

OS HW3

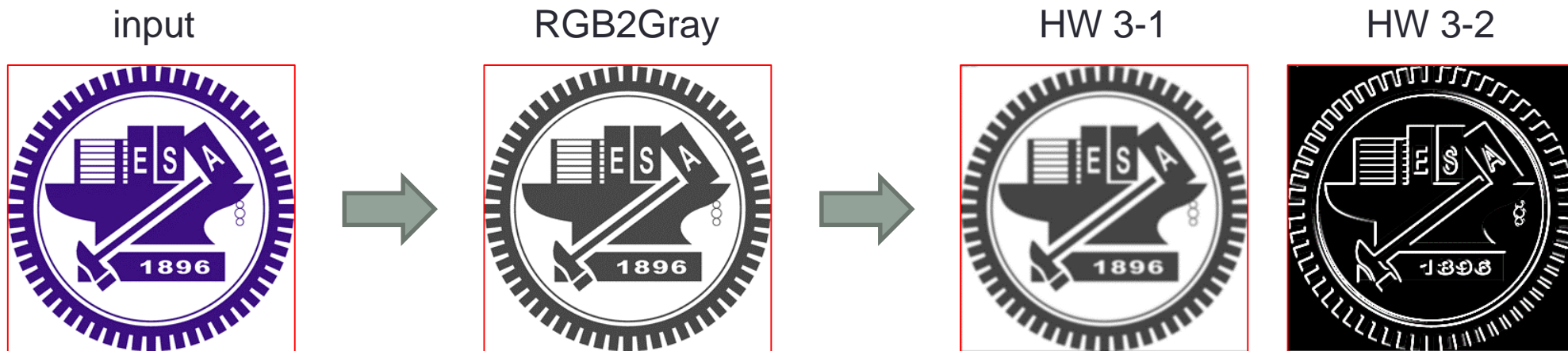
Operating System 106 Fall

W.J.Tsai 蔡文錦 教授

TA 鍾陳恩
倪莞雅
巫怡慧
任佳靜

Goal

- Implement image processing by using **threads** and **synchronization**.
 - HW 3-1: Blur with Gaussian filter
 - HW 3-2: Edge Detection with Sobel filter



Introduction: Gaussian filter

- Gaussian filter: $G(u, v) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/(2\sigma^2)}$
- For convenience, use integer versions.
 - For example:

$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$\frac{1}{273} \times \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array}$$

- Gaussian filter is written in “mask_Gaussian.txt”
 - The first number is the filter size
 - The second number is the filter scalar

→

9	16
1	2
2	4
1	2

Introduction: Blur algorithm

1. Convert RGB image to grey image:

- $\text{grey}(i, j) = (R(i, j) + G(i, j) + B(i, j)) / 3$

2. Do Gaussian blur: convolving the grey image with a Gaussian filter.

- Convolving:

Slide window

16	32	160	80	16
0	48	112	32	176
0	0	48	16	16
0	8	4	0	96
8	0	4	8	160

 $\times \frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

 $=$

11	43	73	64	38
10	44			

- $\rightarrow (16 \times 1 + 32 \times 2 + 160 \times 1 + 0 \times 2 + 48 \times 4 + 112 \times 2 + 0 \times 1 + 0 \times 2 + 48 \times 1) / 16 = 44$

3. Extend the size of image from HxWx1 to HxWx3 (to save the image)

- $R(i, j) = \text{grey}(i, j)$
- $G(i, j) = \text{grey}(i, j)$
- $B(i, j) = \text{grey}(i, j)$

Introduction: Blur algorithm (cont.)

1. Convert RGB image to grey image:

- $\text{grey}(i, j) = (R(i, j) + G(i, j) + B(i, j)) / 3$

2. Do Gaussian blur: convolving the grey image with a Gaussian filter.

- Convolving:

Slide window

16	32	160	80	16
0	48	112	32	176
0	0	48	16	16
0	8	4	0	96
8	0	4	8	160

*

$\frac{1}{16}$

x

1	2	1
2	4	2
1	2	1

=

11	43	73	64	38
10	44	72		

3. Extend the size of image from HxWx1 to HxWx3 (to save the image)

- $R(i, j) = \text{grey}(i, j)$
- $G(i, j) = \text{grey}(i, j)$
- $B(i, j) = \text{grey}(i, j)$

Introduction: Blur algorithm (cont.)

