KANGWON NATIONAL UNIVERSITY

컴퓨터비전 실습

실습11 | Panorama

실습과제 이루리 내 제출

CVMIPLAB @ KNU

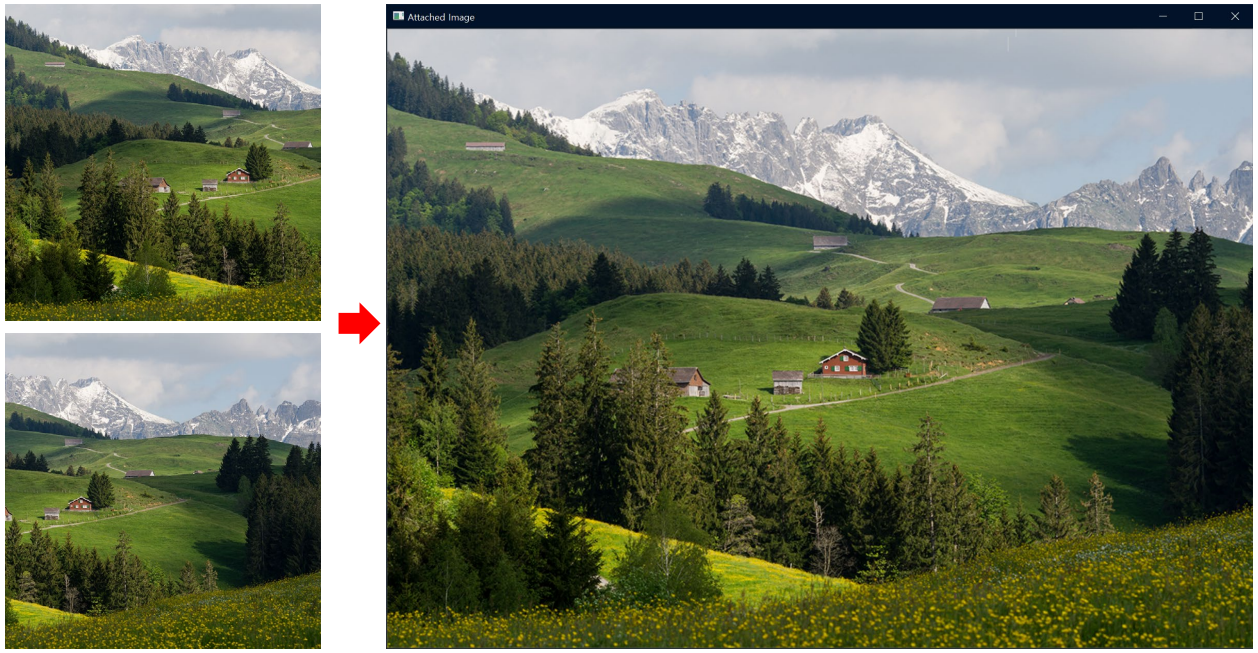
실습 11-1 | Panorama

문제

두 장의 이미지를 붙이는 파노라마 연산을 구현합니다.

요구 결과

주어진 이미지 landscape-left.bmp와 landscape-right.bmp 두 장을 이어붙이는 파노라마 연산을 구현합니다.



과제 제출

- 수행한 모든 실습 cpp 파일 총 4개(**distance_metric.cpp, Moravec.cpp, pransac.cpp**)와 **attached-landscape.bmp** 이미지를 같이 압축하여 **최상위 디렉토리 없이** 이루리 시스템에 제출합니다.

