

今回の実習では以前の二回の実習と違う画像処理ライブラリーを使用しました。以前は libgd という C のライブラリーを使用しましたが今回は C++ の OPENCV というライブラリーを使用しました。この OPENCV はすごく多いバージョンがあって僕が提出する実行形式を OPENCV3.4.0 を使ったコンパイルしました。そのため実行するコンピュータの OPENCV のバージョンと異なる場合実行できないことがあります。そのため実行環境に当てて再度コンパイルをする必要があります。使うコンパイラは g++ で以下のようなコマンドを使って再度コンパイルしてください。

g++ -o [Execute File] [Source File] \$(pkg-config opencv --cflags --libs)

必須課題 1 では libgd も使うので最後の部分に -lgd を追加してください。

必須課題 1

この課題は以前まで使った libgd を利用して画像ファイルを開き OPENCV を使って保存するプログラムを書くことです。libgd と OPENCV で一番違うところは libgd では必要な情報を構造体で持ってくる時先に定義されていたマクロを使用しましたが OPENCV ではクラスで管理されておりメソッドを使ってアクセスします。以下は実行コマンドの形式です。

./ass1 [Input Image] [Output Filename]

出力画像が入力画像と同じため実行画面でプログラムがうまく働くことを見せます。

```
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ ls
add1.cpp  ass1  ass2  ass3  ass4  ass5  at_test.cpp  lena.jpg  string_test.cpp  test.cpp  video_test  video_test.cpp
add1.cpp  ass1.cpp  ass2.cpp  ass3.cpp  ass4.cpp  ass5.cpp  cvlena.jpg  src.JPG  test  time_test.JPG  video_test2
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ ./ass1 lena.jpg dst1.JPG
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ identify lena.jpg
lena.jpg JPEG 225x225 225x225+0+0 8-bit sRGB 7.64KB 0.000u 0:00.000
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ identify dst1.JPG
dst1.JPG JPEG 225x225 225x225+0+0 8-bit sRGB 16.9KB 0.000u 0:00.000
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ display dst1.JPG
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$
```

任意課題 1

OPENCV で画素値にアクセスするためには at メソッドを使います。しかしこのメソッドはかな時間がかかる方法です。ポインタを利用するとより高速にアクセスができます。この課題では at を利用したアクセスとポインタを利用したアクセスを両方を実行して比べます。

実験方式は一つの画像ファイルを入力で受けてそれを OPENCV を使って読み込み両方の方法を使って全部の画素に対してアクセスします。それぞれの方法を百回行ってその平均時間を出力します。まず各方法をコードでみせると以下のようです。

```
for(y=0;y<height;y++) {
    for(x=0;x<width;x++) {
        Vec3b bgr = m.at<Vec3b>(y,x);
    }
}

for(y=0;y<height;y++) {
    Vec3b *p = m.ptr<Vec3b>(y);
    for(x=0;x<width;x++) {
        Vec3b bgr = p[x];
    }
}
```

下はプログラム実行結果です。

```
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ identify lena.jpg
lena.jpg JPEG 225x225 225x225+0+0 8-bit sRGB 7.64KB 0.000u 0:00.000
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ ./add1 lena.jpg
0.000558 sec when using at
0.000441 sec when using pointer
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ identify time_test.JPG
time_test.JPG JPEG 3264x2448 3264x2448+0+0 8-bit sRGB 5.966MB 0.170u 0:00.179
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ ./add1 time_test.JPG
0.080631 sec when using at
0.068726 sec when using pointer
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$
```

小さい画像と大きい画像に対してプログラムをそれぞれ実行しました。lena.jpg に対しては約 20%ほどポインタを使った方が早いです。time_test.jpg に対しては約 15%ほどポインタを使った方が早いです。