1. Convert RGB image to grey image:

- $\text{grey}(i, j) = (R(i, j) + G(i, j) + B(i, j)) / 3$

2. Do Gaussian blur: convolving the grey image with a Gaussian filter.

- Convolving:

Slide window

16	32	160	80	16
0	48	112	32	176
0	0	48	16	16
0	8	4	0	96
8	0	4	8	160

*

$\frac{1}{16}$

x

1	2	1
2	4	2
1	2	1

=

11	43	73	64	38
10	44	72	71	

3. Extend the size of image from HxWx1 to HxWx3 (to save the image)

- $R(i, j) = \text{grey}(i, j)$
- $G(i, j) = \text{grey}(i, j)$
- $B(i, j) = \text{grey}(i, j)$

Introduction: Blur algorithm (cont.)

1. Convert RGB image to grey image:

- $\text{grey}(i, j) = (R(i, j) + G(i, j) + B(i, j)) / 3$

2. Do Gaussian blur: convolving the grey image with a Gaussian filter.

- Convolving:

Slide window

16	32	160	80	16
0	48	112	32	176
0	0	48	16	16
0	8	4	0	96
8	0	4	8	160

$$* \frac{1}{16} \times$$

1	2	1
2	4	2
1	2	1

$$=$$

11	43	73	64	38
10	44	72	71	58

- $\rightarrow (80 \cdot 1 + 16 \cdot 2 + 32 \cdot 2 + 176 \cdot 4 + 16 \cdot 1 + 16 \cdot 2) / 16 = 58$

3. Extend the size of image from HxWx1 to HxWx3 (to save the image)

- $R(i, j) = \text{grey}(i, j)$
- $G(i, j) = \text{grey}(i, j)$
- $B(i, j) = \text{grey}(i, j)$

Introduction: Sobel filter

- Sobel filter:

- Gradient of horizontal direction

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

- Gradient of vertical direction

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- Sobel filter is written in “mask_Sobel.txt”

- The first number is the filter size
- The second line is Gx
- The third line is Gy
- Note: the size of Gx and Gy must be the same.



9
1 0 -1 2 0 -2 1 0 -1
-1 -2 -1 0 0 0 1 2 1

Introduction: Edge Detection algorithm

1. Convert RGB image to grey image:
 - $\text{grey}(i, j) = (R(i, j) + G(i, j) + B(i, j)) / 3$
2. Convolve the grey image with Gx filter and Gy filter, respectively.
→ Get image_x and image_y
3. Compute:
$$\text{Image}(i, j) = \sqrt{\text{image_x}(i, j)^2 + \text{image_y}(i, j)^2}$$
4. Extend the size of image from HxWx1 to HxWx3 (to save the image)
 - $R(i, j) = \text{Image}(i, j)$
 - $G(i, j) = \text{Image}(i, j)$
 - $B(i, j) = \text{Image}(i, j)$

Thread

- Only use:
`#include <pthread.h>`
- Declare:
`pthread_t thread1, thread2;`
- Functions:
 - `int pthread_create(pthread_t * thread, const pthread_attr_t * attr, void * (*start_routine)(void *), void *arg);`
 - `int pthread_join(pthread_t th, void **thread_return);`
 - wait for termination of another thread
 - `void pthread_exit(void *retval);`

Synchronization - mutex lock

- Only use:
`#include <pthread.h>`
- Declare: (global variable)
`pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;`
- Functions:
 - `pthread_mutex_lock()`
 - acquire a lock on the specified mutex variable. If the mutex is already locked by another thread, this call will block the calling thread until the mutex is unlocked.
 - `pthread_mutex_unlock()`
 - unlock a mutex variable. An error is returned if mutex is already unlocked or owned by another thread.
 - `pthread_mutex_trylock()`
 - attempt to lock a mutex or will return error code if busy. Useful for preventing deadlock conditions.

Synchronization - semaphore

- `#include <pthread.h>`
 - Declare: (global variable)
`pthread_cond_t cond1 = PTHREAD_COND_INITIALIZER;`
 - Functions:
 - `pthread_cond_wait`
 - `pthread_cond_signal`
 - `pthread_cond_broadcast`
- `#include <semaphore.h>`
 - Declare: (global variable)
`sem_t sem1;`
 - Functions:
 - `int sem_post(sem_t *);`
 - `int sem_wait(sem_t *);`
 - `int sem_close(sem_t *);`

Synchronization - semaphore (cont.)