실습 11-1 | Panorama

설명자료 - 모라벡 알고리즘

```
/**
 * 1단계 - 모라벡 알고리즘을 이용한 Keypoint 검출
 * CreateConfidence()와 FindKeyPoint()를 구현하세요.
 */
Moravec moravec_first(first_image);
moravec_first.CreateConfidence();
moravec_first.FindKeyPoint();

Moravec moravec_second(second_image);
moravec_second.CreateConfidence();
moravec_second.FindKeyPoint();
```

```
void Moravec::CreateConfidence()
{
    for (int h = 2; h < image.rows - 2; ++h)
        for (int w = 2; w < image.cols - 2; ++w)
        {
            // 3x3 그리드의 중점이 (0, 0)이라고 가정.
            // S1 = (-1, 0), S2 = (1, 0), S3 = (0, -1), S4 = (0, 1)
            int s1 = 0;         int s2 = 0;
            int s3 = 0;         int s4 = 0;

            // 3x3 그리드
            for (int y = -1; y <= 1; ++y)
                for (int x = -1; x <= 1; ++x)
                {
                    // 교재 p.163 / 실습6 Moravec 자료 p.5를 참조하여 제곱차합을 계산하세요.
                    // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
                    float top_diff =  // 상
                    float bottom_diff =  // 하
                    float left_diff = // 좌
                    float right_diff = // 우
                    // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

                    s1 += pow(top_diff, 2);
                    s2 += pow(bottom_diff, 2);
                    s3 += pow(left_diff, 2);
                    s4 += pow(right_diff, 2);
                }

            // s1, s2, s3, s4 중 가장 작은 값을 confidence_map에 추가하시오.
            // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
            int c =
            // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

            confidence_map.at<float>(h, w) = c;
        }
}
```

실습 11-1 | Panorama

설명자료 - 모라벡 알고리즘

```
/**
 * 1단계 - 모라벡 알고리즘을 이용한 Keypoint 검출
 * CreateConfidence()와 FindKeyPoint()를 구현하세요.
 */
Moravec moravec_first(first_image);
moravec_first.CreateConfidence();
moravec_first.FindKeyPoint();

Moravec moravec_second(second_image);
moravec_second.CreateConfidence();
moravec_second.FindKeyPoint();
```

```
void Moravec::FindKeyPoint()
{
    for (int y = 16; y < image.rows - 16; ++y)
        for (int x = 16; x < image.cols - 16; ++x)
        {   // confidence_map을 순회하면서 임계값보다 값이 크면
            // 해당 값이 위치한 좌표를 keyPointVec에 추가하세요.
            // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **


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

실습 11-1 | Panorama

설명자료 - HOG (Histogram of Oriented Gradient)

```
/**
 * 2단계 - 그레이디언트의 히스토그램을 통한 특징 추출
 */
HOG hog;
std::vector<HOGPair> first_image_hog = hog.GetHOG(first_image, moravec_first);
std::vector<HOGPair> second_image_hog = hog.GetHOG(second_image, moravec_second);
```

```
vector<HOGPair> HOG::GetHOG(const Mat& image, Moravec& moravec)
{
    Mat _image = image.clone();

    Mat img_gx; Mat img_gy; // Calculate gradients gx, gy
    Sobel(_image, img_gx, CV_64FC1, 1, 0, 3);
    Sobel(_image, img_gy, CV_64FC1, 0, 1, 3);

    Mat magnitude;  Mat angle;
    cartToPolar(img_gx, img_gy, magnitude, angle, 1);   // magnitude, angle 추출.

    vector<HOGPair> image_histogram;

    int kernelSize = 13;
    int halfKernelSize = kernelSize / 2;

    for (int idx = 0; idx < moravec.keyPointVec.size(); idx++)
    {
        cv::Point keypoint = moravec.keyPointVec[idx];

        if ((keypoint.y - halfKernelSize >= 0) && (keypoint.x - halfKernelSize >= 0)
            && (keypoint.y + halfKernelSize <= image.cols - 1) && (keypoint.x + halfKernelSize <= image.rows - 1))
        {
            Mat mask_Mag = magnitude(cv::Rect(keypoint.x - halfKernelSize, keypoint.y - halfKernelSize, kernelSize, kernelSize)).clone();
            Mat mask_Ang = angle(cv::Rect(keypoint.x - halfKernelSize, keypoint.y - halfKernelSize, kernelSize, kernelSize)).clone();

            vector<double> histogram = StackHistogram(mask_Mag, mask_Ang);  // bin 9인 histogram 쌓기.
            image_histogram.push_back(HOGPair(histogram, keypoint));
        }
    }
    return image_histogram;
}
```

실습 11-1 | Panorama

설명자료 - Euclidean Distance

