**2019년 2학기**

**프로그래밍과 문제해결**

**Assignment #4**

|  |  |
| --- | --- |
| 담당교수 | **윤은영** |
| 학번 | **20190084** |
| 학과 | **무은재학부** |
| 이름 | **권민재** |
| POVIS ID | **mzg00** |

|  |
| --- |
| **명예서약(Honor code)**  “나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.” |

**Problem: 이미지 채널 정합**

**1. 문제의 개요**

본 문제의 개요는 아래와 같다.

* 본 문제에서는 사진의 흔들린 3개의 채널을 정합하고자 한다.
* 사진의 가로, 세로 크기는 유동적이다.
* 정합하는데는 SSD와 NCC 알고리즘을 사용한다.

이 때 사용되는 구조 차트는 아래와 같이 표현될 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

* **입력부:** 입력부는 사용자로부터 메뉴를 입력받고, 그에 따른 행동을 수행한다. 또한, 사용자로부터 파일 이름을 입력받아 파일을 입력받는 역할도 수행한다.
* **처리부:** 처리부는 입력받은 이미지 파일을 가공하여 사용자의 입력에 따라 SSD나 NCC를 이용하여 이미지를 정합하여 저장한다.
* **출력부:** 각종 메뉴를 출력하고, 결과 파일을 출력한다.

**2. 알고리즘**

해당 프로그램의 의사 코드는 아래와 같다.

|  |  |
| --- | --- |
| Pseudo-code for ASSN4 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 | // 프로그램에 필요한 변수들, 전처리 상수들과 자료형은 미리 선언한 것으로 가정하자.  // w와 h는 대체로 x좌표, y좌표에 대응한다.  include essential headers <stdio.h>, <stdlib.h>, <string.h>, <math.h>  function *SSD\_NCC*  find RED optimal point by calling *calc* with channel RED and BLUE using SSD or NCC  find BLUE optimal point by calling *calc* with channel RED and BLUE using SSD or NCC  print result and make result file by calling *result*  function *calc*  loop while delta h and delta w is -15 ~ 15  set result height is original height – abs(delta height)  set result width is original width – abs(delta width)    if SSD mode, get sum of target pixel value  if NCC mode, get sum of (target pixel value \* src pixel value), (src pixel value^2), (taret pixel value^2)  get SSD or NCC value using the sum-values obtained above  if SSD or NCC value is optimal, set optimal value to current value, optimal delta to current delta, optimal reference point to current ref. point.  function *loadImage*  input file name from user and open it  if result of open file is NULL, return 0.  else, return 1.  function *readImage*  First, call *free\_pixel* to free **pixel**.  get width and height from file.  dynamic allocate array **pixel** to save pixel information.  First, allocate size 3\*(int\*\*) to **pixel**  Second, allocate size height\*(int\*) to each **pixel**[0~2]  Third, allocate size width\*(int) to each **pixel**[0~2][0~height]  get pixel value from file  function *writeImage*  set array **w\_pos** with channel number, sorted in ascending order based on delta w.  set array **h\_pos** with channel number, sorted in ascending order based on delta h.  // Then,  // w\_pos[0] = left channel, w\_pos[1] = middle channel, w\_pos[2] = right channel  // h\_pos[0] = lower channel, h\_pos[1] = middle channel, h\_pos[2] = upper channel  set calibration value to {-1\*w\_pos[0], -1\*h\_pos[0]}  add calibration value to all channel’s delta value.  set (w\_pos[i]’s reference w) to (w\_pos[2]’s delta w – w\_pos[i]’s delta w)  set (h\_pos[i]’s reference h) to (h\_pos[i]’s delta h)  set result width to width – (w\_pos[2]’s delta w)  set result height to height – (h\_pos[2]’s delta h)    while w, h in result width and result height,  print pixel value at (reference w + w, reference h + h) each channel.  function *result*  print result of registration  make result file by calling *writeImage*  function *menu*  loop below during user input is not valid  print menu by calling *printMenu*  if user input is not in 1, 2, 3, 4, print error message.  if user input is 2 or 3 but file is not valid, print error message.  function *printMenu*  if file is not valid, print “[1] 이미지 불러오기”  else, print “[1] 이미지 변경 – 현재 이미지: “ + current file name.  print “[2] 이미지 정합(SSD) …(중략)… 메뉴 선택>”  function *free\_pixel*  if **pixel** is NULL, return.  First, free size of width at **pixel**[0~2][0~height]  Second, free size of height at **pixel**[0~2]  Third, free size of channel count at **pixel**  function *swap*  swap a, b using address  function *main*  loop below during user input is not 4:  if mode is 1, load and read image by calling *loadImage, readImage*  if mode is 2 or 3, execute SSD or NCC by calling *SSD\_NCC*  if mode is 4, free dynamic-allocated memory by calling *free\_pixel* |

