글. 조인준 KBS 기술연구소 차장

C군의 네버엔딩 스토리,
디지털 영상처리의 이해 - 4

앞선 [디지털 영상처리의 이해]의 1편부터 3편까지를 통해서 영상은 아주 작은 화소들로 이루어진 모자이크와 같은 것이라는 것을 알았고, 그 모자이크를 구성하는 최소 단위인 화소는 단색으로 이루어진 사각형과 동일하다는 것도 알 수 있었습니다. 화소가 밝기를 띄게 할 수 있는 기술은 있었으나 색을 띄게 할 수 있는 기술이 없던 시대에 각 화소는 밝기 정보만을 가졌고, 자연히 영상은 밝기 정보로만 표현이 가능한 흑백 영상을 사용했었습니다. 이후 화소가 색을 띄게 할 수 있는 기술이 개발되면서 컬러영상이 일반화되었는데, 컬러영상을 구현하는 방법은 3원색인 빨강(Red, 이하 R), 초록(Green, 이하 G), 파랑(Blue, 이하 B) 빛을 혼색하여 임의의 색상을 표현하는 것이었습니다.

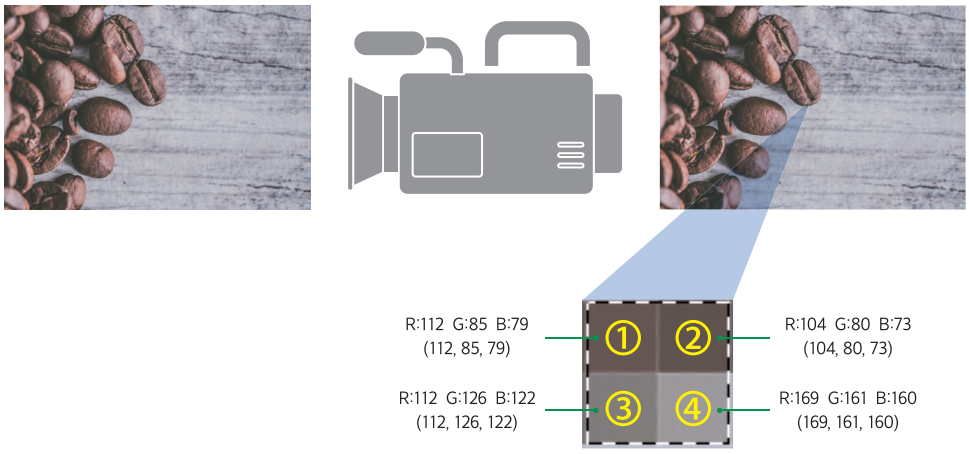

그림 1. 피사체와 카메라에 촬상된 피사체의 영상 데이터

지난 3편까지의 내용을 종합하면 [그림 1]과 같습니다. 카메라는 자연의 피사체로부터 반사되어 나온 빛을 이미지센서를 통해 데이터로 변환합니다. 카메라가 이미지센서를 통해 영상 데이터를 얻는 방법은 1) 분광기와 세장의 이미지센서를 사용해서 화소마다 피사체로부터 나온 빛과 같은 색을 낼 수 있는 3원색 데이터로 변환하는 방법, 2) 컬러 필터 어레이(CFA : Color Filter Array)와 한 장의 이미지센서를 사용하여 화소마다 피사체로부터 나온 빛과 같은 색을 낼 수 있는 3원색 중 하나만을 선택하여 데이터로 변환하고, 컬러 필터 사용으로 인해 잃어버린 나머지 2개 원색에 대한 데이터는 주변 화소의 데이터를 이용하여 보간(interpolation)하는 방법이었습니다. 세장의 이미지센서를 사용하는 경우나, 한 장의 이미지센서를 사용하는 경우 모두 카메라에서 얻은 영상은 화소마다 피사체에서 나온 빛과 같은 색의 3원색 데이터를 갖게 됩니다. 이 3원색 데이터와 우리가 눈으로 보는 가시광선의 색이 대응되는 원리는 다음과 같

습니다. 설명을 위해 8-bit 영상 데이터를 예로 들겠습니다. 8-bit 영상 데이터는 3원색에 해당하는 R(빨강), G(초록), B(파랑)를 나타내는 데이터를 각각 8-bit로 표현한 영상 데이터입니다. 8-bit를 사용하면 이진수로 00000000부터 11111111까지 256(2^8)개의 숫자를 표현할 수 있고 이를 10진수로 바꾸면 0에서 255 사이의 숫자가 됩니다. 그래서 8-bit 영상에서 화소는 R, G, B 각각 0~255의 값을 가질 수 있고, 255는 각 원색의 최대치, 0은 각 원색 성분이 존재하지 않는 것을 나타내며, 그 사이의 값들은 최대치인 255까지 선형적으로 각 원색성분이 증가하는 것을 나타냅니다. 이 R, G, B를 모두 최대치(255)로 혼합하면 백색, 모두 최소치(0)로 혼합하면 제일 어두운 검정이 되며, R, G, B를 혼합하는 비율에 따라 가시광선 영역의 다양한 빛의 색을 표현할 수 있습니다.

