컴퓨터 비전 과제 1

정보통신공학과
12171786 박용민

과제 :

1. Gaussian과 Laplacian pyramid를 만들고 각각 잘 작동했다는 테스트 이미지를 보여라.
2. 1번에서 만든 pyramid를 조합하여 hybrid image를 만들어라.

언어 :

- C++ (opencv lib에서 I/O function만 가져왔다)

본문 :

1) 전체적인 코드 설명 (주석이 있지만 대략적인 부가 설명)

```
12  int N1 = 2; // 전역변수 N1
13  int N2 = 5; // 전역변수 N2
14
```

과제 2번 설명에 나왔던 N1과 N2이다. 해당 프로젝트를 진행할 때는 매뉴얼하게 가장 좋은 결과를 가져오는 숫자로 바꿨다고 하는데 2와 5로 했다. 프로젝트 설명에서 1번 사진의 Laplacian Pyramid에서 앞에서 N1개의 영상, 2번 사진의 Laplacian Pyramid에서 마지막 N2개의 영상과 2번 사진의 Gaussian Pyramid의 마지막 영상으로 Hybrid Image를 만들었다고 설명한다. 매뉴얼하게 쉽게 바꿀 수 있도록 전역변수로 선언했다.

```
int conv3x3_Func(uchar* arr, int a, int b, int width, int height) { // 3x3 컨볼루션 함수
    //a, b : rows, columns
    int kernel[3][3] = { 1,2,1,
                         2,4,2,
                         1,2,1 };    // 3x3 마스크 값 넣어주기

    int sum = 0;
    int sumKernel = 0;

    for (int j = -1; j < 2; j++) {
        for (int i = -1; i < 2; i++) { // 2중 for문
            if ((b + j) >= 0 && (b + j) < height && (a + i) >= 0 && (a + i) < width) { // i,j번째 픽셀이 영상의 총 크기 내부에 있을 때
                sum += arr[(b + j) * width + (a + i)] * kernel[i + 1][j + 1];          // 컨볼루션 연산 진행
                sumKernel += kernel[i + 1][j + 1];
            }
        }
    }

    if (sumKernel == 0) { return sum; }
    else { return (sum / sumKernel); }
}
```

많이 쓰이는 3x3 컨볼루션 연산을 수행하는 함수이다. 2중 for문으로 픽셀들의 컨볼루션 연산을 수행하도록 했고 강의 노트에 있던

1	2	1
2	4	2
1	2	1

이 커널을 사용했다.

```
Mat GaussianFilter_Func(Mat Image) { // gaussian filter를 수행하는 함수

    int width = Image.cols;
    int height = Image.rows;

    Mat Final_Img(Image.size(), CV_8UC1);

    uchar* Prev_Data = Image.data;
    uchar* Final_Data = Final_Img.data;

    for (int j = 0; j < height; j++) {
        for (int i = 0; i < width; i++) { // 2중 for문
            Final_Data[j * width + i] = conv3x3_Func(Prev_Data, i, j, width, height); // 3x3 컨볼루션 진행해주기
        }
    }

    return Final_Img;
```

컨볼루션 함수를 이용해 가우시안 필터를 적용해주는 함수이다. 직접적으로 영상을 입력으로 받아와서 폭과 높이 값을 받고 그 값을 이용해 2중 for 문으로 연산을 돌려준다.

```
Mat Sampling_Func(Mat Image) { // 샘플링 함수

    int width = Image.cols / 2; // 폭 절반으로 줄여주기
    int height = Image.rows / 2; // 높이 절반으로 줄여주기

    Mat dstImg(height, width, CV_8UC1);

    uchar* srcData = Image.data;
    uchar* dstData = dstImg.data;

    for (int j = 0; j < height; j++) {
        for (int i = 0; i < width; i++) {
            dstData[j * width + i] = srcData[(j * 2) * (Image.cols) + (i * 2)]; // 값 한칸씩 띄워서 넣어주기
        }
    }

    return dstImg;
```

