KANGWON NATIONAL UNIVERSITY

컴퓨터비전 실습

실습8 | SIFT 기술자

실습과제 이루리 내 제출

CVMIPLAB @ KNU

실습 8-1 | SIFT 기술자

문제

주어진 코드를 활용하여 “cliff.bmp” 파일을 **흑백으로** 읽은 뒤,
SIFT 기술자 알고리즘을 구현하여 결과를 콘솔에 출력하세요.

요구 결과

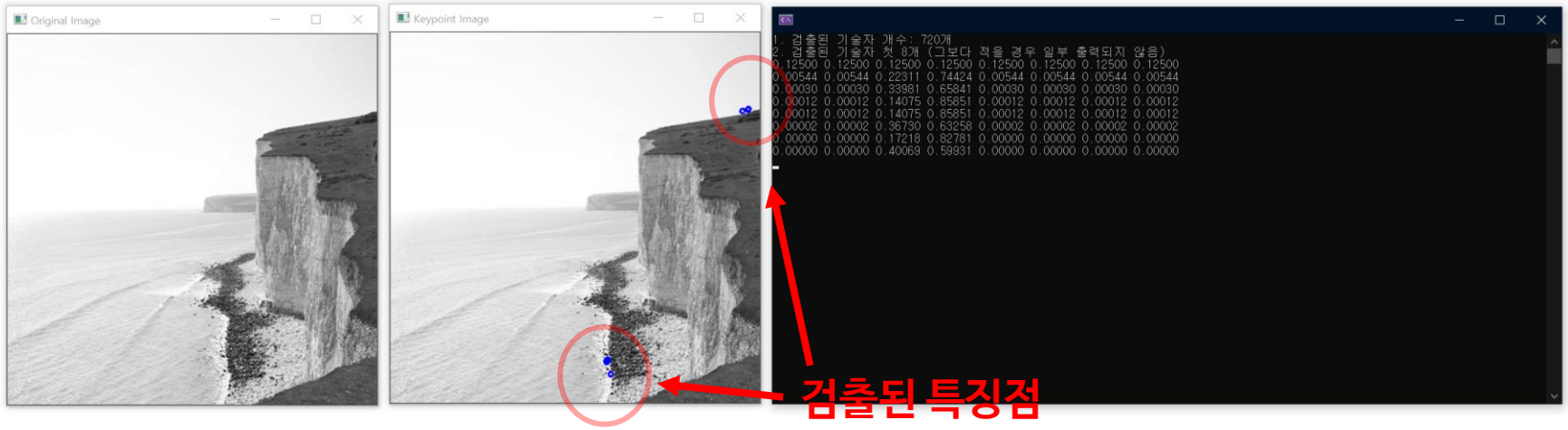
SIFT 기술자 알고리즘을 구현하고 콘솔의 결과를 “cliff.txt”로 저장합니다.

저장된 텍스트 파일과 “.cpp” 총 두 개의 파일을 압축하여 제출합니다.

제출 관련 공지

※ 압축파일 최상위에 폴더를 만들지 말고, 파일만 그대로 압축해 주시기 바랍니다.

※ 실습이 여러 개인 경우 한 압축 파일에 모두 압축하여 주시기 바랍니다.



실습 8-1 | SIFT 기술자

설명자료

알고리즘 6-1 SIFT 기술자 추출

입력 : 입력 영상 f에서 검출된 키포인트 집합 $p_i=(y_i, x_i, \sigma_i)$, $1 \le i \le n$ // 4.4.3절의 알고리즘으로 추출

출력 : 기술자가 추가된 키포인트 집합 $p_i=(y_i, x_i, \sigma_i, \theta_i, \mathbf{x}_i)$, $1 \le i \le m$

