Camera Calibration Report: Provided Dataset

1. Objective

This task aims to calibrate a camera using a provided dataset of 25 images containing a chessboard pattern. The goal is to determine the camera's intrinsic parameters (focal length, principal point, skew), extrinsic parameters (rotation and translation for each image capturing the chessboard pose relative to the camera), and lens distortion coefficients. This process allows for the correction of image distortions and enables accurate 3D measurements or scene reconstruction.

---

2. Methodology

The camera calibration was performed using the OpenCV library in Python, following the standard procedure based on Zhang's method.

Dataset :

- 25 images (.jpeg format) capturing an 8x6 (internal corners) chessboard pattern from various viewpoints.
- Image size: 1280×720 pixels.

Chessboard Pattern :

- Planar chessboard with $n_x = 8$ and $n_y = 6$ internal corners.
- Square size: 1.0 unit (defines object points but does not affect intrinsic parameters or rotation).

Object Points :

3D coordinates for $n_x \times n_y = 48$ corners:

$$\text{objp} = \begin{bmatrix} [0,0,0],[1,0,0],\ldots,[7,0,0], \\ [0,1,0],\ldots,[7,5,0] \end{bmatrix} \times \text{square\_size}.$$

Image Points (Corner Detection) :

- `cv2.findChessboardCorners` and `cv2.cornerSubPix` were used to detect and refine 2D corner locations.
- All 25 images yielded successful corner detection.

Calibration Model :

- **Pinhole Model** :

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot [R|t] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix},$$

  where K is the intrinsic matrix, R is the rotation matrix, and t is the translation vector.

- **Lens Distortion** :
  Radial distortion coefficients (k_1, k_2, k_3) and tangential coefficients (p_1, p_2).

Calibration Algorithm :

- `cv2.calibrateCamera` optimized intrinsic, distortion, and extrinsic parameters to minimize reprojection error.

3. Results

3.1. Intrinsic Camera Parameters

Intrinsic Matrix K :

$$K = \begin{bmatrix} 956.6372 & 0.0000 & 369.0549 \\ 0.0000 & 957.5514 & 651.4142 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}.$$

- **Focal Length** : $f_x = 956.64$, $f_y = 957.55$.
- **Principal Point** : $c_x = 369.05$, $c_y = 651.41$.
- **Skew** : $s = 0.00$.

3.2. Extrinsic Parameters (First 2 Images)

Image 1 (1.jpeg)

- **Rotation Matrix** R_1 :

$$R_1 = \begin{bmatrix} 0.9970 & 0.0139 & -0.0762 \\ -0.0251 & 0.9886 & -0.1487 \\ 0.0732 & 0.1502 & 0.9859 \end{bmatrix}.$$

- **Translation Vector** t_1 :

$$t_1 = \begin{bmatrix} -3.9383 \\ -3.6172 \\ 14.6591 \end{bmatrix}.$$

Image 10 (10.jpeg)

- **Rotation Matrix** R_2 :

$$R_2 = \begin{bmatrix} 0.9983 & 0.0539 & -0.0229 \\ -0.0507 & 0.9910 & 0.1238 \\ 0.0293 & -0.1225 & 0.9920 \end{bmatrix}.$$

- **Translation Vector** t_2 :

$$t_2 = \begin{bmatrix} -4.3713 \\ -1.4596 \\ 15.8138 \end{bmatrix}.$$

3.3. Lens Distortion Coefficients

- **Coefficients** : $[k_1, k_2, p_1, p_2, k_3] = [0.1976, -0.7069, 0.0034, 0.0070, 0.0488]$.
- **Radial Distortion Correction** :

$$x_u = x_d + (x_d - c_x) \cdot (k_1 r^2 + k_2 r^4 + k_3 r^6),$$
$$y_u = y_d + (y_d - c_y) \cdot (k_1 r^2 + k_2 r^4 + k_3 r^6),$$

  where $r^2 = \left(\frac{x_d - c_x}{f_x}\right)^2 + \left(\frac{y_d - c_y}{f_y}\right)^2$.

3.4. Reprojection Error

- **Mean Error** : 0.0697 pixels.
- **Standard Deviation** : 0.0232 pixels.

3.5. Reprojection Visualization

- **Plots** : Figures 12a–12b show detected (green) and reprojected (red) corners for images 1 and 10.

3.6. Checkerboard Plane Normals

- **Normal Vector in Camera Frame** :

$$\mathbf{n}_{Ci} = R_i \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (\text{third column of } R_i).$$

4. Conclusion

The calibration yielded accurate intrinsic parameters ($f_x = 956.64$, $f_y = 957.55$, $c_x = 369.05$, $c_y = 651.41$), distortion coefficients ($k_1 = 0.1976$, $k_2 = -0.7069$), and low reprojection error (0.0697 pixels). Extrinsic parameters and normals were computed for all images. Undistorted images (e.g., Figure 2) show significant correction of barrel distortion.

