

SIFT 기반 이미지 매칭

[가우시안 필터링, 특징점 매칭]

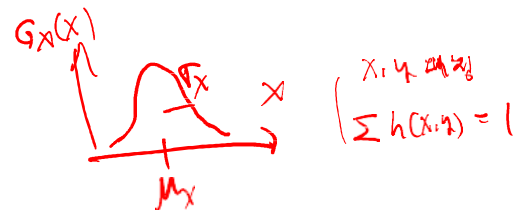
고급소프트웨어실습1 (3주차 실습)

실습 내용

$\frac{1}{273} \times$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2 + (y-\mu_y)^2}{2\sigma^2}}$$



- Scale space 구성

- Scale space 구성을 위해 사용되는 cvSmooth 함수와 같은 기능을 수행하는 사용자 정의 함수를 구현, 위의 간략화된 2D 가우시안 공식을 사용하여 가우시안 커널을 구성한다
- Lab2에서 구현한 convolution 연산 함수를 최대한 재사용하여 Gaussian smoothing 함수 구성

- SIFT를 이용한 이미지 매칭

- 프로그램에 주어진 두 개의 함수는 각각 Image1-Image2, Image1-Image3 간에 SIFT를 이용한 특징점 매칭을 통해 이미지 간의 유사성을 측정한다
- 특징점 간의 유사성을 계산하고 이를 통해 이미지 간의 유사성 비교

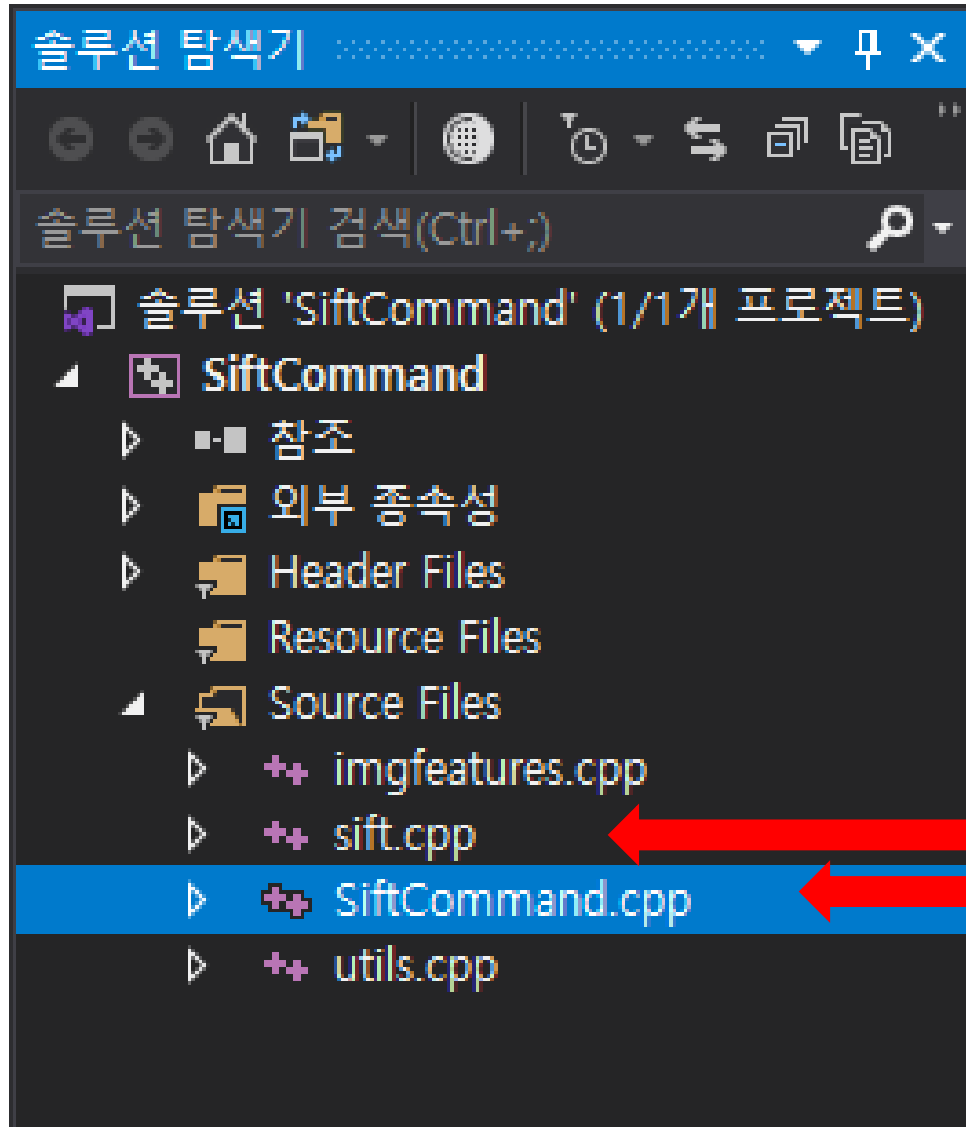
실습 방식

- 사이버캠퍼스->과제->3주차 실습에서 첨부파일을 다운로드하고, SiftTest.zip의 압축을 풀고 해당 프로젝트가 제대로 실행되는지 확인한다

(Visual Studio 2019에서 opencv 설정 후 프로젝트 실행, 실행되지 않으면 공지사항의 Using_Opencv.pdf를 참고하여 설정)

- 사이버캠퍼스 3주차 강의 내용을 바탕으로 SiftCommand.cpp의 PROBLEM 1.1 , PROBLEM 1.2, PROBLEM 2 부분을 작성한다.
- 실습을 다하면, 소회의실에 참석하여 조교에게 본인 실습 파일 실행 화면과 코드 (zoom으로 화면 공유)를 공유 한다. 조교 확인 후 실습 파일을 제출한다.
- 사이버캠퍼스->과제->3주차 실습에 작성한 코드를 첨부파일로 제출한다.