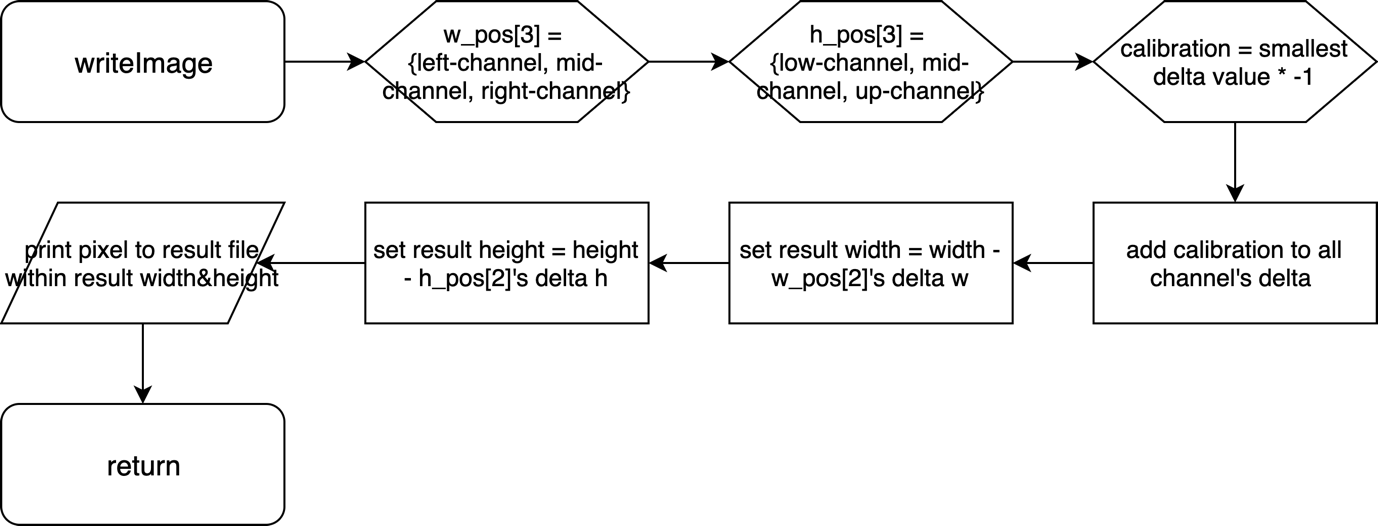
위의 의사 코드를 flowchart를 이용해서 나타내면 아래와 같다. (다음 페이지)

계산기, 전자기기이(가) 표시된 사진

자동 생성된 설명

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

****

**3. 프로그램 구조 및 설명**

**a) 메인 메뉴**

* 메인 메뉴는 기본적으로 무한 루프에 갇힌 구조이다.
* menu(&mode, P)를 호출하여 메인 메뉴를 출력하고, 입력 값을 mode에 저장한다.
* mode의 값이 1이라면, loadImage(&inputImgFile, fileName) && readImage(&inputImgFile, &img)를 호출하여 이미지 파일을 열고, 픽셀 정보를 img에 저장한다.
* mode의 값이 2 또는 3이라면, SSD\_NCC(&img, fileName, targets, 3 - mode)를 호출하여 경우에 맞게 SSD나 NCC를 이용한 정합을 실행한다.
* mode의 값이 4라면, free\_pixel(&img)를 호출하여, 동적 할당된 img의 배열 pixel을 free하고, 무한루프에서 탈출하여 프로그램을 종료한다.