해당 프로젝트에서 필터만큼 많이 쓰이는 샘플링 함수이다. 폭과 높이를 절반으로 줄여주고 이 또한 2중 for문으로 영상의 값들의 개수를 절반으로 줄여준다.

```
vector<Mat> GaussianPyramid_Func(Mat Image) { // Gaussian Pyramid를 만들어주는 Function

    vector<Mat> Gaussian_Vector; // 마지막에 return해줄 모든 영상을 저장할 벡터 매트릭스 선언

    Gaussian_Vector.push_back(Image);

    for (int i = 0; i < 8; i++) { // 8 계층
        Image = Sampling_Func(Image); // 영상 샘플링
        Image = GaussianFilter_Func(Image); // Gaussian filter 적용

        Gaussian_Vector.push_back(Image);
    }
    return Gaussian_Vector;
}
```

가우시안 필터를 이용해 가우시안 피라미드를 만들어주는 함수이다.
피라미드들을 저장해서 return 해줄 컨테이너를 선언하고 8개의 샘플링 및 필터가 적용된 영상들을 push해준다.

```cpp
vector<Mat> LaplacianPyramid_Func(Mat Image) { // Laplacian Pyramid를 만들어주는 Function

    vector<Mat> Laplacian_Vector; // 마지막에 return해줄 모든 영상을 저장할 벡터 매트릭스 선언

    for (int i = 0; i < 8; i++) { // 8 계층
        if (i < 7) {
            Mat Prev_Img = Image; // Filter 적용 이전 영상 백업

            Image = Sampling_Func(Image); // 영상 샘플링
            Image = GaussianFilter_Func(Image); // Gaussian filter 적용

            Mat Next_Img = Image; // filter를 적용한 다음 영상

            resize(Next_Img, Next_Img, Prev_Img.size()); // 다시 이전 영상 사이즈로 복원

            Laplacian_Vector.push_back(Prev_Img - Next_Img + 128); // 빼서 Laplacian 적용
        }
        else {
            Laplacian_Vector.push_back(Image);
        }
    }
    return Laplacian_Vector;
}
```

라플라시안 피라미드를 만들어주는 함수이다. 가우시안과 비슷한데, 컨테이너를 만들고 7번째까지 "이전 영상 백업 -> 가우시안 적용 -> 적용한 영상 다시 이전 영상 크기로 복원 -> 차영상 계산 후 컨테이너에 push"를 반복한다. 마지막 8번째는 가우시안 그대로 push한다.

```cpp
Mat Make_Hybrid_Func(vector<Mat> Hybrid_Material) { // 재료들로 하이브리드 이미지를 만들어주는 함수

    Mat Hyb_I; // 최종적으로 만들어진 hybrid image를 담을 매트릭스 선언

    for (int i = 0; i < Hybrid_Material.size(); i++) {
        if (i == 0) { // 첫번째 Gaussian 이미지는 그대로 불러오기
            Hyb_I = Hybrid_Material[i];
        }
        else {
            resize(Hyb_I, Hyb_I, Hybrid_Material[i].size()); // 그 다음 재료의 사이즈로 조절
            Hyb_I = Hyb_I + Hybrid_Material[i] - 128; // 이후 더해주기 (over-flow 방지를 위해 128 빼주기)
        }
    }

    return Hyb_I;
}
```

하이브리드 이미지(2번)의 재료들을 모두 더해 하이브리드 이미지를 만들어주는 함수이다. 특별한 것은 없고 모두 더해주고 오버플로우 방지를 위해 128을 빼주면 된다.

```
131 int main() {
132
133     // 1번
134
135     Mat Img1, Img2; // 1,2번 영상 매트릭스 선언
136
137     Img1 = imread("cat.jpg", 0); // 1번 영상 불러오기
138     Img2 = imread("dog.jpg", 0); // 2번 영상 불러오기
139
140     vector<Mat> Gaussian_Pyramid_Vector_1 = GaussianPyramid_Func(Img1);
141     vector<Mat> Laplacian_Pyramid_Vector_1 = LaplacianPyramid_Func(Img1);
142
143     vector<Mat> Gaussian_Pyramid_Vector_2 = GaussianPyramid_Func(Img2);
144     vector<Mat> Laplacian_Pyramid_Vector_2 = LaplacianPyramid_Func(Img2);
145
146     for (int i = 0; i < Gaussian_Pyramid_Vector_1.size(); i++) {
147         imshow("Gaussian pyramid", Gaussian_Pyramid_Vector_1[i]);
148         waitKey(0);
149     }
150
151     for (int i = 0; i < Laplacian_Pyramid_Vector_1.size(); i++) {
152         imshow("Laplacian pyramid", Laplacian_Pyramid_Vector_1[i]);
153         waitKey(0);
154     }
```