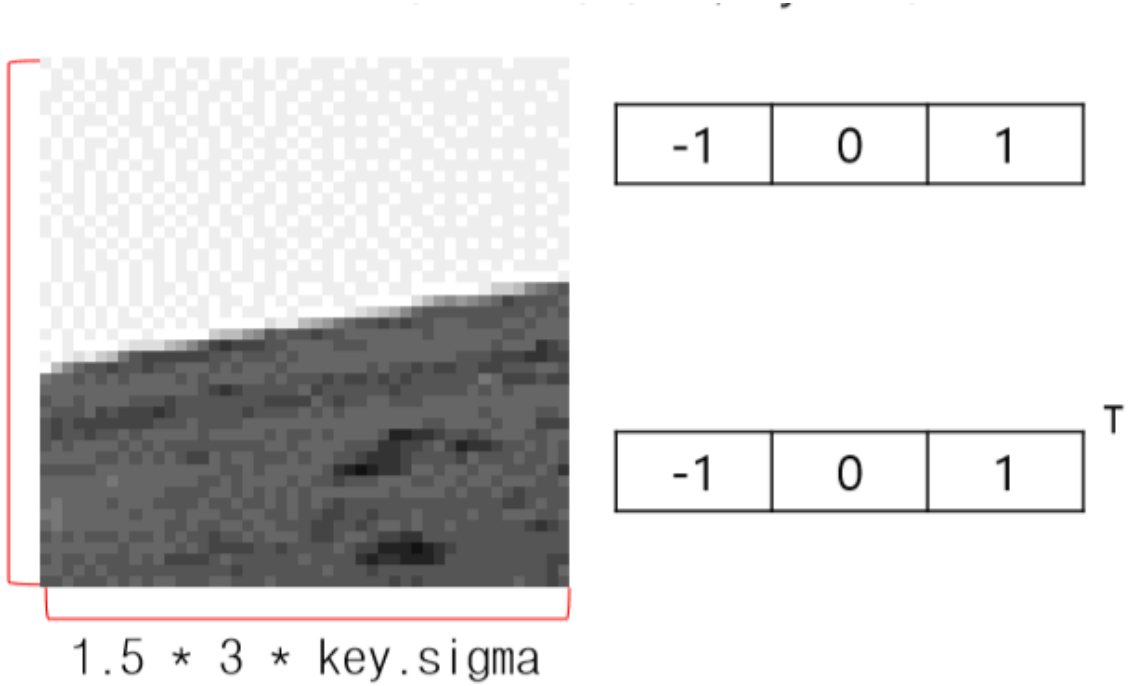
```
1  for(i=1 to n) {
2     p_i의 지배적인 방향 θ_i를 계산한다.  // 이때 하나의 키포인트가 여러 개로 나뉠 수 있음
3     p_i의 특징 벡터 x_i를 계산한다.
4     x_i를 정규화한다.
5  }
```

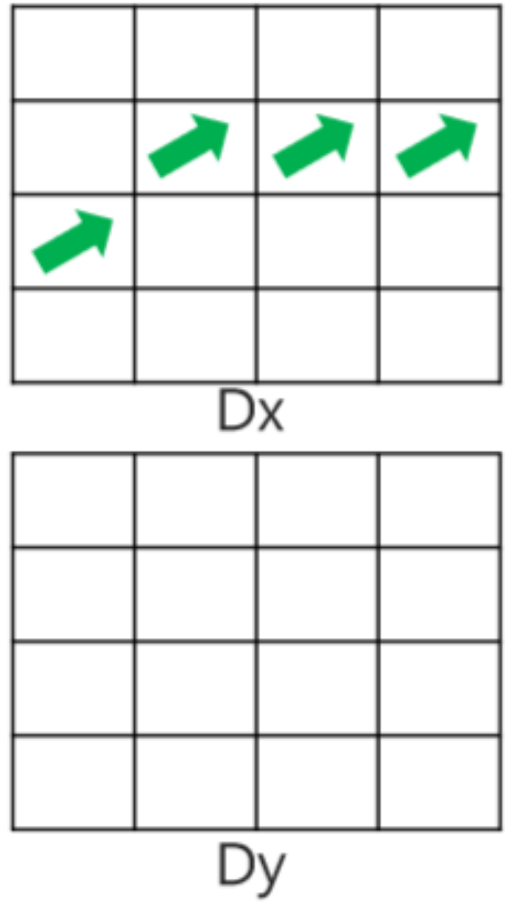
실습 8-1 | SIFT 기술자

설명자료

1단계: Dominant Orientation 찾기

→ 영상에서 D_x, D_y를 계산합니다.