* 제출 방식 준수



Sift.cpp와 SiftCommand.cpp 수정

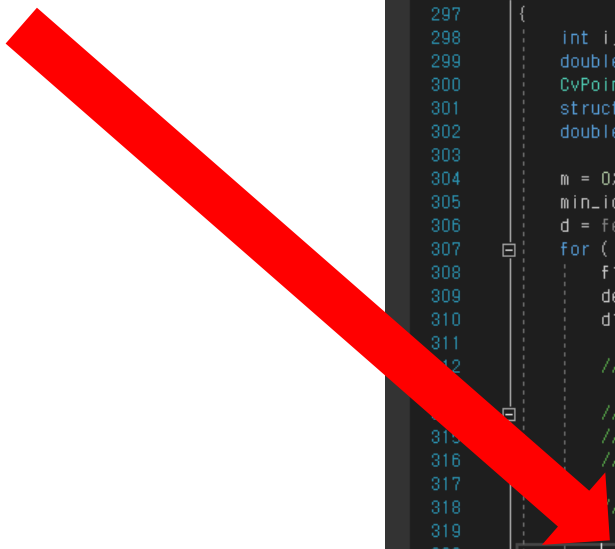
Sift.cpp

PROBLEM 1.1 (line 349)

PROBLEM 1.2 (line 400)

SiftCommand.cpp

PROBLEM 2.1 (line 277)



```
SiftCommand.cpp
SiftTest
(전역 범위)

286 //////////////////////////////////////////////////
287
288
289
290     imageBitsFiltered[r + m_width + c] = (int)mean;;
291 }
292 }
293 cvSetData(dst, imageBitsFiltered, m_width * channels);
294 }
295
296 int MatchAndDraw(struct feature* feat1, struct feature* feat2, int n1, int n2, IplImage* stacked, int width)
297 {
298     int i, j, k, d, m, min_idx;
299     double d0, d1, d2;
300     CvPoint pt1, pt2;
301     struct feature* f1, * f2;
302     double* descr1, * descr2, dist;
303
304     m = 0;
305     min_idx = 0;
306     d = feat1->d; // dimension of the descriptor (fixed to 128)
307     for (i = 0; i < n1; i++) {
308         f1 = feat1 + i;
309         descr1 = f1->descr;
310         d1 = d2 = DBL_MAX;
311
312         //////////////////////////////////////////////////
313
314         // PROBLEM 2 : Calculate the similarity between the descriptors of the image SIFT feature.
315         // Using L2 Distance, the descriptor in f1 can be accessed by f1->descr.
316         // The descriptor is a 1D array with 128 elements.
317
318         //////////////////////////////////////////////////
319
320
321
322         if (d2 < DBL_MAX && d1 < d2 + NN_SQ_DIST_RATIO_THR) {
323             f2 = feat2 + min_idx;
324             pt1 = cvPoint(cvRound(f1->x), cvRound(f1->y));
325             pt2 = cvPoint(cvRound(f2->x), cvRound(f2->y));
326             pt2.x += width;
327             cvLine(stacked, pt1, pt2, cvScalar(255, 0, 255), 1, 8, 0); // draw a line
328             m++;
329         }
330     }
331
332     return m;
333 }
334 }
```

문제	중요 입력 변수	출력 변수
1.1	w0, sigma1	kernel
1.2	src	dst
2	MatchAndDraw() 매개변수와 내부 선언된 변수	min_idx,d1,d2

Scale Invariant Feature Transformation (SIFT)

- SIFT 처리 기법은 이미지에서 회전, 크기 등의 변화에 영향을 받지 않는 특징점(keypoint)을 추출하고 그로부터 이미지의 변화에 강인한 디스크립터(descriptor)를 생성함
- 특징점과 디스크립터를 얻은 후에는 각 이미지 별로 유사한 특징점의 개수를 세어 이미지의 유사성을 판단

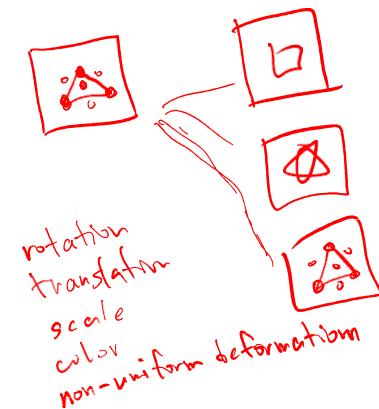
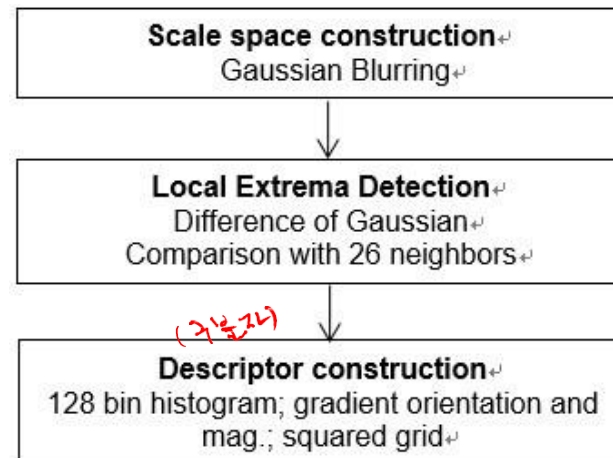


