컬러영역에서 관심영역추출

True color Img

- 크기를 구할 수 있어야함

RGB로 딸기영역 추출하기

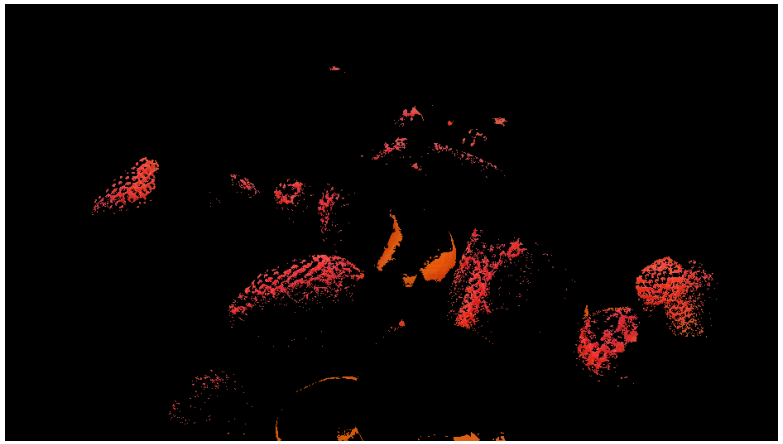
```
829     //딸기 추출하기
830     //BGR 순서인데 Red에 해당하는 값 즉 R이 200보다 큰 값만 내보냄 (마스킹함) -> 나머지는 가린다는 이야기
831 for (int i = 0; i < H; i++) {
832     for (int j = 0; j < W; j++) {
833         //단순하게 딸기니까 빨간색이 많겠지? 라고 생각
834         if (Image[i * W *3 + j*3+2]  > 200) {
835             Output[i * W * 3 + j * 3] = Image[i * W * 3 + j * 3];
836             Output[i * W * 3 + j * 3 + 1] = Image[i * W * 3 + j * 3 + 1];
837             Output[i * W * 3 + j * 3 + 2] = Image[i * W * 3 + j * 3 + 2];
838         }
839         else
840             //전부 0 => black
841             Output[i * W * 3 + j * 3] = Output[i * W * 3 + j * 3 + 1] = Output[i * W * 3 + j * 3 + 2] = 0;
842     }
843 }
844     SaveBMPFile(hf, hInfo, hRGB, Output, hInfo.biWidth, hInfo.biHeight, "output.bmp");
845
```

RGB에서 R이 높다는 건 빨간 값도 많다고 볼 순 있지만
그냥 밝기값이 높을때도 R이 크다

실패! 다른 것도 R이 큰 게 많아
마스킹을 패싱한 것도 많아

심지어 딸기는 다 표시되지도 않았어
기준점을 넘지 못함

```
829     //딸기 추출하기
830     //BGR 순서인데 Red에 해당하는 값 즉 R이 200보다 큰 값만 내보냄 (마스킹함) -> 나머지는 가린다는 이야기
831 for (int i = 0; i < H; i++) {
832     for (int j = 0; j < W; j++) {
833         //단순하게 딸기니까 빨간색이 많겠지? 라고 생각
834         //그럼 R은 밝기도 되니까 G,B도 조정해보자
835         if (Image[i * W *3 + j*3+2]  > 200 && Image[i * W * 3 + j * 3 + 0] < 100 && Image[i * W * 3 + j * 3 + 1] < 100)
836         {
837             Output[i * W * 3 + j * 3] = Image[i * W * 3 + j * 3];
838             Output[i * W * 3 + j * 3 + 1] = Image[i * W * 3 + j * 3 + 1];
839             Output[i * W * 3 + j * 3 + 2] = Image[i * W * 3 + j * 3 + 2];
840         }
841         else
842             //전부 0 => black
843             Output[i * W * 3 + j * 3] = Output[i * W * 3 + j * 3 + 1] = Output[i * W * 3 + j * 3 + 2] = 0;
844     }
845 }
```



그래도 딸기가 아닌 값은 많이 out 시킬 수 있었음

```
828
829     //딸기 추출하기
830     //BGR 순서인데 Red에 해당하는 값 즉 ROI 200보다 큰 값만 내보냄 (마스킹함) -> 나머지는 가린다는 이야기
831 for (int i = 0; i < H; i++) {
832     for (int j = 0; j < W; j++) {
833         //단순하게 딸기니까 빨간색이 많겠지? 라고 생각
834         //그럼 R은 밝기도 되니까 G,B도 조정해보자
835         //근데 글도 ROI 많이 높아서 다른 값을 조정해서 보면 좀 덜 보임 ==> 전통적인 방법 ; 조건을 변경하여 확인해봄
836         if (Image[i * W *3 + j*3+2]  > 130 && Image[i * W * 3 + j * 3 + 0] < 50 && Image[i * W * 3 + j * 3 + 1] < 100)
837         {
838             Output[i * W * 3 + j * 3] = Image[i * W * 3 + j * 3];
839             Output[i * W * 3 + j * 3 + 1] = Image[i * W * 3 + j * 3 + 1];
840             Output[i * W * 3 + j * 3 + 2] = Image[i * W * 3 + j * 3 + 2];
841         }
842         else
843             //전부 0 => black
844             Output[i * W * 3 + j * 3] = Output[i * W * 3 + j * 3 + 1] = Output[i * W * 3 + j * 3 + 2] = 0;
845     }
846 }
```

딸기 부분 좀 더 추가

Green색 감지를 젤 잘 하니까 사람 눈 센싱을 고려하여 Y에서 G를 높게 뽑음

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.16874 & -0.3313 & 0.500 \\ 0.500 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad \text{- (식 2-1)}$$

