CSED101. Programming & Problem solving

Fall, 2019

Programming Assignment #4

(70 points)

김효민 (min00001@postech.ac.kr)

- ***Due:*** **2019.12.01 23:59**
- ***Development Environment: Windows Visual Studio 2019***

- ***제출물***
  - **C Code file (assn4.c)**
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것
  - **보고서 파일** (assn4.docx, assn4.hwp 또는 assn4.pdf)
    - AssnReadMe.pdf 를 참조하여 작성할 것.
    - **명예서약(Honor code)**: 표지에 다음의 내용을 포함한다. "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.
  - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

- ***주의사항***
  - 문제에서 제시한 요구사항을 반드시 지킬 것.
  - 모든 문제의 출력 형식은 아래의 예시들과 동일해야 하며, 같지 않을 시는 감점이 된다.
  - 제시되어 있는 파일이름으로 제출 할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
  - 컴파일 & 실행이 안되면 무조건 0점 처리된다.
  - **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
  - 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)
  - 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

■ Problem: 이미지 채널 정합

(소개)

칼라 사진기가 발명되기 전, 흑백 사진기는 대상의 밝기만 표현할 수 있었다. 얼마 후 3차원의 값(RGB 등)으로 색을 표현할 수 있다는 사실이 알려지고, 사진기에 각 색상의 필름을 씌워 여러 번 촬영하고 합성하여 칼라 사진을 만드는 시도가 있었다. 아래는 합성 과정을 도식화한 그림이다.

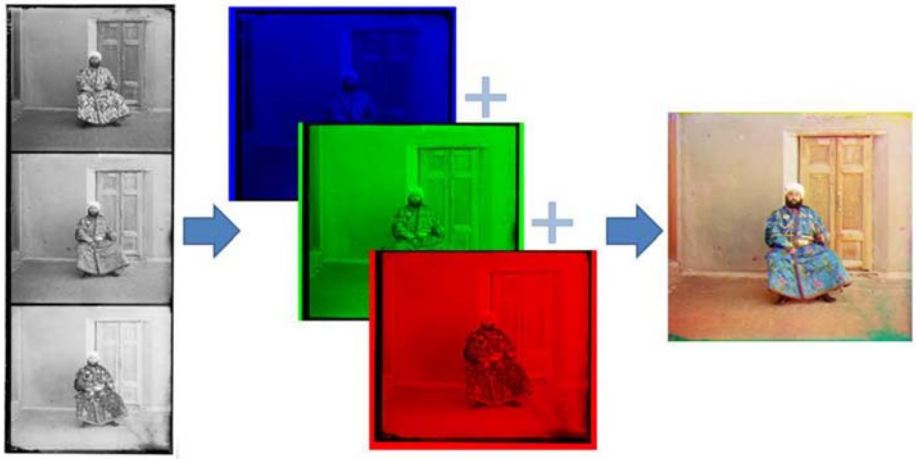


Fig1. Overview

그러나 여러 번의 촬영 중, 사진기가 흔들려, 각 채널의 이미지가 어긋나기 일쑤였기 때문에, 아래의 왼쪽 그림처럼 부정확한 결과를 얻는 경우가 많았다. 이 경우 이미지 채널 정합(4쪽 참고)을 통하여 이미지를 개선하게 되면 오른쪽 이미지를 얻을 수 있다.

