OS HW4

Operating System 107 Fall

W.J.Tsai 蔡文錦 教授

TA 劉晏
蘇聖雅
莊侑穎
盧彥廷
黃資捷

Background

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.

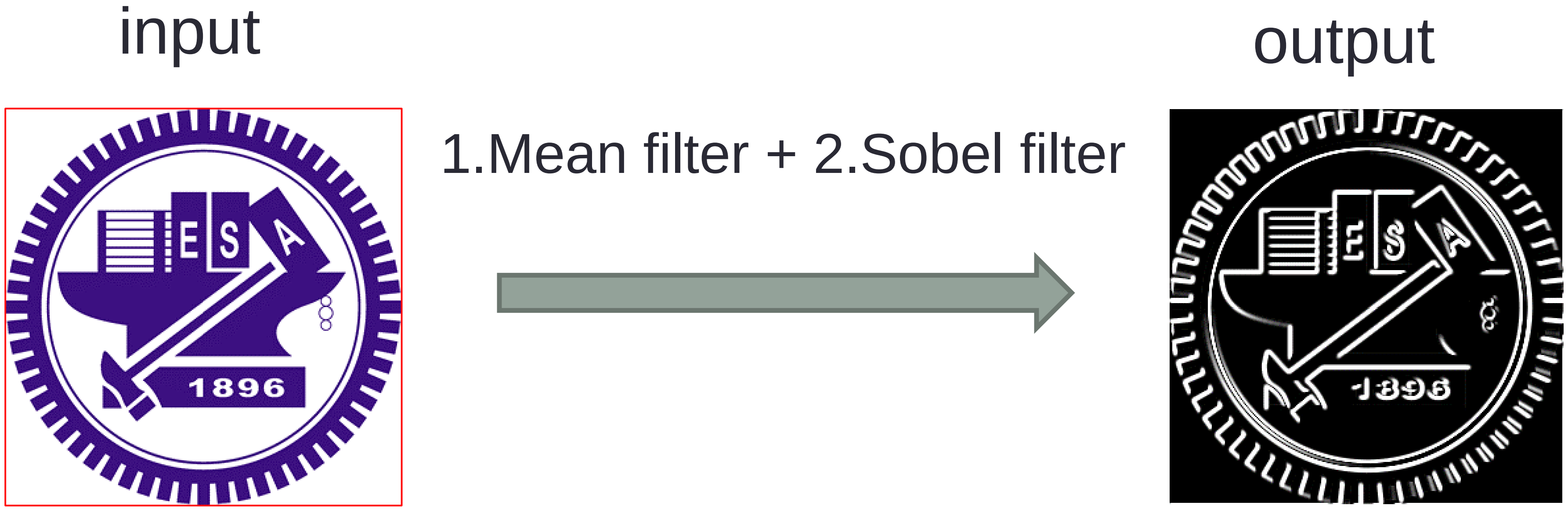
Synchronization - semaphore

- #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 *);

Goal

- **Problem1: (80%)**
- Implement image processing by using threads and synchronization.
  - 1. Smoothing images with Mean filter
  - 2. Edge Detection with Sobel filter
  - ***Please follow the order, Mean filter first ,then Sobel filter.***
  - *****Only create two threads(One for Mean filter , one for Sobel filter).*****