[그림 1]의 하단부에 가로, 세로 2×2로 확대된 화소들을 예로 들어 각 화소에서의 영상 데이터를 조금 더 부연하겠습니다. 2×2 영역의 ①번 화소의 경우 R = 112, G = 85, B = 79의 값을 갖습니다. G나 B에 비해 R값이 더 높습니다. 이는 갈색 계열의 색상으로부터 자연스럽게 예상되는 원색의 혼합비율입니다. ①번 화소 옆의 ②번 화소도 비슷한 양상을 보입니다. ②번 화소의 경우도 갈색 계열의 색상을 띄고 이에 맞게 R = 104, G = 80, B = 73으로 다른 원색에 비해 R값이 높게 나타나고 있습니다. 이와는 다르게 ④번 화소의 경우 회색을 띄고 있고 R = 169, G = 161, B = 160으로 R, G, B 값이 서로 엇비슷합니다. R, G, B 값이 서로 비슷하므로 어느 특정한 색상을 띄지 않는 회색계열이 되는 것이 직관적으로 쉽게 이해됩니다. 앞으로 화소의 색은 R, G, B 값으로 설명이 되므로 표기에 규칙을 적용하여 간단히 하겠습니다. 이후의 설명에서 각 화소의 R, G, B 값을 표시할 때 (R값, G값, B값)으로 표현하도록 하겠습니다. 예를 들면, [그림 1] 하단부 2×2로 확대된 화소 중 ①번 화소의 3원색 데이터 R = 112, G = 85, B = 79는 (112, 85, 79), ②번 화소의 3원색 데이터 R = 104, G = 80, B = 73은 (104, 80, 73)과 같이 표시하겠습니다.

(255, 255, 255)	(128, 128, 128)	(0, 0, 0)
(255, 0, 0)	(0, 255, 0)	(0, 0, 255)
(128, 0,0)	(0, 128,0)	(0, 0, 128)
(255, 255, 0)	(0, 255, 255)	(255, 0, 255)
(255, 128,0)	(0, 255, 128)	(255, 0, 128)
(128, 128, 0)	(0, 128, 128)	(128, 0, 128)
(255, 255, 128)	(128, 255, 255)	(255, 128, 255)

그림 2. R, G, B 3원색 값과 그에 해당하는 색

[그림 2]는 각 화소가 가지는 R, G, B 데이터와 화소의 색에 관한 감각의 구체화를 돕기 위해 화소의 색과 해당 (R값, G값, B값)을 표시한 차트입니다. R, G, B 값을 최대(255), 중간(128), 최소(0) 3단계로 변화시켜가며 혼색의 결과를 보여주고 있습니다. 위 차트의 R, G, B 혼합비에 따른 색의 변화를 보시면 3원색을 혼합하여 색을 구현하는 방법에 관한 이해가 조금 더 선명해질 것이라고 생각합니다.

카메라로 촬영한 영상의 데이터가 각 화소마다 (R값, G값, B값)이라면, 어떻게 이 값들이 반도체 메모리에 기록되어 영상기기 내부 연산을 거쳐 처리되고 최종적으로 저장장치에 파일로 저장되어 기기를 옮겨 다니며 재생되는 걸까요? 이를 설명하기 위해서 우선 영상 데이터가 기록되는 반도체 메모리의 구조를 개념적으로 알아보겠습니다. 반도체 메모리라는 말이 거창하고 뭔가 복잡하게 들릴지 모르지만, 추상적으로 보면 [그림 3]과 같이 바이트(byte, 8-bit과 동일) 단위로 1씩 증가하는 주소가 달린 1차원 배열입니다.