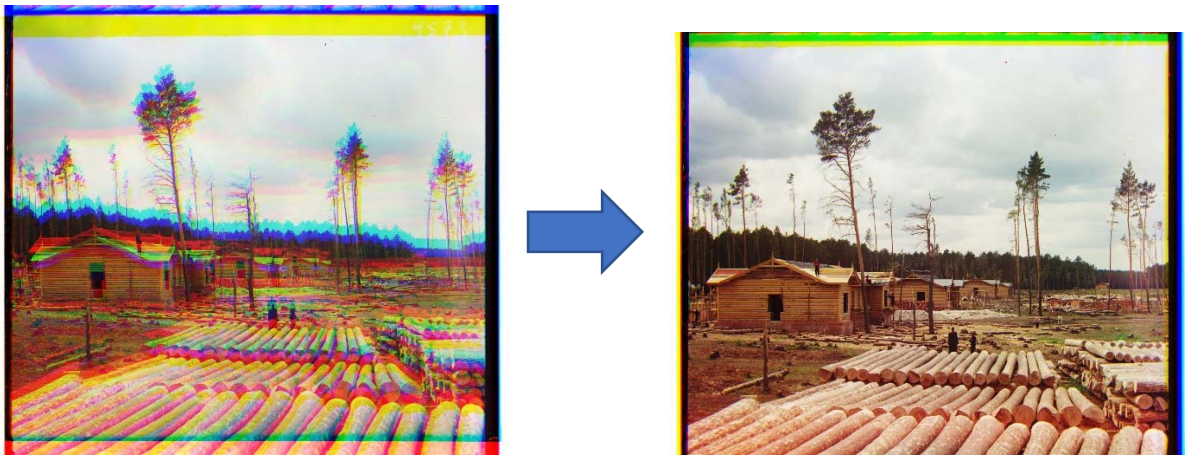


Fig2. Misaligned & aligned image

이번 과제에서는 어긋나 있는 빨간색, 녹색, 파란색 각 채널의 이미지를 읽은 후 정합을 수행하고, 겹치지 않는 테두리 영역을 잘라내어 알맞은 크기의 칼라 사진으로 저장하는 프로그램을 작성해 보자. 단, 제공되는 이미지는 3개의 채널 이미지가 주어지는 것이 아니라, 과제 구현의 편의를 위하여 Fig 2의 왼쪽 그림처럼 각각의 채널이 어긋나 있는 상태 그대로 합쳐진 이미지가 주어진다.

따라서, 프로그램 구현 시에 제공된 이미지 정합 전, 각 채널로 분리하는 작업이 필요하다.
제공되는 원본 이미지는 test1.ppm, test3.ppm, test5.ppm 이며, 이미지 정합 후 생성된 결과 파일도 제공된다.

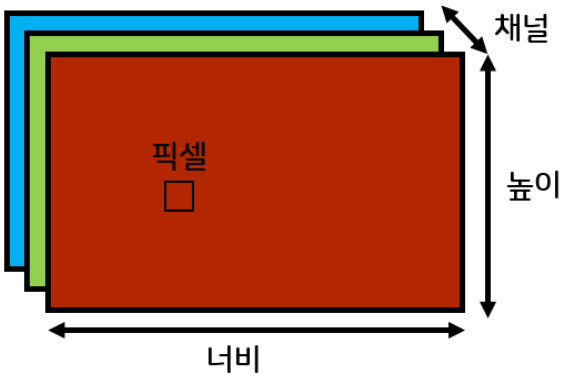
(목표)

이번 과제의 목표는 다음과 같다.

- 파일 입출력을 이용하여 파일을 읽어 들이고 저장할 수 있다.
- 동적할당을 통한 데이터 관리를 할 수 있다.
- 다차원 배열 간의 연산을 수행할 수 있다.

(이미지)

이미지는 3차원 구조로 만들어져 있다. 3차원 구조에서 각각의 축은 채널(C), 높이(H), 너비(W)로 표현된다. 만약, 채널이 3, 높이가 100, 너비가 200인 이미지가 있다고 했을 때, 이 이미지를 표현하기 위해 필요한 데이터의 개수는 총 3*100*200=60000개가 된다. 각각의 데이터는 0부터 255의 정수를 가지게 된다. 일반적으로 이미지는 빨간색, 녹색, 파란색을 이용하여 색을 표현하고 각각을 하나의 채널로 표현하기 때문에 칼라 이미지는 3개의 채널을 가지고 있다.



(ppm 파일 포맷)

이번 과제에서 사용할 이미지 파일 포맷은 "PPM" (Portable PixMap) 포맷이다. 첫 번째 줄은 이미지의 크기에 관련된 정보를 담고 있다.