- Only use:
#include <pthread.h>
#include <semaphore.h>
- Design your method to implement synchronization.
- See details below:
 - <pthread.h>
<http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html>
 - <semaphore.h>
<http://pubs.opengroup.org/onlinepubs/7908799/xsh/semaphore.h.html>

Synchronization

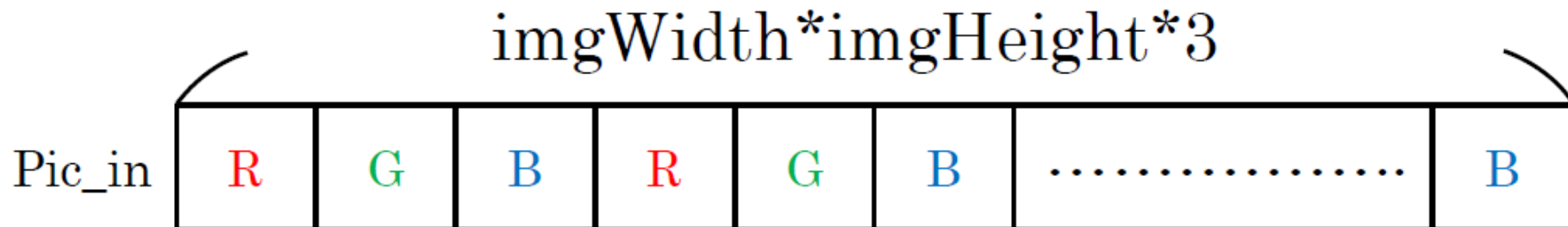
- In HW3, you need use **BOTH** mutex lock and semaphore.
- And for each part, you must use **at least one** of them.
- For example:
 - ✓ HW3-1 with mutex lock, and HW3-2 with semaphore.
 - ✓ HW3-1 with semaphore, and HW3-2 with mutex lock.
 - ✓ HW3-1 with mutex lock and semaphore, and HW3-2 with mutex lock.
 - ✗ HW3-1 with mutex lock and semaphore, and HW3-2 without them.
 - ✗ HW3-1 with mutex lock, and HW3-2 with mutex lock.

Synchronization (optional)

- Idea about synchronization (**you can try other way**):
 - Mutex lock:
 - Pthread: p1, p2, p3
 - Zero-matrix: Z (size = HxWx**1**)
 - p1, p2, p3 do convolve the R,G,B-image with filter, respectively.
 - p1, p2, p3 **add** their results into Z when doing convolve.
 - Semaphore:
 - Pthread: p1, p2
 - p1: convert RGB image to gray image.
 - p2: do convolve the grey image with filter.
 - After p1 has converted enough pixels (but p1 hasn't finished), p2 start.

Image read & write

- Only use “bmpReader.h” and “bmpReader.cpp” we provide to read or write images. (Don’t modify “bmpReader.h” and “bmpReader.cpp”.)
- **Unsigned char** Pic_in[x] array
- Each pixel is represented by three values.
R G B R G B.....
- Accessing the i-th row, j-th col pixel :
 - $\text{pic_in}[3*(i*\text{imgWidth}+j)+\text{color}]$, color = 0,1,2
- **Be careful of the conversion between integer, double (float), and unsigned char.**



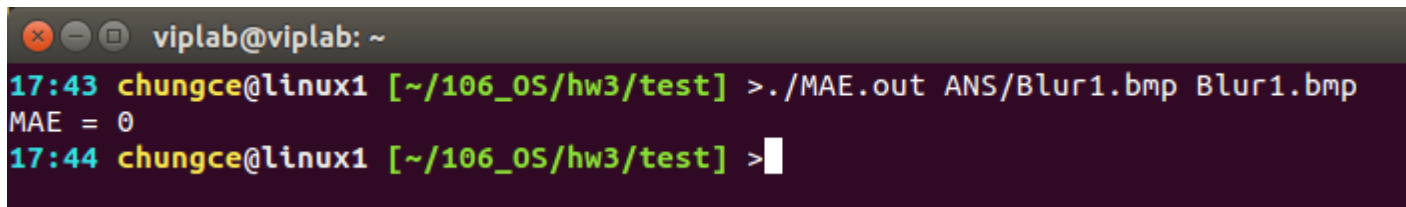
Input & output format

- Input: 5 BMP images and a mask file
 - Image name: input1.bmp, input2.bmp, input3.bmp, input4.bmp, input5.bmp
 - Mask file name:
HW 3-1: mask_Gaussian.txt
HW 3-2: mask_Sobel.txt
 - Input location:
In the same folder with cpp file.
- Output: 5 BMP images for each part
 - Name:
HW 3-1: Blur1.bmp, Blur2.bmp, Blur3.bmp, Blur4.bmp, Blur5.bmp
HW 3-2: Sobel1.bmp, Sobel2.bmp, Sobel3.bmp, Sobel4.bmp, Sobel5.bmp
 - Output location:
In the same folder with cpp file.

