Computer Vision

Assignment 6

Keio University

STEFAN Vlad 82123434
12/06/2022

1 Fundamental matrix estimation

The program begins by importing the necessary libraries and reading both images

```
import cv2 as cv
import numpy as np
import math

# Read the images
img1 = cv.imread('left.jpg', 0)  # queryimage # left image
img2 = cv.imread('right.jpg', 0)  # trainimage # right image
```

Keypoints are identified in both images and matched. The nearest neighbour distance ratio is used to exclude ambiguous matchings.

```
# find the keypoints and descriptors with SIFT
sift = cv.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)

# FLANN parameters
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
flann = cv.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(des1, des2, k=2)
pts1 = []
pts2 = []

# ratio test as per Lowe's paper
for i, (m, n) in enumerate(matches):
    if m.distance < 0.8 * n.distance:
        pts2.append(kp2[m.trainIdx].pt)
        pts1.append(kp1[m.queryIdx].pt)

pts1 = np.int32(pts1)
pts2 = np.int32(pts2)
```

The fundamental matrix is computed. This matrix contains the intrinsic parameters of the camera used and the information characterizing the location of the second camera position relatively to the first one.

```
# Compute the fundamental matrix
F, mask = cv.findFundamentalMat(pts1, pts2, cv.FM_LMEDS)

# We select only inlier points
pts1 = pts1[mask.ravel() == 1]
pts2 = pts2[mask.ravel() == 1]
```

2 Epipolar lines

With the fundamental matrix, the epipolar lines present in each image can be computed. A function is created to draw those lines.

```
def drawlines(img1, img2, lines, pts1, pts2):
    """
    Draw computed epilines
    :param img1: image on which we draw the epilines for the points in img2
    :param img2: image containing the points from which epilines are drawn in img1
    :param lines: corresponding epilines
    :param pts1: points in img1
    :param pts2: points in img2
```

```
    """
    r, c = img1.shape
    img1 = cv.cvtColor(img1, cv.COLOR_GRAY2BGR)
    img2 = cv.cvtColor(img2, cv.COLOR_GRAY2BGR)
    for r, pt1, pt2 in zip(lines, pts1, pts2):
        color = tuple(np.random.randint(0, 255, 3).tolist())
        x0, y0 = map(int, [0, -r[2] / r[1]])
        x1, y1 = map(int, [c, -(r[2] + r[0] * c) / r[1]])
        img1 = cv.line(img1, (x0, y0), (x1, y1), color, 1)
        img1 = cv.circle(img1, tuple(pt1), 5, color, -1)
        img2 = cv.circle(img2, tuple(pt2), 5, color, -1)
    return img1, img2
```

The epipolar lines are then computed and drawn on each image.

```
# Find epilines corresponding to points in right image (second image) and
# drawing its lines on left image
lines1 = cv.computeCorrespondEpilines(pts2.reshape(-1, 1, 2), 2, F)
lines1 = lines1.reshape(-1, 3)
img5, img6 = drawlines(img1, img2, lines1, pts1, pts2)

# Find epilines corresponding to points in left image (first image) and
# drawing its lines on right image
lines2 = cv.computeCorrespondEpilines(pts1.reshape(-1, 1, 2), 1, F)
lines2 = lines2.reshape(-1, 3)
img3, img4 = drawlines(img2, img1, lines2, pts2, pts1)
cv.imwrite("lines_left.jpg", img5)
cv.imwrite("lines_right.jpg", img3)
```

3 Accuracy of the fundamental matrix

To evaluate the accuracy of the fundamental matrix, a function is created to compute the distance between points and epipolar lines.

```
def dist_pt_line(pt, line):
    """
    Compute the distance between a point and a line in 2D space
    :param pt: point coordinates
    :param line: (a, b, c) characterizing line ax + by + c = 0
    :return: distance bewteen the  line and the point
    """
    x, y = pt
    a, b, c = line
    nom = abs(a*x + b*y + c)
    denom = math.sqrt(a**2 + b**2)
    return nom/denom
```

The distance is then computed between corresponding points and lines in the same image, and the average is calculated.

```
# Compute the distance between corresponding points and epipolar lines in img1
distances1 = []
for pt, line in zip(pts1, lines1):
    dist = dist_pt_line(pt, line)
    distances1.append(dist)

# Compute the distance between corresponding points and epipolar lines in img2
distances2 = []
for pt, line in zip(pts2, lines2):
    dist = dist_pt_line(pt, line)
    distances2.append(dist)
```

```
12
13 # Display the average distance in each image
14 print("Average distance between epiline and points in left image: ", sum(distances1)/len(
       distances1))
15 print("Average distance between epiline and points in right image: ", sum(distances2)/len(
       distances2))
```

The previously described code has been adapted from [1][1].

4 Results

Figures 1a and 1b show the same scene photographed from different angles.

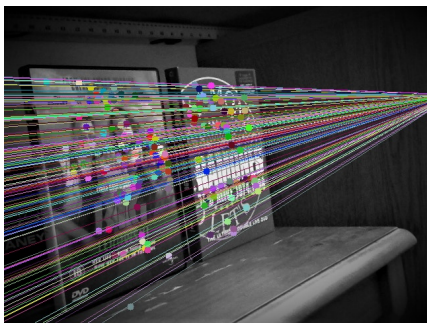
(a) First picture

(b) Second picture

After applying the code described in the previous section on the original pictures, the following fundamental matrix **F** is found:

$$\boldsymbol{F} = \begin{pmatrix} 1.64 \cdot 10^{-6} & 2.54 \cdot 10^{-6} & -1.46 \cdot 10^{-3} \\ 1.63 \cdot 10^{-5} & -2.23 \cdot 10^{-6} & -1.08 \cdot 10^{-2} \\ -2.88 \cdot 10^{-3} & 7.14 \cdot 10^{-3} & 1 \end{pmatrix}$$

With the fundamental matrix **F**, epipolar lines and their correponding points are traced, resulting in figures 2a and 2b.



(a) First picture with epipolar lines

(b) Second picture with epipolar lines

The average distance between a point and its corresponding epipolar line is around 0.479 pixels in figure 2a and 0.497 pixels in figure 2b.

[1]The code and test pictures are available at this link

References

[1] OpenCV. *Epipolar Geometry*. URL: https://docs.opencv.org/4.x/da/de9/tutorial_py_epipolar_geometry.html (visited on 06/06/2022).