Image Calibration and Warping between Cameras for Autonomous Driving

Myra Cropper
Department of Computer Science
The College of William & Mary
email: mpcropper@wm.edu

***Abstract*—This report describes the process of image calibration and warping between two cameras in the Canadian Adverse Driving Conditions (CADC) dataset. We present the camera intrinsics, distortion coefficients, extrinsic transformations, and the process of undistortion and warping to align images from two cameras using OpenCV.**

***Index Terms*—Camera Calibration, Image Warping, Autonomous Driving, CADC Dataset, OpenCV.**

I. Introduction

Image calibration and warping between multiple cameras are essential in autonomous driving for accurate sensor fusion and scene understanding. The CADC dataset [1] provides multi-camera data, where precise alignment between cameras is necessary for downstream tasks such as depth estimation and object detection. This report focuses on transforming images from camera to camera using camera calibration matrices, distortion coefficients, and extrinsic transformations. We use cameras 5 and 6 for our experimentation.

II. Camera Intrinsics

The intrinsic parameters of a camera describe the relationship between the 3D points in the real world and their 2D projection on the image plane.

A. Camera 5

The intrinsic matrix for Camera 5 is:

$$K_5 = \begin{bmatrix} 657.473 & -0.413 & 660.315 \\ 0 & 659.829 & 513.577 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The intrinsic matrix contains:

- f_x = 657.473: The focal length along the x-axis.
- f_y = 659.828: The focal length along the y-axis.
- c_x =660.315: The principal point (image center) along the x-axis.
- c_y =513.577: The principal point (image center) along the y-axis.
- $[-0.413]$: The skew value, indicating how much the cameras axes are not perfectly perpendicular.

The distortion coefficients are:

$$\text{dist}_5 = [-0.20727, 0.09874, -0.000097, 0.000475, -0.02300] \tag{2}$$

This consists of radial distortions: k_1, k_2, k_3 and tangential distortions: p_1, p_2

B. Camera 6

The intrinsic matrix for Camera 6 is:

$$K_6 = \begin{bmatrix} 662.426 & 0 & 645.692 \\ 0 & 663.459 & 519.758 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The intrinsic matrix contains:

- f_x = 662.426: The focal length along the x-axis.
- f_y = 663.459: The focal length along the y-axis.
- c_x = 645.692: The principal point (image center) along the x-axis.
- c_y = 519.758: The principal point (image center) along the y-axis.

The distortion coefficients are:

$$\text{dist}_6 = [-0.17225, 0.04636, 0.00025, -0.00145] \tag{4}$$

III. Extrinsic Transformation

The transformation from Camera 5 to Camera 6 is given by the following matrix:

$$\begin{bmatrix} 0.5920 & 0.0085 & 0.8059 & 0.3954 \\ -0.0105 & 0.9999 & -0.0028 & 0.0042 \\ -0.8059 & -0.0068 & 0.5921 & -0.2559 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

This matrix consists of a rotation matrix R and a translation vector t.

The top-left $3x3$ submatrix is the rotation matrix (R).

The first three elements of the last column represent the translation vector (t).

The last row

$$[0, 0, 0, 1]$$

is for homogenous coordinate representation.

IV. Undistortion of Images

The images from both cameras are distorted due to lens imperfections. The process of undistortion involves using the distortion coefficients and intrinsic matrix to correct the image.

```
def undistort_image(image, K, dist_coeffs):
    h, w = image.shape[:2]
    new_K, roi = cv2.getOptimalNewCameraMatrix
        (K, dist_coeffs, (w, h), 1, (w, h))
    undistorted_img = cv2.undistort(image, K,
        dist_coeffs, None, new_K)
```

```
    x, y, w, h = roi
    return undistorted_img[y:y+h, x:x+w]
```

- $cv2.getOptimalNewCameraMatrix()$ calculates a new camera matrix for undistorted images.
- new_k is new the camera matrix after considering free scaling.
- roi is the region of interest after undistortion, which is $(x, y, width, height)$
- $cv.undistort()$ applies the distortion correction to the image