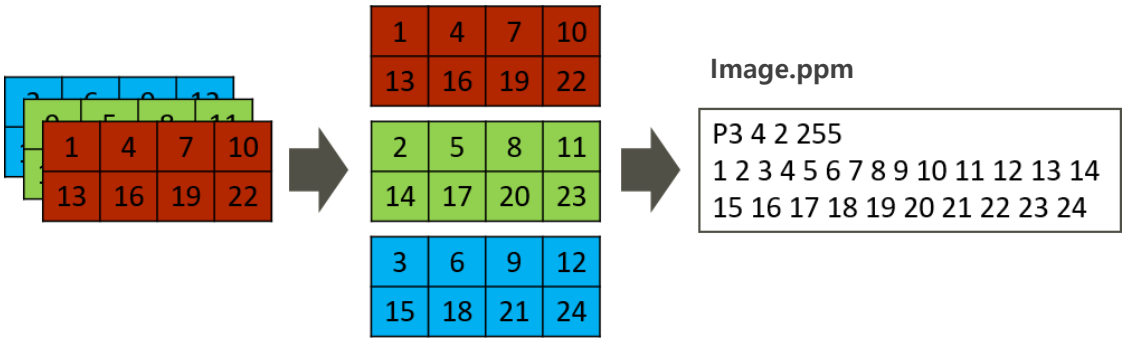
```
P3 600 370 255
```

P3은 사전에 약속된 기호로, 포맷의 타입 및 데이터 저장 형식을 나타낸다. 아래의 표는 각 기호에 대한 표이다.

Type	Magic Number		Extension	Colors
	ASCII	Binary		
Portable BitMap	P1	P4	.pbm	0-1 (white & black)
Portable GrayMap	P2	P5	.pgm	0-255 (gray scale), etc.
Portable PixMap	P3	P6	.ppm	0-255 * 3 (RGB channel)

600은 이미지의 너비가 600 픽셀이고, 370은 높이가 370 픽셀임을 의미한다. 255는 각각의 데이터가 0부터 255까지의 정수형을 표현할 수 있음을 말한다. 이 과제를 위해 제공되는 이미지와 정합을 통하여 생성할 이미지는 항상 P3으로 시작하며 255로 끝난다. 높이와 너비는 다를 수 있다.

두 번째 줄은 각각의 픽셀에 대한 데이터가 들어 있다. 나열되어 있는 방법은 채널, 너비, 높이의 순서로 나열되어 있다. 예를 들어, 높이가 2, 너비가 4, 채널이 3인 이미지의 경우 아래와 같은 파일 구조를 가진다.



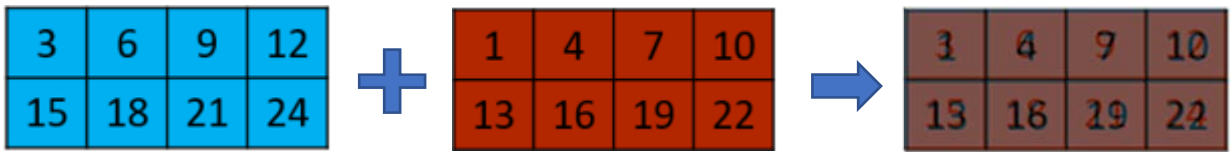
과제에서 요구하는 이미지 정합을 수행하기 위해서는 각 채널이 분리가 되어 있어야 하는데, ppm 파일은 R, G, B채널이 첫번째 픽셀부터 번갈아 차례로 적혀 있는 것을 확인할 수 있다. 이미지를 불러올 때 이에 주의하여 각 채널을 구분하여 읽도록 한다.

ppm 파일 포맷을 이미지로 열 수 있는 뷰어는 꿀뷰, Xnview 등이 있으며, 제공되는 ppm 파일은 ASCII code로 작성되었기 때문에, 텍스트 편집기로도 픽셀 값을 확인할 수 있다.
참고) 꿀뷰 다운로드: https://kr.bandisoft.com/honeyview/

(이미지 정합)

이미지 정합은 두 이미지의 대응되는 부분을 찾고 이동시켜 맞추는 작업으로 다음과 같은 순서로 이루어진다. 이 과제에서는 Blue 채널 이미지를 고정 한 후, Red, Green채널의 변위를 찾도록 하자.

Step 1. 먼저 B, R 채널을 겹쳐본다.



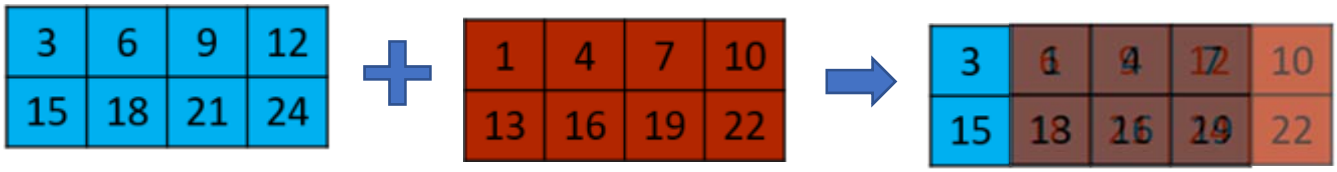
Step2. 겹치는 픽셀끼리의 연관성을 계산하여 이미지 전체가 비슷한 지 판단한다.

