Computer Vision Report

Hemanth N nhemanth2004@gmail.com

- In this task, I implemented a complete camera calibration and distortion correction pipeline using a single checkerboard image.
- The goal was to estimate the intrinsic camera parameters, the distortion coefficients, and to produce an undistorted image of the grid.
- The work was divided into five parts as mentioned in the question :- formulation of a cost function, implementation of robust optimization, RANSAC-based outlier removal, image undistortion, and computation of residuals.

Model formulation:
The image formation process for a pinhole camera model is defined as:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where:

- (X,Y,Z): real-world coordinates
- (u,v): pixel coordinates
- K=[fx,fy,cx,cy]: intrinsic matrix
- R,t: rotation and translation

However, real cameras introduce radial distortion, which causes straight lines to curve, especially near the edges of the image.

I used a 2-parameter radial distortion model:

xd=x(1+k1r^2+k2r^4)
yd=y(1+k1r^2+k2r^4)

where r=√x2+y2
Here k1 and k2 represent the first- and second-order radial distortion coefficients.

Robust Cost Function:

To estimate the parameters, I formulated a cost function that minimizes the reprojection error between the detected corner points in the image and the predicted points from the camera model.

$$E = \sum_i \rho\left(\|p_i - \hat{p}_i(\theta)\|^2 \right)$$

Where:

- pi: detected corner in the distorted image
- pĩ(θ):projected point from model parameters θ=[fx,fy,cx,cy,k1,k2]
- ρ(·): robust loss (Huber loss) used to reduce the effect of outliers

I used Huber loss in the optimization to make the cost less sensitive to noise and misdetected corners.

Robust optimization pipeline:

To refine the distortion parameters, a nonlinear least squares optimization was implemented using `scipy.optimize.least_squares()`.

The pipeline was as follows:

1. Detect feature points:
   Used cv2.findChessboardCorners() to locate the inner corners of the grid.
   Refined them using cv2.cornerSubPix() for subpixel precision.

2. Set up the initial guess:
   Started with approximate values for:
   - fx ,fy=image width
   - cx,cy=image center
   - K1 = k2 = 0
3. Minimize the cost function:
   Used Huber loss to iteratively minimize reprojection error:

```
result = least_squares(residuals, params0, args=(obj_points, corners), loss='huber')
```

The output gave refined parameters for camera intrinsics and distortion coefficients.

4. Refine intrinsics and distortion:
   The optimized values of fx,fy,cx,cy,k1,k2 were then used for undistortion.

RANSAC for Outlier Rejection:

Since some detected corners may be noisy or incorrectly matched,
I implemented a RANSAC (Random Sample Consensus) step before the final optimization.

Algorithm steps:

1. Randomly sample a subset of detected corner points.
2. Estimate parameters using this subset.
3. Compute errors for all points.
4. Mark points with error < threshold (e.g. 2 px) as inliers.
5. Repeat multiple times and select the model with the highest number of inliers.

The final parameter estimation was then refined using only inlier points, ensuring the model was not biased by outliers.

Undistorted the image:

Using the estimated parameters, the distorted image was undistorted as follows:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

The OpenCV function cv2.undistort() was used to warp the image back to an ideal perspective.
This step produced a grid image where the lines were straight and evenly spaced, confirming successful distortion correction.

Additionally, I reprojected the undistorted grid points back onto the distorted image to compute residuals, ensuring the consistency of the model.

Reprojection Error Computation:

The RMS reprojection error was calculated as:

$$\text{RMS} = \sqrt{\frac{1}{N} \sum_i \|p_i - \hat{p}_i\|^2}$$

Where:

- pi: detected corner point (distorted)
- pî: reprojected point from estimated model

A low RMS error (typically below 1–2 pixels) indicates that the camera calibration and distortion parameters fit the image well.

In my results, the RMS reprojection error was low, showing that the model accurately corrected the grid geometry.

<u>Observations:</u>

- The checkerboard corners were successfully detected despite slight noise and uneven lighting.
- RANSAC effectively removed corner outliers.
- The undistorted image had visibly straight grid lines, confirming distortion correction.
- The low RMS error validated the accuracy of the optimization pipeline.

<u>Conclusion:</u>

- The implemented method successfully estimated the camera's radial distortion and intrinsic parameters from a single checkerboard image.
- A robust optimization pipeline using Huber loss and RANSAC produced stable results even in the presence of noise and illumination variation.
- The undistorted image visually confirmed the correction of lens distortion, demonstrating a complete camera calibration and distortion removal workflow.