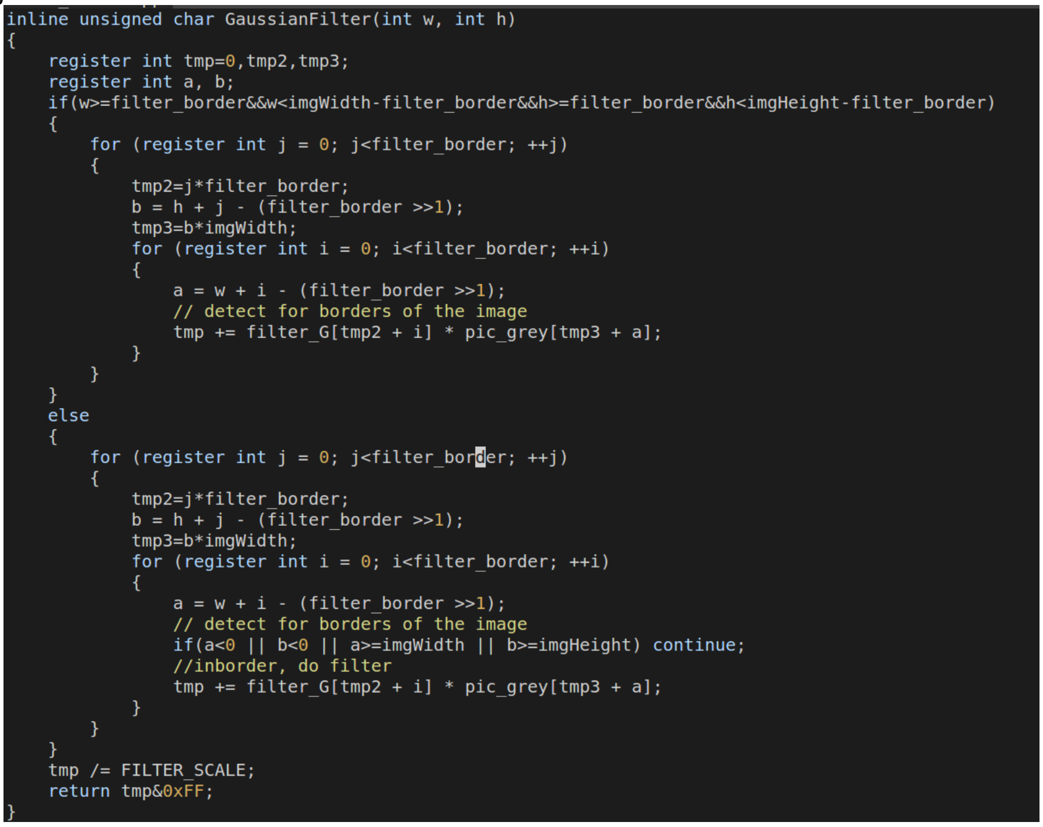
Operating System HW3 Report

0416324 胡安鳳

1.Detailed description of the implementation:

(1) Since there are many duplicated mathematical operation, such as multiplication in the covolution process, we may save some value in advance which have been calculated before to accelerated the process of filter algorithm, which is a bit close to the memorised algorithm.
(2) Use certain amount of thread for parallel processing.
(3) Use register variable to change the memory hierarchy for faster memory access since the register is much faster than the memory with respect to accessing speed.
Aforementioned is an example in Gaussain Filter, the same is true for Sobel Filter
(4) Use calloc insteal of malloc to trade a bit time faster.
(5) Do the border part first to cut down the time overhead(branch) by if-statement. See the following