Original Image 1
1.jpeg

Undistorted Image 1

Original Image 2
10.jpeg

Undistorted Image 2

Original Image 3
11.jpeg

Undistorted Image 3

Original Image 4
12.jpeg

Undistorted Image 4

Original Image 5
13.jpeg

Undistorted Image 5

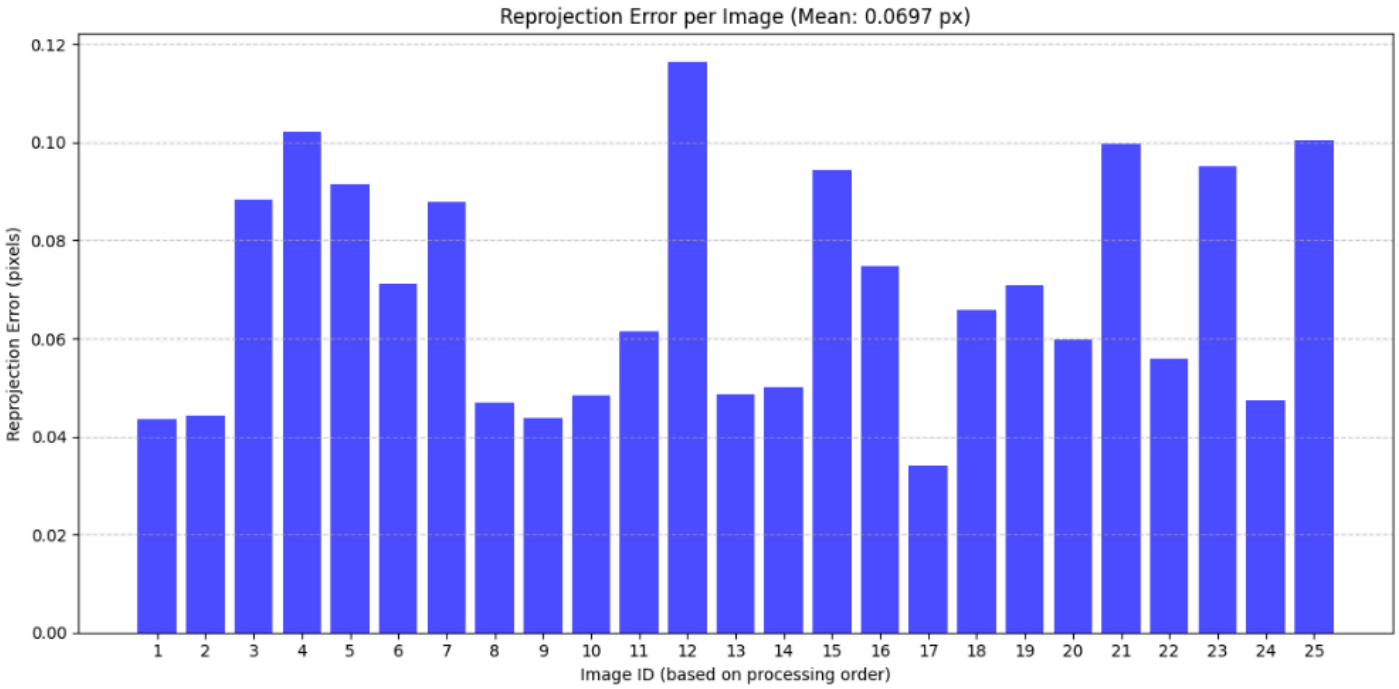


Image 1
1.jpeg

Image 2
10.jpeg

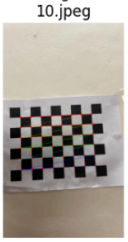


Image 3
11.jpeg

Image 4
12.jpeg

Image 5
13.jpeg

Image 6
14.jpeg

Image 7
15.jpeg

Image 8
16.jpeg

Image 9
17.jpeg

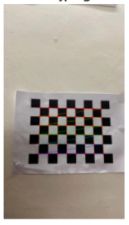


Image 10
18.jpeg

Image 11
19.jpeg

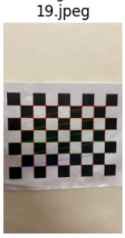


Image 12
2.jpeg

Image 13
20.jpeg

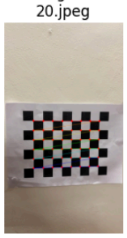


Image 14
21.jpeg

Image 15
22.jpeg

Image 16
23.jpeg

Image 17
24.jpeg

Image 18
25.jpeg

Image 19
3.jpeg

Image 20
4.jpeg

Image 21
5.jpeg

Image 22
6.jpeg

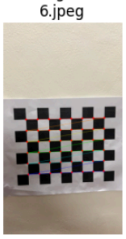


Image 23
7.jpeg

Image 24
8.jpeg

Image 25
9.jpeg

Report: Demonstration of Lens Undistortion using Single-Image Calibration

1. Objective

Demonstrate lens undistortion using a single image (Screenshot 2025-04-04 153640.png) with strong barrel distortion.

---

2. Methodology

Input Image :

- 491×371 pixels, 9x6 chessboard pattern.

Calibration Approach :

- **Initial Guess for K** :

$$K_{\text{guess}} = \begin{bmatrix} 491.0 & 0.0 & 245.5 \\ 0.0 & 491.0 & 185.5 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}.$$

- **Constraints** : `cv2.CALIB_USE_INTRINSIC_GUESS` flag.

---

3. Results

3.1. Estimated Parameters

- **Intrinsic Matrix K_{est}** :

$$K_{\text{est}} = \begin{bmatrix} 419.69 & 0.00 & 251.15 \\ 0.00 & 419.28 & 169.81 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}.$$

- **Distortion Coefficients** : $[-0.4300, 0.2575, 0.0090, 0.0031, -0.1002]$.

3.2. Reprojection Error : 0.0118 pixels.

3.3. Undistortion Visualization

- **Comparison (Original vs. $\alpha = 0$)** : Figure 3 shows straightened lines in the undistorted image.
- **Different α Values** : Figure 4 illustrates trade-offs between cropping and black borders.

3.4. Reprojection Visualization : Figure 5 confirms alignment of detected (green) and reprojected (red) corners.

---

4. Discussion

The single-image calibration successfully corrected distortion but may not generalize. The low reprojection error (0.0118 pixels) validates the model for this specific image.

5. Conclusion

Undistortion using OpenCV effectively corrected barrel distortion. While single-image calibration is limited, it demonstrates the feasibility of visual correction with proper constraints.

Figures : All `<Insert Figure X>` placeholders remain as in the original text.