V. Image Warping

In order to align the image from Camera 5 to Camera 6, we derive the homography matrix H to represent the transformation between two planes. Since the two cameras view the same planar surface, a homography matrix can relate their views.We use the projection relationship between two camera views where the scene is assumed to be planar enough to ignore depth variations [2]. A point P can be projected onto Camera 5 as point $P-5$ and onto Camera 6 as p_6. We can derive the relationship between points via:

$$P_6 = K_6(RP + t) \tag{6}$$

Using Camera 5's intrinsic matrix:

$$P = K_5^{-1} p_5 \tag{7}$$

Substituting P into the first equation and simplifying:

$$P_6 = K_6(R - tK_5^{-1} p_5 + t) \tag{8}$$

Since we are working with homographies, we ignore the depth component using:

$$H = K_6 \left(R - tK_5^{-1}[2,:]\right) K_5^{-1} \tag{9}$$

- K_5^{-1}: The inverse of the intrinsic matrix of Camera 5, used to convert image coordinates to normalized coordinates.
- K_6: The intrinsic matrix of Camera 6, used to convert normalized coordinates to image coordinates.
- R: The rotation matrix describes how Camera 5 is rotated relative to Camera 6 in 3D space.
- T: The translation vector describes how Camera 5 is translated (shifted) relative to Camera 6 in 3D space.
- $K_5^{-1}[2,:]$: Extracts the third row of K_5's inverse. The third row corresponds to the depth component of the projection matrix. Assuming that the scene is planar, we can ignore depth. This allows us to transform a 3D point in the image plane to 2D homography [3].
- $R - tK_5^{-1}[2,:]$: The subtraction operation operation adjusts the rotation matrix by factoring in the translation vector, effectively aligning the two cameras' perspectives. The term tK_5^{-1} represents the translation effect on the projection plane, considering the plane is distant or approximately flat.

The transformation is applied using the OpenCV function `cv2.warpPerspective()`. The Python code is derived via:

```
def warp_camera_5_to_6(img5):
    h, w = img5.shape[:2]
    R = T_05CAMERA_06CAMERA[:3, :3]
    t = T_05CAMERA_06CAMERA[:3, 3]

    H = K_6 @ (R - (t.reshape(3, 1) @ np.
        linalg.inv(K_5)[2, :].reshape(1, 3)))
        @ np.linalg.inv(K_5)

    img5_warped = cv2.warpPerspective(img5, H,
         (w, h))
    return img5_warped
```

VI. Image Results

The following images illustrate the results of each processing step.

Fig. 1. Raw Image from Camera 5 before distortion.

The image shows the original, distorted view captured by Camera 5. The image has visible radial distortion, particularly near the edges of the frame. Straight lines, such as power lines and road edges appear curved due to lense imperfections.

Fig. 2. Raw Image from Camera 6 before distortion.

Similar to the previous image, this is the original distorted image captured by Camera 6. The image is intended to com-

plement Camera 5's perspective, providing a slightly different viewpoint. As with Camera 5, radial distortion is present and can hinder geometric transformations if not corrected.

Fig. 3. Undistorted Image from Camera 5.

The distortion removal process has successfully corrected the curved lines from the raw image.The power lines and road edges now appear straight, indicating that the distortion correction matrix was applied effectively.

Fig. 4. Undistorted Image from Camera 6.

The undistorted image from Camera 6 shows a similar improvement, with straight lines appearing as intended. Unlike Camera 5, Camera 6's original distortion was less severe, but the correction process was still necessary. The cropping of the valid region (ROI) might be reducing the field of view slightly, but it's a necessary trade-off for image quality.

Fig. 5. Stitched Image (Camera 5 to Camera 6) after warping.

This image demonstrates the warping process applied to align Camera 5's viewpoint with Camera 6's. The stitching shows a clear alignment between the two views, but there is a noticeable seam between the two images. While image was warping was successful in terms of geometry, there are apparent lighting conditions and exposure differences between the images.

VII. Next Sections

We plan to extract high level features from the images using a Transformer architecture and fuse the features together using attention. Transformers can effectively learn robust features across multiple views, allowing for seamless integration without relying on manual stitching and blending techniques.

VIII. Conclusion

This report demonstrates the process of aligning images from different cameras using intrinsic and extrinsic calibration. The described process is fundamental for multi-camera fusion in autonomous vehicles.

References

[1] M. Pitropov, D. E. Garcia, J. Rebello, M. Smart, C. Wang, K. Czarnecki, and S. Waslander, "Canadian adverse driving conditions dataset," *The International Journal of Robotics Research*, vol. 40, no. 4–5, p. 681–690, Dec. 2020. [Online]. Available: http://dx.doi.org/10.1177/0278364920979368

[2] Y. Luo, X. Wang, Y. Liao, Q. Fu, C. Shu, Y. Wu, and Y. He, "A review of homography estimation: advances and challenges," *Electronics*, vol. 12, no. 24, p. 4977, 2023.

[3] A. Agarwal, C. Jawahar, and P. Narayanan, "A survey of planar homography estimation techniques," *Centre for Visual Information Technology, Tech. Rep. IIIT/TR/2005/12*, 2005.