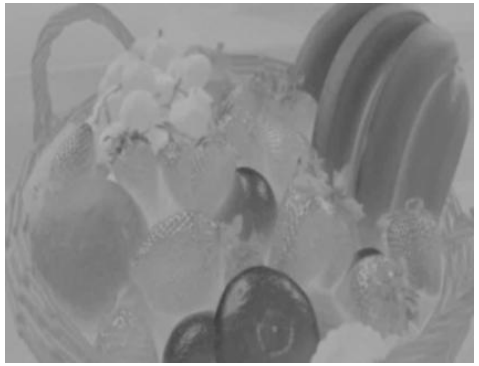
0~255값으로 맞추기 위해 Cb Cr은 +128 해주기

```
682 void RGB2YCbCr(BYTE* Image, BYTE* Y, BYTE* Cb, BYTE* Cr, int W, int H)
683 {
684     for (int i = 0; i < H; i++)
685     {
686         for (int j = 0; j < W; j++)
687         {
688             //GrayScale을 담고있던 y!
689             //이때 주의할 점은, 뒤에가 더블인데 BYTE에 집어넣어줘야해서 형변환을 (BYTE)로 해줘야해
690             Y[i * W + j] = (BYTE)(0.299 * Image[i * W * 3 + j * 3 + 2] + 0.587 * Image[i * W * 3 + j * 3 + 1] + 0.114 * Image[i * W * 3 + j * 3]);
691             Cb[i*W+j] = (BYTE)(-0.168 * Image[i * W * 3 + j * 3 + 2] + -0.3313 * Image[i * W * 3 + j * 3 + 1] + 0.5 * Image[i * W * 3 + j * 3] + 128.0);
692             Cr[i * W + j] = (BYTE)(0.5 * Image[i * W * 3 + j * 3 + 2] + -0.4187 * Image[i * W * 3 + j * 3 + 1] + -0.0813 * Image[i * W * 3 + j * 3] + 128.0);
693         }
694     }
695 }
```

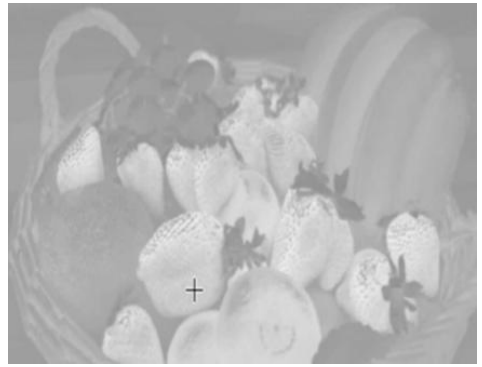
Y

```
818     BYTE* Y = (BYTE*)malloc(ImgSize);
819     BYTE* Cb = (BYTE*)malloc(ImgSize);
820     BYTE* Cr = (BYTE*)malloc(ImgSize);
821
822
823     RGB2YCbCr(Image, Y, Cb, Cr, W, H);
824
825
826     fp = fopen("Y.bmp", "wb");
827     fwrite(Y, sizeof(BYTE), W* H, fp);
828     fclose(fp);
829
830     fp = fopen("Cb.bmp", "wb");
831     fwrite(Cb, sizeof(BYTE), W* H, fp);
832     fclose(fp);
833
834     fp = fopen("Cr.bmp", "wb");
835     fwrite(Cr, sizeof(BYTE), W* H, fp);
836     fclose(fp);
```

Cb



Cr



- 문제 분석
- Cr을 스포이드로 뽑아보면
- Cr값 기준으로 딸기 레드값은 212
- 그런데 귤 값도 200정도가 넘음

- BUT
- Cb부분에서 딸기는 138정도고
- Cb부분에서 귤은 87정도임
- Cb부분에서 딸기는 130보다는 크고 Cr 성분은 200보다 크게 하면 되겠네

- 어두운 부분에선 딸기가 빛을 받으면 R 값도 줄어들어서 RGB 값으로 표현하기 어려워요

- 하지만 이렇게 해도 어려워
- 그래서 딥러닝을 많이 써
- 딸기에 대한 엄청난 이미지를 가지고 딥하게 학습하면 딸기라는 걸 학습 과정중 딸기의 특징또한 알아서 학습하게 돼
- Cb Cr을 우리가 정하고 있는 이 방식이 사실은 전통적인 거야!

```
879
880  BYTE* Y = (BYTE*)malloc(ImgSize);
881  BYTE* Cb = (BYTE*)malloc(ImgSize);
882  BYTE* Cr = (BYTE*)malloc(ImgSize);
883
884
885  RGB2YCbCr(Image, Y, Cb, Cr, W, H);
886
887
888      for (int i = 0; i < H; i++)
889      {
890          for (int j = 0; j < W; j++)
891          {
892              if (Cb[i * W + j] < 130 && Cr[i * W + j]>200)
893              {
894                  Output[i * W * 3 + j * 3] = Image[i * W * 3 + j * 3];
895                  Output[i * W * 3 + j * 3+1] = Image[i * W * 3 + j * 3+1];
896                  Output[i * W * 3 + j * 3+2] = Image[i * W * 3 + j * 3+2];
897              }
898              //블랙으로 보이도록
899              else
900              {
901                  Output[i * W * 3 + j * 3] = Output[i * W * 3 + j * 3 + 1] = Output[i * W * 3 + j * 3+2] = 0;
902              }
903          }
904      }
905
```

과제 : 사람 얼굴

- 얼굴 영역 검출하여 바운딩박스하여 이미지 출력하기