COL-780
ASSIGNMENT-2 REPORT

BY

KURISETI RAVI SRI TEJA
2019CS10369

Contents

1 Method Used 2

1.1 Corner Detection 2

1.2 Proximity Matching 2

1.3 Affine Matrix Computation 3

1.4 Stiching 3

2 Instructions to run the code 4

3 Report 4

4 References 4

1 Method Used

My entry number is **2019CS10369** and ends with 69 i.e., Y=1. Hence I used the Hessian corner detector to detect corners.

1.1 Corner Detection

- I initially used a Gaussian smoothing filter to smoothen the image I.
- Using the respective filters, I computed the matrices I_{xx}, I_{yy}, I_{xy} for each image I.
- I then used Gaussian smoothing filter again to smoothen the double derivative matrices I_{xx}, I_{yy}, I_{xy} to get matrices $I'_{xx}, I'_{yy}, I'_{xy}$.
- Using the above matrices, I computed the Hessian matrix

$$H = \begin{pmatrix} I'_{xx} & I'_{xy} \\ I'_{xy} & I'_{yy} \end{pmatrix}$$

- Then I computed the response matrix R=det(H)-0.05×(tr(H))^2
- I selected the pixels where the response matrix value is at least 0.01 times the maximum value of R at all points and marked them as key-points.

1.2 Proximity Matching

- For each pair of consecutive frames (frame-1,frame-2), I matched the key-points by finding closest key-point in frame-2 for every key-point in frame-1.

- Then I computed the mean square distance(msd) between the pixel values of a 5×5 sized grid centred around the points.
- Then I stored all the key-point pairs whose msd was atmost 10 to get a good estimate for affine transform matrix.

1.3 Affine Matrix Computation

- Using the key-point pairs from successive frames, I computed the affine transform matrix between two successive frames by using the least squares method which minimizes the distance $||P_2 - AP_1||_2{}^2$ where P_1, P_2 are points matrices and A is the affine matrix between them.
- Then I multiplied successive affine matrices to get the affine transform of each image frame to first-frame.

1.4 Stiching

- Using affine matrices, I computed the new corner positions of each frame to get the bounding box coordinates and used them to get the largest bounding box that surrounds all the frames.
- I used the cv2.warpAffine() to obtain the transformed frame for each original frame and then used the bitwise_or to stack up the frames on each other.
- Finally I removed the black rectangles (if any) on borders to get the final panorama image.

2 Instructions to run the code

Use the command **python3 Code.py arg1 arg2 arg3** to run the code where arg1 is the name of the final panorama image, arg2 is path to the input directory containing the frames with naming convention as in the datasets and arg3 is the path to output directory where the output image needs to be saved.

3 Report

The results are uploaded at this drive **link**

4 References

1. **Medium Blog on Corner Detectors**

2. **OpenCV Tutorials**