Ex) 픽셀 간 차이의 제곱합의 평균

$$Var: \frac{1}{8}((3-1)^2 + (6-4)^2 + \cdots + (24-22)^2)$$

Var 값이 클수록 차이가 많이 난다는 것이고, 정합이 되지 않았다는 것을 의미한다.

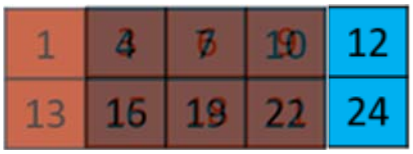
Step3. R 이미지를 1픽셀씩 이동시켜 겹쳐본다.



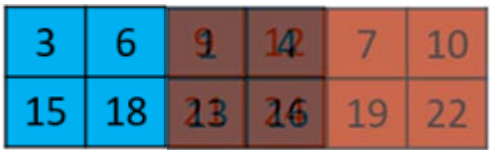
Step4. 겹치는 영역에서만 Step2를 수행한다.

$$Var : \frac{1}{6}((6-1)^2 + (9-4)^2 + \cdots + (24-19)^2)$$

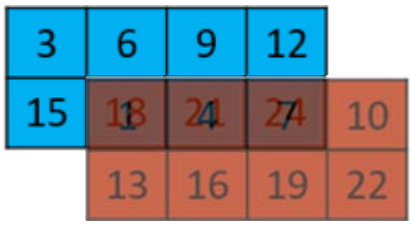
Step5. Step3~4를 반복하되 가로, 세로 방향으로 각각 -15 ~ 15의 값으로만 이동시킨다.



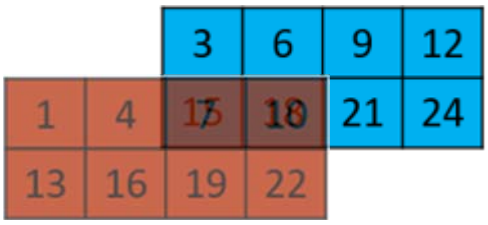
(가로 -1, 세로 0 움직인 경우)



(가로 2, 세로 0 움직인 경우)



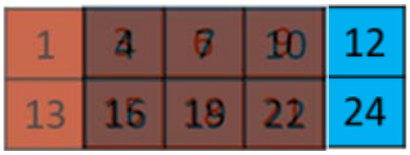
(가로 1, 세로 -1 움직인 경우)



(가로 -2, 세로 -1 움직인 경우)

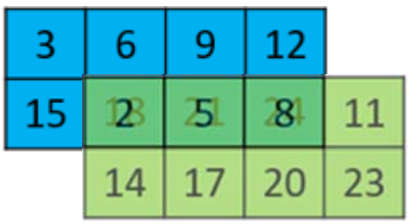
Step6. 모든 이동에 대한 Var 값이 구해지면, 가장 작은 Var값을 가진 이동을 기억한다.

이번 예시에서는 (가로 -1, 세로 0)이 선택되었다고 가정한다.



Step7. B, G에 대해서 1~6을 수행하여 이동 값을 구한다.

이번 예시에서는 (가로 1, 세로 -1)이 선택되었다고 가정한다.



Step8. B 채널을 기준으로 3개의 채널이 겹치는 영역에 대해 결과 이미지(.ppm)를 만든다.

위 설명에서는 두 이미지가 비슷한 정도를 판단하기 위해 픽셀간 차의 제곱합 평균을 사용했다. 아래에는 연관성을 계산하기 위한 2가지의 계산법을 소개한다.

(1) Sum of Squared Differences:

$$\mathrm{SSD}(I_1, I_2) = \frac{1}{\mathrm{XY}} \sum_{x,y} \left(I_1(x,y) - I_2(x,y)\right)^2$$

위 설명에서 사용된 식으로, 값이 클수록 차이가 크다는 뜻이기 때문에, 가장 작은 값을 선택해야 한다.

여기서, x, y는 픽셀의 위치를 의미하고, I_1, I_2는 매칭 정도를 파악하고자 하는 이미지 쌍이다. 즉 $I_1(x,y)$는 첫번째 입력 이미지의 가로 x번째, 세로 y번째인 픽셀의 값을 의미한다. 또한 X,Y는 각각 세로, 가로 길이를 의미한다.