**b) 이미지 로드**

* 함수 loadImage에서 이미지를 열고, 이미지를 여는데 실패했다면, 에러 메세지를 띄운다. 아니라면, 연속적으로 함수 readImage가 실행되어 img->width, img->height에 각각 이미지의 가로 길이와 세로 길이가 저장된다.
* 가로 세로 길이에 따라서 img->pixel[0~2][0~height][0~width] 범위가 될 수 있도록 pixel을 동적 할당한다. img->pixel[0], img->pixel[1], img->pixel[2]에 각각 red, green, blue 채널의 픽셀 2차원 배열이 들어간다.

**c) 이미지 정합**

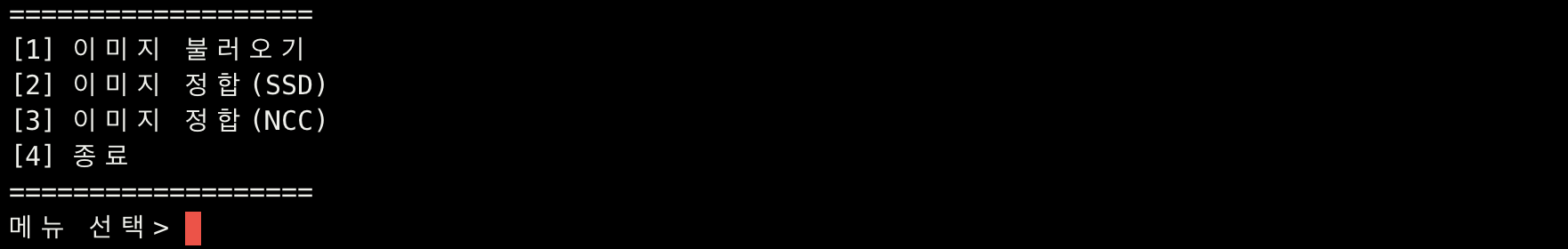
* 함수 SSD\_NCC에서는 calc(img, targets[2], targets[0], isSSD)을 호출하여 BLUE를 기준으로 하는 RED의 optimal한 경우를, calc(img, targets[2], targets[1], isSSD)을 호출하여 BLUE를 기준으로 하는 GREEN의 optimal한 경우를 구한다. 그 후, result(img, filename, resultFileName, targets, isSSD)를 호출하여 결과를 출력하고, 결과 파일을 작성한다.
* 함수 calc에서는 가로의 델타 값이 -15에서 15로 변하고, 세로의 델타 값이 -15에서 15로 변함에 따라, isSSD가 1이면 SSD 값을, 아니면 NCC 값을 구한다. 이후, 그 평가값에 따라 최적인 경우의 델타 값을 img 구조체에 업데이트한다.

**d) 이미지 출력**

* 함수 result에서는 calc의 결과를 출력하고, writeImage(img, resultFileName, targets)를 호출하여 결과 이미지 파일을 출력한다.
* 함수 writeImage에서는 img의 정보를 기반으로 정합된 이미지를 출력한다. w\_pos[3]에는 가로의 델타 값이 작은 순서대로 채널이 저장된다. 즉, w\_pos[0]에는 왼쪽에 있는 채널, w\_pos[1]에는 가운데 있는 채널, h\_pos[2]에는 오른쪽에 있는 채널의 번호가 저장된다. 비슷하게, h\_pos[0]에는 아래에 있는 채널, h\_pos[1]에는 가운데 있는 채널, h\_pos[2]에는 위에 있는 채널의 번호가 저장된다. 이때, 채널 w\_pos[0] \* -1을 가로 델타 값의 보정 값, h\_pos[0] \* -1을 세로 보정 값으로 취한 후, 모든 채널의 델타 값에 이들을 더한다. 그럼, 모든 델타 값들이 0 이상의 값으로 변환될 것이다. 이때, 파일에 작성할 시작점을 구성해보자. 채널 w\_pos[i]의 가로 시작점은 (w\_pos[2]의 가로 델타 값) – (w-pos[i]의 가로 델타 값)이 될 수 있다. 또한, h\_pos[i]의 세로 시작점은 h\_pos[i]의 델타 값이 될 수 있을 것이다. 이를 이용하여 설정된 각 채널의 시작점을 기준으로 출력 이미지의 가로, 세로 크기 만큼 파일에 작성하면 정합된 이미지 파일이 생긴다.