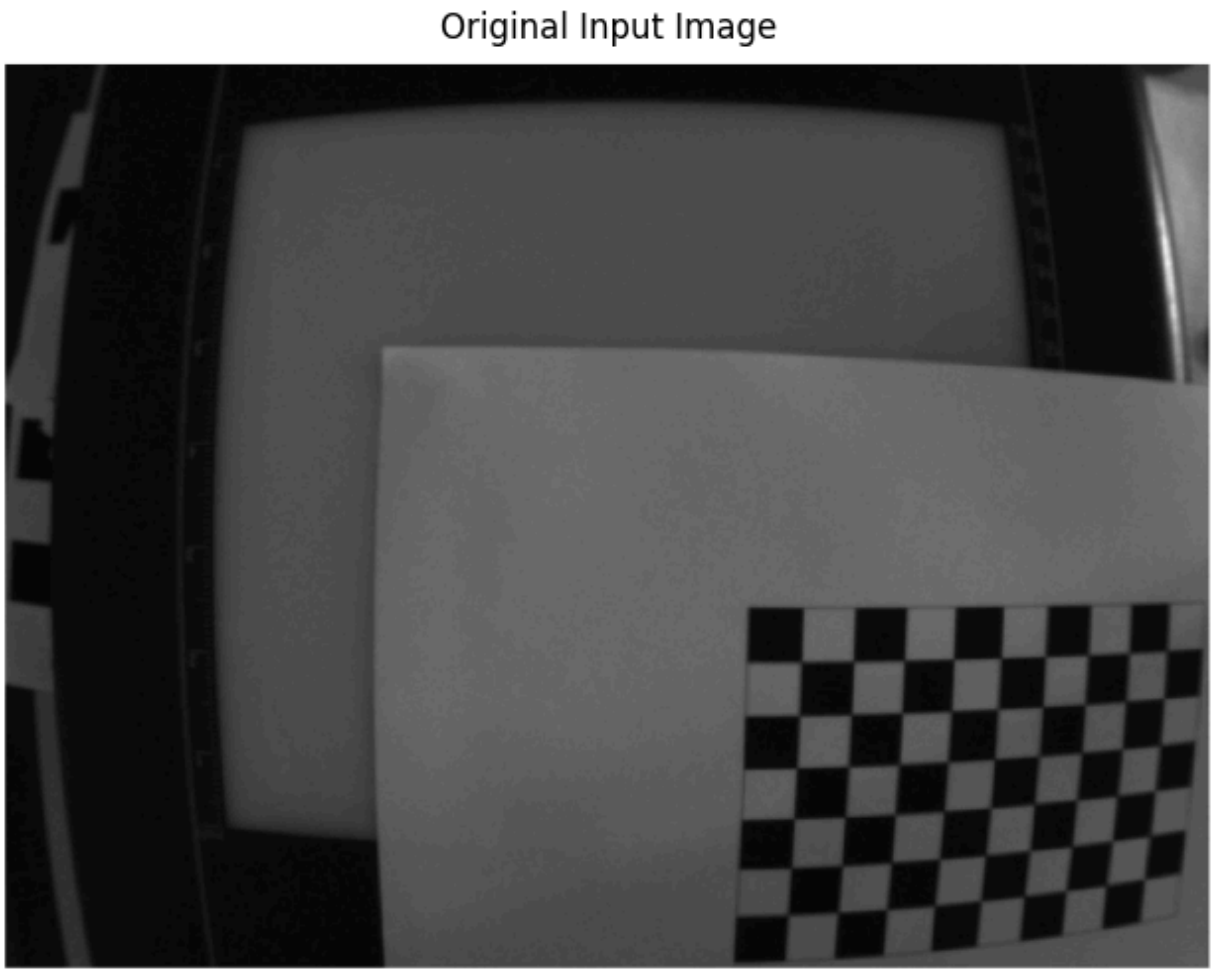
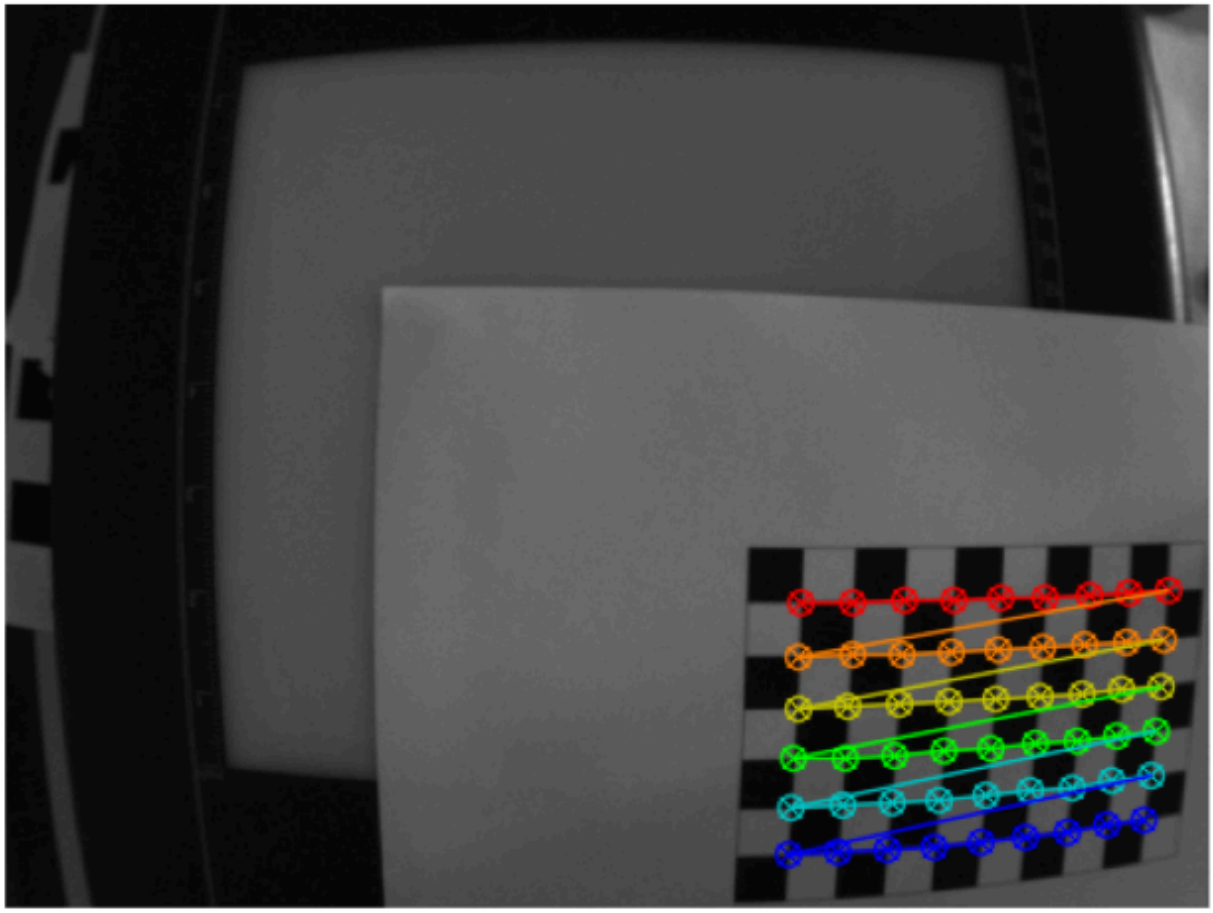
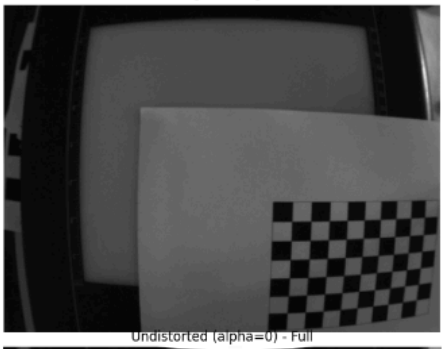