그림 1. SIFT 알고리즘의 도식적 표현

Scale space construction

- scale 이란 이미지 또는 그 안에 나타나는 객체의 크기
- SIFT 알고리즘에서는 입력된 이미지를 인위적으로 여러 개의 scale로 변환하고 그 안에서 관찰되는 특징점을 찾아냄으로써 크기 변화 문제 해결
- 입력된 이미지로부터 여러 개의 스케일로 표현되는 이미지를 생성하기 위하여 직접적으로 이미지의 사이즈를 줄이기 보다는 Gaussian blurring을 통해 scale 변화 효과를 달성
- scale 공간의 모사에는 한계가 있으므로 인위적인 이미지의 크기 변환도 병행하여 처리되며, 이는 Octave라고 표현함

Scale space construction

- 각 이미지 당 여러 개의 Octave를 구성하고 각 Octave 안에 여러 개의 scale 공간을 형성

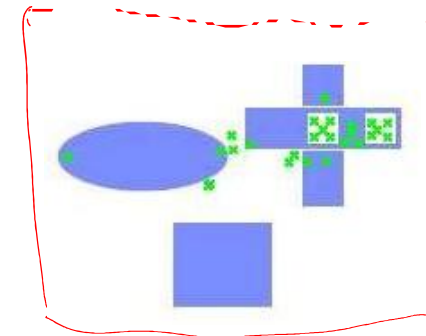
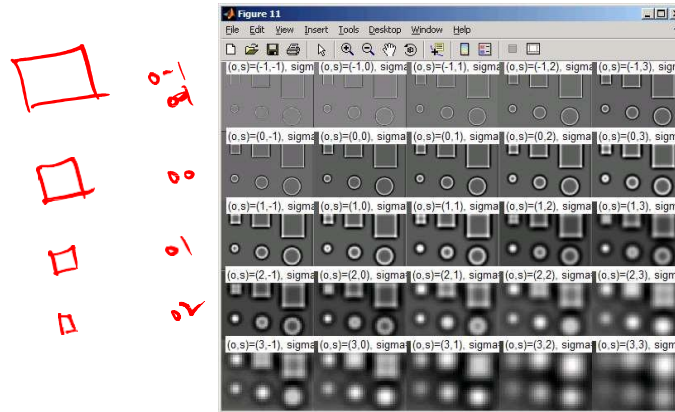
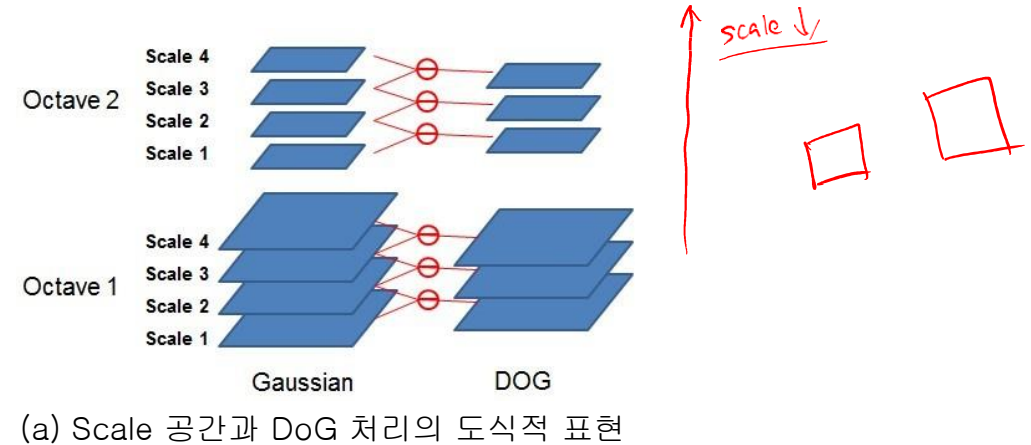
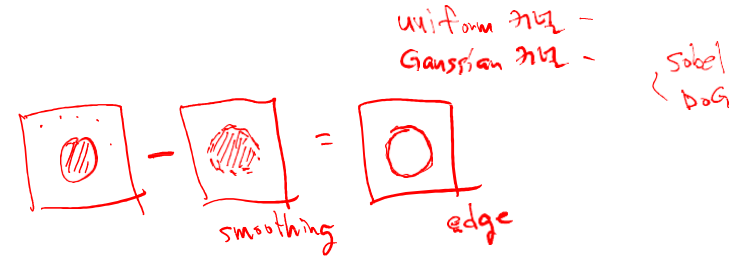


그림 2. Scale 공간의 도식적 표현과 SIFT 특징점의 예

Difference of Gaussian



- DoG는 보통 에지 검출용으로 쓰이던 방법이며, SIFT에서는 scale 공간에서 픽셀 값의 변화량을 찾기 위해 사용됨
- DoG에서 강한 반응을 보이는 픽셀은 Scale 공간에서 변화량이 큰 값