이제 메인함수이다. 먼저 과제 1번을 수행하는 부분을 살펴보면, 고양이와 강아지의 영상을 불러온다. 이후에 각각 두가지 피라미드를 함수로 만들어서 각각의 컨테이너에 저장한다. 과제 1번의 결과를 보기위해 1번 이미지만 imshow() 함수로 확인한다.

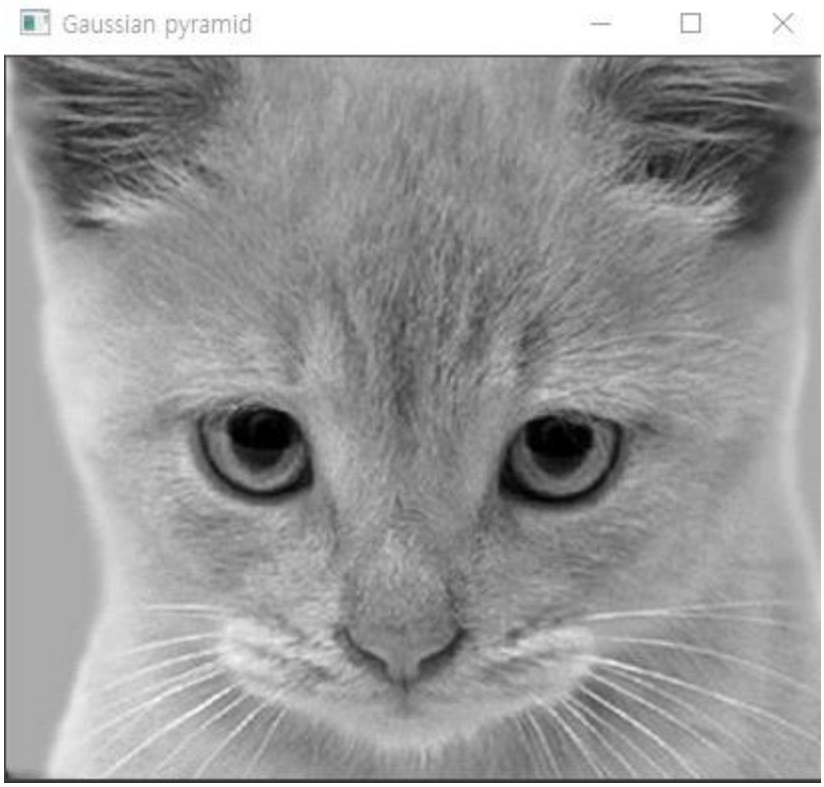
```
156     // 2번
157
158     vector<Mat> Hybrid_Material; // 하이브리드 이미지의 재료들이 담길 벡터 매트릭스
159     Mat Hyb_1; // 하이브리드 이미지 매트릭스 선언
160
161     for (int i = 0; i < 8; i++) {
162         if (i >= 0 && i <= N1) { // N1개의 Laplacian image
163             Hybrid_Material.push_back(Laplacian_Pyramid_Vector_1[i]);
164         }
165         else if (i >= (8 - N2)) { // N2개의 Laplacian image + Last Gaussian image
166             Hybrid_Material.push_back(Laplacian_Pyramid_Vector_2[i]);
167         }
168     }
169
170     reverse(Hybrid_Material.begin(), Hybrid_Material.end());
171
172     Hyb_1 = Make_Hybrid_Func(Hybrid_Material);
173
174     imwrite("Hybrid.jpg", Hyb_1);
175     imshow("Hybrid", Hyb_1);
176     waitKey(0);
177 }
```

