalid ones defined in LRM

- **Semantic Verification :**

  the verifier accepts all valid parse trees and rejects all invalid ones

- **Image Processing Verification :**

  generate both simple and complicated programs that test all the functions of the language such as read and write image, image manipulations.

Test Samples

- test-matrix-access.out
- test-matrix-access.sp
- test-matrix-assign.out
- test-matrix-assign.sp
- test-matrix-assign0.out
- test-matrix-assign0.sp
- test-matrix-gen.sp
- test-matrix-matCreate.out
- test-matrix-matCreate.sp
- test-matrix-slice.sp
- test-ops1.out
- test-ops1.sp
- test-ops2.out
- test-ops2.sp

- test-img-access1.out
- test-img-access1.sp
- test-img-assign1.sp
- test-img-changeGrey.sp
- test-img-elementAdd.sp
- test-img-elementMult.sp
- test-img-elementMult2.sp
- test-img-elementSub.sp
- test-img-flip.sp
- test-img-imgAdd.sp
- test-img-imgCopy.sp
- test-img-imgCreate.sp
- test-img-rotateImage.sp
- test-img-slice.sp
- test-imgread.sp
- test-int2float.out
- test-int2float.sp

- fail-img-access2.err
- fail-img-access2.sp
- fail-img-access3.err
- fail-img-access3.sp
- fail-img-assign1.err
- fail-img-assign1.sp
- fail-img-assign2.err
- fail-img-assign2.sp
- fail-img-assign3.err
- fail-img-assign3.sp
- fail-img-assign4.err
- fail-img-assign4.sp
- fail-matrix-access1.err
- fail-matrix-access1.sp
- fail-matrix-access2.err
- fail-matrix-access2.sp

Test Samples

input image
lena.png

Image Add

```
Image img;
img = imgread("lena.png");
img = img + img;
imgwrite(img, 0,
"lena-imgAdd.png");
```

Element multiplication

```
Image img;
img = imgread("lena.png");
img = img * 0.5;
imgwrite(img, 0,
"lena-elementMult.png");
```

Test Samples

Grey

Image img;
img = changeGrey(img);

Flip

img = flipImage(img);

Rotate

img = rotateImage(img);

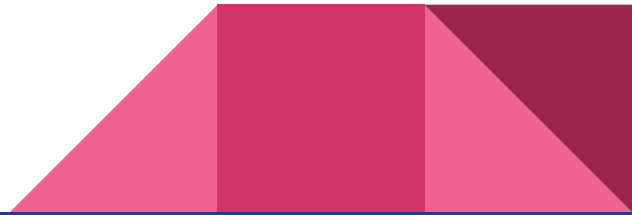
Test Automation

- 80 unit test
- testall.sh script to do test automation
  - Compile, run, and check the output of each expected-to-work test
  - Compile and check the error of each expected-to-fail test

```
yihans-MacBook-Pro-2:SIPL yihan$ ./testall.sh
-n test-add1...
OK
-n test-arith1...
OK
-n test-arith2...
OK
-n test-arith3...
OK
-n test-fib...
OK
-n test-float1...
OK
-n test-float2...
OK
-n test-float2int...
OK
-n test-float3...
OK
```

Lessons Learned

- Yuedong: I learned how to write a simple compiler
- Ci: How to work in a team.
- Simon: I learned how to work in team and trust my teammates.
- Yihan: I learned valuable skills in organizing a group project: conciseness can make both project management process and code elegant.
- Shanshan: Keep asking questions if you don’t understand. Having a good team communication is very important, and Ci is very cute



Demo Time!