```
/**
 * 3단계 - 추출한 특징간 거리 비교 및 두 이미지간 특징 Pair 검출
 * ExtractMinDistance 내 CalculateDistance 함수를 구현하세요.
 */
DistanceCalculator euclidean_distance;
euclidean_distance.ExtractMinDistance(first_image_hog, second_image_hog);
// euclidean_distance.DisplayPairLine(first_image, second_image);
std::vector<PointSet> distance_pair = euclidean_distance.GetDistancePair();
```

```
void DistanceCalculator::ExtractMinDistance(const vector<HOGPair>& h1, const vector<HOGPair>& h2)
{
    vector<PointSet> vector;
    double threshold = 900;

    for (auto pair1 : h1)
    {
        double min_distance = DBL_MAX;
        cv::Point pair2_min_dist_pt;

        for (auto pair2 : h2)
        {
            double distance = CalculateDistance(pair1.hog, pair2.hog);

            if (min_distance > distance)
            {
                min_distance = distance;
                pair2_min_dist_pt = pair2.keypoint;
            }
        }

        if (threshold > min_distance)
        {
            PointSet data { pair1.keypoint, pair2_min_dist_pt, min_distance };
            vector.push_back(data);
        }
    }

    distance_pair.assign(vector.begin(), vector.end());
}
```

실습 11-1 | Panorama

설명자료 - Euclidean Distance

```
/**
 * 3단계 - 추출한 특징간 거리 비교 및 두 이미지간 특징 Pair 검출
 * ExtractMinDistance 내 CalculateDistance 함수를 구현하세요.
 */
DistanceCalculator euclidean_distance;
euclidean_distance.ExtractMinDistance(first_image_hog, second_image_hog);
// euclidean_distance.DisplayPairLine(first_image, second_image);
std::vector<PointSet> distance_pair = euclidean_distance.GetDistancePair();
```

```
double DistanceCalculator::CalculateDistance(const vector<double>& h1, const vector<double>& h2)
{

    double result = 0;
    for (int i = 0; i < h1.size(); i++)
    {
        double hist1_bin_value = h1.at(i);
        double hist2_bin_value = h2.at(i);

        // 두 Histogram Bin 값 간의 Euclidean Distance를 구하세요.
        // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
        result +=
        // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
    }

    return result;
}
```

실습 11-1 | Panorama

설명자료 - RANSAC

- 수식 (7.14)에서 구해진 6x1 행렬 T는 아래와 같이 동차 행렬로 변환하여 두번째 이미지의 좌표를 Match시킬 수 있습니다.

$$\begin{pmatrix} t_{11} \\ t_{21} \\ t_{31} \\ t_{12} \\ t_{22} \\ t_{32} \end{pmatrix} \rightarrow \begin{pmatrix} b_{i1}' & b_{i2}' & 1 \end{pmatrix} = \begin{pmatrix} a_{i1} & a_{i2} & 1 \end{pmatrix} \begin{pmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{pmatrix} \quad (7.12)$$

실습 11-1 | Panorama

설명자료 - RANSAC

- RANSAC 알고리즘 내 GetHomogeneousMatrix 함수를 구현합니다.
- 교재 p.322나 7장 강의자료(매칭) p.30에 있는 수식 (7.14)를 참고하여 변환행렬 T를 구하세요.

$$\begin{pmatrix} \sum_{i=1}^{n} a_{i1}^2 & \sum_{i=1}^{n} a_{i1}a_{i2} & \sum_{i=1}^{n} a_{i1} & 0 & 0 & 0 \\ \sum_{i=1}^{n} a_{i1}a_{i2} & \sum_{i=1}^{n} a_{i2}^2 & \sum_{i=1}^{n} a_{i2} & 0 & 0 & 0 \\ \sum_{i=1}^{n} a_{i1} & \sum_{i=1}^{n} a_{i2} & \sum_{i=1}^{n} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sum_{i=1}^{n} a_{i1}^2 & \sum_{i=1}^{n} a_{i1}a_{i2} & \sum_{i=1}^{n} a_{i1} \\ 0 & 0 & 0 & \sum_{i=1}^{n} a_{i1}a_{i2} & \sum_{i=1}^{n} a_{i2}^2 & \sum_{i=1}^{n} a_{i2} \\ 0 & 0 & 0 & \sum_{i=1}^{n} a_{i1} & \sum_{i=1}^{n} a_{i2} & \sum_{i=1}^{n} 1 \end{pmatrix} \begin{pmatrix} t_{11} \\ t_{21} \\ t_{31} \\ t_{12} \\ t_{22} \\ t_{32} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} a_{i1}b_{i1} \\ \sum_{i=1}^{n} a_{i2}b_{i1} \\ \sum_{i=1}^{n} b_{i1} \\ \sum_{i=1}^{n} a_{i1}b_{i2} \\ \sum_{i=1}^{n} a_{i2}b_{i2} \\ \sum_{i=1}^{n} b_{i2} \end{pmatrix} \quad (7.14)$$

교재에 오타가 있습니다.
다음으로 수정하여 구현합니다.

$$a_{i1}^2 \rightarrow a_{i2}^2$$

실습 11-1 | Panorama

설명자료 - RANSAC