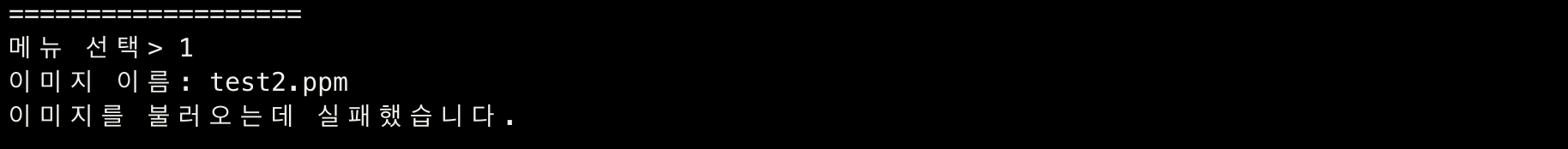
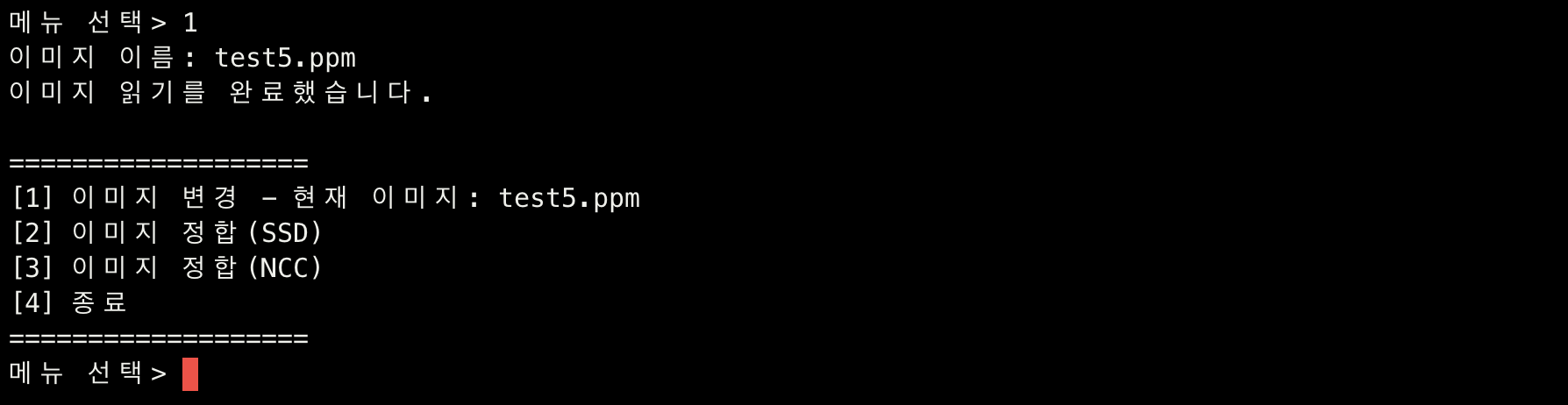
**4. 프로그램 실행 방법 및 예제**