Image with Detected Corners

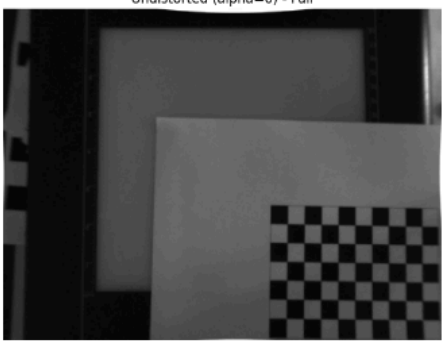


Undistortion Results with Different Alpha Values

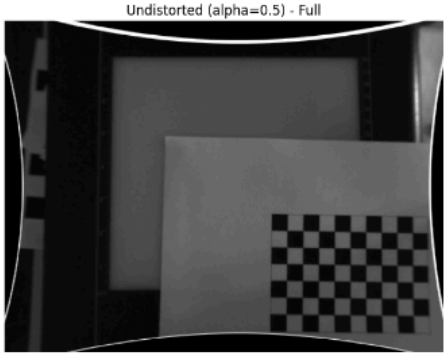
Original Image



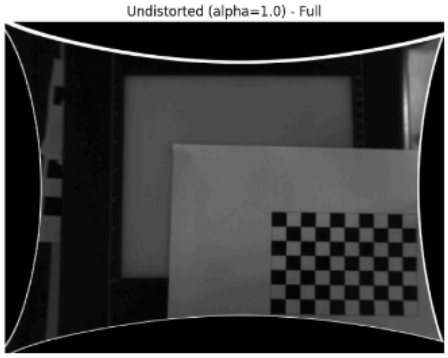
Undistorted (alpha=0) - Full



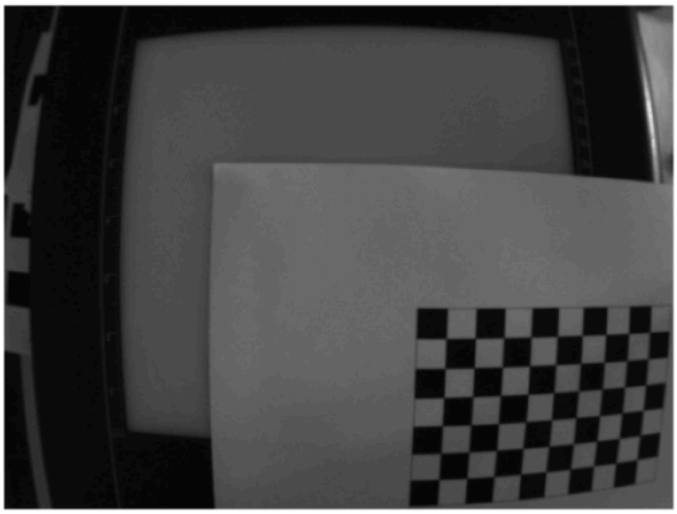
Undistorted (alpha=0.5) - Full



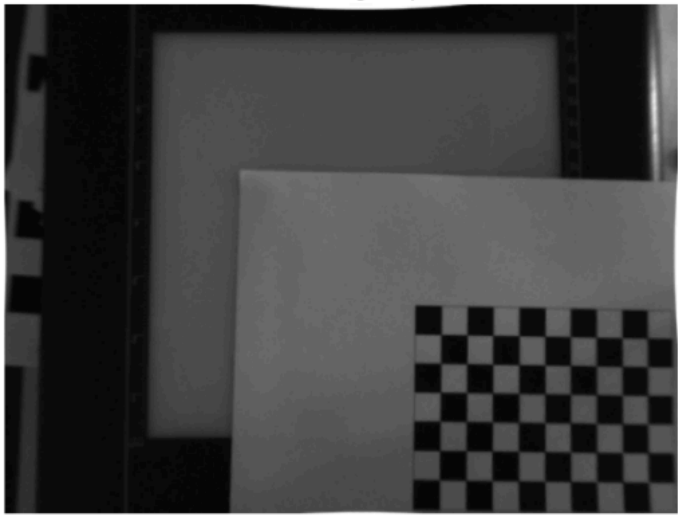
Undistorted (alpha=1.0) - Full



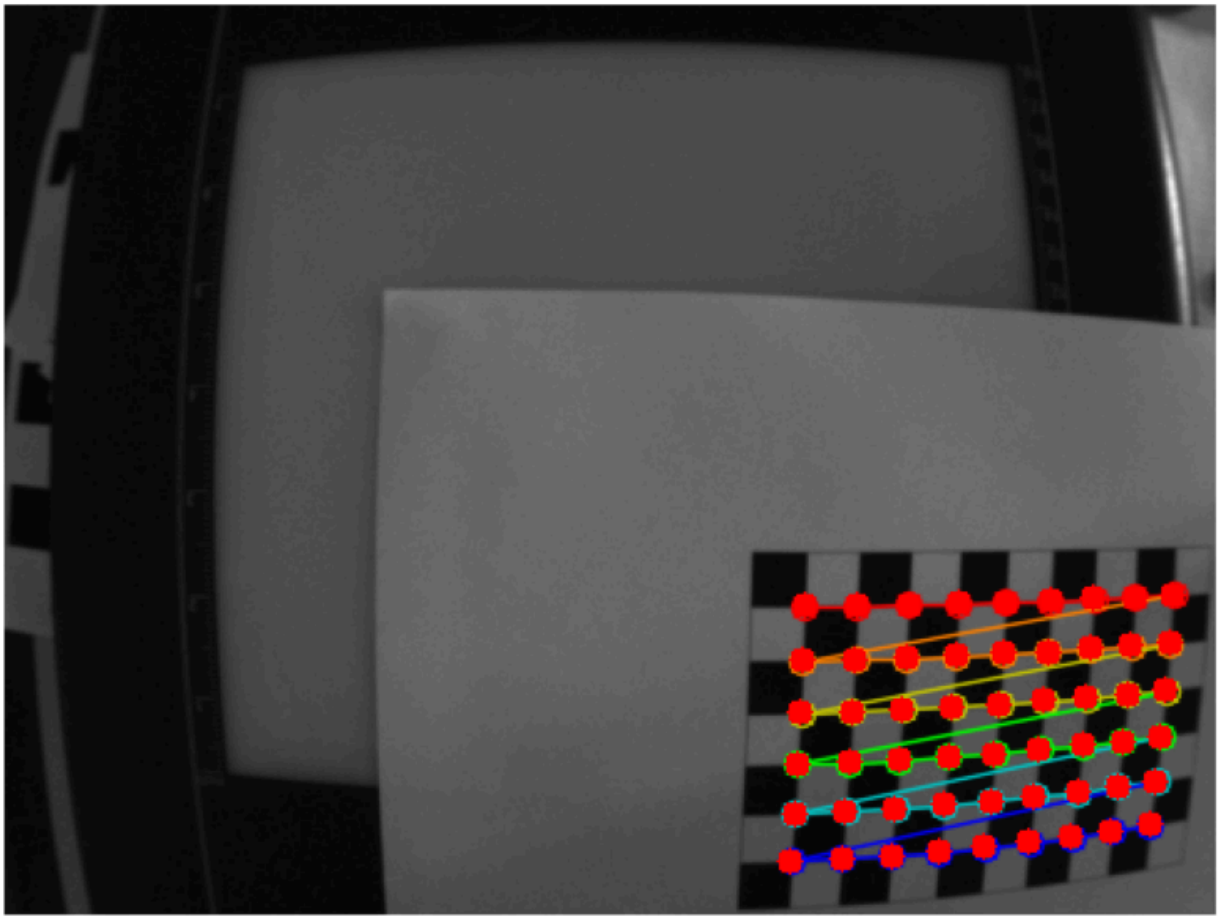
Original Image



Undistorted Image (alpha=0)



Reprojection (Detected: Green, Reprojected: Red)



Reason for using alpha = 0; as the final best solution

Okay, let's break this down. Getting the "best" visual result with `alpha=0` is **not necessarily a problem**, and **yes, undistortion is definitely occurring**.