input

1.Mean filter + 2.Sobel filter

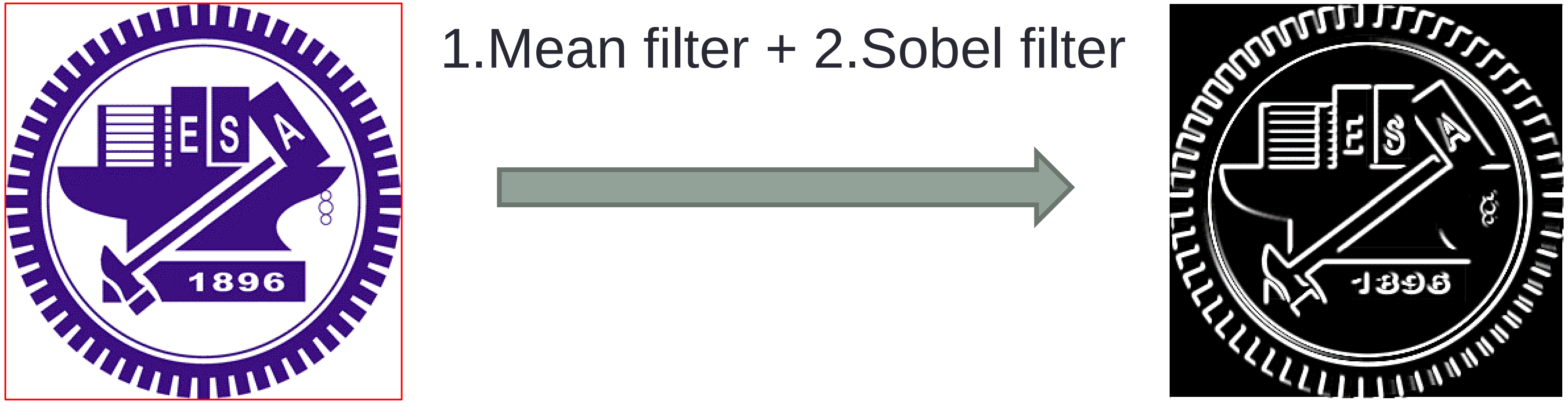
output

Goal

- **Problem2: (bonus 15%)**
- Implement image processing by using threads and synchronization.
  - 1. Smoothing images with Mean filter
  - 2. Edge Detection with Sobel filter
  - ***Please follow the order, Mean filter first ,then Sobel filter.***
  - *****Create more than two threads*****

input

output



1.Mean filter + 2.Sobel filter

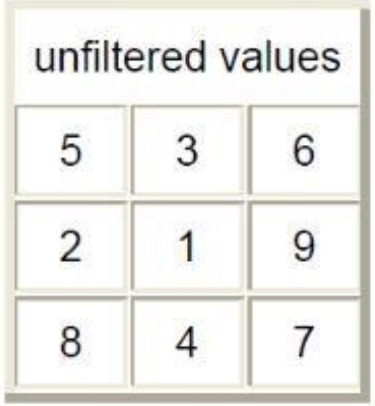
- Notice: You need to do Mean filter and Sobel filter at the same time.
- In HW4, you need use at least one of mutex lock and semaphore.

- For example:
  - ✓HW4 with mutex lock.
  - ✓HW4 with semaphore.
  - ✓HW4 with mutex lock and semaphore.

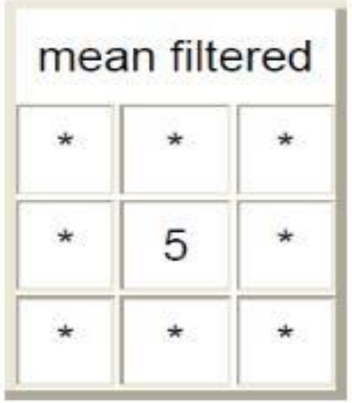
  - ✗HW4 without any one of mutex lock and semaphore.

Introduction: Mean filter

For example:



unfiltered values		
5	3	6
2	1	9
8	4	7



mean filtered		
*	*	*
*	5	*
*	*	*

$5 + 3 + 6 + 2 + 1 + 9 + 8 + 4 + 7 = 45$

$45 / 9 = 5.$

Introduction: Algorithm

1. Convert RGB image to grey image:
   - grey(i, j) = (R(i, j)+G(i, j)+B(i, j))/3

2. Smoothing: convolving the grey image with a Mean filter.

3. Extend the size of image from HxWx1 to HxWx3 (to save the image)
   - R(i, j) = grey(i, j)
   - G(i, j) = grey(i, j)
   - B(i, j) = 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:
   - grey(i, j) = (R(i, j)+G(i, j)+B(i, j))/3
2. Convolving the grey image with Gx filter and Gy filter, respectively.

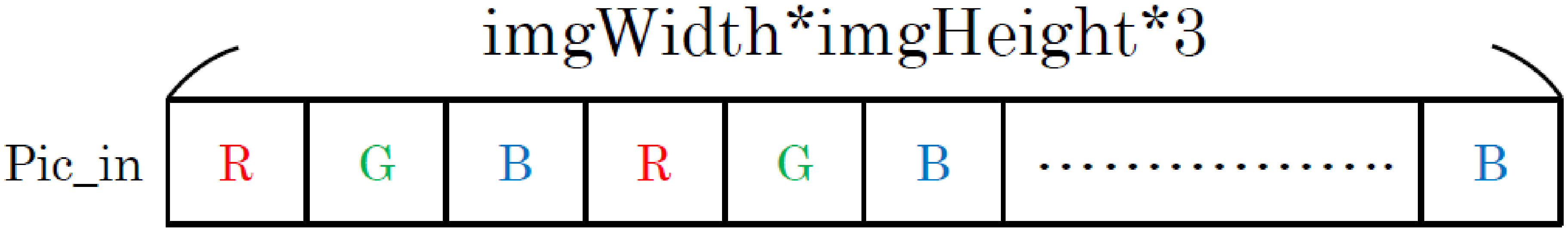
   → Get image_x and image_y
3. Compute:
   Image(i, j) = sqrt(image_x(i, j)*image_x(i, j) + image_y(i, j)* image_y(i, j))
4. Extend the size of image from HxWx1 to HxWx3 (to save the image)
   - R(i, j) = Image(i, j)
   - G(i, j) = Image(i, j)
   - B(i, j) = Image(i, j)