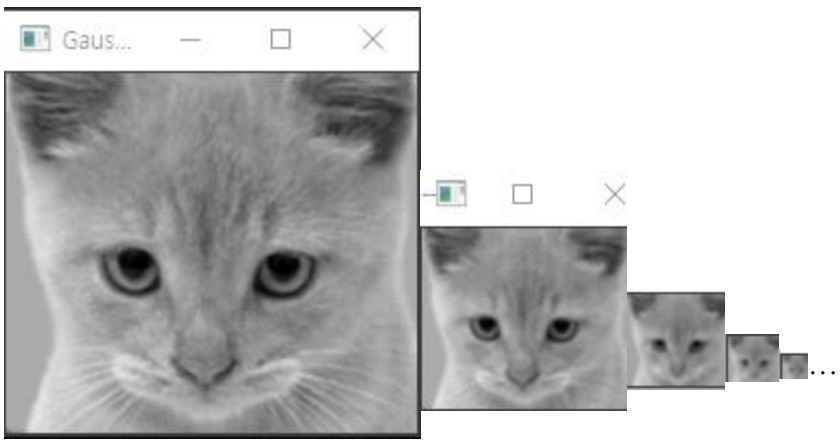
다음으로 과제 2번을 수행하는 부분을 살펴보면, 먼저 하이브리드 이미지가 될 재료들을 모두 담을 벡터 매트릭스를 선언해주고 거기에 고양이와 강아지의 라플라시안 영상들을 N1과 N2 개수에 맞게 담는다. 이후 그 재료들을 하이브리드 이미지를 만드는 함수에 넣어 이미지를 출력한다.