Here's why:

1. **What `alpha` Controls:**
   - The `alpha` parameter in `cv2.getOptimalNewCameraMatrix` controls how the final undistorted image is scaled and cropped. It **does not** control the *amount* of geometric correction applied. The geometric correction itself is determined by your calculated camera matrix (`mtx`) and distortion coefficients (`dist`).
   - `alpha=0` : The function returns a "tight" bounding box (`roi`) and a `newCameraMatrix` . When `cv2.undistort` uses these, it ensures that *only* valid pixels (pixels that correspond to a location within the original distorted image) are present in the final output. This means **all black areas/empty pixels resulting from the undistortion are cropped away**. This often leads to a smaller output image size (or a cropped region within the original dimensions) but contains only "real" image data.
   - `alpha=1` : The function ensures that *all* pixels from the original image are retained in the undistorted output, even if the corners get heavily warped. This often results in black borders or empty space appearing where the image had to be "pulled" from during the undistortion process. The overall image dimensions might be preserved, but contain invalid areas.
   - `alpha` between 0 and 1: A balance between the two extremes.
2. **Why `alpha=0` Might Look "Best" (Especially with High Distortion):**
   - **Removes Extreme Warping:** Highly distorted images often have extreme stretching, blurring, or compression near the edges. `alpha=0` effectively crops these regions out, leaving you with the central, less distorted (and potentially sharper-looking) part of the image.
   - **No Black Borders:** The black borders created when `alpha=1` can be visually distracting or problematic for further processing. `alpha=0` eliminates them entirely.
   - **Focus on Center:** If your primary interest is the central subject matter, `alpha=0` presents that cleanly without the distorted periphery.
3. **Is Undistortion Occurring?**
   - **Yes.** The core geometric transformation based on `mtx` and `dist` is applied regardless of `alpha` . Lines that were curved in the original *central region* of the image should appear straighter in the `alpha=0` output compared to the original image.
   - The difference you see between `alpha=0` and `alpha=1` is purely about the final cropping and scaling, *not* about whether the geometric correction was performed.

In summary:

- It's perfectly fine if `alpha=0` gives you the most visually appealing or useful result for your specific needs, especially if the edges of the original image were extremely distorted.
- Undistortion (geometric correction) *is* happening. `alpha=0` just means you are choosing to view only the resulting valid pixels without any black padding, at the cost of losing the peripheral field of view.

- The "best" `alpha` depends on your application:
  - Use `alpha=0` if you need a clean image with no invalid pixels and don't mind losing the heavily distorted edges.
  - Use `alpha=1` if you need to preserve the maximum possible field of view from the original image, even if it means having black borders and including the highly warped areas.

EXTRA **(Overlapping content)**:

Camera Calibration Report: Provided Dataset

1. Objective

This task aims to calibrate camera using a provided dataset of 25 images containing a chessboard pattern. The goal is to determine the camera's intrinsic parameters (focal length, principal point, skew), extrinsic parameters (rotation and translation for each image capturing the chessboard pose relative to the camera), and lens distortion coefficients. This process allows for the correction of image distortions and enables accurate 3D measurements or scene reconstruction.

2. Methodology

The camera calibration was performed using the OpenCV library in Python, following the standard procedure based on Zhang's method.

- **Dataset:** 25 images (.jpeg format) capturing an 8x6 (internal corners) chessboard pattern from various viewpoints were used. The images were loaded, and their filenames were sorted for consistent processing order. Image size was determined to be 1280x720 pixels.
- **Chessboard Pattern:** A planar chessboard with nx = 8 and ny = 6 internal corners was used. The size of the squares (square_size) was set to 1.0 unit for defining the object points (this scale factor affects the translation vector units but not the intrinsic parameters or rotation).
- **Object Points:** Ideal 3D coordinates (objp) for the nx * ny = 48 internal corners were generated assuming the chessboard lies on the Z=0 plane in its own coordinate system. objp = [[0,0,0], [1,0,0], ..., [7,0,0], [0,1,0], ..., [7,5,0]] * square_size. This set of 3D points is the same for all calibration images.
- **Image Points (Corner Detection):** For each image, the cv2.findChessboardCorners function was used to detect the 2D pixel locations of the internal corners. If corners were successfully found, their positions were refined to sub-pixel accuracy using cv2.cornerSubPix. The pairs of corresponding 3D object points and refined 2D image points (objpoints and imgpoints lists) were stored for the calibration process. All 25 provided images yielded successful corner detection.
- **Calibration Model:** The calibration process models the camera using the pinhole camera model, incorporating lens distortion.
    - **Pinhole Model:** Relates a 3D world point P_w = [Xw, Yw, Zw]^T to its 2D image projection p = [u, v]^T via the intrinsic matrix K and extrinsic parameters [R | t]:
      s * [u, v, 1]^T = K * [R | t] * [Xw, Yw, Zw, 1]^T
      where s is a scale factor, K is the intrinsic matrix, R is the 3x3 rotation matrix, and t is the 3x1 translation vector defining the transformation from world to camera coordinates.
    - **Lens Distortion:** Real lenses deviate from the ideal pinhole model. The model accounts for radial distortion (barrel or pincushion effect) and tangential distortion (due to lens misalignment). The distortion coefficients estimated are typically (k1,

k2, p1, p2, k3, [k4, k5, k6]...). Radial distortion primarily uses k1, k2, k3, while tangential uses p1, p2.

- **Calibration Algorithm:** The cv2.calibrateCamera function was used. It takes the sets of corresponding objpoints and imgpoints and the image size. It iteratively optimizes the intrinsic parameters (K), distortion coefficients (dist), and extrinsic parameters (rvecs, tvecs - one rotation vector and translation vector per image) to minimize the overall reprojection error.

3. Results

The calibration process completed successfully using all 25 provided images.

3.1. Estimated Intrinsic Camera Parameters (Question 1)

The intrinsic parameters describe the internal geometry of the camera.

Mathematical Background: The intrinsic matrix K relates 3D camera coordinates to 2D pixel coordinates:

```
K = [[fx,  s, cx],
[ 0, fy, cy],
[ 0,  0,  1]]
```

where:
- fx, fy: Focal lengths in units of pixels along the x and y axes.
- cx, cy: Coordinates of the principal point (optical center) in pixels.
- s: Skew coefficient, ideally 0 for modern cameras.