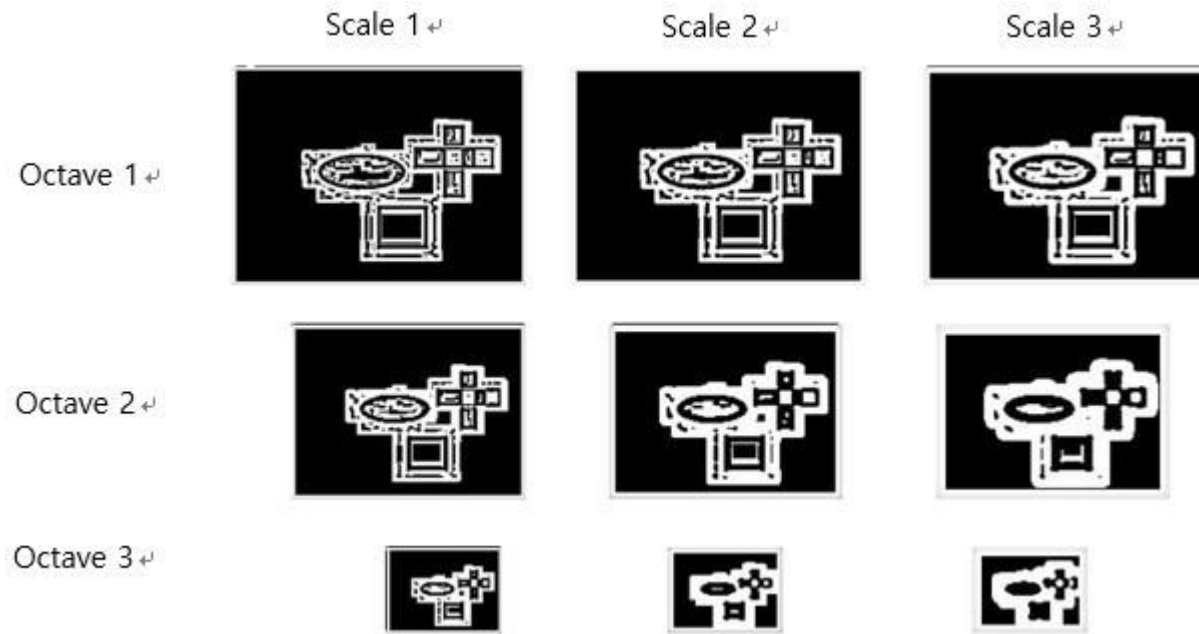


그림 3. 한 이미지를 Difference of Gaussian으로 표현한 예

Local Extrema Detection

- DoG로부터 특징점(Local Extrema, keypoint)을 찾는 방법은 SIFT에서 약간 독특하게 제안됨
- 스케일 공간에서 각 픽셀을 중심으로 주변 26개 픽셀과 비교하여 항상 크거나(최대) 항상 작으면(최소) 극값으로 선택
- 이렇게 얻어진 특징점들은 이후 몇가지 후처리 과정을 거치면서 이미지의 변화에 좀 더 강한 것들만을 추려냄

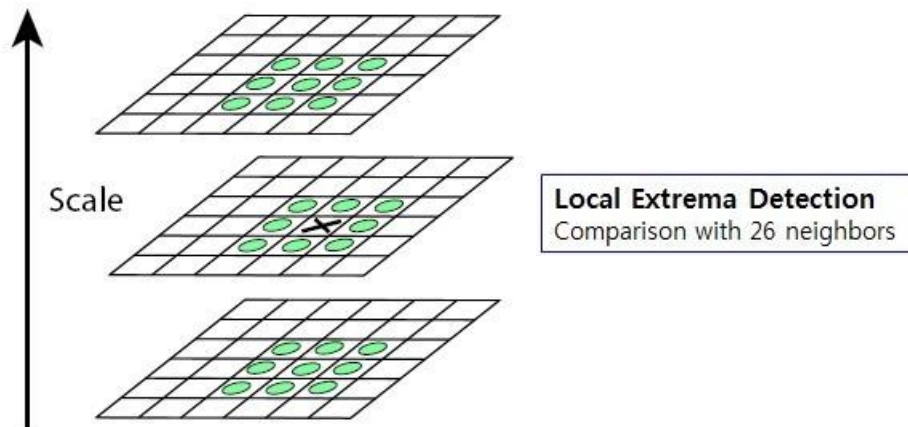
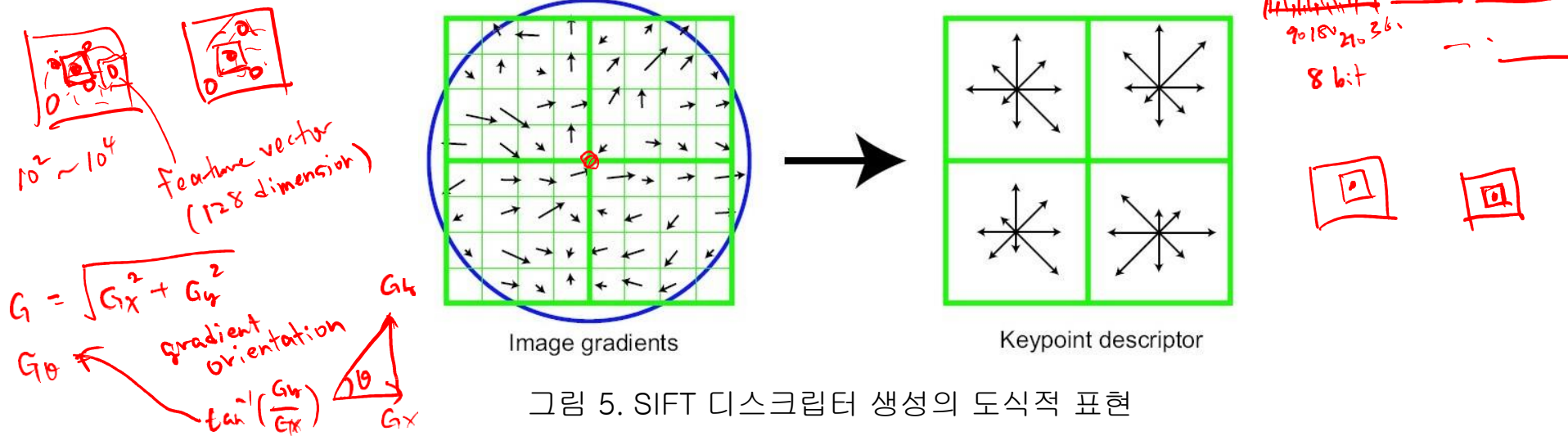