2) 결과

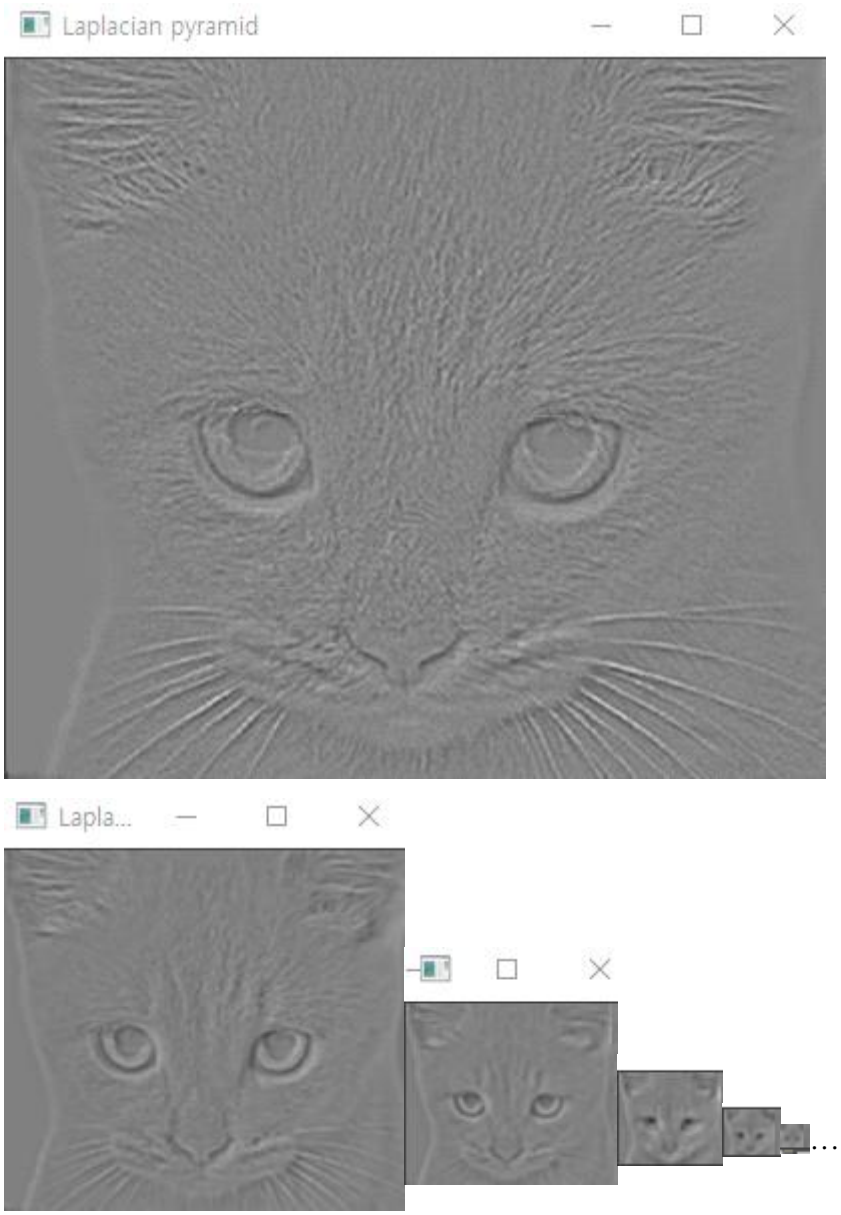
-1번 문제의 결과 영상들

고양이 - Gaussian

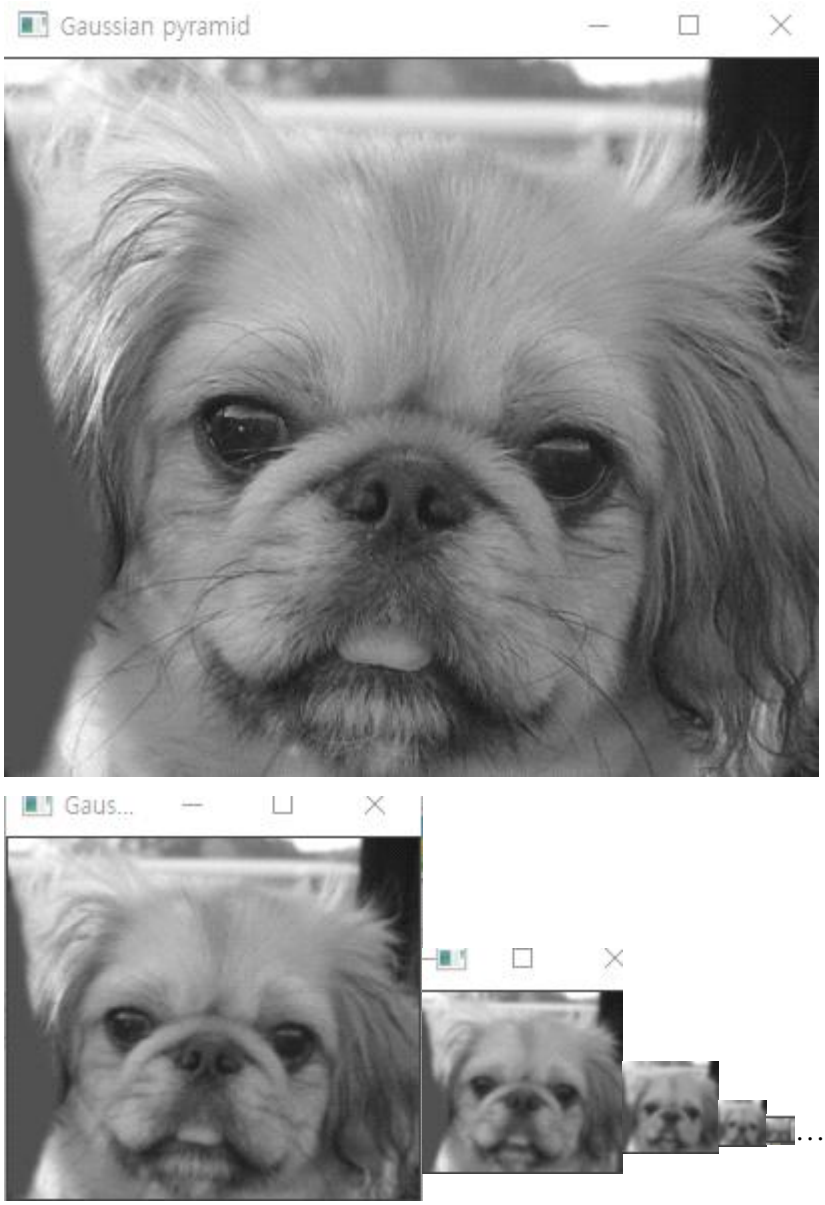




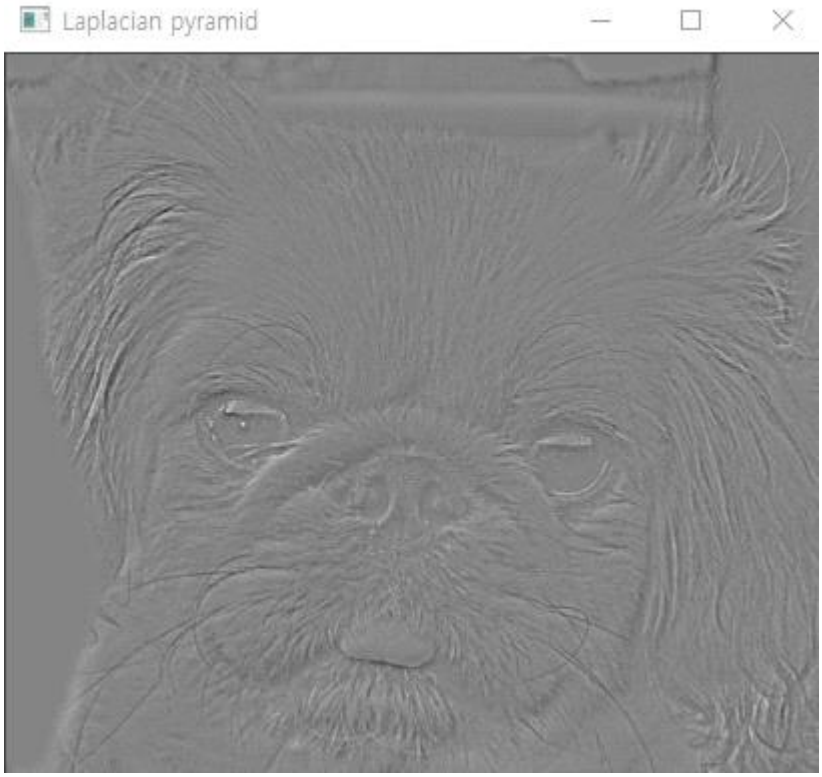