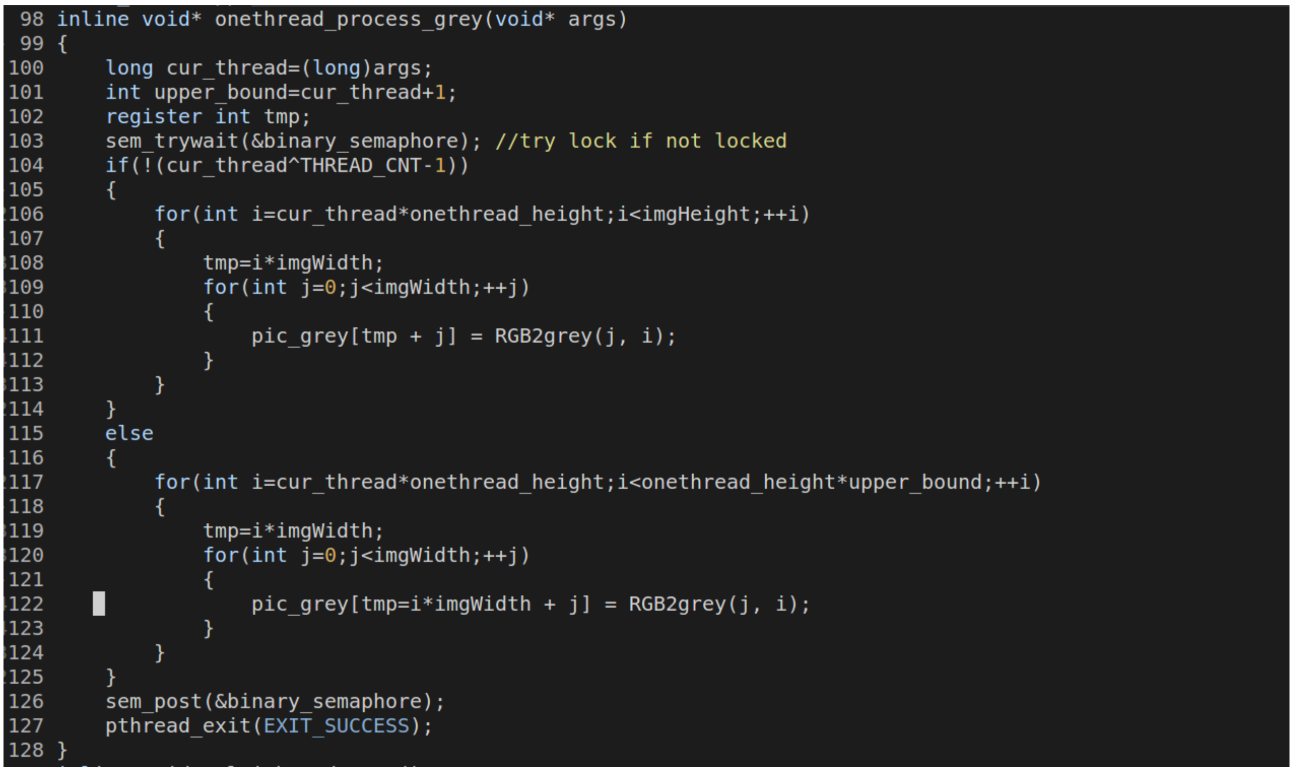


```
inline unsigned char GaussianFilter(int w, int h)
{
    register int tmp=0,tmp2,tmp3;
    register int a, b;
    if(w>=filter_border&&w<imgWidth-filter_border&&h>=filter_border&&h<imgHeight-filter_border)
    {
        for (register int j = 0; j<filter_border; ++j)
        {
            tmp2=j*filter_border;
            b = h + j - (filter_border >>1);
            tmp3=b*imgWidth;
            for (register int i = 0; i<filter_border; ++i)
            {
                a = w + i - (filter_border >>1);
                // detect for borders of the image
                tmp += filter_G[tmp2 + i] * pic_grey[tmp3 + a];
            }
        }
    }
    else
    {
        for (register int j = 0; j<filter_border; ++j)
        {
            tmp2=j*filter_border;
            b = h + j - (filter_border >>1);
            tmp3=b*imgWidth;
            for (register int i = 0; i<filter_border; ++i)
            {
                a = w + i - (filter_border >>1);
                // detect for borders of the image
                if(a<0 || b<0 || a>=imgWidth || b>=imgHeight) continue;
                //inborder, do filter
                tmp += filter_G[tmp2 + i] * pic_grey[tmp3 + a];
            }
        }
    }
    tmp /= FILTER_SCALE;
    return tmp&0xFF;
}
```

2.How to use the semaphore or the mutex in the right time

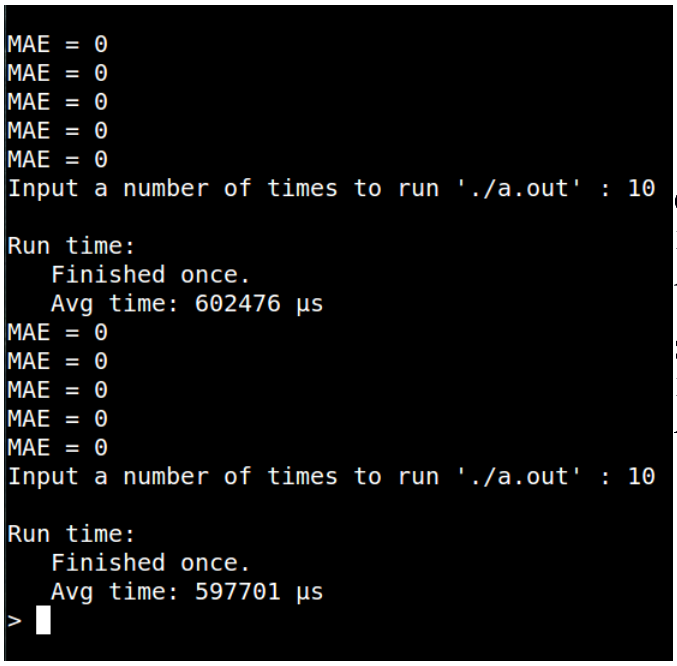
Right before entering the critical section of the shared data, trywait it for semaphore to indicate that a certain process is going to enter the critical section and get the lock if there is no thread in the critical section.