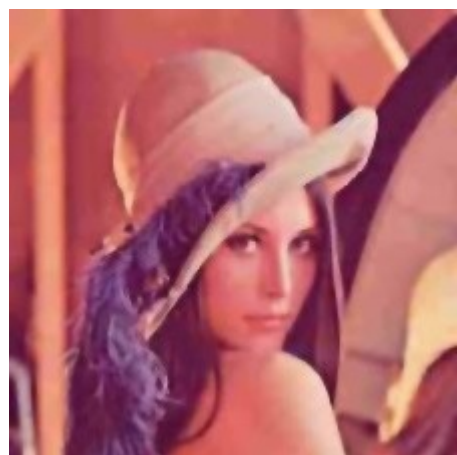
この違いは安全性のためのチェックをするかしないかからくるものだと予想できます。at メソッドはアクセスしようとする座標に対して範囲外なのかチェックしますがポインタを使った方法はそれをしません。

必須課題 2

必須課題 2 は OPENCV を使って何らかの空間フィルタを画像にかけるプログラムを書くことです。先週の課題でフィルタを直接作って適用しました。OPENCV は色々な種類のフィルタを関数として提供するので直接作る必要はありません。ただコード一行でフィルタをかけることができます。この課題では OPENCV が提供するフィルタの中でバイラテラルフィルタを入力画像に適用します。以下は実行コマンドの形式です。

`./ass2 [Input Image] [Output Image]`

処理前と処理後の画像です。



ノイズは減ったんですが少しボヤけました

必須課題 3

この課題は OPENCV を利用してアフィン変換あるいは射影変換を実行することです。僕は画像を横に鏡映するアフィン変換を実行しました。アフィン変換の時使う行列は以下のようです。

$$\begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

この行列で a,b,c,d は先週やった線形変換で t は平行移動を指定します。以下は実行コマンドの形式と結果です。

./ass3 [Input File] [Output File]



必須課題 4

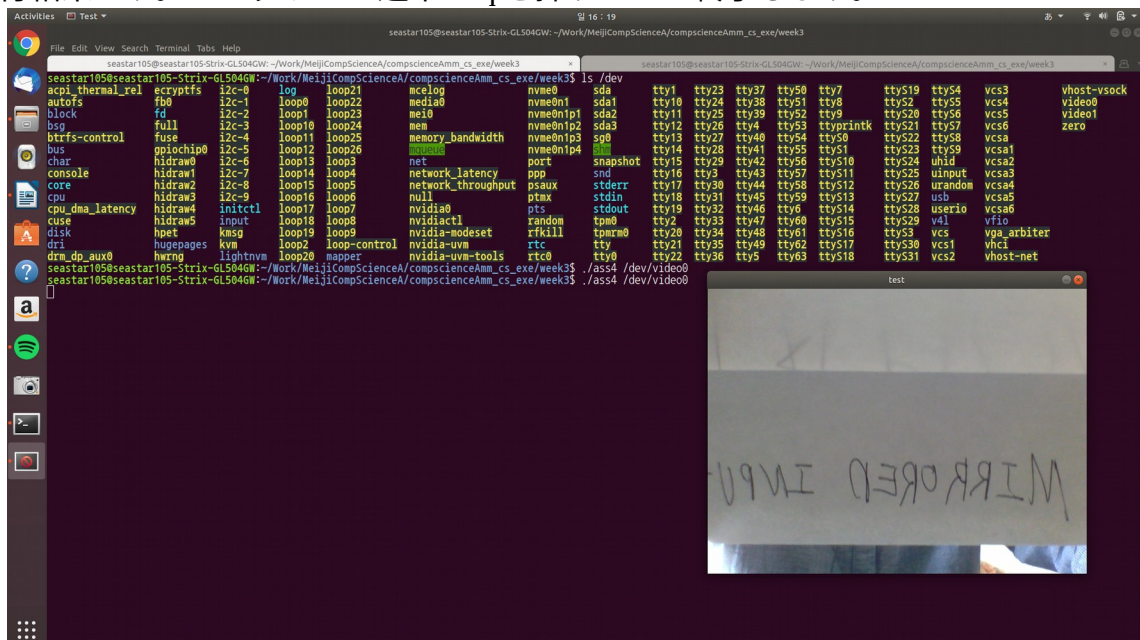
この課題ではカメラから入力をとって何らかの画像処理を行って出力をするプログラムを書くことです。僕が書いたプログラムは必須課題 3 で実行した画像処理をカメラからの入力画像に対して実行した後出力します。以下は実行コマンドの形式です。