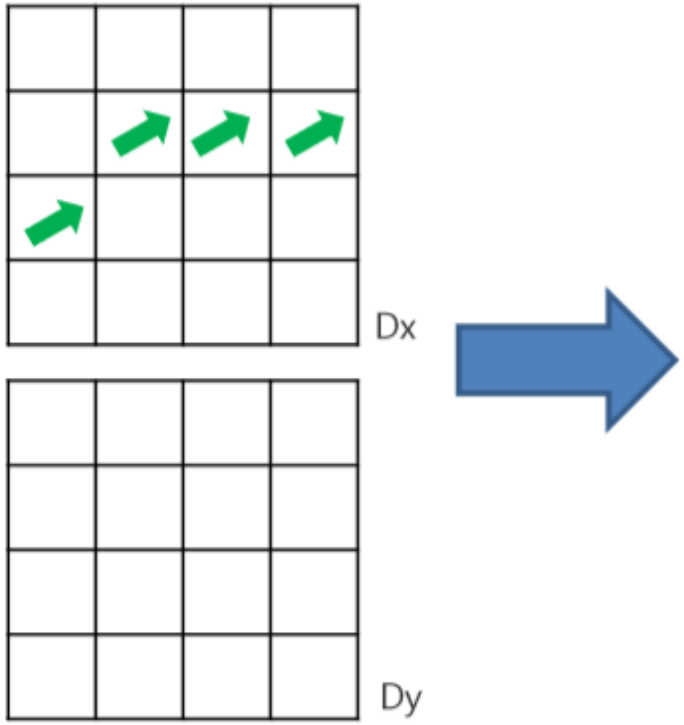


실습 8-1 | SIFT 기술자

설명자료

1단계: Dominant Orientation 찾기

→ D_x, D_y를 이용하여 에지의 방향과 크기를 계산합니다.



$$\text{Edge} = \arctan\left(\frac{dy}{dx}\right) + 90°$$

$$\text{Magnitude} = \sqrt{d_y^2 + d_x^2}$$

0	0	0	0
0	48	43	40
45	0	0	0
0	0	0	0

Edge

0	0	0	0
0	100	90	80
120	0	0	0
0	0	0	0

Magnitude

실습 8-1 | SIFT 기술자

설명자료

1단계: Dominant Orientation 찾기

→ 아래 함수를 이용하여 Edge의 방향과 크기를 계산하세요.

```
CV_EXPORTS_W void cartToPolar(InputArray x, InputArray y,
                             OutputArray magnitude, OutputArray angle,
                             bool angleInDegrees = false);
```

x : dx
y : dy
magnitude : 에지 크기
angle : 에지 방향
angleInDegrees : 각도로 출력할 것인가(실습에서 True로 처리)

실습 8-1 | SIFT 기술자

설명자료 **1단계:** Dominant Orientation 찾기

```cpp
void SIFT::FindDominantOrientation(std::vector<KeyPoint>& key_points, std::vector<OctaveSet>& octave_sets)
{
    // 에지 히스토그램 개수
    const int SIFT_ORI_HIST_BINS = 36;
    for (int i = 0; i < key_points.size(); ++i)
    {
        KeyPoint& key_point = key_points[i];

        /*
            교재p.256
            Hess가 구현한 방법에서는 1.5 x 3 x Sigma를 반올림하여 w를 구한 후
            (2w+1) x (2w+1)의 윈도우를 사용하였다.
        */
        int radius = 1.5 * 3 * key_point.sigma;

        std::vector<double> angles;
        std::vector<double> magnitudes;

        // Octave와 Sigma에 맞는 영상으로 처리
        int sigmaIdx = GetOctaveIdx(key_point.sigma);
        assert((0 <= sigmaIdx) && (sigmaIdx < 6));
        cv::Mat img = octave_sets[key_point.octave].gaussianMat[sigmaIdx];
        for (int h = -radius; h <= radius; ++h)
        {
            int y = key_point.y + h;
            if (y <= 0 || y >= img.rows - 1)
                continue;
            for (int w = -radius; w <= radius; w++)
            {
                int x = key_point.x + w;
                if (x <= 0 || x >= img.cols - 1)
                    continue;

                // 픽셀별 dx와 dy를 계산하시오.
                // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
                float dx =
                float dy =
                // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
```

```cpp
cv::Mat magnitude_map(cv::Size(1, 1), CV_64FC1);
cv::Mat angle_map(cv::Size(1, 1), CV_64FC1);

/*
    void cartToPolar(InputArray x, InputArray y,
                     OutputArray magnitude, OutputArray angle,
                     bool angleInDegrees = false);
    @x dx
    @y dy
    @magnitude 에지 크기
    @angle 에지 방향
    @angleInDegrees true일 경우 degree (여기서는 true)
*/
// 에지 방향과 에지 크기를 계산하시오.
// ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

// ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

angles.push_back(angle_map.at<double>(0, 0));
magnitudes.push_back(magnitude_map.at<double>(0, 0));
```