**a) 메뉴 화면**



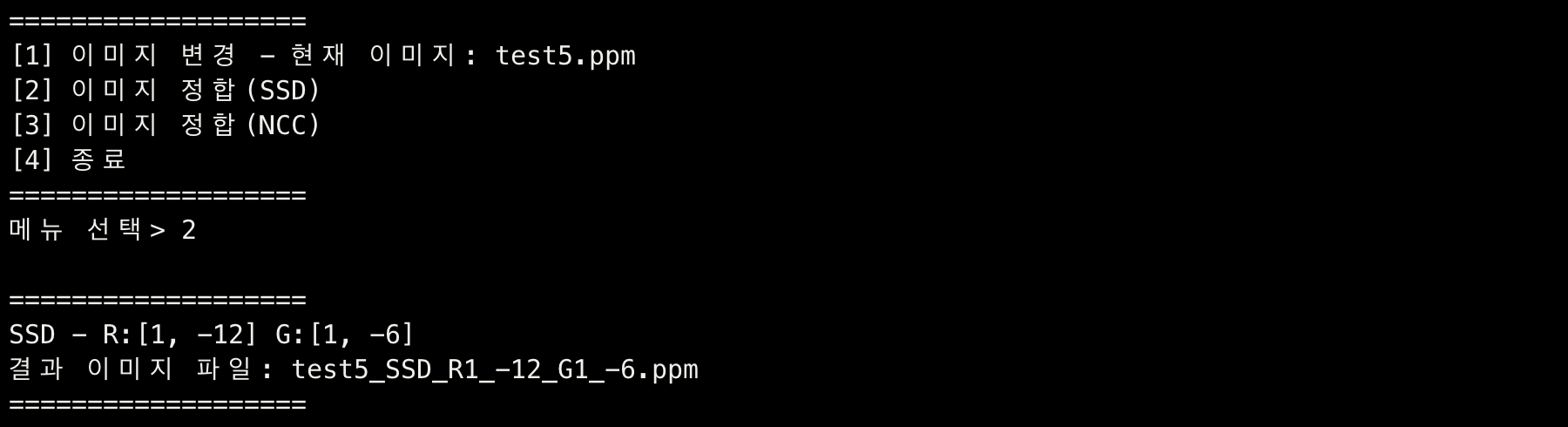
프로그램이 실행되면 위와 같이 메뉴가 출력되며, 하나의 기능이 완료되면 메뉴가 다시 출력된다.

**b) 이미지 불러오기**



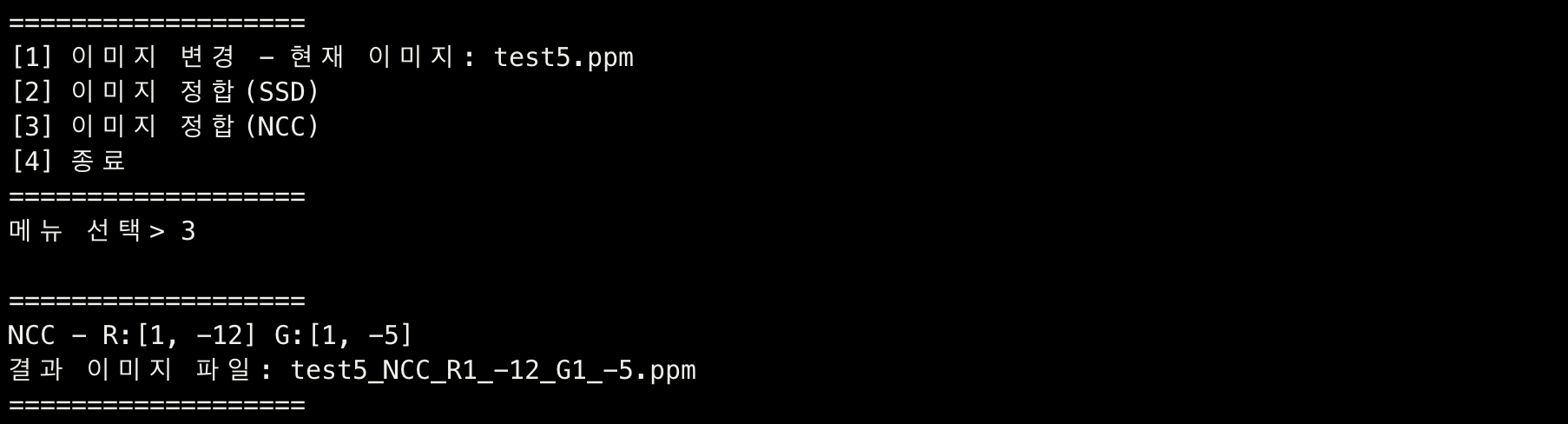
유효한 이미지인 경우에 이미지를 불러오고, 유효하지 않은 이미지인 경우에 적절한 에러 메세지를 띄우는 것을 볼 수 있다. 또한, 이미지를 불러온 후, 메인 메뉴가 변경된 것도 볼 수 있다.