(2) Normalized Cross Correlation:

$$\mathrm{NCC}(I_1, I_2) = \frac{\sum_{x,y} I_1(x,y) I_2(x,y)}{\sqrt{\sum_{x,y} I_1(x,y)^2 \sum_{x,y} I_2(x,y)^2}}$$

이 식을 직관적으로 이해하기위해, 평면상의 벡터 2개 $a(i,j), b(m,n)$을 생각해보자. 두 벡터의 내적을 길이의 곱으로 나누면 코사인 값을 구할 수 있다.

$$\frac{a \cdot b}{|a||b|} = \frac{i \cdot m + j \cdot n}{\sqrt{i^2 + j^2}\sqrt{m^2 + n^2}}$$

그리고 이 코사인 값은 두 벡터가 같은 방향을 보고있을 때 1까지 커지고 다른 방향을 볼 때 -1까지 작아진다. 다시 위의 식을 보면 형태가 같음을 알 수 있다. 즉 이미지를 픽셀 개수만큼 (XY) 의 차원을 가진 벡터라고 생각하고 XY차원 공간에서 비슷한 방향을 가지는가를 판단하여 유사도를 측정한다고 생각할 수 있다. 벡터의 코사인 값과 마찬가지로 유사도가 높으면 크기가 최대 1까지 커지고 반대의 경우에는 최소 -1까지 작아지므로 SSD와는 반대로 큰 값을 선택해야 한다.

NCC는 픽셀들의 전체적인 상관도를 계산하기 때문에, 비교하는 대상들의 값이 전체적으로 차이가 나는 경우에 SSD보다 강인하게 동작한다.

이번 과제에서는 주어진 이미지에서 SSD, NCC를 각각 이용하여 유사도를 계산해본다. 계산된 유사도는 위에서 설명된 대로 정합 알고리즘에 사용하도록 한다.

아래는 추가적인 제약사항이다.

- 계산량에 제한을 두기 위해 가로, 세로 이동거리는 -15 ~ 15으로 제한한다.
- 이동 시 겹치는 부분에 대해서만 식을 계산해야 한다.
- 주어진 예시 중 하나를 처리하는데, 30초 이상 소요되지 않아야 한다.

(주의 사항)

1. 표준 헤더 파일 <string.h>를 include 하여 사용할 수 있다.
2. 과제 작성시 전역변수는 사용할 수 없으며, 각 기능을 수행하는 함수를 별도로 구현할 것.
3. 연결리스트(linked list)와 같은 다른 방법을 사용할 경우, 감점 처리한다.
4. 문제 설명에서 동적으로 할당 받으라고 명시되어 있는 부분은 필히 동적으로 할당 받아 사용하며, 더 이상 사용되지 않을 경우 free 를 이용해 할당 해제할 것
5. 프로그램이 수행 될 때, 발생할 수 있는 예외 상황을 고려하여 구현한다. 단, 아래 명시하지 않은 예외 상황에 대해서는 고려할 필요가 없다.

(구현 기능 설명)

(1) 메뉴 화면

- 프로그램이 실행 되면 아래의 메뉴가 출력되며, 하나의 기능이 완료되면 메뉴는 다시 출력되도록 한다. (메뉴명의 띄어쓰기나 "="의 개수가 예시와 달라도 감점되지 않는다.)

```
====================
[1] 이미지 불러오기
[2] 이미지 정합(SSD)
[3] 이미지 정합(NCC)
[4] 종료
====================
메뉴 선택>
```

- 메뉴 출력 후, 사용자 입력을 받는다.
- 아래의 입력에 대해서 적절한 에러 문장과 함께 다시 메뉴를 출력 후, 입력을 받는다.
  1) 1~4외의 숫자 입력
  2) 이미지 불러오기 기능 수행 전, 2~3을 선택하여 이미지 정합을 수행하려고 하는 경우

(2) 이미지 불러오기

- 1을 입력하여 '이미지 불러오기'를 선택하면, 사용자로부터 이미지 파일 이름을 입력 받고, 이미지가 유효한 경우 이미지 파일로부터 읽어 들인다.
  (예시의 빨간색 밑줄은 사용자 입력에 해당한다.)

```
[4] 종료
====================
메뉴 선택> 1
이미지 이름: test5.ppm
이미지 읽기를 완료했습니다.
```

```
====================
[1] 이미지 변경 – 현재 이미지: test5.ppm
[2] 이미지 정합(SSD)
[3] 이미지 정합(NCC)
[4] 종료
====================
메뉴 선택>
```