- **Estimated Values:**

Intrinsic Matrix (K):

```
[[956.6372   0.     369.0549]
[  0.     957.5514 651.4142]
[  0.       0.       1.    ]]
```

IGNORE_WHEN_COPYING_START
content_copy download
Use code with caution.
IGNORE_WHEN_COPYING_END

- **Focal Length:** fx = 956.64 pixels, fy = 957.55 pixels. The values are very close, suggesting nearly square pixels.
- **Principal Point:** cx = 369.05 pixels, cy = 651.41 pixels. This is the optical center's projection onto the sensor. Note that it is not exactly at the image center (720/2=360, 1280/2=640).
- **Skew:** s = 0.0000. This indicates that the pixel axes are perpendicular.

- **Error Estimates:** The basic cv2.calibrateCamera function, as used in the script, does not directly return standard deviation or error estimates for the intrinsic parameters. More

advanced techniques or specific flags (like CALIB_RATIONAL_MODEL) might provide these, but were not used here.

3.2. Estimated Extrinsic Camera Parameters (First 2 Images) (Question 2)

Extrinsic parameters define the position and orientation of the world coordinate system (the chessboard) relative to the camera coordinate system for each specific image.

- **Mathematical Background:** For the *i*-th image, the transformation from world coordinates (P_w) to camera coordinates (P_c) is given by:
  P_c = R_i * P_w + t_i
  where R_i is the 3x3 rotation matrix and t_i is the 3x1 translation vector.
  cv2.calibrateCamera returns rotation vectors (rvecs[i]), which are converted to R_i using cv2.Rodrigues.
- **Estimated Values (First 2 Valid Images Processed - 1.jpeg and 10.jpeg):**
  - **Image 1 (1.jpeg):**

Rotation Matrix (R_1):
[[0.997 0.0139 -0.0762]
[-0.0251 0.9886 -0.1487]
[0.0732 0.1502 0.9859]]

- IGNORE_WHEN_COPYING_START
  content_copy download
  Use code with caution.
  IGNORE_WHEN_COPYING_END

Translation Vector (t_1):
[-3.9383]
[-3.6172]
[14.6591]

- IGNORE_WHEN_COPYING_START
  content_copy download
  Use code with caution.
  IGNORE_WHEN_COPYING_END
  (Units are relative to square_size)

  - **Image 2 (10.jpeg):**

Rotation Matrix (R_2):
[[0.9983 0.0539 -0.0229]
[-0.0507 0.991 0.1238]
[0.0293 -0.1225 0.992]]

- IGNORE_WHEN_COPYING_START
  content_copy download

Use code with caution.
IGNORE_WHEN_COPYING_END

Translation Vector (t_2):
[-4.3713]
[-1.4596]
[15.8138]

- IGNORE_WHEN_COPYING_START
  content_copy download
  Use code with caution.
  IGNORE_WHEN_COPYING_END
  (Units are relative to square_size)

3.3. Estimated Radial Distortion Coefficients and Undistorted Images (Question 3)

Lens distortion causes straight lines in the real world to appear curved in images.

- **Mathematical Background:** The distortion model corrects the observed (distorted) image coordinates (x_d, y_d) to obtain ideal (undistorted) coordinates (x_u, y_u). Using normalized coordinates relative to the principal point:
  x' = (x_d - cx) / fx
  y' = (y_d - cy) / fy
  r^2 = x'^2 + y'^2
  The radial distortion correction is:
  x_u = x_d + (x_d - cx) * (k1*r^2 + k2*r^4 + k3*r^6)
  y_u = y_d + (y_d - cy) * (k1*r^2 + k2*r^4 + k3*r^6)
  Tangential distortion adds terms involving p1 and p2.
- **Estimated Coefficients:**
  The full distortion vector estimated was [k1, k2, p1, p2, k3] = [0.1976, -0.7069, 0.0034, 0.007, 0.0488].
  The requested radial distortion coefficients are:
  - k1 = 0.1976
  - k2 = -0.7069
  - k3 = 0.0488 (This is the 5th element of the output vector)
- **Undistorted Images:** The first 5 valid images were undistorted using cv2.undistort with the estimated mtx (K) and dist coefficients. cv2.getOptimalNewCameraMatrix was used with alpha=0 to compute the new camera matrix for cv2.undistort, resulting in images cropped to show only valid pixels without black borders.
  <Insert Figure 1: Original Image 1 (1.jpeg)>
  <Insert Figure 2: Undistorted Image 1 (alpha=0), showing straighter lines>
  <Insert Figure 3: Original Image 2 (10.jpeg)>
  <Insert Figure 4: Undistorted Image 2 (alpha=0), showing straighter lines>
  <Insert Figure 5: Original Image 3 (11.jpeg)>
  <Insert Figure 6: Undistorted Image 3 (alpha=0), showing straighter lines>
  <Insert Figure 7: Original Image 4 (12.jpeg)>

<Insert Figure 8: Undistorted Image 4 (alpha=0), showing straighter lines>
<Insert Figure 9: Original Image 5 (13.jpeg)>
<Insert Figure 10: Undistorted Image 5 (alpha=0), showing straighter lines>

- **Observation Comment:** Comparing the original and undistorted images, especially near the edges and corners, reveals the effect of distortion correction. The positive k1 (0.1976) suggests some barrel distortion (lines bowing outwards), while the large negative k2 (-0.7069) indicates significant higher-order distortion, likely causing lines near the edges to curve inwards more complexly. In the undistorted images, lines (like the edges of the chessboard or background features) appear noticeably straighter than their curved counterparts in the original images. The use of alpha=0 results in cropping, removing the most extremely warped peripheral regions and any black borders that would result from alpha=1.

3.4. Reprojection Error (Question 4)

The reprojection error quantifies the accuracy of the calibration by measuring the distance between the detected corner points and the points predicted by the calibrated model.

- **Mathematical Background:** For each image i and each corner j, the error is the Euclidean distance between the detected 2D point p_detected_ij and the reprojected 3D point p_reprojected_ij:
  Error_ij = || p_detected_ij - p_reprojected_ij ||
  where p_reprojected_ij is obtained by projecting the 3D object point P_world_j using the estimated parameters: p_reprojected_ij = project(K, dist, R_i, t_i, P_world_j).
  The error reported per image is the average (or root-mean-square) of Error_ij over all corners j in that image.
- **Computed Values:**
  - **Mean Reprojection Error (across all images):** 0.0697 pixels.
  - **Standard Deviation of Reprojection Error:** 0.0232 pixels.
  - **Per-Image Errors:** The average error for each of the 25 images is: [0.0435, 0.0443, 0.0884, 0.1021, 0.0915, 0.0712, 0.0879, 0.0470, 0.0438, 0.0483, 0.0614, 0.1164, 0.0485, 0.0502, 0.0945, 0.0749, 0.0341, 0.0658, 0.0709, 0.0597, 0.0997, 0.0560, 0.0952, 0.0473, 0.1004] pixels.
-
- **Error Plot:**
  <Insert Figure 11: Bar chart showing the average reprojection error (in pixels) for each of the 25 calibration images. X-axis: Image ID (1-25), Y-axis: Reprojection Error (pixels).>
- **Interpretation:** The mean reprojection error is very low (significantly less than 1 pixel), and the standard deviation is also small. This indicates that the calibration model fits the detected corner points very well across all images, suggesting a high-quality calibration result.