**c) 이미지 정합 (SSD)**



2를 선택하면 SSD를 이용하여 정합을 수행하고, 그 결과를 출력하는 것을 볼 수 있다.

**d) 이미지 정합 (NCC)**



3을 선택하면 NCC를 이용하여 정합을 수행하고, 그 결과를 출력하는 것을 볼 수 있다. c)와 d)의 결과 이미지는 아래와 같다.

|  |  |  |
| --- | --- | --- |
| 사람, 여자이(가) 표시된 사진  자동 생성된 설명 | **벽, 사람, 실내, 여자이(가) 표시된 사진  자동 생성된 설명** | **벽, 사람, 실내, 앉아있는이(가) 표시된 사진  자동 생성된 설명** |
| test5.ppm | test5\_SSD\_R1\_-12\_G1\_-6.ppm | test5\_NCC\_R1\_-12\_G1\_-5.ppm |

**5. 토론**

* 이번 문제에서 큰 숫자를 다룸에 따라 2가지 부분에서 overflow에 관련된 오류가 발생하였다. 우선, 합계를 저장할 때 그 합계의 숫자가 int 범위로는 모자라서 long long 이나 unsigned int를 사용했어야 했다. 또한, NCC를 위해 sqrt 함수를 이용할 때, 제곱근 안에 들어가야 할 값을 그대로 넣으면 overflow로 인해 의도하지 않은 값 (음수)가 리턴되어, 제곱근 안에 들어가야 할 곱셈을 따로 분리하여 제곱근을 계산했어야 했다.
* math.h를 참조해보면, 함수 sqrt는 double형을 매개변수로 받는다. 이때, 정수에 있어서서는 double보다 long long이 커버하는 범위가 더 크기에, 함수 sqrt를 이용할 때 위와 같은 오류가 발생한 것으로 생각할 수 있다.
* 이 프로그램에서 타겟과 기준이 되는 채널을 배열 targets[]로 관리하기에, 이것만 바꾸면 손쉽게 기준 채널을 변경할 수 있다.

**6. 결론**

* 본 과제를 통해 파일 입출력을 통해 ppm 이미지 파일을 다룰 수 있었고, 동적 할당을 통해 다차원 배열을 관리할 수 있었다. 또한, 이를 통해 얻은 다차원 배열의 연산을 연습하는데 큰 도움이 되었다.
* 의도하지 않은 결과가 나올 때, 디버깅 툴을 이용하여 오류를 확인해보고, 자료형을 바꾸는 등 그에 따른 적절한 조치를 잘 취해야 함을 알 수 있었다.

**7. 개선방향**

* 주어진 이미지 파일의 가로 세로 크기가 작은 편 임에도 불구하고, 정합하는데 꽤 많은 소요시간이 드는 것을 알 수 있다. 지금 calc을 두 번 호출 하여 red와 green 채널을 정합하는데, calc이 탐색하는 for문 안에 red와 green 정합을 한 번에 수행한다면 실행 시간을 조금 더 줄일 수 있을 것이다.
* 결과 파일을 fopen할 때, 여러 이유로 파일을 여는데 성공하지 못할 때가 있는데, 이에 대한 예외 처리가 필요하다.