The same idea can be implemented for mutex, where mutex is just a kind of binary semaphore



```
inline void* onethread_process_grey(void* args)
{
    long cur_thread=(long)args;
    int upper_bound=cur_thread+1;
    register int tmp;
    sem_trywait(&binary_semaphore); //try lock if not locked
    if(!(cur_thread^THREAD_CNT-1))
    {
        for(int i=cur_thread*onethread_height;i<imgHeight;++i)
        {
            tmp=i*imgWidth;
            for(int j=0;j<imgWidth;++j)
            {
                pic_grey[tmp + j] = RGB2grey(j, i);
            }
        }
    }
    else
    {
        for(int i=cur_thread*onethread_height;i<onethread_height*upper_bound;++i)
        {
            tmp=i*imgWidth;
            for(int j=0;j<imgWidth;++j)
            {
                pic_grey[tmp=i*imgWidth + j] = RGB2grey(j, i);
            }
        }
    }
    sem_post(&binary_semaphore);
    pthread_exit(EXIT_SUCCESS);
}
```

3.My Speed

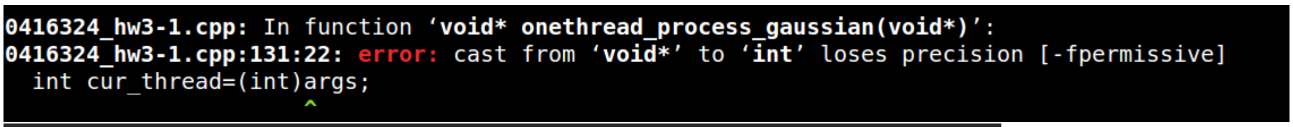


```
MAE = 0
MAE = 0
MAE = 0
MAE = 0
MAE = 0
Input a number of times to run './a.out' : 10

Run time:
   Finished once.
   Avg time: 602476 µs
MAE = 0
MAE = 0
MAE = 0
MAE = 0
MAE = 0
Input a number of times to run './a.out' : 10

Run time:
   Finished once.
   Avg time: 597701 µs
> 
```

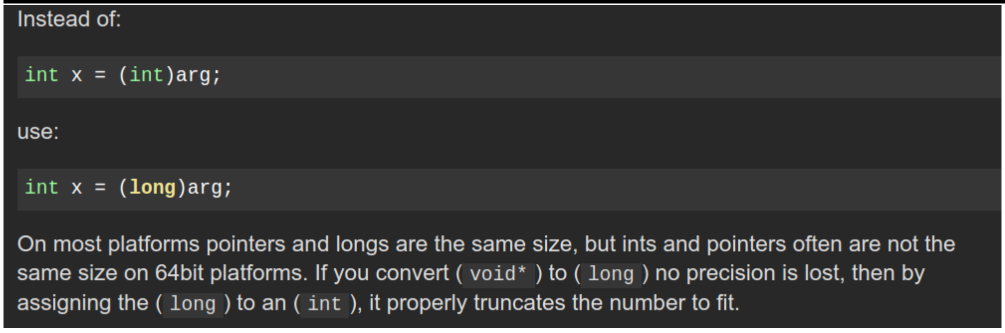
Gaussian filter
1544042/602476=2.56
Accelerate rate 156%

Sobel Filter
1430390/597701=2.39
Accelerate rate 139%

4.Problems encountered and solutions



```
0416324_hw3-1.cpp: In function ‘void* onethread_process_gaussian(void*)’:
0416324_hw3-1.cpp:131:22: error: cast from ‘void*’ to ‘int’ loses precision [-fpermissive]
  int cur_thread=(int)args;
                      ^
```



Instead of:

```
int x = (int)arg;
```

use:

```
int x = (long)arg;
```

On most platforms pointers and longs are the same size, but ints and pointers often are not the same size on 64bit platforms. If you convert (`void*`) to (`long`) no precision is lost, then by assigning the (`long`) to an (`int`), it properly truncates the number to fit.