그림 4. 특징점 추출의 도식적 표현

Descriptor construction

- 특징점이 추출된 뒤에는 각 특징점 주변을 특정 길이의 디스크립터로 표현
- 특징점의 주변에 특정 크기의 윈도우를 정하고, 그 것을 4x4 영역으로 나누고 gradient 방향을 계산
- 단위 영역에서 얻어지는 Gradient(Week 2 실습 참고)의 방향을 8개의 정량화된 히스토그램으로 표현하고(45° 단위) 이것들을 이어 붙이면 $8 \times 4 \times 4 = 128$ 차원을 가지는 디스크립터가 얻어짐



이미지 매칭

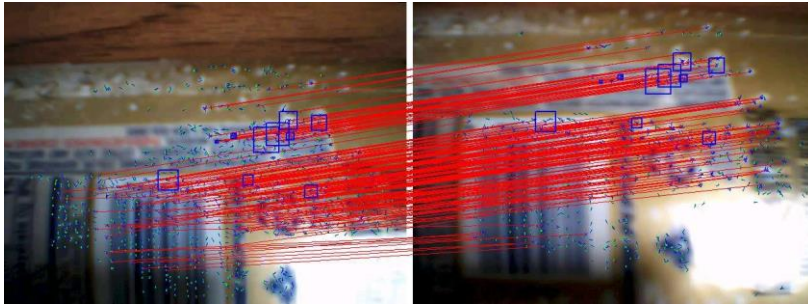
- 최종적으로 이미지의 유사성은 유사한 특징점의 숫자로 판단
- 특징점은 128 차원의 벡터로 표현되므로 L2-norm를 이용하여 공식 1과 같이 유사성을 측정
- 특징점 들의 거리를 측정한 후 특정 threshold 값 이하를 매칭이라고 할 수도 있으나, 여기에는 이미지의 특성에 따라 다양하게 나타나는 특징점들의 거리를 일률적으로 처리하게 되는 문제점 존재

$$d = \sqrt{\sum_{i=1}^{128} (k_{1i} - k_{2i})^2}$$

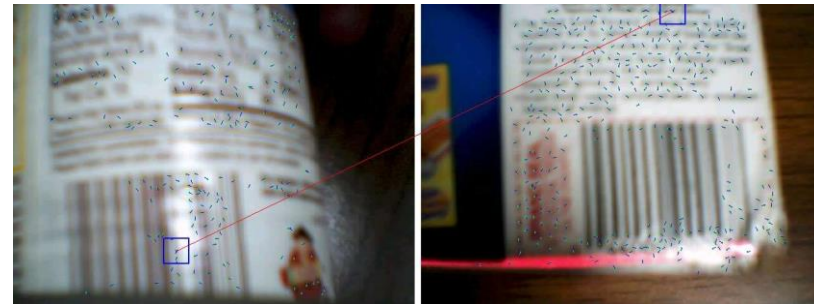
식 1. L2-norm을 이용한 특징점 거리 측정

이미지 매칭

- 따라서 SIFT 알고리즘에서는 유사성이 아니라, 유일성(uniqueness)에 중점을 두고 특징점의 매칭 여부를 결정. 첫번째 이미지에서 얻어진 하나의 특징점에 대하여 두번째 이미지의 모든 특징점들과의 거리를 구하고 가장 가까운 거리 $d1$ 과 그 다음 가까운 거리 $d2$ 를 구하여 $d1/d2$ 가 충분히 작다면 (일반적으로 0.49 사용) 매칭으로 간주



같은 물체의 경우 많은 특징점이 매칭됨

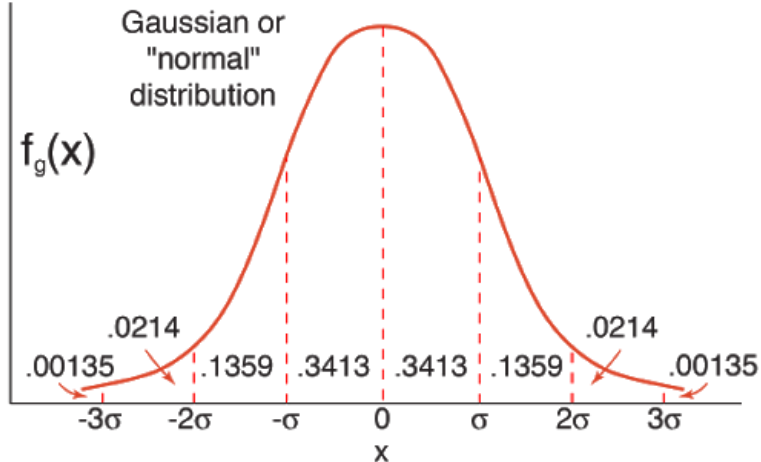


다른 물체의 경우 소수의 특징점이 매칭됨

그림 6. SIFT 매칭의 예

Gaussian 필터 구현(실습 문제 1.1)

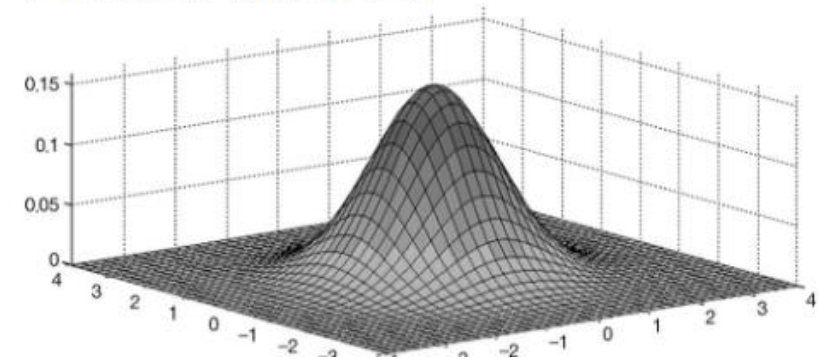
- 가우시안 필터 행렬은 중앙부(평균)에서 상대적으로 큰 값을 가지고 주변부로 갈수록 원소 값이 0에 가까운 값을 가짐