./ass4 [Camera device]

ここで Camera device はつかえるカメラの位置を言います。

```
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ ls /dev
acpi_thermal_rel  encryptfs  i2c-0      log         loop21     mcelog      nvme0      sda         tty1        tty23      tty37      tty50      tty7        tty519     tty54      vcs3        vhost-vsock
autofs            fb0        i2c-1      loop0       loop22     media0      nvme0n1    sda1        tty10       tty24      tty38      tty51      tty8        tty52      tty55      vcs4        video0
block             fd         i2c-2      loop1       loop23     mei0        nvme0n1p1  sda2        tty11       tty25      tty39      tty52      tty9        tty520     tty56      vcs5        video1
bsg               full       i2c-3      loop10      loop24     mem         nvme0n1p2  sda3        tty12       tty26      tty4       tty53      ttyprintk  tty521     tty57      vcs6        zero
btrfs-control     fuse       i2c-4      loop11      loop25     memory_bandwidth nvme0n1p3  sg0         tty13       tty27      tty40      tty54      tty50      tty522     tty58      vcsa
bus               gpiochip0  i2c-5      loop12      loop26     net         nvme0n1p4  smp         tty14       tty28      tty41      tty55      tty51      tty523     tty59      vcsa1
char              hidraw0    i2c-6      loop13      loop3      net         port        snapshot    tty15       tty29      tty42      tty56      tty510     tty524     tty57      vcsa2
console           hidraw1    i2c-7      loop14      loop4      network_latency psaux       stderr      tty16       tty3       tty43      tty57      tty511     tty525     tty59      vcsa3
core              hidraw2    i2c-8      loop15      loop5      network_throughput ppp         stderr      tty17       tty30      tty44      tty58      tty512     tty526     tty59      vcsa4
cpu               hidraw3    i2c-9      loop16      loop6      null         pts         stdin       tty18       tty31      tty45      tty59      tty513     tty527     tty59      vcsa5
cpu_dma_latency   hidraw4    initctl    loop17      loop7      nvme0        random      stdout      tty19       tty32      tty46      tty6       tty514     tty528     tty59      vcsa6
cuse              hidraw5    input      loop18      loop8      nvidia0      random      tpm0        tty2         tty33      tty47      tty60      tty515     tty529     tty59      vcsa7
disk              hpet       kmsg       loop19      loop9      nvidia-modeset rkill       tpmrm0      tty20       tty34      tty48      tty61      tty516     tty530     tty59      vcsa8
dri               hugepages  kvm        loop20      loop-control nvidia-uvm   rtc         tty         tty21       tty35      tty49      tty62      tty517     tty530     tty59      vcsa9
drm_dp_aux0       hvrng      lightnvm   loop20      mapper     nvidia-uvm-tools rtc0        tty0        tty22       tty36      tty5       tty63      tty518     tty531     tty59      vcsa10
```

上の図を見るとつながってる入出力装置を見えますが video0 と video1 がカメラ装置です。だからプログラムの入力として 'dev/video0' を渡さなきゃちゃんと作動しません。以下は実行結果です。プログラムは途中で q を押すことで終了します。



出力画像で文字を見ると左右反転処理を実行したのを確認できます。

必須課題 5

必須課題 5 はカメラからの入力に何らかの文字列を出力して出すことです。

プログラムの使用方法は 4 と同じです。

以下は実行結果です。出力する文字列は JHS です。

```
seastar105@seastar105-Strix-GL504GW:~/Work/MeijiCompScienceA/compscienceAmm_cs_exe/week3$ ./ass5 /dev/video0
```

