Homework#10/center>

컴퓨터소프트웨어학부 2017029589 류지범

#0

- 소스코드는 main.py 에 존재한다.
- Python 3.8.5 환경에서 작성했다.
- Left img와 right img를 받아서 크기를 resize후 특징점을 추출했다.

#1 Feature descriptor

- 특징점 추출은 opencv 모듈을 사용해서 진행했다.

```
def getMatchedKey(leftImg, rightImg):
    gray1 = cv2.cvtColor(leftImg, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(rightImg, cv2.COLOR_BGR2GRAY)
    detector = cv2.ORB_create()
    kp1, desc1 = detector.detectAndCompute(gray1, None)
    kp2, desc2 = detector.detectAndCompute(gray2, None)
    matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
    matches = matcher.match(desc1, desc2)

    left = []
    right = []

    for i in range(len(matches)):
        left.append(kp1[matches[i].queryIdx].pt)
        right.append(kp2[matches[i].trainIdx].pt)
    return left, right
```

- left와 right에 특징점 정보를 담도록 했다.

#2 Hessian Matrix

- 각 a에 대해 편미분한 결과는 다음과 같다.

$$\frac{\partial}{\partial a_{11}} = \frac{x}{a_{31}x + a_{32}y + 1} = \frac{x}{f} \qquad \frac{\partial}{\partial a_{31}} = -\frac{t'}{t^2} = -\frac{(a_{11}x + a_{12}y + a_{13})\cdot x}{(a_{31}x + a_{32}y + 1)^2} = -\frac{gx}{f^2}$$

- $$\frac{\partial}{\partial a_{12}} = \frac{y}{a_{31}x + a_{32}y + 1} = \frac{y}{f} \qquad \frac{\partial}{\partial a_{32}} = -\frac{t'}{t^2} = -\frac{(a_{11}x + a_{12}y + a_{13})y}{(a_{31}x + a_{32}y + 1)^2} = -\frac{gy}{f^2}$$

$$\frac{\partial}{\partial a_{13}} = \frac{1}{a_{31}x + a_{32}y + 1} = \frac{1}{f}$$

```python
def f(d, e, x, y):
    return d * x + e * y + 1


def g(a, b, c, x, y):
    return a * x + b * y + c


def partialA(a, b, c, d, e, x, y):
    return x / f(a, b, x, y)


def partialB(a, b, c, d, e, x, y):
    return y / f(a, b, x, y)


def partialC(a, b, c, d, e, x, y):
    return 1 / f(a, b, x, y)


def partialD(a, b, c, d, e, x, y):
    return -x * g(a, b, c, x, y) / (f(a, b, x, y) ** 2)


def partialE(a, b, c, d, e, x, y):
    return -y * g(a, b, c, x, y) / (f(a, b, x, y) ** 2)
```

#3 Gradient Vector

- x와 y에 대한 gradient vector는 다음과 같이 구할 수 있다.

```python
def gradient(a11, a12, a13, a21, a22, a23, a31, a32, left, right):
    xGradient = np.array([.0 for _ in range(5)], 'float64')
    yGradient = np.array([.0 for _ in range(5)], 'float64')

    for i in range(len(left)):
        leftX = left[i][0]
        leftY = left[i][1]
        rightX = right[i][0]
        rightY = right[i][1]

        ff = f(a31, a32, leftX, leftY)
        g1 = g(a11, a12, a12, leftX, leftY)
        g2 = g(a21, a22, a23, leftX, leftY)

        dx = rightX - g1 / ff
        dy = rightY - g2 / ff
```

```
        for j in range(5):
            xGradient[j] += dx * partialDiff[j](a11, a12, a13, a31, a32, leftX, leftY)
            yGradient[j] += dy * partialDiff[j](a21, a22, a23, a31, a32, leftX, leftY)
        xGradient *= -2
        yGradient *= -2
        xGradient /= (sigma ** 2)
        yGradient /= (sigma ** 2)

        return np.array(xGradient), np.array(yGradient)
```