실습 8-1 | SIFT 기술자

설명자료

2단계: SIFT 기술자 추출하기

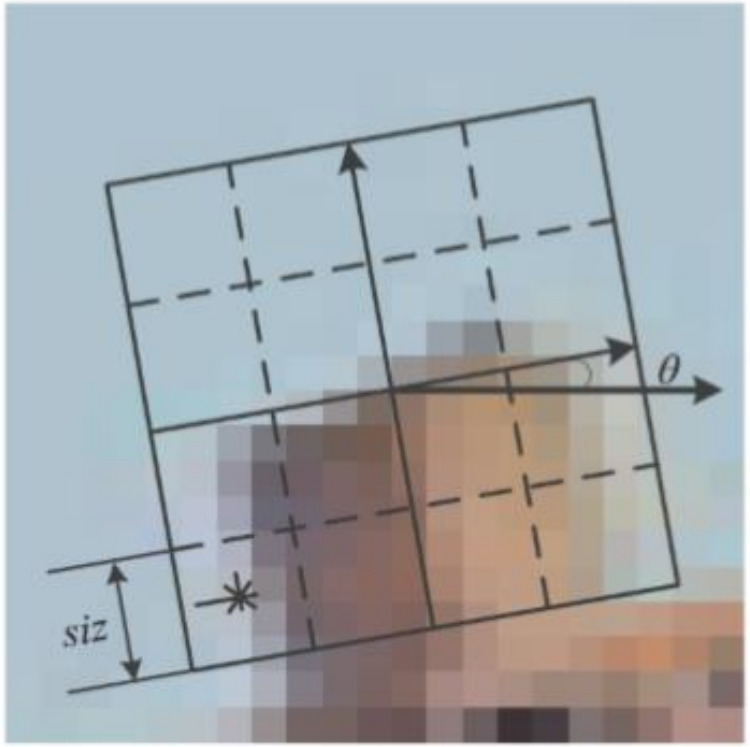


그림 6-4 SIFT 기술자 추출을 위한 좌표계와 4×4 블록

Dominant 방향으로 윈도우를 기울인 후
에지 히스토그램을 계산

실습 8-1 | SIFT 기술자

설명자료

2단계: SIFT 기술자 추출하기

→ 복합 동차 행렬을 이용하여 영상을 중점 회전합니다.

표 2-1 기하 변환을 위한 동차 행렬

변환	동차 행렬 $\dot{\mathbf{H}}$	설명
이동	$T(t_y, t_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_y & t_x & 1 \end{pmatrix}$	y방향으로 t_y, x방향으로 t_x만큼 이동
회전	$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$	원점을 중심으로 시계방향으로 θ만큼 회전
크기	$S(s_y, s_x) = \begin{pmatrix} s_y & 0 & 0 \\ 0 & s_x & 0 \\ 0 & 0 & 1 \end{pmatrix}$	y방향으로 s_y, x방향으로 s_x만큼 확대
기울임	$Sh_y(h_y) = \begin{pmatrix} 1 & 0 & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $Sh_x(h_x) = \begin{pmatrix} 1 & h_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	Sh_y: y방향으로 h_y만큼 기울임 Sh_x: x방향으로 h_x만큼 기울임

실습 8-1 | SIFT 기술자

설명자료

2단계: SIFT 기술자 추출하기

→ 아래 함수를 이용하여 복합 동차 행렬을 계산합니다.

```
CV_EXPORTS_W Mat getRotationMatrix2D( Point2f center, double angle, double scale );
```

center : 중점 – cv::Point로 넘김 ex) cv::Point(centerX, centerY)
angle : 회전 방향
scale : 스케일(실습에서는 1)