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”.)

- Each pixel is represented by three values.
  R G B R G B…..
- Accessing the i-th row, j-th col pixel :
  - pic_in[3*(i*imgWidth+j)+color], color = 0,1,2

- Be careful of the conversion between integer, double (float), and unsigned char.



imgWidth*imgHeight*3

Pic_in

R	G	B	R	G	B	……………….	B

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:
    mask_Sobel.txt
  - Input location:
    In the same folder with cpp file.

- Output: 5 BMP images for each part
  - Image name: output1.bmp, output2.bmp, output3.bmp, output4.bmp, output5.bmp
  - Output location:
    In the same folder with cpp file.

Score

1. **Correctness score**: from 0 to 2 pts for each images (5 images)
   - Mean Absolute Error: $\mathrm{MAE}(\mathrm{X},\mathrm{Y}) = \frac{1}{\mathrm{W}\cdot H\cdot 3}\Sigma|\mathrm{X}(i,j,\mathrm{c}) - \mathrm{Y}(i,j,\mathrm{c})|$,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]
     ```

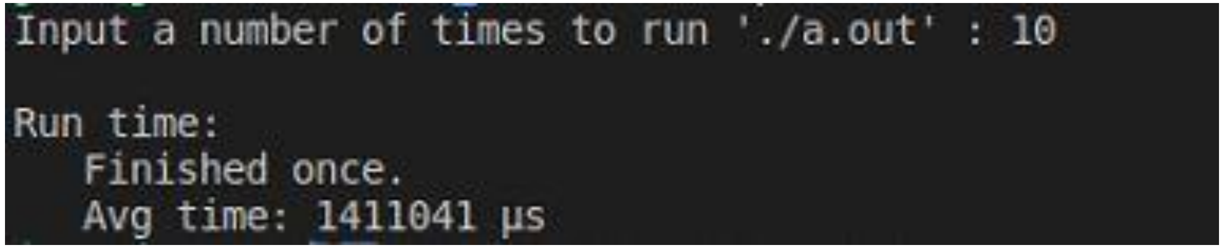
   - If you get “Permission denied” (拒絕不符權限操作), use the following command:
     ```
     chmod +x MAE.out
     ```

Score (cont.)

2. **Speed score**: from 0 to 40 pts
3. We will provide “example_hw4.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

	HW4
Baseline	1411041 μs
filter size	3*3

Speedup	HW4
< 0.9	0
0.9~1.1	0
1.1~1.3	25
1.3~1.5	30
1.5~1.7	35
> 1.7	40



```
Input a number of times to run './a.out' : 10

Run time:
   Finished once.
   Avg time: 1411041 μs
```

- We will use it to compute your average run time. (Input = 10 fixed.)
- **This is a provisional standard table, we may modify after checking all students’ HW4.**

Score (cont.)

3. Report (20 pts):
   - Format is in “report.docx”
   - Written in English or Chinese, up to 2 pages
4. Final score (Total 115 pts):
   Speed score * (Correctness score /5) +
   Report score + bonus
5. Others:
   - Without mutex lock or semaphore.: will get 0pt directly
   - Use other library NOT in “example_hw4.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_hw4.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

- We only use these commends:
- g++ -std=c++11 -pthread StudentID_hw4.cpp
- g++ -std=c++11 -pthread StudentID_hw4_bonus.cpp
- ./a.out
  ↑ no argument
- Put the 3 files into a compressed file named “**StudentID_OS_hw4.zip**”
  - StudentID_hw4.cpp
  - StudentID_hw4_bonus.cpp
  - report.docx (or report.pdf)
- Deadline: 2018/12/23 (Sunday) 23:59