$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu_x)^2}{2\sigma^2}}$$

$\mu = (0,0), \sigma = 1$ 인 2차원 가우시안 함수 그래프

▼그림 8-10 2차원 가우시안 함수의 모습



$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2 + (y-\mu_y)^2}{2\sigma^2}}$$

- 필터의 모든 원소의 합이 1보다 크면 필터링된 영상이 전체적으로 원본보다 밝아지고, 1보다 작으면 어두워지기 때문에 전체 합을 1로 맞춰주는 작업이 중요 -> 각 원소를 전체합으로 나눠주기

Gaussian 필터 구현(실습 문제 1.1)

```
=====>Start my_cvSmooth
```

```
0.000000 0.000001 0.000007 0.000032 0.000053 0.000032 0.000007 0.000001 0.000000
0.000001 0.000020 0.000239 0.001072 0.001768 0.001072 0.000239 0.000020 0.000001
0.000007 0.000239 0.002915 0.013064 0.021539 0.013064 0.002915 0.000239 0.000007
0.000032 0.001072 0.013064 0.058550 0.096533 0.058550 0.013064 0.001072 0.000032
0.000053 0.001768 0.021539 0.096533 0.159156 0.096533 0.021539 0.001768 0.000053
0.000032 0.001072 0.013064 0.058550 0.096533 0.058550 0.013064 0.001072 0.000032
0.000007 0.000239 0.002915 0.013064 0.021539 0.013064 0.002915 0.000239 0.000007
0.000001 0.000020 0.000239 0.001072 0.001768 0.001072 0.000239 0.000020 0.000001
0.000000 0.000001 0.000007 0.000032 0.000053 0.000032 0.000007 0.000001 0.000000
kernel sum : 1.000000
myGaussianKernel[k_size = 9, sigma = 1.000000] min : 0.000000, max : 0.159156
```

Kernel size = 9, sigma = 1.0 일 때의 Gaussian 필터 출력 화면
-> kernel sum = 1.0이 되어야 함

Gaussian 필터링(실습 문제 1.2)

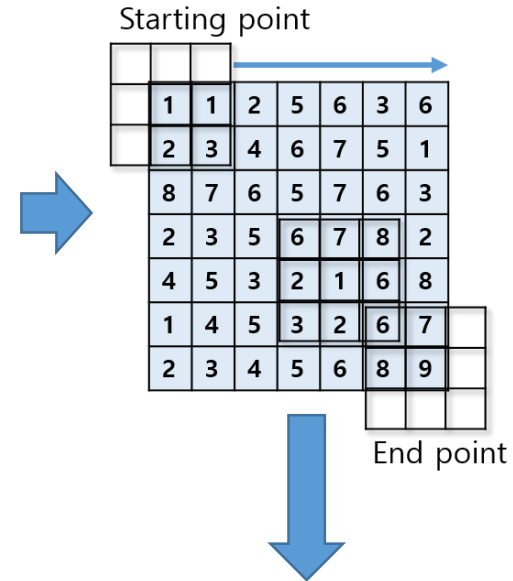
1	1	2	5	6	3	6
2	3	4	6	7	5	1
8	7	6	5	7	6	3
2	3	5	6	7	8	2
4	5	3	2	1	6	8
1	4	5	3	2	6	7
2	3	4	5	6	8	9

src->ImageData

*

0.075114	0.123841	0.075114
0.123841	0.20418	0.123841
0.075114	0.123841	0.075114

Kernel ($s=3 \times 3$, $\sigma=1.0$)



- IplImage 구조체 데이터 접근 : `cvGetReal2D(src, y, x) ;`
- IplImage 구조체 데이터 저장 : `cvSetReal2D(dst, y, x, value);`
- 2주차 필터링 실습 코드 적극 활용, 데이터 접근과 저장 방식 유의

0	1	2	3	3	3	2
2	3	4	5	5	4	2
3	4	5	5	6	5	2
3	4	4	4	5	5	3
2	3	3	3	4	5	4
2	3	3	3	3	5	5
1	2	2	3	3	4	4

dst->ImageData

* src, dst -> IplImage / kernel -> 2D array

특징점 매칭(실습 문제 2)

- SIFT 특징점의 descriptor 간의 similarity를 식 1을 이용해 계산
- 특징점 f_1, f_2 의 descriptor는 $f_1 \rightarrow \text{descr}$ 를 통해 접근 가능
- Descriptor는 128차원 벡터
- 첫번째 이미지에서 얻어진 하나의 특징점에 대하여 두번째 이미지의 모든 특징점들과의 거리를 구하고 가장 가까운 거리 d_1 과 그 다음 가까운 거리 d_2 를 구하여 d_1/d_2 가 충분히 작다면 (일반적으로 0.49 사용) 매칭으로 간주 (유일성에 중점을 둠)

실습/과제 제출

제출 안내

- 사이버캠퍼스 과제/실습 란을 통해 제출
- 제출 기한
 - 과제 : 다음 실습 일 전날 23:59분까지
 - Late 없음. 0점 처리함

예시) 수요일 반의 경우,

- 다음주 화요일 밤 11시 59분 59초까지

제출방식

- 제출 양식

- 첨부 파일

- ✓ [고소실_0주차~~실습~~]0반_20181600_홍길동.zip

- ✓ [고소실_0주차~~과제~~]0반_20181600_홍길동.zip

- 예시) [고소실_3주차실습]1반_20181600_홍길동.zip

**** 형식 틀릴 시 감점!**

***** 형식 미 준수로 인한 불이익은 본인 책임(과제 유실 우려)**

첨부 파일 제출 시 유의사항

- 실습파일 제출 시 첨부파일이 대용량으로 변환된 경우, 다음과 같이 처리하여 제출한다.
 1. 빌드 → 솔루션정리 후 저장
 2. Debug, Release 폴더 삭제
 3. ipch 폴더 삭제 – **필수!**
ex) VS2019의 경우 .vs 폴더 → 프로젝트 이름 폴더 → v16 폴더 → **ipch** 폴더 삭제
 4. 프로젝트 폴더 자체를 압축
- 만약 이렇게 했는데도 압축파일 크기가 30MB를 넘는다면, 실습 시 사용했던 cpp 파일만 압축하여 제출한다.
- **과제 시 코드를 제출할 경우에도 위와 동일한 방법으로 진행**