실습 8-1 | SIFT 기술자

설명자료

2단계: SIFT 기술자 추출하기

```cpp
std::vector<DescriptorVector> SIFT::MakeDescriptor(std::vector<KeyPoint>& key_points, std::vector<OctaveSet>& octave_sets)
{
    std::vector<DescriptorVector> descriptor_vectors;

    // 에지 히스토그램 개수
    const int SIFT_ORI_HIST_BINS = 8;
    for (int i = 0; i < key_points.size(); ++i)
    {
        KeyPoint& key = key_points[i];
        /*
            교재 p.257
            Hess가 구현한 방법에서는 siz를 3 x sigma로 설정하였다.
        */
        int siz = 3 * key.sigma;

        // 36개로 양자화하여 10도씩 에지 히스토그램을 만들어기 때문에
        // 원래 에지를 계산하기 위해서 10을 곱해줌
        float orientation = key.dominantOrientation * 10;

        cv::Point beginPoint = cv::Point(key.x - (2 * siz), key.y - (2 * siz));
        cv::Point endPoint = cv::Point(key.x + (2 * siz), key.y + (2 * siz));

        // Octave와 Sigma에 맞는 영상으로 처리
        int sigmaIdx = GetOctaveIdx(key.sigma);
        const cv::Mat& img = octave_sets[key.octave].gaussianMat[sigmaIdx];

        // 중점에서 회전하는 복합 행렬을 구하시오.
        /*
            getRotationMatrix2D(Point2f center, double angle, double scale);
            @center 중점 (cv::Point). 예를 들어, cv::Point(centerX, centerY)
            @angle 회전 방향
            @scale 스케일 (실습에서는 1)
        */
        // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
        cv::Mat m = [redacted]
        // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
```

실습 8-1 | SIFT 기술자

설명자료

2단계: SIFT 기술자 추출하기

```
std::vector<double> angles;
std::vector<double> magnitudes;
for (int h2 = h; h2 < h + siz; h2++)
{
    for (int w2 = w; w2 < w + siz; w2++)
    {
        cv::Mat orgIdx(cv::Size(1, 2), CV_64FC1);

        orgIdx.at<double>(0) = h2;
        orgIdx.at<double>(1) = w2;

        // 복합 동차 행렬 계산
        int rotationIdxY = int(orgIdx.at<double>(0) * m.at<double>(0) + orgIdx.at<double>(1) * m.at<double>(1) + m.at<double>(2));
        int rotationIdxX = int(orgIdx.at<double>(0) * m.at<double>(3) + orgIdx.at<double>(1) * m.at<double>(4) + m.at<double>(5));

        if (rotationIdxX > 0 && rotationIdxX < img.cols - 1 && rotationIdxY > 0 && rotationIdxY < img.rows - 1)
        {
            // 픽셀별 dx와 dy를 계산하시오.(rotationIdxY와 rotationIdxX를 이용하세요.)
            // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
            float dx =
            float dy =
            // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

            cv::Mat magnitude_map(cv::Size(1, 1), CV_64FC1);
            cv::Mat angle_map(cv::Size(1, 1), CV_64FC1);

            /*
                void cartToPolar(InputArray x, InputArray y,
                                 OutputArray magnitude, OutputArray angle,
                                 bool angleInDegrees = false);
                @x dx
                @y dy
                @magnitude 에지 크기
                @angle 에지 방향
                @angleInDegrees true일 경우 degree (여기서는 true)
            */
            // 에지 방향과 에지 크기를 계산하시오.
            // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

            // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

            angles.push_back(angle_map.at<double>(0));
            magnitudes.push_back(magnitude_map.at<double>(0));
        }
    }
}
```

실습 8-1 | SIFT 기술자

설명자료

2단계: SIFT 기술자 추출하기

```
DescriptorVector vector;
memset(&vector, 0, sizeof(DescriptorVector));

for (int s = 0; s < angles.size(); ++s)
{
    // bin = 8단계로 양자화된 에지 방향 값.
    // cvRound 함수와 SIFT_ORI_HIST_BINS 상수를 이용하여 bin값을 반올림하여 구하세요.
    // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
    int bin =
    // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
    if (bin >= SIFT_ORI_HIST_BINS)
        bin -= SIFT_ORI_HIST_BINS;
    if (bin < 0)
        bin += SIFT_ORI_HIST_BINS;

    vector.descriptor[bin] += (1 * magnitudes[s]);
}

float sum_of_histogram = 0;

// 정규화 구현
// Descriptor 값들에 대한 정규화를 진행합니다.
// SIFT_ORI_HIST_BINS개의 히스토그램 값 vector.descriptor들에 대해,
// 모든 값을 합한 값 sum_of_histogram으로 각 element를 나누어줍니다.

// ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **


// ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

descriptor_vectors.push_back(vector);
```

실습 8-1 | SIFT 기술자

결과영상

※ 기술자 개수나 원의 위치 등 구현에 따라 달라질 수 있음