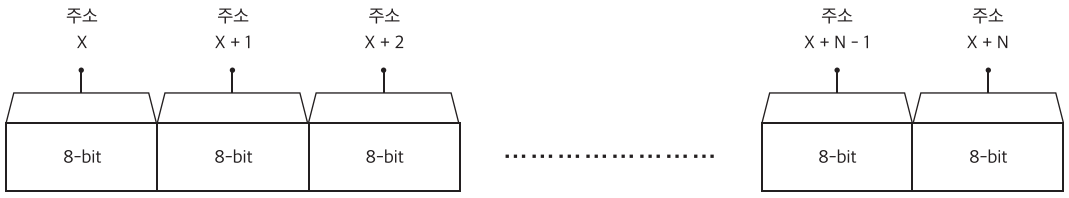

그림 3. 반도체 메모리의 추상적 구조

[그림 3]에서 나타낸 반도체 메모리의 1차원 배열에 화소의 8-bit R, G, B 값이 기록되는 방법을 [그림 4]에 나타내보았습니다. 영상 데이터가 기록되는 방법은 압축방식과 무압축 방식이 있지만, 우선 무압축 방식만을 예로 들어 설명하도록 하겠습니다. [그림 4] 좌측의 커피 원두 영상은 총 n개의 화소로 이루어진 8-bit 영상이라고 가정하겠습니다. 이 영상 첫 줄 좌측 첫 번째 화소의 데이터는 (R 1, G 1, B 1)이며, 첫 줄 k번째 화소의 데이터는 (R k, G k, B k), 그리고 마지막 줄 우측 마지막 화소의 데이터는 (R n, G n, B n)으로 표기하겠습니다. 반도체 메모리에 [그림 4] 좌측 커피 원도와 같은 영상이 무압축으로 기록되는 방법은 의외로 간단하게 영상의 맨 윗줄 좌측으로부터 우로, 그리고 한 줄이 끝날 때마다 다음 줄로 이동하며 [그림 4] 우측의 그림과 같이 순차적으로 해당 화소의 R, G, B 데이터를 반도체 메모리에 채워 나가는 것입니다. [그림 4]에서는 반도체 메모리에 영상 데이터를 기록할 때 윗줄의 좌측부터 우측, 그리고 한 줄이 끝나면 아랫줄로 이동하며 저장하는 방법을 적용하여 설명하였지만, 아랫줄부터 시작하여 윗줄로 올라가는 순서로 기록할 수도 있고, R, G, B 순서가 아닌 B, G, R 순서로 기록하는 등 여러 기록방법이 있습니다.