#4 Error Function

- 제곱오차를 판별하기 위한 Error Function은 다음과 같이 작성했다.

```
def errors(a11, a12, a13, a21, a22, a23, a31, a32, left, right):
    xError = .0
    yError = .0

    for i in range(len(left)):
        leftX = left[i][0]
        leftY = left[i][1]
        rightX = right[i][0]
        rightY = right[i][1]

        ff = f(a31, a32, leftX, leftY)
        g1 = g(a11, a12, a12, leftX, leftY)
        g2 = g(a21, a22, a23, leftX, leftY)

        dx = rightX - g1 / ff
        dy = rightY - g2 / ff
        xError += dx ** 2
        yError += dy ** 2

    xError /= (sigma ** 2)
    yError /= (sigma ** 2)

    return xError, yError
```

#5 Fitting

- Levenberg Marquardt 알고리즘을 적용해서 a를 구할 수 있다.
- 충분히 작은 lambda가 될 때까지 반복해서 Hessian matrix를 적용해주어 계산하도록 한다.

```
def levenberg_marquardt(a11, a12, a13, a21, a22, a23, a31, a32, left, right):
    mLambda = 1e-3
    cnt = 10

    while cnt > 0:

        while True:
            Hx, Hy = hessian(a11, a12, a13, a21, a22, a23, a31, a32, left, right)
            xGradient, yGradient = gradient(a11, a12, a13, a21, a22, a23, a31, a32,
left, right)
            xError, yError = errors(a11, a12, a13, a21, a22, a23, a31, a32, left,
right)
            HxInverse = np.linalg.inv(Hx + np.identity(5) * mLambda)
            HyInverse = np.linalg.inv(Hy + np.identity(5) * mLambda)


            crossX = np.dot(HxInverse, xGradient)
            crossY = np.dot(HyInverse, yGradient)

            na11 = a11 - crossX[0]
            na12 = a12 - crossX[1]
            na13 = a13 - crossX[2]

            na21 = a21 - crossY[0]
            na22 = a22 - crossY[1]
            na23 = a23 - crossY[2]

            na31 = a31 - (crossX[3] + crossY[3]) / 2
            na32 = a32 - (crossX[4] + crossY[4]) / 2

            nxError, nyError = errors(na11, na12, na13, na21, na22, na23, na31,
na32, left, right)

            if (nxError + nyError >= xError + yError):
                mLambda *= 10
                if mLambda > 1e20:
                    mLambda = 1e20
                    break
            else:
                mLambda /= 10
                a11 = na11
                a12 = na12
                a13 = na13
                a21 = na21
                a22 = na22
                a23 = na23
                a31 = na31
                a32 = na32
                if mLambda < 1e-20:
```

```
                    mLambda = 1e-20
                    break

        cnt -= 1
    return a11, a12, a13, a21, a22, a23, a31, a32
```

#6 Result

- a의 결과는 다음과 같다.
- `a11: 1.0001785729621826 a12: 0.00022585896477780544 a13: 45.5038646940816 a21: -9.217034355599932e-07 a22: 0.9999988338958148 a23: 8.28771897006304 a31: -0.00013203682178126547 a32: -0.00016697566834468082`
- Feature descriptor 정보를 추출하고 right image에 대해서는 left에 우리가 구한 a값을 이용한 식을 적용해서 구해준다.

```
detector = cv2.ORB_create()

kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)

matcher = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

matches = matcher.match(desc1, desc2)

for idx in range(len(matches)):
    x, y = kp1[matches[idx].queryIdx].pt
    kp2[matches[idx].trainIdx].pt = (g(a11, a12, a13, x, y) / f(a31, a32, x, y),
g(a21, a22, a23, x, y) / f(a31, a32, x, y))
```

- 이 결과를 drawMatches를 이용해서 그려본 결과는 다음과 같다.
-