Score

1. Correctness: from 0 to 10 pts for each part

- Mean Absolute Error: $MAE(X, Y) = \frac{1}{W \cdot H \cdot 3} \sum |X(i, j, c) - Y(i, j, c)|$, where $c=0,1,2$
- If $MAE=0$, then your output is correct.
- We will give you “MAE.out”. Then you can use it to check the correctness.
Use the following command:
`./MAE.out [image 1] [image 2]`

A terminal window with a dark background and light-colored text. The window title is 'viplab@viplab: ~'. The prompt is '17:43 chungce@linux1 [~/106_OS/hw3/test] >'. The user enters './MAE.out ANS/Blur1.bmp Blur1.bmp'. The output is 'MAE = 0'. The prompt changes to '17:44 chungce@linux1 [~/106_OS/hw3/test] >'.

```
viplab@viplab: ~  
17:43 chungce@linux1 [~/106_OS/hw3/test] > ./MAE.out ANS/Blur1.bmp Blur1.bmp  
MAE = 0  
17:44 chungce@linux1 [~/106_OS/hw3/test] >
```

- If you get “Permission denied” (拒絕不符權限操作), use the following command:
`chmod +x MAE.out`

Score (cont.)

2. Speed: from 0 to 40 pts for each part

- We will provide “example_hw3-1.cpp”, which doesn’t use multithread programming and synchronized, as a speed baseline.
- We will give you “Speed.sh”.
Use the following command:
sh Speed.sh

	HW3-1	HW3-2
Baseline	1544042 μ s	1430390 μ s
filter size	5*5	3*3

```

viplab@viplab: ~
17:44 chungce@linux1 [~/106_OS/hw3/test] >g++ -std=c++11 -pthread example_hw3-1.cpp
17:45 chungce@linux1 [~/106_OS/hw3/test] >sh Speed.sh
Input a number of times to run './a.out' : 10

Run time:
Finished once.
Avg time: 1544042  $\mu$ s
17:45 chungce@linux1 [~/106_OS/hw3/test] >

```

Speedup	HW3-1	HW3-2
< 0.9	0	0
0.9~1.1	0	20
1.1~1.5	20	25
1.5~2	30	30
2~3	35	35
> 3	40	40
*This is a provisional standard table, we may modify after checking all students' HW3.		

- We will use it to compute your average run time. (Input = 10 fixed.)

Score (cont.)

3. Report (20 pts):

- Format is in “report.docx”
- Written in English or Chinese, up to 2 pages

4. Final score (Total 100 pts):

$$\frac{\text{Speed score 1} * \text{Correctness score 1}}{\text{Speed score 2} * \text{Correctness score 2}} / 10 + \frac{\text{Report score}}{\text{Report score}}$$

5. Others:

- Must use thread, mutex lock and semaphore. (otherwise -10pts)
- Use other library NOT in “example_hw3-1.cpp”: will get 0pt directly
- Wrong input/output format: -10pts
- Wrong hand-in file name: -10pts
- Copy or be copied: will get 0pt directly

(you only can use these library)
 “example_hw3-1.cpp”:

```
#include "bmpReader.h"
#include "bmpReader.cpp"
#include <stdio.h>
#include <iostream>
#include <math.h>
#include <pthread.h>
#include <semaphore.h>
using namespace std;
```



Requirements

- Use **NCTU CS Workstation linux1~linux6** as your programming environment.
(No bsd1~bsd6)
- We **only use** these commands on NCTU CS Workstation linux**2**:
 - `g++ -std=c++11 -pthread StudentID_hw3-1.cpp`
 - `g++ -std=c++11 -pthread StudentID_hw3-2.cpp`
 - `./a.out`
 ↑ **no argument**
- Put the 3 files into a compressed file named “**StudentID_OS_hw3.zip**”
 - StudentID_hw3-1.cpp
 - StudentID_hw3-2.cpp
 - report.docx (or report.pdf)
- **Deadline: 2017/12/02 (Saturday) 23:59**