실습 끝난 후 조교 확인 시

파라미터 수정해도 괜찮음(kernel_size, sigma)

1. Problem1() 실행 결과

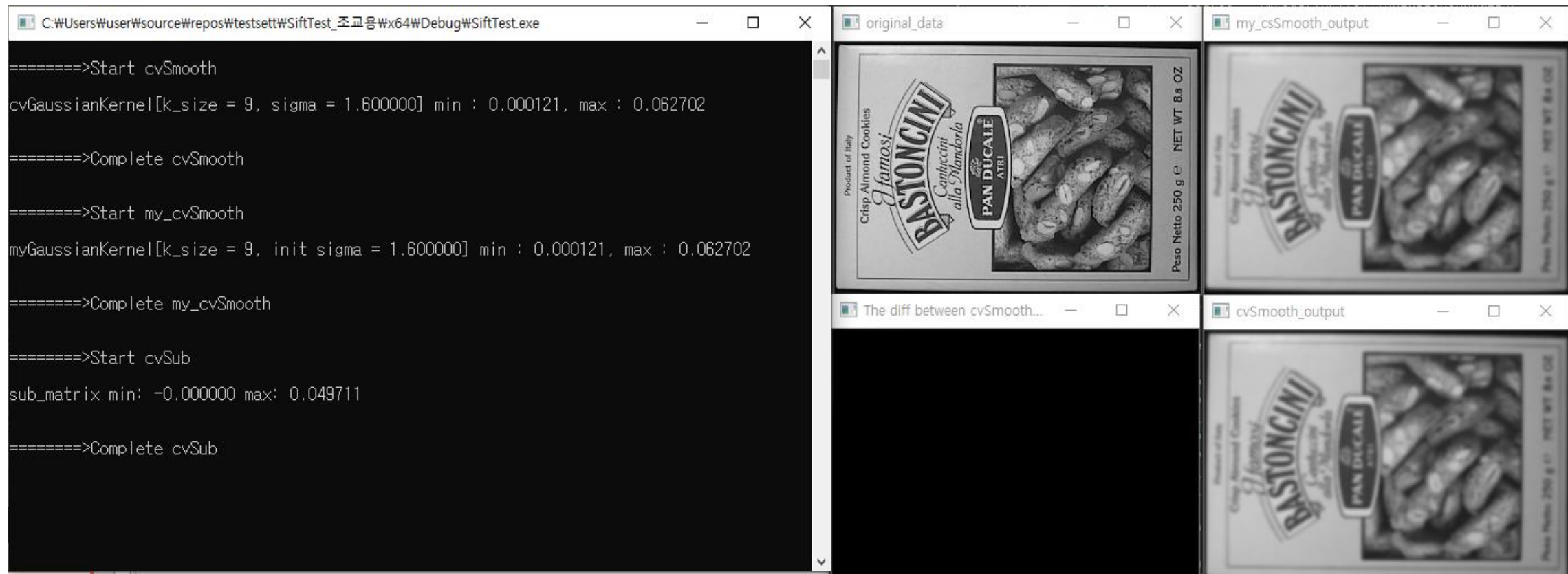
2. my_cvSmooth()의 Problem2() 실행 결과와 cvSmooth()의 Problem2() 실행 결과

```
SiftCommand.cpp  sift.cpp  X
SiftTest
451 }
452
453 if (debug) {
454     for (i = 0; i < intvls + 3; i++)
455         std::printf("sig[%d]= %f ", i, sig[i]+255.0);
456     std::printf("\n");
457 }
458
459
460 for (o = 0; o < octvs; o++)
461     for (i = 0; i < intvls + 3; i++)
462     {
463         //std::printf("[octave %d][intvls %d]= %f\n ", o, i, sig[i]);
464         //printf("[%d][%d]:sig[%d]=%f\n", o, i, i, sig[i]);
465         if (o == 0 && i == 0)
466             gauss_pyr[o][i] = cvCloneImage(base);
467
468         /* base of new octave is halved image from end of previous octave */
469         else if (i == 0)
470             gauss_pyr[o][i] = downsample(gauss_pyr[o - 1][intvls]);
471
472         /* blur the current octave's last image to create the next one */
473         else
474         {
475             gauss_pyr[o][i] = cvCreateImage(cvGetSize(gauss_pyr[o][i - 1]),
476                 IPL_DEPTH_32F, 1);
477
478             if (debug) {
479                 std::printf("[octave %d] [intvls %d]= %f\n ", o, i, sig[i]);
480             }
481
482             //cvSmooth(gauss_pyr[o][i - 1], gauss_pyr[o][i], CV_GAUSSIAN, 9, 0, sig[i], sig[i]);
483             my_cvSmooth(gauss_pyr[o][i - 1], gauss_pyr[o][i], 9, sig[i]);
484
485         }
486     }
487 }
```

```
SiftCommand.cpp  sift.cpp
SiftTest
55 void setPBR(int);
56 int Problem1();
57 int Problem2();
58 void makeSample();
59 int MatchAndDraw(struct feature*, struct feature*, int, int, IplImage*, int);
60 void my_cvSmooth(IplImage*, IplImage*, int, double);
61 double** make_2d_Gaussian_Kernel(int, double);
62 cv::Mat stack_mat_imgs(cv::Mat, cv::Mat);
63 IplImage* convert_to_gray32(IplImage*);
64 IplImage* convert_to_gray8(IplImage*);
65
66 void makeSample()
67 {
68     Mat source, dst;
69     string inFile, outFile;
70     inFile = "data1/box.bmp";
71     outFile = "data1/box2.bmp";
72
73     source = imread(inFile, IMREAD_UNCHANGED);
74
75     //cv::flip(source, dst, 2);
76     cv::rotate(source, dst, 0);
77     cv::resize(dst, dst, Size(223, 343));
78
79     imwrite(outFile, dst);
80 }
81
82 int main()
83 {
84     //Problem1();
85     Problem2();
86
87     // Gaussian Filtering
88     // Sift Descriptor Matching
89
90     return 0;
91 }
```