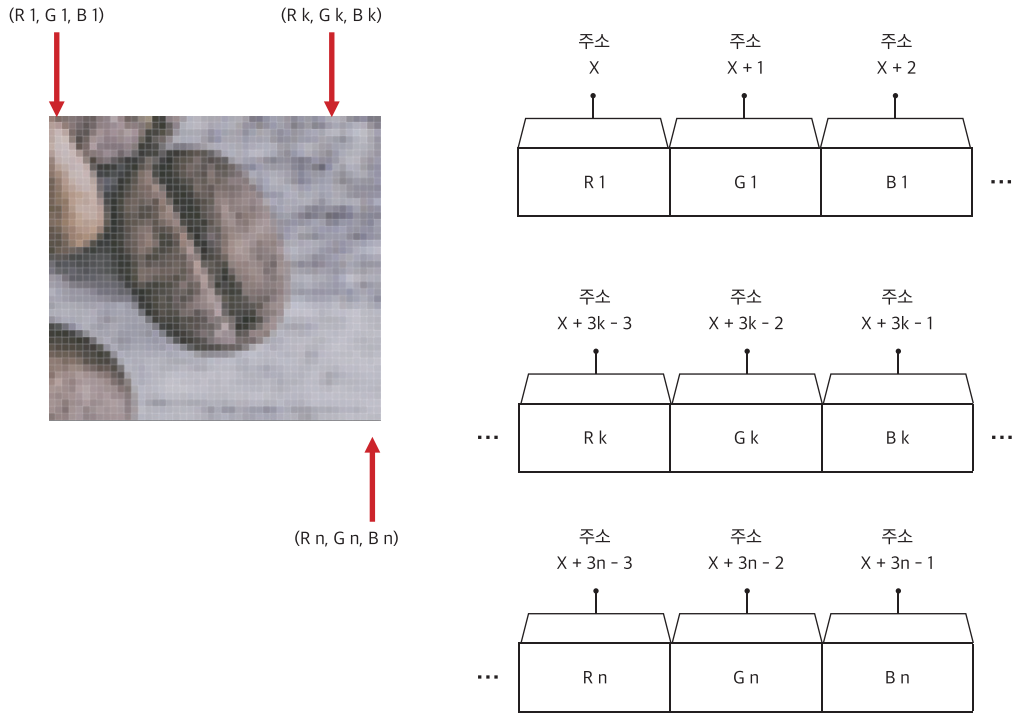

그림 4. 영상 데이터의 반도체 메모리 기록 예

그런데 [그림 4]에서와 같이 영상 데이터만 무압축으로 일렬로 나열한 것을 보고 기록된 영상의 가로 화소수가 얼마이고 세로 화소수가 얼마인지, 그리고 위에서 말했듯 여러 가지 기록방법 중에 윗줄에서 아랫줄 순서로 기록한 것인지, 아랫줄에서 윗줄 순서로 기록한 것인지, 화소의 데이터가 R, G, B 순서인지 B, G, R 순서인지 어떻게 알 수 있을까요? 이런 영상 데이터에 관련된 정보 없이 어떻게 다른 기기에서 영상을 재생할 수 있을까요? 이러한 문제를 해결하기 위해서 영상 데이터가 반도체 메모리로부터 저장장치의 파일로 저장될 때 헤더(Header)라는 것을 붙여서 저장하게 됩니다. 위에서 언급한 문제점들을 헤더가 어떻게 해결하는지 설명하기 위해 임의의 형태로 헤더를 [표 1]과 같이 정의해보겠습니다.

정보	자료량	값
가로 화소수	32비트	임의의 정수
세로 화소수	32비트	임의의 정수
기록 순서	1비트	0 : 윗줄부터 1 : 아랫줄부터
R, G, B 순서	1비트	0 : R, G, B 1 : B, G, R

표 1. 영상 데이터의 구조 정보를 담은 헤더 예

[그림 6]과 같은 헤더를 가지는 영상 파일이 있다면, 그 파일을 열어서 제일 처음 32-bit로 표현된 숫자를 읽으면 파일로 저장된 영상의 가로 화소수를 알 수 있습니다. 그리고 다음 32-bit로 표현된 숫자를 읽으면 세로 화소수를 알 수 있고, 다음 1-bit를 읽으면 파일의 영상 데이터가 기록된 방법(1-bit의 값이 0이면 윗줄부터 아랫줄로 내려오며 기록, 1이면 아랫줄부터 윗줄로 올라가며 기록), 그리고 헤더의 마지막 1-bit를 읽으면 영상 데이터의 R, G, B 값이 기록된 순서(1-bit 값이 0이면 R, G, B 순서, 1이면 B, G, R 순서)를 알 수 있습니다. 이렇게 헤더의 정보를 이용하여 [그림 6]과 같이 헤더 뒤에 붙어 있는 영상 데이터를 읽으면 저장된 영상을 정확히 재생할 수 있습니다. 그런데 헤더의 구조는 어떻게 알 수 있는지 다시 의문이 생길 것입니다. 헤더의 구조는 *.bmp, *.tga 등 파일 확장자를 보면 알 수 있습니다. 다시 말하면, 파일 확장자마다 헤더의 구조를 정의해 놓았습니다. 그래서 확장자를 알면 헤더의 구조를 알 수 있고, 헤더의 정보를 이용해서 실제 헤더 뒤의 데이터를 정확한 방법으로 읽어낼 수 있게 되는 것입니다.

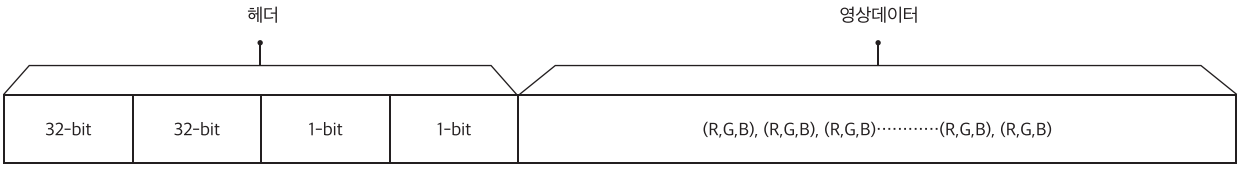

그림 5. 헤더와 영상 데이터가 결합된 영상 파일의 구조 예

지금까지 영상 데이터가 어떻게 표현되고, 메모리에 기록되고, 파일로 저장되는지 대략적으로 알아보았습니다. 다음 편에서는 영상 데이터에 관한 추가적 설명 및 영상 기기에서 어떻게 영상 데이터가 처리되어 재생되는지에 관해 설명하도록 하겠습니다.