- 이 때, 입력 파일 이름은 공백을 포함하지 않으며, 30자를 넘지 않는다고 간주한다.
- 존재하지 않는 파일을 입력 한 경우, 적절한 에러 처리를 하도록 한다.
- 이미지는 동적할당으로 필요한 이미지 크기만큼 할당 받도록 한다. 아래는 이미지 저장을 위한 2차원 배열 동적 할당 방법 예시이다.

```
// Red 채널 이미지 저장을 위한 동적 할당 (이미지 height와 width가 주어진 경우)
int **R;
int i;

R = (int **)malloc(sizeof(int *) * height);
for (i = 0; i < height; i++)
        *(R + i) = (int *)malloc(sizeof(int) * width);
```

- 이미 불러온 다른 이미지가 있는 경우 그 이미지를 삭제(동적 할당 해제) 후, 새로운 이미지를 저장한다.

(3) 이미지 정합(SSD)

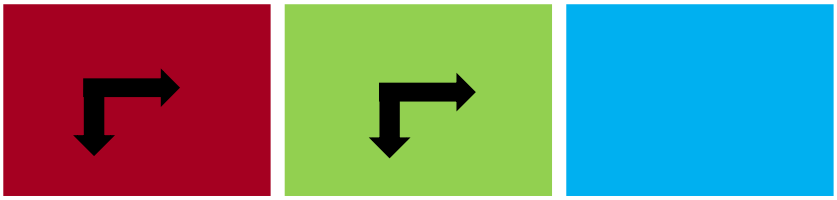
- 2를 선택하여 '이미지정합(SSD)'을 선택하면 SSD 수식을 이용하여 정합을 수행한 후, 그 결과를 출력 한다. 아래 출력 결과는 R : [1, -12] 즉 Red 채널의 이미지는 오른쪽으로 1픽셀만큼, 아래쪽으로 12만큼 움직이고 G : [1, -6] Green 채널의 이미지는 오른쪽 1픽셀, 아래쪽 6만큼 움직이는 것이 정합이 된다는 것을 의미한다.

```
====================
[1] 이미지 변경 – 현재 이미지: test5.ppm
[2] 이미지 정합(SSD)
[3] 이미지 정합(NCC)
[4] 종료
====================
메뉴 선택> 2

====================
SSD - R : [1, -12]   G : [1, -6]
결과 이미지 파일: test5_SSD_R1_-12_G1_-6.ppm
====================
```

- '이미지 정합' 설명의 Step8의 예시처럼, Blue를 기준으로 3개의 채널이 겹치는 영역에 대한 결과 이미지를 ppm 형식으로 저장한다. 즉, 겹치는 영역만큼의 크기를 가지게 된다.

- 이 때, 결과 이미지 파일 이름은 "*원본이미지이름_계산법_R%d_%d_G%d_%d.ppm*"으로 한다.



R : 가로 1 세로 -12	G : 가로 1 세로 -6	B : 고정

  - 결과 이미지 파일 이름: *원본이미지이름_SSD_R1_-12_G1_-6.ppm*
  - 원본 이미지의 가로길이: W, 세로길이: H
  - 결과 이미지의 가로길이: $W-1$, 세로길이: $H-12$

- 생성된 이미지를 뷰어로 보면 오른쪽 아래와 같다. 아래 왼쪽은 원본 이미지(test5.ppm)에 해당한다.

(4) 이미지 정합(NCC)

- 3을 선택하여 '이미지정합(NCC)'을 선택하면 NCC 수식을 이용하여 정합을 수행하고 아래와 같이 결과를 출력 한 후, 정합 결과에 따라 겹치는 영역에 대해서 이미지를 저장한다.
- 결과 이미지 파일 이름은 "*원본이미지이름_계산법_R%d_%d_G%d_%d.ppm*"를 적용해서 '*test5_NCC_R1_-12_G1_-5.ppm*' 이름을 가짐을 볼 수 있다.

```
====================
[1] 이미지 변경 – 현재 이미지: test5.ppm
[2] 이미지 정합(SSD)
[3] 이미지 정합(NCC)
[4] 종료
====================
메뉴 선택> 3

====================
NCC - R : [1, -12]   G : [1, -5]
결과 이미지 파일: test5_NCC_R1_-12_G1_-5.ppm
====================
```

- 생성된 이미지를 뷰어로 보면 오른쪽 아래와 같다.

(5) 종료

- 4를 입력하여 프로그램을 종료한다.
- 동적할당으로 받은 모든 메모리를 할당 해제하고 프로그램을 종료한다.