고양이 Laplacian



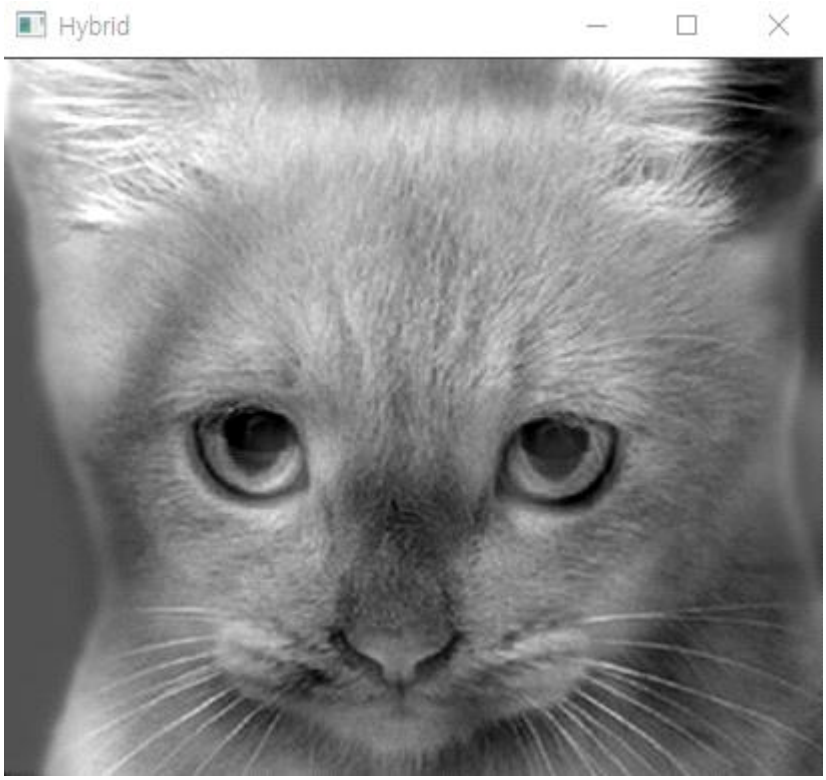
강아지 Gaussian



강아지 Laplacian



-2번 문제의 결과 영상들



강아지와 고양이의 Hybrid Image

open eye man과 closed eye man의 Hybrid Image