```
/**
 * 4단계 - 변환 행렬 구하기 단계
 *
 * 구한 distance_pair를 이용하여 변환 행렬을 구한다.
 * 교재 p.325의 알고리즘 7-9를 이용하여 기하 변환 행렬 T를 추정한다.
 *
 * 기하변환 T는 이미지#1을 이미지#2에 해당하는 점으로 변환하는 행렬이다.
 * 이 특성을 이용하여, 후방 기하 변환을 이용하여 새로운 이미지를 만들어 내도록 한다.
 *
 * GetBestRotationMat 함수 내 GetHomogeneousMatrix 함수를 구현하세요.
 */
PRANSAC ransac(distance_pair);
ransac.GetBestRotationMat();
ransac.AttachImage(first_image_color, second_image_color, attached_image);
```

```
void PRANSAC::GetHomogeneousMatrix(const std::vector<PointSet>& selected_points, const cv::OutputArray& homogeneous_matrix)
{
    // Hint: A dot B = C
    cv::Mat A = cv::Mat::zeros(6, 6, CV_64FC1);
    cv::Mat C = cv::Mat::zeros(6, 1, CV_64FC1);

    // 교재 p.322의 수식 (7.14)에서의
    // 첫번째 6*6 배열을 생성합니다.
    for (auto& pair : selected_points)
    {
        int a_i1 = pair.firstImgPtr.y;
        int a_i2 = pair.firstImgPtr.x;

        // 교재나 강의자료를 참고하여 Matrix A를 채우시오.
        // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **




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

    }
```

실습 11-1 | Panorama

설명자료 - RANSAC

```
/**
 * 4단계 - 변환 행렬 구하기 단계
 *
 * 구한 distance_pair를 이용하여 변환 행렬을 구한다.
 * 교재 p.325의 알고리즘 7-9를 이용하여 기하 변환 행렬 T를 추정한다.
 *
 * 기하변환 T는 이미지#1을 이미지#2에 해당하는 점으로 변환하는 행렬이다.
 * 이 특성을 이용하여, 후방 기하 변환을 이용하여 새로운 이미지를 만들어 내도록 한다.
 *
 * GetBestRotationMat 함수 내 GetHomogeneousMatrix 함수를 구현하세요.
 */
PRANSAC ransac(distance_pair);
ransac.GetBestRotationMat();
ransac.AttachImage(first_image_color, second_image_color, attached_image);
```

```
    // 교재 p.322의 수식 (7.14)에서의
    // 마지막의 결과 6*1 배열을 생성합니다.
    for (auto& pair : selected_points)
    {
        int a1 = pair.firstImgPtr.y;        int a2 = pair.firstImgPtr.x;
        int b1 = pair.secondImgPtr.y;       int b2 = pair.secondImgPtr.x;

        // 교재나 강의자료를 참고하여 Matrix C를 채우시오.
        // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **









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

    // 위에서 구한 행렬 A, C를 이용해서
    // 수식 (7.14) 중간의 [t_11 ~ t_32] 행렬 B을 구하세요.
    // ** 지금부터 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **
    cv::Mat B =
    // ** 여기까지 코드를 작성하세요. 이 줄은 지우시면 안 됩니다 **

    B.copyTo(homogeneous_matrix);
}
```

실습 11-1 | Panorama

결과영상