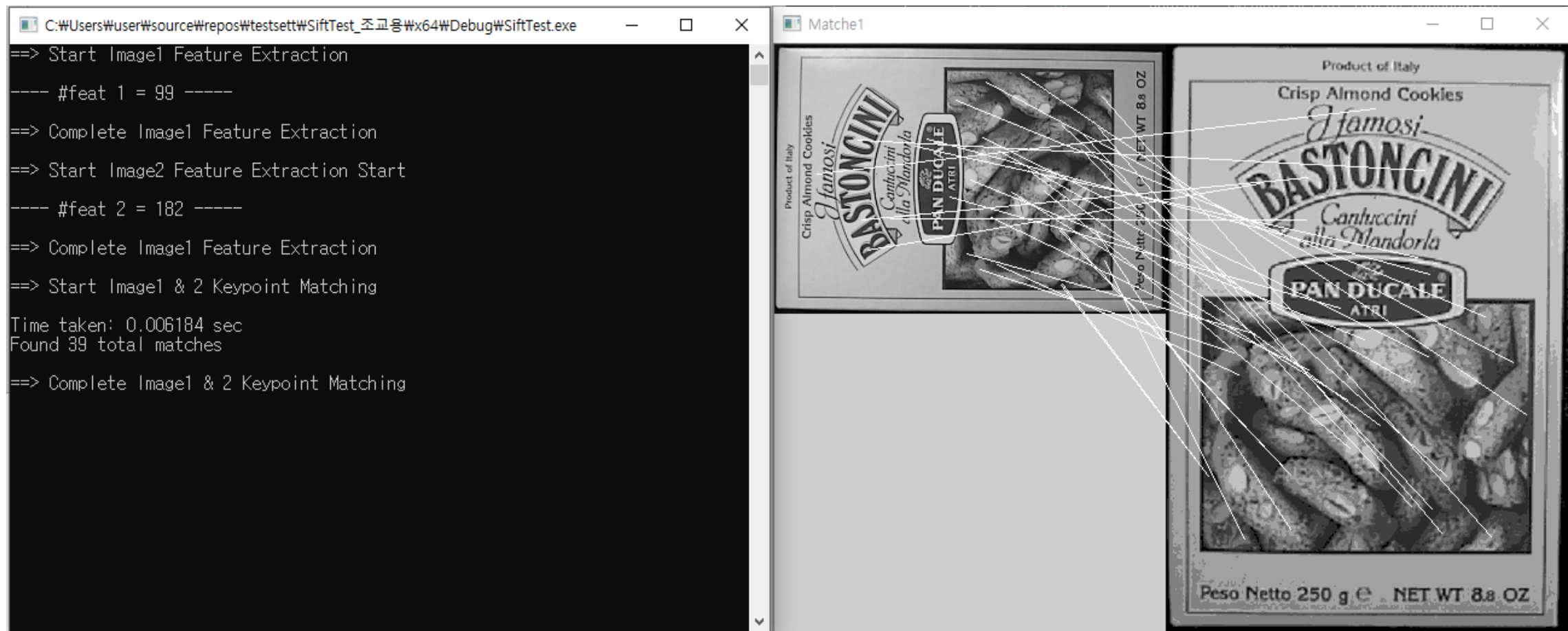
정답 결과 예시(Gaussian Filtering)

문제 1 . Gaussian Filtering 구현 ($k_size = 9$, $\sigma = 1.6$)



정답 결과 예시(Filtering)

문제 2 . 이미지 매칭



참고 사항

문제 1.1 : cv::GetGaussianKernel(kernel_size, sigma);

```
Mat kx = getGaussianKernel(kernel_size, sigma);
Mat ky = getGaussianKernel(kernel_size, sigma);
Mat _2d_Gaussian_Kernel = ky * kx.t();
minMaxIdx(_2d_Gaussian_Kernel, &k_min, &k_max);
printf("cvGaussianKernel[k_size = %d, sigma = %f] min : %lf, max : %lf\n", kernel_size, sigma, k_min, k_max);
cout << _2d_Gaussian_Kernel << endl;
```

문제 1.2 : cv::cvSmooth() & Problem 1 실행 결과(Image3_dst_mat : cvSmooth 결과 저장)

```
131
132 cv::imshow("original_data", Image_origin_mat);
133 cv::imshow("cvSmooth_output", Image3_dst_mat);
134
```

```
C:\Users\User\source\repos\testset\WstTest\조교용\#x64\Debug\#SiftTest.exe
=====>Start cvSmooth
cvGaussianKernel[k_size = 9, sigma = 1.600000] min : 0.000121, max : 0.062702
=====>Complete cvSmooth
=====>Start my_cvSmooth
myGaussianKernel[k_size = 9, init sigma = 1.600000] min : 0.000121, max : 0.062702
=====>Complete my_cvSmooth
=====>Start cvSub
sub_matrix min: -0.000000 max: 0.049711
=====>Complete cvSub
```