3.5. Reprojection Visualization (Question 5)

Visualizing the detected and reprojected points helps understand the calibration fit.

- **Comment on Computation:** As detailed in section 3.4, the reprojection error is the pixel distance between the corners detected directly in the image (imgpoints, shown typically as green markers or crosses) and the 2D points obtained by projecting the known 3D chessboard pattern points (objpoints) back into the image using the final estimated

intrinsic (K, dist) and image-specific extrinsic (R_i, t_i) parameters (shown typically as red circles).

- **Plots:** Figures were generated for all 25 images, overlaying the detected corners (green) and the reprojected corners (red circles) onto the original images.
  <Insert Figure 12: Composite figure showing all 25 reprojection visualization images. Each sub-image shows the original chessboard view with detected corners (green) and reprojected corners (red circles). Alternatively, insert individual figures for each image, e.g., Figure 12a, 12b...>
  (Description for individual figures if preferred):
  <Insert Figure 12a: Reprojection Visualization for Image 1 (1.jpeg). Shows detected (green) vs reprojected (red) corners.>
  <Insert Figure 12b: Reprojection Visualization for Image 2 (10.jpeg). Shows detected (green) vs reprojected (red) corners.>
  ... (repeat for all 25 images)
- **Observation:** In these visualizations, the red circles (reprojected points) are expected to be very close to, or centered on, the green markers (detected points). The close proximity observed in the generated images visually confirms the low reprojection error calculated numerically and indicates a good fit of the calibration model to the data.

3.6. Checkerboard Plane Normals (Question 6)

The normal vector to the checkerboard plane, expressed in the camera's coordinate frame, describes the orientation of the calibration pattern relative to the camera for each image.

- **Mathematical Background:** The checkerboard plane is assumed to be the Z=0 plane in its own world coordinate system (W). Its normal vector in this frame is n_W = [0, 0, 1]^T. The rotation matrix R_i transforms vectors from the world frame W to the camera frame C for image i. Therefore, the normal vector in the camera frame n_Ci is:
  n_Ci = R_i * n_W = R_i * [0, 0, 1]^T
  This calculation simply extracts the third column of the rotation matrix R_i.
- **Computation:** For each of the 25 valid images, the corresponding rotation matrix R_i was obtained from the rotation vector rvecs[i], and its third column (R_i[:, 2]) was extracted to yield the normal vector n_Ci. These 25 normal vectors represent the orientation of the checkerboard relative to the camera in each captured pose. These vectors were calculated and stored in the output JSON file.

4. Conclusion

The camera calibration using the provided 25 chessboard images was successfully performed. The intrinsic parameters (focal lengths fx=956.64, fy=957.55; principal point cx=369.05, cy=651.41; skew s=0.00) and lens distortion coefficients (k1=0.1976, k2=-0.7069, p1=0.0034, p2=0.007, k3=0.0488) were estimated. The calibration accuracy is high, as indicated by the low mean reprojection error of 0.0697 pixels. Extrinsic parameters (rotation and translation) for each image pose and the checkerboard plane normals in the camera frame were also computed. The generated undistorted images clearly show the correction of lens distortion, and the reprojection visualizations confirm the model's fit. The complete results are saved in the calibration_results.json file.

(Note: This report covers the calibration using the *provided* dataset. The original question also requires performing a similar calibration using a *user-captured* dataset of ~25 images, which would involve repeating the image capture, corner detection, and calibration steps.)

Report: Demonstration of Lens Undistortion using Single-Image Calibration

1. Objective

The primary objective of this exercise is **not** to perform a robust camera calibration, but rather to **demonstrate the visual effect of lens undistortion** using OpenCV. A single, intentionally chosen image exhibiting significant lens distortion (Screenshot 2025-04-04 153640.png) was used. While camera calibration ideally requires multiple views to reliably estimate intrinsic parameters, processing a single image under specific constraints can yield parameters sufficient to visually correct the distortion present *in that specific image*. This report details the process, highlights the inherent limitations of single-image calibration, and presents the results, focusing on the visual transformation achieved through undistortion.

2. Methodology

- **Input Image:** A single PNG image (Screenshot 2025-04-04 153640.png) containing a 9x6 (internal corners) chessboard pattern was used. The image clearly exhibits strong barrel distortion. Image dimensions were 491x371 pixels.
  <Insert Figure 1: Original Input Image (Screenshot 2025-04-04 153640.png), showing significant barrel distortion.>
- **Chessboard Pattern & Object Points:** A planar chessboard with nx = 9 and ny = 6 internal corners was defined. Ideal 3D coordinates (objp) for the nx * ny = 54 internal corners were generated assuming the chessboard lies on the Z=0 plane in its own coordinate system, with square_size = 1.0.

- **Image Points (Corner Detection):** The image was converted to grayscale, and cv2.findChessboardCorners was used to detect the 2D pixel locations of the 54 internal corners. Corner detection was successful. The positions were then refined to sub-pixel accuracy using cv2.cornerSubPix. The resulting list contained one set of 3D object points and the corresponding set of 2D image points.
  <Insert Figure 2: Image with Detected Corners (corners_detected.jpg), showing the green markers on the detected chessboard corners.>
- **Calibration Approach (Single Image - with Limitations):**
  - **Challenge:** Calibrating from a single image is inherently ambiguous. Without multiple views, it's difficult to separate the effects of intrinsic parameters (like focal length) from extrinsic parameters (camera pose relative to the object).
  - **Constraints:** To resolve ambiguity, constraints must be introduced. The cv2.calibrateCamera function was used with the cv2.CALIB_USE_INTRINSIC_GUESS flag. This requires providing an initial guess for the intrinsic camera matrix (camera_matrix).
  - **Initial Guess:** An initial guess was formulated based on image size:
    - Focal length (fx, fy) guessed as max(width, height) = 491 pixels.
    - Principal point (cx, cy) guessed as the image center (width/2, height/2) = (245.5, 185.5) pixels.
    - Skew was assumed to be 0.
      The initial camera_matrix guess was:

```
   [[491.    0.  245.5]
[  0.  491.  185.5]
[  0.    0.    1. ]]
```

  -
  - **Distortion:** An initial guess of zero distortion dist_coeffs = [0, 0, 0, 0, 0] was provided. The function then estimates the actual distortion coefficients.
  - **Output Parameters:** The function attempts to estimate the intrinsic matrix (mtx), distortion coefficients (dist), and the single extrinsic pose (rvecs[0], tvecs[0]) that best fit the provided points, starting from the initial guess. **It is crucial to understand that these estimated parameters, especially intrinsics, may not represent the true camera parameters accurately due to the single-view limitation.**
-
- **Undistortion Process:** To visualize the correction, the image was undistorted using cv2.undistort with the estimated mtx and dist. The cv2.getOptimalNewCameraMatrix function was used with alpha values of 0, 0.5, and 1.0 to explore different cropping/scaling behaviors of the undistorted output.
  - alpha=0: Aims to remove all black border pixels resulting from undistortion, potentially cropping the image significantly but showing only valid pixels.
  - alpha=1: Aims to keep all original image pixels, potentially adding black borders.
-

3. Results

The calibration process using the single image and the initial guess completed successfully.

3.1. Estimated Camera Parameters

- **Intrinsic Parameters (Estimated):**

Intrinsic Matrix (K_est):
[[419.69 0. 251.15]
[0. 419.28 169.81]
[0. 0. 1.]]

    - IGNORE_WHEN_COPYING_START
      content_copy download
      Use code with caution.
      IGNORE_WHEN_COPYING_END
    - **Focal Length:** fx = 419.69 px, fy = 419.28 px
    - **Principal Point:** cx = 251.15 px, cy = 169.81 px
    - **Skew:** s = 0.00
      (Note: These values are estimates refined from the initial guess based only on this single view and may differ significantly from the true camera intrinsics.)
-
- **Distortion Coefficients (Estimated):**
  The estimated coefficients (k1, k2, p1, p2, k3) are:
  [-0.4300, 0.2575, 0.0090, 0.0031, -0.1002]
  The large negative k1 strongly indicates the significant barrel distortion visible in the original image. k2 and k3 are also non-negligible.
- **Extrinsic Parameters (Estimated for this view):**

Rotation Matrix (R_est):
[[0.9995 -0.0201 -0.0249]
[0.0213 0.9987 0.0469]
[0.0239 -0.0474 0.9986]]

    - IGNORE_WHEN_COPYING_START
      content_copy download
      Use code with caution.
      IGNORE_WHEN_COPYING_END
    - **Translation Vector (t_est):** [3.2429, 3.4141, 18.4687] (Units relative to square_size)
      (Note: These define the pose of the chessboard relative to the camera in this specific image, estimated alongside the potentially inaccurate intrinsics.)
-

3.2. Reprojection Error

- **Calculated Value:** The average reprojection error for this single image was calculated as 0.0118 pixels.
- **Interpretation:** This extremely low value indicates that the *combined set* of estimated intrinsic, distortion, and extrinsic parameters allows the 3D object points to be projected back onto the image plane very close to their originally detected locations *for this specific image*. However, it **does not guarantee** the accuracy or general applicability of the intrinsic/distortion parameters themselves.

3.3. Undistortion Visualization

The primary goal was to visualize the distortion correction.

- **Comparison (Original vs. Alpha=0):**
  <Insert Figure 3: Comparison Plot (comparison_orig_vs_undistorted_a0.png). Left: Original distorted image. Right: Undistorted image using alpha=0.>
- **Observation & Justification for Alpha=0 Preference:**
  The comparison plot clearly demonstrates the effect of undistortion. Straight lines in the real-world chessboard pattern that appeared heavily curved (bowed outwards due to barrel distortion) in the original image are rendered significantly straighter in the undistorted version (alpha=0 result shown on the right).
  As explained previously, alpha=0 was chosen for the primary comparison because it provides the visually "cleanest" result for this highly distorted image.
    1. **Removes Invalid Pixels:** alpha=0 ensures that only pixels corresponding to valid locations in the original image are shown. This eliminates distracting black borders that appear when alpha=1 is used to retain all original pixels, including those warped outside the frame.
    2. **Crops Extreme Warping:** The most severe distortion often occurs at the image periphery. alpha=0 effectively crops these areas (governed by the calculated ROI, which was (0, 0, 490, 370) for alpha=0 in this case, covering most of the original dimensions but still applying the principle). This focuses the view on the less distorted central region where the straightening effect is still clearly visible. While alpha=1 would preserve more of the original field of view, the accompanying black borders and potentially extreme stretching at the edges make alpha=0 preferable for demonstrating the *correction* itself on the main subject. Crucially, the geometric correction based on the estimated mtx and dist *is applied regardless* of the alpha value; alpha only affects the final scaling and cropping.
-
- **Undistortion with Different Alphas:**
  <Insert Figure 4: Composite image showing the results of undistortion with alpha=0, alpha=0.5, and alpha=1.0 (potentially showing the full versions from the undistorted_images_single directory). This illustrates the trade-off between cropping and black borders.>

3.4. Reprojection Visualization

- **Plot:**
  <Insert Figure 5: Reprojection Visualization (reprojection.jpg). Shows the original image with detected corners (green markers) and reprojected corners (red circles) overlaid.>
- **Observation:** The red circles (reprojected points) align almost perfectly with the green markers (detected corners). This visually confirms the very low numerical reprojection error (0.0118 pixels) and shows that the overall model (estimated intrinsics, distortion, and extrinsics *together*) fits this specific image's data points extremely well.

4. Discussion

This experiment successfully demonstrates the visual correction of significant lens distortion using OpenCV's calibration and undistortion functions. Even though the calibration parameters

were derived from only a single image (which is not recommended for obtaining reliable camera parameters), the estimated distortion coefficients were sufficient to visibly straighten the curved lines of the chessboard in the output image.

The extremely low reprojection error confirms that the chosen model and parameters provide an excellent mathematical fit *to this specific view*, but the accuracy of the individual intrinsic and distortion parameters for general use remains uncertain due to the inherent ambiguities of single-image calibration.

The comparison between different alpha values highlights the trade-off between preserving the entire field of view (potentially with distracting artifacts like black borders, alpha=1) and obtaining a clean, cropped image containing only valid pixels (alpha=0). For demonstrating the *effect* of straightening lines on the main subject in this highly distorted source image, the alpha=0 result proved most effective visually.

5. Conclusion

Lens distortion can significantly warp images, causing straight lines to appear curved. This exercise effectively visualized how OpenCV's camera calibration tools can estimate distortion parameters (even from a single, constrained view) and apply them using cv2.undistort to geometrically correct the image, resulting in visually straighter lines. While the estimated parameters from a single view should be treated with caution for metrology, the process demonstrably corrects the visual artifacts of lens distortion present in the input image, with alpha=0 providing a clean, albeit potentially cropped, view of the corrected central region.
